Università degli studi di Cagliari

FACOLTÀ DI INGEGNERIA
Corso di Laurea in Ingegneria Elettronica

SVILUPPO DI UN TOOLBOX MATLAB PER IL CALCOLO CON MATRICI STRUTTURATE MULTI-INDICE

Tesi di Laurea

Relatori:
Prof. Giuseppe Rodriguez
Dott.ssa Luisa Deias

Candidato:
Francesco Arrai

Sessione Unica
Anno Accademico 2011/2012

Dedico questo traguardo ai miei genitori
e a una persona che avrebbe voluto essere qui

Indice

1 Introduzione **5**

1.1 Matrici strutturate 5

1.1.1 Matrici circolanti 6

1.1.2 Matrici di Toeplitz 10

1.1.3 Altre classi di matrici strutturate 12

1.2 Signal processing 14

1.2.1 Acquisizione di immagini 14

1.2.2 Modello matematico 15

1.3 Equazioni differenziali con valori agli estremi 21

1.4 Approssimazione polinomiale ai minimi quadrati in $L^2[a, b]$ . . 22

1.5 Equazioni integrali 24

1.6 Il toolbox per Matlab `smt` 26

1.7 Obiettivo della tesi 27

2 Introduzione alle matrici multiindice **28**

2.1 Autovalori e autovettori 31

2.2 Matrici di forma particolare 32

2.2.1 Matrici Hermitiane. 33

2.2.2 Matrici unitarie. 33

2.3 Metodi diretti per la soluzione di sistemi lineari. 34

2.4 Norme di matrici bi-indice 36

2.5 Matrici Circolanti bi-indice 39

2.5.1 Autovalori di una matrice circolante bi-indice 40

2.6 Matrici di Toeplitz bi-indice 43

3 Il toolbox `smt2` **46**

3.1 Struttura del toolbox 49

3.2 Test numerici 52

3.3 Sviluppi futuri 55

4 Un problema di elettromagnetismo **56**
4.1 Risoluzione numerica 66

Ringraziamenti **69**

Capitolo 1

Introduzione

1.1 Matrici strutturate

In molti problemi si presentano sistemi lineari la cui matrice dei coefficienti è dotata di una particolare struttura [7], cioè dipende da un numero di parametri inferiore ad n^2. Per matrice strutturata o dotata di 'struttura' si intende una matrice che dipende da αn parametri con α indipendente da n. La struttura di una matrice consente quindi un enorme guadagno nell'occupazione di memoria, soprattutto per matrici di grandi dimensioni. Un semplice esempio di matrici strutturate sono le matrici a banda, i cui elementi a_{ij} sono nulli per $i - j > m_1$ e $i - j < m_2$ essendo m_1 e m_2 gli interi che indicano l'ampiezza di banda. Un algoritmo per la risoluzione di un sistema lineare, o la fattorizzazione di una matrice, è veloce se ha una complessità $O(n^p)$, con $p < 3$. Alcuni algoritmi veloci classici non risultano numericamente stabili, in quanto non prevedono l'uso del pivoting. L'applicazione del pivoting risulta spesso impossibile in quanto può distruggere la struttura di una matrice. In taluni casi la struttura di una matrice risulta nascosta, pur essendo comunque presente. Ad esempio, in una matrice a banda le cui righe e colonne abbiano subito una permutazione risulta visibile ad un primo esame solo la sparsità, ma non la struttura.Vediamo anche un'altro esempio non banale. Sappiamo che una matrice di Toeplitz non singolare nxn, con i minori principali diversi da zero, si inverte con un numero di operazioni aritmetiche dell'ordine di $O(n^2)$ che, comparate all'ordine di $O(n^3)$ necessarie per una matrice generica, rappresentano un enorme vantaggio soprattutto per n abbastanza grande. Ma se sappiamo in qualche modo, che una matrice non di Toeplitz è l'inversa di qualche matrice di Toeplitz, allora si mostra che considerando questo aspetto è possibile invertire la matrice non Toeplitz con $O(n^2)$ moltiplicazioni, anche se la struttura non è di fatto presente. Ad

esempio se T è la matrice di Toeplitz

$$T = \begin{bmatrix} 4 & 3 & 2 & 1 \\ 3 & 4 & 3 & 2 \\ 2 & 3 & 4 & 3 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

la sua inversa è

$$T^{-1} = \frac{1}{10} \begin{bmatrix} 6 & -5 & 0 & 1 \\ -5 & 10 & -5 & 0 \\ 0 & -5 & 10 & -5 \\ 1 & 0 & -5 & 6 \end{bmatrix}$$

che non è di Toeplitz. Risulterebbe utile, in casi come questo, un metodo per rilevare la presenza della struttura, e degli algoritmi che traggano vantaggio da essa. Risulterebbe utile, in casi come questo, un metodo per rilevare la presenza della struttura, e degli algoritmi che traggano vantaggio da essa. Annoveriamo tra le matrici strutturate quelle circolanti e di Toeplitz di cui discorreremo ampiamente in seguito estendendo il discorso anche al caso n-dimensionale, e quelle di Cauchy, Vandermonde,Hankel di cui discorreremo qui brevemente.

1.1.1 Matrici circolanti

Una matrice circolante [4] di ordine n

$$C_n = [a_{(j-k)mod(n)}]_{j,k=0}^{n-1} = \begin{bmatrix} a_0 & a_{n-1} & \cdots & a_1 \\ a_1 & a_0 & \ddots & a_2 \\ \vdots & \ddots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & \cdots & a_0 \end{bmatrix} = F_n^* D_n F_n$$

è una matrice per cui valgono le seguenti condizioni $C_{i,j} = C_{i-j}$,$C_{k-n} = C_n$ con $k = 1, \cdots, n-1$. La principale proprietà di una matrice circolante è che la si può diagonalizzare tramite la matrice normalizzata di Fourier definita da $\mathcal{F}_{k,l} = \frac{1}{\sqrt{n}} \omega^{-(k-1)(l-1)}$ con $k, l = 1, \cdots, n$ dove $\omega = e^{\frac{2\pi i}{n}}$ è la radice n-esima dell'unità e i indica l'unità immaginaria. Questo ci consente di fattorizzare una matrice circolante nella forma $C = \mathcal{F}^* \Delta \mathcal{F}$ dove $\Delta = diag(\hat{C(1)}, \hat{C(\omega)}, \cdots, \hat{C(\omega^{n-1})})$ e $\hat{C(\zeta)} = \sum_{k=0}^{n-1} C_k \zeta^{-k}$ è la trasformata discreta di Fourier della prima colonna di C. Dimostriamo questo fatto:

Consideriamo una matrice circolante $C_{(j,l)}$(ricordiamo che per matrice circolante si intende una matrice $C_{j,l}$ i cui elementi sono: $c_{j-l} = c_k$ con $k = j - l$ e $c_{k+n} = c_k$ dove n é la dimensione di j e l , j é l'indice di riga e l

é l'indice di colonna degli elementi di C, e gli indici variano in questo modo: $j, l = 1 : n$ in quanto la matrice é quadrata e una matrice di Fourier $F_{j,l}$ i cui elementi$f_{i,j}$ sono definiti come ω^{-jl} e moltiplichiamo C per la colonna l di F, otteniamo:

$$\left[C \cdot F_{(\cdot,l)}\right] = \sum_{r=0}^{n-1} C_{(j-r)}\omega^{-rl}$$

dove con ω si intende $\mathrm{e}^{\frac{2\pi\mathbf{i}}{n}}$,ponendo $j - r = k$ si ottiene:

$$\sum_{k=j}^{j-(n-1)} C_{(k)}\omega^{-(j-k)l}$$

che possiamo riscrivere in questo modo:

$$\omega^{-jl} \sum_{k=j}^{j-(n-1)} C_{(k)}\omega^{kl}$$

spezziamo la sommatoria in due, una di soli termini positivi e una di soli termini negativi ottenendo:

$$\omega^{-jl}\left(\sum_{k=j-(n-1)}^{-1} C_{(k)}\omega^{kl} + \sum_{k=0}^{j} C_{(k)}\omega^{kl}\right)$$

focalizziamo l'attenzione sulla sommatoria $\sum_{k=j-(n-1)}^{-1} C_{(k)}\omega^{kl}$ questa puó essere riscritta nel modo seguente:

$$\sum_{k=j-1}^{n-1} C_{(k+n)}\omega^{(k+nl)} = \sum_{k=j-1}^{n-1} C_{(k+n)}\omega^{kl}$$

tenuto conto che $\omega^{nl} = 1$, in quanto sostituendo a ω il suo valore $\mathrm{e}^{\frac{2\pi\mathbf{i}}{n}}$ otteniamo: $\mathrm{e}^{\frac{2\pi n\mathbf{i}}{n}} = \mathrm{e}^{2\pi\mathbf{i}}$, e che $C_{(k+n)} = C_{(k)}$ (per la definizione stessa di matrice circolante) si ottiene:

$$\sum_{k=j-(n-1)}^{-1} C_{(k)}\omega^{kl} = \sum_{k=j+1}^{n-1} C_{(k)}\omega^{kl}$$

e di conseguenza:

$$\sum_{k=j_-(n-1)}^{-1} C_{(k)}\omega^{kl} + \sum_{k=0}^{j} C_{(k)}\omega^{kl} = \sum_{k=0}^{n-1} C_{(k)}\omega^{kl}$$

andando a sostituire otteniamo:

$$\omega^{-jl} \sum_{k=0}^{n-1} C_{(k)} \omega^{kl}$$

dove ω^{-jl} è la colonna di indici l di F e cioè l'autovettore di indice l della matrice C, mentre $\sum_{k=0}^{n-1} C_{(k)} \omega^{kl}$ è l'autovalore corrispondente.
L'occupazione di memoria per una matrice circolante è $O(n)$ perchè basta memorizzare la prima colonna mentre la complessità è $O(nlog(n))$ Citiamo ora alcune semplici operazioni che mettono in risalto le proprietà strutturali delle matrici circolanti che ben vengono sfruttate dal pacchetto `smt` (che verrà illustrato nella sezione 1.7). Partiamo dalle operazioni di somma e sottrazione, per sommare/sottrarre due matrici circolanti, ovviamente con le stesse dimensioni è sufficiente sommare o sottrarre le prime colonne e poi ricostruire la matrice. Per quanto riguarda la moltiplicazione e la divisione se questa viene effettuata tra matrici circolanti, si utilizza la FFT come illustremo a breve effettuando n operazioni anzichè n^2 vediamo qualche esempio: consideriamo due matrici circolanti [4x4] e consideriamo la possibili operazioni

$$C_1 = \begin{bmatrix} 1 & 4 & 3 & 2 \\ 2 & 1 & 4 & 3 \\ 3 & 2 & 1 & 4 \\ 4 & 3 & 2 & 1 \end{bmatrix}$$

$$C_2 = \begin{bmatrix} 5 & 9 & 4 & 3 \\ 3 & 5 & 9 & 4 \\ 4 & 3 & 5 & 9 \\ 9 & 4 & 3 & 5 \end{bmatrix}$$

$$C_1 + C_2 = \begin{bmatrix} 6 & 13 & 7 & 5 \\ 5 & 6 & 13 & 7 \\ 7 & 5 & 6 & 13 \\ 13 & 7 & 5 & 6 \end{bmatrix}$$

$$C_2 - C_1 = \begin{bmatrix} 4 & 5 & 1 & 1 \\ 1 & 4 & 5 & 1 \\ 1 & 1 & 4 & 5 \\ 5 & 1 & 1 & 4 \end{bmatrix}$$

Come si nota è stato sufficiente sommare o sottrarre gli elementi della prima colonna, la matrice risultante dall'operazione mantiene la struttura e gli

autovalori sono la somma o la differenza degli autovalori delle matrici di partenza.

$$C_1 * C_2 = \begin{bmatrix} 5 & 36 & 12 & 6 \\ 6 & 5 & 36 & 12 \\ 12 & 6 & 5 & 36 \\ 36 & 12 & 6 & 5 \end{bmatrix}$$

$$C_1 * C_2^{-1} = \begin{bmatrix} \frac{1}{5} & \frac{4}{9} & \frac{3}{4} & \frac{2}{3} \\ \frac{2}{3} & \frac{1}{5} & \frac{4}{9} & \frac{3}{4} \\ \frac{3}{4} & \frac{2}{3} & \frac{1}{5} & \frac{4}{9} \\ \frac{4}{9} & \frac{3}{4} & \frac{2}{3} & \frac{1}{5} \end{bmatrix}$$

È importante notare che l'inversa di una matrice circolante risulta circolante infatti si ha:

$$C = F_n^* D F_n$$
$$CF^* = F^* D$$

e quindi

$$C^{-1} = F_n^* D^{-1} F_n$$

e in particolare

$$C_2^{-1} = \begin{bmatrix} -0.0579 & 0.0142 & -0.0849 & 0.1763 \\ 0.1763 & -0.0579 & 0.0142 & -0.0849 \\ -0.0849 & 0.1763 & -0.0579 & 0.0142 \\ 0.0142 & -0.0849 & 0.1763 & -0.0549 \end{bmatrix}$$

La moltiplicazione tra matrici circolanti sfrutta il fatto che gli autovettori di matrici di dimensioni uguali essendo la matrici di Fourier sono uguali quindi:

$$C_1 * C_2 = F_n^* D_1 F_n * F_n^* D_2 F_n$$

poichè $F_n^* F_n = I_n$, ovvero la matrice identità di ordine n si ottiene:

$$C_1 * C_2 = F_n^* D_1 * D_2 F_n$$

dove $D_1 * D_2$ è il prodotto di due matrici diagonali contenenti gli autovalori, ed essendo questi ultimi la trasformata di Fourier della prima colonna degli elementi della matrice, il prodotto si riduce al calcolo di 3 FFT e un prodotto punto a punto tra due vettori.
Una matrice αcircolante $C_{n,\alpha}$ è definita in questo modo:

$$C_{n,\alpha} = [a_{(j-\alpha k) mod(n)}]_{j,k=0}^{n-1} = \begin{bmatrix} a_0 & a_{(-\alpha)} & \cdots & a_{(-(n-1)\alpha) mod(n)} \\ a_1 & a_{(1-\alpha)} & \cdots & a_{(1-(n-1)\alpha) mod(n)} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n-1} & a_{(n-\alpha) mod(n)} & \cdots & a_{(n-1-(n-1)\alpha) mod(n)} \end{bmatrix}$$

con $0 \lneq \alpha \lneq n$ e può essere scritta come $C_{n,\alpha} = C_n Z_{n,\alpha} = F_n D_n F_n^* Z_{n,\alpha}$, con $Z_{n,\alpha} = [\delta_{r-\alpha c}]_{r,c=0}^{n-1}$ e $\delta_k = \{_0^1$ Il valore di δ_k dipende dal resto della divisione per$n$. se $n = 5$ e $\alpha = 3$ abbiamo

$$C_{n,\alpha} = \begin{bmatrix} a_0 & a_2 & a_4 & a_1 & a_3 \\ a_1 & a_3 & a_0 & a_2 & a_4 \\ a_2 & a_4 & a_1 & a_3 & a_0 \\ a_3 & a_0 & a_2 & a_4 & a_1 \\ a_4 & a_1 & a_3 & a_0 & a_2 \end{bmatrix}$$

1.1.2 Matrici di Toeplitz

Definizione 1.1 *Una matrice $T \in \mathbb{R}^{nxn}$ si dice Toeplitz [5] se esistono $2n-1$ scalari $t_{-n+1}, \cdots, t_0, \cdots, t_{n-1}$, tali che $T = (t_{ij})_{i,j=1,\cdots,n}$ con $t_{ij} = t_{i-j}$*

$$T = \begin{bmatrix} t_0 & t_{-1} & t_{-2} & \cdots & t_{-n+1} \\ t_1 & t_0 & t_{-1} & \cdots & t_{-n+2} \\ t_2 & t_1 & t_0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & t_{-1} \\ t_{n-1} & t_{n-2} & \cdots & t_1 & t_0 \end{bmatrix}$$

Se la matrice T è simmetrica, allora dipende da n scalari $t_0, \cdots, t_{n-1}$ tali che $t_{ij} = t_{|i-j|}$

$$T = \begin{bmatrix} t_0 & t_1 & t_2 & \cdots & t_{n-1} \\ t_1 & t_0 & t_1 & \cdots & t_{n-2} \\ t_2 & t_1 & t_0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & t_1 \\ t_{n-1} & t_{n-2} & \cdots & t_1 & t_0 \end{bmatrix}$$

Una matrice αToeplitz $T_{n,\alpha}$ è definita in questo modo :

$$T_{n,\alpha} = [b_{j-\alpha k}]_{j,k=0}^{n-1} = \begin{bmatrix} b_0 & b_{-\alpha} & \cdots & b_{-(n-1)\alpha} \\ b_1 & b_{1-\alpha} & \ddots & b_{1-(n-1)\alpha} \\ \vdots & \ddots & \ddots & \vdots \\ b_{n-1} & b_{n-1-\alpha} & \cdots & b_{n-1-(n-1)\alpha} \end{bmatrix}$$

se $n = 5$ e $\alpha = 3$ abbiamo

$$T_{n,\alpha} = \begin{bmatrix} b_0 & b_{-3} & b_{-6} & b_{-9} & b_{-12} \\ b_1 & b_{-2} & b_{-5} & b_{-8} & b_{-11} \\ b_2 & b_{-1} & b_{-4} & b_{-7} & b_{-10} \\ b_3 & b_0 & b_{-3} & b_{-6} & b_{-9} \\ b_4 & b_1 & b_{-2} & b_{-5} & b_{-8} \end{bmatrix}$$

Le matrici di Toeplitz sono un importante esempio di matrici strutturare. Va evidenziato subito, come per queste matrici sia limitata l'occupazione di memoria. Infatti, se per una matrice qualsiasi di ordine n le celle di memoria da occupare sono n^2, per una Toeplitz esse sono appunto $2n-1$ e n nel caso di una matrice di Toeplitz simmetrica. Queste semplici osservazioni, unite alla presenza di queste matrici in numerosi problemi applicativi, giustificano lo studio sistematico di algoritmi per matrici di Toeplitz. Una matrice di Toeplitz di ordine n è una matrice T i cui elementi soddisfano la relazione $T_{i,j} = T_{i-j}$ con $i, j = 1, \cdots, n$ cioè l'elemento $T_{i,j}$ dipende solo dalla differenza dei due indici. Ad esempio per n = 4 si ha :

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

Questa matrice è cioè a diagonali costanti e per esempio in questo specifico campo ci sono 7 elementi da memorizzare e cioè 2n-1. Un caso interessante è quello di matrice Toeplitz Hermitiana per cui vale $T_{i,j} = T^*_{i,j}$ cioè $T_{i-j} = T^*_{j-i}$, nel caso reale la matrice è simmetrica e in tal caso la definizione prevede $T_{i,j} = T_{|i-j|}$ e la matrice risulta la seguente :

$$T = \begin{bmatrix} t_0 & t_1 & t_2 & t_3 \\ t_1 & t_0 & t_1 & t_2 \\ t_2 & t_1 & t_0 & t_1 \\ t_3 & t_2 & t_1 & t_0 \end{bmatrix}$$

Gli elementi da memorizzare in questo caso sono 4 e cioè n. Oltre alla minore occupazione di memoria anche per le matrici di Toeplitz valgono alcune semplificazioni per alcune operazioni, in particolare per somma e sottrazione valgono le stesse regole che abbiamo gia illustrato per le circolanti, tenuto conto che per le Toeplitz hermitiane basta agire sulla prima colonna mentre per le altre sulla prima colonna e sulla prima riga. Per quanto riguarda il prodotto,si può ancora velocizzarlo trasformando le matrici di Toeplitz in matrici circolanti con procedura che vedremo in seguito.

