POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Aerospaziale

Tesi di Laurea

Sviluppo di un'interfaccia grafica in MATLAB per la risoluzione e visualizzazione di campi di moto compositi

Relatrice:
Ing. Stefania Scarsoglio
Correlatore:
Prof. Renzo Arina

Candidata:
Marta Talmelli

Luglio 2015

Indice

Nomenclatura

Simbolo	Descrizione
$\mathbf{V}$	Vettore velocità
(x, y)	Coordinate cartesiane nel piano
(r, θ)	Cordinate polari nel piano
W	Modulo del vettore velocità
α	Inclinazione della corrente uniforme o della doppietta rispetto all'asse x
u	Componente della velocità secondo l'asse x, corrisponde con la componente orizzontale in un campo bidimensionale
v	Componente della velocità secondo l'asse y, corrisponde con la componente verticale in un campo bidimensionale
w	Componente della velocità secondo l'asse z
u_r	Componente della velocità secondo r
u_θ	Componente della velocità secondo θ
$d\mathbf{s}$	Vettore di tratto infinitesimo
ψ	Funzione di corrente
ϕ	Funzione potenziale
Q	Intensità della sorgente o del pozzo
μ	Intensità della doppietta
Γ	Intensità del vortice
ω	Vorticità
p	Pressione
c_p	Coefficiente di pressione
ρ	Densità

Le unità di misura delle grandezze non adimensionali sono coerenti con il Sistema Internazionale di unità di misura (SI).
Le grandezze vettoriali sono indicate con lo stile **grassetto**.

Elenco delle figure

Introduzione

L'elaborato ha lo scopo di realizzare un'interfaccia grafica partendo dall'algoritmo, scritto in MATLAB, che grazie alla validità del principio di sovrapposizione degli effetti fornisce la soluzione del campo di moto composito ottenuto dalla somma di campi di moto semplici. L'interfaccia sarà creata attaverso una GUI di MATLAB personalizzata in modo da risolvere e visualizzare interattivamente i diversi campi di moto.
Nell'interfaccia si vuole poter inserire più campi semplici sia di diverso tipo, sia dello stesso tipo e per ognuno di essi si dovrà poter decidere dove posizionarlo e le corrispondenti intensità ed inclinazione. Inoltre un altro obbiettivo è quello di poter segliere cosa visualizzare del campo di moto finale tra linee di corrente, isolinee della funzione di corrente ψ ed isolinee della funzione potenziale ϕ, oppure queste ultime due contemporaneamente.
Nella stesura dell'elaborato vengono inizialmente date alcune definizioni utili per comprendere il lavoro e successivamente sono definiti ed illustrati i campi di moto semplici. Infine verranno esposti l'architettura, il funzionamento ed il metodo di visualizzazione prima dell'algoritmo e poi dell'interfaccia grafica.

Capitolo 1

Definizioni

In questo capitolo verranno fornite delle definizioni utili per comprendere al meglio il seguito del lavoro.

1.1 Linee di corrente

La linea di corrente o linea di flusso è una curva tangente al vettore velocità in ogni suo punto ad un dato istante di tempo. Le linee di corrente, anche dette streamlines, rappresentano quindi la direzione in cui una particella fluida si muove in ogni istante nel tempo ed equivalgono alle linee di campo del campo vettoriale delle velocità. Esse possono essere considerate come una fotografia istantanea del campo di moto.
Data la condizione di parallelismo tra il vettore velocità

$$\mathbf{V}(\mathbf{r}, t) = u\mathbf{i} + v\mathbf{j} + w\mathbf{k}$$

ed il vettore

$$d\mathbf{s}(\mathbf{r}, t) = dx\mathbf{i} + dy\mathbf{j} + dz\mathbf{k}$$

che rappresenta un tratto infinitesimo della linea stessa, si definisce l'equazione differenziale della linea di corrente:

$$d\mathbf{s}(\mathbf{r}, t) \times \mathbf{V}(\mathbf{r}, t) = 0, \tag{1.1}$$

la quale è equivalente alle tre relazioni scalari:

$$\begin{cases} wdy - vdz = 0 \\ udz - wdx = 0 \\ vdx - udy = 0 \end{cases} \tag{1.2}$$

ed in forma simmetrica:

$$\frac{dx}{u} = \frac{dy}{v} = \frac{dz}{w}. \tag{1.3}$$

Nella trattazione seguente si prenderanno in esame solo campi di moto bidimensionali, aventi componenti x ed y, e stazionari.

1.2 Funzione di corrente

La forma di una linea di corrente si ottiene integrando l'equazione 1.1 coinvolgendone solo le variabili spaziali, cioè per un istante fissato di tempo. La soluzione, quindi la linea di corrente, è individuata dall'intersezione di due superfici:

$$\begin{cases} \psi_1(x, y, z) = C_1 \\ \psi_2(x, y, z) = C_2 \end{cases}$$

con C_1 e C_2 costanti. Le funzioni ψ_1 e ψ_2 sono dette funzioni di corrente. Si nota che lungo la linea di corrente il vettore velocità è perpendicolare ai gradienti delle superfici:

$$\mathbf{V} \cdot \nabla\psi_1 = 0$$

$$\mathbf{V} \cdot \nabla\psi_2 = 0$$

e quindi ne consegue che:

$$\mathbf{V} = \nabla\psi_1 \times \nabla\psi_2$$

Le funzioni di corrente sono particolarmente utili nel caso di flussi bidimensionali e incompressibili che è il caso preso in esame in questo elaborato. Nel seguito si considera costante la direzione z e quindi risulta:

$$\mathbf{V} = u(x, y)\mathbf{i} + v(x, y)\mathbf{j}$$

$$\begin{cases} \psi_1 = \psi(x, y) \\ \psi_2 = z \end{cases}$$

Su una linea di corrente la funzione $\psi(x, y)$ è costante e la relazione tra essa e il campo vettoriale di velocità in coordinate cartesiane è:

$$\begin{cases} u = \frac{\partial\psi}{\partial y} \\ v = -\frac{\partial\psi}{\partial x} \end{cases} \tag{1.4}$$

ed in coordinate cilindriche (r, θ):

$$\begin{cases} u_r = \frac{1}{r}\frac{\partial\psi}{\partial\theta} \\ u_\theta = -\frac{\partial\psi}{\partial r} \end{cases} \tag{1.5}$$

Considerando ora un tratto ds di una linea che congiunge due generici punti Q e P ed il corrispondente versore perpendicolare alla linea $\mathbf{n} = (dx, -dy)$, si ha che la variazione della funzione di corrente lungo tale linea è:

$$\psi(P) - \psi(Q) = \int_Q^P d\psi = \int_Q^P -vdx + udy = \int_Q^P \mathbf{V} \cdot \mathbf{n} ds$$

Quindi la variazione di ψ coincide con la portata in volume attaverso la superficie individuata dalla linea QP e di profondità unitaria. Risulta pertanto che la funzione ψ dipende dal percorso di integrazione ed è così definita a meno di una costante. Si nota che lungo circuiti che racchiudono discontinuità ($\nabla \cdot V \neq 0$) la funzione di corrente è a valori multipli.
Si osserva che le linee a ψ costante coincidono con le linee di corrente.
Infine grazie alle condizioni di incompressibilità ($\nabla \cdot V = 0$) e di irrotazionalità ($\omega = \nabla \times \mathbf{V} = 0$) la funzione di corrente è una funzione armonica soluzione dell'equazione di Laplace $\nabla^2 \psi = 0$.

1.3 Funzione potenziale

La funzione potenziale ϕ è una funzione scalare tale che:

$$\mathbf{V} = \nabla \phi, \tag{1.6}$$

la quale scritta in coordinate cartesiane e considerando un flusso bidimensionale diventa:

$$\begin{cases} u = \frac{\partial \phi}{\partial x} \\ v = \frac{\partial \phi}{\partial y} \end{cases} \tag{1.7}$$

e rispettivamente in coordinate cilindriche:

$$\begin{cases} v_r = \frac{\partial \phi}{\partial r} \\ v_\theta = \frac{1}{r} \frac{\partial \phi}{\partial \theta} \end{cases} \tag{1.8}$$

La funzione ϕ è quindi determinata a meno di una costante poichè, com'è noto, il gradiente di una costante è nullo.
Si osserva che in un dominio semplicemente connesso la funzione potenziale è a valore singolo e non dipende dal percorso di integrazione. Invece in un dominio doppiamente connesso la funzione ϕ è a valori multipli, cioè può assumere più valori che differiscono di un multiplo di una costante Γ che coincide con il valore della circuitazione.

Le linee a ϕ costante sono dette linee equipotenziali.
Infine si nota che per la condizione di incompressibilità:

$$\nabla \cdot \mathbf{V} = \nabla \cdot \nabla\phi = \nabla^2\phi = 0$$

anche la funzione potenziale è una funzione armonica soluzione dell'equazione di Laplace.
Confrontando le relazioni 1.4 e 1.7 si ottengono le condizioni di Cauchy-Reimann:

$$\begin{cases} \frac{\partial\phi}{\partial x} = \frac{\partial\psi}{\partial y} \\ \frac{\partial\phi}{\partial y} = -\frac{\partial\psi}{\partial x} \end{cases} \tag{1.9}$$

Pertanto ψ e ϕ sono dette funzioni armoniche coniugate e risulta che le linee a ψ costante e le linee a ϕ costante sono due famiglie di linee mutualmente ortogonali. Da ciò si deduce che il vettore velocità risulta sempre perpendicolare alle linee equipotenziali.

Capitolo 2

Campi di moto semplici

In questo capitolo vengono descritti i campi di moto semplici su cui saranno basati l'algoritmo e l'interfaccia in MATLAB per ottenere dei campi di moto compositi grazie al principio della sovrapposizione degli effetti. I campi di moto semplici vengono inizialmente definiti posizionati nell'origine del sistema di riferimento, ma nel seguito saranno poi spostati in un punto qualsiasi (x_0, y_0) del dominio stabilito.

2.1 Corrente uniforme

Si considera una corrente uniforme bidimensionale con velocità $\mathbf{V}_\infty = u\mathbf{i} + v\mathbf{j}$ e si ottengono rispettivamente la funzione potenziale e di corrente:

$$\phi = ux + vy \tag{2.1}$$

$$\psi = -vx + uy \tag{2.2}$$

avendo posto $\phi(0,0) = 0$ e $\psi(0,0) = 0$.
Le linee a ψ costante, conincidendo con le linee di corrente, sono parallele al vettore $\mathbf{V}_\infty$, mentre le linee equipotenziali ($\phi =$ costante) sono perpendicolari ad esso.
Inoltre la corrente uniforme può essere inclinata di un angolo α rispetto all'asse x; se consideriamo un'intensità pari a W si ottengono le componenti di velocità:

$$\begin{cases} u = W\cos\alpha \\ v = W\sin\alpha \end{cases}$$

2.2 Sorgente e pozzo puntiformi

Una sorgente puntiforme bidimensionale emette una portata in volume per unità di profondità Q ed è definita dalla condizione che le linee di corrente sono semirette

radiali con origine nella sorgente e che il vettore velocità è $\mathbf{V} = u_r(r)\mathbf{e_r}$, dove r è la distanza radiale dalla sorgente. In questo modo si hanno rispettivamente la funzione potenziale e di corrente in coordinate polari:

$$\phi = \frac{Q}{2\pi} log\left(\frac{r}{r_0}\right) \tag{2.3}$$

$$\psi = \frac{Q}{2\pi}\theta, \quad 0 \leq \theta < 2\pi \tag{2.4}$$

avendo assunto $\phi(r_0, 0) = 0$ e $\psi(r, 0) = 0$. Queste ultime si possono riscrivere in coordinate cartesiane:

$$\phi = \frac{Q}{2\pi} log\left(\frac{x^2 + y^2}{x_0^2 + y_0^2}\right)$$

$$\psi = \frac{Q}{2\pi} arctan\left(\frac{y}{x}\right)$$

Quindi risulta che le linee a ψ costante sono delle semirette con origine nella sorgente e le linee a ϕ costante sono delle circonferenze concentriche anch'esse con origine nella sorgente stessa. Nella figura 2.1 si osserva una discontinuità posizionata in $(r, \theta) = (r, \pi)$ dovuta alla proprietà della funzione di corrente di essere a valori multipli lungo circuiti che racchiudono una singolarità, che in questo caso è proprio la sorgente.

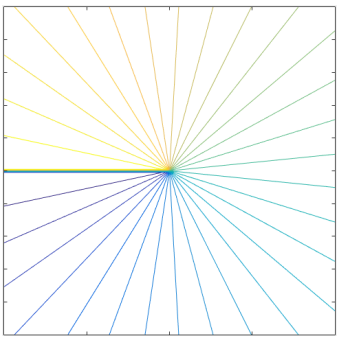

Figura 2.1: Funzione di corrente sorgente

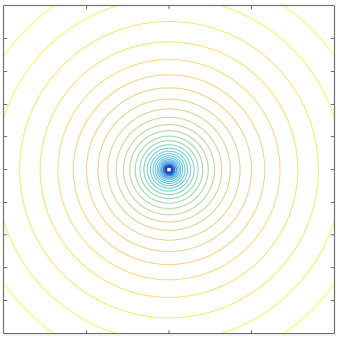

Figura 2.2: Funzione potenziale sorgente

Il corrispondente campo di velocità della sorgente, rispettivamente in coordinate polari e cartesiane, è:

$$\begin{cases} u_r = \frac{1}{r}\frac{\partial \psi}{\partial \theta} = \frac{\partial \phi}{\partial r} = \frac{Q}{2\pi r} \\ u_\theta = -\frac{\partial \psi}{\partial r} = \frac{1}{r}\frac{\partial \phi}{\partial \theta} = 0 \end{cases}$$

$$\begin{cases} u = \frac{\partial \psi}{\partial y} = \frac{\partial \phi}{\partial x} = \frac{Q}{2\pi} \frac{x}{x^2+y^2} \\ v = -\frac{\partial \psi}{\partial x} = \frac{\partial \phi}{\partial y} = \frac{y}{x^2+y^2} \end{cases}$$

Infine si definisce in modo analogo un pozzo puntiforme bidimensionale, ma con la carattesitica di una portata in volume per unità di profondità Q negativa, cioè questa viene assorbita.

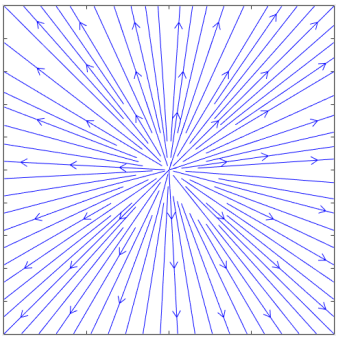

Figura 2.3: Linee di corrente sorgente

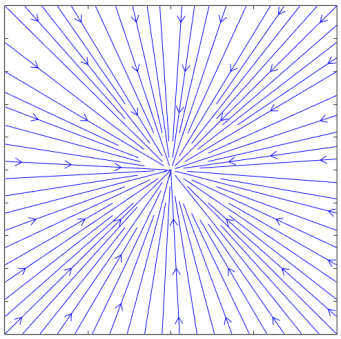

Figura 2.4: Linee di corrente pozzo

2.3 Doppietta

La doppietta si può descrivere come la somma dei campi di una sorgente e di un pozzo di uguale intensità Q la cui distanza l tende a zero, quindi bisogna porre $Q \to \infty$ in modo che il prodotto $\mu = lQ$, chiamato intensità del campo, sia finito. La doppietta è completamente identificata attraverso il vettore μ, detto momento della doppietta, centrato nel punto dov'è applicata la singolarità, cioè la doppietta stessa, con direzione coincidente con l'asse che unisce la sorgente ed il pozzo e verso dal pozzo verso la sorgente.
Nel caso generale la doppietta può essere inclinata di un angolo α rispetto all'asse x e si definiscono così in coordinate polari la funzione potenziale e di corrente:

$$\phi = -\frac{\mu}{2\pi r} \sin(\theta - \alpha) \tag{2.5}$$

$$\psi = \frac{\mu}{2\pi r} \cos(\theta - \alpha) \tag{2.6}$$

ed in coordinate cartesiane:

$$\phi = -\frac{\mu}{2\pi} \frac{x}{x^2 + y^2}$$

$$\psi = \frac{\mu}{2\pi} \frac{y}{x^2 + y^2}$$

Considerando $\alpha = 0$ e una generica costante C si ha che le linee equipotenziali risultano essere delle circonferenze tangenti all'origine e con centro nei punti $(\frac{\mu}{4\pi C}, 0)$, pertanto per $C \to 0$ la circonferenza degenera in una retta coincidente con l'asse y ed è la linea equipotenziale $\phi = 0$. Analogamente le linee di corrente sono delle circonferenze tangenti all'origine e con centro nei punti $(0, \frac{\mu}{4\pi C})$, la linea $\psi = 0$ si ottiene con $C \to 0$ ed è la circonferenza degenere coincidente con l'asse x.

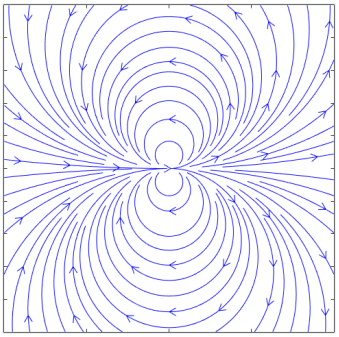

Figura 2.5: Linee di corrente doppietta

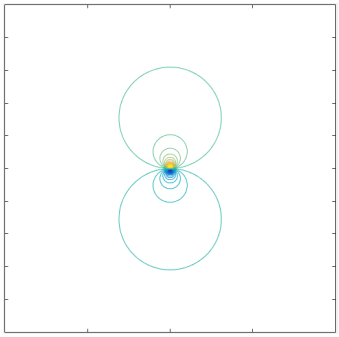

Figura 2.6: Funzione di corrente doppietta

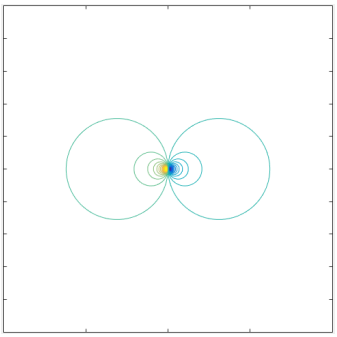

Figura 2.7: Funzione potenziale doppietta

Infine il corrispondente campo di velocità in coordinate polari e cartesiane è:

$$\begin{cases} u_r = \frac{\mu}{2\pi r^2} \cos(\theta - \alpha) \\ u_\theta = \frac{\mu}{2\pi r^2} \sin(\theta - \alpha) \end{cases}$$

$$\begin{cases} u = \frac{\mu}{2\pi} \frac{(x^2 - y^2)\cos\alpha + 2xy\sin\alpha}{(x^2+y^2)^2} \\ v = \frac{\mu}{2\pi} \frac{(y^2 - x^2)\sin\alpha + +2xy\cos\alpha}{(x^2+y^2)^2} \end{cases}$$

2.4 Vortice puntiforme

Il vortice puntiforme irrotazionale è una singolarità in cui è concentrata una vorticita ω di intensità infinita, mentre in ogni altro punto del campo di moto la vorticità è nulla. La singolarità del vortice rende il dominio a connessione multipla.
Il vortice è definito da linee di corrente coincidenti con circonferenze concentriche aventi centro nel vortice stesso e dal vettore velocità $\mathbf{V} = \frac{\Gamma}{2\pi r}\mathbf{e}_\theta$, dove Γ rappresenta l'intensità del vortice ed è positiva quando le particelle traslano lungo le traiettorie circolari in senso antiorario. In questo modo si ottengono in coordinate polari rispettivamente la funzione potenziale e di corrente:

$$\phi = \frac{\Gamma}{2\pi}\theta, \quad 0 \leq \theta < 2\pi \tag{2.7}$$

$$\psi = -\frac{\Gamma}{2\pi} log\left(\frac{r}{r_0}\right) \tag{2.8}$$

avendo posto $\phi(r,0) = 0$ e $\psi(r_0,0) = 0$. Le quali riscritte in coordinate cartesiane sono:

$$\phi = \frac{\Gamma}{2\pi} arctan\left(\frac{y}{x}\right)$$

$$\psi = -\frac{\Gamma}{4\pi} log\left(\frac{x^2+y^2}{x_0^2+y_0^2}\right)$$

Risulta che le linee a ϕ costante sono delle semirette con origine nel vortice, mentre le linee a ψ costante sono delle circonferenze concentriche sempre con origine nel vortice. Come si nota in figura 2.10 la funzione potenziale ha una discontinuità in $(r,\theta) = (r,\pi)$ dovuta alla sua proprietà di essere una funzione a valori multipli in un dominio doppiamente connesso, qual è quello contenente un vortice.

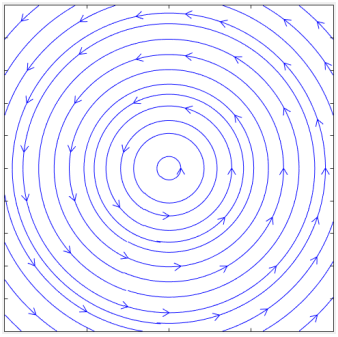

Figura 2.8: Linee di corrente vortice

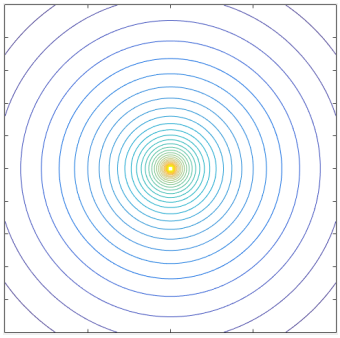

Figura 2.9: Funzione di corrente vortice

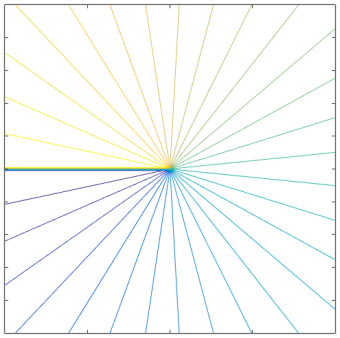

Figura 2.10: Funzione potenziale vortice

Quindi le componenti polari e cartesiane del campo di velocità del vortice sono:

$$\begin{cases} u_r = 0 \\ u_\theta = \frac{\Gamma}{2\pi r} \end{cases}$$

$$\begin{cases} u = -\frac{\Gamma}{2\pi}\frac{y}{x^2+y^2} \\ v = \frac{\Gamma}{2\pi}\frac{x}{x^2+y^2} \end{cases}$$

Capitolo 3

Algoritmo in MATLAB

In questo capitolo vengono descritti il funzionamento dell'algoritmo, scritto in MATLAB, e la modalità di visualizzazione a video dei vari campi di moto.

3.1 Impostazione e funzionamento algoritmo

L'algoritmo si basa sul principio di sovrapposizione degli effetti valido per i campi di moto semplici sopra descritti. Si sviluppa quindi in 5 blocchi di cui 4 riguardanti i 4 diversi campi di moto semplici e l'ultimo nel quale questi vengono sommati per ottenere il rispettivo campo di moto composito.
In tutto lo script si considera un campo di moto incompressibile, stazionario, irrotazionale e bidimensionale. Il dominio è stabilito dalle coordinate $-1 \leq x \leq 1$ e $-1 \leq y \leq 1$. Cliccando sul tasto 'Run' di MATLAB a video, tramite la funzione 'input', viene richiesto l'inserimento dei seguenti dati:

- intensità del campo uniforme W,
- incidenza in gradi del campo uniforme $0 \leq \alpha < 180°$,
- numero di sorgenti/pozzi che si vogliono inserire,
- intensità Q di ogni sorgente/pozzo inserito,
- coordinata orizzontale x e verticale y per ogni sorgente/pozzo inserito,
- numero di doppiette che si vogliono inserire,
- intensità μ di ogni doppietta inserita,
- coordinata orizzontale x e verticale y per ogni doppietta inserita,

- inclinazione in gradi rispetto all'asse x della doppietta, positiva in senso orario,
- numero di vortici che si vogliono inserire,
- intensità Γ di ogni vortice inserito,
- coordinata orizzontale x e verticale y per ogni vortice inserito.

