Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

MEDITOMO: un software parallelo per la ricostruzione di dati 3D SPECT

L. Antonelli - L. Carracciuolo - L. D'Amore - A. Murli

RT-ICAR-NA-06-03 **03-2006**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) – Sede di Napoli, Via P. Castellino 111, I-80131 Napoli, Tel: +39-0816139508, Fax: +39-0816139531, e-mail: napoli@icar.cnr.it, URL: *www.na.icar.cnr.it*

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

MEDITOMO: un software parallelo per la ricostruzione di dati 3D SPECT [1]

L. Antonelli[2] - L. Carracciuolo[2] - L. D'Amore[3] - A. Murli[3]

Rapporto Tecnico N.: **RT-ICAR-NA-06-03**

Data: **03-2006**

---

1 Il lavoro descritto è stato condotto nell'ambito di Progetti PRIN (2000-2002;2002-2004 e 2004-2006) e nell'ambito del Progetto FIRB 2003-2005

[2] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sede di Napoli, Via P. Castellino 111, 80131 Napoli
[3] Dipartimento di Matematica ed Applicazioni, Facoltà di Scienze MMFFNN, Università degli Studi di Napoli, Federico II

MEDITOMO*: un software parallelo per la ricostruzione di dati 3D SPECT

L. Antonelli[2] - L. Carracciuolo[2] - L. D'Amore[1] - A. Murli[1]

1. Dipartimento di Matematica ed Applicazioni, Facoltà di Scienze MMFFNN, Università degli Studi di Napoli, *Federico II*

2. Istituto di Calcolo e Reti ad Alte Prestazioni del Consiglio Nazionale delle Ricerche (ICAR-CNR)

Sommario

In questo lavoro sono descritte le problematiche affrontate sullo sviluppo del software, denominato MEDITOMO, per la ricostruzione di immagini MEDIche provenienti da esami TOMOgrafici di tipo SPECT (*Single Photon Emission Computed Tomography*) in ambiente di calcolo HEC (High End Computing). Tale attività è stata svolta in collaborazione con il Dipartimento di Fisiopatologia Clinica dell'Università di Firenze, il Dipartimento di Fisica dell'Università di Genova e la Sede di Napoli dell'Istituto per il Calcolo e Reti ad Alte Prestazioni del Consiglio Nazionale delle Ricerche. Il contributo principale risiede sia nell'impiego del funzionale di Totale Variazione come funzionale di regolarizzazione *edge preserving*, per la risoluzione del problema inverso che descrive la ricostruzione di dati SPECT, sia nella progettazione ed implementazione di software parallelo effettivamente utilizzato in applicazioni concrete.

1 Introduzione

La SPECT (*Single Photon Emission Computed Tomography*) è una tecnica di indagine medica di tipo tomografia funzionale. Essa permette di ricavare e caratterizzare le funzioni vitali di organi come il cuore ed il cervello, conoscendo la distribuzione in tale organo, della concentrazione di un tracciante radioattivo iniettato nel corpo del paziente. Un'analisi di dati SPECT, consiste nella ricostruzione (del volume) della concentrazione del radiofarmaco attraverso un certo numero di immagini acquisite dalla "gamma camera", che rappresentano la (proiezione 2D della) distribuzione (3D) del tracciante radioattivo.

*Il lavoro descritto è stato condotto nell'ambito di Progetti PRIN (2000-2002; 2002-2004 e 2004-2006) e nell'ambito del Progetto FIRB 2003-2005

Il problema rientra dunque nella classe dei problemi inversi. In particolare, essendo descritto da un'equazione integrale di Fredholm di prima specie, è un problema inverso mal posto [28]. L'obiettivo è fornire soluzioni al problema che siano sufficientemente affidabili e soprattutto calcolarle in un tempo utile([1]) e praticabile nella diagnostica medica.
In tale documento si discutono gli aspetti computazionali legati alla risoluzione del problema in ambiente di calcolo ad alte prestazioni, dalla formulazione matematica del problema, alla scelta di un metodo numerico affidabile, allo sviluppo dell'algoritmo fino alla sua implementazione in ambiente di calcolo parallelo.

Il lavoro è organizzato come segue:

- nel paragrafo 2 è illustrata la formulazione matematica del problema di ricostruzione di dati SPECT. Sono presentati sia i due approcci standard di risoluzione al problema in esame: il metodo dei minimi quadrati ed il metodo della massima verosimiglianza, sia la regolarizzazione con la Totale Variazione (TV);
- nel paragrafo 3 si illustra la discretizzazione del problema;
- nel paragrafo 4 sono presentati gli algoritmi di ricostruzione: *Punto Fisso*, *Gradiente Coniugato* e *Expectation Maximization*;
- nel paragrafo 5 è illustrata un'analisi della complessità computazionale degli algoritmi presentati nel paragrafo 4;
- nel paragrafo 6 è descritta l'introduzione del parallelismo negli algoritmi presentati nel paragrafo 4;
- nel paragrafo 7 sono illustrate le caratteristiche del software parallelo MEDITOMO;
- nel paragrafo 8 sono presentati alcuni risultati numerici.

2 La ricostruzione di dati SPECT

Sia $\{x, y, z\}$ un sistema di assi cartesiani, dove x ed y sono le coordinate del piano della sezione bidimensionale dell'immagine dell'organo in esame e z è l'asse di rotazione della gamma camera intorno all'organo. Si supponga che l'origine del sistema di assi cartesiani coincida con il centro della sezione piana. Si indichi con $g(s, \phi, z)$, la funzione che descrive l'immagine acquisita dalla gamma camera all'angolo di rotazione ϕ. La variabile s indica la distanza del centro dell'immagine piana dal sistema di acquisizione, e la funzione $u(x, y, z)$ è la distribuzione (3D) del radiofarmaco. Il modello matematico del sistema di acquisizione di dati SPECT è descritto dalla *trasformata di Radon sfocata*[12]:

[1] Con l'espressione "tempo utile", si intende un tempo compatibile con quello della diagnostica medica, cioè pochi minuti ed in generale il minimo possibile.

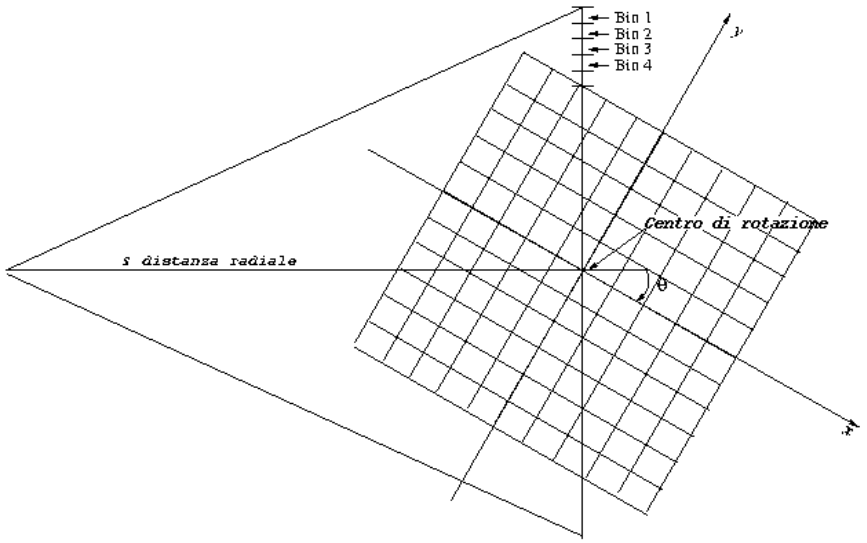


Figura 1: *Schematizzazione del processo di acquisizione delle proiezioni bidimensionali della distribuzione del tracciante radioattivo.*

$$Ku(\theta, s) := \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} u(\mathbf{x}, z')h(s-\mathbf{x}\cdot\theta, \mathbf{x}\cdot\theta^{\perp}, z-z')d\mathbf{x}dz' = g(s, \phi, z) \quad (1)$$

dove $\mathbf{x} = \{x, y\}$, $\theta = \{cos\phi, sin\phi\}$ e $\theta^{\perp} = \{-sin\phi, cos\phi\}$. $h(s, t, z)$ è la funzione che descrive il processo di *sfocatura* (*blurring*), ovvero la *Point Spread Function* (PSF).

Seguendo il modello proposto in [12], si è assunto:

$$h(s, t, z) = h_0(s, t)h_0(z, t) \quad ,$$

dove:

$$h_0(a, b) = \frac{1}{\sigma(t)\sqrt{2\pi}} e^{-\frac{a^2}{2\sigma^2(b)}} \quad ,$$

e

$$\sigma = \sigma_{min} - \frac{\sigma_{max} - \sigma_{min}}{2r}(t - r) \quad , \qquad (2)$$

σ_{min}, σ_{max} ed r sono costanti dipendenti dal sistema di acquisizione([2]).
Il modello (1) è detto *fully 3D*. Se si assume che la PSF sia separabile rispetto

[2]Per la scelta dei valori tipici attribuiti a σ_{min}, σ_{max} ed r, si rimanda alla sezione 8.

a (s,t) e z, cioè:

$$h(s,t,z) = E(s,t)A(z) \quad , \tag{3}$$

ovvero che l'effetto di *blurring* è descritto dalla funzione $E(s,t)$ lungo i piani perpendicolari a z, e da $A(z)$ lungo l'asse z, si ha:

$$Ku(\theta,s) := \int_{-\infty}^{\infty} A(z-z')dz' \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} u(\mathrm{x},z')h(s-\mathrm{x}\cdot\theta,\mathrm{x}\cdot\theta^{\perp})d\mathrm{x} = g(s,\phi,z) \quad . \tag{4}$$

Il modello matematico (4) è detto *2D+1*, per evidenziare che il contributo lungo l'asse z è separato da quello lungo il piano xy [12].

(1) e (4) si possono scrivere come:

$$g = Ku + \eta \Longleftrightarrow \tilde{g} = g - \eta = Ku \quad , \tag{5}$$

dove K è l'operatore integrale, ed η è l'inevitabile contributo del rumore nella fase di acquisizione dei dati. Per semplicità di notazione da questo momento in poi si indicherà $\tilde{g}$ con g, ovvero g indica il dato acquisito dalla gamma camera, ed è quindi perturbato.

Il *Problema Inverso*:

$$u = K^{-1}g \quad , \tag{6}$$

è mal posto [10]. In letteratura, i metodi proposti per il calcolo di una soluzione regolarizzata u^* della (6), sono essenzialmente basati su tre tipi di approcci. Il più tradizionale, è la *retroproiezione filtrata* (*filtered back projection* (FBP)). L'idea è trasformare il problema (5) nel dominio delle frequenze, ovvero applicare l'operatore Trasformata di Fourier ad ambo i membri della (5), determinare la soluzione e antitrasformare il risultato. Questo approccio ha un legame con il funzionale di regolarizzazione introdotto da Wiener [21].
Il secondo e terzo approccio sono metodi classici di regolarizzazione, in cui la soluzione è calcolata come minimo o massimo di un opportuno funzionale, definito in base al tipo di assunzioni sulla distribuzione del rumore η.

Nei paragrafi che seguono si illustrano gli ultimi due approcci, sulla base dei quali è stato sviluppata la libreria MEDITOMO.

2.1 Il metodo dei minimi quadrati

Nell'ipotesi in cui il rumore sui dati sia distribuito uniformemente secondo una distribuzione gaussiana, la soluzione u del problema (6) è calcolata come la soluzione di un problema di migliore approssimazione nel senzo dei minimi quadrati. Sia u^*:

$$u^* = arg\ \min_u F(u) = arg\ \min_u \{\parallel Ku - g \parallel_2\} \quad , \tag{7}$$

dove $\| \cdot \|_2$ indica la norma in $L^2(\Omega)$, $\Omega \subset \Re^3$ è un aperto di $\Re^3$ in cui sono definite u e g. Il minimo u^* del funzionale (7), è soluzione del seguente problema:

$$\begin{cases} K^*(Ku - g) = 0 & , \quad (x, y) \in \Omega \\ \frac{\partial u}{\partial n} = 0 & , \quad (x, y) \in \partial\Omega \end{cases} \tag{8}$$

le (8) sono dette equazioni di Eulero-Lagrange.

Il funzionale (7) introduce una forma di *regolarizzazione* nel problema (6). In generale, l'idea di base della regolarizzazione è quella di sostituire il problema (6), con il seguente:

$$u^* = arg \min_u F(u) = arg \min_u \{\| Ku - g \|_2 + \alpha J(u)\} \quad , \tag{9}$$

dove $J(u)$ è l'*operatore di regolarizzazione* che consente di caratterizzare u^*, includendo informazioni a priori sulla soluzione, (come la grandezza e gli *edge*), α è il *parametro di regolarizzazione*.

2.2 Il metodo della massima verosimiglianza

Nell'ipotesi in cui il rumore sui dati sia distribuito seconodo una distribuzione di Poisson, la soluzione del problema (6) è calcolata come soluzione di un problema di massima verosimiglianza.
Sia u^*:

$$u^* = \arg\max_u p\,(u|g) \quad , \tag{10}$$

dove $p\,(u|g)$ indica la probabilità condizionata di u noto g [26].
La ricerca del massimo di $p\,(u|g)$ è equivalente alla ricerca del massimo di $\log p\,(u|g)$. Applicando il teorema di Bayes([3]), il problema (10) è equivalente a:

$$u^* = \arg\max_u \{\log p\,(g|u) + \log p\,(u)\} \quad . \tag{12}$$

La riformulazione (12) del problema (10) permette di introdurre il termine $p\,(u)$. Tale termine svolge lo stesso ruolo del termine di regolarizzazione $J(u)$ della formulazione (9).

2.3 La regolarizzazione mediante Totale Variazione

Sia nella (9) che nella (12) si è scelto funzionale di regolarizzazione la Totale Variazione (TV) [25].

[3]Si ricorda che per il teorema di Bayes si ha:

$$p(u|g) \propto p(g|u)p(u) \quad . \tag{11}$$

Definizione 2.1 *Sia* $u(x,y,w)$ *una funzione nelle variabili* $x,\ y,\ w$*, si definisce* Totale Variazione *di* u*:*

$$TV(u) = \int |\nabla u|\ dxdydw = \int\int\int \sqrt{u_x^2 + u_y^2 + u_w^2} dxdydw \quad , \tag{13}$$

$u_x,\ u_y$ *e* u_w *indicano rispettivamente le derivate parziali di* $u(x,y,w)$ *rispetto alle variabili* $x,\ y$ *e* w*.*

La scelta di TV è stata motivata dalle sue proprietà *edge preserving*, ovvero di preservare i contorni che caratterizzano le immagini SPECT.

Definizione 2.2 *Gli* edge *di un'immagine sono i punti di massimo locale del modulo del gradiente della funzione intensità luminosa che definisce l'immagine stessa[1].*

La (9) diventa:

$$u^* = arg\ \min_u F(u) = arg\ \min_u \left\{ \| Ku - g \|_2^2 + \alpha TV(u) \right\} \quad . \tag{14}$$

Le equazioni di Eulero Lagrange associate al problema (14) sono:

$$\begin{cases} K^*(Ku - z) + \alpha \cdot div\left(\frac{\nabla u}{|\nabla u|}\right) = 0 & , \quad (x,y) \in \Omega \\ \frac{\partial u}{\partial n} = 0 & , \quad (x,y) \in \partial\Omega \end{cases} \tag{15}$$

ovvero il problema di ricostruzione di dati 3D SPECT è stato ricondotto alla risoluzione di un problema differenziale alle derivate parziali di tipo ellittico non lineare, con condizione di Neumann sulla frontiera.
D'altro canto, l'introduzione del funzionale $TV(u)$ nella (12) conduce a:

$$u^* = \arg\max_u \{\log p(g|u) + TV(u)\} \quad . \tag{16}$$

Si osservi che l'operatore $\left(\frac{\nabla u]}{|\nabla u|}\right)$ non è definito nei punti in cui $|\nabla u| = 0$. Il verificarsi di $|\nabla u| = 0$ non è un'ipotesi remota, un'immagine infatti è un insieme di regioni ad intensità luminosa costante separate da *edge*, pertanto ci sono intere regioni di punti nelle quali risulta $\nabla u = 0$. Per ovviare a tale inconveniente: il funzionale (13) viene modificato con l'aggiunta di una *piccola* costante positiva β al modulo del gradiente di u, ovvero:

$$TV_\beta([u]) = \int \sqrt{|\nabla u|^2 + \beta^2} \quad , \tag{17}$$

la (17) si indicherà come funzionale TV *modificato* [33] [20].

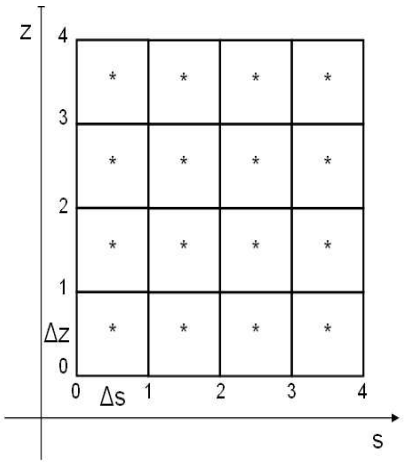


Figura 2: *Griglia bidimensionale, 4 × 4, con m fissato per la discretizzazione di $g(s, \phi_m, z)$. Il simbolo (*) indica il centro di ciascuna cella del reticolo.*

3 Discretizzazione del problema

3.1 Discretizzazione dell'operatore integrale K

Data la funzione $g(s, \phi, z)$ che descrive l'immagine 3D acquisita dlla gamma-camera, supposto che:

$$s \in [-r, r] \quad , \quad z \in [-R, R] \quad , \quad \phi \in [0, 2\pi] \quad , \tag{18}$$
$$\tag{19}$$

si consideri una griglia tridimensionale di dimensione $J \times M \times L$([4]) e siano:

$$\begin{aligned} \phi_m &= m\Delta\phi, \quad \Delta\phi = \tfrac{2\pi}{M-1}, \quad m = 0, \ldots, M-1 \\ s_k &= -r + k\Delta_s, \quad \Delta s = \tfrac{2r}{J-1}, \quad k = 0, \ldots, J-1 \\ z_l &= -R + l\Delta z, \quad \Delta z = \tfrac{2R}{L-1}, \quad l = 0, \ldots, L-1 \end{aligned}$$

gli $J \times M \times L$ valori assunti da ϕ, s e z sui nodi della griglia definita.

