Università degli Studi di Cagliari

Facoltà Di Ingegneria e Architettura

Corso di Laurea in Ingegneria Elettrica ed Elettronica

Un'interfaccia grafica per l'inversione di dati EMI in geofisica applicata

Tesi di Laurea Triennale

Relatore:
Prof.
Giuseppe Rodriguez

Presentata da:
Gabriele Lovicu

Correlatrice:
Dott.ssa
Patricia Díaz de Alba

Anno Accademico 2017/2018

Indice

Introduzione

Le tecniche di induzione elettromagnetica (EMI) sono tecniche non distruttive usate in geofisica nel problema dell'identificazione delle proprietà elettriche e magnetiche del terreno e nell'individuazione della presenza di particolari materiali nel terreno. Lo strumento usato in queste applicazioni è il Ground Conductivity Meter (GCM).

Il GCM è un sistema di misura consistente di due spire, una che trasmette e una che riceve. Il campo magnetico alla spira ricevente è la somma del campo magnetico trasmesso e del campo generato dalle correnti nel suolo. La misura consiste nel rapporto tra i due campi magnetici.

Figura 1: Foto di due GCM *GF Instruments*, a sinistra il modello *CMD-1*, a destra il modello *CMD-Explorer*.
Le immagini sono state tratte dal sito del produttore.

In geofisica i metodi di induzione elettromagnetica sono utilizzati in diverse applicazioni. Trovano impiego nella caratterizzazione idrologica e idrogeologica del terreno [11]; questi metodi sono utilizzati in agricoltura con lo scopo di verificare alcune proprietà del terreno sottostante, come il livello di concentrazione salina [10, 11]; in archeologia questi metodi sono usati per migliorare la precisione negli scavi [13]. Le tecniche EMI possono essere anche utilizzate per la ricerca di ordigni inesplosi nel sottosuolo [9].

Il lavoro svolto in questa tesi è consistito nel costruire una interfaccia grafica usando il toolset di MATLAB App Designer per un algoritmo già implementato in MATLAB per l'inversione dei dati EMI. Il modello matematico e l'algoritmo numerico sono stati illustrati negli articoli [3, 4, 5].

La tesi si divide in cinque capitoli, i primi tre descrivono il modello matematico, i metodi di risoluzione, e il problema inverso. Gli ultimi due capitoli illustrano brevemente il funzionamento di App Designer, il framework e ambiente di sviluppo di MATLAB per interfacce grafiche, e l'interfaccia grafica del programma.

Capitolo 1

Modello matematico

Le spire possono essere orientate sia in modo da avere l'asse del dipolo ortogonale al terreno (orientazione verticale), sia in modo da avere l'asse del dipolo parallelo al terreno (orientazione orizzontale), con la condizione che entrambe le spire siano orientate allo stesso modo.

La spira trasmittente induce un campo magnetico sinusoidale che viene usualmente definito *campo primario* $\mathbf{H_p}$, mentre l'altra spira si occupa di ricevere il campo magnetico definito *campo secondario* $\mathbf{H_s}$. Il campo $\mathbf{H_p}$ induce una densità di corrente sotto la superficie del suolo; queste correnti di conseguenza danno luogo ad un campo magnetico che si somma al campo primario, questa somma costituisce il campo secondario $\mathbf{H_s}$. Le informazioni utili ai fini della misura vengono estratte dal rapporto $\mathbf{H_s}/\mathbf{H_p}$ tra il campo secondario e il campo primario. Gli strumenti di misura in generale forniscono sia la componente reale (*in-phase*), sia la componente immaginaria (*quadrature*) del rapporto tra i campi, la componente reale dà principalmente informazioni sulla permeabilità magnetica del sottosuolo, mentre la parte immaginaria dà principalmente informazioni sulla conducibilità elettrica.

Chiamando H_p e H_s le componenti lungo l'asse del dipolo di $\mathbf{H_p}$ e $\mathbf{H_s}$, rispettivamente, misurate alla spira ricevente, vale [3]:

$$\frac{H_s}{H_p} = M_\nu(\boldsymbol{\sigma}, \boldsymbol{\mu}, h_i, \omega_j), \qquad \nu = 0, 1,\ i = 1, ..., m_h,\ j = 1, ..., m_\omega,$$

dove $\boldsymbol{\sigma}$ è il vettore delle conducibilità elettriche di ogni strato del terreno, $\boldsymbol{\mu}$ è il vettore delle permeabilità magnetiche, h_i sono le varie altezze dello strumento di misura, ω_j sono le varie pulsazioni angolari.

McNeill [12] ha presentato un metodo per stimare le conducibilità e la permeabilità utilizzando dei modelli multistrato del terreno, applicabili sotto l'ipotesi di un basso *numero di induzione*, definito come:

$$B = \frac{r}{\delta} = r\sqrt{\frac{\mu_0 \omega \sigma}{2}},$$

dove

$$\delta = \sqrt{\frac{2}{\omega \mu \sigma}}$$

è la *skin depth*, che in un buon conduttore rappresenta la profondità alla quale il campo $\mathbf{H_p}$ si è attenuato di un fattore e^{-1}, $i = \sqrt{-1}$ è l'unità immaginaria, ω è la

pulsazione angolare, μ è la permeabilità magnetica, μ_0 è la permeabilità magnetica del vuoto, σ è la conducibilità elettrica e r è la distanza tra le spire.

Il modello utilizzato da McNeill è un modello lineare, ipotizza cioè una dipendenza lineare tra la conducibilità elettrica nel suolo e la risposta data dal GCM.

Il modello non lineare elimina questa ipotesi e considera la risposta del GCM dipendente non linearmente dalla conducibilità. Modelli non lineari sono stati proposti in [3, 4, 5, 8, 16].

Il modello non lineare è stato derivato dalle equazioni di Maxwell, sotto l'ipotesi di simmetria cilindrica. Viene fatta l'ipotesi che il terreno sia stratificato in n strati, ciascuno dei quali avente spessore d_k, $k = 1, ..., n$, assumendo che sul k-esimo strato la conducibilità elettrica e la permeabilità magnetica assumano valori costanti, rispettivamente σ_k, e μ_k.

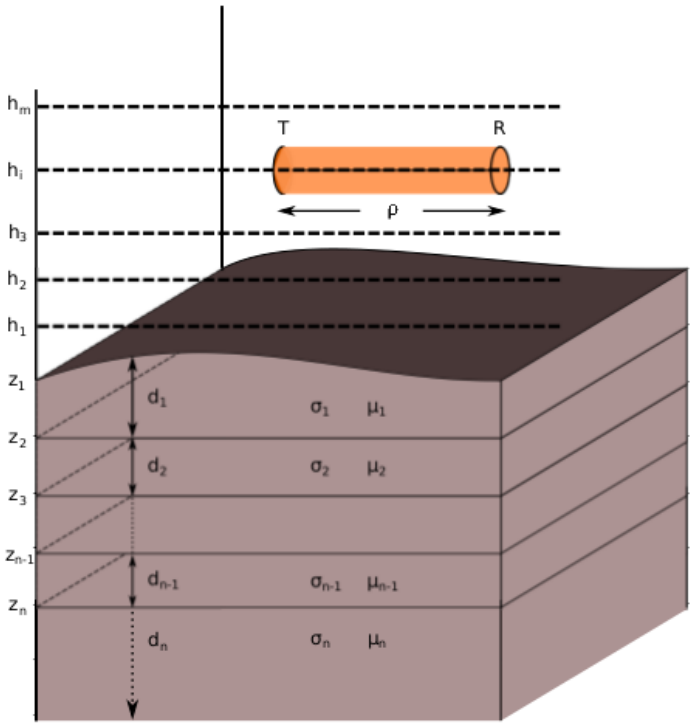

Figura 1.1: Discretizzazione del terreno. L'immagine è stata tratta da [3]

L'incidenza su un mezzo stratificato può venire descritta, nell'ipotesi di interfacce parallele e di campo incidente piano, tramite il modello delle linee di trasmissione [2, 16].

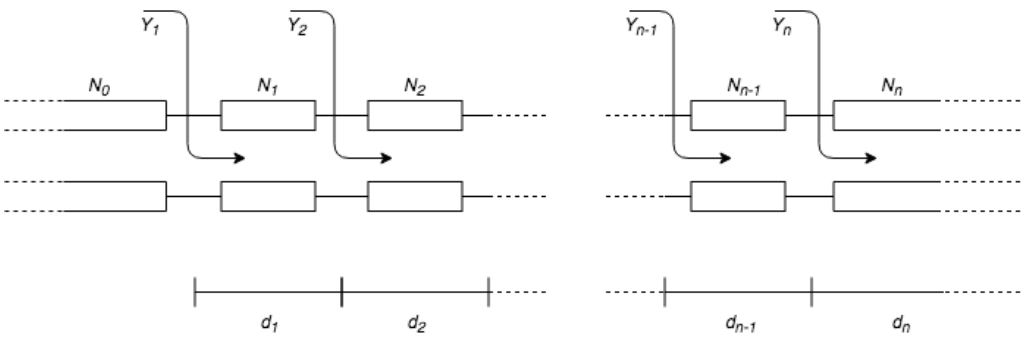

Figura 1.2: Modello equivalente a linee di trasmissione del sistema

L'ammettenza di superficie Y_k al k-esimo strato è data da [8, 16]:

$$Y_k(\lambda) = N_k(\lambda)\frac{Y_{k+1}(\lambda) + N_k(\lambda)\tanh(d_k u_k(\lambda))}{N_k(\lambda) + Y_{k+1}(\lambda)\tanh(d_k u_k(\lambda))}, \qquad k = 1, ..., n-1,$$

dove λ è una costante di integrazione che ha dimensioni dell'inverso della skin depth. Per $k = n$, l'ammettenza di superficie viene inizializzata come $Y_n(\lambda) = N_n(\lambda)$, essendo l'ultimo strato semi-infinito.

La funzione $u_k(\lambda)$ è una costante di propagazione definita da:

$$u_k(\lambda) = \sqrt{\lambda^2 + i\sigma_k\mu_k\omega},$$

dove $i = \sqrt{-1}$.

Il coefficiente di riflessione tra l'aria e il primo strato è

$$R_{\omega,0}(\lambda) = \frac{N_0(\lambda) - Y_1(\lambda)}{N_0(\lambda) + Y_1(\lambda)}.$$

L'ammettenza caratteristica al k-esimo strato è data da

$$N_k(\lambda) = \frac{u_k(\lambda)}{i\mu_k\omega}, \qquad k = 1, ..., n,$$

dove

$$N_0(\lambda) = \frac{\lambda}{i\mu_0\omega}.$$

Nel caso in cui le spire dello strumento siano allineate verticalmente, [3, 8], il rapporto tra il campo secondario e il campo primario è dato da:

$$M_0(\boldsymbol{\sigma}, \boldsymbol{\mu}; h, \omega) = -\rho^3 \int_0^\infty \lambda^2\, e^{-2h\lambda}\, R_{\omega,0}(\lambda)\, J_0(\rho\lambda)\, d\lambda,$$

dove $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, ..., \sigma_n]^T$ è il vettore delle conducibilità e $\boldsymbol{\mu} = [\mu_1, \mu_2, ..., \mu_n]^T$ è il vettore delle permeabilità, e dove $J_\nu(\lambda)$ indica le funzioni di Bessel del primo tipo di ordine ν:

$$J_\nu(\lambda) = \frac{\lambda}{2} \sum_{k=0}^{+\infty} \frac{(-1)^k}{2^{2k}\, k!\, (k+\nu)!}\, \lambda^{2k}.$$

Nel caso di spire allineate orizzontalmente, il rapporto tra il campo secondario e il campo primario è dato da:

$$M_1(\boldsymbol{\sigma}, \boldsymbol{\mu}; h, \omega) = -\rho^2 \int_0^\infty \lambda\, e^{-2h\lambda}\, R_{\omega,0}(\lambda)\, J_1(\rho\lambda)\, d\lambda.$$

Le espressioni di M_0 e M_1 si possono anche esprimere come trasformate di Hankel [8]:

$$\mathcal{H}_\nu\{f\}(\rho) = \int_0^\infty f(\lambda)\, J_\nu(\rho\lambda)\, \lambda\, d\lambda, \qquad \nu = 0, 1.$$

Capitolo 2

Metodi matematici

2.1 Problema ai minimi quadrati non lineare

Un problema non lineare ai minimi quadrati [1] è un problema in cui, data una certa funzione $f : \mathbb{R}^n \to \mathbb{R}$, $f : \boldsymbol{x} \longmapsto f(\boldsymbol{x})$, così definita

$$f(\boldsymbol{x}) = \frac{1}{2} \, \|\boldsymbol{r}(\boldsymbol{x})\|_2^2 = \frac{1}{2} \sum_{i=1}^{m} r_i^2(\boldsymbol{x}), \qquad \boldsymbol{r}(\boldsymbol{x}) \in \mathbb{R}^m, \qquad m \geq n,$$

dove $\boldsymbol{r}(\boldsymbol{x})$ è il vettore dei residui, e le sue componenti $r_i(\boldsymbol{x})$ sono funzioni non lineari di $\boldsymbol{x}$; si vuole trovare il valore che minimizzi:

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} f(\boldsymbol{x}).$$

Sia

$$\boldsymbol{r}(\boldsymbol{x}) = [r_1(\boldsymbol{x}), r_2(\boldsymbol{x}), \ldots, r_m(\boldsymbol{x})]^T$$

con l'ipotesi che le funzioni $r_i(\boldsymbol{x})$ siano differenziabili due volte.