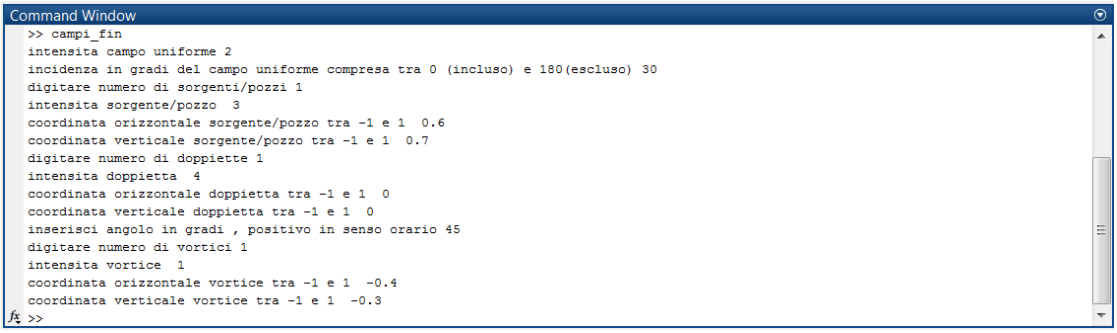

Figura 3.1: Esempio di inserimento dati

Inizialmente si è definito per ogni dato da inserire una matrice di zeri. Dopo aver inserito i dati desiderati vengono effettuati dei controlli su essi ed in particolare si trasformano gli angoli espressi in gradi in radianti tramite la formula $\alpha(rad) = \frac{\alpha\pi}{180}$. L'algoritmo salva ora i vari dati nelle rispettive matrici, grazie alle quali calcola per ogni campo di moto semplice le componenti di velocità u e v, la funzione di corrente ψ e la funzione potenziale ϕ. Questi calcoli vengono effettuati seguendo una procedura iterativa, per ogni campo semplice, tramite l'utilizzo di cicli for annidati: due riguardanti le righe e le colonne delle matrici delle varie grandezze e l'ultimo relativo al numero di campi semplici inseriti per ogni specifico tipo.
Pertanto ora, valendo appunto il principio di sovrapposizione degli effetti, vengono calcolati le componenti di velocità U e V, le funzioni ψ e ϕ del campo di moto composito, ottenuto dai dati inseriti, sommando semplicemente le componenti e le funzioni dei vari campi di moto semplici.
Inoltre, avendo fatto l'ipotesi di campo irrotazionale, se nel campo di moto composito finale, con componenti di velocità U e V, è presente una corrente uniforme di intensità W, l'algoritmo calcola il campo di pressione p tramite l'equazione di Bernoulli:

$$p = \frac{\rho}{2}(U^2 + V^2)$$

dove per questioni di semplicità si è imposto $\rho = 1.225 kg/m^3$ che è la densità dell'aria standard a quota zero ed alla pressione di riferimento di $1 atm$. Infine per evitare questa approssimazione, si preferisce calcolare anche il coefficiente

adimensionale di pressione tramite la relazione:

$$c_p = 1 - \left(\frac{U^2 + V^2}{W}\right)^2$$

3.2 Visualizzazione

Nell'ultimo blocco, sopra citato, dell'algoritmo, oltre ad essere presente la somma dei vari campi di moto semplici, sono scritti anche i comandi per visualizzare a video le varie grandezze calcolate del campo di moto composito. Ognuna di queste appare su una finestra indipendente grazie al comando 'figure', ma esse risultano in finestre sovrapposte l'una sull'altra.
Nella prima figura viene rapprensentato il campo delle velocità, cioè le linee di corrente con rispettivo verso di percorrenza, tramite la funzione 'streamslice($x, y, U, V, 1.5$)', dove 1.5 rappresenta la densità delle linee, cioè varia la spaziatura tra esse.

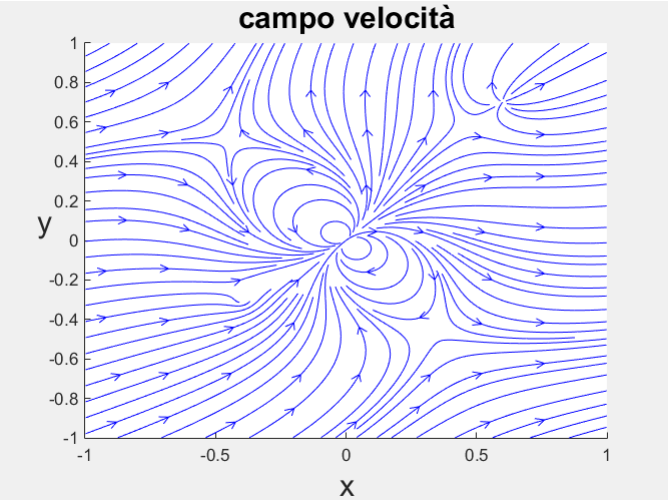

Figura 3.2: Corrispondenti linee di corrente del'esempio in figura 3.1

Successivamente in un'altra figura si rappresenta la funzione di corrente ψ attraverso la funzione 'contour(x, y, ψ,50,'LineWidth',2)'. Nella figura appaiono delle linee coincidenti con le linee di corrente prima rappresentate, ma ciascuna con colore e quindi intensità diversa, perciò viene inserita al fianco dell'immagine una 'colorbar' per identificare i valori di intensità di ogni linea. All'interno delle parentesi tonde si inseriscono le specifiche caratteristiche desiderate: 50 indica quante isolinee verranno rappresentate, mentre 'LineWidth',2 imposta la larghezza della linea di contorno a 2.

Analogamente si rappresenta anche la funzione potenziale ϕ e, come dimostra la teoria, si nota che in ogni punto risulta essere ortogonale alla funzione di corrente. In un'altra figura ancora si visualizza il campo di pressione grazie alla funzione 'surf(x, y, p)'; a differenza delle altre grandezze rappresentate, si ottiene una superficie tridimensionale con picchi di pressione in corrispondenza delle singolarità. Invece anche qui la superficie assume colorazione e quindi valori diversi nei vari punti, perciò è inserita una 'colorbar' per chiarezza.
Infine, nell'ultima figura viene rappresentato il coefficiente di pressione tramite la funzione 'contour(x, y, c_p,50, 'LineWidth',2)', le caratteristiche della rappresentazione sono analoghe a quelle descritte nella funzione di corrente. Inoltre, come nel campo di pressione, anche in questo caso si notano dei picchi, attraverso la diversa colorazione, in corrispondenza delle singolarità presenti nel campo di moto.
In ogni figura sono stati inseriti titolo e nome degli assi, il primo tramite la funzione 'title' e gli altri grazie a 'xlabel' e 'ylabel'.

Capitolo 4

Interfaccia grafica

In questo capitolo si descrive com'è stata creata, come funziona e come appare l'interfaccia grafica personalizzata realizzata tramite MATLAB. Essa è stata impostata partendo dall'algoritmo prima descritto. Infine verranno fatti alcuni esempi di campi di moto compositi ottenibili attraverso l'interfaccia stessa.

4.1 Funzionamento e visualizzazione

L'interfaccia grafica è stata creata su MATLAB, per prima cosa bisogna digitare 'guide' nella command window e selezionare una 'Blank GUI'. Nel foglio bianco ora apparso si decide dove e cosa inserire tra i vari componenti disponibili ed ad ognuno si dà un nome ed un tag che serviranno per poi attribuirgli i comandi che dovrà realizzare. Dopo aver definito l'aspetto grafico, si salva l'interfaccia e di default MATLAB crea uno script corrispondente ad essa che salva come classico 'file.m' ed un altro file di estensione '.fig'.
A questo punto, avendo creato la grafica dell'interfaccia, si devono inserire nello script le varie istruzioni che si vogliono che questa realizzi. Si inizia a scrivere nella 'Opening Function' dove si definisce il corpo principale dello script e poi in corrispondenza di ogni componente che si è inserito precedentemente si scrivono gli specifici comandi che devono essere realizzati da quel determinato componente e si riesce a far ciò grazie al tag prima definito per ciascuno di essi ed alla funzione 'Callback'. Ogni variabile definita e calcolata nel corpo principale dello script viene associata ad una funzione 'handles', cioè un puntatore, che serve poi a chiamare la variabile stessa nel seguito.
Tramite il procedimento appena descritto, per raggiungere l'obiettivo di questo elaborato, è stata realizzata l'interfaccia grafica visibile in figura 4.1. Come si vede essa è composta da 5 static text, 5 pop-up menu, 20 edit text, 5 push botton, 1 axes e 2 panel.

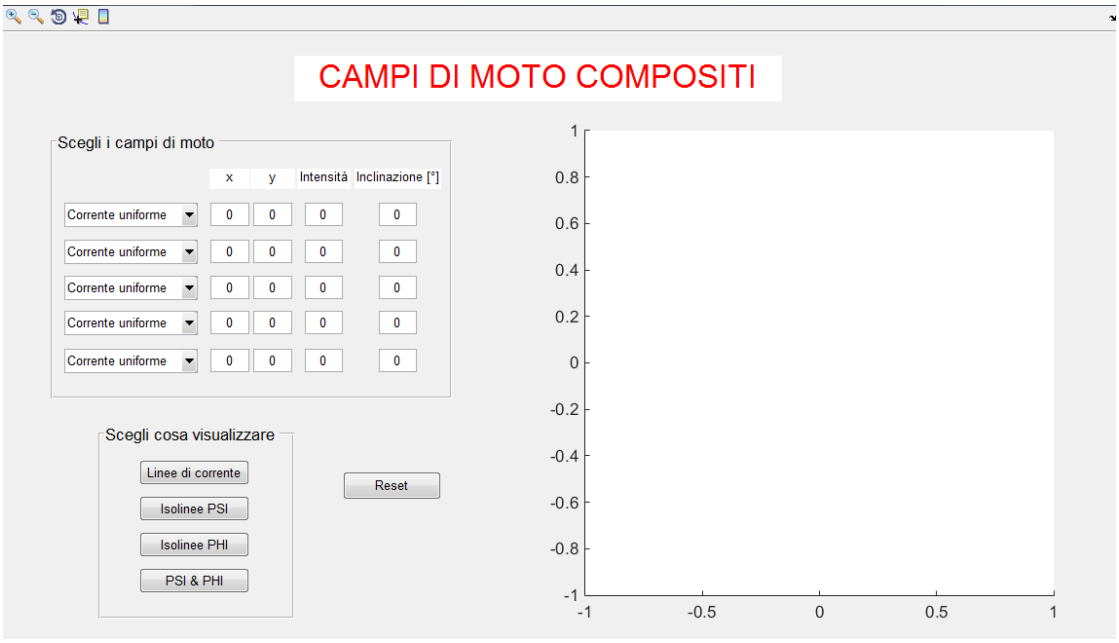

Figura 4.1: Interfaccia grafica

Nello static test principale, posizionato in centro e in alto, è scritto il titolo dell'interfaccia: 'Campi di moto compositi'. I restanti static text sono ciascuno in corrispondenza di una colonna di edit text ed indicano quale caratteristica del campo di moto bisogna inserire in questi ultimi: la posizione secondo x e y, l'intensità e l'inclinazione in gradi. Nel caso degli static text è bastato inserire il loro contenuto direttamente nell'interfaccia senza bisogno di scrivere nessun comando nello script.
I cinque pop-up menu sono identici l'uno all'altro ed ognuno di essi permette di scegliere quale campo di moto semplice inserire tra corrente uniforme, sorgente, pozzo, doppietta e vortice. La scelta effettuata viene recepita e salvata attraverso un ciclo switch presente nello script in corrispondenza di ciascun pop-up menu. Da qui si ottiene il campo di moto finale sommando i dati di tutti i pop-up menu, ognuno riferito ad un singolo campo semplice, ciò sempre grazie alla validità del principio di sovrapposizione degli effetti.
In corrispondenza di ogni pop-up menu sono collocati quattro edit text nei quali, come specificato dagli static text, si devono inserire i valori desiderati delle coordinate x e y, dell'intensità e dell'inclinazione in gradi del campo di moto desiderato. Il dominio stabilito all'interno del quale devono essere fissati i campi di moto è caratterizzato da $-1 < x < 1$ e $-1 < y < 1$. Inizialmente tutti i valori presenti negli edit text sono stati fissati pari a zero. Dopo aver inserito i valori desiderati

da tastiera, lo script li legge e li salva tramite la funzione 'get' e infine li utilizza nei rispettivi calcoli per ottenere corrispondentemente le componenti di velocità, la funzione potenziale e la funzione di corrente. Come nell'algoritmo anche in questo caso l'incidenza inserita in gradi viene trasformata in radianti.
Passando ai push botton inseriti, quattro di essi servono a decidere cosa visualizzare del campo di moto determinato e il quindi è per il reset. Cliccando il primo dei quattro, vengono rappresentate le linee di corrente tramite la funzione 'streamslice(handles.x, handles.y, handles.u, handles.v)' e subito sopra al grafico appare il rispettivo titolo 'Linee di corrente'. Con il secondo push botton si visualizzano le isolinee della funzione di corrente ψ grazie al comando 'contour(handles.x, handles.y, handles.ψ, 30)' e anche in questo caso compare il titolo corrispondente. Analogamente tramite il terzo push botton si possono rappresentare le linee equipotenziali ($\phi = costante$) con la rispettiva funzione 'contour(handles.x, handles.y, handles.ϕ, 30). Infine con il quarto push botton si visualizzano contemporaneamente le isolinee della funzione di corrente e della funzione potenziale . Questa possibilità di visualizzazione è stata inserita con lo scopo di dimostrare che in qualsiasi campo di moto le due funzioni sono mutualmente ortogonali. Tramite questi quattro push botton le diverse immagini del campo di moto sono rappresentate sugli assi, axes per il linguaggio MATLAB, le cui coordinate sia orizzontali, sia verticali sono comprese tra i valori -1 e 1.
Il quinto ed ultimo push botton presente è chiamato 'Reset' in quanto, dopo averlo cliccato, reimposta tutti gli edit text al valore iniziale pari a zero tramite la funzione 'set' e quindi tutte le corrispondenti variabili e grandezze calcolate nello script sono pari a zero. Inoltre pulisce il grafico facendolo tornare all'impostazione iniziale, grazie alla funzione 'cla(axes)'.
Infine sono presenti anche due pannelli, panel nel linguaggio MATLAB, per motivi di aspetto grafico. Infatti nel primo sono raggruppati tutti i componenti che permettono di determinare le caratteristiche del campo di moto e nel secondo sono presenti i quattro push botton attraverso cui si decide cosa visualizzare del campo di moto determinato.

4.2 Esempi

In questa sezione vengono fatti degli esempi di campi di moto compositi ottenibili con l'interfaccia e si riporta per ciascuno il risultato grafico. Come si osserverà i campi di moto compositi possono essere già noti, studiati in precedenza da qualcuno, o del tutto generali.

4.2.1 Prua di Fuhrmann

Tramite pop-up menu ed edit text si definisce un campo di moto composto da:

- una corrente uniforme $\mathbf{V}$ con intensità $W = -2$ ed inclinazione nulla, quindi si ottiene $\mathbf{V} = -u\mathbf{i}$,
- una sorgente posizionata nell'origine $(x, y) = (0, 0)$ e di intensità $Q = 2$,

e si ottiene ciò che si può vedere in figura 4.2.
Tramite la sovrapposizione degli effetti, in un punto qualsiasi del campo di moto, si ha:

$$\phi = \frac{Q}{2\pi} log\left(\frac{r}{r_0}\right) - ux = \frac{Q}{2\pi} log\left(\frac{x^2+y^2}{x_0^2+y_0^2}\right) - ux$$

$$\psi = \frac{Q}{2\pi}\theta - uy = \frac{Q}{2\pi} arctan\left(\frac{y}{x}\right)$$

$$\begin{cases} U = \frac{\partial\phi}{\partial x} = \frac{\partial\psi}{\partial y} = \frac{Q}{2\pi}\frac{x}{x^2+y^2} - u \\ V = \frac{\partial\phi}{\partial y} = -\frac{\partial\psi}{\partial x} = \frac{Q}{2\pi}\frac{y}{x^2+y^2} \end{cases}$$

Ora analizzando la funzione di corrente, si osserva che l'isolinea $\psi = 0$ ha una diramazione:

- una soluzione si ha con $y = 0$ e $\theta = 0$ e quindi un ramo coincide con il semiasse positivo delle x,
- l'altra soluzione si ottiene dall'equazione $y = \frac{Q}{2\pi u} arctan\left(\frac{y}{x}\right)$. Ne consegue che questo ramo coincide con la cosiddetta prua che divide il campo di moto in una zona interna ed una esterna e si estende fino all'infinito a valle tendendo a due asintoti orizzontali. La prua è detta di Fuhrmann perchè fu lui il primo a studiare questo campo di moto composito.

Il punto di diramazione è un punto di arresto, esso ha ascissa x_0 e ordinata $y = 0$. Ponendo quindi pari a zero le componenti di velocità si ottiene:

$$x_0 = \frac{Q}{2\pi u} \simeq 0.159$$

che è un punto molto vicino alla singolarità, ma non coincidente con essa, quindi in realtà il ramo coincidente con il semiasse positivo delle x inizia quando $x = x_0$. Infine si può calcolare la distanza $2h$ tra gli asintoti orizzontali a cui tende la prua ponendo $\psi = 0$, $\theta = \pi$ e $y = h$ e si ha:

$$h = \frac{Q}{2u} = 2$$

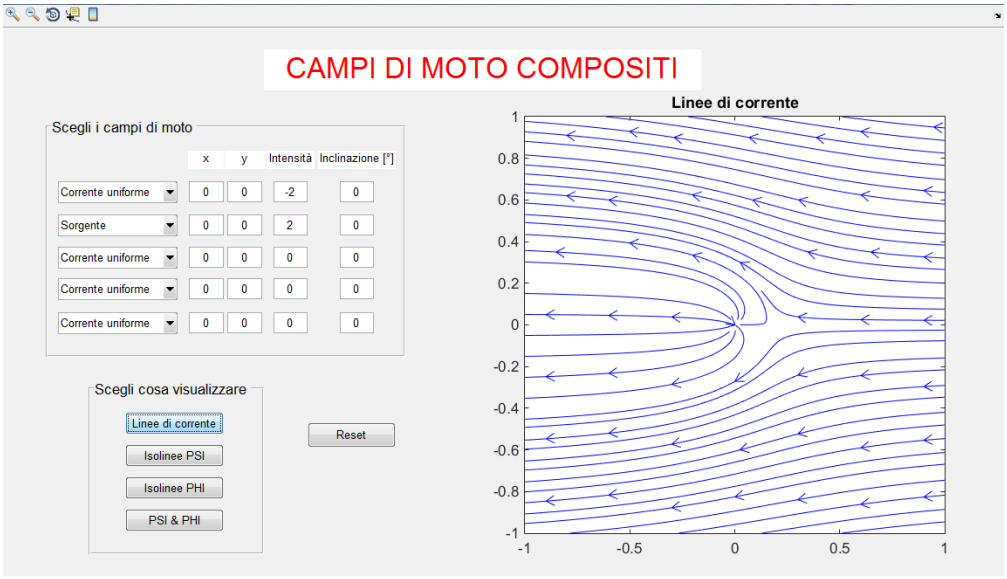

Figura 4.2: Linee di corrente della Prua di Fuhrmann

4.2.2 Ovale di Rankine

Definiamo un campo di moto composto da:

- una corrente uniforme $\mathbf{V}$ con intensità $W = -3$ ed inclinazione nulla, si ha quindi $\mathbf{V} = -u\mathbf{i}$,
- una sorgente posizionata in $x_0 = 0.5$ e $y_0 = 0$ e di intensità $Q = 4$,
- un pozzo posizionato in $x_0 = -0.5$ e $y_0 = 0$, cioè simmetricamente alla sorgente e con la stessa intensità di quest'ultima.

Dopo alcuni calcoli trigonometrici si ricava:

$$\psi = \frac{Q}{2\pi} arctan\left(\frac{2x_0 y}{x^2 + y^2 - x_0^2}\right) - uy$$

$$\begin{cases} U = \frac{\partial \psi}{\partial y} = \frac{Q}{2\pi} \frac{2x_0(x^2 - y^2 - x_0^2)}{(x^2+y^2+x_0^2)+4x_0^2 y^2} - u \\ V = -\frac{\partial \psi}{\partial x} = \frac{Q}{2\pi} \frac{4x_0 xy}{(x^2+y^2+x_0^2)+4x_0^2+y^2} \end{cases}$$

In questo caso analizzando la funzione di corrente, si osserva che l'isolinea $\psi = 0$ ha due diramazioni:

- per $x < -x_0$ e $x > x_0$ la soluzione coincide con l'asse x,

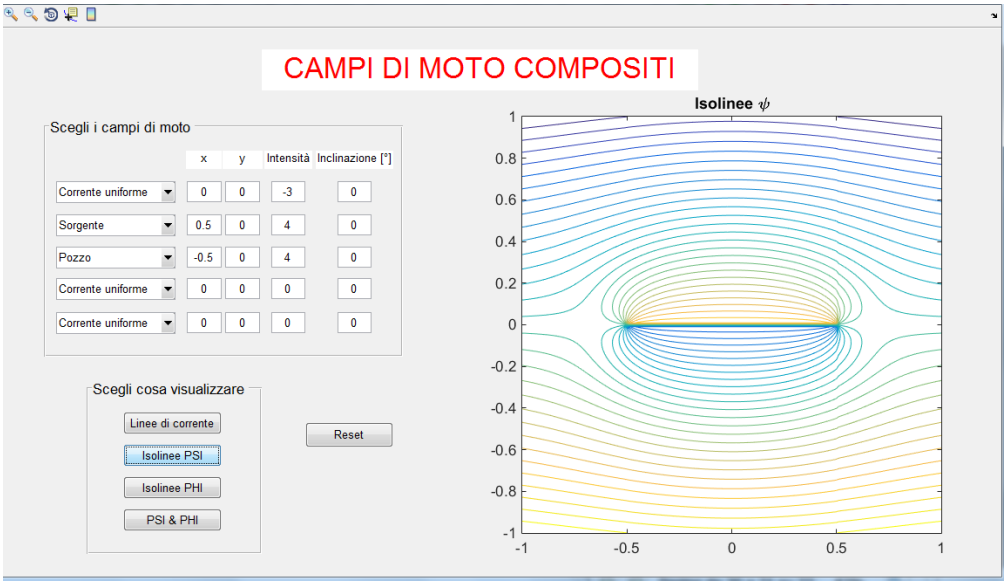

Figura 4.3: Funzione di corrente dell'Ovale di Rankine

- per valori $-x_0 < x < x_0$ bisogna risolvere l'equazione $y = \frac{Q}{2\pi u} arctan\left(\frac{2x_0 y}{x^2+y^2-x_0^2}\right)$ e si ottiene appunto la soluzione rappresentante l'ovale. Anche in questo caso l'ovale è detto di Rankine perche fu lui il primo a studiarlo.

I due punti di diramazione sono anche punti di arresto e sono posizionati in $(x, y) = (\pm x_1, 0)$, dove ricaviamo x_1 ponendo uguali a zero le componenti di velocità:

$$x_1 = \pm\sqrt{\frac{2Qx_0}{2\pi u} + x_0^2} \simeq \pm 0.6798$$

Quindi di nuovo i punti di diramazione non coincidono con le singolarità, ma sono in loro prossimità e pertanto la soluzione coincidente con l'asse x si ottiene per $x < -x_1$ e $x > x_1$. Infine si può calcolare l'altezza b dell'ovale tramite l'equazione ottenuta ponendo $\psi = 0$, $x = 0$ e $y = b$:

$$b = \frac{Q}{2\pi u} arctan \frac{2x_0 b}{b^2 - x_0^2}$$

4.2.3 Un caso generale

Infine si considera un caso del tutto generale per dimostrare che l'interfaccia grafica funziona con qualsiasi combinazione e non solo in caso di campi di moto noti. Componiamo il campo di moto come segue:

- una corrente uniforme di intensità $W = 1$ ed inclinazione $\alpha = 30°$,

- un pozzo posizionato in $x = 0.5$ e $y = 0.6$ e con intensità $Q = 2$,
- un vortice posizionato in $x = 0.7$ e $y = -0.3$ e con intesità $\Gamma = 3$,
- una doppietta posizionata in $x = -0.5$ e $y = 0.4$, con intensità $\mu = 2$ ed inclinazione $\alpha = 90°$,
- una sorgente posizionata in $x = -0.3$ e $y = -0.6$ e con intensità $Q = 4$,

e si ottengo le linee di corrente mostrare in figura 4.4.