Per ogni l fissato e quindi fissato ϕ_l, si ottiene una griglia bidimensionale coincidente con gli $J \times M$ pixel dell'immagine discreta, come in figura 2. Indicati con C_{kml} i centri di ciascun pixel, questi hanno coordinate $(s_k + \frac{\Delta s}{2}, \phi_l, z_m + \frac{\Delta z}{2})$. La discretizzazione degli integrali nella (1), è stata effettuata mediante la formula di quadratura sul punto medio utilizzando come valutazioni della funzione

[4] I valori di J, M, L dipendono generalmente dal sistema di acquisizione. Generalmente i valori tipici di M, numero di viste angolari, sono 120, 60, mentre di J e di L, dimensioni di ciascuna immagine proiezione, sono 2^l, con $l = 7$ o $l = 8$.

g, i valori che essa assume nel centro del pixel. In particolare, si ha:

$$g^{kml} = \sum_{l'=0}^{L-1} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} k_{mkl}^{ijl'} \cdot u^{ijl'} \quad , \tag{20}$$

$$k = 0, \ldots, J-1;\ m = 0, \ldots, M-1;\ l = 0, \ldots, L-1.$$

In forma matriciale, la (20) si può riscrivere come:

$$[g] = [K]_3[u] \quad , \quad [g] \in \Re^{J \times M \times L} \quad , \quad [u] \in \Re^{N \times N \times L'}. \tag{21}$$

sistema lineare nell'incognita $[u]$.

Definizione 3.1 *La matrice* $[K]_3 \in \Re^{((J\times M)\times L\times (N\times N)\times L')}$ *discretizzazione del kernel* $h(s,t,z)$ *dell'operatore integrale* K *della (1), è detta* matrice di proiezione tridimensionale *ed i suoi elementi saranno indicati con* k_{mkl}^{ijl}. $[K]_3$ *è una matrice a blocchi di blocchi di blocchi.*
Analogamente la matrice $[K^*]_3$ *è detta* matrice di retroproiezione tridimensionale.

La (20) e la (21) rappresentano la discretizzazione del modello *fully 3D*.

Analogamente, per il modello *2D+1*, la discretizzazione dell'equazione (4) conduce al seguente sistema lineare:

$$g^{kml} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} E_{km}^{ij} \sum_{l'=0}^{L-1} A_{l'}^{l} \cdot u^{ijl'} \quad , \tag{22}$$

$$k = 0, \ldots, J-1;\ m = 0, \ldots, M-1;\ l = 0, \ldots, L-1.$$

In forma matriciale risulta:

$$[g] = [K]_3[u] \quad , \quad [K]_3 = [E]_2 \otimes [A]_1 \quad , \quad \text{con} \quad [E]_2 \in \Re^{((J\times M)\times (N\times N))} \quad \text{e} \quad [A]_1 \in \Re^{L\times L'}, \tag{23}$$

dove:

$$[E]_2 = \begin{pmatrix} E_{0,0}^{0,0} & \cdots & E_{0,0}^{0,N-1} & \cdots & E_{0,M-1}^{0,0} & \cdots & E_{0,M-1}^{0,N-1} \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ E_{0,0}^{N-1,0} & \cdots & E_{0,0}^{N-1,N-1} & \cdots & E_{0,M-1}^{N-1,0} & \cdots & E_{0,M-1}^{N-1,N-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ E_{J-1,0}^{0,0} & \cdots & E_{J-1,0}^{0,N-1} & \cdots & E_{J-1,M-1}^{0,0} & \cdots & E_{J-1,M-1}^{0,N-1} \\ \vdots & \ddots & \vdots & \cdots & \vdots & \ddots & \vdots \\ E_{J-1,0}^{N-1,0} & \cdots & E_{J-1,0}^{N-1,N-1} & \cdots & E_{J-1,M-1}^{N-1,0} & \cdots & E_{J-1,M-1}^{N-1,N-1} \end{pmatrix} , \tag{24}$$

$[A]_1$ è una matrice di Toeplitz:

$$[A]_1 = \begin{pmatrix} \alpha_0 & \dots & \alpha_{L'-1} \\ \vdots & \ddots & \vdots \\ \alpha_{L-1} & \dots & \alpha_0 \end{pmatrix} \quad , \tag{25}$$

i cui elementi $A^l_{l'}$, sono definiti dalla seguente relazione:

$$A^l_{l'} = \alpha_{l'-l} \quad , \tag{26}$$

dove:

$$\alpha_l = (\Delta z)^2 \int_{\frac{-1}{2}}^{\frac{1}{2}} \int_{\frac{-1}{2}}^{\frac{1}{2}} A[\Delta z(\zeta - \zeta' + l)] d\zeta d\zeta' \quad , \tag{27}$$

e $\otimes$ indica il prodotto tensoriale tra due matrici.
In accordo con la notazione introdotta in [12] i pedici 1, 2 e 3 indicano rispettivamente matrici a 2, 4 e 6 indici.

Come si può notare in (24) gli elementi di $[E]_2$ non dipendono dagli indici l ed l', pertanto il problema *2D+1* si può riguardare come un insieme di problemi *2D* opportunamente combinati tramite i valori della matrice $[A]_1$, cioè gli $\alpha_{l'-l}$, detti per tale motivo *pesi*.

Definizione 3.2 *La matrice* $[E]_2 \in \Re^{(J\times M)\times(N\times N)}$ *discretizzazione del kernel* $h(s - x\cdot\theta, x\cdot\theta^{\perp})$ *dell'operatore integrale* $E(s,t)$ *della (3) è detta* matrice di proiezione bidimensionale. $[E]_2$ *è una matrice a blocchi di blocchi.*

Si osservi che il prodotto di Kronecker della (23), gode della proprietà commutativa, pertanto:

$$[g] = [K]_3[u] = [E]_2 \otimes [A]_1[u] = [A]_1 \otimes [E]_2[u]$$

ovvero:

$$\begin{gathered} [f] = [E]_2[u] \quad , \\ [g] = [A]_1 \otimes [f] \quad . \\ [u] \in \Re^{(N\times N)\times L'}, \quad [f] \in \Re^{(J\times M)\times L'}, \quad [g] \in \Re^{(J\times M)\times L} \end{gathered} \tag{28}$$

La (21) e la (28) sono i modelli discreti utilizzati nell'ambito di questo lavoro.

Nei modelli *fully 3D* e *2D+1*, la matrice $[K]_3$ è molto sparsa. Una stima([5]) del suo grado di sparsità è del 95%([6]). Pertanto anche se $[K]_3$ è di elevate dimensioni([7]), in effetti molti dei suoi coefficienti sono nulli, e di fatto, vengono utilizzate opportune tecniche per la sua memorizzazione, denominate *lookup table*.

[5] La stima del grado di sparsità è stata effettuata sulla base delle sperimentazioni numeriche effettuate.

[6] Si osservi che anche se il grado di sparsità della matrice $[K]_3$ è alto, date le elevate dimensioni della matrice, dire che solo il 5% dei suoi elementi è non nullo, equivale a dire che circa 10^9 dei suoi elementi è diverso da zero rispetto ai 10^{12} dei totali.

[7] $[K]_3$ ha dimensione $(J\times M)\times L\times(N\times N)\times L'$, J, M, N, L ed L', sono almeno dell'ordine di 10^2.

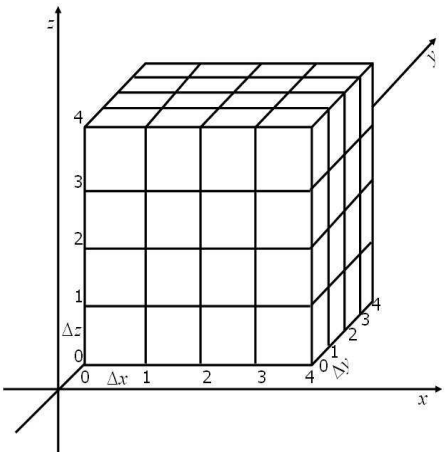


Figura 3: *Griglia tridimensionale* 4×4 *per la discretizzazione di* $u(x, y, z)$ *e per la discretizzazione dell'operatore differenziale* $\mathcal{L}$

3.2 Discretizzazione dell'operatore differenziale

L'operatore differenziale $\mathcal{L} = div \cdot (\frac{\nabla u}{|\nabla u|})$ viene discretizzato sulla griglia 3D del dominio (x, y, z) in figura 3, utilizzando differenze divise sulle celle centrali (CCFD), costruite su uno schema a sette punti (figura 4) [23].
Sia $u\,(x, y, z)$ una funzione nelle variabili x, y, z, il gradiente $\nabla u\,(x, y, z)$ di u è il vettore di componenti:

$$\nabla u\,(x, y) = \left(\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial u}{\partial z}\right).$$

La divergenza del gradiente di u, $div \cdot \nabla u\,(x, y, z)$ è:

$$div \cdot \nabla u\,(x, y, z) = \nabla^2 u\,(x, y, z) = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \quad .$$

Le derivate parziali prime e seconde di $u\,(x, y, z)$, sullo schema in figura 4 con passo di discretizzazione $h = \Delta x = \Delta y = \Delta z$, mediante CCFD, si approssimano con le differenze centrali:

$$\begin{aligned}
\frac{\partial u}{\partial x} &\approx \frac{u_{i+1,j,l'} - u_{i-1,j,l'}}{2h} \quad , \\
\frac{\partial u}{\partial y} &\approx \frac{u_{i,j+1,l'} - u_{i,j-1,l'}}{2h} \quad , \\
\frac{\partial u}{\partial z} &\approx \frac{u_{i,j,l'+1} - u_{i,j,l'-1}}{2h} \quad ,
\end{aligned}$$

$$\begin{aligned}
\frac{\partial^2 u}{\partial x^2} &\approx \frac{u_{i+1,j,l'} + u_{i-1,j,l'} - 2u_{i,j,l'}}{h^2} \quad , \\
\frac{\partial^2 u}{\partial y^2} &\approx \frac{u_{i,j+1,l'} + u_{i,j-1,l'} - 2u_{i,j,l'}}{h^2} \quad , \\
\frac{\partial^2 u}{\partial z^2} &\approx \frac{u_{i,j,l'+1} + u_{i,j,l'-1} - 2u_{i,j,l'}}{h^2} \quad .
\end{aligned}$$

La discretizzazione di $\mathcal{L}$, che indicheremo con la matrice L, può essere espressa in forma matriciale mediante la matrice $B_h[u]$, dove ogni riga di B_h contiene solo due termini diversi da zero, ed uguali ad 1:

$$L([u]) = B_h^T(D_h([u]))^{-1}B_h + B_h^T(D_h([u]))^{-1}B_h \quad ,$$

la matrice diagonale D_h, ha la seguente struttura:

$$D_h = \begin{bmatrix} D_x([u]) & 0 & 0 \\ 0 & D_y([u]) & 0 \\ 0 & 0 & D_z([u]) \end{bmatrix} \quad ,$$

e rappresenta la discretizzazione di $TV(u) = |\nabla u|$, con $D_x([u])$, $D_y([u])$, $D_z([u])$ matrici diagonali di dimensioni rispettivamente $(N-1)\times N\times L'$, $N\times(N-1)\times L'$, $N \times N \times (L'-1)$.

Con le notazioni introdotte, la discretizzazione di (15) conduce al seguente sistema di equazioni non lineari:

$$([K^*]_3[K]_3 - \alpha L([u]))[u] = [K^*]_3[g] \quad , \tag{29}$$

Si osservi che ponendo in (29), $\alpha = 0$, si ottiene il sistema lineare relativo alla discretizzazione del problema (8).

3.3 Discretizzazione del problema (12)

Consideriamo il dominio (x, y, z) discretizzato sulla griglia illustrata in figura (3), ovvero supponiamo di decomporlo in S cubetti (voxel), e denotiamo con u_s il valore di u nel centro del voxel s-mo con $s = 1, \ldots, S$.
La seguente corrispondenza biunivoca tra il generico voxel s, $s = 1, \ldots, S$ e la terna di indici (i, j, l') del reticolo di discretizzazione:

$$(i, j, l') \mapsto s = (l' - 1) * n^2 + (j - 1) * n + i.$$

consente di passare da (i, j, l') a $u(s)$.

Analogamente, considerata la funzione $g(s, \phi, z)$ supponiamo che la gamma camera acquisisca T valori di g, che si indicheranno con g_t, $t = 1, \ldots, T$. Si

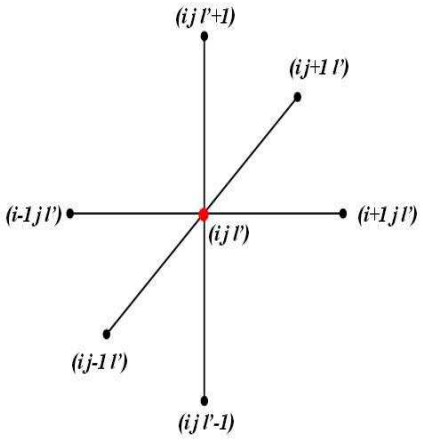


Figura 4: *Schema a sette punti utilizzato nel calcolo delle differenze finite per la discretizzazione dell'operatore differenziale* $\mathcal{L}$

ponga:

$$u = \begin{pmatrix} u_1 \\ \vdots \\ u_S \end{pmatrix} , \tag{30}$$

$$g = \begin{pmatrix} g_1 \\ \vdots \\ g_T \end{pmatrix} . \tag{31}$$

Sia k_{ts} la probabilità che la radiazione emessa dal radiofarmaco relativa al generico voxel $s = 1, \ldots, S$ sia inviduata dal generico detettore $t = \ldots, T$ della gamma camera, allora:

$$g_t^* = \sum_{s=1}^{S} k_{ts} u_s = E(g|u) \tag{32}$$

è la media statistica condizionata relativa a g_t. Si noti che a differenza della discretizzazione adoperata nel paragrafo 3.1, in questo caso la funzione u viene discretizzata sui voxel del dominio spaziale (x, y, z), anziché nei nodi (i, j, l'), associando al generico voxel s il valore di u nel centro del voxel.
La probabilità condizionata $p\,(g_t|u)$ di g_t è quindi:

$$p\,(g_t|u) = \text{Poisson}\left(\sum_{s=1}^{S} k_{ts} u_s\right) = e^{-\left(\sum_{s=1}^{S} k_{ts} u_s\right)^{g_t}} \frac{\left(\sum_{s=1}^{S} k_{ts} u_s\right)^{g_t}}{g_t!} . \tag{33}$$

Sostituendo la (32) nella (33) si ottiene:

$$p\left(g_t|u\right) = e^{-\left(g_t^*\right)^{g_t}} \frac{\left(g_t^*\right)^{g_t}}{g_t!} \quad . \tag{34}$$

Il problema (12) si riscrive come:

$$u^* = \arg\max_u \left\{ \sum_{t=1}^{T} \left(g_t \log \frac{g_t^*}{g_t!} - g_t^* \right) + \log p\left(u\right) \right\} \quad , \tag{35}$$

dove si sono sommati i contributi a tutte le T componenti del vettore g.
Se per semplicità di notazione, si trascura il termine $\log p\left(u\right)$ della (35), il vettore u^* è tale che:

$$\left. \frac{\partial l\left(u\right)}{\partial u_s} \right|_{u_s = u_s^*} = 0, \ \forall s = 1, \ldots, S \quad , \tag{36}$$

dove si è posto:

$$l\left(u\right) = \sum_{t=1}^{T} \left(g_t \log \frac{g_t^*}{g_t!} - g_t^* \right) \quad .$$

Condizioni sufficienti affinchè u^* sia soluzione di (12) sono le seguenti [35]:

$$\left. u_s \frac{\partial l\left(u\right)}{\partial u_s} \right|_{u_s = u_s^*} = 0, \quad u_s > 0 \tag{37}$$

$$\left. \frac{\partial l\left(u\right)}{\partial u_s} \right|_{u_s = u_s^*} \leq 0 \quad u_s = 0 \tag{38}$$

Dalla (37) si ha:

$$0 = u_{s'} \frac{\partial l\left(u\right)}{\partial u_{s'}} = u_{s'} \left(- \sum_{t=1}^{T} k_{ts'} + \sum_{t=1}^{T} \frac{k_{ts'} g_t}{\sum_{s=1}^{S} k_{ts} u_s} \right), \tag{39}$$

cioè equivalentemente:

$$u_{s'} \sum_{t=1}^{T} k_{ts'} = u_{s'} \quad . \sum_{t=1}^{T} \frac{k_{ts'} g_t}{\sum_{s=1}^{S} k_{ts} u_s}. \tag{40}$$

Introdotta la matrice $[K]_3$([8]):

$$[K]_3 := \left(k_{ts}\right)_{\substack{t\,=\,1,\ldots,T \\ s\,=\,1,\ldots,S}}$$

si ha:

[8] Si noti che tale matrice coincide con la matrice $[K]_3$ ottenuta dalla discretizzazione dell'operatore di Radon *sfocata* nella sezione 3

$$
\begin{aligned}
u\left([K]_3^T[1]\right) &= u\left([K]_3^T \frac{g}{([K]_3 u)}\right) && (41)\\
0 &= -u\left([K]_3^T[1]\right) + u\left(K^T \frac{g}{([K]_3 u)}\right) && (42)\\
0 &= -u\left\{\left([K]_3^T[1]\right) + \left([K]_3^T \frac{g}{([K]_3 u)}\right)\right\} && (43)
\end{aligned}
$$

dove $[1] \in \Re^t$ è il vettore costituito da elementi tutti uguali ad 1, e le operazioni di divisione e moltiplicazione fra vettori si intendono componente per componente. Il vettore u^* è quindi soluzione di un sistema di equazioni non lineari.

Si osservi che assunte opportune ipotesi sul sistema di acquisizione, per la matrice $[K]_3$ è ancora valida la fattorizzazione (23).

4 Gli algoritmi risolutivi

Il problema di ricostruzione di dati 3D SPECT è stato ricondotto alla risoluzione di un sistema di equazioni non lineari del tipo (29) o (43). In questo paragrafo sono descritti gli algoritmi iterativi utilizzati per la risoluzione di tale problema. Si noti che, trattandosi di problemi discreti derivanti da problemi inversi mal posti, il numero di iterazioni assume il ruolo di "parametro di regolarizzazione". Infatti, tali metodi hanno tutti lo stesso comportamento: l'errore ad ogni iterazione diminuisce, fino a raggiungere un minimo, dopodiché aumenta. Tale proprietà è detta *semiconvergenza*, e può essere utilizzata per stabilire il massimo numero di iterazioni, con evidenti vantaggi in termini di efficienza.