La *matrice Jacobiana* del vettore dei residui $\boldsymbol{r}(\boldsymbol{x})$ è definita come

$$J_r(\boldsymbol{x}) \in \mathbb{R}^{m \times n}, \qquad J_r(\boldsymbol{x})_{ij} = \frac{\partial r_i(\boldsymbol{x})}{\partial x_j}, \qquad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

La *matrice Hessiana* per ciascun elemento $r_i(\boldsymbol{x})$ del vettore dei residui è definita come

$$\mathcal{G}_i(\boldsymbol{x}) \in \mathbb{R}^{n \times n}, \qquad \mathcal{G}_i(\boldsymbol{x})_{jk} = \frac{\partial^2 r_i(\boldsymbol{x})}{\partial x_j \partial x_k}, \quad i = 1, \ldots, m, \quad j, k = 1, \ldots, n,$$

con inoltre l'ipotesi che la matrice sia simmetrica: $\mathcal{G}_i(\boldsymbol{x}) = \mathcal{G}_i(\boldsymbol{x})^T$.

Il gradiente della funzione $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{r}(\boldsymbol{x})^T\boldsymbol{r}(\boldsymbol{x})$ è dato da

$$\nabla f(\boldsymbol{x}) = J_r(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x}),$$

mentre la matrice Hessiana di $f(\boldsymbol{x})$ è data da

$$\mathcal{G}_f(\boldsymbol{x}) = J_r(\boldsymbol{x})^T J_r(\boldsymbol{x}) + \sum_{i=1}^{m} r_i(\boldsymbol{x}) \mathcal{G}_i(\boldsymbol{x}).$$

Una condizione necessaria perché un punto $\boldsymbol{x}_c$ sia un minimo locale di $f(\boldsymbol{x})$ è che

$$\nabla f(\boldsymbol{x}_c) = J_r(\boldsymbol{x}_c)^T \boldsymbol{r}(\boldsymbol{x}_c) = \boldsymbol{0}.$$

Qualsiasi punto soddisfi questa condizione viene detto *punto critico*.

Da un punto di vista classico, Wedin [1] ha ideato un approccio geometrico al problema, che consiste nel trovare il punto nella superficie n-dimensionale $\boldsymbol{z} = \boldsymbol{r}(\boldsymbol{x})$ più vicino all'origine.

Se, per ipotesi, $\operatorname{rank}(J_r(\boldsymbol{x})) = \min(m, n) = n$, si ha che

$$J_r(\boldsymbol{x})^\dagger J_r(\boldsymbol{x}) = \mathcal{I}_n,$$

dove $J_r(\boldsymbol{x})^\dagger$ è l'*inversa di Moore-Penrose* di $J_r(\boldsymbol{x})$, e $\mathcal{I}_n$ è la matrice identità di dimensione n.

Posti $\boldsymbol{w}(\boldsymbol{x}) = -\frac{\boldsymbol{r}(\boldsymbol{x})}{\|\boldsymbol{r}(\boldsymbol{x})\|_2}$ e $\mathcal{G}_w(\boldsymbol{x}) = \sum_{i=1}^m w_i(\boldsymbol{x})\mathcal{G}_i(\boldsymbol{x})$, si può riscrivere la matrice Hessiana come:

$$\mathcal{G}_f(\boldsymbol{x}) = J_r(\boldsymbol{x})^T J_r(\boldsymbol{x}) - \|\boldsymbol{r}(\boldsymbol{x})\|_2 \mathcal{G}_w(\boldsymbol{x}),$$

moltiplicando a sinistra di $\mathcal{G}_w$ per $\left[J_r(\boldsymbol{x})^\dagger J_r(\boldsymbol{x})\right]^T$ e a destra per $\left[J_r(\boldsymbol{x})^\dagger J_r(\boldsymbol{x})\right]$:

$$\mathcal{G}_f(\boldsymbol{x}) = J_r(\boldsymbol{x})^T J_r(\boldsymbol{x}) - \|\boldsymbol{r}(\boldsymbol{x})\|_2 \left[J_r(\boldsymbol{x})^\dagger J_r(\boldsymbol{x})\right]^T \mathcal{G}_w(\boldsymbol{x}) \left[J_r(\boldsymbol{x})^\dagger J_r(\boldsymbol{x})\right],$$

da cui

$$\mathcal{G}_f(\boldsymbol{x}) = J_r(\boldsymbol{x})^T \left[\mathcal{I} - \|\boldsymbol{r}(\boldsymbol{x})\|_2 \left(J_r(\boldsymbol{x})^\dagger\right)^T \mathcal{G}_w(\boldsymbol{x}) J_r(\boldsymbol{x})^\dagger\right] J_r(\boldsymbol{x}).$$

La matrice

$$K = \left(J_r(\boldsymbol{x})^\dagger\right)^T \mathcal{G}_w(\boldsymbol{x}) J_r(\boldsymbol{x})^\dagger$$

detta *matrice di curvatura*, è una matrice simile a $\mathcal{G}_f(\boldsymbol{x})$, e di conseguenza le matrici hanno gli stessi autovalori.

Siano gli autovalori di K

$$\kappa_1 \geq \kappa_2 \geq \cdots \geq \kappa_n,$$

e siano i reciproci $\rho_i = 1/\kappa_i$ i raggi di curvatura principali della superficie rispetto alla normale $\boldsymbol{w}(\boldsymbol{x})$ (se $J_r(\boldsymbol{x})$ è a rango pieno, come supposto, $\kappa_i \neq 0$).

La matrice Hessiana è data da

$$\mathcal{G}_f(\boldsymbol{x}) = J_r^T(\mathcal{I} - \|\boldsymbol{r}(\boldsymbol{x})\|_2 K) J_r;$$

- $\boldsymbol{x}_c$ è un punto di minimo locale se e solo se $\mathcal{I} - \|\boldsymbol{r}(\boldsymbol{x})\|_2 K$ è definita positiva in $\boldsymbol{x}_c$, ossia se e solo se
$$1 - \|\boldsymbol{r}(\boldsymbol{x})\|_2 \kappa_1 \geq 0;$$
- $\boldsymbol{x}_c$ è un punto di sella se e solo se $\mathcal{I} - \|\boldsymbol{r}(\boldsymbol{x})\| K$ è indefinita in $\boldsymbol{x}_c$, ossia se e solo se
$$1 - \|\boldsymbol{r}(\boldsymbol{x})\|_2 \kappa_1 \leq 0;$$
- $\boldsymbol{x}_c$ è un punto di massimo locale se e solo se $\mathcal{I} - \|\boldsymbol{r}(\boldsymbol{x})\| K$ è definita negativa in $\boldsymbol{x}_c$, ossia se e solo se
$$1 - \|\boldsymbol{r}(\boldsymbol{x})\|_2 \kappa_n < 0.$$

2.2 Metodi per equazioni non lineari

2.2.1 Il metodo di Newton

Numericamente, per trovare il punto in cui il gradiente si annulla, è possibile utilizzare il *metodo di Newton*(o *metodo delle tangenti*) [15, 1]. Sia $\boldsymbol{\phi} : \mathbb{R}^n \longrightarrow \mathbb{R}^n$, $\boldsymbol{\phi} : \boldsymbol{x} \longmapsto \boldsymbol{\phi}(\boldsymbol{x})$.

In $\mathbb{R}^n$ il metodo di Newton consiste, dato un passo k, nella iterazione

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - J_\phi^{-1}(\boldsymbol{x}_k)\boldsymbol{\phi}(\boldsymbol{x}_k),$$

o, equivalentemente, definendo il *passo di iterazione* $\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$

$$J_\phi(\boldsymbol{x}_k)\boldsymbol{s}_k = -\boldsymbol{\phi}(\boldsymbol{x}_k).$$

Per risolvere il problema non lineare ai minimi quadrati, è necessario trovare un punto di minimo locale

$$\nabla f(\boldsymbol{x}) = \boldsymbol{0},$$

di conseguenza si può usare il metodo di Newton per trovare $\boldsymbol{x}_c$ che annulli la funzione:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \nabla f(\boldsymbol{x});$$

$$J_\phi(\boldsymbol{x}) = J_{\nabla f}(\boldsymbol{x}).$$

La matrice Jacobiana del gradiente è la matrice Hessiana:

$$[J_{\nabla f}(\boldsymbol{x})]_{ij} = \frac{\partial}{\partial x_j}[\nabla f(\boldsymbol{x})]_i = \frac{\partial}{\partial x_j}\frac{\partial f(\boldsymbol{x})}{\partial x_i} = \frac{\partial^2 f(\boldsymbol{x})}{\partial x_j \partial x_i} = [\mathcal{G}_f(\boldsymbol{x})]_{ji} = [\mathcal{G}_f(\boldsymbol{x})]_{ij}.$$

Il metodo di Newton diventa quindi

$$\mathcal{G}_f(\boldsymbol{x}_k)\boldsymbol{s}_k = -\nabla f(\boldsymbol{x}_k).$$

Il metodo di Newton ha un grosso inconveniente dal punto di vista computazionale, dal momento che il calcolo della matrice Hessiana ha in genere un costo computazionale oneroso.

2.2.2 Il metodo di Gauss-Newton

Il *metodo di Gauss-Newton* [1] non necessita del calcolo della matrice Hessiana, ma risolve il problema non lineare ai minimi quadrati mediante una sequenza di approssimazioni lineari della funzione $\boldsymbol{r}(\boldsymbol{x})$. Il metodo consiste nel trovare il passo $\boldsymbol{s} \in \mathbb{R}^n$ che risolve il seguente problema lineare ai minimi quadrati:

$$\min_{\boldsymbol{s}\in\mathbb{R}^n} \|\boldsymbol{r}(\boldsymbol{x}_k) + J_r(\boldsymbol{x}_k)\boldsymbol{s}\|_2,$$

ad ogni iterazione si calcola

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k,$$

il problema lineare ai minimi quadrati può essere risolto ad ogni interazione mediante la *fattorizzazione QR*.

Problema ai minimi quadrati lineare. Un problema lineare ai minimi quadrati [1, 15] è un problema in cui, dato $\boldsymbol{x} \in \mathbb{R}^n$, è necessario risolvere:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \|\boldsymbol{r}(\boldsymbol{x})\|_2 ,$$

dove $\boldsymbol{r}(\boldsymbol{x}) \in \mathbb{R}^m$ è funzione lineare di $\boldsymbol{x}$.