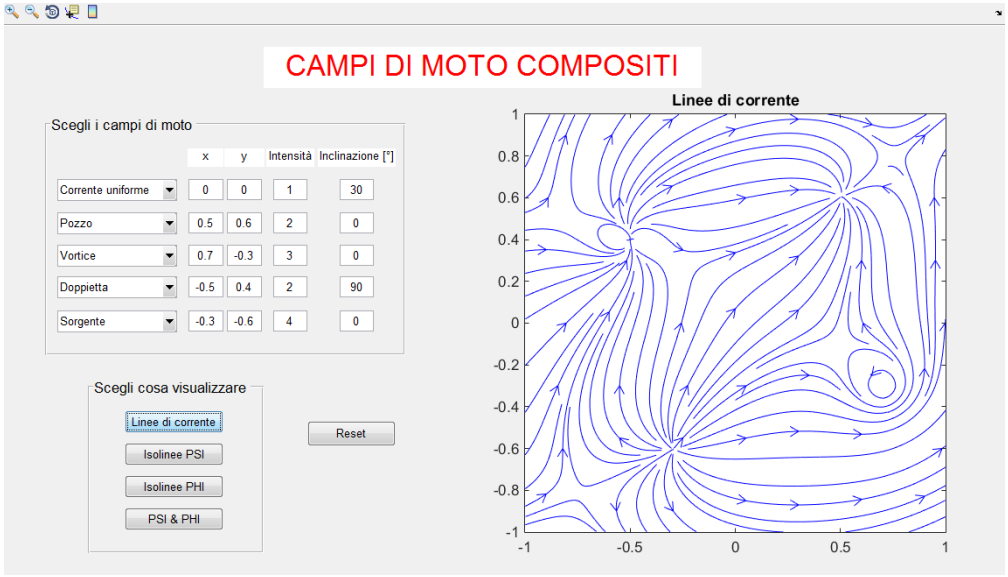

Figura 4.4: Linee di corrente di un campo di moto generale

In questo esempio sono stati inseriti tutti i diversi campi di moto semplici studiati, ma ciò non è obbligatorio. Infatti si possono inserire anche più campi semplici di uno stesso tipo ed è proprio in questo modo che si riesce ad ottenere la simulazione della corrente attorno a corpi chiusi.

Conclusioni

L'obiettivo del lavoro svolto era di realizzare un'interfaccia grafica personalizzata ed interattiva, tramite una GUI di MATLAB, in cui fosse possibile inserire più campi di moto semplici per poi ottenere la visualizzazione del campo di moto composito così determinato, grazie alla validità del principio di sovrapposizione degli effetti.
Si è deciso di dare la possibilità all'utente di inserire un massimo di cinque campi semplici e per ciascuno di essi di determinare i valori delle coordinate della posizione all'interno del dominio stabilito, dell'intensità e dell'inclinazione. In questo modo si possono visualizzare contemporaneamente tutti i diversi campi di moto semplici illustrati nel capitolo 2, oppure si può decidere di sovrapporre più campi semplici di uno stesso tipo. Infine tramite i diversi push botton presenti nell'interfaccia si può selezionare ciò che si preferisce visualizzare tra linee di corrente, funzione di corrente, funzione potenziale e la sovrapposizione di queste ultime due, le quali sono state definite nel capitolo 1.
L'obiettivo dell'elaborato è stato raggiunto; il prossimo passo di ottimizzazione dell'interfaccia grafica sarà quello di non porre un limite massimo di campi semplici che è possibile inserire e di aggiungere ulteriori push botton in modo da poter visualizzare altre grandezze relative al campo di moto composito, come ad asempio il coefficiente adimensionale di pressione.