Come si illustrerà in seguito, sia per la risoluzione di (29) che per la risoluzione di (43) è stato utilizzato il metodo del *Punto Fisso* (FP), specializzato opportunamente per i due sistemi.

4.1 Il Punto Fisso

Fra i metodi proposti in letteratura[19] per la risoluzione di sistemi non lineari, sono stati utilizzati [6]: il Punto Fisso [2] e Newton [17]. Posto $\omega = \frac{\nabla[u]}{|\nabla[u]|}$ [31] [32], il sistema (29) si può riscrivere come un sistema lineare di due equazioni in due incognite u e ω:

$$
\begin{cases}
\mathcal{F}(u,\omega) &\equiv [K^*]_3([K]_3[u] - [g]) + \alpha \nabla \cdot \omega = 0\\
\mathcal{G}(u,\omega) &\equiv \sqrt{(\nabla[u])^2 + \beta^2} \quad \omega - \nabla[u] = 0
\end{cases} \tag{44}
$$

$$
\begin{cases}
\mathcal{F}(u,\omega) = [K^*]_3([K]_3[u] - [g]) + \alpha \nabla \cdot \omega = 0\\
\omega = \frac{\nabla[u]_0}{\sqrt{(\nabla[u]_0)^2 + \beta^2}}
\end{cases} \tag{45}
$$

procedure PCG(K,K*,M,f,g);
* Calcolo del residuo al passo zero.
$r^{(0)} = g$;
* Precondizionamento al passo zero.
$z^{(0)} = M^{-1}\ r^{(0)}$;
* Direzione di ricerca al passo zero.
$p^{(0)} = z^{(0)}$;
$\gamma_0 = 0$;
for k=1, maxit **do**
    * I. Calcolo di γ_k
    $\gamma_k = \|r^{(k-1)}\|_2^2$;
    * II. Calcolo di β_k
    $\beta_k = \gamma_{k-1}\ /\ \gamma_k$;
    * III. Aggiornamento del vettore $p^{(k)}$
    $p^{(k)} = r^{(k)} + \beta_k p^{(k-1)}$
    * IV. Aggiornamento del vettore $q^{(k)}$
    $q^{(k)} = K^*M^{-1}K\ p^{(k)}$;
    $\alpha_k = \gamma_k / \|q^{(k)}\|_2^2$;
    * V. Aggiornamento della soluzione $f^{(k)}$
    $f^{(k)} = f^{(k-1)} + \alpha_k p^{(k)}$;
    * VI. Aggiornamento del residuo $r^{(k)}$
    $r^{(k)} = r^{(k-1)}$ - $\alpha_k q^{(k)}$;
endfor
end procedure PCG.

Figura 5: *Schema dell'algoritmo PCG.*

L'eliminazione della variabile ω dalla prima equazione del sistema, porta all'equazione:

$$[K^*]_3([K]_3[u] + [u]_0) + \alpha\nabla \cdot \left(\frac{\nabla[u]}{\sqrt{(\nabla[u])^2 + \beta^2}} \right) = 0 \quad . \tag{46}$$

L'idea di base del FP è di linearizzare il sistema di equazioni (44) fissando $\nabla(u) = \nabla[u]^m$. Posto $[u]_0 = [g]$ come proposto in [13], si ottiene:

$$[K^*]_3([K]_3[u]^{m+1} + [u]_0) + \alpha\nabla \cdot \left(\frac{\nabla[u]^{m+1}}{\sqrt{(\nabla[u]^m)^2 + \beta^2}} \right) = 0 \quad . \tag{47}$$

Lo schema iterativo (47) ha una convergenza lineare [22].
L'algoritmo FP, come schematizzato in figura 6, è costituito da due cicli di iterazioni innestati, costituiti dalle operazioni principali seguenti:

1. *I ciclo:* definizione delle matrici del sistema (47) delle iterazioni sul FP, al passo m, con $m = 0, 1, 2, \ldots$:
   - 1.a costruzione della matrice $[K]_3$;
   - 1.b costruzione della matrice $[K^*]_3$;
   - 1.c costruzione della matrice $L^m([u]^m)$;
2. *II ciclo*: risoluzione del sistema ottenuto al passo m mediante il metodo del Gradiente Coniugato Precondizionato (PCG):
   - 2.a calcolo del prodotto $[K^*]_3[K]_3[u]^{m+1}$;
   - 2.b calcolo del prodotto $L^m([u]^m)[u]^{m+1}$.

Si osservi che nel *I ciclo* l'unica matrice che dipende dall'iterazione m-esima è $L^m([u]^m)$, tutte le altre sono definite una sola volta al passo $m = 0$. Inoltre, al punto 1.a e 1.b per la sparsità delle matrici $[K]_3$ e $[K^*]_3$, viene utilizzato uno schema di memorizzazione opportuno, la *lookup table*.
Il precondizionatore utilizzato è quello (diagonale proposto in [29]) particolarmente adatto nel problema della ricostruzione di dati SPECT.
L'algoritmo del PCG è illustrato in figura 5.

4.2 Algoritmo EM

La (43) genera in maniera naturale, un'iterazione funzionale di tipo punto fisso e conduce al seguente schema iterativo per il calcolo di un'approssimazione u^* della (12):

$$u^{m+1} = \frac{[1]}{[K]_3^T\,[1]} u^m [K]_3^T \frac{g}{[K]_3 u^m} \quad m = 1, 2, \ldots . \tag{48}$$

begin procedure FP
{
`Costruzione della matrice` F;

$F = [K^*]_3[g]$;

`* Iterazioni sul Punto Fisso`

for $i = 1, \cdots, maxit$ **do**

`Calcolo della matrice` $L^m([u]^m)$;

`Risoluzione del sistema` $([K^*]_3[K]_3 + \alpha L^m([u]^m))[u] = F$
`utilizzando il` PCG `(Innesto di un ciclo interno);`

endfor

}**end procedure** FP

Figura 6: *Schema dell'algoritmo FP.*

Teorema 4.1 *Lo schema iterativo (48) converge alla soluzione u^* del problema di massimo (12) [35].*

Lo schema iterativo (48) è alla base dell'algoritmo di *Expectation Maximization (EM)*. La forma esplicita di EM è la seguente:

$$u_{s'}^{m+1} = u_{s'}^{m} \frac{1}{\sum_{t=1}^{T} k_{ts'}} \sum_{t=1}^{T} \frac{g_t k_{ts'}}{\sum_{s=1}^{S} k_{ts} u_s^m} \tag{49}$$

4.3 Versione accelerata di EM mediante *Ordered Subset*

L'algoritmo EM (48) ha una lenta convergenza [30]. Nel seguito si illustrerà la versione accelerata, detta EMOSn (*Expectation Maximization Ordered Subset*)[30]. Sia $\Xi = \{\Omega_i : i = 1, \ldots, n\}$ una partizione *Ordered Subset* dell'insieme $\Omega = \{1, \ldots, T\}$ delle viste angolari, cioè:

$$\begin{aligned} \Xi &= \{\Omega_i\}_{i=1,\ldots,n}, \\ \Omega_i &\subseteq \Omega, \\ \bigcup_{i=1}^{n} \Omega_i &= \Omega, \\ \Omega_i \bigcap \Omega_j &= \oslash, \quad \forall i \neq j \end{aligned} \tag{50}$$

Tale partizione riorganizza la matrice $[K]_3$ per blocchi di righe:

$$[K]_3 = \begin{bmatrix} [K_1]_3 \\ \vdots \\ [K_n]_3 \end{bmatrix}, \quad [K_i]_3 \in \Re^{\frac{T}{n} \times S} \tag{51}$$

$[K_i]_3$ è una restrizione dell'operatore $[K]_3$ alle sole viste angolari appartenenti al sottoinsieme (*subset*) Ω_i.

Nel caso *2D+1*, per ciascuna matrice $[K_i]_3$ è valida la fattorizzazione (23), cioè:

$$[K_i]_3[u] = ([A]_1 \otimes [E]_2)_i[u] \quad i = 1, \ldots, n \quad . \tag{52}$$

Si osservi che EMOS$_n$ coincide con EM quando la cardinalità n di Ξ è uguale a 1, cioè la partizione Ξ è costituita da un solo *subset* che coincide con Ω. EM rientra nella classe dei metodi SART (*Simultaneous Algebraic Reconstruction Technique*)[14] [15]. Allo stesso modo, EMOS$_n$ è classificabile come metodo BIART (*Block Iterative Algebraic Reconstruction Technique*) [15], cioè riformulazioni a blocchi di opportuni metodi iterativi, realizzate per accelerare la convergenza. Un'iterazione di EM coincide con n iterazioni EMOS$_n$ e si prova infatti che [30] il raggio spettrale, della matrice di iterazione di EMOS$_n$ è una potenza n-esima del raggio spettrale della matrice di iterazione di EM.

La partizione (50) delle viste angolari in *ordered subset* Ω_i, permette di riorganizzare il prodotto $[K]_3[u]$, per entrambi i modelli di acquisizione, in n prodotti matrice vettore di dimensione più piccola.

Uno schema dell'algoritmo EMOSn è illustrato in figura 7.

4.4 La Totale Variazione nell'algorimo EM.

Dalla (43), con l'aggiunta del funzionale TV [5], si ha lo schema seguente [34]:

$$u^{new} = \frac{[1]}{[K]_3^T\,[1] + \alpha L^{old}\,(u^{old})\,u^{old}} u^{old} [K]_3^T \frac{g}{[K]_3 u^{old}}, \tag{53}$$

dove [1] è il vettore costituito da elementi tutti uguali ad 1, e le operazioni di divisione, moltiplicazione e somma fra vettori si intendono componente per componente.
Esplicitando la (53), si ottiene:

$$u_s^{m+1} = u_s^m \frac{1}{\sum_{t=1}^T k_{ts} + \alpha\,(L\,(u^m)\,u^m)_s} \sum_{t=1}^T \frac{g_t k_{ts}}{\sum_{s=1}^S k_{ts} u_s^m} \quad m = 1, 2, \ldots \tag{54}$$

L'algoritmo EMOSn con TV, corrispondente alla formulazione (53) è illustrato in figura 8. Esso è costituito da due cicli di iterazioni innestati, costituiti dalle operazioni principali seguenti:

procedure EMOSn

1. $m = 0$
2. $\hat{u}^m$ costante e positivo
3. ripeti fino alla convergenza di $\hat{u}^m$
   (a) $u^1 = \hat{u}^m$
   (b) $m = m + 1$
   (c) Per livello di OS $i = 1, \ldots, n$
      i. calcolo del prodotto $\mu^i = [K_i]_3 u^i$

$$\mu_t^i = \sum_{s=1,\ldots,S} k_{ts} u_s^i, \ t \in T_i$$

      ii. calcolo di $u^{i+1} = \frac{[1]}{[K]_3^T [1]} u^i [K_i]_3^T \frac{g}{\mu^i}$

$$u_s^{i+1} = u_s^i \frac{1}{\sum_{t=1}^T k_{ts}} \sum_{t \in T_i} \frac{g_t k_{ts}}{\mu_t^i}, \ s = 1, \ldots, S$$

   (d) $\hat{u}^m = u^{n+1}$
4. $u = \hat{u}^m$

end procedure EMOSn

Figura 7: *Schema dell'algoritmo EMOS*n

procedure EMOSn-TV(input: K, g ; output: u)

1. $m = 0$
2. $\hat{u}^m$ costante e positivo
3. ripeti fino alla convergenza di $\hat{u}^m$
   (a) $u^1 = \hat{u}^m$
   (b) $m = m + 1$
   (c) calcolo di $L^m\left(u^1\right)$
   (d) Per livello di OS $i = 1, \ldots, n$
      i. calcolo del prodotto $\mu^i = [K_i]_3 u^i$

$$\mu_t^i = \sum_{s=1,\ldots,S} k_{ts} u_s^i, \ t \in T_i$$

      ii. calcolo di $u^{i+1} = \frac{[1]}{[K]_3^T [1] + \alpha L^m(u^1) u^i} u^i [K_i]_3^T \frac{g}{\mu^i}$

$$u_s^{i+1} = u_s^i \frac{1}{\sum_{t=1}^T k_{ts} + \left(L^m\left(u^1\right) u^i\right)_s} \sum_{t \in T} \frac{g_t k_{ts}}{\mu_t^i}, \ s = 1, \ldots, S$$

   (e) $\hat{u}^m = u^{n+1}$
4. $u = \hat{u}^m$

end procedure EMOSn-TV

Figura 8: *Schema dell'algoritmo EMOS*n *TV-regolarizzato*

1. *ciclo di EM:* definizione delle matrici dello schema (53) delle iterazioni su EM al passo $m = 1, 2, 3, \ldots$:
   1.a costruzione della matrice $[K]_3$;
   1.b costruzione della matrice $[K^*]_3$;
   1.c costruzione della matrice $L^m([u]^m)$;
2. *ciclo di OS*:
   2.a calcolo del prodotto $\mu^i = [K_i]_3 u^i$;
   2.b calcolo del prodotto $L^m([u]^m) u^i$.

5 Analisi della complessità computazionale degli algoritmi risolutivi

In questo paragrafo viene fornita una stima della complessità computazionale degli algoritmi di ricostruzione illustrati nel paragrafo 4: il CG e l'EMOS$_n$([9]), sia per il modello *fully 3D* sia per il modello *2D+1*.

In particolare, sarà calcolata una stima del numero di operazioni floating-point N_{cgXX}([10]) di una singola iterazione dell'algoritmo CG ed una stima del numero di operazioni floating-point N_{em2d+1}([11]) di una singola iterazione dell'algoritmo EM.
N_{cgXX} e N_{em2d+1} saranno poi utilizzati per calcolare una stima teorica di τ_{cgXX} e di τ_{em2d+1}, tempi di esecuzione di una singola iterazione dei nuclei computazionali dei moduli software di MEDITOMO, basati rispettivamente su CG e su EMOS$_n$.

Si noti in figura 5 e 7, che ogni iterazione di CG e di EMOS$_n$, è composta da operazioni di algebra lineare di base fra cui il prodotto matrice vettore $[K]_3[g]$, (o equivalentemente il prodotto matrice vettore $[K^*]_3[u]$). La matrice $[K]_3$ definita in (3.1), ha dimensione $(J \times M) \times L \times (N \times N) \times L'$ ed un coefficiente di sparsità del 95%, pertanto si proverà che il costo effettivo di tale prodotto matrice-vettore non è dell'ordine di $\mathcal{O}((J \times M) \times L \times (N \times N) \times L')$ ma si riduce ad ogni iterazione di circa il 95%.

Nel seguito, per indicare le dimensioni del sistema (47) e (48), saranno utilizzate le notazioni seguenti:

`nsez`	L	numero di sezioni da ricostruire
`kdim`	J	dimensione lineare delle proiezioni
`nang`	M	numero di viste angolari
`nsez`	L'	numero di proiezioni assegnate
`ndim`	N	dimensione della sezione da ricostruire
`nos`	n	numero di *subset*
`kos`	$\frac{M}{n}$	numero di viste angolari contenute in un subset
`nmat`	-	area di lavoro

`nmat` è il numero di punti interni di un cerchio inscritto in un quadrato di

[9] Si osservi, dallo schema in figura 6, che un'iterazione del FP equivale ad un'iterazione del CG, più la costruzione della matrice $L^m([u]^m)$ nel ciclo esterno ed il prodotto di $L^m([u]^m)[u]^{m+1}$, nel ciclo interno. Sperimentazioni numeriche hanno provato che queste due operazioni non incidono significativamente sul costo computazionale totale, pertanto si può affermare che il costo computazionale di una iterazione del FP si può ricondurre ad un'iterazione del CG. Per lo stesso motivo, il costo computazionale dell'algoritmo EMOS$_n$ TV regolarizzato in figura 8, si può ricondurre a quello di EMOS$_n$.

[10] Il pedice "XX" indica `3d` oppure `2d+1`, ovvero il modello di ricostruzione assunto *fully 3D* o *2D+1*.

[11] L'algoritmo EM è stato utilizzato solo per il modello *2D+1*.

diametro `ndim`, pertanto

$$\texttt{nmat} \approx \pi \left(\frac{\texttt{ndim}}{2} \right)^2 \quad .$$

5.1 Stima del Numero di Operazioni Floating Point del CG.

Osservando in figura 5 lo schema dell'algoritmo CG, in sintesi esso è composto dalle seguenti operazioni di algebra lineare di base:

1. due prodotti scalari;
2. tre operazioni *axpy*([12]);
3. due prodotti matrice vettore([13]).

5.1.1 Stima di N_{cg3d}.

La stima di N_{cg3d} è stata effettuata contando le operazioni floating point eseguite in una singola iterazione del CG, seguendo lo schema in figura 5 [7].
Le operazioni dei punti 1 e 2, avvengono tra vettori di lunghezza $\texttt{nmat} \times \texttt{nsez}$, e pertanto, hanno in totale, un costo computazionale uguale a:

$$N_1 = 12 \times \texttt{nmat} \times \texttt{nsez} \quad . \tag{55}$$

I due prodotti matrice vettore del punto 3, hanno un costo computazionale uguale a [7]:

$$N_2 = \texttt{nang} \times \texttt{nsez} \times (24 + 12 \times \texttt{ndim} + 505 \times \texttt{nmat} + 45 \times \texttt{kdim}) \quad . \tag{56}$$

Si osservi che se la matrice $[K]_3$ non fosse sparsa, la (56) dovrebbe assumere il seguente valore:

$$\tilde{N_2} = 2 \times \texttt{nsez}^2 \times \texttt{nang} \times \texttt{kdim} \times \texttt{ndim}^2 \quad . \tag{57}$$

Se si suppone che $\texttt{nsez} = 25$, $\texttt{nang} = 120$, $\texttt{kdim} = \texttt{ndim} = 128$, si ottiene:

$$N_2 \sim 7 \times 10^9 \quad , \quad \widetilde{N_2} = 335 \times 10^9 \Longleftrightarrow \frac{\widetilde{N_2}}{N_2} = 2\% \tag{58}$$

Il rapporto di proporzionalità (58) conferma che la riduzione del costo computazionale di entrambi i prodotti matrice vettore in un'iterazione di CG, è del

[12] Si indica con il termine *axpy*, un'operazione di aggiornamento di un vettore, mediante somma con un altro vettore moltiplicato per uno scalare α, cioè:

$$y \leftarrow y + \alpha x \quad .$$

[13] Si ricorda che tali prodotti non hanno un costo computazionale del $\mathcal{O}((J \times M) \times L \times (N \times N) \times L')$, ma il costo di entrambi si riduce del 95% perché la matrice coinvolta è sparsa.

