



Università degli Studi di Padova

DIPARTIMENTO DI INGEGNERIA CIVILE, EDILE ED AMBIENTALE
Corso di Laurea Magistrale in Ingegneria Idraulica

TESI DI LAUREA MAGISTRALE

**Analisi della forma delle aree di drenaggio
dei canali a marea.**

Laureando:
Francesco Carraro
Matricola 1035404

Relatore:
Prof. Stefano Lanzoni
Correlatore:
Prof. Andrea D'Alpaos

Anno Accademico 2013-2014

Sommario

Nella tesi si sono studiate le caratteristiche geometriche delle aree di drenaggio dei canali a marea. La forma di tali aree è di cruciale importanza per la determinazione del profilo longitudinale e della sezione di equilibrio dei canali a marea.

In particolare, è stato realizzato un programma di calcolo fondato sul modello sviluppato da Rinaldo et al. [1999a] che risolve numericamente alle differenze finite l'equazione di Poisson risultante dalla semplificazione delle equazioni di De Saint Venant delle onde lunghe in acque basse. Tale programma è stato applicato allo studio della forma dei bacini di drenaggio afferenti ai canali lagunari solcanti la Palude Pagliaga (al confine nord-occidentale della laguna di Venezia).

Usando la digitalizzazione dei contorni delle barene effettuata da Rado [2014], è stata sviluppata una nuova procedura per generare, utilizzando il software ArcGis, il file contenente le condizioni al contorno, necessarie per la soluzione numerica dell'equazione di Poisson sopra richiamata.

Il modello numerico per la soluzione di tale equazione, originariamente sviluppato in Fortran [A. D'Alpaos 2001], è stato implementato in ambiente Matlab. Sfruttando le API, è stato infatti possibile realizzare i MEX-Files che traducono le subroutines Fortran.

Il programma così sviluppato è stato quindi utilizzato per analizzare la forma di 63 aree di drenaggio, estratte sulla base della soluzione dell'equazione di Poisson che fornisce la distribuzione spaziale della superficie libera dell'acqua su una superficie di barena. I contorni delle varie aree sono stati rappresentati rispetto a un sistema di riferimento cartesiano (cfr. Appendice B), di coordinate (n, s) , centrato nel baricentro dell'area. La direzione s è individuata dalla retta passante per il centro della sezione di sbocco del canale e il baricentro stesso dell'area.

Tale sistema di riferimento permette di rappresentare in modo oggettivo e con la medesima orientazione tutte le aree di drenaggio. Sono stati così calcolati i seguenti parametri di forma: le deviazioni standard rispetto agli assi (n, s) ; il loro rapporto R_σ ; la *skewness* e la *kurtosis* rispetto all'asse s .

Sulla base dei risultati ottenuti è stata proposta una classificazione delle forme delle aree di drenaggio, fondata sul rapporto R_σ , definita da quattro classi. La classificazione proposta è stata, quindi, verificata tramite confronto grafico dei perimetri delle aree di drenaggio appartenenti alla stessa classe, facendo coincidere l'origine del sistema di riferimento con il punto centrale della sezione di sbocco dei canali. Inoltre, per le stesse aree di drenaggio, in un secondo

grafico è stato riportato l'andamento lungo s delle larghezze B_t , valutate lungo la direzione individuata dall'asse n .

I confronti effettuati hanno mostrato che la classificazione proposta è in grado di individuare, con buona approssimazione, quattro categorie distinte di bacini di drenaggio. Essa fornisce quindi un metodo per organizzare in modo razionale le aree di drenaggio ai fini di uno studio più approfondito delle loro caratteristiche geometriche.

Indice

Sommario	iii
1 Introduzione	1
2 Il modello matematico	5
2.1 Identificazione dei partiacque in un bacino a marea	5
2.2 Equazioni delle onde lunghe in acque basse	7
2.3 Individuazione delle aree afferenti a ciascun canale	8
3 Area di studio ed imposizione delle condizioni al contorno	17
3.1 La laguna di Venezia	17
3.2 L'area di studio: la Palude Pagliaga	20
3.3 Condizioni al contorno e generazione del file di input con ArcGis	22
3.3.1 Condizioni al contorno	23
3.3.2 Il software ArcGis	24
3.3.3 Imposizione delle condizioni al contorno con ArcMap . .	27
3.4 Considerazioni conclusive sul procedimento proposto	35
4 L'implementazione del modello in Matlab	39
4.1 Il modello in Fortran	40
4.1.1 La subroutine <i>telone</i>	40
4.1.2 Definizione delle direzioni di drenaggio	43
4.1.3 La subroutine <i>segnarami</i>	46
4.2 L'implementazione in Matlab	48
4.2.1 I vantaggi della programmazione in Matlab	48
4.2.2 I MEX-File: cosa sono e perché utilizzarli	51
4.2.3 Come ottenere un MEX-File	52
4.2.4 Realizzazione delle funzioni di supporto ai MEX-File . . .	60
5 Analisi della forma delle aree di drenaggio dei canali	67
5.1 I risultati del modello per la Palude Pagliaga	67
5.2 Studio e classificazione della forma delle aree di drenaggio	74
5.3 Confronto tra le classi di aree individuate	82
6 Conclusioni	91

A	Listati dei codici implementati	95
A.1	Programma Matlab AreeDiDrenaggio2.2.m	97
A.2	Funzione Matlab <i>telone</i>	123
A.3	MEX-File della subroutine <i>telone</i>	127
A.4	Funzione Matlab <i>incidenzabarene</i>	137
A.5	MEX-File della subroutine <i>incibarene</i>	139
A.6	Funzione Matlab <i>ampiezza</i>	145
A.7	MEX-File della subroutine <i>ampiezza</i>	149
A.8	Funzione Matlab <i>incidenzacanali</i>	155
A.9	MEX-File della subroutine <i>incicanali</i>	157
A.10	Funzione Matlab <i>segnarami</i>	163
A.11	MEX-File della subroutine <i>segnarami</i>	165
B	Grafici della forma delle aree di drenaggio analizzate	171
	Bibliografia	187
	Ringraziamenti	189