1.1.3 Altre classi di matrici strutturate

Una matrice $C \in \mathbb{C}^{nxn}$ si dice di Cauchy [1] se esistono dei vettori $t, s \in \mathbb{C}^n$ tali che $C = (c_{ij})_{i,j=1,\cdots,n}$ con $c_{ij} = \dfrac{1}{t_i - s_j}$ e quindi

$$C = \begin{bmatrix} \dfrac{1}{t_1 - s_1} & \dfrac{1}{t_1 - s_2} & \cdots & \dfrac{1}{t_1 - s_n} \\ \dfrac{1}{t_2 - s_1} & \dfrac{1}{t_2 - s_2} & \cdots & \dfrac{1}{t_2 - s_n} \\ \vdots & \vdots & & \vdots \\ \dfrac{1}{t_n - s_1} & \dfrac{1}{t_n - s_2} & \cdots & \dfrac{1}{t_n - s_n} \end{bmatrix}$$

Questa definizione classica è stata estesa per comprendere una classe più ampia di matrici. Una matrice $C \in \mathbb{C}^{nxn}$ si dice Cauchy-like se esistono dei vettori $t, s \in \mathbb{C}^n$ e dei vettori $\phi_i, \psi_i \in \mathbb{C}^\alpha$, $i = 1, \cdots, n$, dove α è un numero generalmente piccolo rispetto ad n, tali che $C = (c_{ij})_{i,j=1,\cdots,n}$ con $c_{ij} = \dfrac{\phi_i \psi_j}{t_i - s_j}$ e quindi

$$C = \begin{bmatrix} \dfrac{\phi_1\psi_1}{t_1 - s_1} & \dfrac{\phi_1\psi_2}{t_1 - s_2} & \cdots & \dfrac{\phi_1\psi_n}{t_1 - s_n} \\ \dfrac{\phi_2\psi_1}{t_2 - s_1} & \dfrac{\phi_2\psi_2}{t_2 - s_2} & \cdots & \dfrac{\phi_2\psi_n}{t_2 - s_n} \\ \vdots & \vdots & & \vdots \\ \dfrac{\phi_n\psi_1}{t_n - s_1} & \dfrac{\phi_n\psi_2}{t_n - s_2} & \cdots & \dfrac{\phi_n\psi_n}{t_n - s_n} \end{bmatrix}$$

È immediato osservare che la prima definizione è ricavabile dalla seconda ponendo $\alpha = 1$ e $\phi_i = \psi_i = 1$, $i = 1, \cdots, n$ Le matrici Cauchy-like non hanno una grande rilevanza applicativa, per questa classe di matrici è possibile utilizzare un algoritmo di fattorizzazione veloce e numericamente stabile, per la possibilità di implementare il pivoting di colonna senza distruggere la struttura della matrice.

Una matrice $V \in \mathbb{R}^{(n+1)x(n+1)} = (v_{ij})_{i,j=0,\cdots,n}$ si dice di Vandermonde se $v_{ij} = v_i^j$ e si presenta nella seuente forma:

$$V = \begin{bmatrix} 1 & v_0 & \cdots & v_0^n \\ 1 & v_1 & \cdots & v_1^n \\ 1 & v_2 & \cdots & v_2^n \\ \vdots & \vdots & & \vdots \\ 1 & v_n & \cdots & v_n^n \end{bmatrix}$$

Come si vedrà per le Toeplitz, anche per queste matrici valgono le considerazioni sui vantaggi di lavorare su uno spazio di memoria molto minore di quello richiesto per matrici generiche. I sistemi e quindi le matrici di Vandermonde sorgono spesso in problemi di interpolazione e approssimazione. Infatti risolvere il sistema $Va = f$ è equivalente a calcolare un polinomio interpolante. Fissati i valori (x_i, f_i) per $i = 0, \cdots, n$, il problema dell'interpolazione polinomiale consiste nel trovare il polinomio $p \in \prod_n$ tale che $p(x_i = f_i$ per $i = 0, \cdots, n$. Tale polinomio esiste ed è unico se le ascisse di interpolazione sono distinte. Esprimendo p nella base canonica

$$p(x) = \sum_{j=0}^{n} a_j x^j,$$

le condizioni di interpolazione

$$\Sigma_{j=0}^{n} a_j x^j = f_i$$

con

$$i = 1, \cdots, n,$$

possono essere riscritte in forma matriciale

$$\begin{bmatrix} 1 & x_0 & \cdots & x_0^n \\ 1 & x_1 & \cdots & x_1^n \\ 1 & x_2 & \cdots & x_2^n \\ \vdots & \vdots & & \vdots \\ 1 & x_n & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ \vdots \\ f_n \end{bmatrix}$$

La matrice $V = (x^j)_{i,j=0,\cdots,n}$ è di Vandermonde ed è non singolare se e solo se $x_i \neq x_j$ e per $i \neq j$. Un algoritmo veloce per la risoluzione di questo sistema può essere ottenuto esprimendo il polinomio interpolante nalla forma di Newton

$$p(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots + c_n(x - x_0)(x - x_1)\cdots(x - x_n)$$

e cioè

$$p(x) = c_0 + \sum_{k=1}^{n} c_k(\prod_{i=0}^{k-1}(x - x_i))$$

dove i coefficienti c_k del polinomio coincidono con le differenze divise $c_k = f[x_0, x_1, \cdots, x_n]$. I coefficienti c_k possono quindi essere calcolati applicando la definizione ricorsiva delle differenze divise

$$f[x_i] = f_i i = 0, 1, \cdots, n f[x_i, x_{i+1}, \cdots, x_{i+k}] = \frac{f[x_{i+1}, \cdots, x_{i+k}] - f[x_i, \cdots, x_{i+k-1}]}{x_{i+k} - x_i}$$

Si definiscono i polinomi come

$$p_n(x) = c_n$$

$$p_k = c_k + (x - x_k)p_{k+1}(x), k = n-1, n-2, \cdots, 0$$

e si osserva che $p_0(x) = p(x)$.

Una matrice $H \in \mathbb{R}^{nxn}$ si dice di Hankel se esistono $2n-1$ scalari $h_0, \cdots, h_{2n-2}$ con $h_{ij} = h_{i+j-2}$ e quindi

$$H = \begin{bmatrix} h_0 & h_1 & \cdots & h_{n-2} & h_{n-1} \\ h_1 & h_2 & \cdots & h_{n-1} & h_n \\ \vdots & \vdots & & \vdots & \vdots \\ h_{n-2} & h_{n-1} & \cdots & h_{2n-4} & h_{2n-3} \\ h_{n-1} & h_n & \cdots & h_{2n-3} & h_{2n-2} \end{bmatrix}$$

Spesso le matrici di Hankel si trovano inquadrate nella classe più generale delle matrici Toeplitz plus Hankel. Una matrice A si definisce Toeplitz plus Hankel o più brevemente $T+H$, se è esprimibile nella forma $A = T+H$, dove T è una matrice di Toeplitz e H è una matrice di Hankel. Le matrici T+H sono estremamente diffuse nelle applicazioni, come ad esempio nei processi di ortogonalizzazione, nel campo delle equazioni differenziali a derivate parziali, nel signal processing, etc. Per questo motivo c' è attualmente un grosso interesse per gli algoritmi destinati al trattamento di queste matrici.

1.2 Signal processing

Faremo una breve introduzione su alcuni problemi riguardanti il trattamento dei segnali, in particolare di immagini,iniziando dal caso monodimensionale.

1.2.1 Acquisizione di immagini

Un'immagine digitale viene acquisita con strumenti specifici per il tipo di dato che si sta trattando: ad esempio un telescopio per un'immagine di una galassia, un microscopio elettronico per l'immagine di un battierio. Indipendentemente dal tipo di immagine che si sta studiando, uno strumento per l'acquisizione è comunemente composto da due parti:

la prima parte è in grado di captare le radiazioni emesse dall'oggetto di cui si vuole avere l'immagine. Ad esempio, nel caso di un telescopio il dispositivo in questione è il C.C.D. (Charge Copuled Device, dispositivo a scorrimento di carica): e composto da materiali semiconduttori, che convertono la luce incidente in cariche elettriche;

la seconda parte è costituta dal dispositivo che trasforma il segnale fisico in un segnale digitale, cioè in dati trattabili dagli elaboratori; ad esempio, nel caso del telescopio sopracitato, è costituito dal dispositivo che tramuta le cariche elettriche in dati numerici gestibile da computer.

L'acquisizione di queste immagini non è esente da errori: sussistono alcuni effetti, dovuti a fattori fisici, che non permettono di ottenere un'immagine 'pura' dell'oggetto che si sta esaminando. La degradazione dell'immagine è dovuta ai seguenti fattori: lo sfocamento (blurring) dovuto alla struttura dello strumento con cui si sta lavorando e causato, ad esempio, da diffrazioni, aberrazioni o altri fenomeni fisici analoghi; un'altra causa di degradazione sta nel fatto che si effettua un passaggio da uno spazio continuo (la realtà) ad uno spazio discreto (la memoria del computer), cioè si sta effettuando una transizione tra un'immagine analogica ad una digitale tramite un processo di discretizzazione. Questa discretizzazione dipende ancora una volta dalla struttura dello strumento, nella cui costruzione normalmente si tiene conto di teoremi di campionamento, e i dati sono (nella maggior parte dei casi) sovracampionati. Inoltre, si introducono degli errori statistici dovuti al processo di digitalizzazione; i dati raccolti, quindi, sono in realtà realizzazioni di variabili aleatorie. Il modello generale della formazione di un'immagine digitale è dato da

$$g = k(f) + \eta$$

dove f è l'immagine reale dell'oggetto che si sta considerando, K è un operatore che modifica l'immagine originale, è l'errore di tipo statistico descritto prima e g è l'immagine ottenuta alla fine del processo.

1.2.2 Modello matematico

Un'immagine di un'oggetto viene considerata come una funzione$f : \Omega \to \mathbb{R}$ dove $\Omega \subseteq \mathbb{R}^2$, in cui (x,y) sono le variabili spaziali e dove $f(x_0, y_0)$ determina l'intensità dell'oggetto alle coordinate $(x_0; y_0)$.

Blurring

Il processo di formazione dell'immagine è definito tramite l'azione di un operatore $\tilde{K}$, supposto lineare e continuo. Il modello continuo è quindi definito dalla formula:

$$g(x, y) = \int\int_{(s,t)\in\Omega} \tilde{K}(x, s, y, t) f(s, t) ds dt$$

che ha la forma di un'equazione integrale di Fredholm di prima specie. L'operatore $\tilde{K}$ prende il nome di risposta in impulso del sistema: considerando

una sorgente puntiforme di intensità unitaria centrata nel punto (x_0, y_0), modellata sa una δ di Dirac centrata in quel punto, allora l'immagine restituita dal sistema è

$$g(x,y) = \int\int_{(s,t)\in\Omega} \tilde{K}(x,s,y,t)\delta(s-x_0, t-y_0)dsdt = \tilde{K}(x,x_0,y,y_0)$$

cioè l'immagine di una sorgente puntiforme di intensità 1 è proprio la risposta in impulso del sistema. In questa caso, la risposta in impulso di sistema viene detta anche Point Spread Function (PSF), ovverosia funzione di allargamento del punto. L'azione della PSF sul punto (e quindi su tutta l'immagine) viene detto blurring (sfoca- mento), in quanto rende meno nitidi i dettagli. Nel caso in cui la formazione dell'immagine di una sorgente puntiforme (e quindi anche di un'immagine più complessa) non dipenda dalla posizione nello spazio del punto stesso, si dice che il sistema è spazio invariante e l'operatore $\tilde{K}$ prende la forma

$$\tilde{K}(x,s,y,t) = K(x-s, y-t)$$

Le ipotesi sull'operatore K sono:

$$K(x,y) \geq 0$$

con $x, y \in \mathbb{R}$ e

$$\int_{\mathbb{R}} K(x,y)dxdy = 1$$

Sostituendo si ottiene

$$g(x,y) = \int\int K(x-s, y-t)f(s,t)dsdt$$

cioè l'immagine g ottenuta alla fine del processo di acquisizione è il prodotto di convoluzione $K * f$. In generale, la PSF è conosciuta tramite stime oppure tramite una funzione vera e propria. Le stime si possono ottenere grazie ad esperimenti: considerando una sorgente puntiforme come descritto prima, si acquisiscono varie immagini di questa sorgente e in questo modo, per ciò che si è osservato, si può avere una buona stima della risposta in impulso del sistema. Nel caso migliore, invece, si dispone della funzione matematica che descrive il comportamento della PSF, funzione che viene fornita dai costruttori del sistema di acquisizione di immagini.

Modello Matematico Discreto

Per poter affrontare il problema al calcolatore, è necessario discretizzare il modello. Un'immagine non sarà più una funzione continua $f : \Omega \to \mathbb{R}$,

ma sarà una matrice $f \in M_{m\times n}(\mathbb{R})$, dove $f(i,j)$ sarà l'elemento di riga$i$ e colonna j ed indicherà come nel caso precedente l'intensità dell'immagine. Si lavorerà con livelli di intensità nella scala di grigio, quindi i valori che f può assumere sono reali nell'intervallo $[0,1]$. Considerando il fatto che l'operatore K può essere visto come l'immagine acquisita di una sorgente puntiforme, allora anch'esso si può considerare come una matrice. Supponendo come prima di lavorare con operatori lineari e continui, allora l'acquisizione dell'immagine può essere scritta sotto la forma $g = Af$ dove $f = vec(f)$, cioè la matrice f viene riscritta sotto forma di vettore, ponendo le colonne in ordine lessicografico una sotto l'altra. Le dimensioni della matrice A, costruita a partire da K, saranno $[mn \times mn]$, dove $m \times n$ sono le dimensioni di f e g. Il modello discreto di acquisizione di un'immagine prende quindi la forma di un sistema lineare. Assumendo ancora di lavorare con operatori spazio invarianti, si avrà nuovamente un prodotto di convoluzione, che prenderà la forma $g(i,j) = \Sigma_{i'=1}^{m}\Sigma_{ji'=1}^{n} f(i^{'}, j^{'})P(i^{'} - i, j^{'} - j)$ dove P è la matrice ottenuta mediante lo shift e il prolungamento periodico dell'immagine K della PSF. Effettuare lo shift(rispetto al centro) di una matrice significa invertire le righe e le colonne della matrice rispetto al centro della matrice stessa. Ad esempio, se la matrice P è scritta

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}$$

allora il suo shift rispetto al centro è

$$\begin{bmatrix} p_{33} & p_{32} & p_{31} \\ p_{23} & p_{22} & p_{21} \\ p_{13} & p_{12} & p_{11} \end{bmatrix}$$

Per capire la funzione del prolungamento periodico enunciato prima, si osservi il prodotto di convoluzione: effettuare questo prodotto significa che ogni elemento della matrice g è determinato dalla somma pesata di tutti gli elementi dell'immagine f, dove i pesi sono gli elementi della P. Effettuare una convoluzione, significa far 'scorrere' la matrice P su ogni elemento dell'immagine f, moltiplicare gli elementi corrispondenti e sommarli. Ad esempio, se la matrice P è quella ottenuta nel precedente esempio, per determinare il pixel g_{22} si effettua il passaggio della P in questo modo:

$$\begin{bmatrix} p_{33} \cdot f_{11} & p_{32} \cdot f_{12} & p_{31} \cdot f_{13} \\ p_{23} \cdot f_{21} & p_{22} \cdot f_{22} & p_{21} \cdot f_{23} \\ p_{13} \cdot f_{31} & p_{12} \cdot f_{32} & p_{11} \cdot f_{33} \end{bmatrix}$$

si sommano tutti gli elementi e si ottiene

$$g_{22} = f_{11}p_{33} + f_{12}p_{32} + f_{13}p_{31} + f_{21}p_{23} + f_{22}p_{22} + f_{23}p_{21} + f_{31}p_{13} + f_{32}p_{12} + f_{33}p_{11}$$

Ma effettuando lo stesso passaggio su un elemento del 'bordo' dell'immagine , ad esempio il pixel(picture element) g_{21}, si nota il sorgere di un problema:

$$\begin{bmatrix} p_{33}\cdot? & p_{32} \cdot f_{12} & p_{31} \cdot f_{13} \\ p_{23}\cdot? & p_{22} \cdot f_{22} & p_{21} \cdot f_{23} \\ p_{13}\cdot? & p_{12} \cdot f_{32} & p_{11} \cdot f_{33} \end{bmatrix}$$

non si capisce cioè per cosa vadano moltiplicati gli elementi della prima colonna di$P$. Per fare ciè, si devono assegnare le condizioni al contorno, cioè decidere il comportamento dell'immagine al di fuori dei confini che si stanno considerando, in quanto in realtà si sta considerando solo una porzione finita di spazio. Il prolungamento periodico dipende dalle condizioni al contorno che possono essere di 3 tipi:
1) Zero Bondary conditions: si suppone l'immagine sia immersa in uno schermo totalmente nero. Si effettua il così detto zero-padding, si aggiungono cioè zeri all'immagine considerata. Ad esempio se F è scritta per blocchi nel seguente modo:

$$\begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix}$$

allora il suo zero-padding è

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & F_{11} & F_{12} & 0 \\ 0 & F_{21} & F_{22} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

dove 0 rappresenta una matrice di dimensioni opportune.
2) Periodic Bondary conditions:si suppone che l'immagine si ripeta identicamente in tutte le direzioni, infinitamente. Per ottenere ciò, si circonda l'immagine con tante copie di se stessa. Ad esempio, se si ha un'immagine rappresentata dalla matrice X, per utilizzare queste condizioni al contorno si considererà l'immagine rappresentata dalla matrice seguente:

$$\begin{bmatrix} X & X & X \\ X & X & X \\ X & X & X \end{bmatrix}$$

3) Reflective Bondary conditions: in questo caso, si suppone che l'immagine

sia circondata da riflessioni di se stessa. Ad esempio, se l'immagine F è rappresentata dalla matrice

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

allora imponendo queste condizioni si ottiene:

$$\begin{bmatrix} 9 & 8 & 7 & 7 & 8 & 9 & 9 & 8 & 7 \\ 6 & 5 & 4 & 4 & 5 & 6 & 6 & 5 & 4 \\ 3 & 2 & 1 & 1 & 2 & 3 & 3 & 2 & 1 \\ 3 & 2 & 1 & 1 & 2 & 3 & 3 & 2 & 1 \\ 6 & 5 & 4 & 4 & 5 & 6 & 6 & 5 & 4 \\ 9 & 8 & 7 & 7 & 8 & 9 & 9 & 8 & 7 \\ 9 & 8 & 7 & 7 & 8 & 9 & 9 & 8 & 7 \\ 6 & 5 & 4 & 4 & 5 & 6 & 6 & 5 & 4 \\ 3 & 2 & 1 & 1 & 2 & 3 & 3 & 2 & 1 \end{bmatrix}$$

La scelta delle condizioni al contorno è dettata dal tipo di problema e dal tipo di immagine che si sta considerando: ad esempio lo zero padding si utilizza nel caso di immagini astronomiche, in quanto gli oggetti in considerazione sono immersi nello spazio profondo, cioè in un background totalmente (o quasi) nero. Una volta efettuata questa scelta, si può effettuare il prodotto di convoluzione: tornando all'esempio precedente, imponendo le condizioni al contorno degli zeri e quindi effettuando lo zero padding di F, per ottenere il pixel g_{21} si avrà