Per un sistema *sovradeterminato* $A\boldsymbol{x} = \boldsymbol{b}$, dove $A \in \mathbb{R}^{m\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$ e A è a rango pieno, è necessario risolvere

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \|A\boldsymbol{x} - \boldsymbol{b}\|_2 .$$

La soluzione ai minimi quadrati deve soddisfare la condizione:

$$A^T A\boldsymbol{x} = A^T\boldsymbol{b},$$

queste equazioni sono dette *equazioni normali del primo tipo*, e la loro soluzione formale è

$$\boldsymbol{x} = A^\dagger\boldsymbol{b}, \qquad A^\dagger = (A^T A)^{-1}A^T.$$

Per un sistema *sottodeterminato* $A^T\boldsymbol{y} = \boldsymbol{c}$, dove $\boldsymbol{y} \in \mathbb{R}^m$, $\boldsymbol{c} \in \mathbb{R}^n$ e A^T è a rango pieno, è necessario risolvere

$$\min \|\boldsymbol{y}\|_2 , \qquad A^T\boldsymbol{y} = \boldsymbol{c}.$$

La soluzione deve soddisfare la condizione:

$$A^T A\boldsymbol{z} = \boldsymbol{c}, \qquad \boldsymbol{y} = A\boldsymbol{z},$$

queste equazioni sono dette *equazioni normali del secondo tipo*, e la loro soluzione formale è

$$\boldsymbol{y} = A^\dagger\boldsymbol{c}, \qquad A^\dagger = A(A^T A)^{-1}.$$

2.2.3 Il metodo di Gauss-Newton smorzato

Un metodo più efficiente del metodo di Gauss-Newton è il *metodo di Gauss-Newton smorzato* [1].

Ad ogni iterazione si calcola

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k\boldsymbol{s}_k,$$

dove α_k è un parametro reale detto *lunghezza di passo*, e il passo $\boldsymbol{s}_k$ è la soluzione del problema ai minimi quadrati lineare.

Il vettore $\boldsymbol{s}_k$ rappresenta in questo caso una direzione, quando $J_r(\boldsymbol{x_k})$ non è a rango pieno, la direzione $\boldsymbol{s}_k$ viene scelta come la soluzione di minima norma del problema lineare ai minimi quadrati

$$\min_{\boldsymbol{s}\in\mathbb{R}^n} \|\boldsymbol{r}(\boldsymbol{x}_k) + J_r(\boldsymbol{x}_k)\boldsymbol{s}\|_2$$

la soluzione è

$$\boldsymbol{s}_k = -J_r^\dagger(\boldsymbol{x}_k)\boldsymbol{r}(\boldsymbol{x}_k).$$

Il vettore $\boldsymbol{s}_k$ è invariante rispetto a qualsiasi trasformazione lineare della variabile dipendente $\boldsymbol{x}_k$. Inoltre se $\boldsymbol{x}_k$ non è un punto critico, allora per una lunghezza di passo sufficientemente piccola $\alpha > 0$, si ha:

$$\|\boldsymbol{r}(\boldsymbol{x}_k + \alpha \boldsymbol{s}_k)\|_2 < \|\boldsymbol{r}(\boldsymbol{x}_k)\|_2 .$$

Per la scelta della lunghezza di passo α_k viene usato il *criterio di Armijo-Goldstein* [1, 4], che consiste nello scegliere α_k come il numero più grande della successione $\left\{\frac{1}{2^n}\right\}_{n\in\mathbb{N}}$ che soddisfi la disuguaglianza:

$$\|\boldsymbol{r}(\boldsymbol{x}_k)\|_2^2 - \|\boldsymbol{r}(\boldsymbol{x}_k + \alpha_k \boldsymbol{s}_k)\|_2^2 \geq \frac{1}{2}\alpha_k \|J_r(\boldsymbol{x}_k)\boldsymbol{s}_k\|_2^2 .$$

2.2.4 Metodo di Broyden

Il *metodo di Broyden* [15] è un metodo *quasi-Newton* che consiste in una generalizzazione al caso in più variabili del metodo delle secanti.

Il metodo di Broyden consiste nell'approssimare lo Jacobiano $J_\phi(\boldsymbol{x}_k)$ con una matrice J_k tale che venga rispettata la seguente condizione, detta *equazione della secante*:

$$J_k \boldsymbol{s}_k = \phi(\boldsymbol{x}_k) - \phi(\boldsymbol{x}_{k-1}).$$

Definendo $\phi_k(\boldsymbol{x})$ come:

$$\phi_k(\boldsymbol{x}) = \phi(\boldsymbol{x}_k) + J_k(\boldsymbol{x} - \boldsymbol{x}_k),$$

si ottiene che:

$$\phi_k(\boldsymbol{x}) - \phi_{k-1}(\boldsymbol{x}) = (J_k - J_{k-1})(\boldsymbol{x} - \boldsymbol{x}_{k-1}).$$

Sostituendo

$$\boldsymbol{x} - \boldsymbol{x}_{k-1} = \alpha \boldsymbol{s}_k + \boldsymbol{t},$$

con $\alpha \in \mathbb{R}$ e $\boldsymbol{s}_k^T \boldsymbol{t} = 0$, si ha

$$\phi_k(\boldsymbol{x}) - \phi_{k-1}(\boldsymbol{x}) = \alpha(J_k - J_{k-1})\boldsymbol{s}_k + (J_k - J_{k-1})\boldsymbol{t}.$$

Per minimizzare la variazione è necessario imporre che $(J_k - J_{k-1})\boldsymbol{t} = \boldsymbol{0}$, cioè $(J_k - J_{k-1})$ si può scrivere come una matrice di rango 1 $\boldsymbol{u}\boldsymbol{s}_k^T$, con $\boldsymbol{u} \in \mathbb{R}^n$.

L'equazione della secante deve essere verificata, per cui:

$$(J_k - J_{k-1})\boldsymbol{s}_k = \boldsymbol{u}\boldsymbol{s}_k^T \boldsymbol{s}_k = \phi(\boldsymbol{x}_k) - \phi(\boldsymbol{x}_{k-1}) - J_{k-1}\boldsymbol{s}_k,$$

per cui

$$\boldsymbol{u} = \frac{[\phi(\boldsymbol{x}_k) - \phi(\boldsymbol{x}_{k-1})] - J_{k-1}\boldsymbol{s}_k}{\|\boldsymbol{s}_k\|_2^2},$$

e quindi:

$$J_k = J_{k-1} + \frac{[\phi(\boldsymbol{x}_k) - \phi(\boldsymbol{x}_{k-1})] - J_{k-1}\boldsymbol{s}_k}{\|\boldsymbol{s}_k\|_2^2}\boldsymbol{s}_k^T.$$

Questo metodo viene usato per approssimare numericamente la matrice Jacobiana.

2.3 Regolarizzazione di Tikhonov

Nei problemi malcondizionati è utile aggiungere condizioni per dare al problema la forma aspettata. La regolarizzazione di Tikhonov [1, 7] ha lo scopo di integrare *a priori* delle assunzioni inerenti alla dimensione e alla continuità della soluzione. Si considera una matrice $L \in \mathbb{R}^{t\times n}$ $(t \leq n)$ detta *matrice di regolarizzazione* e un parametro λ che porta al problema di minimizzazione:

$$\boldsymbol{x}_\lambda = \arg\min_{\boldsymbol{x}\in\mathbb{R}^n} \left\{ \|\boldsymbol{r}(\boldsymbol{x})\|_2^2 + \lambda^2 \|L\boldsymbol{x}\|_2^2 \right\}.$$

Questo però significa che per poter scegliere λ vanno calcolate diverse $\boldsymbol{x}_\lambda$. È possibile fare questo con Gauss-Newton, ma è computazionalmente oneroso.

È possibile abbassare il costo computazionale approssimando la Jacobiana con un'approssimazione a basso rango.

Il problema di regolarizzazione può essere detto *standard* se L è una matrice identità, *non standard* altrimenti.

2.4 La decomposizione a valori singolari (SVD)

Sia $A \in \mathbb{C}^{m\times n}$ una matrice di rango non pieno $\operatorname{rank}(A) = r$; allora esistono due matrici unitarie $U \in \mathbb{C}^{m\times m}$ e $V \in \mathbb{C}^{n\times n}$, tali che:

$$A = U\Sigma V^*, \qquad \Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix}$$

dove $\Sigma \in \mathbb{R}^{m\times n}$ e Σ_1 è una matrice quadrata diagonale che contiene gli r valori singolari, $\Sigma_1 = \operatorname{diag}(\sigma_1, \sigma_2, \cdots, \sigma_r)$, dove $\sigma_i \in \mathbb{R}^+, i = 1, \cdots, r$ sono i valori singolari della matrice A ordinati in ordine decrescente:

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0.$$

Tale decomposizione è detta *decomposizione a valori singolari* o (SVD, *Singular Value Decomposition*) [1, 7]. Le matrici U e V possono anche essere espresse tramite i loro vettori colonna:

$$U = [\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_m], \qquad V = [\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_n],$$

gli elementi $\boldsymbol{u}_i$ sono detti *vettori singolari sinistri* e gli elementi $\boldsymbol{v}_i$ sono detti *vettori singolari destri*. È possibile definire U_1 e V_1 come:

$$U_1 = [\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_r], \qquad V_1 = [\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_r].$$

La SVD di A può altresì venire scritta come:

$$A = U_1\Sigma_1 V_1^* = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^*.$$

La matrice inversa di Moore-Penrose può essere scritta come:

$$A^\dagger = \sum_{i=1}^{r} \sigma_i^{-1} \boldsymbol{v}_i \boldsymbol{u}_i^*. \tag{2.1}$$

I valori singolari σ_i sono decrescenti, certe volte è utile ignorare i valori singolari al di sotto di una soglia fissata. La sommatoria della SVD può essere scomposta come:

$$A = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^* = \sum_{i=1}^{p} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^* + \sum_{i=p+1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^*.$$

La SVD può essere approssimata non tenendo conto della seconda sommatoria, questa decomposizione viene detta TSVD (SVD troncata):

$$A \simeq \sum_{i=1}^{p} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^*.$$

Nel caso particolare di matrici reali, $V^* = V^T$,

$$A = U_1 \Sigma_1 V_1^T = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T.$$

I valori singolari hanno inoltre delle importanti proprietà, tra cui il fatto che sono collegati al condizionamento della matrice; data una decomposizione $A = U\Sigma V^T$, il numero di condizionamento di $A, \mathrm{cond}(A)$ può essere espresso come il rapporto tra il massimo e il minimo valore singolare della matrice:

$$\mathrm{cond}(A) = \frac{\max\{\sigma_i\}_{i=1,\cdots,r}}{\min\{\sigma_i\}_{i=1,\cdots,r}} = \frac{\sigma_1}{\sigma_r}.$$

2.5 La decomposizione a valori singolari generalizzata

Siano $A \in \mathbb{R}^{m\times n}$ e $L \in \mathbb{R}^{p\times n}$ una coppia di matrici aventi lo stesso numero di colonne; allora, se $\mathcal{N}(A) \cap \mathcal{N}(L) = \{0\}$ esistono due matrici ortogonali $U \in \mathbb{R}^{m\times n}$ e $V \in \mathbb{R}^{p\times p}$ e una matrice non singolare $Z \in \mathbb{R}^{n\times n}$ tali che

$$A = U \begin{bmatrix} \Sigma & 0 \\ 0 & \mathcal{I}_{n-p} \end{bmatrix} Z^{-1}, \qquad L = V \begin{bmatrix} M & 0 \end{bmatrix} Z^{-1}, \qquad m \geq n \geq p.$$

Questa decomposizione è detta *decomposizione a valori singolari generalizzata* o GSVD (*Generalized* SVD) [7]. La matrice Z è una matrice che contiene colonne che sono ortogonali a A^TA:

$$Z^T A^T A Z = \begin{bmatrix} \Sigma^2 & 0 \\ 0 & \mathcal{I}_{n-p} \end{bmatrix}, \qquad Z^T L^T L Z = \begin{bmatrix} M^2 & 0 \\ 0 & 0 \end{bmatrix},$$

Le matrici Σ e M sono matrici diagonali $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \cdots, \sigma_p)$ e $M = \mathrm{diag}(\mu_1, \mu_2, \cdots, \mu_p)$, dove gli elementi diagonali sono ordinati come segue:

$$0 \leq \sigma_1 \leq \sigma_2 \leq \cdots \leq \sigma_p \leq 1, \qquad 1 \geq \mu_1 \geq \mu_2 \geq \cdots \geq \mu_p \geq 0.$$

e sono normalizzati in modo che

$$\sigma_i + \mu_i = 1, \qquad i = 1, \cdots, p.$$

I *valori singolari generalizzati* γ_i della coppia di matrici (A, L) sono definiti

$$\gamma_i = \frac{\sigma_i}{\mu_i}, \qquad i = 1, \cdots, p.$$

Le coppie $(\gamma_i^2, \boldsymbol{z}_i)$ sono le autosoluzioni generalizzate della coppia di matrici (A^TA, L^TL).