Elenco delle figure

2.1	Grandezze caratteristiche e notazioni del modello matematico. . .	8
3.1	La laguna di Venezia.	19
3.2	Posizione della Palude Pagliaga.	21
3.3	Vista della Palude Pagliaga dall'aeroporto di Venezia.	21
3.4	Aree barenali della Palude Pagliaga.	22
3.5	Esempio della matrice delle condizioni al contorno.	24
3.6	Interfaccia grafica di ArcMap.	25
3.7	Digitalizzazione della Palude Pagliaga	28
3.8	Ricavare le condizioni al contorno in ArcMap: passo 1.	28
3.9	Funzione ArcMap <i>Feature to Polygon</i>	29
3.10	Ricavare le condizioni al contorno in ArcMap: passo 2.	30
3.11	Funzione ArcMap <i>Polygon to Raster</i>	30
3.12	Ricavare le condizioni al contorno in ArcMap: passo 3.	31
3.13	Funzione ArcMap <i>Reclassify</i>	32
3.14	Schermata di ArcMap in cui si visualizza l'immagine raster ottenuta dalla riclassificazione del raster di figura 3.12.	33
3.15	Funzione ArcMap <i>Aggregate</i>	34
3.16	Schermata di ArcMap in cui si visualizza l'immagine raster ottenuta dal raster di figura 3.14, con la toolbox <i>Aggregate</i>	34
3.17	Funzione ArcMap <i>Raster to ASCII</i>	35
3.18	Raster delle condizioni al contorno ad alta risoluzione.	36
3.19	Raster delle condizioni al contorno nella risoluzione utilizzata dal modello.	36
4.1	Scema della soluzione alle differenze finite dell'equazione (2.23) .	41
4.2	Notazione per la definizione delle direzioni di drenaggio.	44
4.3	Particolare dalla matrice <i>sezioni</i> visualizzato con il software <i>Surfer11</i>	47
4.4	Diagramma di flusso dei dati in un Fortran MEX-File	53
4.5	Command Window di Matlab: MEX setup	58
4.6	Command Window di Matlab: compilazione di un MEX-File	59
4.7	Command Window di Matlab: esempio del comando <i>help</i>	61
4.8	Command Window di Matlab: esempio dell'assegnazione dei dati di input alla function <i>telone</i>	65
5.1	Command Window di Matlab: esecuzione del modello	68
5.2	Soluzione dell'equazione di Poisson per le B.C. di Pagliaga	69
5.3	Direzioni di drenaggio sulle barene della Palude Pagliaga	70

5.4	Funzione di Ampiezza lungo i canali della Palude Pagliaga	71
5.5	Direzioni di drenaggio lungo i canali della Palude Pagliaga	71
5.6	Aree di drenaggio dei canali solcanti la Palude Pagliaga per ID	72
5.7	Aree di drenaggio dei canali solcanti la Palude Pagliaga ricolorate	73
5.8	Forma di sei aree di drenaggio ottenute da Fig. 5.6	75
5.9	Forma ri-orientata delle sei aree di drenaggio di Fig. 5.8	76
5.10	Distribuzione delle varianze σ_n e σ_s rispetto all'area dei bacini	79
5.11	Confronto diretto delle varianze σ_n e σ_s per ogni bacino lagunare	80
5.12	Distribuzione statistica dei valori di R_σ per tutti i bacini lagunari	81
5.13	Distribuzione statistica dei valori di R_σ esclusi i bacini con $ID \geq 67$	82
5.14	Confronto della forma delle aree di drenaggio per $R_\sigma \leq 0.8$	84
5.15	Confronto della forma delle aree di drenaggio per $0.8 < R_\sigma \leq 1.3$	85
5.16	Confronto della forma delle aree di drenaggio per $1.3 < R_\sigma \leq 2.0$	87
5.17	Confronto della forma delle aree di drenaggio per $R_\sigma > 2.0$	89
B.1	Forma delle aree di drenaggio di $9 \leq ID \leq 14$	172
B.2	Forma delle aree di drenaggio di $15 \leq ID \leq 20$	173
B.3	Forma delle aree di drenaggio di $21 \leq ID \leq 26$	174
B.4	Forma delle aree di drenaggio di $27 \leq ID \leq 32$	175
B.5	Forma delle aree di drenaggio di $33 \leq ID \leq 38$	176
B.6	Forma delle aree di drenaggio di $39 \leq ID \leq 44$	177
B.7	Forma delle aree di drenaggio di $45 \leq ID \leq 50$	178
B.8	Forma delle aree di drenaggio di $51 \leq ID \leq 56$	179
B.9	Forma delle aree di drenaggio di $57 \leq ID \leq 62$	180
B.10	Forma delle aree di drenaggio di $63 \leq ID \leq 68$	181
B.11	Forma delle aree di drenaggio di $69 \leq ID \leq 71$	182

Capitolo 1

Introduzione

Le caratterizzazione geometrica delle aree di drenaggio che afferiscono ai canali a marea che innervano gli ambienti di barena è di cruciale importanza per la determinare il profilo di equilibrio e la sezione la sezione di equilibrio dei canali stessi.

In letteratura risulta disponibile un metodo estremamente efficiente per l'estrazione automatica dell'andamento planimetrico dei canali lagunari e delle aree di drenaggio a questi afferenti. Tale metodo è descritto in una serie di tre articoli pubblicati nel 1999 sulla rivista *Water Resources Research*.

- Il primo di questi articoli [Fagherazzi et al. 1999], illustra la metodologia per l'estrazione della geometria delle reti di canali e dell'ambiente lagunare circostante, partendo da immagini satellitari e da rilievi topografici, ovvero la preparazione dei dati disponibili;
- Il secondo articolo [Rinaldo et al. 1999a] descrive come, semplificando opportunamente le equazioni di De Saint Venant delle onde lunghe in acque basse, la distribuzione spaziale della superficie libera a un dato istante sia descritta, sia pure in modo approssimativo, da una equazione di Poisson. Risulta così possibile utilizzare il metodo del massimo gradiente determinare in modo automatico i partiacque all'interno di un bacino lagunare;
- Il terzo contributo [Rinaldo et al. 1999b], infine, definisce una relazione per la determinazione delle portate che fluiscono nei canali a marea, partendo dalle dimensioni in pianta dei canali stessi e delle aree di drenaggio ad essi afferenti.

In particolare, nel caso degli ambienti a marea, l'utilizzo del modello sviluppato da Rinaldo et al. [1999a] si rende necessario poiché, a differenza dei bacini idrografici dove a ogni tratto d'alveo è possibile attribuire la corrispondente area drenata seguendo il gradiente topografico massimo, è impossibile ricavare dalla sola altimetria dei fondali l'area afferente a ciascun canale. Il moto, infatti, è determinato dai gradienti della superficie libera dell'acqua.