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ p_{33} \cdot 0 & p_{32} \cdot f_{12} & p_{31} \cdot f_{13} & f_{13} & 0 \\ p_{23} \cdot 0 & p_{23} \cdot f_{21} & p_{22} \cdot f_{22} & f_{23} & 0 \\ p_{13} \cdot 0 & p_{13} \cdot f_{31} & p_{12} \cdot f_{32} & f_{33} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

cioè $g_{21} = f_{12}p_{32} + f_{13}p_{31} + f_{21}p_{23} + f_{22}p_{22} + f_{31}p_{13} + f_{32}p_{12}$ Effettuando lo stesso calcolo per tutti gli altri elementi g_{ij}, ed utilizzado la notazione $g = vec(g)$,

si ottiene

$$\begin{bmatrix} g_{11} \\ g_{21} \\ g_{31} \\ g_{12} \\ g_{22} \\ g_{32} \\ g_{13} \\ g_{23} \\ g_{33} \end{bmatrix} = \begin{bmatrix} p_{22} & p_{12} & 0 & p_{21} & p_{11} & 0 & 0 & 0 & 0 \\ p_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} & 0 & 0 & 0 \\ 0 & p_{23} & p_{22} & 0 & p_{31} & p_{21} & 0 & 0 & 0 \\ p_{23} & p_{13} & 0 & p_{22} & p_{12} & 0 & p_{21} & p_{11} & 0 \\ p_{33} & p_{23} & p_{13} & p_{32} & p_{22} & p_{12} & p_{31} & p_{21} & p_{11} \\ 0 & p_{33} & p_{23} & 0 & p_{32} & p_{22} & 0 & p_{31} & p_{21} \\ 0 & 0 & 0 & p_{23} & p_{13} & 0 & p_{22} & p_{12} & 0 \\ 0 & 0 & 0 & p_{33} & p_{23} & p_{13} & p_{32} & p_{2} & 0p_{12} \\ 0 & 0 & 0 & 0 & p_{33} & p_{23} & 0 & p_{32} & p_{22} \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix}$$

cioè il sistema iniziale $g = Af$ Conoscendo le condizioni al bordo e il tipo di PSF si può quindi costruire la matrice A, che può avere particolari strutture, dettate appunto dalle condizioni al bordo. Le strutture possibili sono:

1) Matrice di Toeplitz a blocchi di Toeplitz (BTTB): una matrice si dice di Toeplitz se i suoi elementi sono costanti su ogni diagonale; una matrice a blocchi si dice di Toeplitz se i blocchi sono costanti su ogni diagonale. Una matrice si dice di Toeplitz a blocchi di Toeplitz se è una matrice a blocchi di Toeplitz e ogni blocco è di Toeplitz (come nell'esempio precedente).
2) Matrice circolante a blocchi circolanti (BCCB): una matrice si dice circolante se ogni colonna (e ogni riga) è uno shift periodico della precedente colonna (riga); si dice circolante a blocchi se ogni riga (colonna) formata dai blocchiè uno shift periodico della precedente riga (colonna). Una matrice si dice quindi circolante a blocchi circolanti se è circolante a blocchi e ogni blocco è circolante.
3) Matrice di Hankel: una matrice si dice di Hankel se i suoi elementi sono costanti su ogni antidiagonale. Si può avere quindi una matrice a blocchi di Toeplitz, con blocchi di Hankel (BTHB), una matrice a blocchi di Hankel con blocchi di Toeplitz (BHTB) oppure una matrice a blocchi di Hankel con blocchi di Hankel (BHHB). Nel caso condizioni al contorno nulle, A è una BTTB (visto nell'esempio precedente), mentre nel caso periodico A è una BCCB; infine, nel caso riflessivo la matrice A è la somma di quattro matrici BTTB, BTHB, BHTB e BHHB. Il modello discreto per la formazione di un'immagine digitale può essere quindi visto come la convoluzione tra la matrice f e l'operatore P, oppure come il prodotto matrice-vettore Af; in entrambi i casi, possono sorgere dei problemi: il costo computazionale della convoluzione discreta per matrici di grandi dimensioni, infatti, è molto alto, mentre il prodotto Af può coinvolgere matrici di dimensioni troppo elevate. Una soluzione a questi problemi può essere l'utilizzo della trasformata di Fourier e del teorema di convoluzione che dice che la trasformata della convoluzione di due funzioni è uguale al prodotto delle trasformate delle singole

funzioni, questo consente di trasformare una convoluzione in un prodotto e 3 FFT velocizzando i calcoli sopratutto se si utilizzano immagini che hanno per dimensioni potenze di 2.

Image deblurring

Avendo modellizzato la formazione dell'immagine con l'equazione $g = K(f) + \eta$ intesa sia come sistema lineare che come prodotto di convoluzione, lo scopo è riottenere l'immagine f. Il problema però è mal condizionato: l'aggiunta del rumore η produce grandi variazioni nel calcolo della soluzione f, ed assieme al blurring, inoltre, non permette di raggiungere il risultato esatto. Sono quindi necessari metodi di regolarizzazione che tengano sotto controllo l'influenza del rumore sulla soluzione esatta.

1.3 Equazioni differenziali con valori agli estremi

Consideriamo la seguente equazione differenziale del secondo ordine:

$$y^{''} + P(x)y^{'} + q(x) = r(x)$$

con $\begin{cases} y(a)=\alpha \\ y(b)=\beta \end{cases}$ e $a \leq x \leq b$ si suppone che il sistema abbia una sola soluzione. Utilizziamo un metodo alle differenze finite e approssimiamo sia la derivata prima che la derivata seconda con errori di discretizzazione $O(h^2)$ dove h è il passo di discretizzazione e $x_{i+1} = x_i + h$ nel seguente modo:

$$y^{''} = \frac{y(x_i + h) - 2y(x_i) + y(x_i - h)}{h^2}$$

$$y^{'} = \frac{y(x + h) - y(x - h)}{2h}$$

sostituendo questi valori nell'equazione di partenza e tenuto conto che per facilità di scrittura abbiamo effettuato la sostituzione $y_i = y(x_i)$ otteniamo:

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + P_i \frac{y_{i+1} - y_{i-1}}{2h} + q_i y_i = r_i$$

ricordando che:

$$\begin{cases} x_i = a + ihi = 0,1,\cdots,n+1 \\ h = \frac{b-a}{n+1} \end{cases}$$

abbiamo in tal modo n punti interni e moltiplicando ambo i membri per h^2 otteniamo:

$$y_{i+1} - 2y_i + y_{i-1} + P_i \frac{h}{2} (y_{i+1} - y_{i-1}) + h^2 q_i y_i = h^2 r_i$$

e riordinando i termini e tenuto conto che

$$b_i = 1 - \frac{h}{2}P_i$$

$$c_i = 1 + \frac{h}{2}P_i$$

$$a_i = 1 - h^2 q_i$$

otteniamo:

$$-b_i y_{i-1} + a_i y_i + c_i y_{i+1} = -h^2 r_i$$

Il corrispondente sistema tridiagonale sarà:

$$\begin{bmatrix} a_1 & -c_1 & & & & \\ -b_2 & a_2 & -c_2 & & & \\ & -b_3 & a_3 & -c_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -b_{n-1} & a_{n-1} & c_{n-1} \\ & & & & -b_n & a_n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ \\ y_n \end{bmatrix} = \begin{bmatrix} -h^2 r_1 + b_1 \alpha \\ -h^2 r_2 \\ -h^2 r_3 \\ \vdots \\ \\ -h^2 r_n + c_n \beta \end{bmatrix}$$

Il sistema deve avere un'unica soluzione e per $h \to 0$ la soluzione di questo problema deve tendere alla soluzione dell'equazione non discretizzata. Questa matrice deve essere diagonalmente dominante e se $q_i \leq 0$ questo avviene. Casi di particolare interesse avvengono quando le funzioni P, q, r sono costanti e in tal caso otteniamo Matrici di Toeplitz e quelle in cui la funzione di partenza è periodica che ci consentono di ottenere matrici circolanti. Lo sviluppo di equazioni differenziale nel caso multidimensionale sotto opportune condizioni da origine a matrici strutturate, limitandoci al caso bidimensionale e a problemi con valori agli estremi otteniamo matrici di Toeplitz a blocchi di Toeplitz o in caso di funzioni periodiche matrici circolanti a blocchi circolanti, questo se si rispettano le opportune condizioni segnalate nel caso monodimensionale.[6]

1.4 Approssimazione polinomiale ai minimi quadrati in $L^2[a, b]$

Consideriamo lo spazio di Hilbert $L^2[a, b]$ con prodotto interno

$$\langle f, g \rangle = \int_a^b f(x)g(x)dx$$

e con la norma

$$\| f \| = \langle f, f \rangle^{\frac{1}{2}}$$

indotta dal prodotto scalare. con $\prod_n$ denotiamo lo spazio dei polinomi di grado al più n. Vogliamo trovare il polinomio $p_n \in \prod_n$ che meglio approssima la f nel senso dei minimi quadrati, cioè

$$min \| f - p \|_2^2 = min_{p \in \prod_n} \int_a^b [f(x) - p(x)]^2 dx.$$

È possibile caratterizzare la migliore approssimazione nel senso dei minimi quadrati mediante il seguente teorema.

Teorema 1.1 *Sia $f(x) \in L^2[a, b]$, allora p_n èl'approssimazione di f in $\prod_n$ nel senso dei minimi quadrati se e solo se*

$$\langle f - p_n, p \rangle = 0$$

per ogni $p \in \prod_n$.

Se dotiamo $\prod_n$ della base canonica $1, x, x^2, \cdots, x^n$, il precedente teorema conduce al sistema lineare

$$\langle f - p_n, x_i \rangle = 0 i = 0, 1, \cdots, n$$

che è detto sistema delle equazioni normali. Se esprimiamo la migliore approssimazione nella base canonica

$$p_n(x) = \sum_{j=0}^{n} \alpha_j x^j$$

otteniamo

$$\langle p_n, x^i \rangle = \langle f, x^i \rangle, i = 0, \cdots, n$$

$$\sum_{j=0}^{n} \langle \alpha_j x^j, x^i \rangle = \langle f, x^i \rangle, i = 0, \cdots, n$$

$$\sum_{j=0}^{n} \langle x^j, x^i \rangle \alpha_j = \langle f, x^i \rangle, i = 0, \cdots, n.$$

Il sistema delle equazioni normali può essere scritto in forma matriciale

$$H\alpha = f$$

dove si ha

$$\alpha = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} \quad f = \begin{bmatrix} \langle f, 1 \rangle \\ \langle f, x \rangle \\ \vdots \\ \langle f, x^n \rangle \end{bmatrix}$$

e dove

$$\langle f, x^i \rangle = \int_a^b x^i f(x) dx$$

sono detti i momenti della funzione f. La matrice H è data da

$$H_{ij} = \langle x^i, x^j \rangle = \int_a^b x^{i+j} dx = \frac{x^{i+j+1}}{i+j+1} \Big|_a^b$$

ed è detta matrice di Gram. In particolare se $[a, b] = [01]$

$$H_{ij} = \int_0^1 x^{i+j} dx = \frac{1}{i+j+1}.$$

La matrice così ottenuta

$$H = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \vdots & & \frac{1}{n+1} \\ \frac{1}{3} & \vdots & \vdots & & \frac{1}{n+2} \\ \vdots & \vdots & \vdots & & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n-1} \end{bmatrix}$$

è detta matrice di Hilbert. Questa matrice ha la proprietà di essere simultaneamente di Cauchy e di Hankel. Essa risulta fortemente malcondizionata e si può dimostrare che l'andamento asintotico del suo condizionamento rispetto alla norma 2 è dato da $\kappa_2 \approx e^{3.5n}$.

1.5 Equazioni integrali

Un'equazione integrale di prima specie, con nucleo di convoluzione, si presenta nella forma

$$\int_a^b k(u, v) f(v) dv = \int_a^b k(u - v) f(v) dv = g(u)$$

passando al discreto, una volta semplificata la notazione nel seguente modo [13]:

$$k(u_i, v_j) = k(u_i - v_j) = k[(i - j) * h] = k(i - j) = k_{ij}$$

, $f(v_i) = f_i$ e $g(u_i) = g_i$ si ottiene:

$$\sum_{j=0}^{n} k_{ij} f_j = g_i$$

dove $u \in [a, b]$ con $u = u_0 \cdots u_n$ con $u_0 = a$, $u_n = b$ e $u_i = u_{i-1} + h = u_0 + i * h$ dove $v \in [a, b]$ con $v = v_0 \cdots v_n$ con $v_0 = a$, $v_n = b$ e $v_i = v_{i-1} + h = v_0 + i * h$ e h è l'intervallo tra due nodi consecutivi. Otteniamo sostanzialmente un sistema $Kf = g$ dove la matrice K è sotto determinate condizioni iniziali una matrice strutturata solitamente Toeplitz o circolante. .Estendere la discussione a un problema 2-D è semplice. Per due immagini $f(x, y)$ e $h(x, y)$ di dimensioni rispettivamente $A \times B$ e $C \times D$, l'immagine estesa di dimensione $M \times N$ può essere formata riempendo la funzione di partenza con zeri, la procedura consiste nel fare:

$$f_e(x, y) = \begin{cases} f(x,y) \begin{cases} 0 \leq x \leq A-1 \\ 0 \leq y \leq B-1 \end{cases} \\ 0 \begin{cases} A \leq x \leq M-1 \\ B \leq y \leq N-1 \end{cases} \end{cases}$$

e

$$h_e(x, y) = \begin{cases} h(x,y) \begin{cases} 0 \leq x \leq C-1 \\ 0 \leq y \leq D-1 \end{cases} \\ 0 \begin{cases} C \leq x \leq M-1 \\ D \leq y \leq N-1 \end{cases} \end{cases}$$

Trattando le funzioni estese $f_e(x, y)$ e $h_e(x, y)$ come periodiche in due dimensioni, con periodi M e N nelle direzioni x e y, rispettivamente, ci si riconduce alla convoluzione di queste due funzioni

$$g_e(x, y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f_e(m, n) h_e(x - m, y - n)$$

con $x = 0, 1, \cdots, M - 1$ e $y = 0, 1, \cdots, N - 1$ La convoluzione $g_e(x, y)$ è periodica dello stesso periodo di $f_e(x, y)$ e $h_e(x, y)$. La sovrapposizione dei singoli periodi di convoluzione è evitata scegliendo $M \geq A + C - 1$ e $N \geq B + D - 1$ f,g sono vettori colonna di dimensione MN. I primi N elementi di f sono gli elementi della prima riga, i secondo N elementi, della seconda riga e così via.

$$g = Hf + \eta$$

e

$$H_{MN*MN}$$

dove per H si intende

$$\begin{bmatrix} H_0 & H_{M-1} & \cdots & & H_1 \\ H_1 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & & \\ & & & & H_{M-1} \\ H_{M-1} & \cdots & & H_1 & H_0 \end{bmatrix}$$

e per H_j

$$\begin{bmatrix} h_e(j,0) & h_e(j,N-1) & \cdots & h_e(j,1) \\ h_e(j,1) & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \\ & & & h_e(j,N-1) \\ h_e(j,N-1) & \cdots & h_e(j,1) & h_e(j,0) \end{bmatrix}$$

1.6 Il toolbox per Matlab `smt`

Matlab [8] è un ambiente di calcolo che permette di lavorare con matrici sparse (memorizzando solo gli elementi diversi da zero e la loro posizione) oltre che piene e consente di aggiungere nuovi tipi di variabili (classi) e ridefinire le operazioni. Le operazioni con matrici sparse sono definite in modo da ridurre i tempi di esecuzione e l'utilizzo di memoria. Matlab utilizza automaticamente questi algoritmi in presenza di matrici sparse, allo stesso modo il pacchetto `smt` [11] si propone di aggiungere a Matlab una struttura di calcolo dedicata a matrici strutturate in maniera trasparente per l'utente. Esso consente il trattamento di matrici Toeplitz e circolanti sfruttando le loro proprietà per minimizzare l'occupazione di memoria , velocizzare le routine di calcolo e diminuire la complessita computazionale Siamo partiti dal toolbox `smt`,un pacchetto in grado di gestire in modo intelligente le proprietà di matrici strutturate Toeplitz e circolanti nel caso monodimensionale, in particolare il pacchetto minimizza l'occupazione di memoria sfruttando il fatto di poter salvare solo determinati elementi visto che entrambi i tipi di matrice sono a diagonali costanti e nella circolante l'ulteriore condizione che ogni singola colonna fosse uguale alla precedente shiftata si una posizione, ciò comporta che per una circolante è sufficiente una sola colonna per conoscere tutti gli elementi della matrice(si usa la prima), mentre per una toeplitz sono sufficienti una riga e una colonna salvo quando la matrice è hermitiana nel qual caso è sufficiente una colonna. Un'altra proprietà fondamentale di queste matrici consiste nel fatto che la matrice circolante moltiplicata per qualsiasi altro

tipo di matrice o vettore sfrutta la Fast Fourier Transform (FFT) quindi in generale il prodotto matrice vettore richiede 3 FFT riducendo la complessità da $O(n^2)$ a $O(nlog(n))$. Le matrici Toeplitz sono facilmente estendibili in forma circolante, ciò ingrandisce la matrice ma il vantaggio ottenuto dalla velocizzazione del calcolo è ben superiore. L'intento del pacchetto `smt` è quello di sfruttare appieno i vantaggi derivati dalla struttura della matrice in particolare sono state aggiunte a Matlab 2 nuove classi smtoep ed smcirc implementando l'ottimizzazione dell'occupazione di memoria e delle routine di calcolo veloce che sfruttano la FFT rimanendo totalmente trasparenti per l'utente. Le uniche matrici strutturate supportate da Matlab sono quelle sparse dove vengono memorizzati solo gli elementi diversi da zero e le loro coordinate, inoltre molte operazioni tra matrici sparse sono definite in modo da ridurre tempi di calcolo e occupazione di memoria. Il calcolo misto tra matrici sparse e non strutturate dà origine a matrici sparse o piene a seconda del tipo di operazioni coinvolte.

1.7 Obiettivo della tesi

L'idea è stata quella di estendere il pacchetto `smt` e creare un toolbox Matlab che sfruttasse le proprietà di matrici strutturate circolanti e di tipo Toeplitz anche quand queste hanno più dimensioni, ossia quando i loro elementi dipendono da indici vettoriali invece che scalari. Nel nostro caso ci siamo limitati a indici bi-dimensionali.

Analogamente a `smt`, l'implementazione è stata eseguita sfruttando la programmazione *object oriented* prevista dal linguaggio Matlab, che consente in modo relativamente semplice la creazione di nuove classi e l'*overloading* degli operatori e delle funzioni che operano sugli oggetti di tali classi. In questo modo è stato possibile ridurre al minimo l'occupazione di memoria e sfruttare opportuni algoritmi veloci per il calcolo che coinvolge le matrici strutturate.

Come applicazione è stato considerato un problema tipico dell'elettromagnetismo, legato allo studio di superfici selettive di frequenza, che viene presentato nell'ultimo capitolo.