Appendice

In questa appendice viene inserito integralmente lo script dell'interfaccia grafica illustrata nel Capitolo 4.

```
function varargout = INTERF(varargin)
% INTERF MATLAB code for INTERF.fig
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @INTERF_OpeningFcn, ...
'gui_OutputFcn',  @INTERF_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before INTERF is made visible.
function INTERF_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see76 OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to INTERF (see VARARGIN)
[x,y] = meshgrid([-1:0.01:-0.01, +0.01:.01:1],[-1:0.01:-0.01, +0.01:.01:1]);
handles.x = x;
handles.y = y;
axes(handles.axes1)
axis([-1 1 -1 1])
```

```
35
   %Primo popup menu
37 handles.X1 = str2double(get(handles.x1,'String'));
   handles.Y1 = str2double(get(handles.y1,'String'));
39 handles.Q1 = str2double(get(handles.q1,'String'));
   handles.ang1 = str2double(get(handles.alpha1,'String'));
41 x1 = handles.X1;
   y1 = handles.Y1;
43 [x,y] = meshgrid([-1:0.01:x1-0.01, x1+0.01:.01:1],...
   ...[-1:0.01:y1-0.01, y1+0.01:.01:1]);
45 handles.x = x;
   handles.y = y;
47 q1 = handles.Q1;
   alpha1 = handles.ang1*pi/180;
49 r1 = sqrt((x-x1).^2+(y-y1).^2);
   theta1 = atan2(y-y1,x-x1);
51 handles.r1 = r1;
   handles.theta1 = theta1;
53 %Corrente uniforme 1
   uc1 = q1*cos(alpha1);
55 vc1 = q1*sin(alpha1);
   Uc1 = uc1*ones(length(x),length(x));
57 Vc1 = vc1*ones(length(x),length(x));
   psicor1 = -Vc1.*x+Uc1.*y;
59 phicor1 = Uc1.*x+Vc1.*y;
   handles.psic1 = psicor1;
61 handles.phic1 = phicor1;
   handles.uc1 = Uc1;
63 handles.vc1 = Vc1;
   %Sorgente 1
65 psisorg1 = q1./(2*pi).*theta1;
   phisorg1 = q1./(2*pi).*log(r1);
67 usorg1 = cos(theta1).*q1./(2*pi*r1);
   vsorg1 = sin(theta1).*q1./(2*pi*r1);
69 handles.psis1 = psisorg1;
   handles.phis1 = phisorg1;
71 handles.us1 = usorg1;
   handles.vs1 = vsorg1;
73 %Pozzo 1
   psipoz1 = -q1./(2*pi).*theta1;
75 phipoz1 = -q1./(2*pi).*log(r1);
   upoz1 = -cos(theta1).*q1./(2*pi*r1);
77 vpoz1 = -sin(theta1).*q1./(2*pi*r1);
   handles.psip1 = psipoz1;
79 handles.phip1 = phipoz1;
   handles.up1 = upoz1;
81 handles.vp1 = vpoz1;
   %Doppietta 1
83 psidop1 = sin(theta1-alpha1).*q1./(2*pi*r1);
```

```
phidop1 = -cos(theta1-alpha1).*q1./(2*pi*r1);
vdop1 = q1./(2*pi*r1).*cos(theta1-alpha1).*sin(theta1)+q1./...
...(2*pi*r1).*sin(theta1-alpha1).*cos(theta1);
udop1 = q1./(2*pi*r1).*cos(theta1-alpha1).*cos(theta1)-q1./...
...(2*pi*r1).*sin(theta1-alpha1).*sin(theta1);
handles.psid1 = psidop1;
handles.phid1 = phidop1;
handles.ud1 = udop1;
handles.vd1 = vdop1;
%Vortice 1
psivor1 = -q1./(2*pi).*log(r1);
phivor1 = q1./(2*pi).*theta1;
uvor1 = -q1./(2*pi*r1).*sin(theta1);
vvor1 = q1./(2*pi*r1).*cos(theta1);
handles.psiv1 = psivor1;
handles.phiv1 = phivor1;
handles.uv1 = uvor1;
handles.vv1 = vvor1;
% Set the current data value.
handles.psi1 = handles.psic1;
handles.phi1 = handles.phic1;
handles.u1 = handles.uc1;
handles.v1 = handles.vc1;

%Secondo popup menu
handles.X2 = str2double(get(handles.x2,'String'));
handles.Y2 = str2double(get(handles.y2,'String'));
handles.Q2 = str2double(get(handles.q2,'String'));
handles.ang2 = str2double(get(handles.alpha2,'String'));
x2 = handles.X2;
y2 = handles.Y2;
[x,y] = meshgrid([-1:0.01:x2-0.01, x2+0.01:.01:1],...
...[-1:0.01:y2-0.01, y2+0.01:.01:1]);
handles.x = x;
handles.y = y;
q2 = handles.Q2;
alpha2 = handles.ang2*pi/180;
r2 = sqrt((x-x2).^2+(y-y2).^2);
theta2 = atan2(y-y2,x-x2);
handles.r2 = r2;
handles.theta2 = theta2;
%Corrente uniforme 2
uc2 = q2*cos(alpha2);
vc2 = q2*sin(alpha2);
Uc2 = uc2*ones(length(x),length(x));
Vc2 = vc2*ones(length(x),length(x));
psicor2 = -Vc2.*x+Uc2.*y;
phicor2 = Uc2.*x+Vc2.*y;
handles.psic2 = psicor2;
```

```
133 handles.phic2 = phicor2;
    handles.uc2 = Uc2;
135 handles.vc2 = Vc2;
    %Sorgente 2
137 psisorg2 = q2./(2*pi).*theta2;
    phisorg2 = q2./(2*pi).*log(r2);
139 usorg2 = cos(theta2).*q2./(2*pi*r2);
    vsorg2 = sin(theta2).*q2./(2*pi*r2);
141 handles.psis2 = psisorg2;
    handles.phis2 = phisorg2;
143 handles.us2 = usorg2;
    handles.vs2 = vsorg2;
145 %Pozzo 2
    psipoz2 = -q2./(2*pi).*theta2;
147 phipoz2 = -q2./(2*pi).*log(r2);
    upoz2 = -cos(theta2).*q2./(2*pi*r2);
149 vpoz2 = -sin(theta2).*q2./(2*pi*r2);
    handles.psip2 = psipoz2;
151 handles.phip2 = phipoz2;
    handles.up2 = upoz2;
153 handles.vp2 = vpoz2;
    %Doppietta 2
155 psidop2 = sin(theta2-alpha2).*q2./(2*pi*r2);
    phidop2 = -cos(theta2-alpha2).*q2./(2*pi*r2);
157 vdop2 = q2./(2*pi*r2).*cos(theta2-alpha2).*sin(theta2)+q2./...
    ...(2*pi*r2).*sin(theta2-alpha2).*cos(theta2);
159 udop2 = q2./(2*pi*r2).*cos(theta2-alpha2).*cos(theta2)-q2./...
    ...(2*pi*r2).*sin(theta2-alpha2).*sin(theta2);
161 handles.psid2 = psidop2;
    handles.phid2 = phidop2;
163 handles.ud2 = udop2;
    handles.vd2 = vdop2;
165 %Vortice 2
    psivor2 = -q2./(2*pi).*log(r2);
167 phivor2 = q2./(2*pi).*theta2;
    uvor2 = -q2./(2*pi*r2).*sin(theta2);
169 vvor2 = q2./(2*pi*r2).*cos(theta2);
    handles.psiv2 = psivor2;
171 handles.phiv2 = phivor2;
    handles.uv2 = uvor2;
173 handles.vv2 = vvor2;
    % Set the current data value.
175 handles.psi2 = handles.psis2;
    handles.phi2 = handles.phis2;
177 handles.u2 = handles.us2;
    handles.v2 = handles.vs2;
179
    %Terzo popup menu
181 handles.X3 = str2double(get(handles.x3,'String'));
```

```
handles.Y3 = str2double(get(handles.y3,'String'));
handles.Q3 = str2double(get(handles.q3,'String'));
handles.ang3 = str2double(get(handles.alpha3,'String'));
x3 = handles.X3;
y3 = handles.Y3;
[x,y] = meshgrid([-1:0.01:x3-0.01, x3+0.01:.01:1],...
...[-1:0.01:y3-0.01, y3+0.01:.01:1]);
handles.x = x;
handles.y = y;
q3 = handles.Q3;
alpha3 = handles.ang3*pi/180;
r3 = sqrt((x-x3).^2+(y-y3).^2);
theta3 = atan2(y-y3,x-x3);
handles.r3 = r3;
handles.theta3 = theta3;
%Corrente uniforme3
uc3 = q3*cos(alpha3);
vc3 = q3*sin(alpha3);
Uc3 = uc3*ones(length(x),length(x));
Vc3 = vc3*ones(length(x),length(x));
psicor3 = -Vc3.*x+Uc3.*y;
phicor3 = Uc3.*x+Vc3.*y;
handles.psic3 = psicor3;
handles.phic3 = phicor3;
handles.uc3 = Uc3;
handles.vc3 = Vc3;
%Sorgente 3
psisorg3 = q3./(2*pi).*theta3;
phisorg3 = q3./(2*pi).*log(r3);
usorg3 = cos(theta3).*q3./(2*pi*r3);
vsorg3 = sin(theta3).*q3./(2*pi*r3);
handles.psis3 = psisorg3;
handles.phis3 = phisorg3;
handles.us3 = usorg3;
handles.vs3 = vsorg3;
%Pozzo 3
psipoz3 = -q3./(2*pi).*theta3;
phipoz3 = -q3./(2*pi).*log(r3);
upoz3 = -cos(theta3).*q3./(2*pi*r3);
vpoz3 = -sin(theta3).*q3./(2*pi*r3);
handles.psip3 = psipoz3;
handles.phip3 = phipoz3;
handles.up3 = upoz3;
handles.vp3 = vpoz3;
%Doppietta 3
psidop3 = sin(theta3-alpha3).*q3./(2*pi*r3);
phidop3 = -cos(theta3-alpha3).*q3./(2*pi*r3);
vdop3 = q3./(2*pi*r3).*cos(theta3-alpha3).*sin(theta3)+q3./...
...(2*pi*r3).*sin(theta3-alpha3).*cos(theta3);
```

```
231 udop3 = q3./(2*pi*r3).*cos(theta3-alpha3).*cos(theta3)-q3./...
    ...(2*pi*r3).*sin(theta3-alpha3).*sin(theta3);
233 handles.psid3 = psidop3;
    handles.phid3 = phidop3;
235 handles.ud3 = udop3;
    handles.vd3 = vdop3;
237 %Vortice 3
    psivor3 = -q3./(2*pi).*log(r3);
239 phivor3 = q3./(2*pi).*theta3;
    uvor3 = -q3./(2*pi*r3).*sin(theta3);
241 vvor3 = q3./(2*pi*r3).*cos(theta3);
    handles.psiv3 = psivor3;
243 handles.phiv3 = phivor3;
    handles.uv3 = uvor3;
245 handles.vv3 = vvor3;
    % Set the current data value.
247 handles.psi3 = handles.psip3;
    handles.phi3 = handles.phip3;
249 handles.u3 = handles.up3;
    handles.v3 = handles.vp3;
251
    %Quarto popup menu
253 handles.X4 = str2double(get(handles.x4,'String'));
    handles.Y4 = str2double(get(handles.y4,'String'));
255 handles.Q4 = str2double(get(handles.q4,'String'));
    handles.ang4 = str2double(get(handles.alpha4,'String'));
257 x4 = handles.X4;
    y4= handles.Y4;
259 [x,y] = meshgrid([-1:0.01:x4-0.01, x4+0.01:.01:1],...
    ...[-1:0.01:y4-0.01, y4+0.01:.01:1]);
261 handles.x = x;
    handles.y = y;
263 q4 = handles.Q4;
    alpha4 = handles.ang4*pi/180;
265 r4 = sqrt((x-x4).^2+(y-y4).^2);
    theta4 = atan2(y-y4,x-x4);
267 handles.r4 = r4;
    handles.theta4 = theta4;
269 %Corrente uniforme 4
    uc4 = q4*cos(alpha4);
271 vc4 = q4*sin(alpha4);
    Uc4 = uc4*ones(length(x),length(x));
273 Vc4 = vc4*ones(length(x),length(x));
    psicor4 = -Vc4.*x+Uc4.*y;
275 phicor4 = Uc4.*x+Vc4.*y;
    handles.psic4 = psicor4;
277 handles.phic4 = phicor4;
    handles.uc4 = Uc4;
279 handles.vc4 = Vc4;
```

```
%Sorgente 4
psisorg4 = q4./(2*pi).*theta4;
phisorg4 = q4./(2*pi).*log(r4);
usorg4 = cos(theta4).*q4./(2*pi*r4);
vsorg4 = sin(theta4).*q4./(2*pi*r4);
handles.psis4 = psisorg4;
handles.phis4 = phisorg4;
handles.us4 = usorg4;
handles.vs4 = vsorg4;
%Pozzo 4
psipoz4 = -q4./(2*pi).*theta4;
phipoz4 = -q4./(2*pi).*log(r4);
upoz4 = -cos(theta4).*(-q4)./(2*pi*r4);
vpoz4 = -sin(theta4).*(-q4)./(2*pi*r4);
handles.psip4 = psipoz4;
handles.phip4 = phipoz4;
handles.up4 = upoz4;
handles.vp4 = vpoz4;
%Doppietta 4
psidop4 = sin(theta4-alpha4).*q4./(2*pi*r4);
phidop4 = -cos(theta4-alpha4).*q4./(2*pi*r4);
vdop4 = q4./(2*pi*r4).*cos(theta4-alpha4).*sin(theta4)+q4./...
...(2*pi*r4).*sin(theta4-alpha4).*cos(theta4);
udop4 = q4./(2*pi*r4).*cos(theta4-alpha4).*cos(theta4)-q4./...
...(2*pi*r4).*sin(theta4-alpha4).*sin(theta4);
handles.psid4 = psidop4;
handles.phid4 = phidop4;
handles.ud4 = udop4;
handles.vd4 = vdop4;
%Vortice 4
psivor4 = -q4./(2*pi).*log(r4);
phivor4 = q4./(2*pi).*theta4;
uvor4 = -q4./(2*pi*r4).*sin(theta4);
vvor4 = q4./(2*pi*r4).*cos(theta4);
handles.psiv4 = psivor4;
handles.phiv4 = phivor4;
handles.uv4 = uvor4;
handles.vv4 = vvor4;
% Set the current data value.
handles.psi4 = handles.psid4;
handles.phi4 = handles.phid4;
handles.u4 = handles.ud4;
handles.v4 = handles.vd4;

%Quinto popup menu
handles.X5 = str2double(get(handles.x5,'String'));
handles.Y5 = str2double(get(handles.y5,'String'));
handles.Q5 = str2double(get(handles.q5,'String'));
handles.ang5 = str2double(get(handles.alpha5,'String'));
```

```
x5 = handles.X5;
y5= handles.Y5;
[x,y] = meshgrid([-1:0.01:x5-0.01, x5+0.01:.01:1],...
...[-1:0.01:y5-0.01, y5+0.01:.01:1]);
handles.x = x;
handles.y = y;
q5 = handles.Q5;
alpha5 = handles.ang5*pi/180;
r5 = sqrt((x-x5).^2+(y-y5).^2);
theta5 = atan2(y-y5,x-x5);
handles.r4 = r5;
handles.theta4 = theta5;
%Corrente uniforme 5
uc5 = q5*cos(alpha5);
vc5 = q5*sin(alpha5);
Uc5 = uc5*ones(length(x),length(x));
Vc5 = vc5*ones(length(x),length(x));
psicor5 = -Vc5.*x+Uc5.*y;
phicor5 = Uc5.*x+Vc5.*y;
handles.psic5 = psicor5;
handles.phic5 = phicor5;
handles.uc5 = Uc5;
handles.vc5 = Vc5;
%Sorgente 5
psisorg5 = q5./(2*pi).*theta5;
phisorg5 = q5./(2*pi).*log(r5);
usorg5 = cos(theta5).*q5./(2*pi*r5);
vsorg5 = sin(theta5).*q5./(2*pi*r5);
handles.psis5 = psisorg5;
handles.phis5 = phisorg5;
handles.us5 = usorg5;
handles.vs5 = vsorg5;
%Pozzo 5
psipoz5 = -q5./(2*pi).*theta5;
phipoz5 = -q5./(2*pi).*log(r5);
upoz5 = -cos(theta5).*(-q5)./(2*pi*r5);
vpoz5 = -sin(theta5).*(-q5)./(2*pi*r5);
handles.psip5 = psipoz5;
handles.phip5 = phipoz5;
handles.up5 = upoz5;
handles.vp5 = vpoz5;
%Doppietta 5
psidop5 = sin(theta5-alpha5).*q5./(2*pi*r5);
phidop5 = -cos(theta5-alpha5).*q5./(2*pi*r5);
vdop5 = q5./(2*pi*r5).*cos(theta5-alpha5).*sin(theta5)+q5./...
...(2*pi*r5).*sin(theta5-alpha5).*cos(theta5);
udop5 = q5./(2*pi*r5).*cos(theta5-alpha5).*cos(theta5)-q5./...
...(2*pi*r5).*sin(theta5-alpha5).*sin(theta5);
handles.psid5 = psidop5;
```

```
handles.phid5 = phidop5;
handles.ud5 = udop5;
handles.vd5 = vdop5;
%Vortice 5
psivor5 = -q5./(2*pi).*log(r5);
phivor5 = q5./(2*pi).*theta5;
uvor5 = -q5./(2*pi*r5).*sin(theta5);
vvor5 = q5./(2*pi*r5).*cos(theta5);
handles.psiv5 = psivor5;
handles.phiv5 = phivor5;
handles.uv5 = uvor5;
handles.vv5 = vvor5;
% Set the current data value.
handles.psi5 = handles.psiv5;
handles.phi5 = handles.phiv5;
handles.u5 = handles.uv5;
handles.v5 = handles.vv5;

%Set the final current data
handles.psi = handles.psi1 + handles.psi2 + handles.psi3 +...
...handles.psi4 + handles.psi5;
handles.phi = handles.phi1 + handles.phi2 + handles.phi3 +...
...handles.phi4 + handles.phi5;
handles.u = handles.u1 + handles.u2 + handles.u3 +...
...handles.u4 + handles.u5;
handles.v = handles.v1 + handles.v2 + handles.v3 +...
...handles.v4 + handles.v5;
% Choose default command line output for INTERF
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes INTERF wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = INTERF_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in reset.
function reset_Callback(hObject, eventdata, handles)
% hObject    handle to reset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
a = 0;
cla(handles.axes1)
box on
axis([-1 1 -1 1])
title('␣')
set(handles.x1,'String',a)
set(handles.y1,'String',a)
set(handles.q1,'String',a)
set(handles.alpha1,'String',a)
set(handles.x2,'String',a)
set(handles.y2,'String',a)
set(handles.q2,'String',a)
set(handles.alpha2,'String',a)
set(handles.x3,'String',a)
set(handles.y3,'String',a)
set(handles.q3,'String',a)
set(handles.alpha3,'String',a)
set(handles.x4,'String',a)
set(handles.y4,'String',a)
set(handles.q4,'String',a)
set(handles.alpha4,'String',a)
set(handles.x5,'String',a)
set(handles.y5,'String',a)
set(handles.q5,'String',a)
set(handles.alpha5,'String',a)
handles.uc = 0;
handles.vc = 0;
handles.psic = 0;
handles.phic = 0;
handles.u1 = 0;
handles.v1 = 0;
handles.psi1 = 0;
handles.phi1 = 0;
handles.u2 = 0;
handles.v2 = 0;
handles.psi2 = 0;
handles.phi2 = 0;
handles.u3 = 0;
handles.v3 = 0;
handles.psi3 = 0;
handles.phi3 = 0;
handles.u4 = 0;
handles.v4 = 0;
handles.psi4 = 0;
handles.phi4 = 0;
handles.u5 = 0;
handles.v5 = 0;
handles.psi5 = 0;
```

```
handles.phi5 = 0;
% Save the handles structure.
guidata(hObject,handles)
% Choose default command line output
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% --- Executes on button press in velocita.
function velocita_Callback(hObject, eventdata, handles)
%Visualizzo linee di corrente
cla(handles.axes1)
box on
axes(handles.axes1)
streamslice(handles.x,handles.y,handles.u,handles.v);
axis([-1 1 -1 1])
title('Linee di corrente')

% --- Executes on button press in psi.
function psi_Callback(hObject, eventdata, handles)
%Visualizzo funzione psi
cla(handles.axes1)
box on
axes(handles.axes1)
contour(handles.x,handles.y,handles.psi,30);
axis([-1 1 -1 1])
title('Isolinee \psi')


% --- Executes on button press in phi.
function phi_Callback(hObject, eventdata, handles)
%Visualizzo funzione phi
cla(handles.axes1)
box on
axes(handles.axes1)
contour(handles.x,handles.y,handles.phi,30);
axis([-1 1 -1 1])
title('Isolinee \phi')

% --- Executes on button press in psiphi.
function psiphi_Callback(hObject, eventdata, handles)
%Visualizzo psi e phi contemporaneamente
cla(handles.axes1)
box on
axes(handles.axes1)
contour(handles.x,handles.y,handles.psi,30);
hold on
contour(handles.x,handles.y,handles.phi,30);
axis([-1 1 -1 1])
```

```
525 title('Isolinee␣\psi␣e␣\phi')

527 % --- Executes on selection change in popupmenu1.
    function popupmenu1_Callback(hObject, eventdata, handles)
529 str = get(hObject, 'String');
    val1 = get(hObject,'Value');
531 % Set current data to the selected data set.
    switch str(val1,:)
533 case 'Corrente␣uniforme'
    handles.psi1 = handles.psic1;
535 handles.phi1 = handles.phic1;
    handles.u1 = handles.uc1;
537 handles.v1 = handles.vc1;
    case 'Sorgente␣␣␣␣␣␣␣␣␣'
539 handles.psi1 = handles.psis1;
    handles.phi1 = handles.phis1;
541 handles.u1 = handles.us1;
    handles.v1 = handles.vs1;
543 case 'Pozzo␣␣␣␣␣␣␣␣␣␣␣␣'
    handles.psi1 = handles.psip1;
545 handles.phi1 = handles.phip1;
    handles.u1 = handles.up1;
547 handles.v1 = handles.vp1;
    case 'Doppietta␣␣␣␣␣␣␣␣'
549 handles.psi1 = handles.psid1;
    handles.phi1 = handles.phid1;
551 handles.u1 = handles.ud1;
    handles.v1 = handles.vd1;
553 case 'Vortice␣␣␣␣␣␣␣␣␣␣'
    handles.psi1 = handles.psiv1;
555 handles.phi1 = handles.phiv1;
    handles.u1 = handles.uv1;
557 handles.v1 = handles.vv1;
    end
559 % Save the handles structure.
    guidata(hObject,handles)
561
    % --- Executes during object creation, after setting all properties.
563 function popupmenu1_CreateFcn(hObject, eventdata, handles)

565 if ispc && isequal(get(hObject,'BackgroundColor'),...
    ...get(0,'defaultUicontrolBackgroundColor'))
567 set(hObject,'BackgroundColor','white');
    end
569
    % --- Executes on selection change in popupmenu2.
571 function popupmenu2_Callback(hObject, eventdata, handles)
    str = get(hObject, 'String');
573 val2 = get(hObject,'Value');
```

```
    % Set current data to the selected data set.
575 switch str(val2,:)
    case 'Corrente␣uniforme'
577 handles.psi2 = handles.psic2;
    handles.phi2 = handles.phic2;
579 handles.u2 = handles.uc2;
    handles.v2 = handles.vc2;
581 case 'Sorgente␣␣␣␣␣␣␣␣␣'
    handles.psi2 = handles.psis2;
583 handles.phi2 = handles.phis2;
    handles.u2 = handles.us2;
585 handles.v2 = handles.vs2;
    case 'Pozzo␣␣␣␣␣␣␣␣␣␣␣␣'
587 handles.psi2 = handles.psip2;
    handles.phi2 = handles.phip2;
589 handles.u2 = handles.up2;
    handles.v2 = handles.vp2;
591 case 'Doppietta␣␣␣␣␣␣␣␣'
    handles.psi2 = handles.psid2;
593 handles.phi2 = handles.phid2;
    handles.u2 = handles.ud2;
595 handles.v2 = handles.vd2;
    case 'Vortice␣␣␣␣␣␣␣␣␣␣'
597 handles.psi2 = handles.psiv2;
    handles.phi2 = handles.phiv2;
599 handles.u2 = handles.uv2;
    handles.v2 = handles.vv2;
601 end
    % Save the handles structure.
603 guidata(hObject,handles)

605 % --- Executes during object creation, after setting all properties.
    function popupmenu2_CreateFcn(hObject, eventdata, handles)
607
    if ispc && isequal(get(hObject,'BackgroundColor'),...
609 ...get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
611 end

613 % --- Executes on selection change in popupmenu3.
    function popupmenu3_Callback(hObject, eventdata, handles)
615 str = get(hObject, 'String');
    val3 = get(hObject,'Value');
617 % Set current data to the selected data set.
    switch str(val3,:)
619 case 'Corrente␣uniforme'
    handles.psi3 = handles.psic3;
621 handles.phi3 = handles.phic3;
    handles.u3 = handles.uc3;
```

```
623 handles.v3 = handles.vc3;
    case 'Sorgente␣␣␣␣␣␣␣␣␣'
625 handles.psi3 = handles.psis3;
    handles.phi3 = handles.phis3;
627 handles.u3 = handles.us3;
    handles.v3 = handles.vs3;
629 case 'Pozzo␣␣␣␣␣␣␣␣␣␣␣␣'
    handles.psi3 = handles.psip3;
631 handles.phi3 = handles.phip3;
    handles.u3 = handles.up3;
633 handles.v3 = handles.vp3;
    case 'Doppietta␣␣␣␣␣␣␣␣'
635 handles.psi3 = handles.psid3;
    handles.phi3 = handles.phid3;
637 handles.u3 = handles.ud3;
    handles.v3 = handles.vd3;
639 case 'Vortice␣␣␣␣␣␣␣␣␣␣'
    handles.psi3 = handles.psiv3;
641 handles.phi3 = handles.phiv3;
    handles.u3 = handles.uv3;
643 handles.v3 = handles.vv3;
    end
645 % Save the handles structure.
    guidata(hObject,handles)
647
    % --- Executes during object creation, after setting all properties.
649 function popupmenu3_CreateFcn(hObject, eventdata, handles)

651 if ispc && isequal(get(hObject,'BackgroundColor'),...
    ...get(0,'defaultUicontrolBackgroundColor'))
653 set(hObject,'BackgroundColor','white');
    end
655
    % --- Executes on selection change in popupmenu4.
657 function popupmenu4_Callback(hObject, eventdata, handles)
    str = get(hObject, 'String');
659 val4 = get(hObject,'Value');
    % Set current data to the selected data set.
661 switch str(val4,:)
    case 'Corrente␣uniforme'
663 handles.psi4 = handles.psic4;
    handles.phi4 = handles.phic4;
665 handles.u4 = handles.uc4;
    handles.v4 = handles.vc4;
667 case 'Sorgente␣␣␣␣␣␣␣␣␣'
    handles.psi4 = handles.psis4;
669 handles.phi4 = handles.phis4;
    handles.u4 = handles.us4;
671 handles.v4 = handles.vs4;
```

```
    case 'Pozzo␣␣␣␣␣␣␣␣␣␣␣␣'
673 handles.psi4 = handles.psip4;
    handles.phi4 = handles.phip4;
675 handles.u4 = handles.up4;
    handles.v4 = handles.vp4;
677 case 'Doppietta␣␣␣␣␣␣␣␣'
    handles.psi4 = handles.psid4;
679 handles.phi4 = handles.phid4;
    handles.u4 = handles.ud4;
681 handles.v4 = handles.vd4;
    case 'Vortice␣␣␣␣␣␣␣␣␣␣'
683 handles.psi4 = handles.psiv4;
    handles.phi4 = handles.phiv4;
685 handles.u4 = handles.uv4;
    handles.v4 = handles.vv4;
687 end
    % Save the handles structure.
689 guidata(hObject,handles)

691 % --- Executes during object creation, after setting all properties.
    function popupmenu4_CreateFcn(hObject, eventdata, handles)
693
    if ispc && isequal(get(hObject,'BackgroundColor'),...
695 ...get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
697 end

699 % --- Executes on selection change in popupmenu5.
    function popupmenu5_Callback(hObject, eventdata, handles)
701 str = get(hObject, 'String');