98%.

Concludendo, il numero totale di operazioni floating point, effettuate in un'iterazione di CG, con modello *fully 3D*, è uguale alla somma di (55) e (56), cioè N_{cg3d} è uguale a:

$$\begin{aligned} N_{cg3d} &= 12 \times \texttt{nmat} \times \texttt{nsez} + \\ &+ \texttt{nang} \times \texttt{nsez} \times (24 + 12 \times \texttt{ndim} + 505 \times \texttt{nmat} + 45 \times \texttt{kdim}) \quad . \end{aligned} \tag{59}$$

5.1.2 Stima di N_{cg2D+1}.

Anche la stima di N_{cg2D+1}, è stata effettuata contando le operazioni floating point, eseguite in una singola iterazione del CG, seguendo lo schema in figura 5 [8]. Le operazioni dei punti 1 e 2, avvengono tra vettori di lunghezza $\texttt{ndim}^2 \times \texttt{nsez}$ e pertanto, hanno in totale, un costo computazionale uguale a:

$$N_1 = 12 \times \texttt{ndim}^2 \times \texttt{nsez} + 1 \quad . \tag{60}$$

I due prodotti matrice vettore del punto 3, hanno un costo computazionale uguale a[8]:

$$N_2 = \texttt{nsez} \times (\texttt{nang} \times (106 \times \texttt{ndim}^2 + 61 \times \texttt{kdim})) \quad . \tag{61}$$

Anche in questo caso, se la matrice $[K]_3$ non fosse sparsa, la (61) dovrebbe assumere il valore (57), supponendo che $\texttt{nsez} = 25$, $\texttt{nang} = 120$, $\texttt{kdim} = \texttt{ndim} = 128$, si ottiene:

$$N_2 \sim 4 \times 10^9 \quad , \quad \widetilde{N_2} = 335 \times 10^9 \Longleftrightarrow \frac{N_2}{\widetilde{N_2}} = 1,5\% \quad . \tag{62}$$

Il rapporto di proporzionalità (62) conferma che la riduzione del costo computazionale di entrambi i prodotti matrice vettore in un'iterazione di CG, è del 98.5%.

Concludendo, il numero totale di operazioni floating point, effettuate in un'iterazione del CG, con modello *2D+1*, è uguale alla somma di (60) e di (61), cioè N_{cg2d+1} è uguale a:

$$\begin{aligned} N_{cg2d+1} &= 12 \times \texttt{ndim}^2 \times \texttt{nsez} + 1 + \\ &+ \texttt{nsez} \times (\texttt{nang} \times (106 \times \texttt{ndim}^2 + 61 \times \texttt{kdim})) \quad . \end{aligned} \tag{63}$$

5.2 Stima del Numero di Operazioni Floating Point di EMOS$_n$.

Osservando in figura 7 lo schema dell'algoritmo EMOS$_n$, in sintesi esso risulta composto dalle seguenti operazioni di algebra lineare di base:

1. due prodotti scalari;
2. due prodotti matrice vettore([13]).

5.2.1 Stima di $\mathbf{N}_{em2d+1}$

La stima di N_{em2d+1} è stata effettuata contando le operazioni floating point eseguite in una singola iterazione di EMOS$_n$, seguendo lo schema in figura 7 [8]. Le operazioni del punto 1, avvengono `nos` tra vettori di lunghezza $\texttt{ndim}^2 \times \texttt{nsez}$, e pertanto hanno un costo computazionale uguale a:

$$N_1 = 2 \times \texttt{ndim}^2 \times \texttt{nsez} \quad . \tag{64}$$

I due prodotti matrice vettore hanno un costo computazionale uguale a [8]:

$$N_2 = 2 \times \texttt{nsez} \times (\texttt{nang} \times (48 \times \texttt{ndim}^2 + 31 \times \texttt{kdim})) \quad . \tag{65}$$

Anche in questo caso, se la matrice $[K]_3$ non fosse sparsa, la (65) dovrebbe assumere il valore (57), supponendo che $\texttt{nsez} = 25$, $\texttt{nang} = 120$, $\texttt{kdim} = \texttt{ndim} = 128$, si ottiene:

$$N_2 \sim 2 \times 10^9 \quad , \quad \widetilde{N_2} = 335 \times 10^9 \Longleftrightarrow \frac{N_2}{\widetilde{N_2}} = 0,5\% \quad . \tag{66}$$

Il rapporto di proporzionalità (66) conferma che la riduzione del costo computazionale di entrambi i prodotti matrice-vettore, in ciascuna iterazione di EMOS$_n$, è del 99.5%.

Concludendo, il numero totale di operazioni floating point eseguite in una iterazione di EMOS$_n$, con modello *2D+1*, è uguale alla somma di (64) e di (65), cioè N_{em2d+1} è uguale a:

$$\begin{aligned} N_{em2d+1} &= 2 \times \texttt{nos} \times \texttt{nsez} \times \texttt{ndim}^2 + \\ &+ 2 \times \texttt{nsez} \times (\texttt{nang} \times (48 \times \texttt{ndim}^2 + 31 \times \texttt{kdim})) \quad . \end{aligned} \tag{67}$$

Si osservi che confrontando la (63) e la (67) risulta che se $\texttt{nos} \approx 6$ allora $N_{cg2d+1} \approx N_{em2d+1}$, ovvero il numero di operazioni floating point di una singola iterazione di CG coincide con il numero di operazioni floating point di una singola iterazione di EMOS$_n$.

5.3 Una sperimentazione numerica.

Il modello analitico assunto per la stima del tempo di esecuzione impiegato da un processore in una singola iterazione del metodo iterativo di ricostruzione in esame è il seguente:

$$\tau_{yyXX} = N_{yyXX} \times t_{flop} \quad , \tag{68}$$

dove:

- N_{yyXX}, è il numero di operazioni fl.p. eseguite in una singola iterazione del metodo iterativo in esame([14]);
- t_{flop}, è il tempo di esecuzione di una singola operazione floating point.

Per le variabili descritte nel paragrafo 5 sono stati utilizzati i valori seguenti:

- per il modello *fully 3D*:

$$\begin{aligned} L = &\quad \texttt{nsez} = \quad 25 \\ J = &\quad \texttt{kdim} = \quad 158 \\ M = &\quad \texttt{nang} = \quad 120 \\ N = &\quad \texttt{ndim} = \quad 128 \end{aligned}$$

$$\texttt{nmat} \approx \quad \pi \left(\frac{\texttt{ndim}}{2} \right)^2$$

- per il modello *2D+1*:

$$\begin{aligned} L = &\quad \texttt{nsez} = \quad 70 \\ M = &\quad \texttt{nang} = \quad 120 \\ J = &\quad \texttt{kdim} = \quad 158 \\ N = &\quad \texttt{ndim} = \quad 134 \\ &\quad \texttt{nos} = \quad 15 \\ &\quad \texttt{kos} = \quad 8 \end{aligned}$$

In tabella 1, sono riportate le stime teoriche τ_t di τ_{cgXX} e τ_{em2d+1}, a confronto con le stime sperimentali τ_s del tempo di esecuzione rispettivamente di una singola iterazione del CG e di una singola iterazione di EMOS$_n$, su un'architettura monoprocessore con le seguenti caratteristiche:

[14] Il pedice "*yy*" indica che N_{yyXX} è utilizzato nel modello per rappresentare *cg* o *em*, ovvero N_{cgXX} o N_{em2d+1}, conseguentemente τ_{yyXX} rappresenterà il tempo di esecuzione τ_{cgXX} di una iterazione del CG, o il tempo di esecuzione τ_{em2d+1} di una iterazione di EMOS$_n$.

	τ_t	τ_s	α
τ_{cg3d}	80 sec	250 sec	1/3
τ_{cg2d+1}	16 sec	65 sec	1/4
τ_{em2d+1}	14 sec	75 sec	1/4

Tabella 1: *Stime teoriche e sperimentali dei tempi di esecuzione di una singola iterazione di CG e di EMOS$_n$.*

Processore	-	Pentium 4.1.5
Cache	-	256Kb
Memoria	-	512Mb (Rimm PC800)
Sistema Operativo	-	Red Hat linux 7.2 Kernel 2.4.20 - 10
BLAS	-	Intel Math Kernel
Compilatore FORTRAN	-	GNU fortran 0.5.26

Ipotizzando che $t_p \sim 0.1 \times 10^{-8} sec$([15]), in tabella è riportata la stima teorica τ_t, la stima sperimentale τ_s ed il fattore di proporzionalità α, che lega τ_t e τ_s, sia per τ_{cgXX} che per τ_{em2d+1}, ovvero:

$$\tau_t \approx \alpha \cdot \tau_s \quad . \tag{69}$$

Dai risultati ottenuti, la stima teorica τ_t del tempo e la stima sperimentale τ_s, differiscono nel modello *fully 3D* di un fattore $\alpha = 1/3$, mentre nel modello *2D+1* di un fattore $\alpha = 1/4$, per entrambi gli algoritmi di ricostruzione, ovvero:

- modello *fully 3D*:
$$\tau_t \approx \frac{1}{3} \quad \tau_s \quad ; \tag{70}$$
- modello *2D+1*:
$$\tau_t \approx \frac{1}{4} \quad \tau_s \quad . \tag{71}$$

Si osservi, dai risultati riportati in tabella 1 che lo scarto tra τ_t e τ_s è dovuto anche alla scelta effettuata di trascurare il tempo impiegato nelle operazioni fra gli interi e il tempo impiegato per gli accessi alla memoria.

In conclusione, le relazioni di proporzionalità (70) e (71), per entrambi i modelli assunti e per entrambi gli algoritmi considerati inducono ad affermare che la stima teorica sviluppata può essere considerata affidabile e realistica.

[15] Il valore di t_p è stato fissato al valore $0.1 \times 10^{-8} sec$ effettuando una media di alcune stime sperimentali effettuate sull'architettura in esame.

6 Introduzione del parallelismo negli algoritmi di ricostruzione di dati 3D SPECT

Dalla tabella 1, si osserva che per ottenere una ricostruzione di dati 3D SPECT, dal momento che tipicamente, sono necessarie in media 10 iterazioni del CG o di EMOS$_n$, il tempo richiesto è di circa 40 minuti col modello *fully 3D* e circa 18 minuti con il modello *2D+1*. La necessità di produrre una soluzione in tempo reale ha condotto quindi, verso lo sviluppo del software in ambiente di calcolo parallelo. L'introduzione del parallelismo è stata rivolta principalmente a ridurre i costi computazionali delle operazioni più onerose degli algoritmi di ricostruzione illustrati (FP, PCG, EM ed opportune varianti). L'analisi computazionale descritta nel paragrafo 5 e le esperienze numeriche effettuate hanno mostrato che indipendentemente dal modello di ricostruzione assunto, *fully 3D* (1) o *2D+1* (4), l'operazione computazionalmente più costosa che si indicherà col nome di *proiezione/retroproiezione* è la seguente:

$$z = [K^*]_3[K]_3 u \quad . \tag{72}$$

A seconda poi, del modello di ricostruzione, è stato introdotto opportunamente il parallelismo, nel calcolo della (72). In particolare:

fully 3D l'approccio adottato si basa sulla riorganizzazione della fase di elaborazione dei dati in operazioni concorrenti rispetto all'insieme delle viste angolari, pertanto il parallelismo è stato introdotto sul processo fisico di elaborazione (*subproblems decomposition*);

2D+1 l'approccio adottato sfrutta la decomposizione del problema *2D+1* in un insieme di problemi bidimensionali che possono essere risolti concorrentemente (*coarse-grain decomposition*);

2D l'approccio adottato si basa sull'introduzione del parallelismo nei nuclei computazionali interni (*fine-grain decomposition*).

I dettagli delle strategie di parallelizzazione a seconda del modello di acquisizione assunto, sono illustrati nei paragrafi 6.1 e 6.2.

6.1 Strategia di parallelizzazione nel modello *fully 3D*

Dalla (20) si ha:

$$\mathbf{g}^l = \sum_{l'=0}^{L'-1} K_l^{l'} \mathbf{u}_{l'} \quad , l = 0, \ldots, L-1 \quad , \tag{73}$$

che si può esprimere in forma matriciale a blocchi, come segue:

$$\begin{pmatrix} \mathbf{g}^0 \\ \vdots \\ \mathbf{g^{L-1}} \end{pmatrix} = \begin{pmatrix} [E_0]_2 & \ldots & [E_{L'-1}]_2 \\ \vdots & \ddots & \vdots \\ [E_{L-1}]_2 & \ldots & [E_0]_2 \end{pmatrix} \begin{pmatrix} \mathbf{u}^0 \\ \vdots \\ \mathbf{u^{L'-1}} \end{pmatrix} \quad . \tag{74}$$

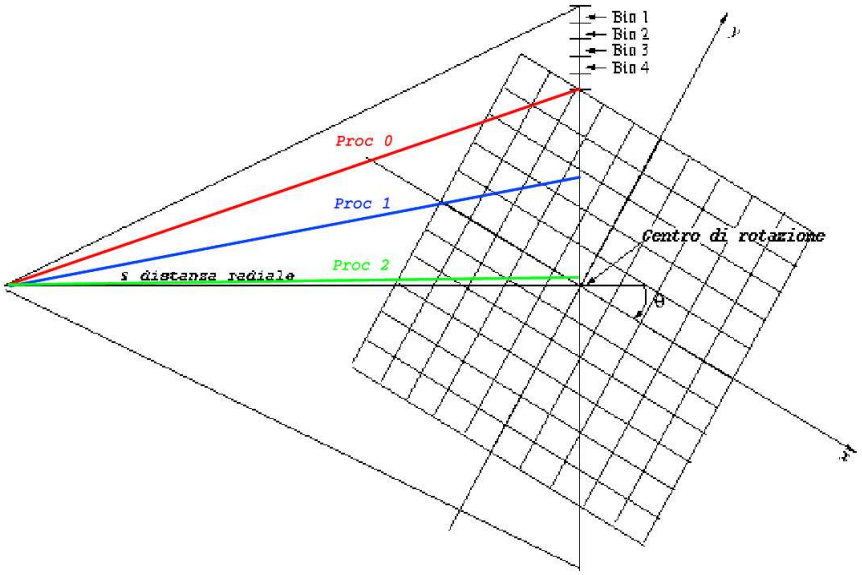


Figura 9: *Schematizzazione del processo di acquisizione delle proiezioni bidimensionali e della distribuzione delle viste angolari fra i processori.*

ciò permette di riorganizzare il prodotto $[K]_3\mathbf{u}$, in un insieme di prodotti $[E_i]_2 \cdot \mathbf{u}^j$ indipendenti gli uni dagli altri([16]). L'esecuzione di ciascuno di questi prodotti può essere assegnato ad un processore. Ciò coincide dal pundo di vista del modello fisico([17]), nella distribuzione fra i processori dell'insieme delle viste angolari M (e per semplicità si supponga $L' = L$). Uno schema illustrativo della distribuzione delle viste angolari fra i processori è rappresentato in figura 9.

6.1.1 Dettagli implementativi nella strategia del *fully 3D*

Considerando l'introduzione del parallelismo effettuata sul problema, la topologia di configurazione dei processori più naturale è una *griglia monodimensionale periodica*, di dimensione $1 \times N_{proc}$, dove N_{proc} è il numero totale di processori utilizzati. Come distribuzione dei dati, rispetto alla formulazione matriciale a blocchi (74) si è utilizzato uno *schema ciclico a blocchi di colonne* per la matrice $[K]_3$, ed uno *schema ciclico a blocchi di righe* per la matrice $[K^*]_3$([18]), per bilanciare il carico computazionale fra i processori, essendo la matrice sparsa. Per cui ogni processore, possiede $r = J \times M/Nproc$ righe e $c = N \times N$ colonne

[16] La possibilità di scomporre un'operazione onerosa in (sotto)operazioni più semplici ed indipendenti permette di introdurre il parallelismo in maniera "naturale".

[17] Si osservi che il parallelismo è dunque indotto in maniera "naturale" dal processo fisico del sistema di acquisizione di dati SPECT

[18] Per la matrice $[K^*]_3$ è valida una rappresentazione matriciale a blocchi analoga alla (74).

procedure $[K^*]_3[K]_3 u$

`Ciclo sul numero dei processori.`

for $p = 0, Nproc - 1$

`Calcolo del prodotto parziale:` $\mathtt{f}_p^i \in \Re^{J \cdot M/Nproc}$

passo 1. for $i = 0, \ldots, L$

$\mathbf{f}_p^i = E_i(p \cdot (J \cdot M/nproc) : (p+1) \cdot (J \cdot M/Nproc) - 1; :) \cdot \mathrm{u}^i;$

`Calcolo del prodotto parziale:` $z_p^i \in \Re^{N \cdot N}$

passo 2. for $i = 0, \ldots, L$

$\mathbf{z}_p^i = E_i^*(:, p \cdot (J \cdot M/nproc) : (p+1) \cdot (J \cdot M/Nproc) - 1) \cdot \mathrm{f}_p^i;$

endfor

`Operazione collettiva di tipo somma globale.`

$\mathrm{z}^i = \sum_{p=0}^{Nproc-1} \mathbf{z}_p$

end procedure

Figura 10: *Schema dell'algoritmo parallelo, nel modello fully 3D, per il prodotto* $z = [K^*]_3[K]_3 u$.

di $[E_i]_2$. Analogamente ogni processore della griglia, possiede $r = N \times N$ righe e $c = J \times M/Nproc$ colonne di $[E_i]_2^*$. Con la distribuzione dei dati effettuata, il prodotto (72), viene eseguito in parallelo attraverso i tre seguenti passi:

1. *proiezione*: $\mathbf{f}^i = \sum_{j=0}^{L-1} [E_{(j+i-1} \mathrm{mod} L)]_2 \cdot \mathbf{u}^{j+1}$, $i = 1, \ldots, L$;
2. *retroproiezione*: $\mathbf{z}^i = \sum_{j=0}^{L-1} [E^*_{(j+i-1} \mathrm{mod} L)]_2 \cdot \mathbf{f}^{j+1}$, $i = 1, \ldots, L$;
3. *somma collettiva*: $\mathbf{z}^i = \sum_{p=0}^{Nproc-1} \mathbf{z}_p$

Uno schema dell'algoritmo che esegue il prodotto (72) in parallelo è illustrato in figura 10.