Capitolo 2

Introduzione alle matrici multiindice

Siano, $m, n \in \mathcal{Z}^2$,cioè $m = (m_1, m_2)\,; n = (n_1, n_2)$ definiamo $\mathcal{I}_n = (i_1, i_2)$ tale che $i_1 = 1 \dots n_1, i_2 = 1 \dots n_2$ ciò implica che se avessimo un sistema $Ax = y$ esso potrebbe esprimersi nel modo seguente:

$$y_i = \sum_{j \in \mathcal{I}_n} a_{ij} x_j = \sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} a_{(i_1 i_2),(j_1 j_2)} x_{(j_1 j_2)}$$

Per $\mathcal{R}^{\underline{n}}$ intendiamo $\mathcal{R}^{n_1 * n_2}$,ove $\mathcal{R}$ può essere sostituito da $\mathcal{C}$,quindi $x \in \mathcal{R}^{\underline{n}}$ significa

$$x = (x_{\underline{i}}) = x_{(i_1,i_2)} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1i_2} \\ x_{21} & x_{22} & \cdots & x_{2i_2} \\ \vdots & \vdots & & \vdots \\ x_{1_1,1} & x_{i_1,2} & \cdots & x_{i_1,i_2} \end{bmatrix}$$

Una matrice bi-indice $A : \mathcal{R}^{\underline{n}} \to \mathcal{R}^{\underline{m}}$ è un array $A \in \mathcal{R}^{\underline{m} * \underline{n}}$ i cui elementi sono $a_{\underline{i},\underline{j}} = a_{(i_1,i_2),(j_1,j_2)}$ con $i_1 = 1 : m_1; i_2 = 1 : m_2; j_1 = 1 : n_1; j_2 = 1 : n_2$. A è quadrata se $\underline{m} = \underline{n}$ cioè $m_i = n_i$ con $i = 1, 2$. Sia $A \in \mathcal{C}^{\underline{m} * \underline{n}}$ allora la matrice aggiunta A^* si otterrà scambiando le righe di A con le sue colonne e coniugandone gli elementi $(A^*)_{\underline{i},\underline{j}} = \overline{a_{\underline{j},\underline{i}}}$ Se la matrice A è reale ($a_{\underline{i},\underline{j}} \in \mathcal{R}$)si parla di matrice trasposta $\left(A^T\right)_{\underline{i},\underline{j}} = a_{\underline{j},\underline{i}}$ Un vettore colonna x è una matrice $\underline{n} * 1$ cioè $x_{\underline{n}} = x_{(n_1,n_2)}$, per vettore riga x^T intendiamo una matrice $1 * \underline{n}$ cioè essa si presenta apparentemente in tutto e per tutto simile a una colonna ma dobbiamo tenere conto delle diverse dimensioni. Le matrici costituiscono uno spazio lineare di dimensione $\underline{m} * \underline{n}$ con somma e prodotto per uno scalare definiti da:

$$(A + B)_{\underline{i},\underline{j}} = a_{\underline{i},\underline{j}} + b_{\underline{i},\underline{j}}$$

$$(\alpha A)_{\underline{i},\underline{j}} = \alpha a_{\underline{i},\underline{j}}$$

Se $A \in \mathcal{R}^{\underline{m}*\underline{n}}$ e $B \in \mathcal{R}^{\underline{n}*\underline{p}}$ allora il prodotto righe per colonne, o più semplicemente prodotto, sarà la matrice $C = AB \in \mathcal{R}^{\underline{m}*\underline{p}}$ definita da:

$$c_{\underline{i},\underline{j}} = \sum_{\underline{k}=1}^{\underline{n}} a_{\underline{i},\underline{k}} * b_{\underline{k},\underline{j}} = \sum_{k_1}^{n_1} \sum_{k_2}^{n_2} a_{(i_1,i_2),(k_1,k_2)} * b_{(k_1,k_2),(j_1,j_2)}$$

dove $i_t = 1 : m_t; j_t = 1 : p_t, t = 1, 2$. Il prodotto matriciale è distributivo rispetto alla somma, associativo ma non commutativo. L'elemento neutro è la matrice identità $I = diag\,(1, \cdots, 1)$ tale che $i_{\underline{k},\underline{j}} = \delta_{\underline{k},\underline{j}}$ cioè $i = 1 \Longleftrightarrow \underline{k} = \underline{j}$ o meglio $k_t = j_t; t = 1, 2$. La **potenza** p-esima di una matrice è definita come il prodotto

$$A^p = \underbrace{AA \cdots A}_{p \text{ volte}}.$$

Ricordiamo che una matrice reale $m \times n$ rappresenta la generica **trasformazione lineare** tra gli spazi lineari $\mathcal{R}^{\underline{n}}$ e $\mathcal{R}^{\underline{m}}$. Il prodotto matriciale corrisponde allora alla composizione di due trasformazioni lineari. Osserviamo che il prodotto scalare di $\mathcal{R}^{\underline{n}}$ definito in precedenza e la norma da esso indotta possono essere espressi nella forma

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y}, \qquad \|\mathbf{x}\|_2 = \sqrt{\mathbf{x}^T \mathbf{x}}.$$

Una matrice quadrata A si dice **invertibile** o **non singolare** se esiste una matrice A^{-1}, detta matrice inversa, tale che $AA^{-1} = A^{-1}A = I$.

Alcune proprietà:

- $(AB)^T = B^T A^T$;
- $(AB)^{-1} = B^{-1} A^{-1}$;
- $(A^T)^{-1} = (A^{-1})^T$ (scriveremo per brevità A^{-T});
- A è invertibile se e solo se ha righe (e colonne) linearmente indipendenti.

Sia A una matrice biindice $n \times n$. Sia $\mathcal{I}_n \subset \mathbb{Z}^2$ l'insieme in cui varia ciascun indice della matrice. Possiamo definire

$$\det(A) = \sum_{\sigma \in \mathcal{S}(\mathcal{I}_n)} (-1)^{\#\sigma} \prod_{i \in \mathcal{I}_n} A_{i,\sigma_i}$$

dove

- $\mathcal{S}(\mathcal{I}_n)$ contiene tutte le permutazioni σ dell'insieme $\mathcal{I}_n$;

- $\sigma_i \in \mathcal{I}_n$ è l'i-esimo elemento della permutazione σ;
- $(-1)^{\#\sigma}$ è il segno di σ, cioè -1 elevato il numero di scambi che hanno prodotto la permutazione;

La formula è chiaramente indipendente dall'ordinamento, visto che nella sommatoria possiamo considerare le permutazioni σ in qualsiasi ordine (ognuna conserverà comunque il suo segno) e possiamo calcolare la produttoria in qualsiasi ordine. È quindi immediato osservare che per ottenere il valore di $\det(A)$ possiamo introdurre su $\mathcal{I}_n$ un ordine lineare e calcolare il determinante della corrispondente matrice monoindice. Chiaramente tale ordine trasforma anche ogni vettore biindice $\mathbf{x}$ in un vettore monoindice $\mathrm{vec}(\mathbf{x})$. Ricordiamo che, fissata una riga i, il **determinante** può essere calcolato mediante la **formula di Laplace**

$$\det(A) = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} \det(A_{ij}),$$

essendo A_{ij} la sottomatrice che si ottiene da A eliminando la i-esima riga e la j-esima colonna. Richiamiamo, inoltre, alcune sue proprietà:

- $\det(A^T) = \det(A)$, $\det(A^*) = \overline{\det(A)}$, $\det(A^{-1}) = \det(A)^{-1}$;
- $\det(AB) = \det(A)\det(B)$, $\det(\alpha A) = \alpha^n \det(A)$ dove $n = (n_1 n_2)$;
- uno scambio di due righe (o due colonne) in una matrice induce un cambio di segno del suo determinante;
- A è non singolare se e solo se $\det(A) \neq 0$.

Il **rango** *rank*(A) di una matrice può essere definito indifferentemente come il massimo numero di righe (o colonne) linearmente indipendenti o come l'ordine della più grande sottomatrice con determinante non nullo.

Consideriamo ora il sistema lineare biindice $A\mathbf{x} = \mathbf{b}$. In componenti si può scrivere

$$b_i = \sum_{j \in \mathcal{I}_n} A_{ij} x_j, \quad i \in \mathcal{I}_n,$$

ma vale anche la rappresentazione

$$\mathbf{b} = \sum_{j \in \mathcal{I}_n} A_{:,j} x_j,$$

dove $A_{:,j}$ indica il j-esimo vettore "colonna" di A. Chiaramente si può anche scrivere

$$\mathrm{vec}(\mathbf{b}) = \sum_{j \in \mathcal{I}_n} \mathrm{vec}(A_{:,j}) x_j.$$

Dire che il sistema è risolubile significa che per ogni $\mathbf{b}$ esiste un unico $\mathbf{x}$ che verifica il sistema, quindi che i vettori $A_{:,j}$, $j \in \mathcal{I}_n$, sono linearmente indipendenti. Questo avverrà se e solo se la matrice (monoindice) le cui colonne sono le componenti dei vettori $A_{:,j}$ prese in un dato ordine, ossia le cui colonne sono $\mathrm{vec}(A_{:,j})$[l'ordine lessicografico di Matlab prevede che tramite la funzione vec(:) che trasforma la mia matrice in un vettore vengano ordinati gli elementi per colonne e in caso di colonna multi-indice prima l'indice 1 rispetto all'indice 2], ha rango massimo, cioè se e solo se il suo determinante è diverso da zero.

Per la definizione data di determinante, questo equivale alla condizione $\det(A) \neq 0$.

2.1 Autovalori e autovettori

Si dicono **autovalore** ed **autovettore** di una matrice A uno scalare λ ed un vettore $\mathbf{x} \neq 0$ che verifichino la relazione

$$A\mathbf{x} = \lambda \mathbf{x}. \tag{2.1}$$

Riscriviamo l'equazione (2.1) nella forma

$$(A - \lambda I)\mathbf{x} = 0.$$

Perché questo sistema lineare omogeneo ammetta una soluzione non nulla è necessario che il suo determinante

$$p_A(\lambda) = \det(A - \lambda I)$$

si annulli. Dal momento che $p_A(\lambda)$ è un polinomio di grado n in λ, detto **polinomio caratteristico** di A, il *Teorema fondamentale dell'Algebra* assicura l'esistenza di n autovalori (non necessariamente reali e distinti) che possono essere determinati calcolando gli zeri di $p_A(\lambda)$.

Per ciascun autovalore λ_k, $k = 1, \ldots, n$, una soluzione non nulla del sistema singolare omogeneo

$$(A - \lambda_k I)\mathbf{x} = 0$$

fornisce il corrispondente autovettore, che, nel caso in cui la matrice $A - \lambda_k I$ abbia rango $n-1$, resta quindi determinato a meno di una costante moltiplicativa.

Definiamo **spettro** di una matrice l'insieme dei suoi autovalori

$$\sigma(A) = \{\lambda_1, \ldots, \lambda_n\}$$

e **raggio spettrale** il massimo dei moduli degli autovalori

$$\rho(A) = \max_{k=1,\ldots,n} |\lambda_k|.$$

Alcune proprietà:

- $\det(A) = \prod_{k=1}^{n} \lambda_k$;
- $\sigma(A^T) = \sigma(A)$, $\sigma(A^{-1}) = \{\lambda_1^{-1}, \ldots, \lambda_n^{-1}\}$, $\sigma(A^p) = \{\lambda_1^p, \ldots, \lambda_n^p\}$;
- ad autovalori distinti corrispondono autovettori indipendenti;
- se un autovettore $\mathbf{x}$ è noto, il **quoziente di Rayleigh** $\frac{\mathbf{x}^T A \mathbf{x}}{\mathbf{x}^T \mathbf{x}}$ fornisce il corrispondente autovalore.

La **molteplicità algebrica** di un autovalore è la sua molteplicità come zero del polinomio caratteristico. Si definisce, invece, **molteplicità geometrica** di un autovalore il massimo numero di autovettori linearmente indipendenti ad esso corrispondenti. In generale per ogni autovalore si ha sempre

$$\text{molteplicità geometrica} \leq \text{molteplicità algebrica}.$$

Se per un autovalore vale il minore stretto, la matrice viene detta **difettiva**.

2.2 Matrici di forma particolare

[12]

Una matrice possiede una **struttura** (o *è strutturata*) quando ha delle proprietà che rendono più agevole la risoluzione di un problema che la coinvolge (inversione, risoluzione di un sistema lineare, calcolo di autovalori, etc.). Elenchiamo di seguito i tipi più comuni di matrici strutturate.

2.2.1 Matrici Hermitiane.

Una matrice quadrata complessa è **Hermitiana** se coincide con la sua aggiunta ($A = A^*$). Una matrice reale A si dice **simmetrica** se $A = A^T$. Una matrice Hermitiana A è **definita positiva** se

$$\mathbf{x}^* A\mathbf{x} > 0, \quad \forall \mathbf{x} \in \mathbb{C}^n \setminus \{0\}. \tag{2.2}$$

La stessa definizione è valida per matrici simmetriche reali, sostituendo $\mathbb{C}^n$ con $\mathbb{R}^n$. Una matrice si dice **semidefinita positiva** se nella diseguaglianza (2.2) vale il maggiore uguale invece del maggiore stretto.

Il seguente teorema enuncia la principale proprietà delle matrici Hermitiane.

Se A è Hermitiana, i suoi autovalori sono reali ed esiste per $\mathbb{C}^n$ una base di autovettori ortonormali. Se A è anche definita positiva, gli autovalori sono positivi.

2.2.2 Matrici unitarie.

Una matrice complessa è **unitaria** se la sua inversa coincide con l'aggiunta

$$Q^*Q = QQ^* = I.$$

Una matrice reale è **ortogonale** se l'inversa coincide con la trasposta

$$Q^TQ = QQ^T = I.$$

Ovviamente, dal punto di vista computazionale il maggior vantaggio di una matrice unitaria è quello di poter essere invertita senza alcun calcolo, il che consente di risolvere immediatamente un sistema lineare. Ricordiamo anche alcune proprietà, altre verranno richiamate in seguito.

Se Q è unitaria, allora

1. $|\det(Q)| = 1$, (se Q è reale $\det(Q) = \pm 1$);
2. $\|Q\mathbf{x}\|_2 = \|\mathbf{x}\|_2$, per ogni vettore $\mathbf{x} \in \mathbb{C}^n$.

La dimostrazione si ottiene applicando le proprietà dei determinanti e la definizione di norma 2.

Una matrice è **diagonale** se ha diversi da zero solo gli elementi diagonali cioè se

$$a_{i,j} = 0 \Longrightarrow i \neq j; a_{i,j} \neq 0 \Longrightarrow i = j;$$

I sistemi lineari con matrice diagonale sono particolarmente semplici da risolvere, come vedremo. Un'altra proprietà importante è la seguente:

Teorema 2.1 *Se T è diagonale il suo determinante è dato dal prodotto degli elementi diagonali. Inoltre gli autovalori coincidono con gli elementi diagonali.*

Dimostrazione *Si ottiene sviluppando il determinante di T (e di $T-\lambda I$) mediante la regola di Laplace.*

Ricordiamo che le matrici diagonali formano un'*algebra*. Questo significa che l'inversa di una matrice diagonale, quando esiste, e il prodotto di due matrici diagonali sono ancora matrici diagonali dello stesso tipo. La stessa proprietà vale per le matrici unitarie, ma in questo caso non si può parlare di algebra dato che l'unitarietà non si trasmette attraverso la somma di matrici ed il prodotto per uno scalare.

2.3 Metodi diretti per la soluzione di sistemi lineari.

I sistemi di equazioni lineari sono i preblemi numerici che si incontrano più spesso nelle applicazioni della Matematica. Per la loro risoluzione a differenza della quasi totalità dei problemi non lineari, sono disponibili algoritmi finiti, i cosiddetti metodi diretti. Esistono numerosi metodi diretti, spesso basati su idee molto diverse tra loro, ma che condividono la stessa strategia di base. Questa consiste nel trasformare, con un numero finito di passi, un sistema lineare generico in un sistema equivalente, ma dotato di una struttura particolare che ne rende più semplice la risoluzione.

Condizionamento di un sistema lineare

Un sistema di n equazioni lineari in n incognite assume la forma

ma viene spesso indicato in maniera più compatta utilizzando la notazione matriciale

$$A\mathbf{x} = \mathbf{b}. \tag{2.3}$$

dove $A = (a_{i,j})_{i,j=1}^{n}$ è la matrice dei coefficienti,

$$\mathbf{b} = (b_1, \cdots, b_n)^T$$

è il vettore dei termini noti e

$$\mathbf{x} = (x_1, \cdots, x_n)^T$$

è la soluzione. Fissata una norma matriciale, si definisce numero di condizionamento di un sistema lineare la quantità $\kappa(A) = \|A\| \cdot \|A^{-1}\|$,essendo A la matrice dei coefficienti del sistema. Il numero di condizionamento di un problema quantifica la sensibilità del risultato rispetto agli errori sui dati. Specificatamente si parlerà di problema malcondizionato se per piccole perturbazioni sui dati si hanno grandi perturbazioni sui risultati. Viceversa un problema sarà ben condizionato se piccole perturbazioni sui dati causano piccole perturbazioni nei risultati. Un esempio chiarificatore di questo concetto è dato dal seguente sistema

$$\begin{cases} x-y=1 \\ x-\alpha y=0 \end{cases}$$

con $\alpha \in \mathbb{R}$ Questo sistema dal punto di vista matriciale si può scrivere nella forma $Ax = b$ con

$$A = \begin{bmatrix} 1 & -1 \\ 1 & -\alpha \end{bmatrix} b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Consideriamo una piccola perturbazione sui dati ed in particolare

$$\alpha_1 = 1.00001$$

e

$$\alpha = 0.99999$$

e osserviamo cosa succede ai risultati. Nel caso di $\alpha = \alpha_1$ abbiamo

$$\begin{cases} x=10^5+1 \\ y=10^5 \end{cases}$$

e con $\alpha = \alpha_2$ abbiamo

$$\begin{cases} x=-10^5+1 \\ y=-10^5 \end{cases}$$

si ha cioè un errore relativo dell'ordine di 10^5 sulla soluzione a fronte di una variazione di 10^{-5} sui dati. Infatti in entrambi i casi per il condizionamento si ha il valore $\kappa(A) = 4 \cdot 10^5$ relativamente alla norma 2.Le informazioni sul condizionamento di un sistema lineare sono cruciali, in quanto operando su un calcolatore digitale tutti i numeri reali utilizzati sono affetti da un errore relativo dell'ordine della precisione di macchina (circa 10^{-16}, se si utilizzano variabili in doppia precisione). Inoltre, nelle applicazioni i dati sono spesso perturbati in modo molto più consistente a causa di errori sperimentali e/o di misura. La successione delle operazioni che porta alla soluzione di un problema è detta algoritmo di risoluzione. Spesso per uno stesso problema esistono diversi algoritmi e quindi è cruciale sapere quale degli eventuali

candidati risulta il migliore. Per questo ultimo concetto si devono fare una serie di distinzioni, infatti un algoritmo può risultare più veloce dal punto di vista del tempo impiegato da un elaboratore per eseguirlo, ma non essere sufficientemente accurato. Definiremo stabilità di un algoritmo la sensibilità dell'algoritmo rispetto agli errori sulle operazioni. La stabilità è una proprietà intrinseca di un algoritmo e algoritmi differenti risulteranno più o meno stabili. Definiremo complessità computazionale di un algoritmo il numero di operazioni aritmetiche che sono necessarie per il calcolo effettivo della soluzione. Il termine complessità avrà prevalentemente un significato aritmetico come l'intuizione della parola lascia intendere. Nella sua valutazione la complessità di un algoritmo è sempre legata alla dimensione del problema in esame. Se quest'ultima è, la funzione di n che rappresenta la sua complessità sarà spesso considerata per il suo comportamento asintotico, cioè al crescere indefinito di n. Per chiarezza con $g(n) = O(f(n))(f(n), g(n) \geq 0)$ si intender'a il fatto che la funzione $g(n)$ non cresce più rapidamente di $f(n)$, cio'e che esiste una costante c tale che $g(n) \leq cf(n)$ con l'eccezione di un insieme (eventualmente vuoto) di valori non negativi di n Il simbolo $O(1)$ denoterà una funzione delimitata, al crescere di n , da una costante. Per unità di misura della complessità di un algoritmo si intenderà una singola operazione aritmetica $+, -, *, /$, spesso indicata con flop (floating point operation). La complessità di calcolo può essere riguardata principalmente da due punti di vista: la sintesi e l'analisi di algoritmi. La sintesi consiste nella costruzione di algoritmi sempre più efficienti e veloci; l'analisi consiste invece nello studio di limiti inferiori della complessità di un problema e riguarda più direttamente il grado di difficoltà intrinseca della sua risoluzione.