    val5 = get(hObject,'Value');
703 % Set current data to the selected data set.
    switch str(val5,:)
705 case 'Corrente␣uniforme'
    handles.psi5 = handles.psic5;
707 handles.phi5 = handles.phic5;
    handles.u5 = handles.uc5;
709 handles.v5 = handles.vc5;
    case 'Sorgente␣␣␣␣␣␣␣␣␣'
711 handles.psi5 = handles.psis5;
    handles.phi5 = handles.phis5;
713 handles.u5 = handles.us5;
    handles.v5 = handles.vs5;
715 case 'Pozzo␣␣␣␣␣␣␣␣␣␣␣␣'
    handles.psi5 = handles.psip5;
717 handles.phi5 = handles.phip5;
    handles.u5 = handles.up5;
719 handles.v5 = handles.vp5;
    case 'Doppietta␣␣␣␣␣␣␣␣'
```

```
721 handles.psi5 = handles.psid5;
    handles.phi5 = handles.phid5;
723 handles.u5 = handles.ud5;
    handles.v5 = handles.vd5;
725 case 'Vortice          '
    handles.psi5 = handles.psiv5;
727 handles.phi5 = handles.phiv5;
    handles.u5 = handles.uv5;
729 handles.v5 = handles.vv5;
    end
731 % Save the handles structure.
    guidata(hObject,handles)
733
    % --- Executes during object creation, after setting all properties.
735 function popupmenu5_CreateFcn(hObject, eventdata, handles)

737 if ispc && isequal(get(hObject,'BackgroundColor'),...
    ...get(0,'defaultUicontrolBackgroundColor'))
739 set(hObject,'BackgroundColor','white');
    end
741
    function x1_Callback(hObject, eventdata, handles)
743 handles.X1 = str2double(get(handles.x1,'String'));
    handles.Y1 = str2double(get(handles.y1,'String'));
745 handles.Q1 = str2double(get(handles.q1,'String'));
    handles.ang1 = str2double(get(handles.alpha1,'String'));
747 x1 = handles.X1;
    y1 = handles.Y1;
749 [x,y] = meshgrid([-1:0.01:x1-0.01, x1+0.01:.01:1],...
    ...[-1:0.01:y1-0.01, y1+0.01:.01:1]);
751 handles.x = x;
    handles.y = y;
753 q1 = handles.Q1;
    alpha1 = handles.ang1*pi/180;
755 r1 = sqrt((x-x1).^2+(y-y1).^2);
    theta1 = atan2(y-y1,x-x1);
757 handles.r1 = r1;
    handles.theta1 = theta1;
759 %Corrente uniforme 1
    uc1 = q1*cos(alpha1);
761 vc1 = q1*sin(alpha1);
    Uc1 = uc1*ones(length(x),length(x));
763 Vc1 = vc1*ones(length(x),length(x));
    psicor1 = -Vc1.*x+Uc1.*y;
765 phicor1 = Uc1.*x+Vc1.*y;
    handles.psic1 = psicor1;
767 handles.phic1 = phicor1;
    handles.uc1 = Uc1;
769 handles.vc1 = Vc1;
```

```
    %Sorgente 1
771 psisorg1 = q1./(2*pi).*theta1;
    phisorg1 = q1./(2*pi).*log(r1);
773 usorg1 = cos(theta1).*q1./(2*pi*r1);
    vsorg1 = sin(theta1).*q1./(2*pi*r1);
775 handles.psis1 = psisorg1;
    handles.phis1 = phisorg1;
777 handles.us1 = usorg1;
    handles.vs1 = vsorg1;
779 %Pozzo 1
    psipoz1 = -q1./(2*pi).*theta1;
781 phipoz1 = -q1./(2*pi).*log(r1);
    upoz1 = -cos(theta1).*q1./(2*pi*r1);
783 vpoz1 = -sin(theta1).*q1./(2*pi*r1);
    handles.psip1 = psipoz1;
785 handles.phip1 = phipoz1;
    handles.up1 = upoz1;
787 handles.vp1 = vpoz1;
    %Doppietta 1
789 psidop1 = sin(theta1-alpha1).*q1./(2*pi*r1);
    phidop1 = -cos(theta1-alpha1).*q1./(2*pi*r1);
791 vdop1 = q1./(2*pi*r1).*cos(theta1-alpha1).*sin(theta1)+q1./...
    ...(2*pi*r1).*sin(theta1-alpha1).*cos(theta1);
793 udop1 = q1./(2*pi*r1).*cos(theta1-alpha1).*cos(theta1)-q1./...
    ...(2*pi*r1).*sin(theta1-alpha1).*sin(theta1);
795 handles.psid1 = psidop1;
    handles.phid1 = phidop1;
797 handles.ud1 = udop1;
    handles.vd1 = vdop1;
799 %Vortice 1
    psivor1 = -q1./(2*pi).*log(r1);
801 phivor1 = q1./(2*pi).*theta1;
    uvor1 = -q1./(2*pi*r1).*sin(theta1);
803 vvor1 = q1./(2*pi*r1).*cos(theta1);
    handles.psiv1 = psivor1;
805 handles.phiv1 = phivor1;
    handles.uv1 = uvor1;
807 handles.vv1 = vvor1;
    % Set the current data value with popupmenu_1.
809 str = get(handles.popupmenu1, 'String');
    val1 = get(handles.popupmenu1,'Value');
811 % Set current data to the selected data set.
    switch str(val1,:)
813 case 'Corrente␣uniforme'
    handles.psi1 = handles.psic1;
815 handles.phi1 = handles.phic1;
    handles.u1 = handles.uc1;
817 handles.v1 = handles.vc1;
    case 'Sorgente␣␣␣␣␣␣␣␣␣'
```

```
819 handles.psi1 = handles.psis1;
    handles.phi1 = handles.phis1;
821 handles.u1 = handles.us1;
    handles.v1 = handles.vs1;
823 case 'Pozzo␣␣␣␣␣␣␣␣␣␣␣␣'
    handles.psi1 = handles.psip1;
825 handles.phi1 = handles.phip1;
    handles.u1 = handles.up1;
827 handles.v1 = handles.vp1;
    case 'Doppietta␣␣␣␣␣␣␣␣'
829 handles.psi1 = handles.psid1;
    handles.phi1 = handles.phid1;
831 handles.u1 = handles.ud1;
    handles.v1 = handles.vd1;
833 case 'Vortice␣␣␣␣␣␣␣␣␣␣'
    handles.psi1 = handles.psiv1;
835 handles.phi1 = handles.phiv1;
    handles.u1 = handles.uv1;
837 handles.v1 = handles.vv1;
    end
839 %Set the final current data
    handles.psi = handles.psi1 + handles.psi2 + handles.psi3 +...
841 ...handles.psi4 + handles.psi5;
    handles.phi = handles.phi1 + handles.phi2 + handles.phi3 +...
843 ...handles.phi4 + handles.phi5;
    handles.u = handles.u1 + handles.u2 + handles.u3 +...
845 ...handles.u4 + handles.u5;
    handles.v = handles.v1 + handles.v2 + handles.v3 +...
847 ...handles.v4 + handles.v5;
    % Save the handles structure.
849 guidata(hObject,handles)
    % Choose default command line output
851 handles.output = hObject;
    % Update handles structure
853 guidata(hObject, handles);

855 % --- Executes during object creation, after setting all properties.
    function x1_CreateFcn(hObject, eventdata, handles)
857
    if ispc && isequal(get(hObject,'BackgroundColor'),...
859 ...get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
861 end

863 function y1_Callback(hObject, eventdata, handles)
    handles.X1 = str2double(get(handles.x1,'String'));
865 handles.Y1 = str2double(get(handles.y1,'String'));
    handles.Q1 = str2double(get(handles.q1,'String'));
867 handles.ang1 = str2double(get(handles.alpha1,'String'));
```

```
x1 = handles.X1;
y1 = handles.Y1;
[x,y] = meshgrid([-1:0.01:x1-0.01, x1+0.01:.01:1],...
...[-1:0.01:y1-0.01, y1+0.01:.01:1]);
handles.x = x;
handles.y = y;
q1 = handles.Q1;
alpha1 = handles.ang1*pi/180;
r1 = sqrt((x-x1).^2+(y-y1).^2);
theta1 = atan2(y-y1,x-x1);
handles.r1 = r1;
handles.theta1 = theta1;
%Corrente uniforme 1
uc1 = q1*cos(alpha1);
vc1 = q1*sin(alpha1);
Uc1 = uc1*ones(length(x),length(x));
Vc1 = vc1*ones(length(x),length(x));
psicor1 = -Vc1.*x+Uc1.*y;
phicor1 = Uc1.*x+Vc1.*y;
handles.psic1 = psicor1;
handles.phic1 = phicor1;
handles.uc1 = Uc1;
handles.vc1 = Vc1;
%Sorgente 1
psisorg1 = q1./(2*pi).*theta1;
phisorg1 = q1./(2*pi).*log(r1);
usorg1 = cos(theta1).*q1./(2*pi*r1);
vsorg1 = sin(theta1).*q1./(2*pi*r1);
handles.psis1 = psisorg1;
handles.phis1 = phisorg1;
handles.us1 = usorg1;
handles.vs1 = vsorg1;
%Pozzo 1
psipoz1 = -q1./(2*pi).*theta1;
phipoz1 = -q1./(2*pi).*log(r1);
upoz1 = -cos(theta1).*q1./(2*pi*r1);
vpoz1 = -sin(theta1).*q1./(2*pi*r1);
handles.psip1 = psipoz1;
handles.phip1 = phipoz1;
handles.up1 = upoz1;
handles.vp1 = vpoz1;
%Doppietta 1
psidop1 = sin(theta1-alpha1).*q1./(2*pi*r1);
phidop1 = -cos(theta1-alpha1).*q1./(2*pi*r1);
vdop1 = q1./(2*pi*r1).*cos(theta1-alpha1).*sin(theta1)+q1./...
...(2*pi*r1).*sin(theta1-alpha1).*cos(theta1);
udop1 = q1./(2*pi*r1).*cos(theta1-alpha1).*cos(theta1)-q1./...
...(2*pi*r1).*sin(theta1-alpha1).*sin(theta1);
handles.psid1 = psidop1;
```

```
917 handles.phid1 = phidop1;
    handles.ud1 = udop1;
919 handles.vd1 = vdop1;
    %Vortice 1
921 psivor1 = -q1./(2*pi).*log(r1);
    phivor1 = q1./(2*pi).*theta1;
923 uvor1 = -q1./(2*pi*r1).*sin(theta1);
    vvor1 = q1./(2*pi*r1).*cos(theta1);
925 handles.psiv1 = psivor1;
    handles.phiv1 = phivor1;
927 handles.uv1 = uvor1;
    handles.vv1 = vvor1;
929 % Set the current data value with popupmenu_1.
    str = get(handles.popupmenu1, 'String');
931 val1 = get(handles.popupmenu1,'Value');
    % Set current data to the selected data set.
933 switch str(val1,:)
    case 'Corrente␣uniforme'
935 handles.psi1 = handles.psic1;
    handles.phi1 = handles.phic1;
937 handles.u1 = handles.uc1;
    handles.v1 = handles.vc1;
939 case 'Sorgente␣␣␣␣␣␣␣␣␣'
    handles.psi1 = handles.psis1;
941 handles.phi1 = handles.phis1;
    handles.u1 = handles.us1;
943 handles.v1 = handles.vs1;
    case 'Pozzo␣␣␣␣␣␣␣␣␣␣␣␣'
945 handles.psi1 = handles.psip1;
    handles.phi1 = handles.phip1;
947 handles.u1 = handles.up1;
    handles.v1 = handles.vp1;
949 case 'Doppietta␣␣␣␣␣␣␣␣'
    handles.psi1 = handles.psid1;
951 handles.phi1 = handles.phid1;
    handles.u1 = handles.ud1;
953 handles.v1 = handles.vd1;
    case 'Vortice␣␣␣␣␣␣␣␣␣␣'
955 handles.psi1 = handles.psiv1;
    handles.phi1 = handles.phiv1;
957 handles.u1 = handles.uv1;
    handles.v1 = handles.vv1;
959 end
    %Set the final current data
961 handles.psi = handles.psi1 + handles.psi2 + handles.psi3 +...
    ...handles.psi4 + handles.psi5;
963 handles.phi = handles.phi1 + handles.phi2 + handles.phi3 +...
    ...handles.phi4 + handles.phi5;
965 handles.u = handles.u1 + handles.u2 + handles.u3 +...
```

```
    ...handles.u4 + handles.u5;
967 handles.v = handles.v1 + handles.v2 + handles.v3 +...
    ...handles.v4 + handles.v5;
969 % Save the handles structure.
    guidata(hObject,handles)
971 % Choose default command line output
    handles.output = hObject;
973 % Update handles structure
    guidata(hObject, handles);
975
    % --- Executes during object creation, after setting all properties.
977 function y1_CreateFcn(hObject, eventdata, handles)

979 if ispc && isequal(get(hObject,'BackgroundColor'),...
    ...get(0,'defaultUicontrolBackgroundColor'))
981 set(hObject,'BackgroundColor','white');
    end
983
    function q1_Callback(hObject, eventdata, handles)
985 handles.X1 = str2double(get(handles.x1,'String'));
    handles.Y1 = str2double(get(handles.y1,'String'));
987 handles.Q1 = str2double(get(handles.q1,'String'));
    handles.ang1 = str2double(get(handles.alpha1,'String'));
989 x1 = handles.X1;
    y1 = handles.Y1;
991 [x,y] = meshgrid([-1:0.01:x1-0.01, x1+0.01:.01:1],...
    ...[-1:0.01:y1-0.01, y1+0.01:.01:1]);
993 handles.x = x;
    handles.y = y;
995 q1 = handles.Q1;
    alpha1 = handles.ang1*pi/180;
997 r1 = sqrt((x-x1).^2+(y-y1).^2);
    theta1 = atan2(y-y1,x-x1);
999 handles.r1 = r1;
    handles.theta1 = theta1;
1001 %Corrente uniforme 1
    uc1 = q1*cos(alpha1);
1003 vc1 = q1*sin(alpha1);
    Uc1 = uc1*ones(length(x),length(x));
1005 Vc1 = vc1*ones(length(x),length(x));
    psicor1 = -Vc1.*x+Uc1.*y;
1007 phicor1 = Uc1.*x+Vc1.*y;
    handles.psic1 = psicor1;
1009 handles.phic1 = phicor1;
    handles.uc1 = Uc1;
1011 handles.vc1 = Vc1;
    %Sorgente 1
1013 psisorg1 = q1./(2*pi).*theta1;
    phisorg1 = q1./(2*pi).*log(r1);
```

```
usorg1 = cos(theta1).*q1./(2*pi*r1);
vsorg1 = sin(theta1).*q1./(2*pi*r1);
handles.psis1 = psisorg1;
handles.phis1 = phisorg1;
handles.us1 = usorg1;
handles.vs1 = vsorg1;
%Pozzo 1
psipoz1 = -q1./(2*pi).*theta1;
phipoz1 = -q1./(2*pi).*log(r1);
upoz1 = -cos(theta1).*q1./(2*pi*r1);
vpoz1 = -sin(theta1).*q1./(2*pi*r1);
handles.psip1 = psipoz1;
handles.phip1 = phipoz1;
handles.up1 = upoz1;
handles.vp1 = vpoz1;
%Doppietta 1
psidop1 = sin(theta1-alpha1).*q1./(2*pi*r1);
phidop1 = -cos(theta1-alpha1).*q1./(2*pi*r1);
vdop1 = q1./(2*pi*r1).*cos(theta1-alpha1).*sin(theta1)+q1./...
...(2*pi*r1).*sin(theta1-alpha1).*cos(theta1);
udop1 = q1./(2*pi*r1).*cos(theta1-alpha1).*cos(theta1)-q1./...
...(2*pi*r1).*sin(theta1-alpha1).*sin(theta1);
handles.psid1 = psidop1;
handles.phid1 = phidop1;
handles.ud1 = udop1;
handles.vd1 = vdop1;
%Vortice 1
psivor1 = -q1./(2*pi).*log(r1);
phivor1 = q1./(2*pi).*theta1;
uvor1 = -q1./(2*pi*r1).*sin(theta1);
vvor1 = q1./(2*pi*r1).*cos(theta1);
handles.psiv1 = psivor1;
handles.phiv1 = phivor1;
handles.uv1 = uvor1;
handles.vv1 = vvor1;
% Set the current data value with popupmenu_1.
str = get(handles.popupmenu1, 'String');
val1 = get(handles.popupmenu1,'Value');
% Set current data to the selected data set.
switch str(val1,:)
case 'Corrente␣uniforme'
handles.psi1 = handles.psic1;
handles.phi1 = handles.phic1;
handles.u1 = handles.uc1;
handles.v1 = handles.vc1;
case 'Sorgente␣␣␣␣␣␣␣␣␣'
handles.psi1 = handles.psis1;
handles.phi1 = handles.phis1;
handles.u1 = handles.us1;
```

```
    handles.v1 = handles.vs1;
1065 case 'Pozzo␣␣␣␣␣␣␣␣␣␣␣␣'
    handles.psi1 = handles.psip1;
1067 handles.phi1 = handles.phip1;
    handles.u1 = handles.up1;
1069 handles.v1 = handles.vp1;
    case 'Doppietta␣␣␣␣␣␣␣␣'
1071 handles.psi1 = handles.psid1;
    handles.phi1 = handles.phid1;
1073 handles.u1 = handles.ud1;
    handles.v1 = handles.vd1;
1075 case 'Vortice␣␣␣␣␣␣␣␣␣␣'
    handles.psi1 = handles.psiv1;
1077 handles.phi1 = handles.phiv1;
    handles.u1 = handles.uv1;
1079 handles.v1 = handles.vv1;
    end
1081 %Set the final current data
    handles.psi = handles.psi1 + handles.psi2 + handles.psi3 +...
1083 ...handles.psi4 + handles.psi5;
    handles.phi = handles.phi1 + handles.phi2 + handles.phi3 +...
1085 ...handles.phi4 + handles.phi5;
    handles.u = handles.u1 + handles.u2 + handles.u3 +...
1087 ...handles.u4 + handles.u5;
    handles.v = handles.v1 + handles.v2 + handles.v3 +...
1089 ...handles.v4 + handles.v5;
    % Save the handles structure.
1091 guidata(hObject,handles)
    % Choose default command line output
1093 handles.output = hObject;
    % Update handles structure
1095 guidata(hObject, handles);

1097 % --- Executes during object creation, after setting all properties.
    function q1_CreateFcn(hObject, eventdata, handles)
1099
    if ispc && isequal(get(hObject,'BackgroundColor'),...
1101 ...get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
1103 end

1105 function alpha1_Callback(hObject, eventdata, handles)
    handles.X1 = str2double(get(handles.x1,'String'));
1107 handles.Y1 = str2double(get(handles.y1,'String'));
    handles.Q1 = str2double(get(handles.q1,'String'));
1109 handles.ang1 = str2double(get(handles.alpha1,'String'));
    x1 = handles.X1;
1111 y1 = handles.Y1;
    [x,y] = meshgrid([-1:0.01:x1-0.01, x1+0.01:.01:1],...
```

```
...[-1:0.01:y1-0.01, y1+0.01:.01:1]);
handles.x = x;
handles.y = y;
q1 = handles.Q1;
alpha1 = handles.ang1*pi/180;
r1 = sqrt((x-x1).^2+(y-y1).^2);
theta1 = atan2(y-y1,x-x1);
handles.r1 = r1;
handles.theta1 = theta1;
%Corrente uniforme 1
uc1 = q1*cos(alpha1);
vc1 = q1*sin(alpha1);
Uc1 = uc1*ones(length(x),length(x));
Vc1 = vc1*ones(length(x),length(x));
psicor1 = -Vc1.*x+Uc1.*y;
phicor1 = Uc1.*x+Vc1.*y;
handles.psic1 = psicor1;
handles.phic1 = phicor1;
handles.uc1 = Uc1;
handles.vc1 = Vc1;
%Sorgente 1
psisorg1 = q1./(2*pi).*theta1;
phisorg1 = q1./(2*pi).*log(r1);
usorg1 = cos(theta1).*q1./(2*pi*r1);
vsorg1 = sin(theta1).*q1./(2*pi*r1);
handles.psis1 = psisorg1;
handles.phis1 = phisorg1;
handles.us1 = usorg1;
handles.vs1 = vsorg1;
%Pozzo 1
psipoz1 = -q1./(2*pi).*theta1;
phipoz1 = -q1./(2*pi).*log(r1);
upoz1 = -cos(theta1).*q1./(2*pi*r1);
vpoz1 = -sin(theta1).*q1./(2*pi*r1);
handles.psip1 = psipoz1;
handles.phip1 = phipoz1;
handles.up1 = upoz1;
handles.vp1 = vpoz1;
%Doppietta 1
psidop1 = sin(theta1-alpha1).*q1./(2*pi*r1);
phidop1 = -cos(theta1-alpha1).*q1./(2*pi*r1);
vdop1 = q1./(2*pi*r1).*cos(theta1-alpha1).*sin(theta1)+q1./...
...(2*pi*r1).*sin(theta1-alpha1).*cos(theta1);
udop1 = q1./(2*pi*r1).*cos(theta1-alpha1).*cos(theta1)-q1./...
...(2*pi*r1).*sin(theta1-alpha1).*sin(theta1);
handles.psid1 = psidop1;
handles.phid1 = phidop1;
handles.ud1 = udop1;
handles.vd1 = vdop1;
```

```
    %Vortice 1
1163 psivor1 = -q1./(2*pi).*log(r1);
    phivor1 = q1./(2*pi).*theta1;
1165 uvor1 = -q1./(2*pi*r1).*sin(theta1);
    vvor1 = q1./(2*pi*r1).*cos(theta1);
1167 handles.psiv1 = psivor1;
    handles.phiv1 = phivor1;
1169 handles.uv1 = uvor1;
    handles.vv1 = vvor1;
1171 % Set the current data value with popupmenu_1.
    str = get(handles.popupmenu1, 'String');
1173 val1 = get(handles.popupmenu1,'Value');
    % Set current data to the selected data set.
1175 switch str(val1,:)
    case 'Corrente␣uniforme'
1177 handles.psi1 = handles.psic1;
    handles.phi1 = handles.phic1;
1179 handles.u1 = handles.uc1;
    handles.v1 = handles.vc1;
1181 case 'Sorgente␣␣␣␣␣␣␣␣␣'
    handles.psi1 = handles.psis1;
1183 handles.phi1 = handles.phis1;
    handles.u1 = handles.us1;
1185 handles.v1 = handles.vs1;
    case 'Pozzo␣␣␣␣␣␣␣␣␣␣␣␣'
1187 handles.psi1 = handles.psip1;
    handles.phi1 = handles.phip1;
1189 handles.u1 = handles.up1;
    handles.v1 = handles.vp1;
1191 case 'Doppietta␣␣␣␣␣␣␣␣'
    handles.psi1 = handles.psid1;
1193 handles.phi1 = handles.phid1;
    handles.u1 = handles.ud1;
1195 handles.v1 = handles.vd1;
    case 'Vortice␣␣␣␣␣␣␣␣␣␣'
1197 handles.psi1 = handles.psiv1;
    handles.phi1 = handles.phiv1;
1199 handles.u1 = handles.uv1;
    handles.v1 = handles.vv1;
1201 end
    %Set the final current data
1203 handles.psi = handles.psi1 + handles.psi2 + handles.psi3 +...
    ...handles.psi4 + handles.psi5;
1205 handles.phi = handles.phi1 + handles.phi2 + handles.phi3 +...
    ...handles.phi4 + handles.phi5;
1207 handles.u = handles.u1 + handles.u2 + handles.u3 +...
    ...handles.u4 + handles.u5;
1209 handles.v = handles.v1 + handles.v2 + handles.v3 +...
    ...handles.v4 + handles.v5;
```

```
% Save the handles structure.
guidata(hObject,handles)
% Choose default command line output
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function alpha1_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
...get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function x2_Callback(hObject, eventdata, handles)
handles.X2 = str2double(get(handles.x2,'String'));
handles.Y2 = str2double(get(handles.y2,'String'));
handles.Q2 = str2double(get(handles.q2,'String'));
handles.ang2 = str2double(get(handles.alpha2,'String'));
x2 = handles.X2;
y2 = handles.Y2;
[x,y] = meshgrid([-1:0.01:x2-0.01, x2+0.01:.01:1],...
...[-1:0.01:y2-0.01, y2+0.01:.01:1]);
handles.x = x;
handles.y = y;
q2 = handles.Q2;
alpha2 = handles.ang2*pi/180;
r2 = sqrt((x-x2).^2+(y-y2).^2);
theta2 = atan2(y-y2,x-x2);
handles.r2 = r2;
handles.theta2 = theta2;
%Corrente uniforme 2
uc2 = q2*cos(alpha2);
vc2 = q2*sin(alpha2);
Uc2 = uc2*ones(length(x),length(x));
Vc2 = vc2*ones(length(x),length(x));
psicor2 = -Vc2.*x+Uc2.*y;
phicor2 = Uc2.*x+Vc2.*y;
handles.psic2 = psicor2;
handles.phic2 = phicor2;
handles.uc2 = Uc2;
handles.vc2 = Vc2;
%Sorgente 2
psisorg2 = q2./(2*pi).*theta2;
phisorg2 = q2./(2*pi).*log(r2);
usorg2 = cos(theta2).*q2./(2*pi*r2);
vsorg2 = sin(theta2).*q2./(2*pi*r2);
handles.psis2 = psisorg2;
```

```
    handles.phis2 = phisorg2;
1261 handles.us2 = usorg2;
    handles.vs2 = vsorg2;
1263 %Pozzo 2
    psipoz2 = -q2./(2*pi).*theta2;
1265 phipoz2 = -q2./(2*pi).*log(r2);
    upoz2 = -cos(theta2).*q2./(2*pi*r2);
1267 vpoz2 = -sin(theta2).*q2./(2*pi*r2);
    handles.psip2 = psipoz2;
1269 handles.phip2 = phipoz2;
    handles.up2 = upoz2;
1271 handles.vp2 = vpoz2;
    %Doppietta 2
1273 psidop2 = sin(theta2-alpha2).*q2./(2*pi*r2);
    phidop2 = -cos(theta2-alpha2).*q2./(2*pi*r2);
1275 vdop2 = q2./(2*pi*r2).*cos(theta2-alpha2).*sin(theta2)+q2./...
    ...(2*pi*r2).*sin(theta2-alpha2).*cos(theta2);
1277 udop2 = q2./(2*pi*r2).*cos(theta2-alpha2).*cos(theta2)-q2./...
    ...(2*pi*r2).*sin(theta2-alpha2).*sin(theta2);
1279 handles.psid2 = psidop2;
    handles.phid2 = phidop2;
1281 handles.ud2 = udop2;
    handles.vd2 = vdop2;
1283 %Vortice 2
    psivor2 = -q2./(2*pi).*log(r2);
1285 phivor2 = q2./(2*pi).*theta2;
    uvor2 = -q2./(2*pi*r2).*sin(theta2);
1287 vvor2 = q2./(2*pi*r2).*cos(theta2);
    handles.psiv2 = psivor2;
1289 handles.phiv2 = phivor2;
    handles.uv2 = uvor2;
1291 handles.vv2 = vvor2;
    % Set the current data value with popupmenu_2.
1293 str = get(handles.popupmenu2, 'String');
    val2 = get(handles.popupmenu2,'Value');
1295 % Set current data to the selected data set.
    switch str(val2,:)
1297 case 'Corrente␣uniforme'
    handles.psi2 = handles.psic2;
1299 handles.phi2 = handles.phic2;
    handles.u2 = handles.uc2;
1301 handles.v2 = handles.vc2;
    case 'Sorgente␣␣␣␣␣␣␣␣␣'
1303 handles.psi2 = handles.psis2;
    handles.phi2 = handles.phis2;
1305 handles.u2 = handles.us2;
    handles.v2 = handles.vs2;
1307 case 'Pozzo␣␣␣␣␣␣␣␣␣␣␣␣'
    handles.psi2 = handles.psip2;
```

```
handles.phi2 = handles.phip2;
handles.u2 = handles.up2;
handles.v2 = handles.vp2;
case 'Doppietta         '
handles.psi2 = handles.psid2;
handles.phi2 = handles.phid2;
handles.u2 = handles.ud2;
handles.v2 = handles.vd2;
case 'Vortice           '
handles.psi2 = handles.psiv2;
handles.phi2 = handles.phiv2;
handles.u2 = handles.uv2;
handles.v2 = handles.vv2;
end
%Set the final current data
handles.psi = handles.psi1 + handles.psi2 + handles.psi3 +...
...handles.psi4 + handles.psi5;
handles.phi = handles.phi1 + handles.phi2 + handles.phi3 +...
...handles.phi4 + handles.phi5;
handles.u = handles.u1 + handles.u2 + handles.u3 +...
...handles.u4 + handles.u5;