2.6 Approssimazione a basso rango

Quando si ha a che fare con matrici con alto numero di condizionamento, può essere utile approssimare la matrice in questione tramite una approssimazione a basso rango [1]. Sia $A \in \mathbb{C}^{m\times n}$ con rango $\operatorname{rank}(A) = r$, la sua SVD è data da:

$$A = U\Sigma V^* = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^*.$$

Sia inoltre $B \in \mathcal{M}_k^{m\times n}$, con $\mathcal{M}_k^{m\times n} \subset \mathbb{C}^{m\times n}$ l'insieme delle matrici complesse $m \times n$ di rango $k < r$. Allora

$$\min \|A - Z\|_2, \qquad Z \in \mathcal{M}_k^{m\times n},$$

è ottenuto per $Z = B$, e

$$B = \sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^*, \qquad \|A - B\|_2 = \sigma_{k+1}.$$

Questo risultato è noto come *teorema di Eckhart-Young*, inizialmente era stato dimostrato solo per la norma di Frobenius, ma poi è stato generalizzato.

2.7 La GSVD troncata

Nelle applicazioni è spesso preferibile l'utilizzo di regolarizzazioni in forma non-standard [7].

Tuttavia da un punto di vista numerico, sono più convenienti le regolarizzazioni in forma standard, in quanto si ha, nella pratica, una matrice in meno.

Sia un problema ai minimi quadrati lineare con residuo $A\boldsymbol{x} - \boldsymbol{b}$, con $A \in \mathbb{R}^{m\times n}$, $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^m$, per la regolarizzazione di Tikhonov:

$$\min\{\|A\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \lambda^2 \|L\boldsymbol{x}\|_2^2\}, \quad L \in \mathbb{R}^{p\times n},$$

questo problema, agendo su A, $\boldsymbol{x}$ e $\boldsymbol{b}$ può essere portato nella forma standard equivalente [7]:

$$\min\{\left\|\bar{A}\bar{\boldsymbol{x}} - \bar{\boldsymbol{b}}\right\|_2^2 + \lambda^2 \|\bar{\boldsymbol{x}}\|_2^2\}.$$

Se A è quadrata e invertibile, le nuove matrici sono $\bar{A} = AL^{-1}$, $\bar{\boldsymbol{b}} = \boldsymbol{b}$, $\boldsymbol{x} = L^{-1}\bar{\boldsymbol{x}}$.

Nel caso di matrici non quadrate, si può definire una matrice $L_A^\dagger$ come:

$$L_A^\dagger = (\mathcal{I}_n - (A(\mathcal{I}_n - L^\dagger L))^\dagger A)L^\dagger.$$

Se $p \geq n$, allora $L_A^\dagger = L^\dagger$, altrimenti le matrici sono generalmente differenti. La componente $\boldsymbol{x}_0$ nel nucleo di L della soluzione regolarizzata è data da:

$$\boldsymbol{x}_0 = (A(\mathcal{I}_n - L^\dagger L))^\dagger \boldsymbol{b}.$$

Scomponendo mediante GSVD la coppia di matrici (A, L): $A = U \begin{bmatrix} \Sigma & 0 \\ 0 & \mathcal{I}_{n-p} \end{bmatrix} X^{-1}$, $L = V \begin{bmatrix} M & 0 \end{bmatrix} X^{-1}$, si possono esprimere $L_A^\dagger$ e $\boldsymbol{x}_0$ come:

$$L_A^\dagger = X \begin{bmatrix} M^{-1} \\ 0 \end{bmatrix} V^T, \qquad \boldsymbol{x}_0 = \sum_{i=p+1}^{n} \boldsymbol{u}_i^T \boldsymbol{b} \boldsymbol{x}_i$$

mentre le matrici per la regolarizzazione in forma standard sono date da:

$$\bar{A} = AL_A^{\dagger}, \quad \bar{\boldsymbol{b}} = \boldsymbol{b} - A\boldsymbol{x}_0, \quad \boldsymbol{x} = L_A^{\dagger}\bar{\boldsymbol{x}} + \boldsymbol{x}_0.$$

È possibile calcolare una matrice Z_k tramite TSVD: $Z_k = \sum_{i=p-k+1}^{p} \boldsymbol{u}_i \gamma_i \boldsymbol{v}_i^T$ che sia la migliore approssimazione di rango k di $AL_A^{\dagger}$, allora la soluzione troncata è data da

$$\boldsymbol{x}_k = L_A^{\dagger} Z_k^{\dagger}(\boldsymbol{b} - A\boldsymbol{x}_0) + \boldsymbol{x}_0,$$

da cui deriva

$$\boldsymbol{x}_k = \sum_{i=p-k+1}^{p} \frac{\boldsymbol{u}_i^T \boldsymbol{b}}{\sigma_i} \boldsymbol{x}_i + \sum_{i=p+1}^{n} \boldsymbol{u}_i^T \boldsymbol{b} \boldsymbol{x}_i.$$

Questo risultato è detto *GSVD troncata* o *TGSVD*.

2.8 Alcuni metodi per la scelta del parametro di regolarizzazione

2.8.1 Il metodo della discrepanza

Il *metodo della discrepanza* [7], generalmente attribuito a Morozov, è un metodo basato sulla norma 2 del *noise* (indicato con $\boldsymbol{e}$). Il metodo consiste nello scegliere il parametro ℓ più piccolo possibile, sotto la condizione:

$$\|\boldsymbol{r}(\boldsymbol{x}_\ell)\|_2 \leq \tau_{discr} \|\boldsymbol{e}\|_2 ,$$

dove τ_{discr} è il parametro della discrepanza. Il metodo della discrepanza è applicabile quindi solo se si conosce il noise, quindi se si ha a che fare con dati sintetici o con sistemi in cui si possono fare delle ipotesi sul noise.

2.8.2 Il metodo della curva L

Un metodo proposto da due matematici, Hansen e O'Leary è quello della curva L [7]. Il criterio della curva L consiste nel costruire una curva in scala logaritmica

$$\left\{\log \left\|\boldsymbol{r}(\boldsymbol{x}^{(\ell)})\right\|_2 , \log \left\|L\boldsymbol{x}^{(\ell)}\right\|_2\right\},$$

dove sulle ascisse si hanno i valori dei residui e sulle ordinate le soluzioni regolarizzate. Il criterio della curva L consiste nel considerare come parametro di regolarizzazione il parametro ℓ corrispondente all'angolo (corner) della curva.

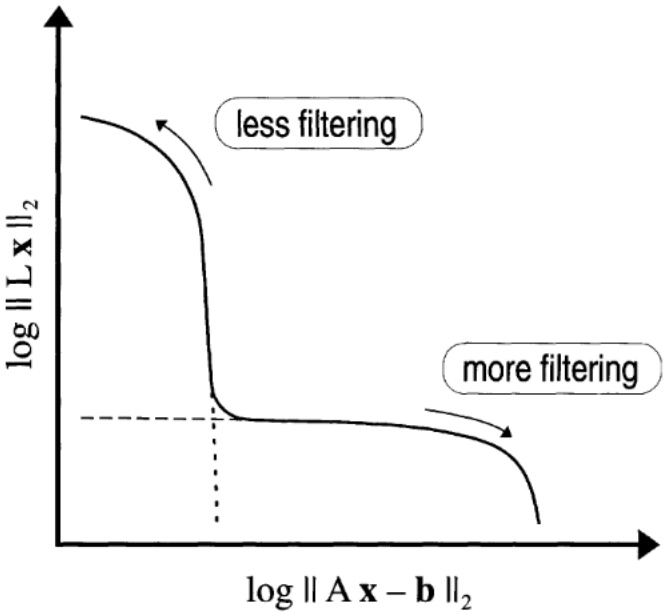

Figura 2.1: Esempio di curva L. L'immagine è stata tratta da [7]

.

2.8.3 Il criterio Optimality

Il criterio *Optimality* [4, 5] consiste nel misurare l'errore relativo:

$$e_\ell = \frac{\left\|\boldsymbol{x} - \boldsymbol{x}^{(\ell)}\right\|_2}{\|\boldsymbol{x}\|_2},$$

il parametro di regolarizzazione ottimale ℓ_{opt} è scelto in modo da minimizzare il valore di e_ℓ.

Tale criterio non è utilizzabile se non con dati sintetici, dal momento che per utilizzarlo è necessaria la conoscenza della soluzione del problema, pertanto può essere usato solo per valutare i risultati di prove numeriche.

2.8.4 Il criterio Quasi-Optimality

Il criterio *quasi-optimality* [14] sceglie ℓ senza la conoscenza del livello di rumore e senza delle ipotesi sul livello di rumore, ma in modo che sia il più piccolo indice j_q per cui:

$$\left\|\boldsymbol{x}_{j_q+1} - \boldsymbol{x}_{j_q}\right\|_2 = \min_{1\le j<\tau} \|\boldsymbol{x}_{j+1} - \boldsymbol{x}_j\|_2 .$$

2.8.5 Il criterio Quasi-Optimality ibrido

La funzione $j \to \|\boldsymbol{x}_{j+1} - \boldsymbol{x}_j\|_2$ ha generalmente molti minimi locali. Quando i minimi locali sono vicini ad essere dei minimi globali, il criterio Quasi-Optimality potrebbe fallire.

Il metodo ibrido [14] calcola un indice iniziale j_{init} con il criterio della curva L o con altri metodi e solo dopo applica il criterio Quasi-Optimality su un intorno del punto j_{init}.

Capitolo 3

Il problema inverso

Il problema inverso consiste nel trovare la distribuzione spaziale della conducibilità elettrica o della permeabilità magnetica a partire dal rapporto tra il campo primario e il campo secondario [4, 5, 3]. Quando si inverte la conducibilità elettrica, viene ipotizzato che la permeabilità magnetica abbia valori assegnati nel terreno, o comunque conosciuti, viceversa quando si inverte la permeabilità magnetica, viene ipotizzato che sia nota la conducibilità elettrica.

Il problema consiste nel trovare la conducibilità elettrica $\boldsymbol{\sigma}$ e la permeabilità magnetica $\boldsymbol{\mu}$ che approssimino al meglio:

$$M_\nu(\boldsymbol{\sigma}, \boldsymbol{\mu}; h_i, \omega_j) \approx b_{ij}^\nu,$$

dove $\nu = 0, 1$ indica l'orientazione presa in considerazione: 0 nel caso verticale e 1 nel caso orizzontale, h_i è la i-esima altezza, $i = 1, \cdots, m_h$, e ω_j è la j-esima frequenza, $j = 1, \cdots, m_\omega$.

I valori b_{ij}^ν vengono conservati all'interno di un vettore $\boldsymbol{b} \in \mathbb{C}^m$, con $m = 2m_h m_\omega$, similarmente i valori di M_ν vengono conservati in un vettore $\boldsymbol{M} \in \mathbb{C}^m$.

In [4] si è trovata la conducibilità supponendo che la permeabilità fosse conosciuta, risolvendo il problema ai minimi quadrati

$$\boldsymbol{\sigma}^* = \arg\min_{\boldsymbol{\sigma} \in \mathbb{R}^n} \frac{1}{2} \left\| \boldsymbol{r}(\boldsymbol{\sigma}) \right\|^2,$$

mentre in [3] si è trovata la permeabilità supponendo che fosse nota la conducibilità, risolvendo il problema ai minimi quadrati

$$\boldsymbol{\mu}^* = \arg\min_{\boldsymbol{\mu} \in \mathbb{R}^n} \frac{1}{2} \left\| \boldsymbol{r}(\boldsymbol{\mu}) \right\|^2,$$

in entrambi i casi il simbolo di norma è riferito alla norma 2.