Le barene costituiscono la porzione topograficamente più elevata di un bacino lagunare, trovandosi a quote superiori a quella del medio mare. Conseguentemente, esse sono periodicamente interessate dalle sole fasi di alta marea, che le

sommergono durante la fase di flusso, scoprendole durante la successiva fase di riflusso. Generalmente, le barene sono caratterizzate da una struttura prevalentemente piatta e da un andamento altimetrico piuttosto tormentato per la presenza di una rete diffusa di piccoli canali, che le incidono e consentono alle correnti di marea di invaderle in modo regolare e progressivo.

I partiacque, che individuano le zone di barena drenate/alimentate dai vari canali presenti in un bacino, sono sostanzialmente determinati dall'idrodinamica del sistema, in particolar modo dai gradienti locali della superficie libera, piuttosto che dalla conformazione topografica dei fondali. Nello studio di ambienti a marea è quindi necessario utilizzare come direzione di drenaggio quella individuata dal gradiente della superficie libera, anche se è ovvio che la topografia, a sua volta, influenza il campo di moto e le elevazioni istantanee della superficie libera all'interno del bacino.

L'obiettivo che ci si prefigge con questa tesi è quello di sviluppare un programma di calcolo per analizzare oggettivamente le forme delle aree di drenaggio dei canali a marea. L'area di studio considerata per la messa a punto del metodo in questione è la porzione della laguna di Venezia che costituisce la Palude Pagliaga, ovvero il complesso di piccole isole barenali, situato nei pressi dell'aeroporto Marco Polo, ai confini nord-occidentali della laguna.

La schematizzazione della geometria del sistema a marea considerato viene affrontata con il supporto del software ArcGis. In particolare, viene proposta una nuova procedura per la discretizzazione dell'area di studio, basata sulla digitalizzazione dei contorni delle barene della Palude di Pagliaga [Rado 2014].

Definita la configurazione planimetrica della Palude Pagliaga, per questa parte della laguna si risolve numericamente il modello matematico proposto da Rinaldo et al. [1999a]. Nella tesi tale modello è stato implementato completamente in linguaggio Matlab, traducendo opportunamente (tramite MEX-Files, cfr. The MathWorks [2014b]) le subroutines originariamente codificate in linguaggio Fortran [A. D'Alpaos 2001].

Sulla base dei risultati prodotti dal programma, si propone, infine, una classificazione delle forme delle aree di drenaggio. Tale classificazione, effettuata in base ad alcuni parametri di forma, valutati rispetto ad un opportuno sistema di riferimento, consente di organizzare in modo razionale le forme assunte dalle aree di drenaggio, fornendo una solida base di partenza per uno studio più approfondito delle caratteristiche geometriche assunte da tali aree.

Il resto della tesi è organizzato come segue:

- Nel Capitolo 2 si riporta il modello matematico proposto da Rinaldo et al. [1999a];
- Nel Capitolo 3 si descrive l'area di studio e come questa può essere discretizzata utilizzando il software gis ArcMap;
- Nel Capitolo 4 si illustra come avviene l'implementazione del programma per la soluzione numerica dell'equazione di Poisson, ai fini di determinare la forma delle aree di drenaggio afferenti a vari canali, e si descrive la procedura adottata per la traduzione delle subroutines in MEX-Files;

- Nel Capitolo 5 si illustrano i risultati ottenuti dall'applicazione del programma di calcolo alla Palude Pagliaga, e si propone un metodo per l'analisi e la classificazione della forma delle aree di drenaggio dei canali a marea.

Capitolo 2

Il modello matematico

Per comprendere come avviene la determinazione della forma dei bacini di drenaggio, in questo capitolo è riportata la trattazione matematica alla base del modello utilizzato per la definizione dei bacini afferenti ai canali solcanti la Palude Pagliaga¹[A. D'Alpaos 2001].

In particolare, in questo capitolo sono studiati i legami tra la superficie dei bassifondi e delle barene e la struttura dei canali che le innervano². È evidente come i canali siano in competizione tra loro per drenare o alimentare le circostanti zone del bacino e come tale competizione regoli l'aggregazione della rete. Si definisce, sulla base di un'analisi idrodinamica, un modello semplificato per l'individuazione dei sottobacini afferenti a ciascun canale e della posizione dei partiacque relativi ai differenti rami della rete.

2.1 Identificazione dei partiacque in un bacino a marea

Le forti analogie che si riscontrano tra la struttura morfologica delle reti fluviali e delle reti di canali a marea suggeriscono di utilizzare, per quest'ultime, le definizioni e le metodologie sviluppate per descrivere la rete idrografica che drena un dato bacino idrografico.

Nei bacini fluviali è possibile ricavare l'area drenata da ciascun elemento di un corso d'acqua direttamente dalla topografia. Tale grandezza costituisce un'importante caratteristica fisica del sistema ed è stata ampiamente utilizzata per studiare i fenomeni fluviali. Infatti, in tali bacini, esiste una stretta correlazione tra la portata che fluisce in una generica sezione del bacino e l'area drenata dalla sezione stessa (e.g. tra area afferente e portata media annua).

Trascurando l'effetto d'invaso nel bacino e supponendo le precipitazioni sempre uniformemente distribuite sul bacino stesso, la relazione di proporzionalità tra area drenata e portata può essere estesa anche ad eventi idrologici

¹Piccola porzione della laguna di Venezia nord-occidentale, oggetto di studio in questa tesi (cfr. paragrafo 3.2, pag. 20).

²Per la descrizione dei tre ambienti lagunari citati (*barene*, *bassifondi* e *canali*) si rimanda al paragrafo 3.1 (pag. 17)

geomorfologicamente significativi (i cosiddetti “*landscape forming events*”) per la modellazione degli alvei e la formazione dei reticoli idrografici. Questa proporzionalità tra portata fluente in una sezione e area drenata, è stata utilizzata per studiare le proprietà di scala delle reti fluviali [Rodriguez-Iturbe e Rinaldo 1997], per costruire modelli evolutivi dei bacini fluviali [Rinaldo et al. 1995] e per studiare i meccanismi che portano alla formazione della parte canalizzata di un bacino idrografico [Montgomery e Dietrich 1988, 1992].

Si tratta, tuttavia, di risultati che non possono essere estesi direttamente allo studio dei bacini a marea. In un bacino a marea non esiste, infatti, una relazione semplice tra l’area di un generico sottobacino e le portate fluenti nei canali. Infatti, mentre nei bacini fluviali è possibile attribuire a ogni tratto d’alveo la corrispettiva area drenata seguendo il gradiente topografico massimo, nei canali a marea è impossibile ricavare dalla sola altimetria dei fondali l’area afferente a ciascun canale.