handles.v = handles.v1 + handles.v2 + handles.v3 +...
...handles.v4 + handles.v5;
% Save the handles structure.
guidata(hObject,handles)
% Choose default command line output
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function x2_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
...get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function y2_Callback(hObject, eventdata, handles)
handles.X2 = str2double(get(handles.x2,'String'));
handles.Y2 = str2double(get(handles.y2,'String'));
handles.Q2 = str2double(get(handles.q2,'String'));
handles.ang2 = str2double(get(handles.alpha2,'String'));
x2 = handles.X2;
y2 = handles.Y2;
[x,y] = meshgrid([-1:0.01:x2-0.01, x2+0.01:.01:1],...
...[-1:0.01:y2-0.01, y2+0.01:.01:1]);
handles.x = x;
handles.y = y;
```

```
    q2 = handles.Q2;
1359 alpha2 = handles.ang2*pi/180;
    r2 = sqrt((x-x2).^2+(y-y2).^2);
1361 theta2 = atan2(y-y2,x-x2);
    handles.r2 = r2;
1363 handles.theta2 = theta2;
    %Corrente uniforme 2
1365 uc2 = q2*cos(alpha2);
    vc2 = q2*sin(alpha2);
1367 Uc2 = uc2*ones(length(x),length(x));
    Vc2 = vc2*ones(length(x),length(x));
1369 psicor2 = -Vc2.*x+Uc2.*y;
    phicor2 = Uc2.*x+Vc2.*y;
1371 handles.psic2 = psicor2;
    handles.phic2 = phicor2;
1373 handles.uc2 = Uc2;
    handles.vc2 = Vc2;
1375 %Sorgente 2
    psisorg2 = q2./(2*pi).*theta2;
1377 phisorg2 = q2./(2*pi).*log(r2);
    usorg2 = cos(theta2).*q2./(2*pi*r2);
1379 vsorg2 = sin(theta2).*q2./(2*pi*r2);
    handles.psis2 = psisorg2;
1381 handles.phis2 = phisorg2;
    handles.us2 = usorg2;
1383 handles.vs2 = vsorg2;
    %Pozzo 2
1385 psipoz2 = -q2./(2*pi).*theta2;
    phipoz2 = -q2./(2*pi).*log(r2);
1387 upoz2 = -cos(theta2).*q2./(2*pi*r2);
    vpoz2 = -sin(theta2).*q2./(2*pi*r2);
1389 handles.psip2 = psipoz2;
    handles.phip2 = phipoz2;
1391 handles.up2 = upoz2;
    handles.vp2 = vpoz2;
1393 %Doppietta 2
    psidop2 = sin(theta2-alpha2).*q2./(2*pi*r2);
1395 phidop2 = -cos(theta2-alpha2).*q2./(2*pi*r2);
    vdop2 = q2./(2*pi*r2).*cos(theta2-alpha2).*sin(theta2)+q2./...
1397 ...(2*pi*r2).*sin(theta2-alpha2).*cos(theta2);
    udop2 = q2./(2*pi*r2).*cos(theta2-alpha2).*cos(theta2)-q2./...
1399 ...(2*pi*r2).*sin(theta2-alpha2).*sin(theta2);
    handles.psid2 = psidop2;
1401 handles.phid2 = phidop2;
    handles.ud2 = udop2;
1403 handles.vd2 = vdop2;
    %Vortice 2
1405 psivor2 = -q2./(2*pi).*log(r2);
    phivor2 = q2./(2*pi).*theta2;
```

```
1407  uvor2 = -q2./(2*pi*r2).*sin(theta2);
      vvor2 = q2./(2*pi*r2).*cos(theta2);
1409  handles.psiv2 = psivor2;
      handles.phiv2 = phivor2;
1411  handles.uv2 = uvor2;
      handles.vv2 = vvor2;
1413  % Set the current data value with popupmenu_2.
      str = get(handles.popupmenu2, 'String');
1415  val2 = get(handles.popupmenu2,'Value');
      % Set current data to the selected data set.
1417  switch str(val2,:)
      case 'Corrente uniforme'
1419  handles.psi2 = handles.psic2;
      handles.phi2 = handles.phic2;
1421  handles.u2 = handles.uc2;
      handles.v2 = handles.vc2;
1423  case 'Sorgente          '
      handles.psi2 = handles.psis2;
1425  handles.phi2 = handles.phis2;
      handles.u2 = handles.us2;
1427  handles.v2 = handles.vs2;
      case 'Pozzo             '
1429  handles.psi2 = handles.psip2;
      handles.phi2 = handles.phip2;
1431  handles.u2 = handles.up2;
      handles.v2 = handles.vp2;
1433  case 'Doppietta         '
      handles.psi2 = handles.psid2;
1435  handles.phi2 = handles.phid2;
      handles.u2 = handles.ud2;
1437  handles.v2 = handles.vd2;
      case 'Vortice           '
1439  handles.psi2 = handles.psiv2;
      handles.phi2 = handles.phiv2;
1441  handles.u2 = handles.uv2;
      handles.v2 = handles.vv2;
1443  end
      %Set the final current data
1445  handles.psi = handles.psi1 + handles.psi2 + handles.psi3 +...
      ...handles.psi4 + handles.psi5;
1447  handles.phi = handles.phi1 + handles.phi2 + handles.phi3 +...
      ...handles.phi4 + handles.phi5;
1449  handles.u = handles.u1 + handles.u2 + handles.u3 +...
      ...handles.u4 + handles.u5;
1451  handles.v = handles.v1 + handles.v2 + handles.v3 +...
      ...handles.v4 + handles.v5;
1453  % Save the handles structure.
      guidata(hObject,handles)
1455  % Choose default command line output
```

```
    handles.output = hObject;
1457 % Update handles structure
    guidata(hObject, handles);
1459
    % --- Executes during object creation, after setting all properties.
1461 function y2_CreateFcn(hObject, eventdata, handles)

1463 if ispc && isequal(get(hObject,'BackgroundColor'),...
    ...get(0,'defaultUicontrolBackgroundColor'))
1465 set(hObject,'BackgroundColor','white');
    end
1467
    function q2_Callback(hObject, eventdata, handles)
1469 handles.X2 = str2double(get(handles.x2,'String'));
    handles.Y2 = str2double(get(handles.y2,'String'));
1471 handles.Q2 = str2double(get(handles.q2,'String'));
    handles.ang2 = str2double(get(handles.alpha2,'String'));
1473 x2 = handles.X2;
    y2 = handles.Y2;
1475 [x,y] = meshgrid([-1:0.01:x2-0.01, x2+0.01:.01:1],...
    ...[-1:0.01:y2-0.01, y2+0.01:.01:1]);
1477 handles.x = x;
    handles.y = y;
1479 q2 = handles.Q2;
    alpha2 = handles.ang2*pi/180;
1481 r2 = sqrt((x-x2).^2+(y-y2).^2);
    theta2 = atan2(y-y2,x-x2);
1483 handles.r2 = r2;
    handles.theta2 = theta2;
1485 %Corrente uniforme 2
    uc2 = q2*cos(alpha2);
1487 vc2 = q2*sin(alpha2);
    Uc2 = uc2*ones(length(x),length(x));
1489 Vc2 = vc2*ones(length(x),length(x));
    psicor2 = -Vc2.*x+Uc2.*y;
1491 phicor2 = Uc2.*x+Vc2.*y;
    handles.psic2 = psicor2;
1493 handles.phic2 = phicor2;
    handles.uc2 = Uc2;
1495 handles.vc2 = Vc2;
    %Sorgente 2
1497 psisorg2 = q2./(2*pi).*theta2;
    phisorg2 = q2./(2*pi).*log(r2);
1499 usorg2 = cos(theta2).*q2./(2*pi*r2);
    vsorg2 = sin(theta2).*q2./(2*pi*r2);
1501 handles.psis2 = psisorg2;
    handles.phis2 = phisorg2;
1503 handles.us2 = usorg2;
    handles.vs2 = vsorg2;
```

```
%Pozzo 2
psipoz2 = -q2./(2*pi).*theta2;
phipoz2 = -q2./(2*pi).*log(r2);
upoz2 = -cos(theta2).*q2./(2*pi*r2);
vpoz2 = -sin(theta2).*q2./(2*pi*r2);
handles.psip2 = psipoz2;
handles.phip2 = phipoz2;
handles.up2 = upoz2;
handles.vp2 = vpoz2;
%Doppietta 2
psidop2 = sin(theta2-alpha2).*q2./(2*pi*r2);
phidop2 = -cos(theta2-alpha2).*q2./(2*pi*r2);
vdop2 = q2./(2*pi*r2).*cos(theta2-alpha2).*sin(theta2)+q2./...
...(2*pi*r2).*sin(theta2-alpha2).*cos(theta2);
udop2 = q2./(2*pi*r2).*cos(theta2-alpha2).*cos(theta2)-q2./...
...(2*pi*r2).*sin(theta2-alpha2).*sin(theta2);
handles.psid2 = psidop2;
handles.phid2 = phidop2;
handles.ud2 = udop2;
handles.vd2 = vdop2;
%Vortice 2
psivor2 = -q2./(2*pi).*log(r2);
phivor2 = q2./(2*pi).*theta2;
uvor2 = -q2./(2*pi*r2).*sin(theta2);
vvor2 = q2./(2*pi*r2).*cos(theta2);
handles.psiv2 = psivor2;
handles.phiv2 = phivor2;
handles.uv2 = uvor2;
handles.vv2 = vvor2;
% Set the current data value with popupmenu_2.
str = get(handles.popupmenu2, 'String');
val2 = get(handles.popupmenu2,'Value');
% Set current data to the selected data set.
switch str(val2,:)
case 'Corrente␣uniforme'
handles.psi2 = handles.psic2;
handles.phi2 = handles.phic2;
handles.u2 = handles.uc2;
handles.v2 = handles.vc2;
case 'Sorgente␣␣␣␣␣␣␣␣␣'
handles.psi2 = handles.psis2;
handles.phi2 = handles.phis2;
handles.u2 = handles.us2;
handles.v2 = handles.vs2;
case 'Pozzo␣␣␣␣␣␣␣␣␣␣␣␣'
handles.psi2 = handles.psip2;
handles.phi2 = handles.phip2;
handles.u2 = handles.up2;
handles.v2 = handles.vp2;
```

```
    case 'Doppietta␣␣␣␣␣␣␣␣'
1555 handles.psi2 = handles.psid2;
    handles.phi2 = handles.phid2;
1557 handles.u2 = handles.ud2;
    handles.v2 = handles.vd2;
1559 case 'Vortice␣␣␣␣␣␣␣␣␣␣'
    handles.psi2 = handles.psiv2;
1561 handles.phi2 = handles.phiv2;
    handles.u2 = handles.uv2;
1563 handles.v2 = handles.vv2;
    end
1565 %Set the final current data
    handles.psi = handles.psi1 + handles.psi2 + handles.psi3 +...
1567 ...handles.psi4 + handles.psi5;
    handles.phi = handles.phi1 + handles.phi2 + handles.phi3 +...
1569 ...handles.phi4 + handles.phi5;
    handles.u = handles.u1 + handles.u2 + handles.u3 +...
1571 ...handles.u4 + handles.u5;
    handles.v = handles.v1 + handles.v2 + handles.v3 +...
1573 ...handles.v4 + handles.v5;
    % Save the handles structure.
1575 guidata(hObject,handles)
    % Choose default command line output
1577 handles.output = hObject;
    % Update handles structure
1579 guidata(hObject, handles);

1581 % --- Executes during object creation, after setting all properties.
    function q2_CreateFcn(hObject, eventdata, handles)
1583
    if ispc && isequal(get(hObject,'BackgroundColor'),...
1585 ...get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
1587 end

1589 function alpha2_Callback(hObject, eventdata, handles)
    handles.X2 = str2double(get(handles.x2,'String'));
1591 handles.Y2 = str2double(get(handles.y2,'String'));
    handles.Q2 = str2double(get(handles.q2,'String'));
1593 handles.ang2 = str2double(get(handles.alpha2,'String'));
    x2 = handles.X2;
1595 y2 = handles.Y2;
    [x,y] = meshgrid([-1:0.01:x2-0.01, x2+0.01:.01:1],...
1597 ...[-1:0.01:y2-0.01, y2+0.01:.01:1]);
    handles.x = x;
1599 handles.y = y;
    q2 = handles.Q2;
1601 alpha2 = handles.ang2*pi/180;
    r2 = sqrt((x-x2).^2+(y-y2).^2);
```

```
theta2 = atan2(y-y2,x-x2);
handles.r2 = r2;
handles.theta2 = theta2;
%Corrente uniforme 2
uc2 = q2*cos(alpha2);
vc2 = q2*sin(alpha2);
Uc2 = uc2*ones(length(x),length(x));
Vc2 = vc2*ones(length(x),length(x));
psicor2 = -Vc2.*x+Uc2.*y;
phicor2 = Uc2.*x+Vc2.*y;
handles.psic2 = psicor2;
handles.phic2 = phicor2;
handles.uc2 = Uc2;
handles.vc2 = Vc2;
%Sorgente 2
psisorg2 = q2./(2*pi).*theta2;
phisorg2 = q2./(2*pi).*log(r2);
usorg2 = cos(theta2).*q2./(2*pi*r2);
vsorg2 = sin(theta2).*q2./(2*pi*r2);
handles.psis2 = psisorg2;
handles.phis2 = phisorg2;
handles.us2 = usorg2;
handles.vs2 = vsorg2;
%Pozzo 2
psipoz2 = -q2./(2*pi).*theta2;
phipoz2 = -q2./(2*pi).*log(r2);
upoz2 = -cos(theta2).*q2./(2*pi*r2);
vpoz2 = -sin(theta2).*q2./(2*pi*r2);
handles.psip2 = psipoz2;
handles.phip2 = phipoz2;
handles.up2 = upoz2;
handles.vp2 = vpoz2;
%Doppietta 2
psidop2 = sin(theta2-alpha2).*q2./(2*pi*r2);
phidop2 = -cos(theta2-alpha2).*q2./(2*pi*r2);
vdop2 = q2./(2*pi*r2).*cos(theta2-alpha2).*sin(theta2)+q2./...
...(2*pi*r2).*sin(theta2-alpha2).*cos(theta2);
udop2 = q2./(2*pi*r2).*cos(theta2-alpha2).*cos(theta2)-q2./...
...(2*pi*r2).*sin(theta2-alpha2).*sin(theta2);
handles.psid2 = psidop2;
handles.phid2 = phidop2;
handles.ud2 = udop2;
handles.vd2 = vdop2;
%Vortice 2
psivor2 = -q2./(2*pi).*log(r2);
phivor2 = q2./(2*pi).*theta2;
uvor2 = -q2./(2*pi*r2).*sin(theta2);
vvor2 = q2./(2*pi*r2).*cos(theta2);
handles.psiv2 = psivor2;
```

```
    handles.phiv2 = phivor2;
1653 handles.uv2 = uvor2;
    handles.vv2 = vvor2;
1655 % Set the current data value with popupmenu_2.
    str = get(handles.popupmenu2, 'String');
1657 val2 = get(handles.popupmenu2,'Value');
    % Set current data to the selected data set.
1659 switch str(val2,:)
    case 'Corrente␣uniforme'
1661 handles.psi2 = handles.psic2;
    handles.phi2 = handles.phic2;
1663 handles.u2 = handles.uc2;
    handles.v2 = handles.vc2;
1665 case 'Sorgente␣␣␣␣␣␣␣␣␣'
    handles.psi2 = handles.psis2;
1667 handles.phi2 = handles.phis2;
    handles.u2 = handles.us2;
1669 handles.v2 = handles.vs2;
    case 'Pozzo␣␣␣␣␣␣␣␣␣␣␣␣'
1671 handles.psi2 = handles.psip2;
    handles.phi2 = handles.phip2;
1673 handles.u2 = handles.up2;
    handles.v2 = handles.vp2;
1675 case 'Doppietta␣␣␣␣␣␣␣␣'
    handles.psi2 = handles.psid2;
1677 handles.phi2 = handles.phid2;
    handles.u2 = handles.ud2;
1679 handles.v2 = handles.vd2;
    case 'Vortice␣␣␣␣␣␣␣␣␣␣'
1681 handles.psi2 = handles.psiv2;
    handles.phi2 = handles.phiv2;
1683 handles.u2 = handles.uv2;
    handles.v2 = handles.vv2;
1685 end
    %Set the final current data
1687 handles.psi = handles.psi1 + handles.psi2 + handles.psi3 +...
    ...handles.psi4 + handles.psi5;
1689 handles.phi = handles.phi1 + handles.phi2 + handles.phi3 +...
    ...handles.phi4 + handles.phi5;
1691 handles.u = handles.u1 + handles.u2 + handles.u3 +...
    ...handles.u4 + handles.u5;
1693 handles.v = handles.v1 + handles.v2 + handles.v3 +...
    ...handles.v4 + handles.v5;
1695 % Save the handles structure.
    guidata(hObject,handles)
1697 % Choose default command line output
    handles.output = hObject;
1699 % Update handles structure
    guidata(hObject, handles);
```

```
1701
    % --- Executes during object creation, after setting all properties.
1703 function alpha2_CreateFcn(hObject, eventdata, handles)

1705 if ispc && isequal(get(hObject,'BackgroundColor'),...
    ...get(0,'defaultUicontrolBackgroundColor'))
1707 set(hObject,'BackgroundColor','white');
    end
1709
    function x3_Callback(hObject, eventdata, handles)
1711 handles.X3 = str2double(get(handles.x3,'String'));
    handles.Y3 = str2double(get(handles.y3,'String'));
1713 handles.Q3 = str2double(get(handles.q3,'String'));
    handles.ang3 = str2double(get(handles.alpha3,'String'));
1715 x3 = handles.X3;
    y3 = handles.Y3;
1717 [x,y] = meshgrid([-1:0.01:x3-0.01, x3+0.01:.01:1],...
    ...[-1:0.01:y3-0.01, y3+0.01:.01:1]);
1719 handles.x = x;
    handles.y = y;
1721 q3 = handles.Q3;
    alpha3 = handles.ang3*pi/180;
1723 r3 = sqrt((x-x3).^2+(y-y3).^2);
    theta3 = atan2(y-y3,x-x3);
1725 handles.r3 = r3;
    handles.theta3 = theta3;
1727 %Corrente uniforme 3
    uc3 = q3*cos(alpha3);
1729 vc3 = q3*sin(alpha3);
    Uc3 = uc3*ones(length(x),length(x));
1731 Vc3 = vc3*ones(length(x),length(x));
    psicor3 = -Vc3.*x+Uc3.*y;
1733 phicor3 = Uc3.*x+Vc3.*y;
    handles.psic3 = psicor3;
1735 handles.phic3 = phicor3;
    handles.uc3 = Uc3;
1737 handles.vc3 = Vc3;
    %Sorgente 3
1739 psisorg3 = q3./(2*pi).*theta3;
    phisorg3 = q3./(2*pi).*log(r3);
1741 usorg3 = cos(theta3).*q3./(2*pi*r3);
    vsorg3 = sin(theta3).*q3./(2*pi*r3);
1743 handles.psis3 = psisorg3;
    handles.phis3 = phisorg3;
1745 handles.us3 = usorg3;
    handles.vs3 = vsorg3;
1747 %Pozzo 3
    psipoz3 = -q3./(2*pi).*theta3;
1749 phipoz3 = -q3./(2*pi).*log(r3);
```

```
    upoz3 = -cos(theta3).*q3./(2*pi*r3);
1751 vpoz3 = -sin(theta3).*q3./(2*pi*r3);
    handles.psip3 = psipoz3;
1753 handles.phip3 = phipoz3;
    handles.up3 = upoz3;
1755 handles.vp3 = vpoz3;
    %Doppietta 3
1757 psidop3 = sin(theta3-alpha3).*q3./(2*pi*r3);
    phidop3 = -cos(theta3-alpha3).*q3./(2*pi*r3);
1759 vdop3 = q3./(2*pi*r3).*cos(theta3-alpha3).*sin(theta3)+q3./...
    ...(2*pi*r3).*sin(theta3-alpha3).*cos(theta3);
1761 udop3 = q3./(2*pi*r3).*cos(theta3-alpha3).*cos(theta3)-q3./...
    ...(2*pi*r3).*sin(theta3-alpha3).*sin(theta3);
1763 handles.psid3 = psidop3;
    handles.phid3 = phidop3;
1765 handles.ud3 = udop3;
    handles.vd3 = vdop3;
1767 %Vortice 3
    psivor3 = -q3./(2*pi).*log(r3);
1769 phivor3 = q3./(2*pi).*theta3;
    uvor3 = -q3./(2*pi*r3).*sin(theta3);
1771 vvor3 = q3./(2*pi*r3).*cos(theta3);
    handles.psiv3 = psivor3;
1773 handles.phiv3 = phivor3;
    handles.uv3 = uvor3;
1775 handles.vv3 = vvor3;
    % Set the current data value with popupmenu_3.
1777 str = get(handles.popupmenu3, 'String');
    val3 = get(handles.popupmenu3,'Value');
1779 % Set current data to the selected data set.
    switch str(val3,:)
1781 case 'Corrente␣uniforme'
    handles.psi3 = handles.psic3;
1783 handles.phi3 = handles.phic3;
    handles.u3 = handles.uc3;
1785 handles.v3 = handles.vc3;
    case 'Sorgente␣␣␣␣␣␣␣␣␣'
1787 handles.psi3 = handles.psis3;
    handles.phi3 = handles.phis3;
1789 handles.u3 = handles.us3;
    handles.v3 = handles.vs3;
1791 case 'Pozzo␣␣␣␣␣␣␣␣␣␣␣␣'
    handles.psi3 = handles.psip3;
1793 handles.phi3 = handles.phip3;
    handles.u3 = handles.up3;
1795 handles.v3 = handles.vp3;
    case 'Doppietta␣␣␣␣␣␣␣␣'
1797 handles.psi3 = handles.psid3;
    handles.phi3 = handles.phid3;
```

```
1799  handles.u3 = handles.ud3;
      handles.v3 = handles.vd3;
1801  case 'Vortice␣␣␣␣␣␣␣␣␣␣'
      handles.psi3 = handles.psiv3;
1803  handles.phi3 = handles.phiv3;
      handles.u3 = handles.uv3;
1805  handles.v3 = handles.vv3;
      end
1807  %Set the final current data
      handles.psi = handles.psi1 + handles.psi2 + handles.psi3 +...
1809  ...handles.psi4 + handles.psi5;
      handles.phi = handles.phi1 + handles.phi2 + handles.phi3 +...
1811  ...handles.phi4 + handles.phi5;
      handles.u = handles.u1 + handles.u2 + handles.u3 +...
1813  ...handles.u4 + handles.u5;
      handles.v = handles.v1 + handles.v2 + handles.v3 +...
1815  ...handles.v4 + handles.v5;
      % Save the handles structure.
1817  guidata(hObject,handles)
      % Choose default command line output
1819  handles.output = hObject;
      % Update handles structure
1821  guidata(hObject, handles);

1823  % --- Executes during object creation, after setting all properties.
      function x3_CreateFcn(hObject, eventdata, handles)
1825
      if ispc && isequal(get(hObject,'BackgroundColor'),...
1827  ...get(0,'defaultUicontrolBackgroundColor'))
      set(hObject,'BackgroundColor','white');
1829  end

1831  function y3_Callback(hObject, eventdata, handles)
      handles.X3 = str2double(get(handles.x3,'String'));
1833  handles.Y3 = str2double(get(handles.y3,'String'));
      handles.Q3 = str2double(get(handles.q3,'String'));
1835  handles.ang3 = str2double(get(handles.alpha3,'String'));
      x3 = handles.X3;
1837  y3 = handles.Y3;
      [x,y] = meshgrid([-1:0.01:x3-0.01, x3+0.01:.01:1],...
1839  ...[-1:0.01:y3-0.01, y3+0.01:.01:1]);
      handles.x = x;
1841  handles.y = y;
      q3 = handles.Q3;
1843  alpha3 = handles.ang3*pi/180;
      r3 = sqrt((x-x3).^2+(y-y3).^2);
1845  theta3 = atan2(y-y3,x-x3);
      handles.r3 = r3;
1847  handles.theta3 = theta3;
```

```
    %Corrente uniforme 3
1849 uc3 = q3*cos(alpha3);
    vc3 = q3*sin(alpha3);
1851 Uc3 = uc3*ones(length(x),length(x));
    Vc3 = vc3*ones(length(x),length(x));
1853 psicor3 = -Vc3.*x+Uc3.*y;
    phicor3 = Uc3.*x+Vc3.*y;
1855 handles.psic3 = psicor3;
    handles.phic3 = phicor3;
1857 handles.uc3 = Uc3;
    handles.vc3 = Vc3;
1859 %Sorgente 3
    psisorg3 = q3./(2*pi).*theta3;
1861 phisorg3 = q3./(2*pi).*log(r3);
    usorg3 = cos(theta3).*q3./(2*pi*r3);
1863 vsorg3 = sin(theta3).*q3./(2*pi*r3);
    handles.psis3 = psisorg3;
1865 handles.phis3 = phisorg3;
    handles.us3 = usorg3;
1867 handles.vs3 = vsorg3;
    %Pozzo 3
1869 psipoz3 = -q3./(2*pi).*theta3;
    phipoz3 = -q3./(2*pi).*log(r3);
1871 upoz3 = -cos(theta3).*q3./(2*pi*r3);
    vpoz3 = -sin(theta3).*q3./(2*pi*r3);
1873 handles.psip3 = psipoz3;
    handles.phip3 = phipoz3;
1875 handles.up3 = upoz3;
    handles.vp3 = vpoz3;
1877 %Doppietta 3
    psidop3 = sin(theta3-alpha3).*q3./(2*pi*r3);
1879 phidop3 = -cos(theta3-alpha3).*q3./(2*pi*r3);
    vdop3 = q3./(2*pi*r3).*cos(theta3-alpha3).*sin(theta3)+q3./...
1881 ...(2*pi*r3).*sin(theta3-alpha3).*cos(theta3);
    udop3 = q3./(2*pi*r3).*cos(theta3-alpha3).*cos(theta3)-q3./...
1883 ...(2*pi*r3).*sin(theta3-alpha3).*sin(theta3);
    handles.psid3 = psidop3;
1885 handles.phid3 = phidop3;
    handles.ud3 = udop3;
1887 handles.vd3 = vdop3;
    %Vortice 3
1889 psivor3 = -q3./(2*pi).*log(r3);
    phivor3 = q3./(2*pi).*theta3;
1891 uvor3 = -q3./(2*pi*r3).*sin(theta3);
    vvor3 = q3./(2*pi*r3).*cos(theta3);
1893 handles.psiv3 = psivor3;
    handles.phiv3 = phivor3;
1895 handles.uv3 = uvor3;
    handles.vv3 = vvor3;
```

```
1897 % Set the current data value with popupmenu_3.
     str = get(handles.popupmenu3, 'String');
1899 val3 = get(handles.popupmenu3,'Value');
     % Set current data to the selected data set.
1901 switch str(val3,:)
     case 'Corrente␣uniforme'
1903 handles.psi3 = handles.psic3;
     handles.phi3 = handles.phic3;
1905 handles.u3 = handles.uc3;
     handles.v3 = handles.vc3;
1907 case 'Sorgente␣␣␣␣␣␣␣␣␣'
     handles.psi3 = handles.psis3;
1909 handles.phi3 = handles.phis3;
     handles.u3 = handles.us3;
1911 handles.v3 = handles.vs3;
     case 'Pozzo␣␣␣␣␣␣␣␣␣␣␣␣'
1913 handles.psi3 = handles.psip3;
     handles.phi3 = handles.phip3;
1915 handles.u3 = handles.up3;
     handles.v3 = handles.vp3;
1917 case 'Doppietta␣␣␣␣␣␣␣␣'
     handles.psi3 = handles.psid3;
1919 handles.phi3 = handles.phid3;
     handles.u3 = handles.ud3;
1921 handles.v3 = handles.vd3;
     case 'Vortice␣␣␣␣␣␣␣␣␣␣'
1923 handles.psi3 = handles.psiv3;
     handles.phi3 = handles.phiv3;
1925 handles.u3 = handles.uv3;
     handles.v3 = handles.vv3;
1927 end
     %Set the final current data
1929 handles.psi = handles.psi1 + handles.psi2 + handles.psi3 +...
     ...handles.psi4 + handles.psi5;
1931 handles.phi = handles.phi1 + handles.phi2 + handles.phi3 +...
     ...handles.phi4 + handles.phi5;
1933 handles.u = handles.u1 + handles.u2 + handles.u3 +...
     ...handles.u4 + handles.u5;