I vettori $\boldsymbol{r}(\boldsymbol{x})$, dove $\boldsymbol{x}$ può essere $\boldsymbol{\sigma}$ o $\boldsymbol{\mu}$, sono detti *residui*, sono definiti come $\boldsymbol{r}(\boldsymbol{x}) = \mathrm{Re}(\boldsymbol{b} - \boldsymbol{M}(\boldsymbol{x}))$ nel caso di componenti in fase, e come $\boldsymbol{r}(\boldsymbol{x}) = \mathrm{Im}(\boldsymbol{b} - \boldsymbol{M}(\boldsymbol{x}))$ nel caso di componenti in quadratura.

Negli articoli l'algoritmo usato è quello di approssimare il residuo, supponendo $\boldsymbol{r}(\boldsymbol{x})$ differenziabile, con:

$$\boldsymbol{r}(\boldsymbol{x}_{k+1}) \simeq \boldsymbol{r}(\boldsymbol{x}_k) + J(\boldsymbol{x}_k)\boldsymbol{s}_k,$$

dove $J(\boldsymbol{x}_k)$ è la matrice Jacobiana dei residui $\boldsymbol{r}(\boldsymbol{x}) = [r_1(\boldsymbol{x}), \cdots, r_m(\boldsymbol{x})]^T$:

$$[J(\boldsymbol{x})]_{ij} = \frac{\partial r_i(\boldsymbol{x})}{\partial x_j}, \qquad i = 1, \cdots, m, \quad j = 1, \cdots, n.$$

Il metodo che è stato usato è il *metodo di Gauss-Newton smorzato* (sezione 2.2.3), che prevede la soluzione del problema ai minimi quadrati lineare:

$$\boldsymbol{s}_k = \arg\min_{\boldsymbol{s}\in\mathbb{R}^n} \|\boldsymbol{r}(\boldsymbol{x}_k) + J_k\boldsymbol{s}\|,$$

$J_k = J(\boldsymbol{x}_k)$, e il calcolo del passo successivo attraverso una lunghezza di passo α_k

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k\boldsymbol{s}_k$$

trovata con il criterio di Armijo-Goldstein.

Il calcolo della Jacobiana dei residui per $\boldsymbol{\sigma}$ è stato sviluppato in [4], mentre il calcolo della Jacobiana dei residui per $\boldsymbol{\mu}$ è stato sviluppato in [3].

La Jacobiana può essere espressa mediante SVD (sezione 2.4):

$$J_k = U\Gamma V^T,$$

dove $\Gamma = \mathrm{diag}(\gamma_1, \cdots, \gamma_p)$, con $\gamma_1 \geq \gamma_2 \geq \cdots \geq \gamma_p > 0$ e $p = \mathrm{rank}(J_k)$, mentre U e V sono matrici ortogonali con colonne $\boldsymbol{u}_i$ e $\boldsymbol{v}_i$, rispettivamente. Il numero di condizionamento della Jacobiana può essere quindi espresso nei confronti dei valori singolari:

$$\mathrm{cond}(J_k) = \frac{\gamma_1}{\gamma_p}.$$

In generale [3] ha dimostrato che la Jacobiana J_k ha un numero di condizionamento alto sia per $\boldsymbol{\sigma}$ che per $\boldsymbol{\mu}$.

Il metodo che è stato usato in [4] e [3] per risolvere questo problema è stato utilizzare l'approssimazione a basso rango (sezione 2.6) di J_k, cioè l'approssimazione di J_k con una matrice A_ℓ di rango $\ell \leq p$, in modo che:

$$\|J_k - A_\ell\|_2 = \min_{\mathrm{rank}(A)=\ell} \|J_k - A\|_2.$$

La matrice A_ℓ può essere espressa mediante SVD, $A_\ell = U_\ell\Gamma_\ell V_\ell^T$, con

$$\Gamma_\ell = \mathrm{diag}(\gamma_1, \cdots, \gamma_\ell), \qquad U_\ell = [\boldsymbol{u}_1, \cdots, \boldsymbol{u}_\ell], \qquad V_\ell = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_\ell].$$

La soluzione del problema lineare ai minimi quadrati $\min_{\boldsymbol{s}\in\mathbb{R}^n} \|\boldsymbol{r}(\boldsymbol{x}_k) + J_k\boldsymbol{s}\|$ è $\boldsymbol{s} = -J_k^\dagger\boldsymbol{r}(\boldsymbol{x}_k)$, che sostituendo A_ℓ diventa:

$$\boldsymbol{s}^{(\ell)} = -A_\ell^\dagger\boldsymbol{r}(\boldsymbol{x_k}) = -\sum_{i=1}^{\ell} \frac{\boldsymbol{u}_i^T\boldsymbol{r}(\boldsymbol{x}_k)}{\gamma_i}\boldsymbol{v}_i,$$

la soluzione $\boldsymbol{s}^{(\ell)}$ è quindi data dalla TSVD troncata ad ℓ di $-J_k^\dagger\boldsymbol{r}(\boldsymbol{x}_k)$.

Avendo la soluzione delle caratteristiche note *a priori*, ossia che conducibilità e permeabilità sono funzioni continue, in [4, 3] è stata introdotta una matrice di regolarizzazione $L \in \mathbb{R}^{t\times n}, t \leq n$, sotto l'assunto $\mathcal{N}(J_k) \cap \mathcal{N}(L) = \{0\}$.

Se $\boldsymbol{s}$ è una soluzione del problema ai minimi quadrati $\min_{\boldsymbol{s}\in\mathbb{R}^n} \|\boldsymbol{r}(\boldsymbol{x}_k) + J_k\boldsymbol{s}\|$, allora il problema diventa:

$$\min_{\boldsymbol{s}\in\mathcal{S}} \|L\boldsymbol{s}\|, \qquad \mathcal{S} = \left\{\boldsymbol{s} \in \mathbb{R}^n : J_k^TJ_k\boldsymbol{s} = -J_k^T\boldsymbol{r}(\boldsymbol{x}_k)\right\}.$$

La decomposizione a valori singolari generalizzata (GSVD, 2.5) della coppia di matrici (J_k, L) è la fattorizzazione

$$J_k = \tilde{U}\Sigma_J Z^{-1}, \qquad L = \tilde{V}\Sigma_L Z^{-1},$$

dove Σ_J e Σ_L sono matrici diagonali, le matrici $\tilde{U}$ e $\tilde{V}$ sono matrici ortogonali, e la matrice Z è non singolare.Troncando la GSVD al parametro di regolarizzazione ℓ è possibile ottenere la TGSVD 2.7:

$$\boldsymbol{s}^{(\ell)} = -\sum_{i=\bar{p}-\ell+1}^{\bar{p}} \frac{\boldsymbol{u}_{2m-p+i}^T \boldsymbol{r}}{c_i} \boldsymbol{z}_{n-p+i} - \sum_{i=\bar{p}+1}^{p} (\boldsymbol{u}_{2m-p+i}^T \boldsymbol{r}) \boldsymbol{z}_{n-p+i}, \tag{3.1}$$

dove $t > \max(0, n-2m)$, se $2m \geq n$ allora $\bar{p} = t$ e se $2m < n$ allora $\bar{p} = 2m - n + t$. Gli elementi c_i sono gli elementi di Σ_J diversi da 0 e 1.

Il passo troncato $\boldsymbol{s}_k^{(\ell)}$ serve poi per aggiornare le successive iterazioni:

$$\boldsymbol{x}_{k+1}^{(\ell)} = \boldsymbol{x}_k^{(\ell)} + \alpha_k \boldsymbol{s}_k^{(\ell)}.$$

Capitolo 4

App Designer

4.1 Scelta di App Designer

Il toolset App Designer consente di costruire interfacce grafiche in MATLAB. Questo strumento è stato introdotto dalla versione R2016a di MATLAB. L'interfaccia grafica presentata in questa tesi è stata scritta su MATLAB R2018a, ed è compatibile con tale versione e con le versioni successive di MATLAB.

Le alternative ad App Designer per la costruzione di GUI in codice MATLAB sono GUIDE, un toolset MATLAB precedente ad App Designer, e, solo col codice compatibile con GNU Octave, il framework reso disponibile da GNU Octave.

Il vantaggio di App Designer è quello di fornire un ambiente più "moderno" per la costruzione di interfacce grafiche, la stessa MathWorks invita a considerare una migrazione da GUIDE ad App Designer e ha reso disponibile un toolset per la migrazione.

App Designer ha due *View*: *Design View* e *Code View*. Quanto descritto nel seguito fa riferimento alla versione di App Designer di MATLAB R2018a.

4.2 Design View

La View *Design View* permette di definire il layout dell'interfaccia grafica, inserire graficamente i componenti nell'interfaccia. I componenti si dividono in diverse categorie:

- *Common*, i componenti in questa categoria sono i più utilizzati e ne fanno parte i grafici, i pulsanti, i pulsanti di stato, le check box, i menù, campi di testo testuali e numerici, spinner, tabelle;
- *Containers*, i componenti in questa categoria sono i componenti utili a racchiudere altri componenti, fanno parte di questa categoria i pannelli e i gruppi tab;
- *Figure Tools*, questa categoria contiene soltanto la barra dei menù;
- *Instrumentation*, questa categoria contiene vari indicatori e degli switch particolari.

All'interno della *Code View* è invece presente il codice del programma.

4.3 Code View

Parte del codice viene generato automaticamente da App Designer.
Da App Designer viene definita inizializzata una classe (`classdef`):

```
classdef nome_interfaccia < matlab.apps.AppBase
  ...
end
```

All'interno della classe si trovano `properties` e `methods`.

Nelle `properties` l'accesso può essere pubblico o privato, l'accesso pubblico comporta che quanto definito all'interno delle `properties` può essere condiviso al di fuori dell'app, l'accesso privato comporta che quanto quanto definito "rimanga" all'interno dell'app.

App Designer definisce subito a partire da quanto presente in *Design View* delle `properties` ad accesso pubblico:

```
properties (Access = public)
  nome_del_componente    classe_del_componente
end
```

Ad esempio all'interno di queste `properties` viene istanziata la figura principale:

```
properties (Access = public)
  FDEM            matlab.ui.Figure
end
```

In `methods`, con accesso privato, sempre a livello grafico (da *Design View*), vengono definiti i vari componenti attraverso la funzione `createComponents(app)`. Ad esempio l'oggetto correlato al componente della figura principale viene definito come `uifigure`

```
methods (Access = private)
  function createComponents(app)
    app.FDEM = uifigure;
  end
end
```

In `methods`, con accesso pubblico, a livello grafico, vengono definite le istruzioni principali che vengono eseguite quando viene lanciata l'interfaccia e alla chiusura dell'interfaccia:

```
methods (Access = public)
    function app = gui

        createComponents(app)
        registerApp(app, app.FDEM)

        if nargout == 0
            clear app
        end
```

```
    end

    function delete(app)
        delete(app.FDEM)
    end
end
```

`createComponents(app)` definisce gli oggetti e le loro proprietà (titolo, dimensione, etc.), mentre `registerApp(app,app.FDEM)` serve solo a definire l'app in App Designer, la function `delete(app)` si occupa della chiusura dell'app.

4.3.1 Funzioni di App Designer

Le funzioni di App Designer generano degli oggetti, all'interno di App Designer sono definite automaticamente dopo aver scelto il layer a livello grafico in *Design Tools*.

Questi oggetti possiedono delle proprietà. Alcune di queste proprietà, come la posizione, possono essere direttamente impostate da UI da *Design View*, ma solo le più utilizzate. Questo capitolo elencherà brevenente alcune proprietà, naturalmente senza avere la pretesa di esaurire le proprietà di tutti gli oggetti usati nella creazione dell'interfaccia. Una lista esaustiva delle funzioni necessarie a definire gli oggetti si trova nella documentazione di MATLAB all'indirizzo: `https://www.mathworks.com/help/matlab/functionlist.html#gui-development`. Una lista esaustiva dei componenti e della relative proprietà si trova, sempre all'interno della documentazione di MATLAB, all'indirizzo: `https://www.mathworks.com/help/matlab/creating_guis/choose-components-for-your-app-designer-app.html`.