I partiacque, che individuano le zone d’acqua bassa afferente ai vari canali presenti in un bacino, si spostano durante un ciclo mareale, sotto l’azione forzante della marea agente in corrispondenza delle bocche e sono sostanzialmente determinati dall’idrodinamica del sistema, in particolar modo dai gradienti locali della superficie libera, piuttosto che dalla conformazione topografica dei fondali. Nello studio di ambienti a marea appare, quindi, più sensato utilizzare come direzione di drenaggio quella individuata dal gradiente della superficie libera, anche se è ovvio che la topografia, a sua volta, influenza il campo di moto e le elevazioni istantanee della superficie libera all’interno del bacino.

É in ogni caso evidente come i canali presenti in un bacino a marea siano in competizione tra di loro per drenare, in fase di riflusso, o alimentare, in fase di flusso, le varie zone del bacino stesso e come i canali di maggiori dimensioni (e quindi con maggiori profondità) costituiscano percorsi preferenziali per quanto concerne i flussi e/o i riflussi interessanti le zone del bacino ad essi limitrofe.

Poiché gli spostamenti dei partiacque sono contenuti, la possibilità di suddividere, sia pure in prima approssimazione, l’area complessiva del bacino in zone di influenza dei diversi canali è di fondamentale importanza per studiare i meccanismi di aggregazione della rete [Rinaldo et al. 1999a].

Con questi obiettivi Boom [1975] e Pethick [1980], presupponendo nota e fissa nel tempo l’area afferente a un determinato canale che alimenta o drena una superficie di barena, trascurando la deformazione dell’onda di marea nel propagarsi sulla barena stessa, hanno sviluppato un modello idrodinamico molto semplificato basato sull’equazione di continuità, in grado di calcolare la velocità e la portata nella sezione terminale del canale in esame.

Altri studiosi hanno utilizzato il concetto di “bacino corto” (nel quale si potessero trascurare i tempi caratteristici dei processi propagatori), o di comportamento quasi-statico degli specchi d’acqua, per giustificare l’assunzione di un livello d’acqua costante nello spazio e dipendente solo dal tempo [Shuttelaars e De Swart 1996; Van Dongeren e De Vriend 1994]. Si tratta di lavori preliminari, rivolti soprattutto a comprendere le modalità di evoluzione morfologica di una superficie di barena, con aree in gioco poco estese e di gran lunga inferiori rispetto a quelle di bacini lagunari, quali la Laguna di Venezia, ai quali il sopra citato

schema statico non può applicarsi, se non in prima approssimazione, essendo relativamente importanti le differenze istantanee di livello che si riscontrano da punto a punto al suo interno.

Per superare i limiti dello schema quasi-statico, in studi successivi, si è, infine, mostrato come, in un bacino a marea, la componente d'attrito possa risultare preponderante rispetto ai termini inerziali, consentendo una più razionale ed accettabile semplificazione delle equazioni delle onde lunghe in acque basse [Lanzoni e Seminara 1998].

Utilizzando congiuntamente i contributi di questi filoni di ricerca, è stato derivato analiticamente un modello in grado di individuare i partiacque [Rinaldo et al. 1999a], ovvero le aree afferenti a ciascun canale, considerando che la propagazione della marea lungo i canali avvenga istantaneamente, ovvero con celerità di propagazione infinitamente più grande di quella con cui l'onda si propaga su le barene.

Come si potrà vedere nei prossimi capitoli, l'applicazione di questo modello, originariamente effettuata tramite dei codici scritti in *Fortran*, verrà eseguita all'interno del software *Matlab* al fine di studiare un bacino naturale, estratto da una porzione della parte Nord-orientale della Laguna di Venezia.

Una volta determinate, mediante la modellazione matematica nel seguito riasunta, le aree afferenti a ogni canale, si tenterà di capire quali siano le grandezze in grado di caratterizzarne la forma e se vi sia una relazione tra queste grandezze e le caratteristiche della rete dei canali.

2.2 Equazioni delle onde lunghe in acque basse

Il moto a superficie libera di onde lunghe in acque basse, quali si possono con buona approssimazione considerare le onde di marea che si propagano su una barena, è descritto dalle equazioni di De Saint Venant, le quali derivano dall'integrazione delle equazioni di Reynolds lungo la verticale.

Sulla base di queste equazioni è possibile studiare il comportamento di una laguna, il cui campo di moto si sviluppa prevalentemente nel piano orizzontale.

Assumendo l'approssimazione di Boussinesq lungo la direzione orizzontale e la distribuzione idrostatica delle pressioni lungo la verticale, le equazioni di De Saint Venant si scrivono [e.g. Dronkers 1964, p. 518]:

$$\frac{\partial u^*}{\partial t^*} + \left[u^* \frac{\partial u^*}{\partial x^*} + v^* \frac{\partial u^*}{\partial y^*} \right] = -g \frac{\partial \eta^*}{\partial x^*} - g \frac{u^*}{\chi^2 D^*} \sqrt{u^{*2} + v^{*2}} \quad (2.1)$$

$$\frac{\partial v^*}{\partial t^*} + \left[u^* \frac{\partial v^*}{\partial x^*} + v^* \frac{\partial v^*}{\partial y^*} \right] = -g \frac{\partial \eta^*}{\partial y^*} - g \frac{v^*}{\chi^2 D^*} \sqrt{u^{*2} + v^{*2}} \quad (2.2)$$

$$\frac{\partial}{\partial x^*} (D^* u^*) + \frac{\partial}{\partial y^*} (D^* v^*) + \frac{\partial D^*}{\partial t^*} = 0 \quad (2.3)$$

in cui, seguendo la notazione illustrata in Figura 2.1:

- (u^*, v^*) sono, rispettivamente, le componenti lungo le direzioni (x^*, y^*) delle velocità medie lungo la verticale;

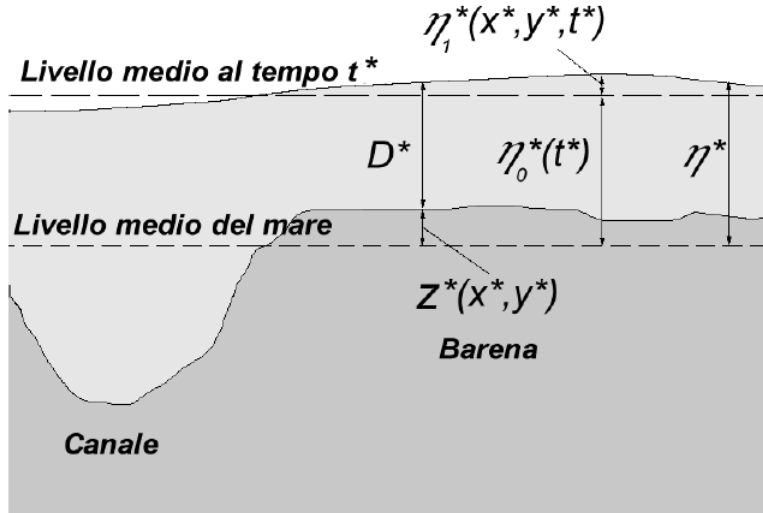


Figura 2.1: Grandezze caratteristiche e notazioni: η^* è l'elevazione della marea rispetto al medio mare, la quale può essere divisa in η_0^* , media spaziale dell'elevazione al tempo t^* , ed η_1^* , differenza locale rispetto ad η_0^* ; D^* è il tirante d'acqua sulla barena; z^* è l'elevazione della superficie della barena rispetto al medio mare [Rinaldo et al. 1999a].