2.4 Norme di matrici bi-indice

Siano, $A_{i,j}$ e y_j rispettivamente una matrice e un vettore bi-indice dove:

$$i \in \mathcal{I}_m, j \in \mathcal{I}_n$$

dove $i_1 = 1 : m_1, i_2 = 1 : m_2, j_1 = 1 : n_1, j_2 = 1 : n_2$, ció implica che se avessimo un sistema y=Ax esso potrebbe esprimersi nel modo seguente:

$$y_i = \sum_{j \in \mathcal{I}_n} a_{ij} x_j = \sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} a_{(i_1 i_2),(j_1 j_2)} x_{(j_1 j_2)}$$

La norma infinito di un vettore é il massimo dei valori assoluti dei singoli componenti del vettore ($\| y \|_\infty = \max_j |y_j|$), ció comporta nel bi-indice che

$\| y \|_\infty = \max_{j\in\mathcal{I}_n} |y_j|$, vogliamo dimostrare che:

$$\|A\|_\infty = \max_{\|y\|_\infty=1} \|Ay\|_\infty$$

partiamo da:

$$\|Ay\|_\infty = \max_{i\in\mathcal{I}_m} \left| \sum_{j\in\mathcal{I}_n} a_{i,j} \cdot y_j \right| \leq \max_{i\in\mathcal{I}_m} \sum_{j\in\mathcal{I}_n} |a_{(i,j)}| \cdot |y_j| \leq \max_{j\in\mathcal{I}_n} |y_j| \cdot \max_{i\in\mathcal{I}_m} \sum_{j\in\mathcal{I}_n} |a_{(i,j)}|$$

ma $\max_{j\in\mathcal{I}_n} |y_j| = 1$ ció implica:

$$\|A\|_\infty = \max_{i\in\mathcal{I}_m} \sum_{j\in\mathcal{I}_n} |a_{(i,j)}|$$

per avere uguaglianza ci serve un vettore y siffatto: $y = sign(a_{i_{max},j})$ infatti in questo modo gli elementi della riga i_{max} di A vengono sommati in valore assoluto, a questo punto $\|A\|_\infty = \max \|Ay\|_\infty$ cioé la norma infinito di A é il massimo delle somme righe. Ma in una matrice bi-indice la riga i é cosí definita:

$$row_i(A) = a_{i,:} = (a_{i,j})_{j\in\mathcal{I}_n}$$

la norma infinito di A corrisponde al massimo della sommatoria degli elementi di ciascuna riga presi in valore assoluto. Ora concentriamoci sulla norma 1, quindi stavolta abbiamo bisogno di un vettore x la cui norma 1 é 1 e cioé nel bi-indice:

$$\| x \|_1 = \sum_{j\in\mathcal{I}_n} |x_j| = 1$$

quindi:

$$\| A \|_1 = \max_{\|x\|_1=1} \| Ax \|_1$$

partiamo da:

$$\| Ax \|_1 = \sum_{i\in\mathcal{I}_m} |(Ax)_i| = \sum_{i\in\mathcal{I}_m} \left| \sum_{j\in\mathcal{I}_n} a_{(i,j)} \cdot x_j \right| \leq \sum_{i\in\mathcal{I}_m} \sum_{j\in\mathcal{I}_n} |a_{(i,j)}| \cdot |x_j|$$

che puó essere riscritta nella seguente maniera:

$$\sum_{j\in\mathcal{I}_n} |x_j| \sum_{i\in\mathcal{I}_m} |a_{(i,j)}| \leq \max_{j\in\mathcal{I}_n} \sum_{i\in\mathcal{I}_m} |a_{(i,j)}| \cdot \sum_{j\in\mathcal{I}_n} |x_j|$$

tenuto conto che:

$$\sum_{j\in\mathcal{I}_n} |x_j| = 1$$

ció implica:

$$\| A \|_1 \leq \max_{j \in \mathcal{I}_n} \sum_{i \in \mathcal{I}_m} |a_{(i,j)}|$$

per avere l'uguaglianza abbiamo bisogno di un vettore x siffatto : $x = e_{j_{max}}$ cioé si tratta di un vettore che ha tutti zeri tranne l'elemento $j_{max} \in \mathcal{I}_n$ che corrisponde a quello per il quale si ha la colonna di A con la somma massima. Infatti:

$$Ae_{j_{max}} = \sum_{i \in \mathcal{I}_m} \sum_{j \in \mathcal{I}_n} a_{i,j} e_{j_{max}} = \sum_{i \in \mathcal{I}_m} a_{i,j_{max}}$$

La norma 1 di A é il massimo delle somme colonna e visto che in una matrice bi-indice la colonna j é cosí definita:

$$col_j(A) = a_{:,j} = (a_{i,j})_{i \in \mathcal{I}_m}$$

la norma 1 di A corrisponde al massimo della sommatoria degli elementi di ciascuna colonna presi in valore assoluto. Passiamo ora a trattare il caso della norma 2, sia y tale che$\|y\|_2 = 1$ e cioé:

$$\sqrt{y^T y} = \sum_{j \in \mathcal{I}_n} y_j^2 = 1$$

Ció implica che la sommatoria dei quadrati dei singoli elementi del vettore bi-indice sia 1. Si parte come al solito da:

$$\|A\|_2 = \sup_{\|y\|_2 = 1} \|Ay\|_2$$

$$\|Ay\|_2 = (Ay)^T (Ay) = y^T A^T A y$$

poiché $A^T A$ é hermitiana, essa ha un insieme di $n_1 * n_2$ autovettori ortogonali $u_1, u_2, u_{n_1 \cdot n_2}$ tali che: $u_j^T u_k = \delta_{jk}, A^T A u_s = \lambda_s u_s$ Inoltre, essendo la matrice $A^T A$ semidefinita positiva, si ha: $\lambda_s \geq 0$. Dalla rappresentazione de vettore y nella base $\{u_i\}$:

$$y = \sum_{s \in \mathcal{I}_n} \alpha_s u_s$$

si ha:

$$\|A\|_2 = \sum_{r \in \mathcal{I}_n} \alpha_r u_r^T A^T A \sum_{s \in \mathcal{I}_n} \alpha_s u_s = \sum_{r \in \mathcal{I}_n} \alpha_r u_r^T \sum_{s \in \mathcal{I}_n} \alpha_s \lambda_s u_s = \sum_{s \in \mathcal{I}_n} \lambda_s |\alpha_s|^2$$

$$\leq \max_{s \in \mathcal{I}_n} \lambda_s \sum_{r \in \mathcal{I}_n} |\alpha_r|^2 = \max_s \lambda_s = \rho(A^T A)$$

e quindi, $\rho^{\frac{1}{2}}(A^TA)$ é una limitazione superiore di $\|A\|_2$. Ponendo $y = u_s$ ove s é definito come l'indice per cui $\lambda_s = \rho(A^TA)$, abbiamo:

$$\|Au_s\|_2 = (u_s^TA^TAu_s)^{\frac{1}{2}} = \rho^{\frac{1}{2}}(A^TA)$$

la norma 2 é quindi uguale anche nel bi-dimensionale alla radice del raggio spettrale di A^TA. Passiamo ora a trattare la norma di Frobenius:

$$\|A\|_F = (\sum_{i\in\mathcal{I}_m}\sum_{j\in\mathcal{I}_n} a_{ij}^2)^{\frac{1}{2}}$$

quindi nel bi-indice la norma di Frobenius é la radice quadrata della sommatoria dei quadrati dei singoli elementi.

2.5 Matrici Circolanti bi-indice

Le definizioni per una matrice circolante nel caso bi-indice non variano rispetto al mono-indice a patto di considerare la dimensione n della matrice come un vettore di componenti $[n_1, n_2]$ e quindi gli elementi c_{ij} della matrice saranno: $c_{k_1,k_2} = c_{(i_1-j_1,i_2-j_2)}$ Come si vedrà nel capitolo 3, abbiamo due possibilità per costruire la nostra matrice bi-indice e abbiamo deciso di visualizzarla come matrice mono-indice per coerenza col trattamento classico di queste strutture e per poter eseguire dei test sui calcoli. Diamo una breve descrizione di alcune semplici operazioni (somma, prodotto, calcolo degli autovalori,inversione). Consideriamo le seguenti matrici:

$$C_1 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \\ 3 & 4 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix}$$

generata dalle due possibili istruzioni:
1) $C_1 = smcirc2(C_1.c)$ con $C_1.c = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$
2 $C_1 = smcirc2(smcirc([1;2]); smcirc([3;4])$

$$C_2 = \begin{bmatrix} 5 & 6 & 7 & 8 \\ 6 & 5 & 8 & 7 \\ 7 & 8 & 5 & 6 \\ 8 & 7 & 6 & 5 \end{bmatrix}$$

con $C_2.c = \begin{bmatrix} 5 & 7 \\ 6 & 8 \end{bmatrix}$ Per sommare le due matrici è sufficiente sommare i due gnomoni, ovvero le componenti $C_i.c$ e da esse ricostruire la matrice, gli autovalori come nel caso monodimensionale saranno dati dalla somma degli autovalori delle matrici di partenza. Se indichiamo con $C_{1,2}$ la matrice risultante dalla somma otteniamo da $C_{1,2}.c = C_1.c + C_2.c = \begin{bmatrix} 6 & 10 \\ 8 & 12 \end{bmatrix}$

$$C_{1,2} = \begin{bmatrix} 6 & 8 & 10 & 12 \\ 8 & 6 & 12 & 10 \\ 10 & 12 & 6 & 8 \\ 12 & 10 & 8 & 6 \end{bmatrix}$$

analogo discorso vale per la differenza. Vediamo il caso della moltiplicazione. Come nel caso monodimensionale essendo gli autovettori di una matrice circolante di dimensione n con $n = [n_1, n_2]$ dipendenti solo da n il prodotto tra due matrici si svolge molto velocemente tramite 3 FFT2 e un prodotto punto a punto(vedi paragrafo 2.5.1 e 1.1.4). È altresì molto semplice il calcolo degli autovalori e di conseguenza il calcolo della matrice inversa. Gli autovalori, come dimostreremo in seguito(paragrafo 2.5.1) sono la FFT2 dello gnomone della matrice, ovvero dalla sua 'prima colonna'(intesa come vettore bi-indice), quindi per invertire una matrice circolante è sufficiente invertire punto a punto ogni singolo autovalore e calcolare la IFFT2 della matrice(vettore bi-indice) dei valori ottenuti e otterremo lo gnomone della matrice inversa.

2.5.1 Autovalori di una matrice circolante bi-indice

Consideriamo una matrice circolante bidimensionale $C_{(j,l)}$(ricordiamo che per matrice circolante si intende una matrice $C_{j,l}$ i cui elementi sono: $c_{j-l} = c_k$ dove $k = j - l$ e $c_{k+n} = c_k$ dove n é la dimensione di j e l , nel bidimensionale $k = k_1, k_2$, $n = n_1, n_2$),dove $j = j_1 j_2$ é l'indice di riga e $l = l_1 l_2$ é l'indice di colonna degli elementi di C, dove gli indici variano in questo modo: $j_1, l_1 = 1 : n_1; j_2, l_2 = 1 : n_2$ in quanto la matrice é quadrata e una matrice bidimensionale di Fourier $F_{j,l}$ i cui elementi$f_{i,j}$ sono definiti come ω^{-jl} e in due dimensioni $\omega_1^{-j_1 l_1} \cdot \omega_2^{-j_2 l_2}$ e moltiplichiamo C per la colonna l di F, otteniamo:

$$\left[C \cdot F_{(\cdot,l)}\right] = \sum_{r_1=0}^{n_1-1} \sum_{r_2=0}^{n_2-1} C_{(j_1-r_1, j_2-r_2)} \omega_1^{-r_1 l_1} \omega_2^{-r_2 l_2}$$

dove con ω_i si intende $\mathrm{e}^{\frac{2\pi\mathbf{i}}{n_i}}$,ponendo $j_i - r_i = k_i$ si ottiene:

$$\sum_{k_1=j_1}^{j_1-(n_1-1)} \sum_{k_2=j_2}^{j_2-(n_2-1)} C_{(k_1,k_2)} \omega_1^{-(j_1-k_1)l_1} \omega_2^{-(j_2-k_2)l_2}$$

che possiamo riscrivere in questo modo:

$$\omega_1^{-j_1 l_1} \omega_2^{-j_2 l_2} \sum_{k_1=j_1}^{j_1-(n_1-1)} \omega_1^{k_1 l_1} \sum_{k_2=j_2}^{j_2-(n_2-1)} C_{(k_1,k_2)} \omega_2^{k_2 l_2}$$

spezziamo la seconda sommatoria in due, una di soli termini positivi e una di soli termini negativi ottenendo:

$$\omega_1^{-j_1 l_1} \omega_2^{-j_2 l_2} \sum_{k_1=j_1}^{j_1-(n_1-1)} \omega_1^{k_1 l_1} \left(\sum_{k_2=j_2-(n_2-1)}^{-1} C_{(k_1,k_2)} \omega_2^{k_2 l_2} + \sum_{k_2=0}^{j_2} C_{(k_1,k_2)} \omega_2^{k_2 l_2} \right)$$

focalizziamo l'attenzione sulla sommatoria $\sum_{k_2=j_2-(n_2-1)}^{-1} C_{(k_1,k_2)} \omega_2^{k_2 l_2}$ questa puó essere riscritta nel modo seguente:

$$\sum_{k_2=j_2-1}^{n_2-1} C_{(k_1,k_2+n_2)} \omega_2^{(k_2+n_2) l_2} = \sum_{k_2=j_2-1}^{n_2-1} C_{(k_1,k_2+n_2)} \omega_1^{n_1 l_1} \omega_2^{k_2 l_2}$$

tenuto conto che $\omega_2^{n_2 l_2} = 1$, in quanto sostituendo a ω_2 il suo valore $\mathrm{e}^{\frac{2\pi\mathbf{i}}{n_2}}$ otteniamo: $\mathrm{e}^{\frac{2\pi n_2 \mathbf{i}}{n_2}} = \mathrm{e}^{2\pi\mathbf{i}}$, e che $C_{(k_1,k_2+n_2)} = C_{(k_1,k_2)}$ (per la definizione stessa di matrice circolante) si ottiene:

$$\sum_{k_2=j_2-(n_2-1)}^{-1} C_{(k_1,k_2)} \omega_2^{k_2 l_2} = \sum_{k_2=j_2+1}^{n_2-1} C_{(k_1,k_2)} \omega_2^{k_2 l_2}$$

e di conseguenza:

$$\sum_{k_2=j_2-(n_2-1)}^{-1} C_{(k_1,k_2)} \omega_2^{k_2 l_2} + \sum_{k_2=0}^{j_2} C_{(k_1,k_2)} \omega_2^{k_2 l_2} = \sum_{k_2=0}^{n_2-1} C_{(k_1,k_2)} \omega_2^{k_2 l_2}$$

andando a sostituire otteniamo:

$$\omega_1^{-j_1 l_1} \omega_2^{-j_2 l_2} \sum_{k_1=j_1}^{j_1-(n_1-1)} \omega_1^{k_1 l_1} \sum_{k_2=0}^{n_2-1} C_{(k_1,k_2)} \omega_2^{k_2 l_2}$$

facciamo lo stesso ragionamento con l'altra sommatoria:

$$\omega_1^{-j_1 l_1}\omega_2^{-j_2 l_2}\sum_{k_2=0}^{(n_2-1)}\omega_2^{k_2 l_2}\Big(\sum_{k_1=j_1-(n_1-1)}^{-1}C_{(k_1,k_2)}\omega_1^{k_1 l_1}+\sum_{k_1=0}^{j_1}C_{(k_1,k_2)}\omega_1^{k_1 l_1}\Big)$$

focalizziamo l'attenzione su:

$$\sum_{k_1=j_1-(n_1-1)}^{-1}C_{(k_1,k_2)}\omega_1^{k_1 l_1}$$

questa puó essere riscritta nel modo seguente:

$$\sum_{k_1=j_1-1}^{n_1-1}C_{(k_1+n_1,k_2)}\omega_1^{(k_1+n_1)l_1}=\sum_{k_1=j_1-1}^{n_1-1}C_{(k_1+n_1,k_2)}\omega^{n_1 l_1}\omega_1^{k_1 l_1}$$

ma $\omega_1^{n_1 l_1}=1$ e $C_{(k_1+n_1,k_2)}=C_{(k_1,k_2)}$ ció implica che:

$$\sum_{k_1=j_1-(n_1-1)}^{-1}C_{(k_1,k_2)}\omega_1^{k_1 l_1}=\sum_{k_1=j_1-1}^{n_1-1}C_{(k_1,k_2)}\omega_1^{k_1 l_1}$$

e di conseguenza:

$$\sum_{k_1=j_1-(n_1-1)}^{-1}C_{(k_1,k_2)}\omega_1^{k_1 l_1}+\sum_{k_1=0}^{j_1}C_{(k_1,k_2)}\omega_1^{k_1 l_1}=\sum_{k_1=0}^{n_1-1}C_{(k_1,k_2)}\omega_1^{k_1 l_1}$$

andando a sostituire otteniamo:

$$\omega_1^{-j_1 l_1}\omega_2^{-j_2 l_2}\sum_{k_1=0}^{n_1-1}\omega_1^{k_1 l_1}\sum_{k_2=0}^{n_2-1}C_{(k_1,k_2)}\omega_2^{k_2 l_2}$$

e riordinando:

$$\omega_1^{-j_1 l_1}\omega_2^{-j_2 l_2}\sum_{k_1=0}^{n_1-1}\sum_{k_2=0}^{n_2-1}C_{(k_1,k_2)}\omega_1^{k_1 l_1}\omega_2^{k_2 l_2}$$

ció dimostra che:

$$\big[C\cdot F_{(\cdot,l)}\big]=\omega_1^{-j_1 l_1}\omega_2^{-j_2 l_2}\sum_{k_1=j_1+1}^{n_1-1}\sum_{k_2=j_2+1}^{n_2-1}C_{(k_1,k_2)}\omega_1^{k_1 l_1}\omega_2^{k_2 l_2}$$

dove $\omega_1^{-j_1 l_1}\omega_2^{-j_2 l_2}$ é la colonna di indici $l_1 l_2$ di F e cioé l'autovettore di indice $l_1 l_2$ della matrice C, mentre $\sum_{k_1=j_1+1}^{n_1-1}\sum_{k_2=j_2+1}^{n_2-1}C_{(k_1,k_2)}\omega_1^{k_1 l_1}\omega_2^{k_2 l_2}$ é l'autovalore corrispondente.

per quanto riguarda le norme 1,infinito e frobenius possiamo sfruttare le proprietá della matrice circolante per semplificarne il calcolo. In particolare gli elementi che compongono una riga sono gli stessi non mantenendone l'ordine per tutte le righe e per tutte le colonne, ció comporta che:

$$\|C\|_1 = \|C\|_{\text{inf}} = \sum_{j=1}^{n} a_{1,j}$$

e che:

$$\|C\|_{fro} = \sqrt{n \cdot \sum_{j=1}^{n} a_{1,j}^2}$$

ció significa che la norma 1 e quella infinito sono uguali alla somma degli elementi della prima colonna(o riga) e che la norma di frobenius é uguale alla radice quadrata della sommatoria dei quadrati degli elementi della prima colonna(o riga) moltiplicata per la radice quadrata delle dimensioni della prima colonna(o riga).