La funzione `uifigure` serve a definire un oggetto figura, e ha alcune proprietà fondamentali: `Name`, `Position`, `Resize`, `Color`.

È possibile definire subito queste proprietà:

```
figura = uifigure('Name', 'Mia figura', ...
        'Position', [403   246   560   420], ...
        'Color', [0.9400   0.9400   0.9400], ...
        'Resize', 'off');
```

oppure modificarle in seguito:

```
figura.Name = 'Nuovo nome';
figura.Resize = 'on';
```

Per quanto riguarda il componente Axes, la `function` relativa è `uiaxes`, e ha diverse proprietà, tra le più importanti: `Visible` permette di rendere invisibili gli assi quando non servono, è una proprietà utile quando si vogliono nascondere gli assi perché non si vuole plottare qualcosa, o perché si vuole plottare una foto, `Title`, `Legend`. Un oggetto dato da `uiaxes` deve essere inserito in una figura, il primo valore in ingresso alla funzione è la `uifigure` a cui il componente appartiene:

```
a = uiaxes(figura);
a.Title.String = 'Grafico';
```

Esistono due tipi di pulsanti in App Designer, Button e State Button. Il primo esegue un comando quando viene premuto, il secondo può assumere un valore booleano (premuto/non premuto). Per il componente Button, alcune delle principali proprietà sono `Text`, `Icon`, `Enable`, `Visible`.

```
p = uibutton(figura, 'Text', 'Avvio', ...
        'Icon', '/home/user/Immagini/icon.jpg');
p.Enable = 'off';
p.Position = [450, 100, 100, 22];
```

La proprietà `Text` specifica il testo che comparirà sopra il pulsante, `Icon` è utile qualora si voglia aggiungere un'icona al pulsante, ed `Enable` darà modo di rendere premibile o non premibile un pulsante.

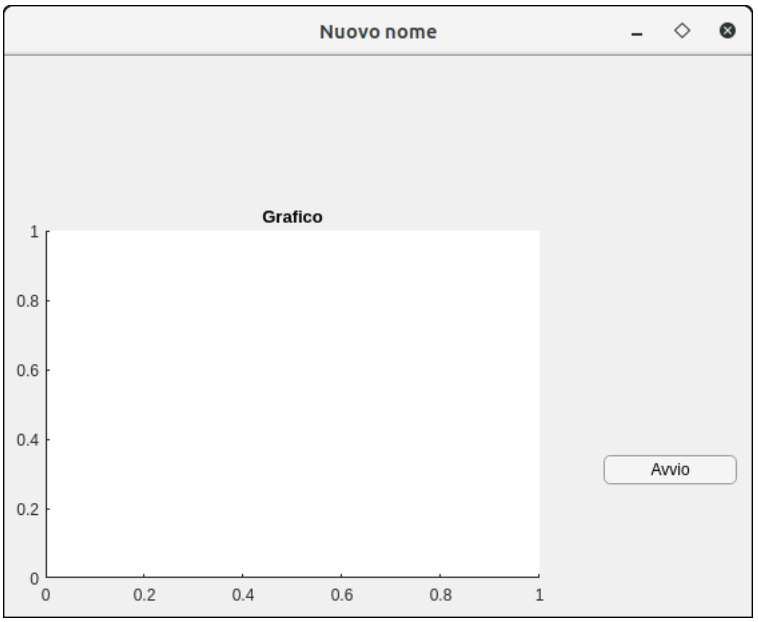

Figura 4.1: Esempio di figure in App Designer con degli assi e un pulsante.

Per il componente StateButton esiste un'altra importante proprietà, `Value`, che rappresenta il fatto che il pulsante sia o no premuto:

```
s = uibutton(figura, 'state', 'Text', 'Pulsante di stato');
s.Position = [450, 150, 100, 22];
s.Enable = 'off';
```

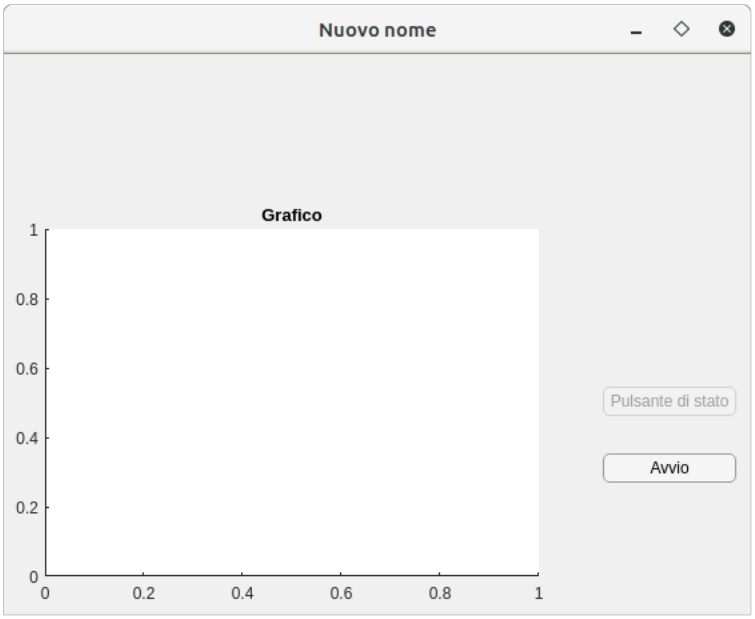

Figura 4.2: Alla figure dell'esempio precedente è stato inserito un pulsante di stato disabilitato tramite il precedente codice.

Un altro oggetto utile è la Check Box. Le Check Box (definite dalla funzione `uicheckbox`) sono delle caselle che devono essere spuntate. Anche in questo caso le proprietà fondamentali sono `Value`, `Text`, `Enable`, si comportano in modo simile agli State Button.

Le liste Drop Down permettono di selezionare tra un numero finito di elementi, le proprietà principali sono `Value` e `Items`, i quali però possono anche essere aggiunti dalla *Design View*. Anche i campi di testo possono inoltre venire abilitati o disabilitati con `Enable` o resi visibili o invisibili con `Visible`.

```
list = uidropdown(figura,...
    'Items', {'Valore 1', 'Valore 2', 'Valore 3'});
list.Enable = 'on';
list.Visible = 'on';
list.Position = [100, 365, 100, 22];
```

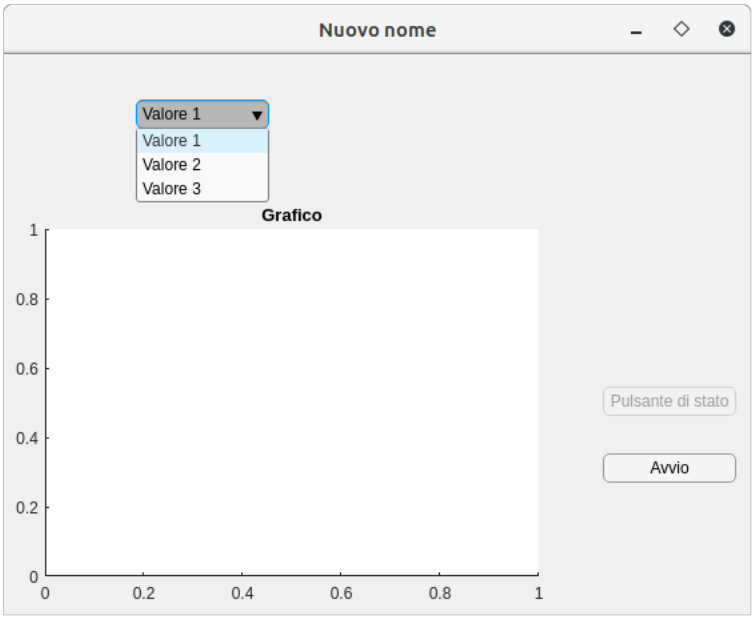

Figura 4.3: Alla figure dell'esempio precedente ora è stata inserita una lista DropDown, definita nel precedente codice.

I campi di testo si dividono in due categorie, i campi di testo numerici e i campi di testo testuali. I primi supportano un valore numerico, i secondi una stringa. Anche i campi di testo possiedono le proprietà `Enable` e `Visible`, inoltre un'altra proprietà utile è `Editable`, che rende non modificabile il valore del campo.

Per definire un oggetto *campo di testo* è necessaria la funzione `uieditfield`, il secondo parametro dato in ingresso alla funzione è *style*, che può essere `numeric` o `text`. Non definendo lo *style*, l'oggetto verrà definito automaticamente come `text`.

```
campo_numerico = uieditfield(figura, 'numeric');
campo_numerico.Editable = 'off';
campo_numerico.Value = 3.14;

campo_testuale = uieditfield(figura, 'text');
campo_testuale.Enable = 'off';
campo_testuale.Value = 'Valore';
```

Per poter effettuare una selezione tra diverse scelte sono anche utili i controlli chiamati *pulsanti di opzione*, un insieme di pulsanti viene definito dalla funzione `uibuttongroup`, mentre i singoli pulsanti vengono definiti dalla funzione `uiradiobutton`:

```
gruppo_pulsanti = uibuttongroup(figura);
pulsante_1 = uiradiobutton(gruppo_pulsanti, 'Text', 'Scelta 1');
pulsante_2 = uiradiobutton(gruppo_pulsanti, 'Text', 'Scelta 2');
```

Un'altra tipologia utile di oggetti sono gli slider, definiti dalla funzione `uislider`. Tra le proprietà più importanti di questi oggetti vi sono: `Value`, che imposta il valore dello slider; `Limits`, che definisce il range dello slider; `Enable`, che permette di abilitare o disabilitare l'oggetto; `Position`, che permette di definire la posizione dello slider.

```
slider = uislider(figura);
slider.Position = [100, 340, 200, 22];
```

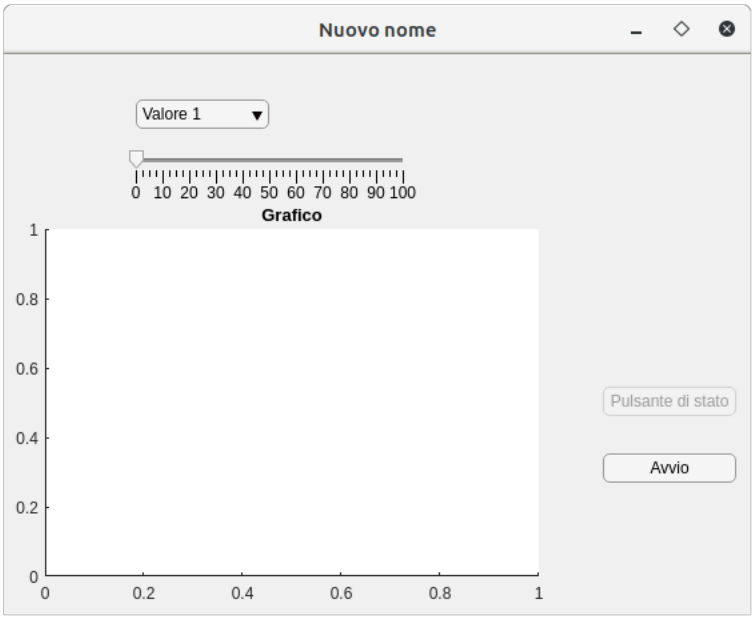

Figura 4.4: Figure dell'esempio precedente con l'aggiunta dello slider definito dal precedente codice.

4.3.2 Callbacks

Inoltre è possibile creare delle *callbacks*, ossia delle chiamate dei componenti alle relative funzioni. Le *callbacks* sono indispensabili per il funzionamento dell'interfaccia. Le callbacks sono definite all'interno dei `methods` ad accesso privato, ad esempio, per un oggetto definito come `uibutton`:

```
methods (Access = private)
  function ButtonPushed(app, event)
    ...
  end
end
```

mentre un oggetto è associato alla relativa *callbacks* attraverso la funzione `createComponents`, ad esempio per un pulsante:

```
function createComponents(app)
    app.UIFigure = uifigure;
    app.UIFigure.Position = [100 100 640 480];
    app.UIFigure.Name = 'UI Figure';
    app.Button = uibutton(app.UIFigure, 'push');
    app.Button.ButtonPushedFcn = createCallbackFcn(app, @ButtonPushed,
    true);
    app.Button.Position = [407 392 100 22]
end
```

Questo processo però è svolto direttamente da App Designer quando non si ha necessità di creare finestre esterne particolari, perciò basterà dalla *Design View* cliccare sull'oggetto desiderato col tasto destro e selezionare *Callbacks* → *Add ... callback* (nel caso del pulsante *Add ButtonPushedFcn callback*), per definire la callback.