6.2 Strategia di parallelizzazione nel modello *2D+1*

Anche per il modello *2D+1* è possibile una formulazione a blocchi per il prodotto (72). Infatti, la (22) esprime che una componente g^l di $[g]$ di dimensione $J \times M$ si ottiene effettuando L' prodotti con la matrice $[E]_2$ e poi sommando i vettori così ottenuti, moltiplicati per opportuni *pesi* $\alpha_{l'-l}$, ovvero:

$$\begin{pmatrix} \mathbf{g}^0 \\ \mathbf{g}^1 \\ \vdots \\ \mathbf{g^{L-1}} \end{pmatrix} = \begin{pmatrix} \alpha_0[E]_2 & \alpha_1[E]_2 & \dots & \alpha_{L'-1}[E]_2 \\ \alpha_{-1}[E]_2 & \alpha_0[E]_2 & \dots & \alpha_{L'-1}[E]_2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{1-L}\,[E]_2 & \alpha_{2-L}\,[E]_2 & \dots & \alpha_0[E]_2 \end{pmatrix} \begin{pmatrix} \mathbf{u}^0 \\ \mathbf{u}^1 \\ \vdots \\ \mathbf{u^{L'-1}} \end{pmatrix} \quad . \tag{75}$$

Assunta la partizione (50), la (75) è valida per ciascun $[K_i]_3$, ovvero:

$$\begin{pmatrix} \left(\mathbf{g}^0\right)_i \\ \left(\mathbf{g}^1\right)_i \\ \vdots \\ \left(\mathbf{g}^{L'-1}\right)_i \end{pmatrix} = \begin{pmatrix} \alpha_0\left([E]_2\right)_i & \alpha_1\left([E]_2\right)_i & \dots & \alpha_{L'-1}\left([E]_2\right)_i \\ \alpha_{-1}\left([E]_2\right)_i & \alpha_0\left([E]_2\right)_i & \dots & \alpha_{L'-2}\left([E]_2\right)_i \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{1-L'}\left([E]_2\right)_i & \alpha_{2-L'}\left([E]_2\right)_i & \dots & \alpha_0\left([E]_2\right)_i \end{pmatrix} \begin{pmatrix} \mathbf{u}^0 \\ \mathbf{u}^1 \\ \vdots \\ \mathbf{u}^{L'-1} \end{pmatrix} \quad i = 1, \dots, n \quad . \tag{76}$$

6.2.1 Dettagli implementativi nella strategia del *2D+1*

A partire da tale riformulazione, sono state sviluppate due strategie di parallelizzazione per il calcolo del prodotto (76) che si descriveranno nel seguito. La differenza sostanziale fre le due strategie, è che la seconda tiene conto della struttura a banda della matrice $[A]_1$ (figure 12 e 13).

Per entrambe le strategie si è assunta come topologia di configurazione dei processori una *griglia monodimensionale periodica*, di dimensione $1 \times N_{proc}$.

6.2.2 Prima Strategia

A ciascun processore p, sono assegnati $L_{loc} = L/N_{proc}$ blocchi di colonne di $[K_i]_3$, pertanto il calcolo del prodotto (76) è eseguito in parallelo in due passi:

1. *proiezione*: $\left(\left(\mathbf{g}^j\right)_i\right)_p = \sum_{k=p*L'_{loc}+1,\dots,(p+1)*L'_{loc}} \alpha_{j-k}\left([E]_2\right)_i \mathbf{u}^k \quad j = 1, \dots, L;$
2. *somma collettiva*: $\mathbf{g}^j = \sum_{p=0}^{N_{proc}} ((\mathbf{g}^j)_i)_p \quad , \quad j = 1, \dots, L.$

L'algoritmo parallelo, relativo a tale prodotto è illustrato in figura 11.
In maniera analoga, è stato progettato l'algoritmo parallelo per il calcolo di $\mathbf{u} = [K_i^*]_3\,\mathbf{g}$.

6.2.3 Seconda Strategia

A ciascun processore p sono assegnati $L'_{loc} = L'/N_{proc}$ vettori $\mathbf{u}^j$, $L_{loc} = L/N_{proc}$ vettori $\left(\mathbf{g}^i\right)_k$ ed L'_{loc} blocchi di colonne di $[K_i]_3$. Per ogni coppia (p, q) di processori si definisce l'insieme:

$$E_p^q \quad = \quad \left\{ i \in I^q : A_i^j \neq 0 \quad , \forall j \in J^p \right\}, \tag{77}$$

procedure $[K_i]_3 u$

passo 1. for $j = p * L_{loc} + 1, \ldots, (p+1) * L_{loc}$

$p = ([E]_2)_k \, \mathbf{f}_j$

for $i = 1, 2, \ldots, L$

$(\mathbf{g}^i)_k = (\mathbf{g}^i)_k + \alpha_{j-i} p$

endfor

endfor

passo 2. for $i = 1, 2, \ldots, L$

`Operazione collettiva di tipo somma globale`

$\mathbf{g}^i = \sum_{p=0}^{N_{proc}} (\mathbf{g}^i)_p$

end procedure

Figura 11: *Schema dell'algoritmo parallelo mediante Prima Strategia, nel modello 2D+1, per il calcolo di* $\mathbf{g} = [K_i]_3 \, \mathbf{u}$.

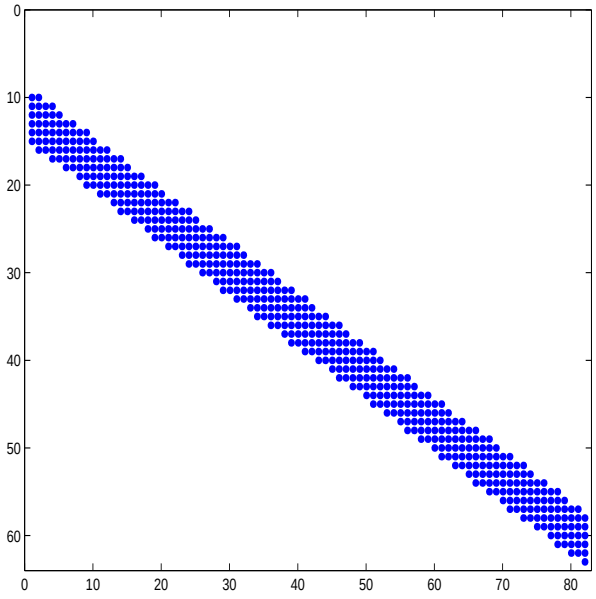


Figura 12: *Schematizzazione della sparsità di* $[A]_1$.

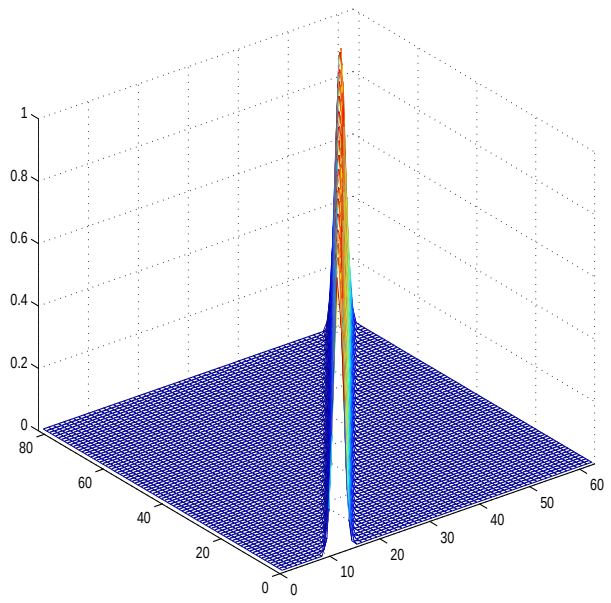


Figura 13: *Andamento degli elementi di* $[A]_1$.

dove I^p ed J^p indicano rispettivamente gli insiemi degli indici dei vettori $\mathbf{u}^j$ e $\left(\mathbf{g}^i\right)_k$ assegnati al processore p. L'insieme E_p^q contiene quindi, gli indici dei vettori $\left(\mathbf{g}^i\right)_k$ assegnati al processore q al cui calcolo contribuiscono, i vettori $\mathbf{u}^j$ assegnati al processore p.
Un esempio della distribuzione dei dati effettuata è illustrata in figura 14.
Il calcolo del prodotto (76), è eseguito in parallelo in due passi:

1. *proiezione:*

   1.a *Calcolo di* $\left(\mathbf{g}^i\right)_k$ *dell'insieme* I^p*:*

   $$\left(\mathbf{g}^i\right)_k = \sum_{j \in J^p} \alpha_{i-j} \left([E]_2\right)_k \mathbf{u}^j, \quad \forall i \in I^p$$

   1.b *Contributo a* $\left(\mathbf{g}^i\right)_k$ *rispetto all'insieme* E_p^q*:*

   $$\left(\left(\mathbf{g}^i\right)_k\right)_p = \sum_{j \in J^p} \alpha_{i-j} \left([E]_2\right)_k \mathbf{u}^j, \quad \forall q : E_p^q \neq \emptyset$$

2. *somma tra processori "vicini":*

   2.a *spedizione*: di $\left(\left(\mathbf{g}^i\right)_k\right)_p$ al processore q, $\forall i \in E_p^q$.

   2.b *ricezione*: di $\left(\left(\mathbf{g}^i\right)_k\right)_q$ dal processore q, $\forall q : E_q^p \neq \emptyset$

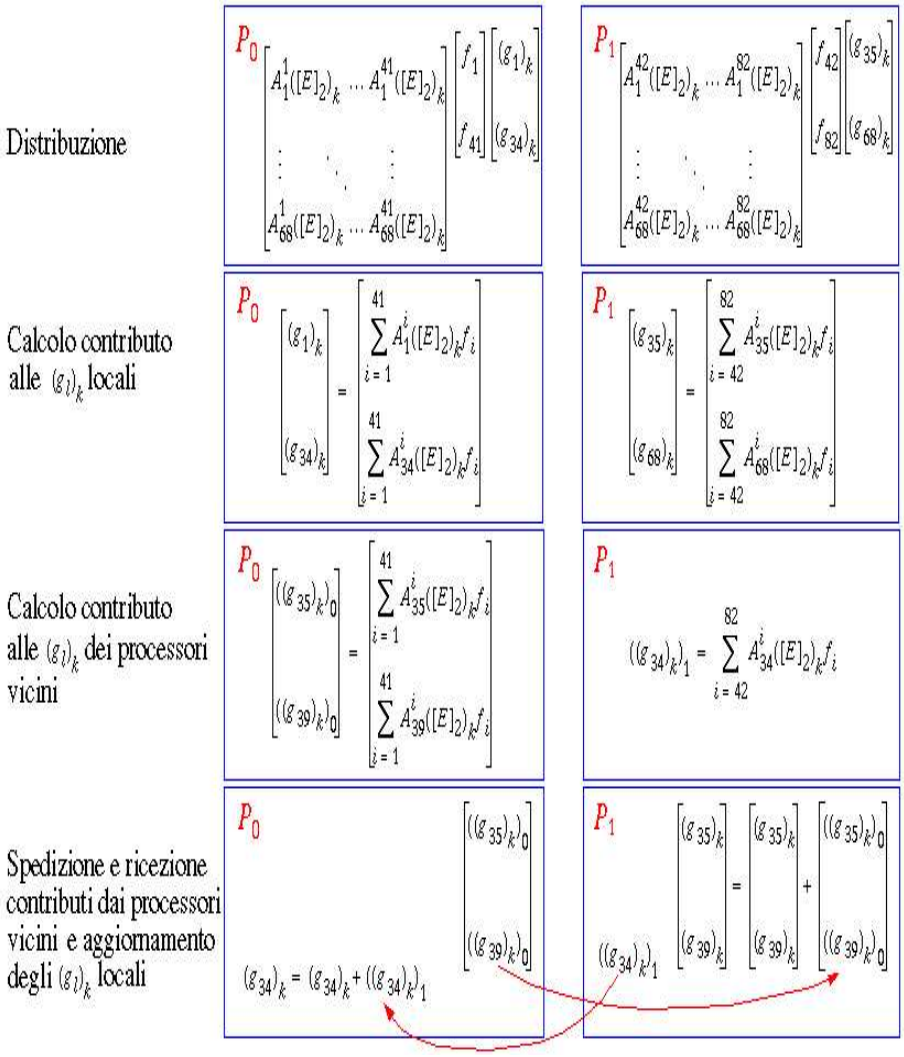


Figura 14: *Esempio di distribuzione di dati per la seconda strategia di parallelizzazione modello 2D+1, con* $L' = 82$, $L = 68$ *e* $N_{proc} = 2$.

2.c *somma locale*:

$$\left(\mathbf{g}^i\right)_k = \left(\mathbf{g}^i\right)_k + \sum_{q:E_q^p \neq \emptyset} \left(\left(\mathbf{g}^i\right)_k\right)_q, \quad \forall i \in E_q^p$$

Uno schema dell'algoritmo parallelo relativo a tale strategia è illustrato in figura 15.
In maniera analoga, è stato realizzato l'algoritmo parallelo per il calcolo di $\mathbf{u} = [K_i^*]_3 \, \mathbf{g}$.

6.2.4 Un confronto tra Prima e Seconda Strategia di parallelizzazione del *2D+1*

Si consideri la griglia di processori di dimensione $1 \times N_{proc}$. Nel seguito è illustrato un confronto fra le due strategie di parallelizzazione adottate per il modello *2D+1* sia in termini di N_{calc} e N_{com} sia in termini di speed-up.
Osservando le due strategie di parallelizzazione proposte, si osserva che in entrambi i casi il prodotto (76) è realizzato in due passi, denominati:

1. *proiezione*;
2. *somma*.

Nella Prima Strategia per effettuare il passo 1, ogni processore p, calcola in parallelo, un contributo ad L componenti del vettore $\mathbf{g}$, cioè a ciascuna delle L proiezioni $(\mathbf{g}^i)_k$, $i = 1, \ldots L$, mediante un prodotto matrice-vettore a blocchi con un costo computazionale di $\mathcal{O}((L \times L_{loc}) \times (J \times M) \times (N \times N))$([19]), pertanto il costo totale è:

$$N_{calc}^{(1.Prima)} = \mathcal{O}(N_{proc} \times (L \times L_{loc}) \times (J \times M) \times (N \times N)) \quad . \tag{78}$$

Il passo 2 viene realizzato mediante un'operazione di *somma globale*, con un costo computazionale([20]) di:

$$N_{calc}^{(2.Prima)} = \mathcal{O}(N_{proc} \times log_2(N_{proc}) \times (J \times M) \times L) \quad , \tag{79}$$

ed un numero di comunicazioni([21]) uguale a:

$$N_{com}^{Prima} = \mathcal{O}(N_{proc} \times log_2(N_{proc}) \times (J \times M) \times L) \quad , \tag{80}$$

[19] Il prodotto matrice-vettore coinvolge una matrice locale di dimensione $L \times (J \times M) \times (N \times N) \times L_{loc}$ ed un vettore locale di dimensione $(N \times N) \times L_{loc}$.

[20] Si ipotizza che l'operazione di somma globale, tra i vettori $(\mathbf{g})_p$ di lunghezza $(J \times M) \times L$, sia eseguita dagli N_{proc} processori secondo uno schema di comunicazione ad albero con profondità $log_2 N_{proc}$. In ciascun livello dell'albero gli N_{proc} processori si comunicano a due a due il proprio vettore $(\mathbf{g})_p$, e lo aggiornano mediante un'operazione di somma. Pertanto ad ogni livello dell'albero vengono eseguite N_{proc} somme tra vettori di lunghezza $(J \times M) \times L$. L'ipotesi è attendibile, in quanto nel software l'operazione di somma globale è realizzata mediante la routine di MPI, `MPI_ALLREDUCE`.

[21] In ciascun livello dell'albero avvengono N_{proc} comunicazioni, pertanto il numero di comunicazioni effettuate nell'operazione di *somma globale* è uguale a $N_{proc} \times log_2 N_{proc}$.