2.6 Matrici di Toeplitz bi-indice

Anche per quanto riguarda le matrici di Toeplitz le definizioni non cambiano passando al bi-indice a patto diconsiderare le dimensioni n, m della matrice come vettori con due componenti, come abbiamo visto nel paragrafo 1.7 abbiamo 3 modi diversi per generare la matrice, a differenza delle circolanti, assegnando lo gnomone occorre assegnare anche le dimensioni della matrice in quanto, uno stesso gnomone può generare più matrci di Toeplitz. Oltretutto le matrici di Toeplitz possono essere anche rettangolari. Vediamo di seguito alcuni esempi di operazioni semplici. Per quanto riguarda la somma è sufficiente costruire la matrice Toeplitz di dimensioni pari a quelle delle due di partenza e con come gnomone la somma dei due gnomoni. Se ad esempio si ha:

$$T_1 = \begin{bmatrix} 1 & 3 & 2 & 4 \\ 3 & 1 & 4 & 2 \\ 2 & 4 & 1 & 3 \\ 4 & 2 & 3 & 1 \end{bmatrix}$$

generata dall'istruzione $T_1 = smtoep2(T.t, [2,2], [2,2])$ con $T_1.t = \begin{bmatrix} 4 & 3 & 4 \\ 2 & 1 & 2 \\ 4 & 3 & 4 \end{bmatrix}$ e

$$T_2 = \begin{bmatrix} 5 & 0 & 1 & -1 \\ 0 & 5 & -1 & 1 \\ 1 & -1 & 5 & 0 \\ -1 & 1 & 0 & 5 \end{bmatrix}$$

con gnomone $T_2.t = \begin{bmatrix} -1 & 0 & -1 \\ 1 & 5 & 1 \\ -1 & 0 & -1 \end{bmatrix}$ Se chiamiamo T_{21} la matrice somma essa sarà generata dallo gnomone

$$T_{21}.t = T_1.t + T_2.t = \begin{bmatrix} 3 & 2 & 3 \\ 3 & 6 & 3 \\ 3 & 2 & 3 \end{bmatrix}$$

e la matrice sarà

$$T_{21} = \begin{bmatrix} 6 & 3 & 3 & 3 \\ 3 & 6 & 3 & 3 \\ 3 & 3 & 6 & 3 \\ 3 & 3 & 3 & 6 \end{bmatrix}$$

Per quanto riguarda la moltiplicazione anche nel caso multi-dimensionale per sfruttare la velocità di calcolo tramite FFT le matrici Toeplitz vengono trasformate in circolanti mediante la funzione toeprem che merita un approfondimento. La funzione trasforma una matrice Toeplitz in una circolante il cui gnomone nella versione tight ha come dimensioni $[(m_1+n_1-1), (m_2+n_2-1)]$ e $[nextpow2(m_1+n_1-1), nextpow2(m_2+n_2-1)]$ dove nel secondo caso la matrice subisce un processo di embedding con degli zeri in determinate posizioni(o vedremo tra poco in dettaglio) che utilizza come dimensione la più piccola potenza di 2 maggiore di $n_i + m_i - 1$, questo consente un miglioramento dei risultati a fronte un leggero aumento dell'occupazione di memoria. Per quanto riguarda la trasformazione tight l'occupazione di memoria rimane invariata ma si ha enorme vantaggio dallo sfruttare per i calcoli la FFT2. Vediamo in dettaglio un esempio di come agisce la funzione toeprem. Una prima distinzione rispetto al caso mono-indice sta nel fatto che l'embedding tight non ha bisogno dell'aggiunta di zeri e sotto alcune condizioni dovute a paticolari dimensionamenti anche l'embedding $nextpow2$ potrebbe non aver bisogno dell'aggiunta di zeri come ad esempio una smtoep2 di dimensioni $[2,3] \times [3,2]$ in quanto lo gnomone generatore ha dimensione [4 4] che sono anche le potenze di 2 appena superiore a 2 e 3 che sono le dimensioni generatrici. Come regola generale si può asserire che se l'embedding tight dà

origine a uno gnomone le cui dimensioni sono potenze di due questo coincide con l'embedding *nextpow*2 dando anche vantaggi di calcolo. La prima fase dell'embedding consiste in un doppio circshift dello gnomone in maniera tale che gli elementi costituenti la prima colonna di T si ritrovino nell'angolo in alto a sinistra della matrice a questo punto si inseriscono le eventuali colonne di zeri, poi le eventuali righe di zeri e se completa col resto degli elementi. Vediamo 2 esempi: 1) matrice T[2 3]x[3 2]

$$T.t = \begin{bmatrix} t_{-2-1} & t_{-20} & t_{-21} & t_{-22} \\ t_{-1-1} & t_{-10} & t_{-11} & t_{-12} \\ t_{0-1} & t_{00} & t_{01} & t_{02} \\ t_{1-1} & t_{10} & t_{11} & t_{12} \end{bmatrix}$$

effettuato il doppio circshift(l'istruzione è circshift(T.t,(1-T.dim2(1),1-T.dim2(2)) otteniamo lo gnomone della matrice C che sarà

$$C.c = \begin{bmatrix} t_{00} & t_{01} & t_{02} & t_{0-1} \\ t_{10} & t_{11} & t_{12} & t_{1-1} \\ t_{-20} & t_{-21} & t_{-22} & t_{-2-1} \\ t_{-10} & t_{11} & t_{12} & t_{1-1} \end{bmatrix}$$

1) matrice T[2 2]x[2 2]

$$T.t = \begin{bmatrix} t_{-1-1} & t_{-10} & t_{-11} \\ t_{0-1} & t_{00} & t_{01} \\ t_{1-1} & t_{10} & t_{11} \end{bmatrix}$$

effettuato il doppio circshift(l'istruzione è circshift(T.t,(1-T.dim2(1),1-T.dim2(2)) otteniamo lo gnomone della matrice A che sarà

$$A = \begin{bmatrix} t_{00} & t_{01} & t_{0-1} \\ t_{10} & t_{11} & t_{1-1} \\ t_{-10} & t_{11} & t_{1-1} \end{bmatrix}$$

a questo punto tra gli elementi con indici tutti positivi e gli altri elementi dobbiamo aggiungere righe e colonne di zeri fino a raggiungere uno gnomone di dimensioni pari alla più piccola potenza di 2 maggiore maggiore delle dimensioni dello gnomone, in questo caso essendo lo gnomone 3x3, lo gnomone della circolante corrispondente sarà:

$$C.c = \begin{bmatrix} t_{00} & t_{01} & 0 & t_{0-1} \\ t_{10} & t_{11} & 0 & t_{1-1} \\ 0 & 0 & 0 & 0 \\ t_{-10} & t_{11} & 0 & t_{1-1} \end{bmatrix}$$

Capitolo 3

Il toolbox `smt2`

Si è cercato di tenere un approccio compatibile con quello di `smt`, così ad esempio una circolante riceve in ingresso la prima colonna, questa può essere sia sotto forma di matrice compatibile col fatto che sarebbe effettivamente la colonna con indice [1 1] di una matrice C a 4 indici, sia come insieme di m matrici smcirc compatibile con la trattazione classica BCCB vediamo un esempio: l'istruzione smcirc2(smcirc([a;b;c]);smcirc([d;e;f])) genererà la seguente matrice

$$C = \begin{bmatrix} a & c & b & d & f & e \\ b & a & c & e & d & f \\ c & b & a & f & e & d \\ d & f & e & a & c & b \\ e & d & f & b & a & c \\ f & e & d & c & b & a \end{bmatrix}$$

dove il comando smcirc([a;b;c]) darà origine alla matrice

$$A = \begin{bmatrix} a & c & b \\ b & a & c \\ c & b & a \end{bmatrix}$$

e il comando smcirc([d;e;f]) darà origine alla matrice

$$B = \begin{bmatrix} d & f & e \\ e & d & f \\ f & e & d \end{bmatrix}$$

nel complesso possiamo vedere la matrice come una concatenazione di matrici circolanti monodimensionale

$$C = \begin{bmatrix} A & B \\ B & A \end{bmatrix}$$

dove A e B sono matrici circolanti 3×3. Come nel caso monodimensionale non occorre passare le dimensioni della matrice in quanto una matrice circolante è sempre quadrata (e questo nel multi-indice significa che se indichiamo con m ed n i vettori dimensione di riga e colonna, questi debbono essere uguali e cioè m(i)=n(i) per tutti gli i(caso bi-indice i=2)). Le dimensioni di una circolante si ricavano dal vettore colonna, se questo viene dato sonno forma di colonna bidimensionale, le dimensioni di quest'ultima forniscono la dimensione delle righe e delle colonne di C, se viene data sotto forma di colonna di smcirc, il numero di elementi della colonna fornisce il primo elemento del vettore dimensione sia della riga che della colonna mentre la dimensione delle smcirc monodimensionali fornisce la seconda dimensione di riga e colonna. Ovviamente le matrici smcirc che compongono il vettore colonna devono avere tutte la stessa dimensione. Si ha un notevole vantaggio in termini di occupazione di memoria dall'utilizzo del toolbox in quanto a fronte di una matrice con $[m \times n] \times [m \times n]$ elementi ne memorizziamo $m \times n$ anzichè $m^2 \times n^2$ cioè questo consente di poter continuare ad alloccare la matrice in memoria come facente parte della classe 'smcirc2' o 'smtoep2' anche quando la corrispondente matrice in versione 'full'(cioè senza considerare la struttura) richiedesse troppo spazio per essere memorizzata. Si è preferito per motivi di coerenza con le routine di Matlab esistenti e col trattamento classico di matrici multi-indice strutturate visualizzare la matrice 'full' come matrice standard composta di blocchi a loro volta strutturati come visto in precedenza. Un procedimento analogo avviene per le matrici Toeplitz che possono ricevere in ingresso:
1) un vettore colonna composto da matrici smtoep[nel caso della circolante multi-indice devono avere tutte la stessa dimensione], in qual caso la matrice è hermitiana ovviamente condizione necessaria affinchè ciò accada è che la prima matrice smtoep della colonna sia a sua volta hermitiana;
2) un vettore riga e un vettore colonna sempre con componenti smtoep in questo caso la prima matrice del vettore riga e del vettore colonna devono essere uguali, se ciò non accade viene visualizzato uno warning e si impone che il primo elemento del vettore riga sia uguale al primo elemento del vettore colonna;
3) lo gnomone,che è un vettore bi-indice di dimensioni $[m_1+n_1-1, m_2+n_2-1]$ contenente tutti gli elementi che si ripetono all'interno della matrice,e le dimensioni della matrice, questo caso corrisponde nel monoindice al passaggio di un unico vettore composto dalla riga i cui elementi hanno subito un flip orizzontale e una trasposizione(coniugazione nel caso complesso) e dalla colonna privata del primo elemento. Se si assegna lo gnomone senza dimensioni la matrice viene generata quadrata. Vediamo alcuni esempi relativi a possibili istruzioni: se passiamo un colonna l'istruzione sarà del

tipo T=smtoep2(smtoep([d;e;a;b;c],3,3);smtoep([f;g;h])) in uscita avremo la T visualizzata come BTTB

$$T = \begin{bmatrix} a & e & d & f & g & h \\ b & a & e & g^* & f & g \\ c & b & a & h^* & g^* & f \\ f & g^* & h^* & a & e & d \\ g & f & g^* & b & a & e \\ h & g & f & c & b & a \end{bmatrix}$$

dove l'elemento f è reale e la struttura se indichiamo con A la smtoep([d;e;a;b;c],3,3) e con B la smtoep([f;g;h]) otteniamo:

$$T = \begin{bmatrix} A & B^* \\ B & A \end{bmatrix}$$

se invece consideriamo un'istruzione del tipo T=smtoep2(A;B;D,A E) dove il primo elemento di riga e colonna è lo stesso ma se così non fosse Matlab manderebbe un warning e metterebbe di default il primo elemento del vettore riga uguale al primo elemento del vettore colonna. Ovviamente anche le matrici D,E sono delle smtoep di dimensione 3×3 e l'uscita in questo caso sarà:

$$T = \begin{bmatrix} A & E \\ B & A \\ D & B \end{bmatrix}$$

se

$$D = \begin{bmatrix} q & w^* & r^* \\ w & q & w^* \\ r & w & q \end{bmatrix}$$

se

$$E = \begin{bmatrix} u & i^* & o^* \\ i & u & i^* \\ o & i & u \end{bmatrix}$$

si ottiene:

$$T = \begin{bmatrix} a & e & d & u & i^* & o^* \\ b & a & e & i & u & i^* \\ c & b & a & o & i & u \\ f & g^* & h^* & a & e & d \\ g & f & g^* & b & a & e \\ h & g & f & c & b & a \\ q & w^* & r^* & f & g^* & h^* \\ w & q & w^* & g & f & g^* \\ r & w & q & h & g & f \end{bmatrix}$$

Vediamo ora il comportamento se si riceve in ingresso lo gnomone, esso come già precedentemente accennato corrisponde al caso in cui nel mono-indice si passa un vettore con le dimensioni, nel pacchetto `smt` il passaggio di una colonna restituisce una matrice Toeplitz hermitiana per evitare ridondanza si è usato lo gnomone al posto di un eventuale vettore. Un unico gnomone genera a seconda delle dimensioni date diverse matrici di Toeplitz, questo fatto giustifica l'introduzione della funzione reshape che ricevendo in ingresso una matrice Toeplitz bi-indice e le dimensioni, se queste sono compatibili con lo gnomone generatore cioè se, ipotizzando $m = (m_1 m_2)$;$n = (n_1 n_2)$ le dimensioni della matrice che si vuole ottenere e $k = (k_1 k_2)$le dimensioni dello gnomone, devono valere contemporaneamente le relazioni: $k_1 = m_1 + n_1 - 1$ e $k_2 = m_2 + n_2 - 1$ Reshape si implementa facilmente, infatti una volta verificata la coerenza delle dimensioni crea una matrice smtoep2 partendo dal vecchio gnomone e con le nuove dimensioni. Come nel caso delle circolanti sono state implementate alcune opzioni di funzioni solo per coerenza di comportamento delle stesse tra operatori full ad esempio è possibile sommare/sottrarre e moltiplicare/dividere tra loro matrici Toeplitz o Circolanti bi-indice in cui sono uguali i prodotti tra le dimensioni di riga e colonna, ad esempio è possibile sottrarre una T([3 2],[4 4]) da una U([2 3],[2 8]) questo per coerenza con l'operazione full(T)-full(U).

3.1 Struttura del toolbox

Il toolbox è strutturato nello stesso modo di `smt` e ad esso è strettamente collegato non potendo girare indipendentemente da esso. Una volta scaricato il software sul p.c è necessario aggiungere questo al Matlab search path in modo che esso possa essere usato da qualsiasi directory. La cartella `smt2` cantiene una cartella @smcirc2, una @smtoep2, una private e una demo. vediamole in dettaglio:

- @smcirc2 e @smtoep2 contengono le funzioni che creano e manipolano gli oggetti di classe smcirc2 e smtoep2, ovvero matrici circolanti e di Toeplitz.
- private contiene alcune funzioni interne che non sono direttamente accessibili all'utente.
- demo contiene una cartella di tests che servono per effettuare le prove riportate nel capitolo 4 ed inoltre è presente la funzione validate 2 che serve a fare un check del sistema e verificare che le routine diano gli stessi risultati di Matlab.

La classe smcirc2 Un oggetto di classe smcirc2 si presenta nella seguente forma (compatta): $C = smcirc2$ con i campi

$$c : [n, m]$$

$$dim = [n, m]$$

$$ev = []$$

il campo ev che dovrebbe contenere gli autovalori è posto vuoto di default ma lo si può settare modificando il file smtconfig in modo che calcoli in automatico gli autovalori appena la matrice viene memorizzata. Sono state ridefinite molte operazioni (overloading) per la classe smcirc2, non di tutte in quanto alcune non sono state ritenute utili. Quando una nuova classe è aggiunta a Matlab, cè sempre un certo numero di funzioni che deve essere definito in modo che la classe si conformi alle regole sintattiche di Matlab.

Operatori matriciali			
`plus`	`A+B`	`power`	`A.^2`
`uplus`	`+A`	`mldivide`	`A\B`
`minus`	`A-B`	`mrdivide`	`A/B`
`uminus`	`-A`	`ldivide`	`A.\B`
`mtimes`	`A*B`	`rdivide`	`A./B`
`times`	`A.*B`	`transpose`	`A.'`
`mpower`	`A^2`	`ctranspose`	`A'`

Tabella 3.1: Ridefinizione operatori

Citiamo alcune funzioni:

- get serve per estrarre un campo da un oggetto;
- display definisce come visualizzare un oggetto;
- subsref ci consente di accedere al campo .c o a un singolo elemento della matrice circolante e di estrarre una parte della matrice(ricordiamo che se togliamo una o più righe e/o colonne a una matrice circolante otteniamo una matrice di Toeplitz);
- subsasgn serve per modificare il campo .c ;
- smtvalid viene chiamata da altre funzioni del toolbox per verificare se un oggetto è un operando valido nell'espressione.

Funzioni matematiche elementari			
`abs`	valore assoluto	`fix`	troncamento
`angle`	fase	`floor`	arrotondamento verso $-\infty$
`conj`	complesso coniugato	`ceil`	arrotondamento verso $+\infty$
`imag`	parte immaginaria	`round`	arrotondamento
`real`	parte reale	`sign`	funzione segno
Informazioni di base			
`size`	dimensione of array	`get`	estrae un campo
`length`	lunghezza dell'array	`isempty`	vero per array vuoti
`display`	mostra l'array	`isequal`	vero per array uguali
Operazioni e manipolazioni di array			
`diag`	estrae lediagonali di una matrice	`prod`	fa il prodotto degli elementi
`diff`	differenza/approx. derivativa	`reshape`	cambia le dimensioni
`full`	converte in formato full	`sum`	somma gli elementi
`max`	estrae la componente maggiore	`min`	estrae la componente minore
Funzioni di servizio			
`double`	converte in formato double	`subsasgn`	modifica i campi .c etc
`single`	converte in formato single	`subsindex`	consente di modificare l'indicizzazione
`isa`	vero per un oggetto della classe	`subsref`	consente l'accesso ai campi .c etc
`isfloat`	vero per oggetti in virgola mobile	`end`	fine indice
`isreal`	vero per elementi reali		
Operatori di algebra lineare			
`cond`	numero di condizionamento	`inv`	inverte la matrice
`det`	determinante	`norm`	norma della matrice
`eig`	autovalori e autovettori		

Tabella 3.2: Funzioni ridefinite

La classe smtoep2 Abbiamo già discusso come creare una matrice di Toeplitz, ora vediamo come essa viene visualizzata in forma compatta: $T = smtoep2$ con i campi

$$t : [n, m]$$

$$dim1 = [m_1, m_2]$$

$$dim2 = [n_1, n_2]$$

$$cev = []$$

Funzioni generiche	
`issmcirc2`	vero per smcirc2
`issmtoep2`	vero smtoep2
`smtcheck`	controlla l'installatione del toolbox
`smtconfig`	configura il toolbox
`smtgallery2`	matrici test

Tabella 3.3: Funzioni generiche e di calcolo

parametro	valore	descrizione
`display`	full/compact	definisce come sono visualizzate le variabili
`convreal`	true/false	converte l'uscita di alcune funzioni quando l'ingresso è reale
`convint`	true/false	converte in intero l'uscita di alcune funzioni quando l'ingresso è intero
`crautoeig`	true/false	calcola in automatico gli autovalori per matrici circolanti
`tpautoeig`	true/false	calcola in automatico gli autovalori della circolante, estensione della Toeplitz
`tpemb`	func. handle	restituisce la dimensione dell'embedding per matrici di Toeplitz

Tabella 3.4: Parametri configurati da `smtconfig`

matrici Circolanti	
`crrand2`	matrice casuale con distribuzione uniforme
`crrandn2`	matrice casuale con distr. unif. normalizzata
matrici di Toeplitz	
`gaussian2`	matrice Gaussiana
`tprand2`	matrice casuale con distribuzione uniforme
`tprandn2`	matrice casuale con distr. unif. normalizzata

Tabella 3.5: Matrici test contenute in `smtgallery2`

il campo cev contiene gli autovalori della matrice circolante estensione della Toeplitz, di default è settato a zero e si modifica tramite smtconfig. Per smtoep2 non sono state implementate funzioni come inv, det,eig,cond perchè non esiste un metodo standard per invertire una matrice di Toeplitz e calcolarne gli autovalori. Occorre altresì rimarcare che l'inversa di una matrice di Toeplitz non è di Toeplitz.