La finestra principale avrà tre callback principali: la *StartupFcn*, la quale permette di definire delle istruzioni da eseguire all'avvio; la *CloseRequestFcn*, la quale permette di definire delle istruzioni da eseguire alla chiusura del programma; la *SizeChangedFcn*, che permette di definire le istruzioni da eseguire quando la finestra viene ridimensionata. La *StartupFcn* si rivela particolarmente utile quando si devono attribuire degli attributi particolari non impostabili da *Design View*, come per esempio far leggere una lista di elementi per una lista DropDown direttamente da una struttura dati, oppure per rendere invisibili alcuni elementi all'avvio. La *CloseRequestFcn* invece è utile per "forzare" la chiusura delle finestre secondarie quando viene chiusa la finestra principale. Nel caso di un pulsante "push" la callback sarà `ButtonPushedFcn`, che si attiverà in coincidenza dell'evento di click sul pulsante. Ad esempio può essere utile che, premendo un pulsante, su degli assi (`app.UIAxes`) compaia un grafico di un certo tipo, ad esempio la callback per il pulsante potrebbe essere:

```
function Plotta(app, event)
    t = linspace(0,5);
    y = sin(t);
    plot(app.UIAxes,t,y)
end
```

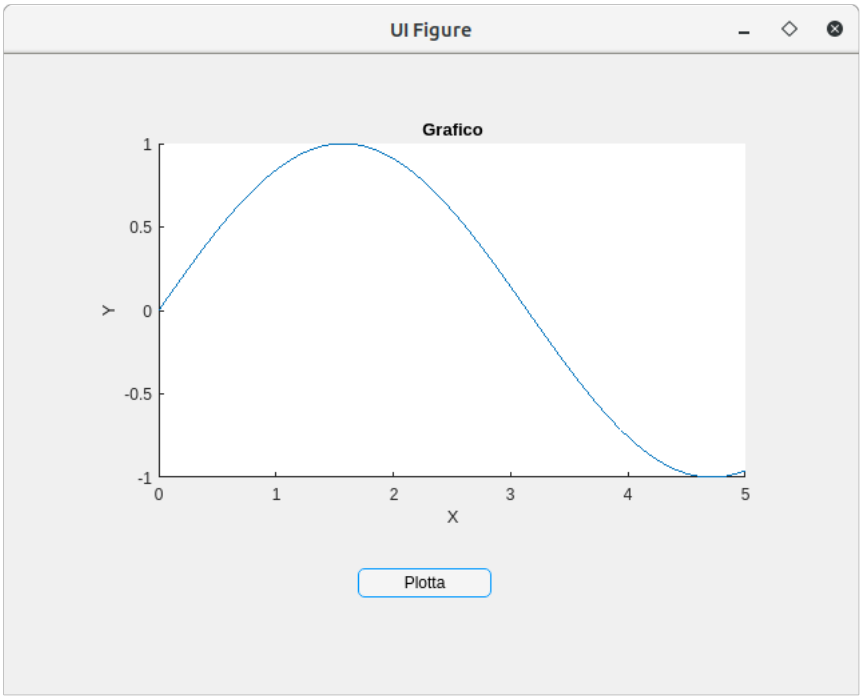

Figura 4.5: Esempio di una piccola app. Premendo il pulsante, viene disegnato il grafico.

Agli slider possono essere attribuite due callbacks, `ValueChangedFcn` e `ValueChangingFcn`, la prima callback è statica, e viene eseguita quando il valore dello slider è cambiato,

la seconda callback è più dinamica, e interviene mentre si sta cambiando il valore dello slider. Per esempio, si può sostituire al pulsante "Plotta" della precedente figura, uno slider che permetta di definire il valore della frequenza della sinusoide.

```
function omegaValueChanging(app, event)
    omega_value = event.Value;
    t = linspace(0,5);
    y = sin(omega_value*t);
  plot(app.UIAxes,t,y);
end
```

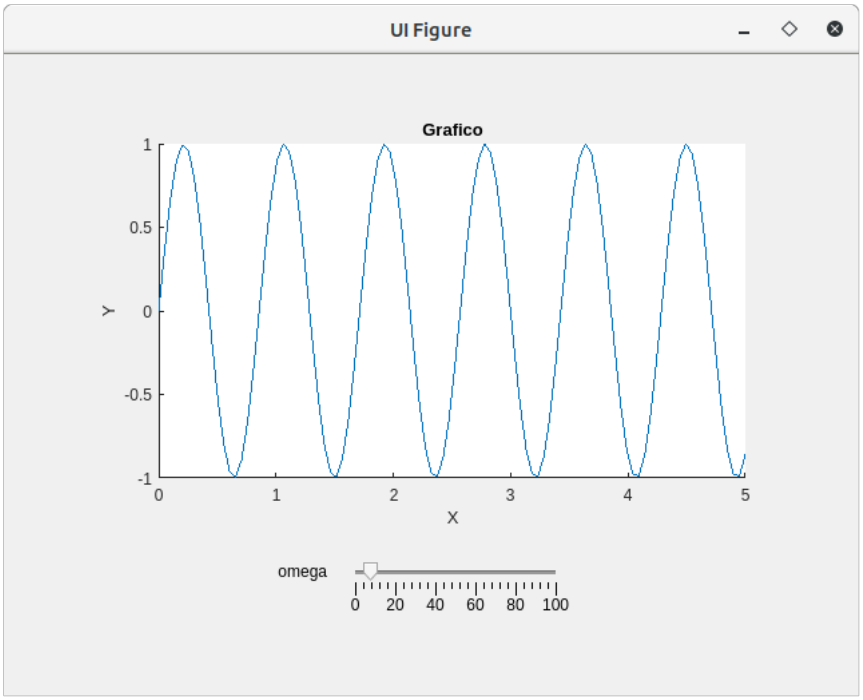

Figura 4.6: Esempio di una piccola app. Scorrendo lo slider, viene modificato il valore della pulsazione della sinusoide.

Capitolo 5

L'interfaccia grafica

L'interfaccia grafica all'esecuzione si presenta in questo modo:

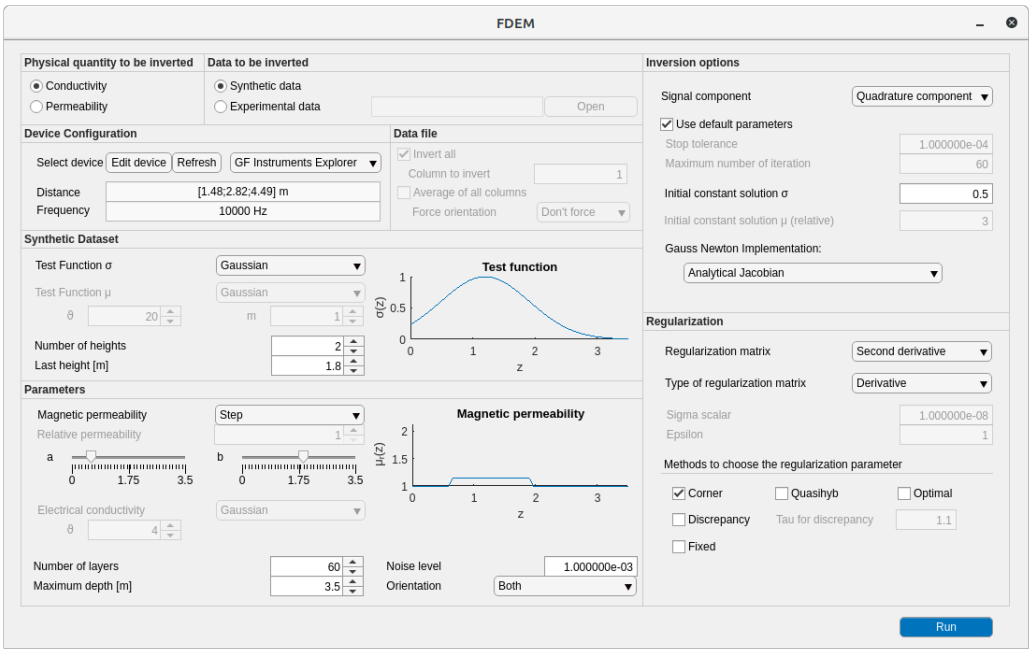

Figura 5.1: L'interfaccia grafica in esecuzione su Ubuntu GNU/Linux 17.10 (Desktop Environment GNOME 3), versione MATLAB R2018a.

Il pannello *Physical Quantity to be inverted* permette di scegliere se invertire la conducibilità elettrica o la permeabilità magnetica, considerando di conseguenza rispettivamente conosciute permeabilità magnetica e conducibilità elettrica.

Dal pannello *Data to be inverted* è possibile scegliere se lavorare con dati sintetici (*Synthetic Data*) o con dati sperimentali (*Experimental data*).

Il pannello *Device Configuration* si abilita automaticamente quando si lavora con dati sintetici, permettendo di scegliere il tipo di strumento. Cliccando sul tasto *Edit device*, si può aprire un'altra finestra da cui si possono aggiungere o rimuovere o modificare strumenti. Dopo aver modificato gli strumenti con l'editor, è utile premere *Refresh* per aggiornare la lista all'interno dell'interfaccia principale.

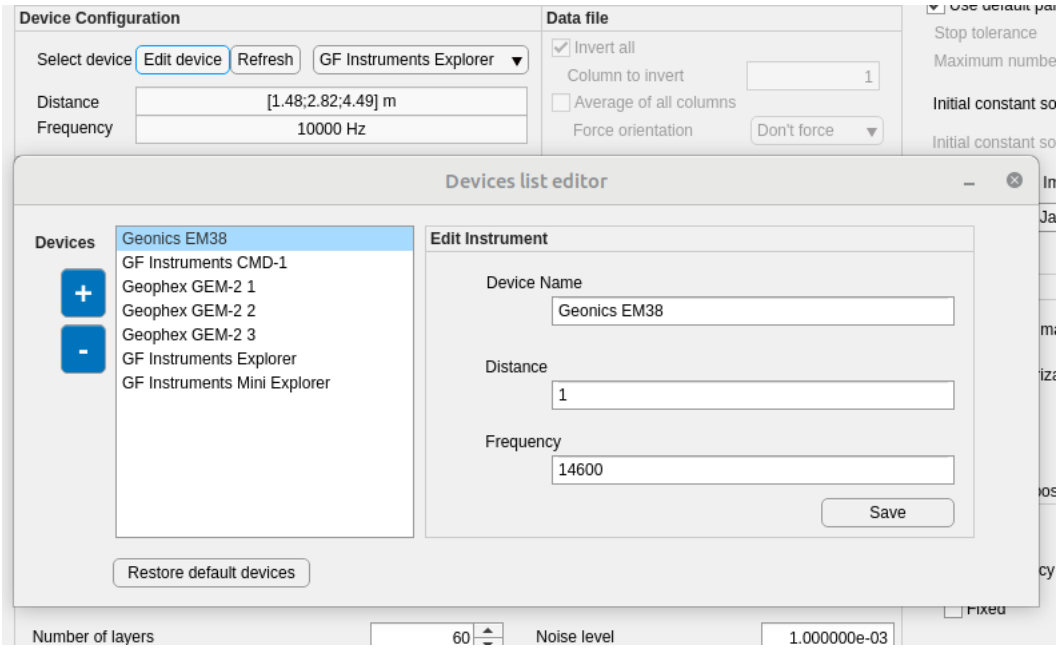

Figura 5.2: Editor degli strumenti

Il pannello *Data file* si abilita automaticamente quando vengono selezionati dati sperimentali. Da questo pannello si possono impostare varie opzioni sulle colonne, scegliere il numero di colonne da invertire nel file dei dati, invertire una colonna media, forzare l'orientazione (orizzontale o verticale) dei dati.