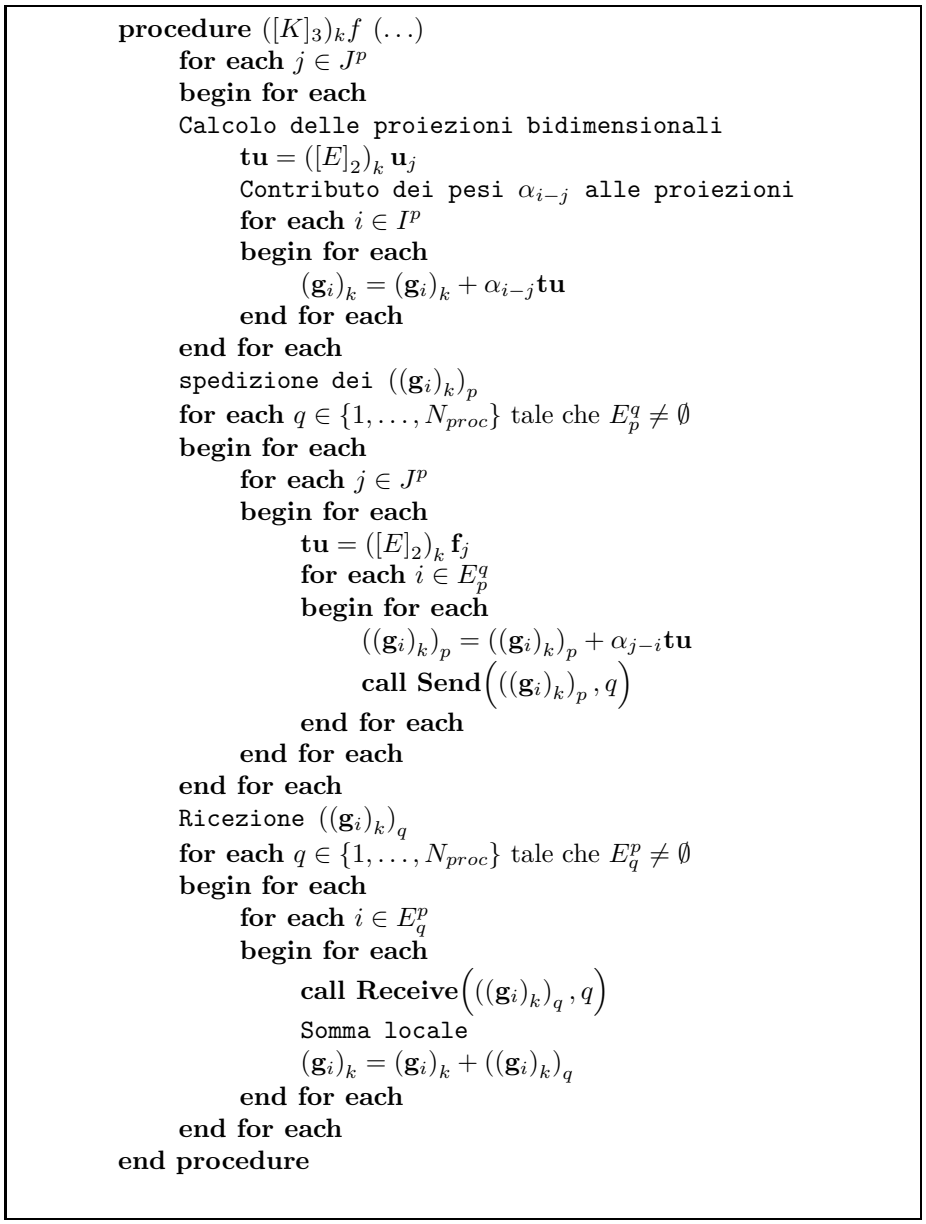


Figura 15: *Schema dell'algoritmo parallelo mediante Seconda Strategia, nel modello 2D+1, per il calcolo di* $\mathbf{g} = [K_i]_3\, \mathbf{u}$.

perché in ciascuna comunicazione avviene uno scambio a due a due fra gli N_{proc} processori, del vettore $((\mathbf{g})_k)_p$ di dimensione $L \times (J \times M)$.
Nella Seconda Strategia invece, per effettuare il passo 1, ogni processore p calcola $|J^p|$([22]) componenti del vettore $\mathbf{g}$, cioè calcola le proiezioni $(\mathbf{g}^i)_k$ con $i \in J^p$, più un contributo ad altre $|E_p^q|$ componenti del vettore $\mathbf{g}$, ovvero alle altre proiezioni $(\mathbf{g}^i)_k$ con $i \in E_p^q$, mediante un prodotto matrice-vettore a blocchi con un costo computazionale di $\mathcal{O}((|J^p|+|E_p^q|)\times(J\times M)\times(N\times N))$. Si osservi che $|J^p| = L_{loc}$ pertanto il costo totale è:

$$N_{calc}^{(1.Seconda)} = \mathcal{O}(N_{proc} \times ((L_{loc} + |E_p^q|) \times L_{loc}^{'}) \times (J \times M) \times (N \times N)) \quad . \quad (81)$$

Il passo 2, viene effettuato mediante una somma tra al più $|E_p^q|$ processori adiacenti nella griglia definita, con un costo computazionale di:

$$N_{calc}^{(2.Seconda)} = \mathcal{O}(N_{proc} \times |E_p^q| \times (J \times M) \times |E_p^q|) \quad , \quad (82)$$

ed un numero di comunicazioni uguale a:

$$N_{com}^{(Seconda)} = \mathcal{O}(N_{proc} \times |E_p^q| \times (J \times M) \times |E_p^q|) \quad , \quad (83)$$

perché in ciascuna comunicazione avviene uno scambio tra i processori p e q tale che $|E_p^q| \neq 0$, di sottovettori di $(g)_k$ di dimensione $|E_p^q| \times (J \times M)$.

Nell'ipotesi che $L = L'$ e quindi che $L_{loc} = L'_{loc}$, si osserva che:

- la (78) e la (81) differiscono perché nella prima è presente il fattore $(L \times L_{loc})$ mentre nella seconda il fattore $(L_{loc} + |E_p^q|) \times L_{loc})$, considerando che al più $|E_p^q| = |J^q| = L_{loc}$ si ha:

  $$L \times L_{loc} = \frac{L^2}{N_{proc}} > 2\frac{L^2}{N_{proc}^2} = (L_{loc} + |E_p^q|) \times L_{loc}) \quad ,$$

  pertanto:

  $$\boxed{N_{calc}^{(1.Prima)} > N_{calc}^{(1.Seconda)}} \quad (84)$$

- La (79) e la (82) differiscono per la presenza del fattore $log_2 N_{proc}$ nella prima e del fattore $|E_p^q|$, per le osservazioni del punto precedente e considerando che all'aumentare di N_{proc}, $log_2 N_{proc}$ aumenta, mentre L_{loc} diminuisce, si può dedurre che:

  $$\boxed{N_{calc}^{(2.Prima)} > N_{calc}^{(2.Seconda)}} \quad (85)$$

[22] Nel seguito, col simbolo $|\mathcal{I}|$, si indicherà la cardinalità dell'insieme $|\mathcal{I}|$.

- Osservando che $N_{calc}^{(2.Prima)} = N_{com}^{(Prima)}$ e che $N_{calc}^{(2.Seconda)} = N_{com}^{(Seconda)}$, dal risultato (85) si può affermare che:

$$\boxed{N_{com}^{(Prima)} > N_{com}^{(Seconda)}} \tag{86}$$

Si consideri ora, lo speed-up $S_{N_{proc}}$:

$$S_{N_{proc}} = \frac{T_1}{T_{N_{proc}}} = \frac{N \times t_{calc}}{N_{calc} \times t_{calc} + T_{com} \times t_{com}} \quad , \tag{87}$$

dove T_1 e $T_{N_{proc}}$ indicano rispettivamente, il tempo di esecuzione su un processore e su N_{proc} processori, mentre N ed N_{calc} il numero di operazioni effettuate da uno e da N_{proc} processori, t_{calc} è il tempo di esecuzione di un'operazione floating point, t_{com} è il tempo per la comunicazione di un dato floating point tra due processori.
Il prodotto (76), ha il seguente costo computazionale:

$$N_{calc} = \mathcal{O}((L \times L) \times (J \times M) \times (N \times N)) \quad , \tag{88}$$

pertanto:

$$T_1 = [(L \times L) \times (J \times M) \times (N \times N)] \times t_{calc} \quad . \tag{89}$$

Se per semplicità di notazione si suppone $J = M = N$ allora, la (89) si semplifica come segue:

$$T_1 = [L^2 \times N^4] \times t_{calc} \quad . \tag{90}$$

$T_{N_{proc}}$ per le due strategie di parallelizzazione assume il seguente valore:

$$T_{N_{proc}}^{(*)} = [N_{calc}^{(1.*)} + N_{calc}^{(2.*)}] \times t_{calc} + [N_{com}^{(*)}] \times t_{com} \quad , \tag{91}$$

dove l'apice "*" indica il termine "*Prima*" o "*Seconda*" strategia, in accordo con le notazioni usate in precedenza.

Sostituendo (78), (79) e (80) nella (91) si ottiene:

$$\begin{aligned} T_{N_{proc}}^{(Prima)} &= [N_{proc} \times (L \times L_{loc}) \times (J \times M) \times (N \times N) + \\ &+ N_{proc} \times log_2(N_{proc}) \times (J \times M) \times L] \times t_{calc} + \\ &+ [N_{proc} \times log_2(N_{proc}) \times (J \times M) \times L] \times t_{com} \quad . \end{aligned} \tag{92}$$

Si ipotizzi per semplicità di calcolo che $t_{com} = 2t_{calc}$ e che $J = M = N$, allora la (92) si semplifica nel seguente modo:

$$T_{N_{proc}}^{(Prima)} = [(N^2 \times L) + (N^2 \times L + 3N_{proc} \times log_2 N_{proc})] \times t_{calc} \quad , \tag{93}$$

da cui:

$$S_{N_{proc}}^{Prima} = \frac{L^2 \times N^4}{L^2 \times N^4 + L \times N^2 \times 3N_{proc} \times log_2 N_{proc}} = \\ = \frac{L \times N^2}{L \times N^2 + 3N_{proc} \times log_2 N_{proc}} \quad . \tag{94}$$

Dalla (94), mantenendo costante N ed L ed aumentando N_{proc}, lo speed-up degrada.
Sostituendo invece, (81), (82) e (83) nella (91) si ottiene:

$$T_{N_{proc}}^{(Seconda)} = [N_{proc} \times ((L_{loc} + |E_p^q|) \times L'_{loc}) \times (J \times M) \times (N \times N) + \tag{95}$$
$$+ \quad N_{proc} \times |E_p^q|) \times (J \times M) \times |E_p^q|] \times t_{calc} +$$
$$+ \quad [N_{proc} \times |E_p^q| \times (J \times M) \times |E_p^q|] \times t_{com} \quad .$$

Si ipotizzi per semplicità di calcolo che $t_{com} = 2t_{calc}$, che $J = M = N$ e $L = L'$ e considerando che al più $|E_p^q| = |J^q| = L_{loc}$, allora con semplici calcoli, la (95) si semplifica nel seguente modo:

$$T_{N_{proc}}^{(Seconda)} = [2 \times \frac{L^2}{N_{proc}} \times N^4 + 3\frac{L^2}{N_{proc}} \times N^2] \times t_{calc} \quad , \tag{96}$$

da cui:

$$S_{N_{proc}}^{Seconda} = \frac{L^2 \times N^4}{2 \times \frac{L^2}{N_{proc}} \times N^4 + 3\frac{L^2}{N_{proc}} \times N^2} = \\ = \frac{N^2 \times N_{proc}}{2(N^2 + 3)} \quad . \tag{97}$$

Dalla (97), mantenendo costante N ed L ed aumentando N_{proc}, lo speed-up aumenta.
Con la Prima Strategia all'aumentare del numero N_{proc} di processori, lo speed-up degrada, mentre con la Seconda Strategia lo speed-up cresce linearmente con N_{proc}, pertanto è possibile affermare che la Seconda Strategia di parallelizzazione per l'esecuzione del prodotto (76), scala meglio all'aumentare del numero dei processori. Alcuni risultati sperimentali sono presentati in [16].

7 Il software parallelo MEDITOMO

Questo paragrafo è dedicato alla descrizione dei moduli di software parallelo di MEDITOMO, uno schema è illustrato in tabella 2. I moduli software si possono

MEDITOMO Library	
`Acquisition Models`	
Fully 3D	*2D+1*
`Reconstruction Algorithms`	`Reconstruction Algorithms`
⋆ FP (TV) ⋆ CG ⋆ EMOS ⋆ EMOS-TV	⋆ FP (TV) ⋆ CG ⋆ EMOS

Tabella 2: La libreria MEDITOMO

logicamente riorganizzare, in base al tipo di problema risolto, al modello di acquisizione dati, agli algoritmi di ricostruzione e alla geometria del sistema di acquisizione, mediante lo schema che segue in cui il nome([23]) del modulo software è riportato in parentesi:

- Ricostruzione dati 3D SPECT
  - Modello di approssimazione: *fully 3D*
    - Geometria del collimatore *di tipo fan*
      - Algoritmo di ricostruzione *Gradiente Coniugato* (`3dtomo_fan_cg`)
      - Algoritmo di ricostruzione *Punto Fisso* (`3dtomo_fan_fp+tv`)
      - Algoritmo di ricostruzione *EMOSn* (`3dtomo_fan_em`)
      - Algoritmo di ricostruzione *EMOSn* con regolarizzazione TV (`3dtomo_fan_em+tv`)
    - Geometria del collimatore *di tipo parallelo*
      - Algoritmo di ricostruzione *Gradiente Coniugato* (`3dtomo_par_cg`)
  - Modello di approssimazione: *2D+1*
    - Geometria del collimatore *di tipo fan*
      - Algoritmo di ricostruzione *Gradiente Coniugato* (`2d+1tomo_fan_cg`)
      - Algoritmo di ricostruzione *Punto Fisso* (`2d+1tomo_fan_fp+tv`)
      - Algoritmo di ricostruzione *EMOSn* con regolarizzazione TV (`2d+1tomo_fan_em+tv`)

[23] `XXtomo_YYY_ZZ` è il prototipo del nome di un modulo software di MEDITOMO dove:

- `XX` indica il modello di ricostruzione dei dati SPECT (`3d` o `2d+1`);
- `YYY` indica la geometria del sistema di acquisizione dei dati (parallela `par` o a ventaglio `fan`);
- `ZZ` indica l'algoritmo di ricostruzione (gradiente coniugato `cg`, expectation maximization `em` o punto fisso `fp+tv`);

Il software sviluppato è stato scritto nel linguaggio di programmazione **FORTRAN 77**. Per il modello *2D+1* esiste una versione ottimizzata del modulo `2d+1tomo_fan_cg`, scritta in linguaggio **Ansi C**. La librearia utilizzata per l'ambiente di comunicazione è **M**essage **P**assing **I**nterface (MPI).

8 Risultati numerici

In questa sezione si riportano alcuni dei risultati ottenuti con il software MEDITOMO. In particolare sono stati confrontati i risultati ottenuti, tra le seguenti coppie di moduli:

1. `3dtomo_fan_cg` e `3dtomo_fan_fp+tv`;
2. `3dtomo_fan_em` e `3dtomo_fan_em+tv`;
3. `2d+1tomo_fan_em` e `2d+1tomo_fan_cg`.

I confronti dei punti 1 e 2 sono stati effettuati per evidenziare gli effetti della regolarizzazione TV, ([3] [4] [5]), rispetto agli algoritmi che non ne fanno uso, mentre il confronto 3 è stato effettuato per comparare le ricostruzioni ottenute con i due algoritmi di base presentati: EM e CG.
I test sono stati effettuati sia su immagini sintetiche per testarne efficienza ed accuratezza, sia su dati reali per testarne l'effettivo utilizzo in ambiente clinico. Pertanto i risultati ottenuti sono stati organizzati in due gruppi:

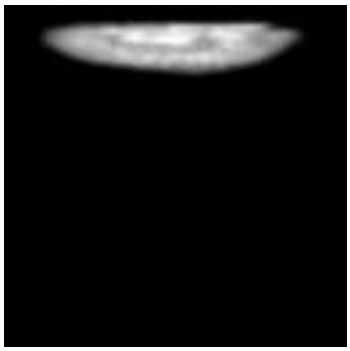
- dati sintetici in sezione 8.1, ovvero proiezioni ottenute da fantoccio di Hoffman, di cui riportiamo una sezione in figura 16;
- dati reali acquisiti in ospedale in sezione 8.3, ovvero proiezioni ottenute da un paziente sottoposto ad un reale sistema di acquisizione SPECT (CERASPECT)([24]).

8.1 Dati sintetici.

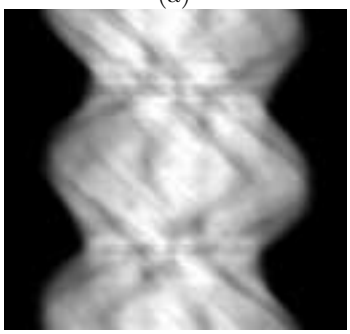
L'insieme delle proiezioni è stato generato simulando il sistema di acquisizione di dati SPECT, ovvero la matrice $[K]_3$, mediante una gaussiana [9] sia nella direzione del piano xy, sia nella direzione dell'asse z. Per la (2) si è assunto $\sigma_{min} = 3$, $\sigma_{max} = 8$, $r = 192$. Con tale simulazione sono state generate 120 sezioni sinogramma di dimensione 128×128, dove 120 rappresenta il numero di viste angolari nell'intervallo $[0, \pi)$.

[24] I dati reali sono stati gentilmente concessi dall'ospedale Careggi di Firenze.

Figura 16: *Una sezione del fantoccio di Hoffman.*



(a)



(b)

Figura 17: Una sezione proiezione (a) ed un sinogramma (b)

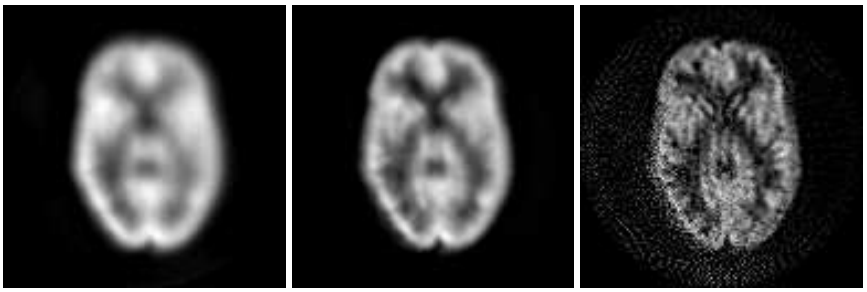


Figura 18: *Sezione del fantoccio di Hoffmann ricostruita con* `3dtomo_fan_cg`*, con numero di iterazioni CG uguale a* 7*,* 21*,* 63*.*

8.1.1 Ricostruzioni ottenute con `3dtomo_fan_cg` e `3dtomo_fan_fp+tv`

In figura 19 è illustrato l'andamento del *Restoration Error*, ovvero il rapporto tra l'errore quadratico medio (*Mean Square Error (MSE)*) tra la soluzione ideale e la soluzione calcolata, e la soluzione ideale:

$$RE = \frac{\| [u]_{true} - [u^*] \|_2}{\| [u]_{true} \|_2} \quad , \tag{98}$$

ottenuto con i moduli software `3dtomo_fan_cg` e `3dtomo_fan_fp+tv`, al variare del numero di iterazioni([25]). Si osserva che il numero ottimale di iterazioni è 10-12.
In figura 18 è riportata la ricostruzione effettuata da `3dtomo_fan_cg`, della sezione del fantoccio di Hoffman presa in esame. Mentre in figura 20 è riportata la ricostruzione effettuata da `3dtomo_fan_fp+tv`.
In figura 21 è illustrato l'andamento dei profili([26]) lineari della sezione di Hoffman presa in esame ricostruita da `3dtomo_fan_fp+tv` e da `3dtomo_fan_cg`.