1935 handles.v = handles.v1 + handles.v2 + handles.v3 +...
     ...handles.v4 + handles.v5;
1937 % Save the handles structure.
     guidata(hObject,handles)
1939 % Choose default command line output
     handles.output = hObject;
1941 % Update handles structure
     guidata(hObject, handles);
1943
     % --- Executes during object creation, after setting all properties.
1945 function y3_CreateFcn(hObject, eventdata, handles)
```

```
1947  if ispc && isequal(get(hObject,'BackgroundColor'),...
      ...get(0,'defaultUicontrolBackgroundColor'))
1949  set(hObject,'BackgroundColor','white');
      end
1951
      function q3_Callback(hObject, eventdata, handles)
1953  handles.X3 = str2double(get(handles.x3,'String'));
      handles.Y3 = str2double(get(handles.y3,'String'));
1955  handles.Q3 = str2double(get(handles.q3,'String'));
      handles.ang3 = str2double(get(handles.alpha3,'String'));
1957  x3 = handles.X3;
      y3 = handles.Y3;
1959  [x,y] = meshgrid([-1:0.01:x3-0.01, x3+0.01:.01:1],...
      ...[-1:0.01:y3-0.01, y3+0.01:.01:1]);
1961  handles.x = x;
      handles.y = y;
1963  q3 = handles.Q3;
      alpha3 = handles.ang3*pi/180;
1965  r3 = sqrt((x-x3).^2+(y-y3).^2);
      theta3 = atan2(y-y3,x-x3);
1967  handles.r3 = r3;
      handles.theta3 = theta3;
1969  %Corrente uniforme 3
      uc3 = q3*cos(alpha3);
1971  vc3 = q3*sin(alpha3);
      Uc3 = uc3*ones(length(x),length(x));
1973  Vc3 = vc3*ones(length(x),length(x));
      psicor3 = -Vc3.*x+Uc3.*y;
1975  phicor3 = Uc3.*x+Vc3.*y;
      handles.psic3 = psicor3;
1977  handles.phic3 = phicor3;
      handles.uc3 = Uc3;
1979  handles.vc3 = Vc3;
      %Sorgente 3
1981  psisorg3 = q3./(2*pi).*theta3;
      phisorg3 = q3./(2*pi).*log(r3);
1983  usorg3 = cos(theta3).*q3./(2*pi*r3);
      vsorg3 = sin(theta3).*q3./(2*pi*r3);
1985  handles.psis3 = psisorg3;
      handles.phis3 = phisorg3;
1987  handles.us3 = usorg3;
      handles.vs3 = vsorg3;
1989  %Pozzo 3
      psipoz3 = -q3./(2*pi).*theta3;
1991  phipoz3 = -q3./(2*pi).*log(r3);
      upoz3 = -cos(theta3).*q3./(2*pi*r3);
1993  vpoz3 = -sin(theta3).*q3./(2*pi*r3);
      handles.psip3 = psipoz3;
```

```
handles.phip3 = phipoz3;
handles.up3 = upoz3;
handles.vp3 = vpoz3;
%Doppietta 3
psidop3 = sin(theta3-alpha3).*q3./(2*pi*r3);
phidop3 = -cos(theta3-alpha3).*q3./(2*pi*r3);
vdop3 = q3./(2*pi*r3).*cos(theta3-alpha3).*sin(theta3)+q3./...
...(2*pi*r3).*sin(theta3-alpha3).*cos(theta3);
udop3 = q3./(2*pi*r3).*cos(theta3-alpha3).*cos(theta3)-q3./...
...(2*pi*r3).*sin(theta3-alpha3).*sin(theta3);
handles.psid3 = psidop3;
handles.phid3 = phidop3;
handles.ud3 = udop3;
handles.vd3 = vdop3;
%Vortice 3
psivor3 = -q3./(2*pi).*log(r3);
phivor3 = q3./(2*pi).*theta3;
uvor3 = -q3./(2*pi*r3).*sin(theta3);
vvor3 = q3./(2*pi*r3).*cos(theta3);
handles.psiv3 = psivor3;
handles.phiv3 = phivor3;
handles.uv3 = uvor3;
handles.vv3 = vvor3;
% Set the current data value with popupmenu_3.
str = get(handles.popupmenu3, 'String');
val3 = get(handles.popupmenu3,'Value');
% Set current data to the selected data set.
switch str(val3,:)
case 'Corrente␣uniforme'
handles.psi3 = handles.psic3;
handles.phi3 = handles.phic3;
handles.u3 = handles.uc3;
handles.v3 = handles.vc3;
case 'Sorgente␣␣␣␣␣␣␣␣␣'
handles.psi3 = handles.psis3;
handles.phi3 = handles.phis3;
handles.u3 = handles.us3;
handles.v3 = handles.vs3;
case 'Pozzo␣␣␣␣␣␣␣␣␣␣␣␣'
handles.psi3 = handles.psip3;
handles.phi3 = handles.phip3;
handles.u3 = handles.up3;
handles.v3 = handles.vp3;
case 'Doppietta␣␣␣␣␣␣␣␣'
handles.psi3 = handles.psid3;
handles.phi3 = handles.phid3;
handles.u3 = handles.ud3;
handles.v3 = handles.vd3;
case 'Vortice␣␣␣␣␣␣␣␣␣␣'
```

```
handles.psi3 = handles.psiv3;
handles.phi3 = handles.phiv3;
handles.u3 = handles.uv3;
handles.v3 = handles.vv3;
end
%Set the final current data
handles.psi = handles.psi1 + handles.psi2 + handles.psi3 +...
...handles.psi4 + handles.psi5;
handles.phi = handles.phi1 + handles.phi2 + handles.phi3 +...
...handles.phi4 + handles.phi5;
handles.u = handles.u1 + handles.u2 + handles.u3 +...
...handles.u4 + handles.u5;
handles.v = handles.v1 + handles.v2 + handles.v3 +...
...handles.v4 + handles.v5;
% Save the handles structure.
guidata(hObject,handles)
% Choose default command line output
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function q3_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
...get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function alpha3_Callback(hObject, eventdata, handles)
handles.X3 = str2double(get(handles.x3,'String'));
handles.Y3 = str2double(get(handles.y3,'String'));
handles.Q3 = str2double(get(handles.q3,'String'));
handles.ang3 = str2double(get(handles.alpha3,'String'));
x3 = handles.X3;
y3 = handles.Y3;
[x,y] = meshgrid([-1:0.01:x3-0.01, x3+0.01:.01:1],...
...[-1:0.01:y3-0.01, y3+0.01:.01:1]);
handles.x = x;
handles.y = y;
q3 = handles.Q3;
alpha3 = handles.ang3*pi/180;
r3 = sqrt((x-x3).^2+(y-y3).^2);
theta3 = atan2(y-y3,x-x3);
handles.r3 = r3;
handles.theta3 = theta3;
%Corrente uniforme 3
uc3 = q3*cos(alpha3);
vc3 = q3*sin(alpha3);
```

```
2093 Uc3 = uc3*ones(length(x),length(x));
     Vc3 = vc3*ones(length(x),length(x));
2095 psicor3 = -Vc3.*x+Uc3.*y;
     phicor3 = Uc3.*x+Vc3.*y;
2097 handles.psic3 = psicor3;
     handles.phic3 = phicor3;
2099 handles.uc3 = Uc3;
     handles.vc3 = Vc3;
2101 %Sorgente 3
     psisorg3 = q3./(2*pi).*theta3;
2103 phisorg3 = q3./(2*pi).*log(r3);
     usorg3 = cos(theta3).*q3./(2*pi*r3);
2105 vsorg3 = sin(theta3).*q3./(2*pi*r3);
     handles.psis3 = psisorg3;
2107 handles.phis3 = phisorg3;
     handles.us3 = usorg3;
2109 handles.vs3 = vsorg3;
     %Pozzo 3
2111 psipoz3 = -q3./(2*pi).*theta3;
     phipoz3 = -q3./(2*pi).*log(r3);
2113 upoz3 = -cos(theta3).*q3./(2*pi*r3);
     vpoz3 = -sin(theta3).*q3./(2*pi*r3);
2115 handles.psip3 = psipoz3;
     handles.phip3 = phipoz3;
2117 handles.up3 = upoz3;
     handles.vp3 = vpoz3;
2119 %Doppietta 3
     psidop3 = sin(theta3-alpha3).*q3./(2*pi*r3);
2121 phidop3 = -cos(theta3-alpha3).*q3./(2*pi*r3);
     vdop3 = q3./(2*pi*r3).*cos(theta3-alpha3).*sin(theta3)+q3./...
2123 ...(2*pi*r3).*sin(theta3-alpha3).*cos(theta3);
     udop3 = q3./(2*pi*r3).*cos(theta3-alpha3).*cos(theta3)-q3./...
2125 ...(2*pi*r3).*sin(theta3-alpha3).*sin(theta3);
     handles.psid3 = psidop3;
2127 handles.phid3 = phidop3;
     handles.ud3 = udop3;
2129 handles.vd3 = vdop3;
     %Vortice 3
2131 psivor3 = -q3./(2*pi).*log(r3);
     phivor3 = q3./(2*pi).*theta3;
2133 uvor3 = -q3./(2*pi*r3).*sin(theta3);
     vvor3 = q3./(2*pi*r3).*cos(theta3);
2135 handles.psiv3 = psivor3;
     handles.phiv3 = phivor3;
2137 handles.uv3 = uvor3;
     handles.vv3 = vvor3;
2139 % Set the current data value with popupmenu_3.
     str = get(handles.popupmenu3, 'String');
2141 val3 = get(handles.popupmenu3,'Value');
```

```
    % Set current data to the selected data set.
2143 switch str(val3,:)
    case 'Corrente␣uniforme'
2145 handles.psi3 = handles.psic3;
    handles.phi3 = handles.phic3;
2147 handles.u3 = handles.uc3;
    handles.v3 = handles.vc3;
2149 case 'Sorgente␣␣␣␣␣␣␣␣␣'
    handles.psi3 = handles.psis3;
2151 handles.phi3 = handles.phis3;
    handles.u3 = handles.us3;
2153 handles.v3 = handles.vs3;
    case 'Pozzo␣␣␣␣␣␣␣␣␣␣␣␣'
2155 handles.psi3 = handles.psip3;
    handles.phi3 = handles.phip3;
2157 handles.u3 = handles.up3;
    handles.v3 = handles.vp3;
2159 case 'Doppietta␣␣␣␣␣␣␣␣'
    handles.psi3 = handles.psid3;
2161 handles.phi3 = handles.phid3;
    handles.u3 = handles.ud3;
2163 handles.v3 = handles.vd3;
    case 'Vortice␣␣␣␣␣␣␣␣␣␣'
2165 handles.psi3 = handles.psiv3;
    handles.phi3 = handles.phiv3;
2167 handles.u3 = handles.uv3;
    handles.v3 = handles.vv3;
2169 end
    %Set the final current data
2171 handles.psi = handles.psi1 + handles.psi2 + handles.psi3 +...
    ...handles.psi4 + handles.psi5;
2173 handles.phi = handles.phi1 + handles.phi2 + handles.phi3 +...
    ...handles.phi4 + handles.phi5;
2175 handles.u = handles.u1 + handles.u2 + handles.u3 +...
    ...handles.u4 + handles.u5;
2177 handles.v = handles.v1 + handles.v2 + handles.v3 +...
    ...handles.v4 + handles.v5;
2179 % Save the handles structure.
    guidata(hObject,handles)
2181 % Choose default command line output
    handles.output = hObject;
2183 % Update handles structure
    guidata(hObject, handles);
2185
    % --- Executes during object creation, after setting all properties.
2187 function alpha3_CreateFcn(hObject, eventdata, handles)

2189 if ispc && isequal(get(hObject,'BackgroundColor'),...
    ...get(0,'defaultUicontrolBackgroundColor'))
```

```
2191  set(hObject,'BackgroundColor','white');
      end
2193
      function x4_Callback(hObject, eventdata, handles)
2195  handles.X4 = str2double(get(handles.x4,'String'));
      handles.Y4 = str2double(get(handles.y4,'String'));
2197  handles.Q4 = str2double(get(handles.q4,'String'));
      handles.ang4 = str2double(get(handles.alpha4,'String'));
2199  x4 = handles.X4;
      y4= handles.Y4;
2201  [x,y] = meshgrid([-1:0.01:x4-0.01, x4+0.01:.01:1],...
      ...[-1:0.01:y4-0.01, y4+0.01:.01:1]);
2203  handles.x = x;
      handles.y = y;
2205  q4 = handles.Q4;
      alpha4 = handles.ang4*pi/180;
2207  r4 = sqrt((x-x4).^2+(y-y4).^2);
      theta4 = atan2(y-y4,x-x4);
2209  handles.r4 = r4;
      handles.theta4 = theta4;
2211  %Corrente uniforme 4
      uc4 = q4*cos(alpha4);
2213  vc4 = q4*sin(alpha4);
      Uc4 = uc4*ones(length(x),length(x));
2215  Vc4 = vc4*ones(length(x),length(x));
      psicor4 = -Vc4.*x+Uc4.*y;
2217  phicor4 = Uc4.*x+Vc4.*y;
      handles.psic4 = psicor4;
2219  handles.phic4 = phicor4;
      handles.uc4 = Uc4;
2221  handles.vc4 = Vc4;
      %Sorgente 4
2223  psisorg4 = q4./(2*pi).*theta4;
      phisorg4 = q4./(2*pi).*log(r4);
2225  usorg4 = cos(theta4).*q4./(2*pi*r4);
      vsorg4 = sin(theta4).*q4./(2*pi*r4);
2227  handles.psis4 = psisorg4;
      handles.phis4 = phisorg4;
2229  handles.us4 = usorg4;
      handles.vs4 = vsorg4;
2231  %Pozzo 4
      psipoz4 = -q4./(2*pi).*theta4;
2233  phipoz4 = -q4./(2*pi).*log(r4);
      upoz4 = -cos(theta4).*(-q4)./(2*pi*r4);
2235  vpoz4 = -sin(theta4).*(-q4)./(2*pi*r4);
      handles.psip4 = psipoz4;
2237  handles.phip4 = phipoz4;
      handles.up4 = upoz4;
2239  handles.vp4 = vpoz4;
```

```
    %Doppietta 4
2241 psidop4 = sin(theta4-alpha4).*q4./(2*pi*r4);
    phidop4 = -cos(theta4-alpha4).*q4./(2*pi*r4);
2243 vdop4 = q4./(2*pi*r4).*cos(theta4-alpha4).*sin(theta4)+q4./...
    ...(2*pi*r4).*sin(theta4-alpha4).*cos(theta4);
2245 udop4 = q4./(2*pi*r4).*cos(theta4-alpha4).*cos(theta4)-q4./...
    ...(2*pi*r4).*sin(theta4-alpha4).*sin(theta4);
2247 handles.psid4 = psidop4;
    handles.phid4 = phidop4;
2249 handles.ud4 = udop4;
    handles.vd4 = vdop4;
2251 %Vortice 4
    psivor4 = -q4./(2*pi).*log(r4);
2253 phivor4 = q4./(2*pi).*theta4;
    uvor4 = -q4./(2*pi*r4).*sin(theta4);
2255 vvor4 = q4./(2*pi*r4).*cos(theta4);
    handles.psiv4 = psivor4;
2257 handles.phiv4 = phivor4;
    handles.uv4 = uvor4;
2259 handles.vv4 = vvor4;
    % Set the current data value with popupmenu_4.
2261 str = get(handles.popupmenu4, 'String');
    val4 = get(handles.popupmenu4,'Value');
2263 % Set current data to the selected data set.
    switch str(val4,:)
2265 case 'Corrente␣uniforme'
    handles.psi4 = handles.psic4;
2267 handles.phi4 = handles.phic4;
    handles.u4 = handles.uc4;
2269 handles.v4 = handles.vc4;
    case 'Sorgente␣␣␣␣␣␣␣␣␣'
2271 handles.psi4 = handles.psis4;
    handles.phi4 = handles.phis4;
2273 handles.u4 = handles.us4;
    handles.v4 = handles.vs4;
2275 case 'Pozzo␣␣␣␣␣␣␣␣␣␣␣␣'
    handles.psi4 = handles.psip4;
2277 handles.phi4 = handles.phip4;
    handles.u4 = handles.up4;
2279 handles.v4 = handles.vp4;
    case 'Doppietta␣␣␣␣␣␣␣␣'
2281 handles.psi4 = handles.psid4;
    handles.phi4 = handles.phid4;
2283 handles.u4 = handles.ud4;
    handles.v4 = handles.vd4;
2285 case 'Vortice␣␣␣␣␣␣␣␣␣␣'
    handles.psi4 = handles.psiv4;
2287 handles.phi4 = handles.phiv4;
    handles.u4 = handles.uv4;
```

```
handles.v4 = handles.vv4;
end
%Set the final current data
handles.psi = handles.psi1 + handles.psi2 + handles.psi3 +...
...handles.psi4 + handles.psi5;
handles.phi = handles.phi1 + handles.phi2 + handles.phi3 +...
...handles.phi4 + handles.phi5;
handles.u = handles.u1 + handles.u2 + handles.u3 +...
...handles.u4 + handles.u5;
handles.v = handles.v1 + handles.v2 + handles.v3 +...
...handles.v4 + handles.v5;
% Save the handles structure.
guidata(hObject,handles)
% Choose default command line output
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function x4_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
...get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function y4_Callback(hObject, eventdata, handles)
handles.X4 = str2double(get(handles.x4,'String'));
handles.Y4 = str2double(get(handles.y4,'String'));
handles.Q4 = str2double(get(handles.q4,'String'));
handles.ang4 = str2double(get(handles.alpha4,'String'));
x4 = handles.X4;
y4= handles.Y4;
[x,y] = meshgrid([-1:0.01:x4-0.01, x4+0.01:.01:1],...
...[-1:0.01:y4-0.01, y4+0.01:.01:1]);
handles.x = x;
handles.y = y;
q4 = handles.Q4;
alpha4 = handles.ang4*pi/180;
r4 = sqrt((x-x4).^2+(y-y4).^2);
theta4 = atan2(y-y4,x-x4);
handles.r4 = r4;
handles.theta4 = theta4;
%Corrente uniforme 4
uc4 = q4*cos(alpha4);
vc4 = q4*sin(alpha4);
Uc4 = uc4*ones(length(x),length(x));
Vc4 = vc4*ones(length(x),length(x));
psicor4 = -Vc4.*x+Uc4.*y;
```

```
    phicor4 = Uc4.*x+Vc4.*y;
2339 handles.psic4 = psicor4;
    handles.phic4 = phicor4;
2341 handles.uc4 = Uc4;
    handles.vc4 = Vc4;
2343 %Sorgente 4
    psisorg4 = q4./(2*pi).*theta4;
2345 phisorg4 = q4./(2*pi).*log(r4);
    usorg4 = cos(theta4).*q4./(2*pi*r4);
2347 vsorg4 = sin(theta4).*q4./(2*pi*r4);
    handles.psis4 = psisorg4;
2349 handles.phis4 = phisorg4;
    handles.us4 = usorg4;
2351 handles.vs4 = vsorg4;
    %Pozzo 4
2353 psipoz4 = -q4./(2*pi).*theta4;
    phipoz4 = -q4./(2*pi).*log(r4);
2355 upoz4 = -cos(theta4).*(-q4)./(2*pi*r4);
    vpoz4 = -sin(theta4).*(-q4)./(2*pi*r4);
2357 handles.psip4 = psipoz4;
    handles.phip4 = phipoz4;
2359 handles.up4 = upoz4;
    handles.vp4 = vpoz4;
2361 %Doppietta 4
    psidop4 = sin(theta4-alpha4).*q4./(2*pi*r4);
2363 phidop4 = -cos(theta4-alpha4).*q4./(2*pi*r4);
    vdop4 = q4./(2*pi*r4).*cos(theta4-alpha4).*sin(theta4)+q4./...
2365 ...(2*pi*r4).*sin(theta4-alpha4).*cos(theta4);
    udop4 = q4./(2*pi*r4).*cos(theta4-alpha4).*cos(theta4)-q4./...
2367 ...(2*pi*r4).*sin(theta4-alpha4).*sin(theta4);
    handles.psid4 = psidop4;
2369 handles.phid4 = phidop4;
    handles.ud4 = udop4;
2371 handles.vd4 = vdop4;
    %Vortice 4
2373 psivor4 = -q4./(2*pi).*log(r4);
    phivor4 = q4./(2*pi).*theta4;
2375 uvor4 = -q4./(2*pi*r4).*sin(theta4);
    vvor4 = q4./(2*pi*r4).*cos(theta4);
2377 handles.psiv4 = psivor4;
    handles.phiv4 = phivor4;
2379 handles.uv4 = uvor4;
    handles.vv4 = vvor4;
2381 % Set the current data value with popupmenu_4.
    str = get(handles.popupmenu4, 'String');
2383 val4 = get(handles.popupmenu4,'Value');
    % Set current data to the selected data set.
2385 switch str(val4,:)
    case 'Corrente␣uniforme'
```

```
2387  handles.psi4 = handles.psic4;
      handles.phi4 = handles.phic4;
2389  handles.u4 = handles.uc4;
      handles.v4 = handles.vc4;
2391  case 'Sorgente          '
      handles.psi4 = handles.psis4;
2393  handles.phi4 = handles.phis4;
      handles.u4 = handles.us4;
2395  handles.v4 = handles.vs4;
      case 'Pozzo              '
2397  handles.psi4 = handles.psip4;
      handles.phi4 = handles.phip4;
2399  handles.u4 = handles.up4;
      handles.v4 = handles.vp4;
2401  case 'Doppietta         '
      handles.psi4 = handles.psid4;
2403  handles.phi4 = handles.phid4;
      handles.u4 = handles.ud4;
2405  handles.v4 = handles.vd4;
      case 'Vortice           '
2407  handles.psi4 = handles.psiv4;
      handles.phi4 = handles.phiv4;
2409  handles.u4 = handles.uv4;
      handles.v4 = handles.vv4;
2411  end
      %Set the final current data
2413  handles.psi = handles.psi1 + handles.psi2 + handles.psi3 +...
      ...handles.psi4 + handles.psi5;
2415  handles.phi = handles.phi1 + handles.phi2 + handles.phi3 +...
      ...handles.phi4 + handles.phi5;
2417  handles.u = handles.u1 + handles.u2 + handles.u3 +...
      ...handles.u4 + handles.u5;
2419  handles.v = handles.v1 + handles.v2 + handles.v3 +...
      ...handles.v4 + handles.v5;
2421  % Save the handles structure.
      guidata(hObject,handles)
2423  % Choose default command line output
      handles.output = hObject;
2425  % Update handles structure
      guidata(hObject, handles);
2427
      % --- Executes during object creation, after setting all properties.
2429  function y4_CreateFcn(hObject, eventdata, handles)

2431  if ispc && isequal(get(hObject,'BackgroundColor'),...
      ...get(0,'defaultUicontrolBackgroundColor'))
2433  set(hObject,'BackgroundColor','white');
      end
2435
```

```
function q4_Callback(hObject, eventdata, handles)
handles.X4 = str2double(get(handles.x4,'String'));
handles.Y4 = str2double(get(handles.y4,'String'));
handles.Q4 = str2double(get(handles.q4,'String'));
handles.ang4 = str2double(get(handles.alpha4,'String'));
x4 = handles.X4;
y4= handles.Y4;
[x,y] = meshgrid([-1:0.01:x4-0.01, x4+0.01:.01:1],...
...[-1:0.01:y4-0.01, y4+0.01:.01:1]);
handles.x = x;
handles.y = y;
q4 = handles.Q4;
alpha4 = handles.ang4*pi/180;
r4 = sqrt((x-x4).^2+(y-y4).^2);
theta4 = atan2(y-y4,x-x4);
handles.r4 = r4;
handles.theta4 = theta4;
%Corrente uniforme 4
uc4 = q4*cos(alpha4);
vc4 = q4*sin(alpha4);
Uc4 = uc4*ones(length(x),length(x));
Vc4 = vc4*ones(length(x),length(x));
psicor4 = -Vc4.*x+Uc4.*y;
phicor4 = Uc4.*x+Vc4.*y;
handles.psic4 = psicor4;
handles.phic4 = phicor4;
handles.uc4 = Uc4;
handles.vc4 = Vc4;
%Sorgente 4
psisorg4 = q4./(2*pi).*theta4;
phisorg4 = q4./(2*pi).*log(r4);
usorg4 = cos(theta4).*q4./(2*pi*r4);
vsorg4 = sin(theta4).*q4./(2*pi*r4);
handles.psis4 = psisorg4;
handles.phis4 = phisorg4;
handles.us4 = usorg4;
handles.vs4 = vsorg4;
%Pozzo 4
psipoz4 = -q4./(2*pi).*theta4;
phipoz4 = -q4./(2*pi).*log(r4);
upoz4 = -cos(theta4).*(-q4)./(2*pi*r4);
vpoz4 = -sin(theta4).*(-q4)./(2*pi*r4);
handles.psip4 = psipoz4;
handles.phip4 = phipoz4;
handles.up4 = upoz4;
handles.vp4 = vpoz4;
%Doppietta 4
psidop4 = sin(theta4-alpha4).*q4./(2*pi*r4);
phidop4 = -cos(theta4-alpha4).*q4./(2*pi*r4);
```

```
2485 vdop4 = q4./(2*pi*r4).*cos(theta4-alpha4).*sin(theta4)+q4./...
     ...(2*pi*r4).*sin(theta4-alpha4).*cos(theta4);
2487 udop4 = q4./(2*pi*r4).*cos(theta4-alpha4).*cos(theta4)-q4./...
     ...(2*pi*r4).*sin(theta4-alpha4).*sin(theta4);
2489 handles.psid4 = psidop4;
     handles.phid4 = phidop4;
2491 handles.ud4 = udop4;
     handles.vd4 = vdop4;
2493 %Vortice 4
     psivor4 = -q4./(2*pi).*log(r4);
2495 phivor4 = q4./(2*pi).*theta4;
     uvor4 = -q4./(2*pi*r4).*sin(theta4);
2497 vvor4 = q4./(2*pi*r4).*cos(theta4);
     handles.psiv4 = psivor4;
2499 handles.phiv4 = phivor4;
     handles.uv4 = uvor4;
2501 handles.vv4 = vvor4;
     % Set the current data value with popupmenu_4.
2503 str = get(handles.popupmenu4, 'String');
     val4 = get(handles.popupmenu4,'Value');
2505 % Set current data to the selected data set.
     switch str(val4,:)
2507 case 'Corrente␣uniforme'
     handles.psi4 = handles.psic4;
2509 handles.phi4 = handles.phic4;
     handles.u4 = handles.uc4;
2511 handles.v4 = handles.vc4;
     case 'Sorgente␣␣␣␣␣␣␣␣␣'
2513 handles.psi4 = handles.psis4;
     handles.phi4 = handles.phis4;
2515 handles.u4 = handles.us4;
     handles.v4 = handles.vs4;
2517 case 'Pozzo␣␣␣␣␣␣␣␣␣␣␣␣'
     handles.psi4 = handles.psip4;
2519 handles.phi4 = handles.phip4;
     handles.u4 = handles.up4;
2521 handles.v4 = handles.vp4;
     case 'Doppietta␣␣␣␣␣␣␣␣'
2523 handles.psi4 = handles.psid4;
     handles.phi4 = handles.phid4;
2525 handles.u4 = handles.ud4;
     handles.v4 = handles.vd4;
2527 case 'Vortice␣␣␣␣␣␣␣␣␣␣'
     handles.psi4 = handles.psiv4;
2529 handles.phi4 = handles.phiv4;
     handles.u4 = handles.uv4;
2531 handles.v4 = handles.vv4;
     end
2533 %Set the final current data
```

```
    handles.psi = handles.psi1 + handles.psi2 + handles.psi3 +...
2535 ...handles.psi4 + handles.psi5;
    handles.phi = handles.phi1 + handles.phi2 + handles.phi3 +...
2537 ...handles.phi4 + handles.phi5;
    handles.u = handles.u1 + handles.u2 + handles.u3 +...
2539 ...handles.u4 + handles.u5;
    handles.v = handles.v1 + handles.v2 + handles.v3 +...
2541 ...handles.v4 + handles.v5;
    % Save the handles structure.
2543 guidata(hObject,handles)
    % Choose default command line output
2545 handles.output = hObject;
    % Update handles structure
2547 guidata(hObject, handles);