- η^* è la quota della superficie libera rispetto al medio mare;
- t^* è il tempo;
- D^* è il tirante d'acqua;
- χ è il coefficiente di resistenza al moto secondo Chèzy;
- g è l'accelerazione di gravità.

2.3 Individuazione delle aree afferenti a ciascun canale

Si esamina ora il moto di un'onda di marea che si propaga su una superficie di barena. Il tirante d'acqua istantaneo può essere decomposto nel seguente modo (cfr. Figura 2.1):

$$D^*(x^*, y^*, t^*) = \eta_0^*(t^*) + \eta_1^*(x^*, y^*, t^*) - z^*(x^*, y^*) \quad (2.4)$$

in cui $\eta_0^*(t^*)$ è la quota media, rispetto al livello del medio mare, della superficie libera nel bacino considerato al tempo generico t^* (η_0^* è dunque la media spaziale dell'elevazione del pelo libero dovuta alla marea), $\eta_1^*(x^*, y^*, t^*)$ è la differenza locale tra la quota istantanea della superficie libera e il valor medio istantaneo η_0^* , $z^*(x^*, y^*)$ è la quota del fondo della barena riferita al medio mare (z^* può assumere quindi valori negativi).

Al fine di introdurre eventuali semplificazioni delle equazioni di de Saint Venant, è necessario valutare l'importanza relativa dei vari termini che vi compaiono. Sarà in questo modo possibile trascurare i termini che, per la particolare situazione considerata, risultassero di ordine superiore rispetto ai termini conservati. Per perseguire questo proposito è conveniente adimensionalizzare le equazioni del moto, introducendo le seguenti grandezze scala:

$$(\eta_0^*, z^*) = D_0^* (\eta_0, z), \quad \eta_1^* = a^* \eta_1 \quad (2.5a)$$

$$(u^*, v^*) = U^* (u, v), \quad t^* = t \omega_*^{-1} \quad (2.5b)$$

$$(x^*, y^*) = L^* (x, y) \quad (2.5c)$$

in cui, nelle (2.5):

- ω_* è la frequenza dominante dell'onda di marea;
- L^* è la scala spaziale significativa per descrivere le variazioni planimetriche delle grandezze geometriche del campo di moto;
- D_0^* indica il valore caratteristico (valore massimo) assunto dal tirante medio sulla barena durante un ciclo di marea (raggiunto durante la marea primaverile);
- U^* è il valore caratteristico della velocità mediata sulla verticale;
- a^* è la scala tipica delle variazioni spaziali della superficie libera indotte dalla marea.

L'assunzione di tali grandezze per l'adimensionalizzazione delle equazioni del moto differisce da quella generalmente utilizzata per la loro linearizzazione [si veda, ad esempio, LeBlond 1978]. Solitamente, infatti, si procede riscaldando il tirante con la profondità media dell'acqua e la quota locale dell'onda di marea con l'escursione della marea stessa. In un secondo tempo si trascura l'escursione della marea rispetto al tirante, linearizzando così le equazioni. In questo modo ci si limita, però, a studiare ambienti a marea dove i tiranti d'acqua sono sensibilmente superiori rispetto alle variazioni di livello indotte dalla marea stessa, ovvero situazioni molto diverse da quelle che si vogliono qui considerare.

In molte situazioni accade che la quota delle barene sia superiore al livello del medio mare, per cui il tirante d'acqua medio su di esse, dovuto all'escursione della marea, è del tutto confrontabile con quest'ultima grandezza.

La modesta estensione delle barene e il gran numero di canali che le solcano suggeriscono, in ogni caso, che le differenze istantanee del pelo libero tra i vari punti della superficie barenale siano generalmente piccole e non confrontabili con il tirante medio d'acqua sulla barena, che assume valori decisamente superiori durante buona parte del ciclo di marea. Ovviamente tali ipotesi vengono meno quando la barena tende a scoprirsi, ovvero quando il tirante d'acqua tende ad annullarsi.

Ciò premesso, utilizzando le (2.4) e (2.5) per descrivere le diverse grandezze adimesionali che compaiono nelle (2.1), (2.2), (2.3), scritte per semplicità omettendo l'asterisco, porta alla seguente forma adimensionale delle equazioni della

quantità di moto e di continuità:

$$\frac{U\omega L}{ga} \frac{\partial u}{\partial t} + \frac{U^2}{ga} \left[u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \right] = -\frac{\partial \eta_1}{\partial x} - \frac{LU^2}{\chi^2 a} \frac{u}{\eta_0 + \epsilon \eta_1 - z} \sqrt{u^2 + v^2} \quad (2.6)$$

$$\frac{U\omega L}{ga} \frac{\partial v}{\partial t} + \frac{U^2}{ga} \left[u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} \right] = -\frac{\partial \eta_1}{\partial y} - \frac{LU^2}{\chi^2 a} \frac{v}{\eta_0 + \epsilon \eta_1 - z} \sqrt{u^2 + v^2} \quad (2.7)$$

$$\frac{\partial}{\partial t}(\eta_0 + \epsilon \eta_1) + \frac{U}{\omega L} \left\{ \frac{\partial}{\partial x} [u (\eta_0 + \epsilon \eta_1 - z)] + \frac{\partial}{\partial y} [v (\eta_0 + \epsilon \eta_1 - z)] \right\} = 0 \quad (2.8)$$