3.2 Test numerici

Veniamo al commento dei test effettuati per la verifica dell'utilità del codice proposto. Abbiamo testato il nostro software su matrici quadrate di dimensioni variabili crescenti e abbiamo misurato l'occupazione di memoria, la velocità della somma e quella dei prodotti tra matrice e vettore e tra matrici. Abbiamo inoltre monitorato la differenza di velocità tra il caso di embedding tight e nextpow2. Dai dati rilevati si nota dalla figura 1 che l'occupazione di memoria è molto migliore con l'`smt2` dovendo infatti meomrizzare molti meno termini, in una matrice quadrata circa n^2 anzichè n^4 e come si vede no-

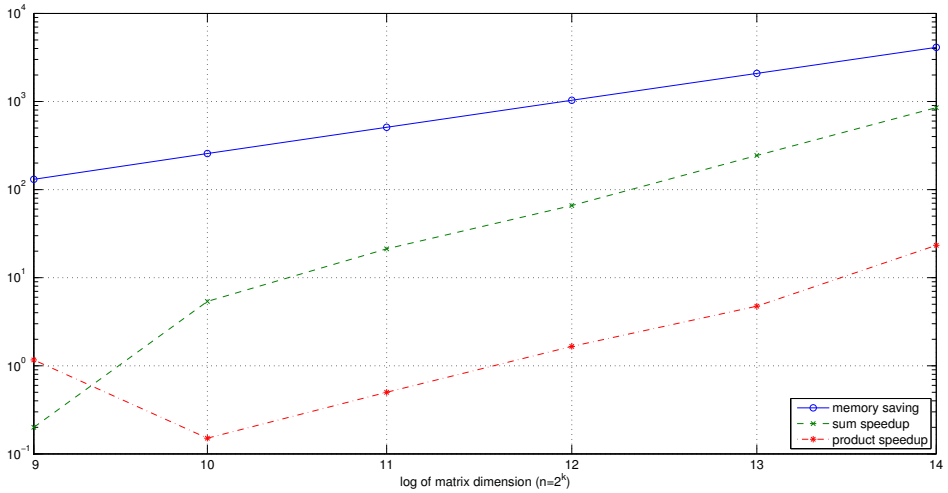

Figura 3.1: confronto full-`smt`.

nostante Matlab sia un programma veramente efficiente già per $n = 2^9$ si ha un rapporto 100:1 tra le due occupazioni di memoria. Per quanto riguarda la somma anche qui si vede che da subito, anche per matrici 'piccole', l'`smt2` è molto più veloce perchè ci limitiamo a sommare una colonna o nel caso delle Toeplitz una riga e una colonna quindi una complessità $O(n^2)$ anzichè n^4. Per quanto riguarda il prodotto abbiamo testato sia quello matrice-vettore che matrice-matrice per quanto riguarda il primo è stato testato anche senza la chiamata alla funzione per verificare se questa comportava un dispendio aggiuntivo. In realtà come si vede dalla figura 2 la chiamata alla funzione non crea nessuno svantaggio. Per matrici di dimensioni maggiori di $n = 2^{15}$ il toolbox `smt2` raggiunge risultati migliori di Matlab. La figura 3 mostra invece la differenza di velocità tra matrici Toeplitz con embedding tight e con embedding basato sulla più piccola potenza di $2 \geq n + m - 1$ ma contrariamente al caso monodimensionale non si presentano vantaggi sostanziali dal ricercare questo tipo di embedding. La tabella mostra i dati di un sistema di dimensioni variabili risolto con un metodo di iterazione. Come si vede dai dati, oltre $n = 2^7$ non ci è stato possibile risolvere il sistema per la matrice full. Per matrici di dimensioni fino a $n = 2^6$ è più veloce Matlab, poi `smt2` ed è interessante notare come `smt2` risolva un sistema con $n = 2^9$ in un tempo inferiore a quello che ci mette Matlab per un sistema con $n = 2^6$.

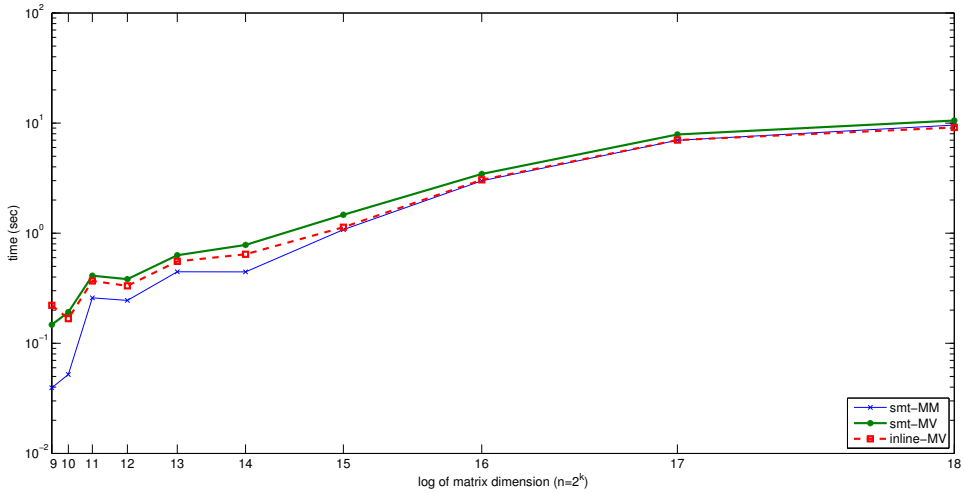

Figura 3.2: confronto prodotti mm e mv

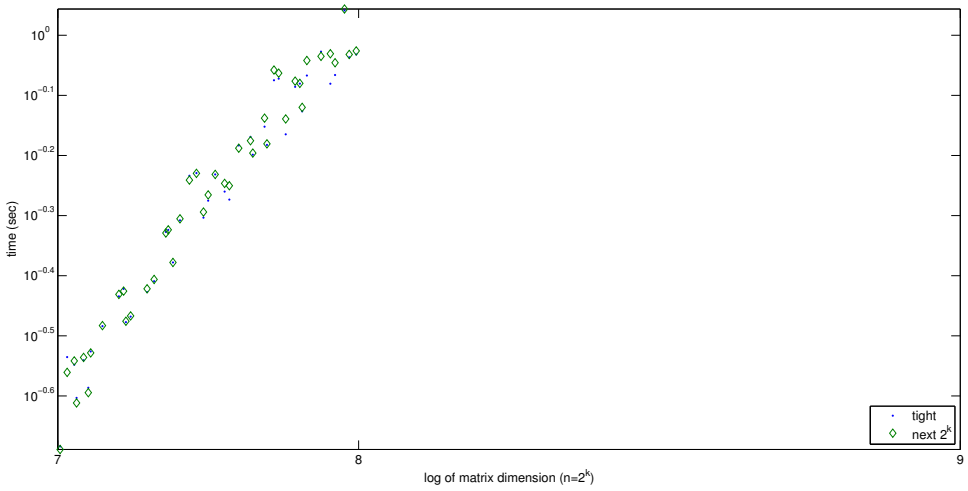

Figura 3.3: confronto prodotti mm e mv

$dim.$	$full$	smt
$2^{4.5}$	$0/0.04s$	$0/0.15s$
2^{5}	$100/0.08s$	$100/0.23s$
$2^{5.5}$	$100/0.28s$	$100/0.50s$
2^{6}	$100/1.13s$	$100/0.49s$
$2^{6.5}$	$100/3.87s$	$100/0.94s$
2^{7}	$100/16,54s$	$100/100s$
$2^{7.5}$	–	$100/1.98s$
2^{8}	–	$100/4.38s$
$2^{8.5}$	–	$100/9.25s$
2^{9}	–	$100/14.5s$

Tabella 3.6: confronto tempi Gauss-Gmres

3.3 Sviluppi futuri

- completamento funzioni non ancora implementate,alcune funzione per le Toelitz che non sono state ridefinite ad esempio l'impostazione del calcolo in automatico degli autovalori;
- introduzione dei precondizionatori per migliorare le routine di calcolo mediante metodi iterativi;
- eventuale estensione ad altre classi di matrici strutturate, tipo Hankel,Cauchy,etc.. per sfruttare la struttura di displacement e velocizzare i calcoli;
- estensione al caso n-dimensionale per il trattamento di matrici a 3 o più indici.

Capitolo 4

Un problema di elettromagnetismo

Una superficie selettiva in frequenza (FSS) può essere studiata come un array periodico bidimensionale di elementi metallici su di un substrato dielettrico o in alternativa di aperture su di uno schermo metallico che poggia su di un substrato metallico.

L'approccio "aperture–oriented" [2] si è mostrato particolarmente efficace nel trattare questo tipo di strutture, soprattutto quando si passa a considerare strutture multi–strato. La formulazione del problema elettromagnetico nel caso di una struttura FSS, come mostrata in Fig.4.1, consiste nel mettere in relazione le correnti superficiali sulla superficie FSS indotte dal campo incidente, tipicamente un'onda piana, con il campo riflesso dalla struttura, in particolare con il coefficiente di riflessione. Alla risonanza, la superficie si comporterà come una superficie ad alta impedenza e il coefficiente di riflessione avrà modulo unitario e fase nulla.

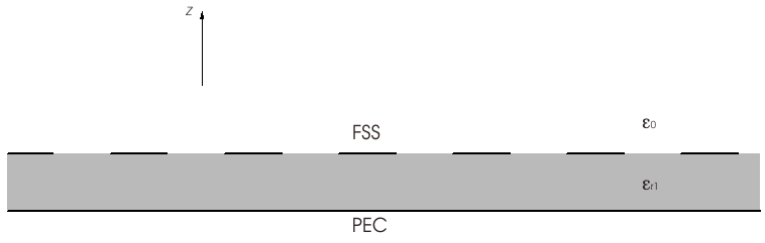

Figura 4.1: Sezione di una superficie FSS con substrato dielettrico e piano di massa.

Applicando il teorema di equivalenza [3] l'apertura è sostituita con conduttore elettrico perfetto (CEP) e vengono introdotte due correnti magneti-

che incognite dai lati opposti del piano metallico $\mathbf{M}$ e $-\mathbf{M}$,come mostrato in Fig.4.2, tali che venga garantita la continuità del campo magnetico tangente.

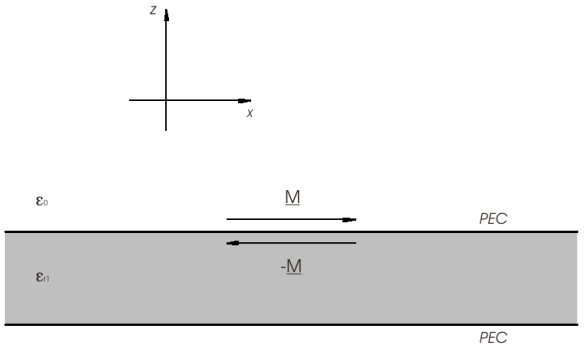

Figura 4.2: Teorema di equivalenza.

L'equazione integrale risultante, chiamata Magnetic Field Integral Equation (MFIE), viene risolta con il Metodo dei Momenti (MoM), una delle tecniche numeriche più conosciute nel campo dell'elettromagnetismo. L'incognita in questo caso è la corrente magnetica che viene espressa come combinazione lineare di funzioni base.

$$\mathbf{M} = \sum_{n=1}^{N_b} a_n \, \mathbf{f}_n \tag{4.1}$$

dove a_n sono i coefficienti incogniti e $\mathbf{f}_n$ le funzioni base.

La matrice del Metodo dei Momenti si ottiene imponendo in forma "debole" la continuità del campo magnetico tangente attraverso l'apertura:

$$\mathbf{H}_t^{(1)} = \mathbf{H}_t^{(2)} + \mathbf{H}_{inc,t} \tag{4.2}$$

ovvero viene selezionato un insieme di funzioni test e la condizione al contorno sul campo magnetico viene moltiplicata per ciasuna funzione test e il risultato integrato sul supporto della funzione test.

Nel nostro caso si sono scelte delle funzioni a sottodominio Rao–Wilton–Glisson (RWG) [10] per una discretizzazione triangolare. Le funzioni base RWG vengono definite su una coppia di triangoli adiacenti. Pertanto, nel nostro problema, considereremo tre tipi diversi di funzioni RWG, mostrate in Fig.4.3.

Seguendo il metodo Galerkin, le funzioni test sono le stesse funzioni base utilizzate per rappresentare le correnti magnetiche incognite sull'apertura. L'equazione precedente può essere quindi riscritta come:

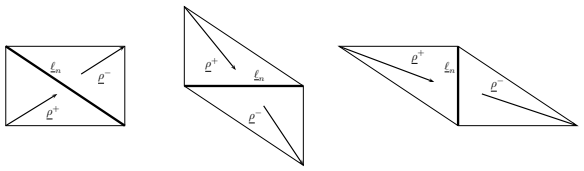

Figura 4.3: Funzioni base (RWG)

$$\int_S \mathbf{H}_t^{(1)} \cdot \mathbf{f}_m = \int_S \mathbf{H}_t^{(2)} \cdot \mathbf{f}_m + \int_S \mathbf{H}_{inc,t} \cdot \mathbf{f}_m, \qquad m = 1, \ldots, N_b \tag{4.3}$$

Considerando la cella unitaria, i campi su entrambi i lati della metallizzazione possono essere espressi intermini di funzione di Green nel dominio spettrale $\hat{\mathbf{G}}(u, v)$, che collega la densità di corrente magnetica trangente sull'apertura al campo magnetico trasverso:

$$\mathbf{H}_t^{unit\ cell}(x, y) = \langle \hat{\mathbf{G}}(u, v), \mathbf{M}(u, v) \rangle = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \hat{\mathbf{G}}(u, v) \cdot \hat{\mathbf{M}}(u, v)\, e^{-j(ux+vy)}\, du\, dv \tag{4.4}$$

dove $\mathbf{M}$ é la corrente magnetica sull'apertura della cella unitaria.

Possiamo scrivere il campo magnetico trasverso complessivo dell'array planare infinito come somma dell'eq.(4.4) sulle celle infinite della FSS e utilizzando la formula di Poisson [9],otteniamo la seguente espressione:

$$\mathbf{H}_t(x, y) = \frac{4\pi^2}{d_x d_y} \sum_r \sum_s \hat{\mathbf{G}}(u_r, v_s) \cdot \hat{\mathbf{M}}(u_r, v_s)\, e^{-j(u_r x + v_s y)} \tag{4.5}$$

dove la sommatoria è fatta sui modi di Floquet, e le componenti trasverse del vettore di propagazione hanno ora i valori discretizzati:

$$u_r = \left(\frac{2\pi r}{d_x} + k_0 \sin\theta \cos\phi \right), \qquad v_s = \left(\frac{2\pi s}{d_y} + k_0 \sin\theta \sin\phi \right) \tag{4.6}$$

L'eq.(4.5) può essere utilizzata per entrambi i campi sopra e sotto la metallizzazione, utilizzando la funzione di Green pertinente.

La periodicità della struttura consente di imporre la continuità del campo magnetico sulla singola cella. Pertanto, utilizzando come funzioni test lo stesso insieme di funzioni base $\mathbf{f}_n$ (Galerkin method), otteniamo la matrice MoM matrix come:

$$\frac{4\pi^2}{d_x d_y}\sum_{n=1}^{N_b} a_n \sum_r \sum_s \hat{\mathbf{G}}_{tot}(u_r, v_s)\cdot\hat{\mathbf{f}}_n(u_r, v_s)\cdot\hat{\mathbf{f}}_m^*(u_r, v_s) = \\ -\int_S \mathbf{H}_{inc}\cdot\mathbf{f}_m(x,y)\,dS \qquad (4.7)$$

per $m = 1, \ldots, N_b$. In eq.(4.7) la funzione di Green complessiva è la somma di due termini, uno per ciascuna regione:

$$\hat{\mathbf{G}}_{tot}(u_r, v_s) = \hat{\mathbf{G}}_1(u_r, v_s) + \hat{\mathbf{G}}_2(u_r, v_s) \qquad (4.8)$$

dove $\hat{\mathbf{G}}_1(u_r, v_s)$ e $\hat{\mathbf{G}}_2(u_r, v_s)$ sono le funzioni di Green rispettivamente nella regione sotto e sopra la metallizzazione, che possono essere computate una indipendentemente dall'altra utilizzando il circuito equivalente mostrato in Fig.4.4.

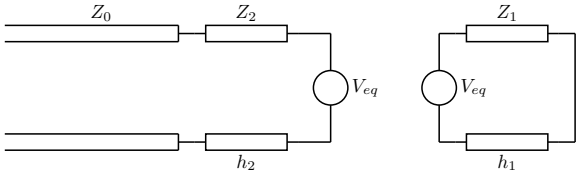

Figura 4.4: Circuito equivalente.

Nel caso più generale si considera la presenza di un substrato dielettrico di copertura di spessore h_2. Il principale vantaggio di questo metodo risiede proprio nel disaccoppiamento tra le due, regioni sopra e sotto la metallizzazione.

La funzione di Green rappresenta il campo magnetico dovuto a un dipolo magnetico unitario infinitesimo posto all'interfaccia aria–substrato, in $z = 0$. L'espressione che si ottiene è la seguente:

$$\hat{G}^{xx} = \left(\frac{u^2}{Z_{in}^{TE}} + \frac{v^2}{Z_{in}^{TM}}\right)\frac{1}{u^2+v^2}, \qquad \hat{G}^{yy} = \left(\frac{v^2}{Z_{in}^{TE}} + \frac{u^2}{Z_{in}^{TM}}\right)\frac{1}{u^2+v^2}, \\ \hat{G}^{xy} = \hat{G}^{yx} = \left(\frac{1}{Z_{in}^{TE}} - \frac{1}{Z_{in}^{TM}}\right)\frac{uv}{u^2+v^2} \qquad (4.9)$$

dove, considerando il caso $h_2 = 0$, ovvero assenza di substrato superiore, abbiamo per la regione 1 (sotto la metallizzazione):

$$Z_{in}^{(1)} = j\,Z_1 \tan\beta_1 h_1 \qquad (4.10)$$

e per la regione 2 (sopra la metallizzazione):

$$Z_{in}^{(2)} = Z_0 \tag{4.11}$$

e dove per ciascun caso(TE or TM) dobbiamo utilizzare l'espressione corretta per l'impedenza (i=0 aria, i=1 substrato):

$$Z_i^{TE} = \frac{\omega\mu}{w_i}\,, \qquad\qquad Z_i^{TM} = \frac{w_i}{\omega\varepsilon_{r_i}\varepsilon_0} \tag{4.12}$$

Si può notare dall'equazione 7 che essendo un numero $f_m * f_n^*$ si può mettere in evidenza per ogni m,n e fissati u e v ciò comporta che sommando le componenti della funzione di Green separatamente per ogni tipo di superficie otteniamo una dipendenza da $(u-v)^2$ e questo giustifica il fatto che la matrice dei momenti risultante sia oltre che Toeplitz vista la dipendenza da (u-v), anche simmetrica. Abbiamo analizzato i dati ricevuti da questo tipo di analisi e abbiamo ri-indicizzato i bordi in modo da sfruttare la struttura multidimensionale, abbiamo prima di tutto trasformato il vettore in input in una matrice quadrata(questo in quanto anche il foro era quadrato e i passi di discretizzazione uguali), dopo di ciò abbiamo rilevato la ripetizione di alcuni elementi in posizioni casuali, facendo una verifica sull'interazione tra quali bordi producesse ogni singolo valore abbiamo trovato conferma del fatto che i valori dipendevano oltre che dal tipo di bordi dalla posizione reciproca. Per mettere in evidenza tutte queste relazioni abbiamo prima di tutto analizzato una matrice di piccole dimensioni 40x40, dalla geometria della struttura abbiamo numerato i bordi per tipo come diagonali, verticali e orizzontali, nella struttura di partenza erano presenti 4x4 bordi diagonali, 4x3 bordi verticali e 3x4 bordi orizzontali.

Quindi ora nel primo blocco 16x16 avevamo le interazioni tra bordi dello stesso tipo e sulla diagonale principale gli autotermini, i termini sulle diagonali erano uguali in quanto il valore dell'interazione dipendeva dalla posizione reciproca e la matrice era inoltre simmetrica, anche se il fatto non era sfruttabile perchè non esistono algoritmi specifici per matrici complesse simmetriche. Nel secondo e terzo blocco di dimensioni 16×12 erano presenti le interazioni tra bordi di tipo 1 e rispettivamente bordi di tipo 2 e 3, anche questi blocchi in quanto i valori dipendevano dalle posizioni reciproche erano Toeplitz. Ovviamente l'interazione tra un bordo e un altro era uguale all'interazione di quest'ultimo rispetto al primo quindi i blocchi 2 e 3 si ripetevano trasposti al di sotto del blocco 1. Lo stesso discorso appena fatto era evidentemente valido per i bordi di tipo 2 e 3 e per l'interazione tra bordi di tipo 2 e 3. Alla fine si sono ottenuti 3 blocchi quadrati uno 16×16 e 2 12×12 posizionati in diagonale e 2 blocchi rettangolari 16×12 e 1 blocco quadrato

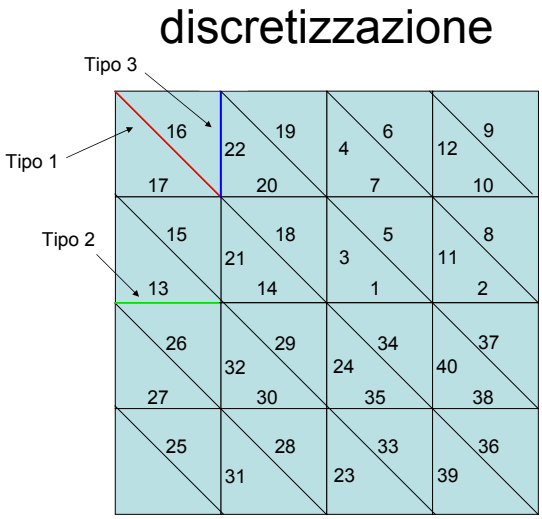

Figura 4.5: indicizzazione precedente.