2549 % --- Executes during object creation, after setting all properties.
    function q4_CreateFcn(hObject, eventdata, handles)
2551
    if ispc && isequal(get(hObject,'BackgroundColor'),...
2553 ...get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
2555 end

2557 function alpha4_Callback(hObject, eventdata, handles)
    handles.X4 = str2double(get(handles.x4,'String'));
2559 handles.Y4 = str2double(get(handles.y4,'String'));
    handles.Q4 = str2double(get(handles.q4,'String'));
2561 handles.ang4 = str2double(get(handles.alpha4,'String'));
    x4 = handles.X4;
2563 y4= handles.Y4;
    [x,y] = meshgrid([-1:0.01:x4-0.01, x4+0.01:.01:1],...
2565 ...[-1:0.01:y4-0.01, y4+0.01:.01:1]);
    handles.x = x;
2567 handles.y = y;
    q4 = handles.Q4;
2569 alpha4 = handles.ang4*pi/180;
    r4 = sqrt((x-x4).^2+(y-y4).^2);
2571 theta4 = atan2(y-y4,x-x4);
    handles.r4 = r4;
2573 handles.theta4 = theta4;
    %Corrente uniforme 4
2575 uc4 = q4*cos(alpha4);
    vc4 = q4*sin(alpha4);
2577 Uc4 = uc4*ones(length(x),length(x));
    Vc4 = vc4*ones(length(x),length(x));
2579 psicor4 = -Vc4.*x+Uc4.*y;
    phicor4 = Uc4.*x+Vc4.*y;
2581 handles.psic4 = psicor4;
    handles.phic4 = phicor4;
```

```
handles.uc4 = Uc4;
handles.vc4 = Vc4;
%Sorgente 4
psisorg4 = q4./(2*pi).*theta4;
phisorg4 = q4./(2*pi).*log(r4);
usorg4 = cos(theta4).*q4./(2*pi*r4);
vsorg4 = sin(theta4).*q4./(2*pi*r4);
handles.psis4 = psisorg4;
handles.phis4 = phisorg4;
handles.us4 = usorg4;
handles.vs4 = vsorg4;
%Pozzo 4
psipoz4 = -q4./(2*pi).*theta4;
phipoz4 = -q4./(2*pi).*log(r4);
upoz4 = -cos(theta4).*(-q4)./(2*pi*r4);
vpoz4 = -sin(theta4).*(-q4)./(2*pi*r4);
handles.psip4 = psipoz4;
handles.phip4 = phipoz4;
handles.up4 = upoz4;
handles.vp4 = vpoz4;
%Doppietta 4
psidop4 = sin(theta4-alpha4).*q4./(2*pi*r4);
phidop4 = -cos(theta4-alpha4).*q4./(2*pi*r4);
vdop4 = q4./(2*pi*r4).*cos(theta4-alpha4).*sin(theta4)+q4./...
...(2*pi*r4).*sin(theta4-alpha4).*cos(theta4);
udop4 = q4./(2*pi*r4).*cos(theta4-alpha4).*cos(theta4)-q4./...
...(2*pi*r4).*sin(theta4-alpha4).*sin(theta4);
handles.psid4 = psidop4;
handles.phid4 = phidop4;
handles.ud4 = udop4;
handles.vd4 = vdop4;
%Vortice 4
psivor4 = -q4./(2*pi).*log(r4);
phivor4 = q4./(2*pi).*theta4;
uvor4 = -q4./(2*pi*r4).*sin(theta4);
vvor4 = q4./(2*pi*r4).*cos(theta4);
handles.psiv4 = psivor4;
handles.phiv4 = phivor4;
handles.uv4 = uvor4;
handles.vv4 = vvor4;
% Set the current data value with popupmenu_4.
str = get(handles.popupmenu4, 'String');
val4 = get(handles.popupmenu4,'Value');
% Set current data to the selected data set.
switch str(val4,:)
case 'Corrente␣uniforme'
handles.psi4 = handles.psic4;
handles.phi4 = handles.phic4;
handles.u4 = handles.uc4;
```

```
    handles.v4 = handles.vc4;
2633 case 'Sorgente␣␣␣␣␣␣␣␣␣'
    handles.psi4 = handles.psis4;
2635 handles.phi4 = handles.phis4;
    handles.u4 = handles.us4;
2637 handles.v4 = handles.vs4;
    case 'Pozzo␣␣␣␣␣␣␣␣␣␣␣␣'
2639 handles.psi4 = handles.psip4;
    handles.phi4 = handles.phip4;
2641 handles.u4 = handles.up4;
    handles.v4 = handles.vp4;
2643 case 'Doppietta␣␣␣␣␣␣␣␣'
    handles.psi4 = handles.psid4;
2645 handles.phi4 = handles.phid4;
    handles.u4 = handles.ud4;
2647 handles.v4 = handles.vd4;
    case 'Vortice␣␣␣␣␣␣␣␣␣␣'
2649 handles.psi4 = handles.psiv4;
    handles.phi4 = handles.phiv4;
2651 handles.u4 = handles.uv4;
    handles.v4 = handles.vv4;
2653 end
    %Set the final current data
2655 handles.psi = handles.psi1 + handles.psi2 + handles.psi3 +...
    ...handles.psi4 + handles.psi5;
2657 handles.phi = handles.phi1 + handles.phi2 + handles.phi3 +...
    ...handles.phi4 + handles.phi5;
2659 handles.u = handles.u1 + handles.u2 + handles.u3 +...
    ...handles.u4 + handles.u5;
2661 handles.v = handles.v1 + handles.v2 + handles.v3 +...
    ...handles.v4 + handles.v5;
2663 % Save the handles structure.
    guidata(hObject,handles)
2665 % Choose default command line output
    handles.output = hObject;
2667 % Update handles structure
    guidata(hObject, handles);
2669
    % --- Executes during object creation, after setting all properties.
2671 function alpha4_CreateFcn(hObject, eventdata, handles)

2673 if ispc && isequal(get(hObject,'BackgroundColor'),...
    ...get(0,'defaultUicontrolBackgroundColor'))
2675 set(hObject,'BackgroundColor','white');
    end
2677
    function x5_Callback(hObject, eventdata, handles)
2679 handles.X5 = str2double(get(handles.x5,'String'));
    handles.Y5 = str2double(get(handles.y5,'String'));
```

```
handles.Q5 = str2double(get(handles.q5,'String'));
handles.ang5 = str2double(get(handles.alpha5,'String'));
x5 = handles.X5;
y5= handles.Y5;
[x,y] = meshgrid([-1:0.01:x5-0.01, x5+0.01:.01:1],...
...[-1:0.01:y5-0.01, y5+0.01:.01:1]);
handles.x = x;
handles.y = y;
q5 = handles.Q5;
alpha5 = handles.ang5*pi/180;
r5 = sqrt((x-x5).^2+(y-y5).^2);
theta5 = atan2(y-y5,x-x5);
handles.r4 = r5;
handles.theta4 = theta5;
%Corrente uniforme 5
uc5 = q5*cos(alpha5);
vc5 = q5*sin(alpha5);
Uc5 = uc5*ones(length(x),length(x));
Vc5 = vc5*ones(length(x),length(x));
psicor5 = -Vc5.*x+Uc5.*y;
phicor5 = Uc5.*x+Vc5.*y;
handles.psic5 = psicor5;
handles.phic5 = phicor5;
handles.uc5 = Uc5;
handles.vc5 = Vc5;
%Sorgente 5
psisorg5 = q5./(2*pi).*theta5;
phisorg5 = q5./(2*pi).*log(r5);
usorg5 = cos(theta5).*q5./(2*pi*r5);
vsorg5 = sin(theta5).*q5./(2*pi*r5);
handles.psis5 = psisorg5;
handles.phis5 = phisorg5;
handles.us5 = usorg5;
handles.vs5 = vsorg5;
%Pozzo 5
psipoz5 = -q5./(2*pi).*theta5;
phipoz5 = -q5./(2*pi).*log(r5);
upoz5 = -cos(theta5).*q5./(2*pi*r5);
vpoz5 = -sin(theta5).*q5./(2*pi*r5);
handles.psip5 = psipoz5;
handles.phip5 = phipoz5;
handles.up5 = upoz5;
handles.vp5 = vpoz5;
%Doppietta 5
psidop5 = sin(theta5-alpha5).*q5./(2*pi*r5);
phidop5 = -cos(theta5-alpha5).*q5./(2*pi*r5);
vdop5 = q5./(2*pi*r5).*cos(theta5-alpha5).*sin(theta5)+q5./...
...(2*pi*r5).*sin(theta5-alpha5).*cos(theta5);
udop5 = q5./(2*pi*r5).*cos(theta5-alpha5).*cos(theta5)-q5./...
```

```
    ...(2*pi*r5).*sin(theta5-alpha5).*sin(theta5);
2731 handles.psid5 = psidop5;
    handles.phid5 = phidop5;
2733 handles.ud5 = udop5;
    handles.vd5 = vdop5;
2735 %Vortice 5
    psivor5 = -q5./(2*pi).*log(r5);
2737 phivor5 = q5./(2*pi).*theta5;
    uvor5 = -q5./(2*pi*r5).*sin(theta5);
2739 vvor5 = q5./(2*pi*r5).*cos(theta5);
    handles.psiv5 = psivor5;
2741 handles.phiv5 = phivor5;
    handles.uv5 = uvor5;
2743 handles.vv5 = vvor5;
    % Set the current data value with popupmenu_5.
2745 str = get(handles.popupmenu5, 'String');
    val5 = get(handles.popupmenu5,'Value');
2747 % Set current data to the selected data set.
    switch str(val5,:)
2749 case 'Corrente␣uniforme'
    handles.psi5 = handles.psic5;
2751 handles.phi5 = handles.phic5;
    handles.u5 = handles.uc5;
2753 handles.v5 = handles.vc5;
    case 'Sorgente␣␣␣␣␣␣␣␣␣'
2755 handles.psi5 = handles.psis5;
    handles.phi5 = handles.phis5;
2757 handles.u5 = handles.us5;
    handles.v5 = handles.vs5;
2759 case 'Pozzo␣␣␣␣␣␣␣␣␣␣␣␣'
    handles.psi5 = handles.psip5;
2761 handles.phi5 = handles.phip5;
    handles.u5 = handles.up5;
2763 handles.v5 = handles.vp5;
    case 'Doppietta␣␣␣␣␣␣␣␣'
2765 handles.psi5 = handles.psid5;
    handles.phi5 = handles.phid5;
2767 handles.u5 = handles.ud5;
    handles.v5 = handles.vd5;
2769 case 'Vortice␣␣␣␣␣␣␣␣␣␣'
    handles.psi5 = handles.psiv5;
2771 handles.phi5 = handles.phiv5;
    handles.u5 = handles.uv5;
2773 handles.v5 = handles.vv5;
    end
2775 %Set the final current data
    handles.psi = handles.psi1 + handles.psi2 + handles.psi3 +...
2777 ...handles.psi4 + handles.psi5;
    handles.phi = handles.phi1 + handles.phi2 + handles.phi3 +...
```

```
...handles.phi4 + handles.phi5;
handles.u = handles.u1 + handles.u2 + handles.u3 +...
...handles.u4 + handles.u5;
handles.v = handles.v1 + handles.v2 + handles.v3 +...
...handles.v4 + handles.v5;
% Save the handles structure.
guidata(hObject,handles)
% Choose default command line output
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function x5_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
...get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function y5_Callback(hObject, eventdata, handles)
handles.X5 = str2double(get(handles.x5,'String'));
handles.Y5 = str2double(get(handles.y5,'String'));
handles.Q5 = str2double(get(handles.q5,'String'));
handles.ang5 = str2double(get(handles.alpha5,'String'));
x5 = handles.X5;
y5= handles.Y5;
[x,y] = meshgrid([-1:0.01:x5-0.01, x5+0.01:.01:1],...
...[-1:0.01:y5-0.01, y5+0.01:.01:1]);
handles.x = x;
handles.y = y;
q5 = handles.Q5;
alpha5 = handles.ang5*pi/180;
r5 = sqrt((x-x5).^2+(y-y5).^2);
theta5 = atan2(y-y5,x-x5);
handles.r4 = r5;
handles.theta4 = theta5;
%Corrente uniforme 5
uc5 = q5*cos(alpha5);
vc5 = q5*sin(alpha5);
Uc5 = uc5*ones(length(x),length(x));
Vc5 = vc5*ones(length(x),length(x));
psicor5 = -Vc5.*x+Uc5.*y;
phicor5 = Uc5.*x+Vc5.*y;
handles.psic5 = psicor5;
handles.phic5 = phicor5;
handles.uc5 = Uc5;
handles.vc5 = Vc5;
%Sorgente 5
```

```
    psisorg5 = q5./(2*pi).*theta5;
2829 phisorg5 = q5./(2*pi).*log(r5);
    usorg5 = cos(theta5).*q5./(2*pi*r5);
2831 vsorg5 = sin(theta5).*q5./(2*pi*r5);
    handles.psis5 = psisorg5;
2833 handles.phis5 = phisorg5;
    handles.us5 = usorg5;
2835 handles.vs5 = vsorg5;
    %Pozzo 5
2837 psipoz5 = -q5./(2*pi).*theta5;
    phipoz5 = -q5./(2*pi).*log(r5);
2839 upoz5 = -cos(theta5).*q5./(2*pi*r5);
    vpoz5 = -sin(theta5).*q5./(2*pi*r5);
2841 handles.psip5 = psipoz5;
    handles.phip5 = phipoz5;
2843 handles.up5 = upoz5;
    handles.vp5 = vpoz5;
2845 %Doppietta 5
    psidop5 = sin(theta5-alpha5).*q5./(2*pi*r5);
2847 phidop5 = -cos(theta5-alpha5).*q5./(2*pi*r5);
    vdop5 = q5./(2*pi*r5).*cos(theta5-alpha5).*sin(theta5)+q5./...
2849 ...(2*pi*r5).*sin(theta5-alpha5).*cos(theta5);
    udop5 = q5./(2*pi*r5).*cos(theta5-alpha5).*cos(theta5)-q5./...
2851 ...(2*pi*r5).*sin(theta5-alpha5).*sin(theta5);
    handles.psid5 = psidop5;
2853 handles.phid5 = phidop5;
    handles.ud5 = udop5;
2855 handles.vd5 = vdop5;
    %Vortice 5
2857 psivor5 = -q5./(2*pi).*log(r5);
    phivor5 = q5./(2*pi).*theta5;
2859 uvor5 = -q5./(2*pi*r5).*sin(theta5);
    vvor5 = q5./(2*pi*r5).*cos(theta5);
2861 handles.psiv5 = psivor5;
    handles.phiv5 = phivor5;
2863 handles.uv5 = uvor5;
    handles.vv5 = vvor5;
2865 % Set the current data value with popupmenu_5.
    str = get(handles.popupmenu5, 'String');
2867 val5 = get(handles.popupmenu5,'Value');
    % Set current data to the selected data set.
2869 switch str(val5,:)
    case 'Corrente␣uniforme'
2871 handles.psi5 = handles.psic5;
    handles.phi5 = handles.phic5;
2873 handles.u5 = handles.uc5;
    handles.v5 = handles.vc5;
2875 case 'Sorgente␣␣␣␣␣␣␣␣␣'
    handles.psi5 = handles.psis5;
```

```
2877 handles.phi5 = handles.phis5;
     handles.u5 = handles.us5;
2879 handles.v5 = handles.vs5;
     case 'Pozzo␣␣␣␣␣␣␣␣␣␣␣␣'
2881 handles.psi5 = handles.psip5;
     handles.phi5 = handles.phip5;
2883 handles.u5 = handles.up5;
     handles.v5 = handles.vp5;
2885 case 'Doppietta␣␣␣␣␣␣␣␣'
     handles.psi5 = handles.psid5;
2887 handles.phi5 = handles.phid5;
     handles.u5 = handles.ud5;
2889 handles.v5 = handles.vd5;
     case 'Vortice␣␣␣␣␣␣␣␣␣␣'
2891 handles.psi5 = handles.psiv5;
     handles.phi5 = handles.phiv5;
2893 handles.u5 = handles.uv5;
     handles.v5 = handles.vv5;
2895 end
     %Set the final current data
2897 handles.psi = handles.psi1 + handles.psi2 + handles.psi3 +...
     ...handles.psi4 + handles.psi5;
2899 handles.phi = handles.phi1 + handles.phi2 + handles.phi3 +...
     ...handles.phi4 + handles.phi5;
2901 handles.u = handles.u1 + handles.u2 + handles.u3 +...
     ...handles.u4 + handles.u5;
2903 handles.v = handles.v1 + handles.v2 + handles.v3 +...
     ...handles.v4 + handles.v5;
2905 % Save the handles structure.
     guidata(hObject,handles)
2907 % Choose default command line output
     handles.output = hObject;
2909 % Update handles structure
     guidata(hObject, handles);
2911
     % --- Executes during object creation, after setting all properties.
2913 function y5_CreateFcn(hObject, eventdata, handles)

2915 if ispc && isequal(get(hObject,'BackgroundColor'),...
     ...get(0,'defaultUicontrolBackgroundColor'))
2917 set(hObject,'BackgroundColor','white');
     end
2919
     function q5_Callback(hObject, eventdata, handles)
2921 handles.X5 = str2double(get(handles.x5,'String'));
     handles.Y5 = str2double(get(handles.y5,'String'));
2923 handles.Q5 = str2double(get(handles.q5,'String'));
     handles.ang5 = str2double(get(handles.alpha5,'String'));
2925 x5 = handles.X5;
```

```
    y5= handles.Y5;
2927 [x,y] = meshgrid([-1:0.01:x5-0.01, x5+0.01:.01:1],...
    ...[-1:0.01:y5-0.01, y5+0.01:.01:1]);
2929 handles.x = x;
    handles.y = y;
2931 q5 = handles.Q5;
    alpha5 = handles.ang5*pi/180;
2933 r5 = sqrt((x-x5).^2+(y-y5).^2);
    theta5 = atan2(y-y5,x-x5);
2935 handles.r4 = r5;
    handles.theta4 = theta5;
2937 %Corrente uniforme 5
    uc5 = q5*cos(alpha5);
2939 vc5 = q5*sin(alpha5);
    Uc5 = uc5*ones(length(x),length(x));
2941 Vc5 = vc5*ones(length(x),length(x));
    psicor5 = -Vc5.*x+Uc5.*y;
2943 phicor5 = Uc5.*x+Vc5.*y;
    handles.psic5 = psicor5;
2945 handles.phic5 = phicor5;
    handles.uc5 = Uc5;
2947 handles.vc5 = Vc5;
    %Sorgente 5
2949 psisorg5 = q5./(2*pi).*theta5;
    phisorg5 = q5./(2*pi).*log(r5);
2951 usorg5 = cos(theta5).*q5./(2*pi*r5);
    vsorg5 = sin(theta5).*q5./(2*pi*r5);
2953 handles.psis5 = psisorg5;
    handles.phis5 = phisorg5;
2955 handles.us5 = usorg5;
    handles.vs5 = vsorg5;
2957 %Pozzo 5
    psipoz5 = -q5./(2*pi).*theta5;
2959 phipoz5 = -q5./(2*pi).*log(r5);
    upoz5 = -cos(theta5).*q5./(2*pi*r5);
2961 vpoz5 = -sin(theta5).*q5./(2*pi*r5);
    handles.psip5 = psipoz5;
2963 handles.phip5 = phipoz5;
    handles.up5 = upoz5;
2965 handles.vp5 = vpoz5;
    %Doppietta 5
2967 psidop5 = sin(theta5-alpha5).*q5./(2*pi*r5);
    phidop5 = -cos(theta5-alpha5).*q5./(2*pi*r5);
2969 vdop5 = q5./(2*pi*r5).*cos(theta5-alpha5).*sin(theta5)+q5./...
    ...(2*pi*r5).*sin(theta5-alpha5).*cos(theta5);
2971 udop5 = q5./(2*pi*r5).*cos(theta5-alpha5).*cos(theta5)-q5./...
    ...(2*pi*r5).*sin(theta5-alpha5).*sin(theta5);
2973 handles.psid5 = psidop5;
    handles.phid5 = phidop5;
```

```
handles.ud5 = udop5;
handles.vd5 = vdop5;
%Vortice 5
psivor5 = -q5./(2*pi).*log(r5);
phivor5 = q5./(2*pi).*theta5;
uvor5 = -q5./(2*pi*r5).*sin(theta5);
vvor5 = q5./(2*pi*r5).*cos(theta5);
handles.psiv5 = psivor5;
handles.phiv5 = phivor5;
handles.uv5 = uvor5;
handles.vv5 = vvor5;
% Set the current data value with popupmenu_5.
str = get(handles.popupmenu5, 'String');
val5 = get(handles.popupmenu5,'Value');
% Set current data to the selected data set.
switch str(val5,:)
case 'Corrente uniforme'
handles.psi5 = handles.psic5;
handles.phi5 = handles.phic5;
handles.u5 = handles.uc5;
handles.v5 = handles.vc5;
case 'Sorgente          '
handles.psi5 = handles.psis5;
handles.phi5 = handles.phis5;
handles.u5 = handles.us5;
handles.v5 = handles.vs5;
case 'Pozzo             '
handles.psi5 = handles.psip5;
handles.phi5 = handles.phip5;
handles.u5 = handles.up5;
handles.v5 = handles.vp5;
case 'Doppietta         '
handles.psi5 = handles.psid5;
handles.phi5 = handles.phid5;
handles.u5 = handles.ud5;
handles.v5 = handles.vd5;
case 'Vortice           '
handles.psi5 = handles.psiv5;
handles.phi5 = handles.phiv5;
handles.u5 = handles.uv5;
handles.v5 = handles.vv5;
end
%Set the final current data
handles.psi = handles.psi1 + handles.psi2 + handles.psi3 +...
...handles.psi4 + handles.psi5;
handles.phi = handles.phi1 + handles.phi2 + handles.phi3 +...
...handles.phi4 + handles.phi5;
handles.u = handles.u1 + handles.u2 + handles.u3 +...
...handles.u4 + handles.u5;
```

```
handles.v = handles.v1 + handles.v2 + handles.v3 +...
...handles.v4 + handles.v5;
% Save the handles structure.
guidata(hObject,handles)
% Choose default command line output
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function q5_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),...
...get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

function alpha5_Callback(hObject, eventdata, handles)
handles.X5 = str2double(get(handles.x5,'String'));
handles.Y5 = str2double(get(handles.y5,'String'));
handles.Q5 = str2double(get(handles.q5,'String'));
handles.ang5 = str2double(get(handles.alpha5,'String'));
x5 = handles.X5;
y5= handles.Y5;
[x,y] = meshgrid([-1:0.01:x5-0.01, x5+0.01:.01:1],...
...[-1:0.01:y5-0.01, y5+0.01:.01:1]);
handles.x = x;
handles.y = y;
q5 = handles.Q5;
alpha5 = handles.ang5*pi/180;
r5 = sqrt((x-x5).^2+(y-y5).^2);
theta5 = atan2(y-y5,x-x5);
handles.r4 = r5;
handles.theta4 = theta5;
%Corrente uniforme 5
uc5 = q5*cos(alpha5);
vc5 = q5*sin(alpha5);
Uc5 = uc5*ones(length(x),length(x));
Vc5 = vc5*ones(length(x),length(x));
psicor5 = -Vc5.*x+Uc5.*y;
phicor5 = Uc5.*x+Vc5.*y;
handles.psic5 = psicor5;
handles.phic5 = phicor5;
handles.uc5 = Uc5;
handles.vc5 = Vc5;
%Sorgente 5
psisorg5 = q5./(2*pi).*theta5;
phisorg5 = q5./(2*pi).*log(r5);
usorg5 = cos(theta5).*q5./(2*pi*r5);
```

```
3073  vsorg5 = sin(theta5).*q5./(2*pi*r5);
      handles.psis5 = psisorg5;
3075  handles.phis5 = phisorg5;
      handles.us5 = usorg5;
3077  handles.vs5 = vsorg5;
      %Pozzo 5
3079  psipoz5 = -q5./(2*pi).*theta5;
      phipoz5 = -q5./(2*pi).*log(r5);
3081  upoz5 = -cos(theta5).*q5./(2*pi*r5);
      vpoz5 = -sin(theta5).*q5./(2*pi*r5);
3083  handles.psip5 = psipoz5;
      handles.phip5 = phipoz5;
3085  handles.up5 = upoz5;
      handles.vp5 = vpoz5;
3087  %Doppietta 5
      psidop5 = sin(theta5-alpha5).*q5./(2*pi*r5);
3089  phidop5 = -cos(theta5-alpha5).*q5./(2*pi*r5);
      vdop5 = q5./(2*pi*r5).*cos(theta5-alpha5).*sin(theta5)+q5./...
3091  ...(2*pi*r5).*sin(theta5-alpha5).*cos(theta5);
      udop5 = q5./(2*pi*r5).*cos(theta5-alpha5).*cos(theta5)-q5./...
3093  ...(2*pi*r5).*sin(theta5-alpha5).*sin(theta5);
      handles.psid5 = psidop5;
3095  handles.phid5 = phidop5;
      handles.ud5 = udop5;
3097  handles.vd5 = vdop5;
      %Vortice 5
3099  psivor5 = -q5./(2*pi).*log(r5);
      phivor5 = q5./(2*pi).*theta5;
3101  uvor5 = -q5./(2*pi*r5).*sin(theta5);
      vvor5 = q5./(2*pi*r5).*cos(theta5);
3103  handles.psiv5 = psivor5;
      handles.phiv5 = phivor5;
3105  handles.uv5 = uvor5;
      handles.vv5 = vvor5;
3107  % Set the current data value with popupmenu_5.
      str = get(handles.popupmenu5, 'String');
3109  val5 = get(handles.popupmenu5,'Value');
      % Set current data to the selected data set.
3111  switch str(val5,:)
      case 'Corrente␣uniforme'
3113  handles.psi5 = handles.psic5;
      handles.phi5 = handles.phic5;
3115  handles.u5 = handles.uc5;
      handles.v5 = handles.vc5;
3117  case 'Sorgente␣␣␣␣␣␣␣␣␣'
      handles.psi5 = handles.psis5;
3119  handles.phi5 = handles.phis5;
      handles.u5 = handles.us5;
3121  handles.v5 = handles.vs5;
```

```
    case 'Pozzo␣␣␣␣␣␣␣␣␣␣␣␣'
3123 handles.psi5 = handles.psip5;
    handles.phi5 = handles.phip5;
3125 handles.u5 = handles.up5;
    handles.v5 = handles.vp5;
3127 case 'Doppietta␣␣␣␣␣␣␣␣'
    handles.psi5 = handles.psid5;
3129 handles.phi5 = handles.phid5;
    handles.u5 = handles.ud5;
3131 handles.v5 = handles.vd5;
    case 'Vortice␣␣␣␣␣␣␣␣␣␣'
3133 handles.psi5 = handles.psiv5;
    handles.phi5 = handles.phiv5;
3135 handles.u5 = handles.uv5;
    handles.v5 = handles.vv5;
3137 end
    %Set the final current data
3139 handles.psi = handles.psi1 + handles.psi2 + handles.psi3 +...
    ...handles.psi4 + handles.psi5;
3141 handles.phi = handles.phi1 + handles.phi2 + handles.phi3 +...
    ...handles.phi4 + handles.phi5;
3143 handles.u = handles.u1 + handles.u2 + handles.u3 +...
    ...handles.u4 + handles.u5;
3145 handles.v = handles.v1 + handles.v2 + handles.v3 +...
    ...handles.v4 + handles.v5;
3147 % Save the handles structure.
    guidata(hObject,handles)
3149 % Choose default command line output
    handles.output = hObject;
3151 % Update handles structure
    guidata(hObject, handles);
3153
    % --- Executes during object creation, after setting all properties.
3155 function alpha5_CreateFcn(hObject, eventdata, handles)

3157 if ispc && isequal(get(hObject,'BackgroundColor'),...
    ...get(0,'defaultUicontrolBackgroundColor'))
3159 set(hObject,'BackgroundColor','white');
    end
```

Bibliografia

[1] Arina, Renzo (2014), *Fondamenti di aerodinamica parte I: nozioni introduttive e teoria del fluido inviscido*, Levrotto & Bella, Torino.

[2] Scarsoglio, Stefania, Documenti del corso di aerodinamica anno accademico 2014-2015.

[3] http://it.mathworks.com/help/matlab/index.html

[4] http://it.mathworks.com/discovery/matlab-gui.html

[5] http://it.mathworks.com/help/matlab/gui-building-basics.html