Nell'ipotesi che le quote della superficie della barena possano essere ritenute costanti (cioé $z(x, y) \simeq z_b$, superficie non canalizzata piana) si ottiene:

$$\mathcal{S} \frac{\partial u}{\partial t} + \mathcal{S} \mathcal{F} \left[u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \right] + \frac{\partial \eta_1}{\partial x} + \mathcal{R} \frac{u}{\eta_0 + \epsilon \eta_1 - z_b} \sqrt{u^2 + v^2} = 0 \quad (2.9)$$

$$\mathcal{S} \frac{\partial v}{\partial t} + \mathcal{S} \mathcal{F} \left[u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} \right] + \frac{\partial \eta_1}{\partial y} + \mathcal{R} \frac{v}{\eta_0 + \epsilon \eta_1 - z_b} \sqrt{u^2 + v^2} = 0 \quad (2.10)$$

$$\frac{\partial}{\partial t}(\eta_0 + \epsilon \eta_1) + \mathcal{F} \left\{ \frac{\partial}{\partial x} [u (\eta_0 + \epsilon \eta_1 - z_b)] + \frac{\partial}{\partial y} [v (\eta_0 + \epsilon \eta_1 - z_b)] \right\} = 0 \quad (2.11)$$

Nelle (2.9), (2.10) e (2.11), i parametri adimensionali \mathcal{S} , \mathcal{F} , \mathcal{R} e ϵ sono espressi dalle seguenti relazioni:

$$\begin{aligned} \mathcal{F} &= \frac{U}{\omega L}, & \mathcal{S} &= \frac{\omega UL}{ga} \\ \mathcal{R} &= \frac{LU^2}{\chi^2 D_0 a}, & \epsilon &= \frac{a}{D_0} \end{aligned} \quad (2.12)$$

É immediato osservare che il parametro adimensionale \mathcal{S} , che compare nelle equazioni della quantità di moto, rappresenta il rapporto esistente, punto per punto, tra le forze d'inerzia e quelle di gravità, mentre il parametro \mathcal{R} rappresenta il rapporto tra le forze d'attrito e le forze di gravità. Vale la pena sottolineare che il rapporto $\mathcal{S}/\mathcal{R} = (\chi^2 D_0 \omega)/(gU)$, ovvero il rapporto tra le forze d'inerzia e quelle d'attrito, è indipendente sia dalla scala spaziale L , sia dalla scala tipica delle variazioni spaziali dell'elevazione della superficie liquida.

I dati sperimentali ricavati dalla letteratura, riportati in Tabella 2.1, evidenziano come, in molte situazioni, risulti $\mathcal{S}/\mathcal{R} \ll 1$, sia sulla superficie della barena che nei piccoli canali (*ghebi*). Pertanto, ove questo accada, le caratteristiche del campo di moto sono dominate dai fenomeni di resistenza, che prevalgono rispetto al ruolo dell'accelerazione locale e convettiva [e.g, LeBlond 1978; Friedrichs e Madsen 1992; Lanzoni e Seminara 1998].

Se si considera l'equazione di continuità (2.11), al fine di bilanciare il termine dipendente dal tempo con i termini che tengono conto del flusso d'acqua

Tabella 2.1: Grandezze scala e parametri adimensionali. Fonti: Myrick 1963; Bayliss-Smith et al. 1979; Healey et al. 1981; French e Stoddart 1992; Lynn, Hine e Luther 1995; Shi, Pethick e Pye 1995.

	$U(m/s)$	$D_0(m)$	$\chi(m^{1/2}/s)$	a/L	S/\mathcal{R}
Ghebi	$0.5 \div 0.7$	$2 \div 3$	$30 \div 40$	$\sim 10^{-5}$	~ 0.1
Barena	$0.05 \div 0.1$	$0.2 \div 0.3$	$10 \div 20$	$\sim 10^{-4}$	~ 0.01

nelle due direzioni spaziali (per garantire la continuità della massa), si ottiene immediatamente che $\mathcal{F} = 1$, risultato che permette di stimare la scala spaziale L .

Sulla base dei dati riportati in Tabella 2.1 si può vedere come, in molte situazioni, i valori assunti da L varino tra $300 - 700 m$; sono questi i valori rappresentativi del percorso compiuto dalle particelle d'acqua sulla superficie della barena prima di giungere a un canale. Inoltre, nelle equazioni (2.9) e (2.10), che esprimono il bilancio della quantità di moto, una volta trascurati i termini inerziali, essendo $S/\mathcal{R} \ll 1$, necessariamente si deve porre $R = 1$ affinché i termini che restano di queste equazioni abbiano significato. Ne discende come conseguenza:

$$\frac{a}{L} = \frac{U^2}{\chi^2 D_0} \quad (2.13)$$

Dalla (2.13), si ottiene, in accordo con i valori raccolti in campo e riportati in letteratura [Healey et al. 1981; French e Stoddart 1992], l'ordine di grandezza dei gradienti della superficie libera a/L , che sono $O(10^{-5})$ per la rete dei *ghebi* e $O(10^{-4})$ per la superficie della barena (cfr. Tabella 2.1).

Trascurando, sulla base di queste considerazioni, i termini di accelerazione locale e convettiva, tenuto conto della sola pendenza della superficie libera e dei termini d'attrito, si ottengono le seguenti equazioni semplificate del moto:

$$\frac{\partial \eta_1}{\partial x} = -\frac{u}{\eta_0 + \epsilon \eta_1 - z_b} \sqrt{u^2 + v^2} \quad (2.14)$$

$$\frac{\partial \eta_1}{\partial y} = -\frac{v}{\eta_0 + \epsilon \eta_1 - z_b} \sqrt{u^2 + v^2} \quad (2.15)$$

Nelle (2.14) e (2.15) il termine di resistenza al moto può essere linearizzato, utilizzando il criterio energetico introdotto da Lorenz [Lorenz 1926] ponendo:

$$\frac{u}{\eta_0 + \epsilon \eta_1 - z_b} \sqrt{u^2 + v^2} = \Lambda \frac{u}{\eta_0 + \epsilon \eta_1 - z_b} \quad (2.16)$$

$$\frac{v}{\eta_0 + \epsilon \eta_1 - z_b} \sqrt{u^2 + v^2} = \Lambda \frac{v}{\eta_0 + \epsilon \eta_1 - z_b} \quad (2.17)$$

nelle quali Λ è un coefficiente efficace d'attrito che dipende dal modulo della velocità $\sqrt{u^2 + v^2}$. Se si assume che esso vari in modo proporzionale ad una velocità U_{\max} costante pari al massimo valore della velocità caratteristica U in un

ciclo di marea, si ottiene la linearizzazione del il termine quadratico che compare nelle (2.14) e (2.15).

L'approssimazione ora introdotta è appropriata per una analisi delle caratteristiche medie del campo di moto di un bacino a marea (come ad esempio la determinazione dei partiacque), ma non per la determinazione dell'esatta distribuzione delle velocità [Zimmerman 1982].