Il pannello *Synthetic Dataset* si attiva in caso di dati sperimentali e permette di scegliere il tipo di funzione di test per σ (gaussiana, onda triangolare, gradino), o il tipo di funzione per μ (gaussiana), il numero delle altezze e l'ultima altezza. Per σ la gaussiana, la distribuzione triangolare e il gradino sono, rispettivamente:

$$\sigma(z) = e^{-(z-1.2)^2}, \quad \sigma(z) = \begin{cases} \frac{8z+1}{5} & \text{se } z \leq \frac{1}{2} \\ \frac{6-2z}{5} & \text{se } > \frac{1}{2} \end{cases}, \quad \sigma(z) = \begin{cases} \frac{1}{5} & \text{se } z \leq \frac{1}{2}, z \geq \frac{3}{2} \\ 1 & \text{se } \frac{1}{2} < z < \frac{3}{2} \end{cases},$$

per μ la distribuzione è una gaussiana definita come:

$$\mu(z) = \vartheta_\mu e^{-(z-1.2)^2} + m,$$

dove ϑ_μ ed m sono impostabili da interfaccia.
Dal pannello *Parameters* si può scegliere il tipo di distribuzione della variabile conosciuta: μ nel caso in cui si inverta σ; σ nel caso in cui si inverta μ. È possibile decidere se μ abbia un andamento a gradino, oppure se abbia un andamento costante.
Le variabili a e b sono modificabili attraverso gli slider, se $a \leq b$:

$$\mu(z) = \begin{cases} 1 & \text{se } z < a \\ 1.14 & \text{se } a \leq z < b \\ 1 & \text{se } z \geq b \end{cases},$$

se $b < a$, allora:

$$\mu(z) = \begin{cases} 1 & \text{se } z < b \\ 2 & \text{se } b \leq z < a \\ 1 & \text{se } z \geq a \end{cases}.$$

Per quanto riguarda σ, gli andamenti impostabili sono l'andamento gaussiano, l'andamento triangolare e il gradino, dove la gaussiana, definita come:

$$\sigma(z) = \vartheta_\sigma e^{-(z-1.2)^2},$$

dall'interfaccia può essere impostato il parametro ϑ_σ facendo variare lo spinner. In *Parameters* possono anche essere modificati il numero di strati, la profondità massima, l'orientazione delle spire e il noise.

In *Inversion Options* è possibile scegliere quale componente del segnale elaborare, cioè componenti in quadratura o in fase. Alcuni parametri sono preimpostati, ma in alcune situazioni possono essere modificati, ossia il numero massimo di iterazioni del metodo di Gauss-Newton smorzato e regolarizzato, e la tolleranza per il metodo di Gauss-Newton smorzato e regolarizzato. Da questo pannello può anche essere modificata la soluzione iniziale per $\boldsymbol{\sigma}$ o $\boldsymbol{\mu}$.

Nel pannello *Inversion options* è anche possibile scegliere tra le quattro opzioni previste dal programma per l'implementazione del metodo di Gauss-Newton per il calcolo dello Jacobiano.

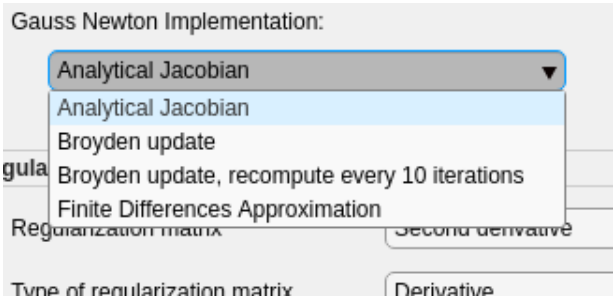

Figura 5.3: Scelta del metodo per il calcolo dello Jacobiano.

Scegliendo l'opzione *Analytical Jacobian* la Jacobiana viene calcolata ad ogni iterazione secondo i metodi analitici che sono stati presentati in [4] (per la conducibilità) e in [3] (per la permeabilità). Con l'opzione *Broyden update*, lo Jacobiano viene calcolato analiticamente la prima volta, e poi aggiornato attraverso il metodo di Broyden, mentre con l'opzione *Broyden update, recompute every 10 iterations*, lo Jacobiano viene calcolato analiticamente la prima volta, aggiornato col metodo di Broyden, e ricalcolato analiticamente ogni dieci iterazioni. Selezionando l'opzione *Finite Differences Approximation* viene approssimata la Jacobiana tramite differenze finite.

Nella finestra *Regularization*, si può scegliere la matrice di regolarizzazione (identità, approssimazione delle derivate prime, approssimazione delle derivate seconde), il tipo di matrice di regolarizzazione (derivate o regolarizzazione di precisione) e alcuni parametri (*Sigma scalar* e *Epsilon*)per la regolarizzazione di precisione.

Dal pannello *Methods to choose the regularization parameter* si possono scegliere alcuni metodi per la scelta del parametro di regolarizzazione. Il metodo *Corner* (predefinito) consiste nella ricerca dell'angolo della curva L (sezione 2.8.2); *Discrepancy* è il metodo della discrepanza (sezione 2.8.1), l'errore $\boldsymbol{e}$ è il noise definito nel pannello *Parameters* e *Tau for discrepancy* è il parametro τ_{discr}; *Optimal* è il metodo Optimality (sezione 2.8.4), *Quasihyb* corrisponde al metodo Quasi-Optimality ibrido (sezione 2.8.5), il metodo *Fixed* invece è un metodo che fissa il parametro di regolarizzazione e usa sempre quello, dopo la prima *run*, si ha la possibilità di modificare il metodo fixed usando un parametro più basso.

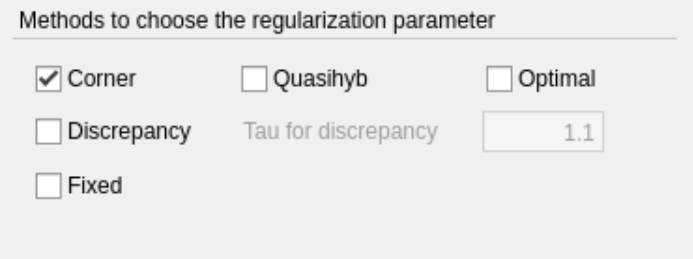

Figura 5.4: Metodi per la scelta del parametro di regolarizzazione.

Infine, dopo aver scelto le impostazioni si può avviare il programma premendo il tasto blu *Run*.

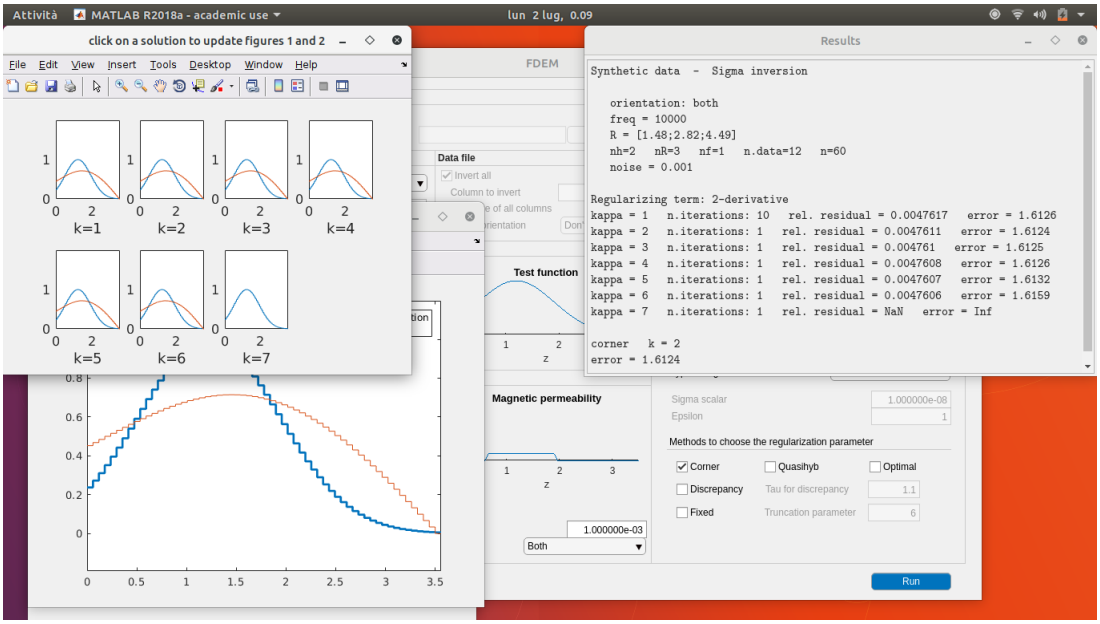

Figura 5.5: Avvio del programma dall'interfaccia.

Conclusione

L'obiettivo di questa tesi è quello di costruire una interfaccia grafica per un programma in MATLAB per l'inversione di dati EMI, in modo da rendere più agevole accedere alle funzionalità di base del programma. Per farlo è stato usato uno strumento messo a disposizione da MATLAB chiamato App Designer.
Questa interfaccia non ha la pretesa di esaurire le funzionalità e le complessità del programma che è stato sviluppato negli anni da Prof. Rodriguez, dalla Dott.ssa Díaz de Alba e da altri docenti e studenti.
Dall'interfaccia è possibile interagire col programma per fare test con dati sintetici e con dati sperimentali, risolvere il problema inverso per conducibilità elettrica e permeabilità magnetica, e fare prove con alcuni dei metodi di regolarizzazione presenti nel programma originale.

Questo lavoro è inserito nel progetto più ampio di realizzare un toolbox open source per l'inversione di dati elettromagnetici, destinato ad applicazioni nell'ambito della geofisica applicata.

Bibliografia

[1] Å. Björck, 1996, *Numerical Methods for Least Squares Problems*, SIAM.

[2] D. K. Cheng, 1989, *Field and Wave Electromagnetics* 2 ed., Addison Wesley.

[3] G. P. Deidda, P. Díaz de Alba, G. Rodriguez, 2017, *Identifying the magnetic permeability in multi-frequency EM data inversion*, Electronic Transactions on Numerical Analysis. Volume 47, pp. 1–17, 2017.

[4] G. P. Deidda, C. Fenu, G. Rodriguez, 2014, *Regularized solution of a nonlinear problem in electromagnetic sounding.*

[5] P. Díaz de Alba, G. Rodriguez, 2016, *Regularized Inversion of Multi-Frequency EM Data in Geophysical Applications*, Springer International Publishing Switzerland 2016.

[6] P. C. Hansen, 1987, *The truncated SVD as a method for regularization.*

[7] P. C. Hansen, 1999, *Rank Deficient and Discrete Ill-Posed Problems.*

[8] J. M. H. Hendrickx, B. Borchers, D. L. Corwin, S. M. Lesch, A. C. Hilgendorf, J. Schlue, 2002, *Inversion of soil conductivity profiles from electromagnetic induction measurements.*

[9] H. Huang, I. J. Won, 2003, *Automated anomaly picking from broadband electromagnetic data in an unexploded ordnance (UXO) survey*, Geophysics.

[10] J. A. Doolittle, E. C. Brevik, 2014, *The use of electromagnetic induction techniques in soils studies.*

[11] S. M. Lesch, D. J. Strauss, J. D. Rhoades, 1995, *Spatial prediction of soil salinity using electromagnetic induction techniques: 1. Statistical prediction models: A comparison of multiple linear regression and cokriging.*

[12] J. D. McNeill, 1980, *Electromagnetic terrain conductivity measurement at low induction numbers.*

[13] A. Osella, M. de la Vega, E. Lascano, 2005, *3-D electrical imaging of archaeological sites using electrical and electromagnetic method*, Geophysics.

[14] L. Reichel, G. Rodriguez, 2013, *Old and new parameter choice rules for discrete ill-posed problems.*

[15] G. Rodriguez, 2008, *Algoritmi numerici*, Pitagora Editrice Bologna.

[16] J. R. Wait, 1982, *Geo-Electromagnetism*, Academic Press.