8.1.2 Ricostruzioni ottenute con `3dtomo_fan_em` e `3dtomo_fan_em+tv`

In figura 24 è illustrato l'andamento del *Restoration Error* (RE), relativo alla ricostruzione mediante l'algoritmo in figura 8 con 10 passi di EM e livello di OS pari a 10, per alcuni valori del parametro di regolarizzazione α.
In figura 22 sono illustrate le ricostruzioni della sezione di Hoffman presa in esame, con e senza regolarizzazione per alcuni valori del numero di iterazioni di EM. Le immagini ottenute con regolarizzazione TV sono relative al valore $\alpha = 0.04$ del parametro di regolarizzazione.

[25] Per un confronto omogeneo tra i risultati ottenuti con i due moduli software, sono state riportate per `3dtomo_fan_fp+tv` il numero totale di iterazioni it_{tot}, ovvero il numero di iterazioni sul ciclo esterno FP, it_{FP} per il numero di iterazioni del ciclo interno CG, it_{CG} ovvero $it_{tot} = it_{FP} \times it_{CG}$.

[26] Il profilo di una sezione è la scala dei grigi di una colonna o di una riga della sezione stessa.

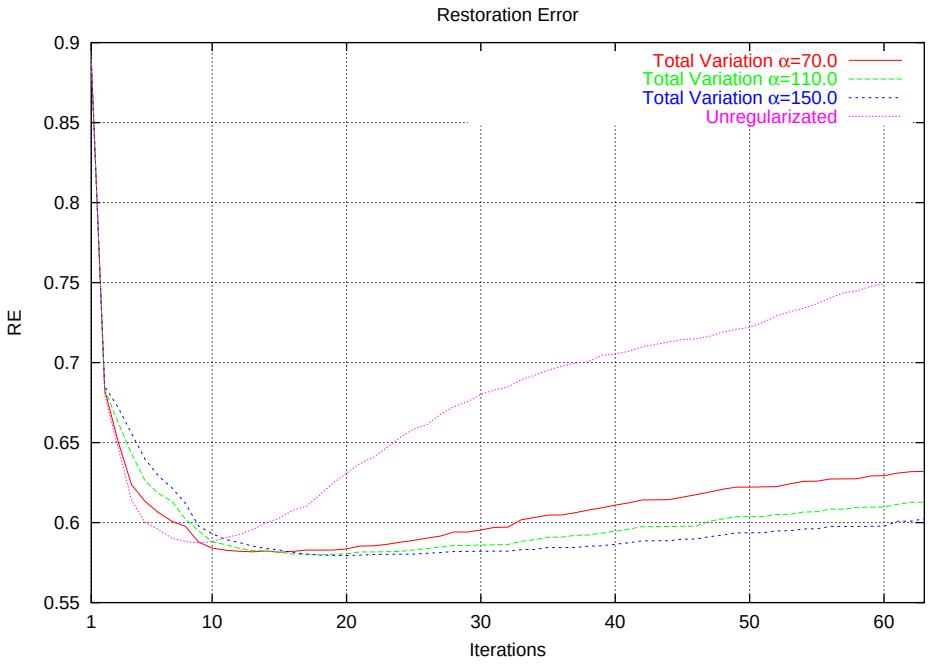
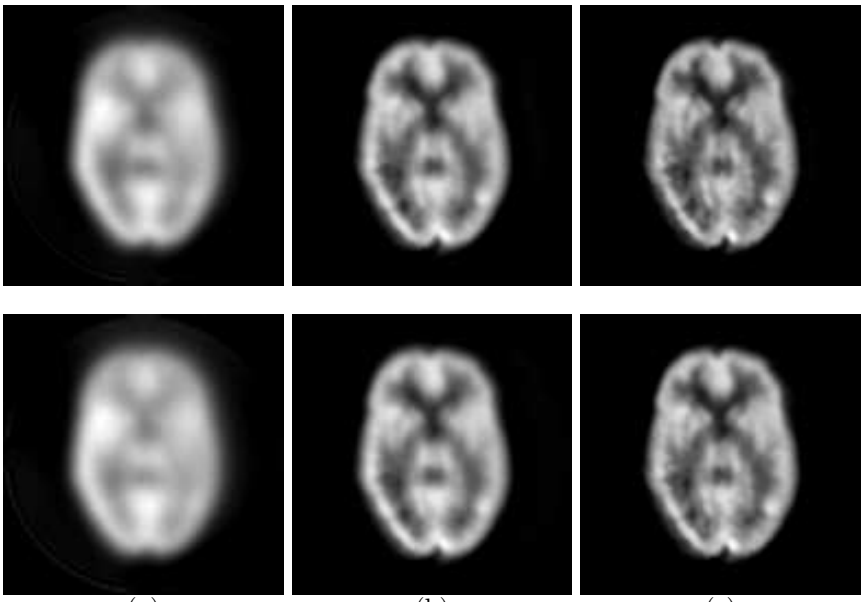


Figura 19: *Andamento di RE relativo alle ricostruzioni ottenute con* `3dtomo_fan_cg` *e con* `3dtomo_fan_fp+tv`*, al variare del numero di iterazioni di FP per alcuni valori del parametro di regolarizzazione* α*.*



(a) (b) (c)

Figura 20: *Sezione del fantoccio di Hoffman, ricostruita con* `3dtomo_fan_fp+tv`*, al passo* $m = 1$ *(a), passo* $m = 3$ *(b) ed al passo* $m = 9$ *(c) di FP, con numero di iterazioni CG uguale a* 7 *(Numero totale iterazioni:* 7, 21 63*), con parametro* $\alpha = 70$ *nella prima riga,* $\alpha = 150$ *nella seconda riga.*

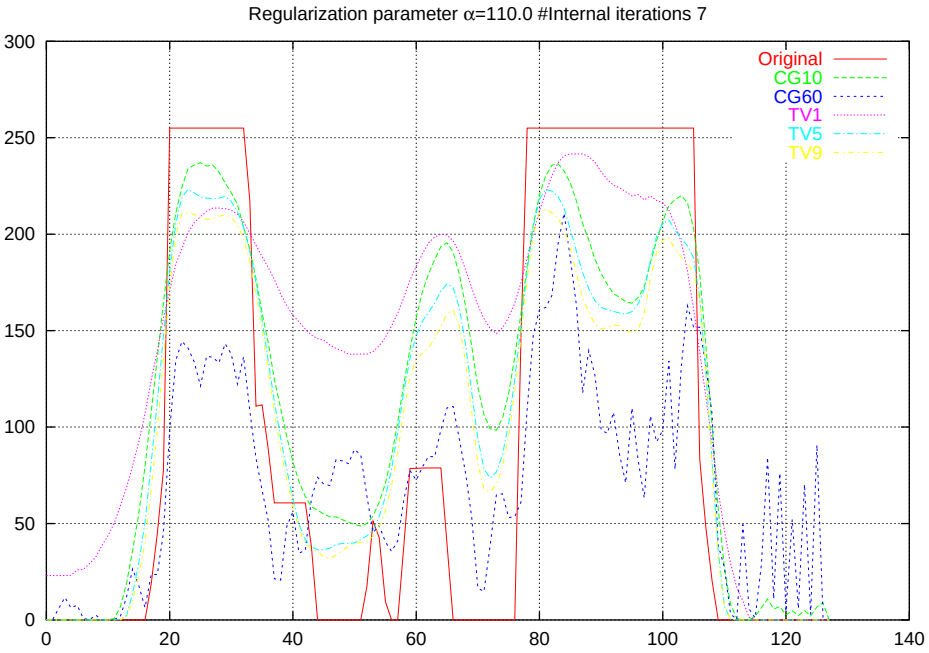


Figura 21: *Andamento dei profili lineari relativo alle ricostruzioni ottenute con* `3dtomo_fan_fp+tv` *e con* `3dtomo_fan_cg`.

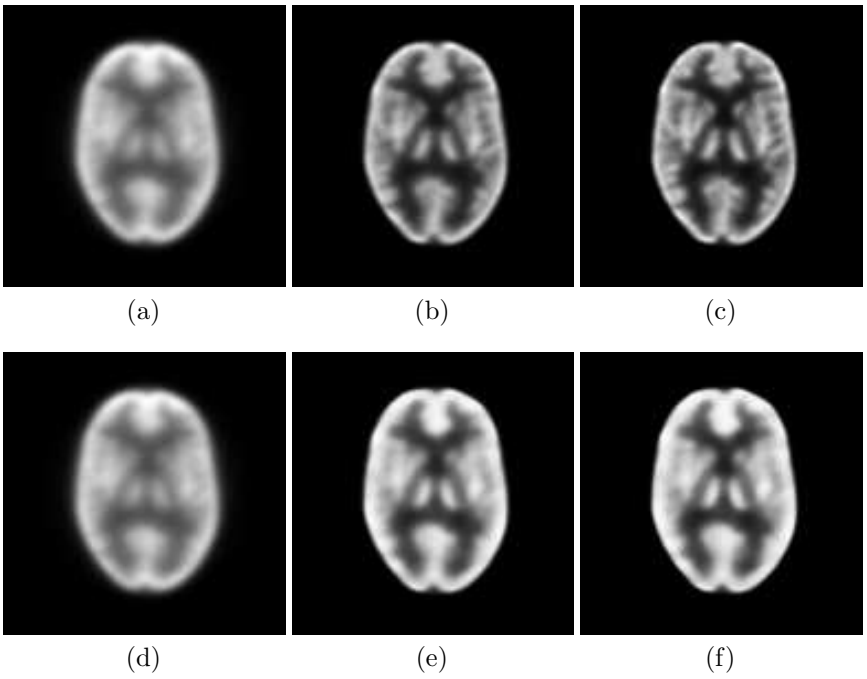


Figura 22: *Sezione del fantoccio di Hoffman, ricostruita con* `3dtomo_fan_em` *al I passo (a), VI passo (b) e X passo (c) di EM. Sezione del fantoccio di Hoffman, ricostruite con* `3dtomo_fan_em+tv` *con parametro di regolarizzazione* $\alpha = 0.04$ *al I passo (d), VI passo (e) e X passo (f) di EM.*

In figura 23 è illustrato l'andamento dei profili lineari di una sezione trasversale del volume ricostruito, per alcuni valori del numero di iterazioni, relativi sia all'algoritmo in figura 8 (a) che alla versione standard di EMOSn (versione non regolarizzata) (b).

8.1.3 Ricostruzioni ottenute con `2d+1tomo_fan_cg` e `2d+1tomo_fan_em`

In figura 25 è illustrato l'andamento del *Restoration Error* (RE), relativo alla ricostruzioni mediante l'algoritmo del PCG in figura 5, e mediante l'algoritmo di EMOS$_n$ in figura 7, per alcuni valori del numero n (`nos`) di *subset*. Si osserva che EM è più accurato di CG, e che tale accuratezza aumenta con l'aumentare del numero n dei *subset*.

8.2 Risultati ottenuti

Riassumendo i risultati ottenuti in 8.1.1 8.1.2 e 8.1.3, si possono trarre le seguenti conclusioni:

1. Come si osserva dai paragrafi 6.1 e 6.2, il modello *2D+1* si presta meglio all'introduzione del parallelismo. Inoltre, (in ambiente clinico) il *2D+1* è preferito al modello *fully 3D* in quanto rappresenta in modo più realistico il sistema di acquisizione di dati SPECT.

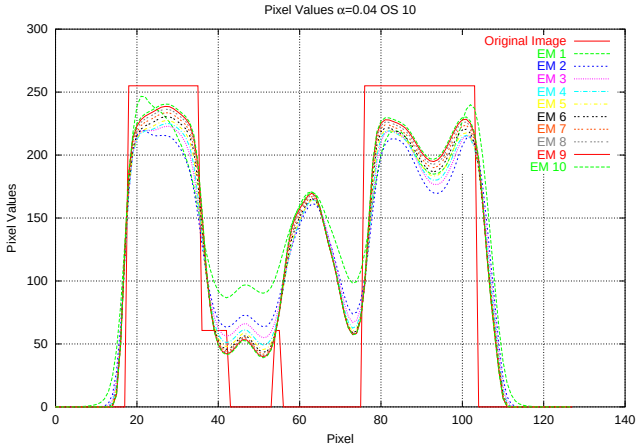
2. Gli algoritmi CG ed EM sono caratterizzati dalla *proprietà di semiconvergenza* [10]: l'RE dopo aver raggiunto il minimo, ad un determinato numero di iterazioni, che si indicherà con it_{opt}, finisce col divergere, con conseguente deterioramento della ricostruzione (come si può osservare in figura 18); al contrario gli algoritmi basati sulla regolarizzazione TV, garantiscono la convergenza per differenti paramentri di regolarizzazione α, come illustrato in figura 19 e 24.

3. Il valore di it_{opt} vale 10 per l'algoritmo CG e 6 per l'algoritmo EM.

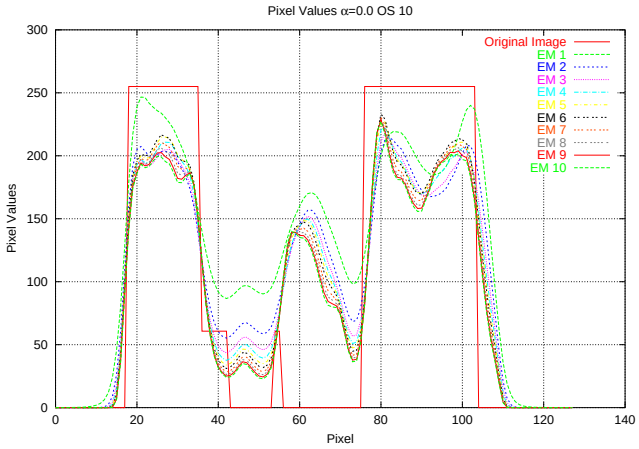
4. L'uso della regolarizzazione TV permette di ottenere una migliore qualità della ricostruzione, infatti gli *edge* delle ricostruzioni sono preservati, come si evidenzia dalle figure 21 e 23 che illustrano le ricostruzioni dei profili della sezione di Hoffman in esame. Inoltre, gli algoritmi di ricostruzione basati sulla regolarizzazione TV, per differenti paramentri di regolarizzazione α, preservano la qualità della ricostruzione anche dopo l'iterazione it_{opt}, non migliorando significativamente l'accuratezza ma nemmeno corrompendola, si osservi tale risultato in figura 20.

5. Scelta dei parametri α e β:

   4.a la scelta dei valori da assegnare al *parametro di regolarizzazione* α è stata effettuata "*trial and error*". Dagli esperimenti numerici effettuati si è osservato che il suo valore ottimale è 70, per i moduli software basati su FP, mentre è 0.04 per i moduli basati su EM;



(a)



(b)

Figura 23: *Andamento dei profili lineari relativo alle ricostruzioni ottenute con* `3dtomo_fan_em+tv` *(a) e di* `3dtomo_fan_em` *(b) per alcuni valori del numero di iterazioni di EM.*

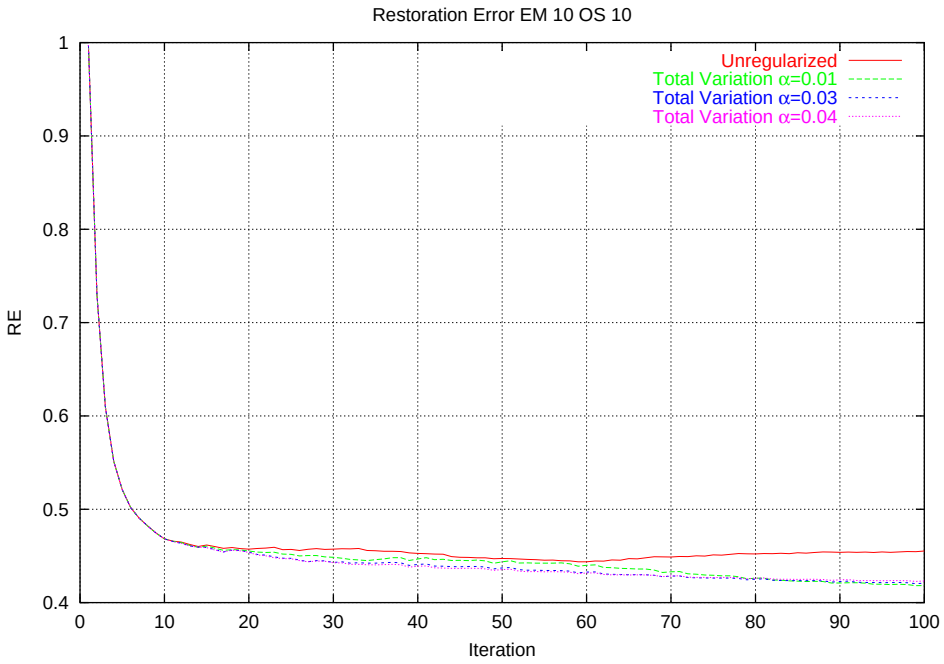


Figura 24: *Andamento di RE relativo alle ricostruzioni ottenute con* `2d+1tomo_fan_em` *e con* `2d+1tomo_fan_em+tv` *al variare del numero di iterazioni di OS per alcuni valori del parametro di regolarizzazione* α.