12×12 che si ripetono trasposti. Vediamo come questi blocchi erano a loro volta strutturati. Partiamo dai blocchi quadrati in diagonale, iniziamo dal primo ma il discorso si può poi generalizzare. Si è notato che era presente una struttura a blocchi che si ripeteva e in particolare si trattava di una matrice a blocchi con blocchi di Toeplitz che vediamo in dettaglio:

$$K = \begin{bmatrix} A & E & F & G \\ B & A & E & F \\ C & B & A & E \\ D & C & B & A \end{bmatrix}$$

Tutti i blocchi erano di dimensione 4×4 e solo 7 blocchi erano indipendenti mentre gli altri si ripetevano essendo costanti le diagonali, per memorizzare una matrice Toeplitz $m \times n$ erano necessari $m + n - 1$ elementi e quindi in questo caso 7 per ogni singolo blocco e 49 in totale,e non 256. Consideriamo ora i due blocchi quadrati 12×12 questi sono costituiti in un caso da 3×3 matrici toeplitz 4×4 e nell'altro da 4×4 matrici toeplitz 3×3 Si avrà quindi nel primo caso:

$$L = \begin{bmatrix} H & O & P \\ I & H & O \\ M & I & H \end{bmatrix}$$

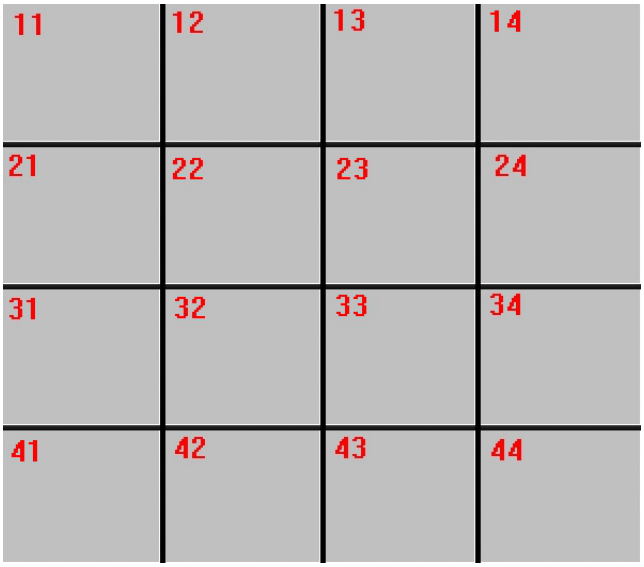

Figura 4.6: nuova indicizzazione bordi diagonali.

Tutti i blocchi sono Toeplitz di dimensioni 4×4, quindi per ciascuna di esse ho bisogno di 7 elementi,e i blocchi in totale sono 9 di cui 5 indipendenti quindi ho bisogno di memorizzare 35 elementi anzichè 144. Nel secondo caso si avrà:

$$J = \begin{bmatrix} S & W & X & Y \\ T & S & W & X \\ U & T & S & W \\ V & U & T & S \end{bmatrix}$$

Tutti i blocchi sono Toeplitz di dimensioni 3×3 e quindi necessitano di 5 elementi per essere memorizzati, sono presenti 9 blocchi di cui 5 indipendenti quindi ho bisogno di conoscere 35 elementi anzichè 144. Occupiamoci ora delle matrici rettangolari 16×12, anche queste sono Toeplitz, inoltre due delle quattro sono uguali alle trasposte delle altre due(pur non essendo sfruttabile questo fatto). Le due matrici indipendenti possono essere viste una come 4×3 blocchi 4×4 e l'altra come 4×4 blocchi 4×3 quindi in un caso ho bisogno di conoscere 7 blocchi e di ogni blocco 6 elementi e nell'altro di conoscere 6 blocchi e di ogni blocco 7 elementi vediamo in dettaglio come si presentano i 2 blocchi:

$$Y1 = \begin{bmatrix} S1 & T1 & U1 & Z1 \\ W1 & S1 & T1 & U1 \\ J1 & W1 & S1 & T1 \\ M1 & J1 & W1 & S1 \end{bmatrix}$$

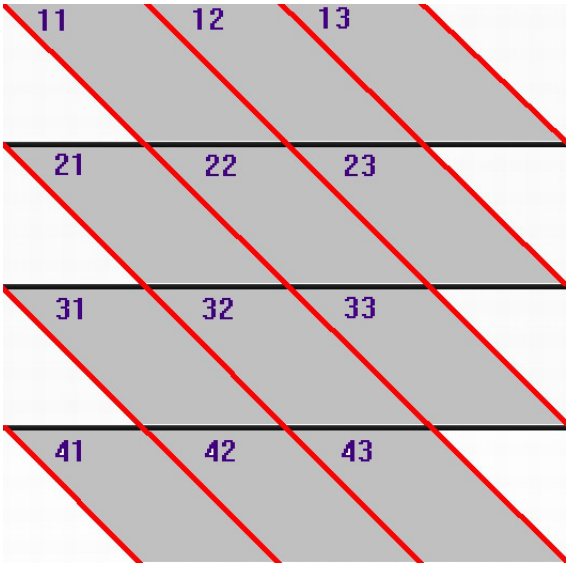

Figura 4.7: nuova indicizzazione bordi verticali.

e

$$E1 = \begin{bmatrix} A1 & O1 & I1 \\ B1 & A1 & O1 \\ C1 & B1 & A1 \\ D1 & C1 & B1 \end{bmatrix}$$

Per memorizzare i blocchi Y ed E sono necessari quindi 42 elementi ciascuno e 168 in totale. Rimane ora da analizzare il blocco quadrato dato dall'interazione tra i bordi di tipo '2' e quelli di tipo '3' in realtà questo blocco si compone di 3×4 blocchi toeplitz di dimensioni 4×3 ha pertanto bisogno di conoscere 6 blocchi indipendenti e 6 elementi per ciascun blocco, quindi 72 elementi in totale considerando quelli per il blocco trasposto. Abbiamo quindi memorizzato 359 elementi (49 +2*35 + 2*36 + 4*42) anzichè 1600. Vediamo ora i vantaggi di calcolo conseguenti dalla struttura Toeplitz. In realtà se teniamo conto che le matrici indipendenti sono solo 6 visto che 3 sono trasposizioni, in particolare il blocco 2 è il trasposto del blocco 4, il 3 di quello 7 e il 6 di quello 8, serve memorizzare solo 239 elementi. Il sistema può essere spezzato nella risoluzione di 9 sottosistemi Toeplitz, vediamo in dettaglio come: Dopo aver ri-indicizzato gli elementi della matrice dobbiamo fare la stessa cosa col vettore di termini noti che da vettore monoindice viene trasformato in una concatenazione di vettori bi-indice, in particolare nel nostro caso otteniamo: un vettore bi-indice 4×4, uno 3×4 e uno 4×3. Per sfruttare appieno la struttura trasformiamo le matrici di Toeplitz in matrici Circolanti,questo comporta una da un lato un leggero appesantimento dell'occupazione di me-

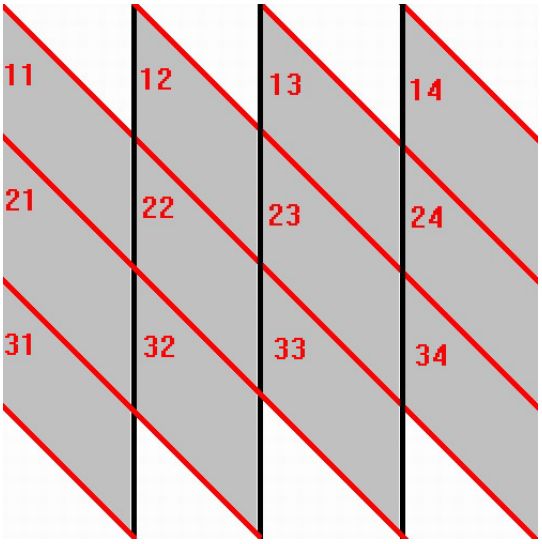

Figura 4.8: nuova indicizzazione bordi orizzontali.

moria in quanto la matrice, nel caso non embedded, diventa di dimensioni $[m1+n1-1, m2+n2-1] \times [m1+n1-1, m2+n2-1]$ ma si può sfruttare la velocità di calcolo della FFT2. Ora la complessità computazionale è per ogni singolo sistema $O(k_1 k_2 log_2(k_1 k_2)$ dove k_1, k_2 sono le dimensioni dello gnomone della matrice circolante e cioè $k_{1i} = m_{1i} + n_{1i} - 1, k_{2i} = m_{2i} + n_{2i} - 1$, se indichiamo con k_{1max} e k_{2max} le maggiore dimensioni tra i k_{1i}, k_{2i} e tenuto conto che abbiamo 9 sistemi la complessità è $<= 9 \cdot O(k_{1max} k_{2max} log(k_{1max} k_{2max}))$ questo a fronte di una complessità $\frac{1}{3}O(n^3)$ se si risolve il sistema con Gauss e tenendo conto che visto che esistono 3 tipologie di bordi le dimensioni delle singole sottomatrici sono circa $[\frac{n}{3}\frac{n}{3}]$ e che le dimensioni dei sottoblocchi Toeplitz sono circa $[\sqrt{\frac{n}{3}}\sqrt{\frac{n}{3}}]$ per k crescente si ha una notevole riduzione della complessità. Scriviamo il nostro sistema nella forma compatta $T = b$ dove indichiamo:

$$T = \begin{bmatrix} A & B^T & C^T \\ B & D & E^T \\ C & E & F \end{bmatrix}$$

Ora la nostra matrice A è una Toeplitz bi-indice di dimensioni $[4,4] \times [4,4]$, la B $[4,3] \times [4,4]$,la C $[3,4] \times [4,4]$, la D $[4,3] \times [4,3]$, la E $[4,3] \times [3,4]$, la F $[3,4] \times [3,4]$.

$$x = \begin{bmatrix} X1 \\ X2 \\ X3 \end{bmatrix}$$

dove

$$X1 = \begin{bmatrix} x_1 & x_5 & x_9 & x_{13} \\ x_2 & x_6 & x_{10} & x_{14} \\ x_3 & x_7 & x_{11} & x_{15} \\ x_4 & x_8 & x_{12} & x_{16} \end{bmatrix}$$

dove

$$X2 = \begin{bmatrix} x_{17} & x_{21} & x_{25} \\ x_{18} & x_{22} & x_{26} \\ x_{19} & x_{23} & x_{27} \\ x_{20} & x_{24} & x_{28} \end{bmatrix}$$

dove

$$X3 = \begin{bmatrix} x_{29} & x_{32} & x_{35} & x_{38} \\ x_{30} & x_{33} & x_{36} & x_{39} \\ x_{31} & x_{34} & x_{37} & x_{40} \end{bmatrix}$$

e

$$b = \begin{bmatrix} B1 \\ B2 \\ B3 \end{bmatrix}$$

dove

$$B1 = \begin{bmatrix} b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \\ b_4 & b_8 & b_{12} & b_{16} \end{bmatrix}$$

dove

$$B2 = \begin{bmatrix} b_{17} & b_{21} & b_{25} \\ b_{18} & b_{22} & b_{26} \\ b_{19} & b_{23} & b_{27} \\ b_{20} & b_{24} & b_{28} \end{bmatrix}$$

dove

$$B3 = \begin{bmatrix} b_{29} & b_{32} & b_{35}b_{38} \\ b_{30} & b_{33} & b_{36}b_{39} \\ b_{31} & b_{34} & b_{37}b_{40} \end{bmatrix}$$

questo ovviamente dopo il reshape iniziale. Dopo l'embedding(tight o meno) occorre modificare allo stesso modo le matrici B1, B2 e B3 queste saranno moltiplicate per le inverse delle matrici circolanti multi-indice e si otterranno le corrispondenti soluzioni X1, X2, X3 dopo aver sommato i 3 contributi e aver tenuto conto in maniera inversa dell'embedding.

4.1 Risoluzione numerica

In realtà abbiamo risolto il problema mediante un metodo iterativo. La struttura della matrice era a blocchi con blocchi che a loro volta erano matrici di Toeplitz multi-indice, quindi abbiamo dovuto ricostruire la matrice iniziale in modo da evidenziare la struttura presente e abbiamo creato una funzione che ricevendo in ingresso il sistema come insieme di matrici Toeplitz multi-indice e un vettore ne facesse il prodotto e restituisse il vettore soluzione in uscita. Essendo la struttura non hermitiana non è stato possibile utilizzare il metodo del gradiente coniugato, abbiamo quindi usato Gmres. Abbiamo dato in ingresso il nostro insieme di matrici strutturate e il vettore termini noti e poi confrontato la soluzione con quella ottenuta mediante metodo di Gauss. Abbiamo impostato un errore di 10^{-15} e un numero massimo di 40 iterazioni. La soluzione è andata a convegenza dopo 35 iterazioni. Abbiamo poi simulato matrici di dimensioni molto maggiori con dati compatibili con quelli rilevati, per testare la bontà del codice, in particolare abbiamo usato matrici di dimensioni crescenti con numero di condizionamento intorno a 10^3 e impostato un errore di approssimazione di 10^{-4} e 100 come numero massimo di iterazioni. Si nota dalle rilevazioni come per numeri di condizionamento della matrice full intorno a 10^3 si arrivi rapidamente a convergenza ed in un tempo inferiore alla risoluzione del sistema full con metodo di Gauss. Ovviamente per matrici di dimensioni troppo gradi ad esempio con $m = 10000$ non è possibile nemmeno memorizzare la matrice full. Dalla figura 1 possiamo notare come fino ad $n = 35$ che corrisponde a una dimensione $m = 3n^2 - 2n = 4970$ è più veloce il metodo di Gauss,dopo diventa più veloce Gmres. Abbiamo bloccato la memorizzazione della matrice full per $m = 10000$ e così pure lo svolgimento col metodo di Gauss. Per quanto riguarda l'errore ovviamente col metodo di Gauss è migliore ma è stata scelta una tolleranza accettabile.

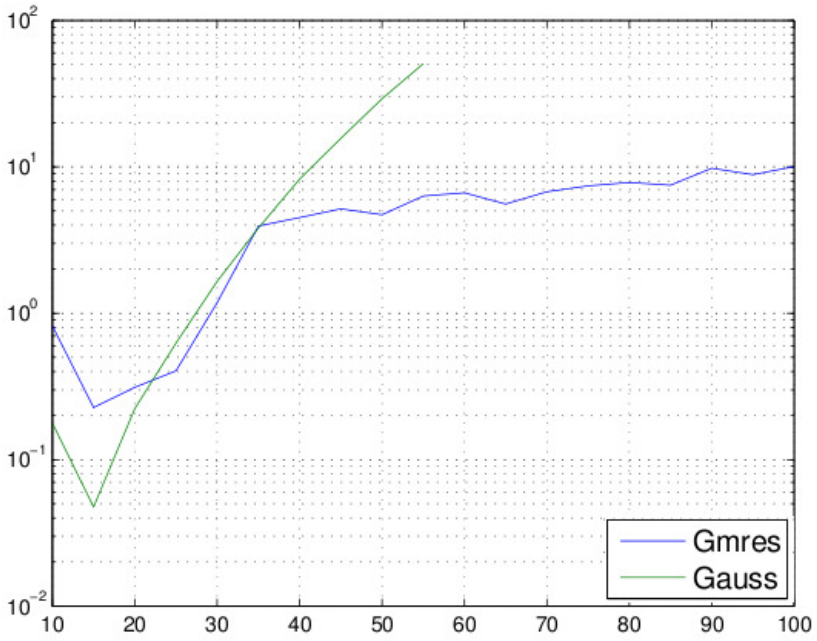

Figura 4.9: confronto tempi Gmres-Gauss

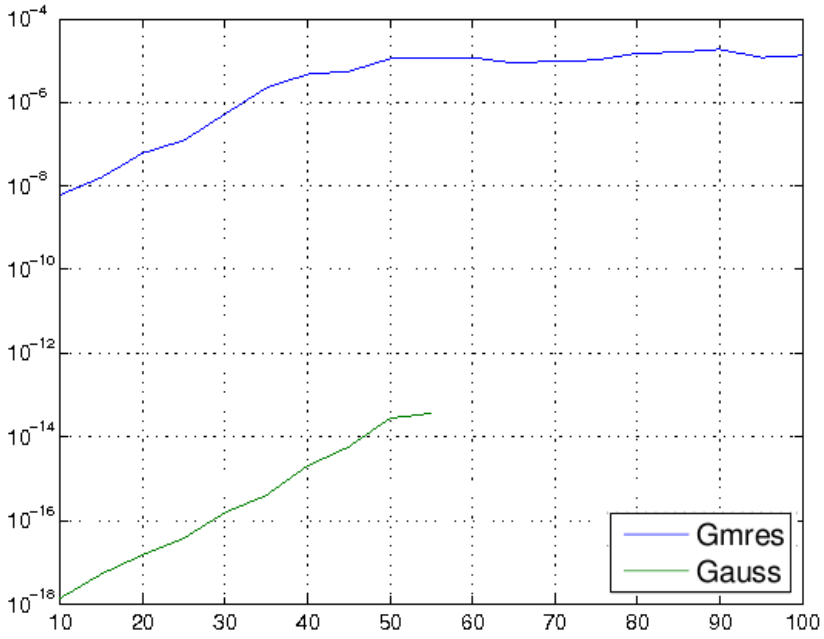

Figura 4.10: confronto errori Gmres-Gauss

Ringraziamenti

Ringrazio sentitamente il professor Giuseppe Rodriguez per la pazienza, la competenza e la disponibiltà nonchè l'indispensabile aiuto nello sviluppo del toolbox, ringrazio altresì il professor Giuseppe Mazzarella e sopratutto la dottoressa Luisa Deias per i dati e le spiegazioni tecniche fornite per uno sviluppo applicativo del programma riguardante un problema di antenne da loro studiato.

Bibliografia

[1] A. Aricò and G. Rodriguez. A fast solver for linear systems with displacement structure. *Numer. Algorithms*, 55(4):529–556, 2010. DOI: 10.1007/s11075-010-9421-x.

[2] F Asole, L Deias, and G Mazzarella. A flexible full-wave analysis of multilayered amc using an aperture oriented approach. *Journal of Electromagnetic Waves and Applications*, 21(14):2059–2072, 2007.

[3] Robert E Collin. *Field theory of guided waves*, volume 2. IEEE press New York, 1991.

[4] P. J. Davis. *Circulant Matrices*. Wiley, New York, 1979.

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

[6] Woods R. E. Gonzalez R. C. *Digital Image Processing*. Addison-Wesley Pubblishing Company, New York, 1993.

[7] T. Kailath and A. H. Sayed, editors. *Fast Reliable Algorithms for Matrices with Structure*. SIAM, Philadelphia, 1999.

[8] The MathWorks, Natick. *Matlab ver. 7.7*, 2008.

[9] D Pozar and D Schaubert. Scan blindness in infinite phased arrays of printed dipoles. *Antennas and Propagation, IEEE Transactions on*, 32(6):602–610, 1984.

[10] Sadasiva Rao, D Wilton, and Allen Glisson. Electromagnetic scattering by surfaces of arbitrary shape. *Antennas and Propagation, IEEE Transactions on*, 30(3):409–418, 1982.

[11] M. Redivo-Zaglia and G. Rodriguez. `smt`: a Matlab toolbox for structured matrices. *Numer. Algorithms*, 59(4):639–659, 2012. DOI: 10.1007/s11075-011-9527-9.

[12] G. Rodriguez. *Algoritmi Numerici*. Pitagora Editrice, Bologna, 2008.

[13] Zahrt John D Wing G Milton. *A primer on integral equations of the first kind: the problem of deconvolution and unfolding*. Siam, New York, 1991.