Sempre con riferimento agli effetti delle approssimazioni introdotte nelle equazioni del moto, è da osservare che i termini non lineari che in esse compaiono tendono a deformare l'onda di marea. Pertanto la propagazione di un'onda di marea è generalmente accompagnata da un processo di distorsione che favorisce la generazione di armoniche di frequenza superiore a quelle della forzante. Nel caso specifico, la linearizzazione del termine di attrito può conseguentemente portare a una non corretta rappresentazione del campo idrodinamico, specialmente nel caso in cui gli effetti di tale termine risultino particolarmente importanti [Lanzoni e Seminara 1998].

Con l'ipotesi di Lorentz, le equazioni del moto diventano:

$$\frac{\partial \eta_1}{\partial x} = -\Lambda \frac{u}{\eta_0 + \epsilon \eta_1 - z_b} \quad (2.18)$$

$$\frac{\partial \eta_1}{\partial y} = -\Lambda \frac{v}{\eta_0 + \epsilon \eta_1 - z_b} \quad (2.19)$$

le quali evidenziano che la velocità, in una assegnata direzione, dipende dal gradiente del pelo libero nella stessa direzione. Noto, dunque, il livello della superficie liquida in ogni punto del bacino, è determinata la direzione del flusso.

Sostituendo le (2.18) e (2.19) nell'equazione di continuità (2.11) si ottiene:

$$\frac{\partial}{\partial t}(\eta_0 + \epsilon \eta_1) - \frac{1}{\Lambda} \left\{ \frac{\partial}{\partial x} \left[(\eta_0 + \epsilon \eta_1 - z_b)^2 \frac{\partial \eta_1}{\partial x} \right] + \frac{\partial}{\partial y} \left[(\eta_0 + \epsilon \eta_1 - z_b)^2 \frac{\partial \eta_1}{\partial y} \right] \right\} = 0 \quad (2.20)$$

Va osservato che, nella (2.20), il parametro ϵ non è necessariamente piccolo. Infatti, come già definito in precedenza, ϵ rappresenta il rapporto tra la scala tipica delle variazioni spaziali istantanee della quota della superficie liquida durante un ciclo di marea e il massimo valore assunto dal tirante d'acqua caratteristico sulla barena, valori che possono essere dello stesso ordine.

Relativamente a questo aspetto, si consideri dapprima un bacino molto esteso, in cui la marea impiega un tempo dell'ordine di grandezza del periodo mareale per raggiungere i bordi del bacino stesso. In questa situazione la deviazione della superficie libera rispetto alla media spaziale in quell'istante è, per forza di cose, dell'ordine di grandezza dell'escursione della marea. Infatti, se alla bocca la marea è al culmine, nelle estreme propaggini del bacino essa potrebbe trovarsi sfasata anche di mezzo periodo e presentare il suo valore minimo, per cui ϵ viene ad assumere valori addirittura superiori all'unità.

Si diminuiscano ora le dimensioni del bacino, se la marea impiega un intervallo di tempo pari ad un quarto del periodo per raggiungere le sue estreme propaggini, la differenza in elevazione può essere, al più, pari alla metà dell'escursione complessiva della marea ed ϵ risulta, in questo caso, pari ad 1.

Formalizzando questi concetti e assumendo in prima approssimazione che l'onda rimanga sinusoidale e con la stessa ampiezza mentre si propaga nel bacino, la differenza massima d'elevazione può essere scritta come:

$$\Delta h_{max} = \max_t [A \sin \omega t - A \sin \omega(t - \Delta t)] = A \sin \omega \frac{\Delta t}{2} \quad (2.21)$$

se si indica con Δt il tempo che l'onda impiega per andare dal primo al secondo punto.

Diminuendo le dimensioni del bacino, diminuisce il ritardo Δt con cui arriva l'onda e, quindi, anche la differenza in elevazione tra punto e punto all'interno del bacino stesso. Al diminuire di Δt , perciò, il parametro ϵ diventa sempre più piccolo.

Il ragionamento resta valido anche considerando una marea reale che, oltre a sfasarsi, si deforma e si attenua (o si amplifica). Infatti, sia lo sfasamento che l'attenuazione dell'onda risultano legate al cammino che l'onda di marea deve percorrere propagandosi: se si considerano dimensioni sempre minori del bacino, anche tale cammino diminuisce, fino a che il ritardo di fase e l'attenuazione diventano trascurabili.

Sulle basi di queste premesse, è evidente che il coefficiente ϵ è piccolo solo per bacini a marea di modesta estensione per i quali i tempi di propagazione sono relativamente modesti e l'attenuazione dell'onda di marea è così debole da risultare praticamente trascurabile.

Eliminando i termini di ordine ϵ nell'equazione (2.20) si ottiene:

$$\nabla^2 \eta_1 = \frac{\Lambda}{(\eta_0 - z_b)^2} \frac{\partial \eta_0}{\partial t} \quad (2.22)$$

che è l'equazione differenziale di Poisson che governa il fenomeno nelle ipotesi assunte.

Per quanto riguarda la soluzione della (2.22), si deve tener presente che il termine a secondo membro è funzione della forzante mareale ($\partial \eta_0 / \partial t$), del parametro di attrito Λ e del tirante medio dell'acqua sulla barena ($\eta_0 - z_b$), i cui valori variano nel tempo.

In un assegnato istante t , tuttavia, il termine a secondo membro della (2.22) può essere determinato e assume un valore costante. Conseguentemente, la soluzione della (2.22) è possibile, una volta assegnate le condizioni al contorno, e fornisce, per l'istante considerato, l'andamento della superficie liquida $\eta_1(x, y, t)$. Da tale andamento, attraverso le (2.18) e (2.19), utilizzando il metodo della discesa lungo il gradiente massimo, si possono dedurre le direzioni di drenaggio. Una volta calcolati i gradienti $\nabla \eta_1(\mathbf{x})$, è quindi possibile determinare la posizione degli eventuali partiacque, che consentono a loro volta di individuare le aree afferenti a ciascun canale.

In particolare, il bacino a marea viene discretizzato in elementi di maglia quadrata (cfr. par. 3.3.1). Ciascun elemento è assunto drenare in direzione del punto avente la quota del livello d'acqua minore tra gli otto punti adiacenti.

In generale la soluzione della (2.22) viene a dipendere, in un dato istante, sia dalle condizioni al contorno, sia dal valore del termine (costante) a secondo

membro, ovvero dalla fase della marea. Ne consegue che, in un bacino a marea, la linea di partiacque non è fissa nel tempo, ma varia istante per istante.