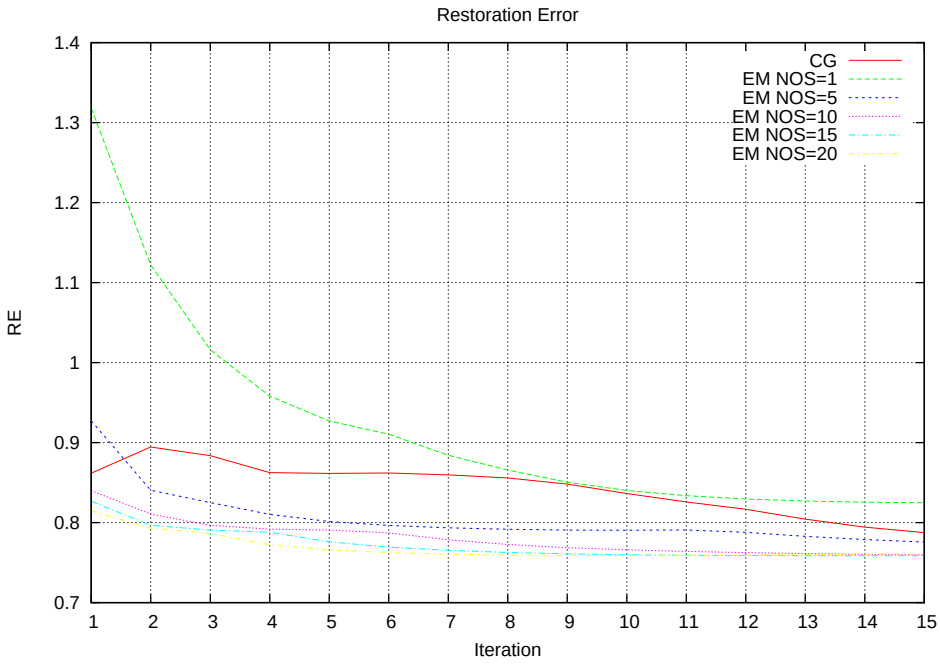


Figura 25: *Andamento di RE relativo alle ricostruzioni ottenute con* `2d+1tomo_fan_cg` *e con* `2d+1tomo_fan_em` *al variare del numero di iterazioni per alcuni valori del numero numero di subset n.*

4.b il parametro di *correzione* β della (17) è stato fissato al valore 0.01 per gli algoritmi basati su CG, ed a 0.001 per gli algoritmi basati su EM. Il valore di β per EM è più piccolo perché EM lavora su immagini normalizzate tra $[0, 1]$, e β "corregge" gli eventuali elementi nulli della matrice del gradiente di $[u]$, senza alterare "significativamente" il loro ordine di grandezza.

6. Dagli esperimenti effettuati si osserva che EM è più accurato di CG, e che tale accuratezza aumenta con l'aumentare del numero n dei *subset*. Si osserva inoltre, in figura 25 che il valore ottimale per n è 15-20.

8.3 Dati reali.

I moduli software presi in esame sono stati testati anche su dati reali, ovvero i moduli software hanno prodotto ricostruzioni di sezioni 2D, a partire da proiezioni di un cervello di un paziente ottenute dal sistema di acquisizione di dati SPECT, CERASPECT, dell'ospedale di Careggi di Firenze.
I test effettuati hanno prodotto risultati sui quali è possibile fare le stesse considerazioni fatte per i risultati ottenuti con dati sintetici, ovvero una migliore qualità dell'immagine e una ricostruzione più enfatizzata degli *edge* dei moduli software che fanno uso della regolarizzazione TV. Si riportano ad esempio in figura 26 le ricostruzioni di una sezione di un cervello ottenute con `3dtomo_fan_cg` e con `3dtomo_fan_fp+tv` con due diversi parametri di regolarizzazione α.

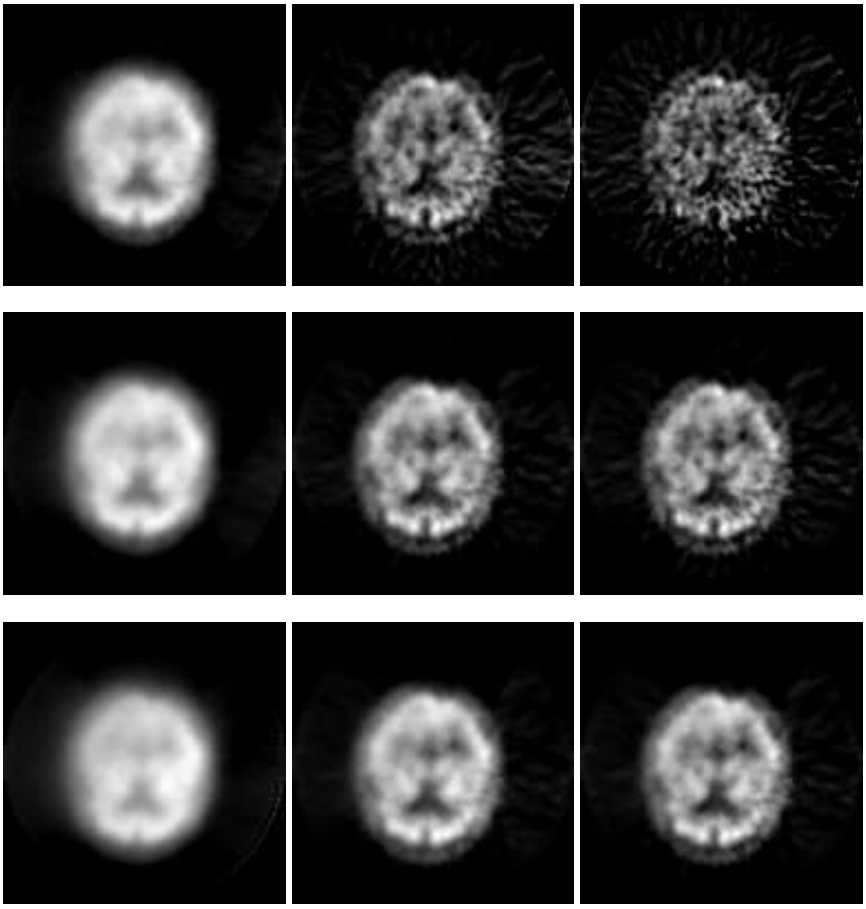
8.4 Prestazioni del software MEDITOMO

Il software MEDITOMO è stato testato su diverse architetture parallele di tipo MIMD a memoria distribuita. Si riportano ad esempio le prestazioni in termini di tempo di esecuzione, Speed-up ed Efficienza del software `3dtomo_fan_fp+tv` rispettivamente in figura 27, 28 e 29, ottenute su un sistema *Beowulf* a 32 nodi. Ogni nodo ha un processore *Pentium II* a 450 Mhz, con *Fast Ethernet 10/100 MBit*. Il tempo di esecuzione di una sola iterazione, si è ridotto da circa 30 minuti, ad un solo minuto. Considerando che il numero ottimale it_{opt} di iterazioni è inferiore a 10, il tempo di ricostruzione di dati 3D SPECT, si riduce globalmente ad una decina di minuti. Tale tempo è perfettamente compatibile con quello della diagnostica medica.

9 Conclusioni

Tale documento si è proposto di illustrare gli obiettivi principali del lavoro svolto:

1. l'introduzione del funzionale di Totale Variazione, come regolarizzazione *edge preserving* nei metodi numerici per la ricostruzione di dati 3D SPECT;



(a) (b) (c)

Figura 26: *Sezione di un cervello umano, prima riga, ricostruita con* `3dtomo_fan_cg` *con numero di iterazioni uguale a* 5, 15 45*; seconda e terza riga, con* `3dtomo_fan_fp+tv` *con* α *rispettivamente uguale a* 50 *e* 70*, al passo* $m = 1$ *(a), passo* $m = 3$ *(b) ed al passo* $m = 5$ *(c) di FP, con numero di iterazioni CG uguale a* 5 *(Numero totale iterazioni,* 5, 15 45*).*

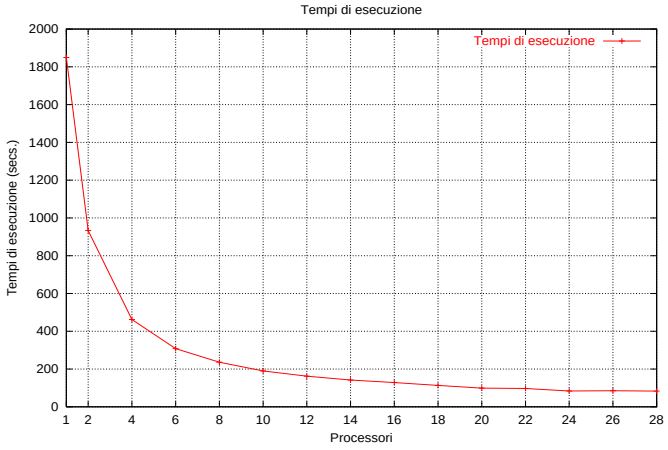


Figura 27: *Tempo di esecuzione di* `3dtomo_fan_fp+tv` *in funzione del numero di processori.*

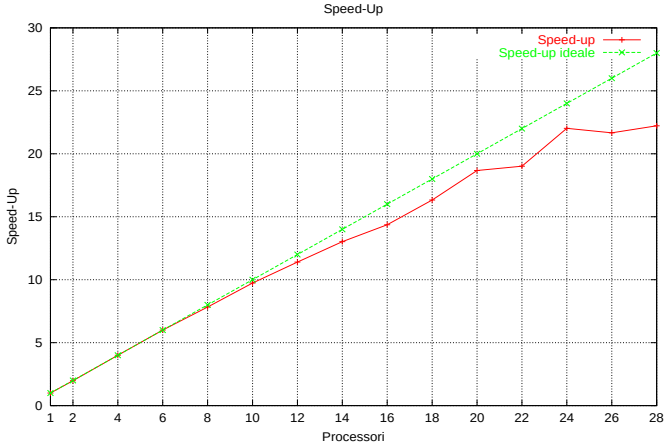


Figura 28: *Speedup di* `3dtomo_fan_fp+tv` *in funzione del numero di processori.*

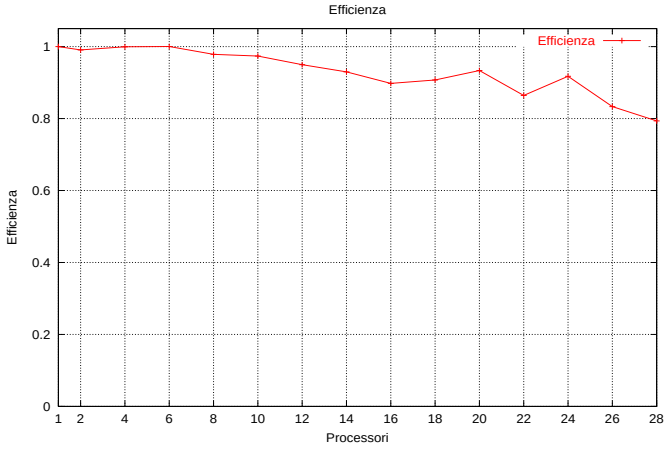


Figura 29: *Efficienza di* `3dtomo_fan_fp+tv` *in funzione del numero di processori.*

2. lo sviluppo di una collezione di moduli software per la ricostruzione di dati 3D SPECT, in ambiente di calcolo ad alte prestazioni, con tempi di risposta compatibili con quelli della diagnostica medica: i risultati ottenuti mostrano che i tempi di ricostruzione di dati 3D SPECT, riportati in figura 27, si riducono ad alcune decine di secondi (circa 1 minuto ad iterazione).

10 Sviluppi Futuri

Lo sviluppo di MEDITOMO si colloca nell'ambito di un'attività di ricerca in collaborazione col Dipartimento di Fisiopatologia Clinica dell'Università di Firenze, il Dipartimento di Fisica dell'Università di Genova e l'Istituto per il Calcolo e Reti ad Alte Prestazioni del Consiglio Nazionale delle Ricerche Sede di Napoli. Tali moduli software sono oggetto di un'effettiva sperimentazione clinica presso l'ospedale Careggi di Firenze.
MEDITOMO si colloca in un progetto più ampio dal nome MEDIGRID [11]. MEDIGRID è un'applicazione che consentirà ai dottori di medicina nucleare, di utilizzare in maniera semplice e trasparente, ambienti di calcolo ad alte prestazioni e sistemi di memorizzazione (*storage*), per l'elaborazione, la visualizzazione e l'analisi di dati PET e SPECT.

Ringraziamenti

Si ringrazia il prof. Mario Bertero per i consigli forniti, nati dalla sua profonda esperienza sull'argomento.

Si ringraziano Andreas Robert Formiconi e Piero Calvini per averci fornito i software prototipali di ricostruzione e per la calorosa collaborazione di questi anni.

Si ringrazia l'ospedale Careggi di Firenze per averci fornito sia i dati sperimentali, utilizzati nel corso delle esperienze numeriche, sia l'esperienza dei dottori in medicina nucleare.

Gli Autori

Riferimenti bibliografici

[1] L. Alvarez, F. Guichard, P.L. Lions, J.M. Morel ,*Axioms and fundamental equations of image processing*, Arch. Rational Mechanics, 123, (1993).

[2] L. Antonelli, *Un software parallelo basato sul metodo dei minimi quadrati con vincoli*, Degree Thesis, (1999).

[3] L. Antonelli, L. Carracciuolo, M. Ceccarelli, L. D'Amore, A. Murli, *High Performance Edge Preserving Restoration in 3D SPECT Imaging*, Parallel Computing Special Issue "Parallel and Distribuited Image and Video Processing" (to appear).

[4] L. Antonelli, L. Carracciuolo, M. Ceccarelli, L. D'Amore, A. Murli, *Total Variation Regularization for Edge Preserving 3D SPECT Imaging in High Performance Computing Environments* Lecture Notes in Computer Science, (2002).

[5] L. Antonelli, L. Carracciuolo, L. D'Amore, A. Murli, Il funzionale di Totale Variazione nella ricostruzione di dati 3D SPECT mediante l'algoritmo di EMOS$_n$, ICAR-CNR technical report, TR-02-01, (2002)

[6] L. Antonelli, *Sulla risoluzione numerica di un problema inverso mal posto in ambiente di calcolo ad alte prestazioni*, PhD Thesis, (2003).

[7] L. Antonelli, L. Carracciuolo, L. D'Amore, A. Murli, *Stima teorica e validazione sperimentale del nucleo computazionale per la ricostruzione di dati SPECT 3D basato sull'algoritmo del Gradiente Coniugato*, ICAR-CNR technical report, TR-04-15, (2004).

[8] L. Antonelli, *Stima teorica e sua validazione sperimentale delle prestazioni degli algoritmi CG ed EM alla base della ricostruzione di dati SPECT*, ICAR-CNR technical report, TR-05-4, (2005).

[9] D. Baldini, P. Calvini, A.R. Formiconi, *Image reconstruction with conjugate gradient algorithm and compensation of the variable system response for an annular SPECT system*, Physic Medica, vol 14, (1998).

[10] M. Bertero, P. Boccacci, *Introduction to Inverse Problems in Imaging*, Institute of Physics Publishing, Bristol (1998).

[11] M. Bertero, P. Bonetto, L. Carracciuolo, L. D'Amore, A.R. Formiconi, M.R. Guarracino, G. Laccetti, A. Murli, G. Oliva, *MedIGrid: a medical image application for computational Grid*, 17th International Parallel and Distributed Processing Symposium, IPDPS-2003 (2003).

[12] P. Boccacci, P. Bonetto, P. Calvini, A. Formiconi, *A simple model for the efficient correction of collimator blur in 3D SPECT imaging*, Inverse Problems 15 (1999).

[13] P. Blomgren, T. Chan, P. Mulet, C. Wong, *Total Variation image restoration: numerical methods and extensions*, IEEE International Conference on Image Processing (1997).

[14] C. Byrne, *Accelerating the EMML algorithm and related iterative algorithms by rescaled block-iterative methods*, IEEE Transactions on Image Processing 7, (1998).

[15] C. Byrne, *Notes on block-iterative or ordered subset methods for image reconstruction*, University of Massachusettes technical report, MA01854, (2000).

[16] L. Carracciuolo, L. D'Amore, A. Murli, *Sviluppo di software parallelo per la ricostruzione di dati 3D SPECT basato sull'algoritmo $EMOS_n$*, ICAR-CNR technical report, TR-02-02, (2002).

[17] D. Casaburi, *Un software parallelo basato sul metodo di Newton*, Degree Thesis, (2002).

[18] Y. Censor, *Row-action methods for huge and sparse systems and their applications*, SIAm Review 23 (1981).

[19] T. Chan, P. Mulet, *Iterative methods for Total Variation image restoration*, SIAM Journal on Applied Mathematics, (1995).

[20] T.F. Chan, G.H. Golub, P. Mulet, *A non linear Primal-Dual Method for Total Variation-Based Image Restoration*, Lecture Notes in Control and Information Sciencies, (1996)

[21] L. D'Amore, V. De Simone, A. Murli, *The Wiener Filter and Regularization Methods for Image Restoration Problems*, Proceedings of the International Conference on Image Analysis and Processing, IEEE Computer Society Publisher, (1999).

[22] D. Dobson, C. Vogel *Convergence of an iterative method for Total Variation*, SIAM Journal on Numerical Analysis, (1997)

[23] R.E. Ewing, J. Shen, *A multigrid algorithm for the cell-center finite difference scheme*, 6th Copper Mountain Conference Multigrid Methods, Imperial College Press, (1993)

[24] A. Formiconi, *Geometrical response of multihole collimators*, Physics in Medicine and Biology 43 (1998).

[25] E. Giusti, *Minimal Surface and Function of Bounded Variation*, Birkhauser, (1984)

[26] P.J. Green, *Bayesian reconstruction from emission tomography data using a modified EM algorithm*, IEEE Transactions on Medical Imaging 9, (1990).

[27] P.J. Green, *On use of the EM algorithm for penalized likelihood estimation*, Journal of the Royal Statistical Society B 52, (1990).

[28] C.W. Groetsch, *Inverse problems in the mathematical sciences*, Vieweg Verlag, Wiesbaden, (1993).

[29] R.H. Huesman, G.T. Gullberg, W.L. Greenberg, T.F. Budinger, *User's Manual; Donner Algorithms for Reconstruction Tomography*, Lawrence Berkeley Laboratory, University of California (1977).

[30] H. Hudson, R. Larkin, *Accelerated image reconstruction using ordered subsets of projection data*, IEEE Transactions on Medical Imaging 4, (1994).

[31] M. Oman, C. Vogel, *Iterative method for Total Variation denoising*, SIAM Journal on Scientific and Statistical Computing (1996).

[32] M. Oman, C. Vogel, *Fast robust Total Variation-based reconstruction of noisy, blurred images*, IEEE Transactions on Image Processing 7 (1998).

[33] M. Oman, C. Vogel,*Continuation method for Total Variation denoising problem* SIAM Journal on Applied Mathematics (1995).

[34] V. Y. Panin, G.L. Zeng, G.T. Gullberg ,*Total Variation Regulated EM Algorithm*, IEEE Transactions on Nuclear Science 6, (1999).

[35] L. Shepp, Y. Vardi, *Maximum likelihood reconstruction for emission tomography*, IEEE Transactions on Medical Imaging 4, (1982).