Con riferimento al problema esaminato, in generale, le condizioni al contorno, che consentono di dare pratica soluzione alla (2.22), possono essere applicate in corrispondenza a tre diversi tipi di contorno che, dal punto di vista morfologico, possono essere suddivisi in canali (o *ghebi*), contorni impermeabili (terraferma), o scarpate che dividono la barena dai bassifondi adiacenti.

Una soluzione della (2.22), conseguita introducendo condizioni al contorno particolari e valide solo in prima approssimazione, è stata data in [Rinaldo et al. 1999a]. L'ipotesi fondamentale, sulla quale tale soluzione si fonda, implica che l'onda di marea si propaghi con celerità infinita lungo i canali e, quindi, istantaneamente rispetto a quanto avviene sulle barene o sui bassifondi. E, infatti, l'onda di marea in genere viaggia molto più velocemente lungo la rete dei canali che nelle zone di acque basse ad essi adiacenti, essendo la sua celerità di propagazione proporzionale, in prima approssimazione, a $\sqrt{gD_0}$, dove D_0 è il valore del tirante medio. Tale ipotesi è d'altra parte confermata dall'osservazione sperimentale che, come è mostrato nella Tabella 2.1, i gradienti della superficie libera, lungo i canali che fiancheggiano o incidono una barena, sono generalmente almeno di un ordine di grandezza inferiore rispetto a quelli che si rilevano per la superficie liquida sulla barena stessa [Healey et al. 1981]. Come conseguenza è lecito, in prima approssimazione, supporre un livello di marea costante lungo i canali, in un assegnato istante, ovvero assumere, come già sottolineato, celerità di propagazione infinita lungo i canali stessi.

È, questa, una ipotesi che ragionevolmente può essere assunta anche per i tratti di contorno costituiti dalle scarpate che delimitano le barene dai bassifondi. La profondità dei bassifondi, infatti, è spesso dello stesso ordine della profondità di molti dei canali che solcano le barene, per cui anche su di essi si stabiliscono gradienti istantanei del pelo libero non dissimili da quelli dei canali e, comunque, molto meno accentuati di quelli che si riscontrano sulle barene.

Condizioni di flusso nullo si devono, invece, assumere lungo i bordi impermeabili in direzione normale al contorno.

In definitiva, in termini matematici, indicata con Γ la linea di confine del dominio di calcolo, con le ipotesi assunte, il problema è ricondotto alla soluzione del seguente sistema di equazioni:

$$\nabla^2 \eta_1 = K \quad (2.23a)$$

$$\partial \eta_1 / \partial n = 0 \quad \text{in: } \Gamma_1 \quad (2.23b)$$

$$\eta_1 = 0 \quad \text{in: } \Gamma_2 \quad (2.23c)$$

in cui Γ_1 è la parte di bordo, con n direzione della normale, costituita da zone di terraferma impermeabili al flusso e Γ_2 è la parte di bordo che separa la barena dai bassifondi, dai canali e dai *ghebi*. Lungo quest'ultima parte del contorno, la condizione prescritta da Rinaldo et al. [1999a], $\eta_1 = 0$, indica che il livello è indipendente dallo spazio e pari al livello istantaneo di marea alla bocca.

I limiti di questa soluzione risiedono non tanto nelle approssimazioni introdotte per giungere alla (2.22), ovvero nell'aver trascurato i termini di accelerazione

convettiva e locale nelle equazioni del moto, quanto piuttosto nella semplificazione adottata per le condizioni al contorno, assumendo per i canali e per i bassifondi il medesimo (e costante) livello.

Ne deriva un primo discutibile risultato per cui, indipendentemente dalla profondità e dalla larghezza dei canali, la cosiddetta linea di partiacque si collocherebbe in posizione intermedia tra questi elementi. Nella realtà, la morfologia della rete e le sezioni dei suoi canali dovrebbero assumere un ruolo non trascurabile nel processo di riempimento e vuotamento delle zone di barena: i canali di maggiori dimensioni, che sono percorsi anche da maggiori portate, dovrebbero drenare, in fase di flusso, o alimentare, in fase di riflusso, superfici più ampie.³

D'altra parte, l'interesse principale della soluzione proposta risiede nel fatto che la superficie drenata da ciascun canale o dai bassifondi risulta essere indipendente dalla profondità dei canali e dei bassifondi e richiede solamente di conoscere la struttura planimetrica della rete, che determina univocamente l'area afferente a ciascuna sezione di canale. In questo modo la soluzione numerica della (2.23a) risulta molto semplice da ottenere, ben integrandosi con i nuovi software per l'elaborazione delle immagini aeree ad alta risoluzione.

³Un primo passo per rimuovere alcune delle semplificazioni introdotte, considerando per i canali e per i *ghebi* un andamento istantaneo dei livelli non costante, ma variabile nello spazio come conseguenza del valore finito della celerità di propagazione dell'onda di marea lungo i canali stessi, lo si può trovare in [A. D'Alpaos 2001, pp. 33–51].

Capitolo 3

Area di studio ed imposizione delle condizioni al contorno

Nel paragrafo 2.3 è stato definito come sia possibile determinare le aree dei bacini afferenti ai diversi canali che incidono una barena, sulla base della soluzione dell'equazione (2.23a). Assumendo che, lungo la rete canalizzata, la celerità di propagazione dell'onda di marea sia infinitamente più grande rispetto a quella che si realizza sulle barene [Rinaldo et al. 1999a], la soluzione del problema risulta essere indipendente dalla profondità dei canali e dei bassifondi e richiede solamente di conoscere la struttura planimetrica della rete. Ovvero, la soluzione del problema è univocamente determinata note le condizioni sul contorno dell'area barenale che si vuole analizzare: ovvero lungo i canali che la solcano, lungo le scarpate che la separano dai bassifondi e lungo i contorni impermeabili che la dividono dalla terraferma.

In questo capitolo, dopo aver descritto quali sono le caratteristiche della laguna di Venezia e dell'area di studio analizzata, si propone una procedura per ricavare la matrice che definisce le condizioni al contorno per la soluzione della (2.23a), sfruttando le potenzialità del software gis ArcMap.

3.1 La laguna di Venezia

La Laguna di Venezia, il più esteso ambiente umido in Italia, occupa un'area di circa 550 km^2 e ha un bacino scolante approssimativamente pari a 1870 km^2 . La laguna è il prodotto di una coevoluzione soggetta a processi umani e naturali che ne hanno ripetutamente cambiato i caratteri idrodinamici e morfologici. L'influenza esercitata dall'uomo nel corso dei secoli ha portato il sistema in uno stato molto lontano da quello che si sarebbe prodotto naturalmente [L. D'Alpaos 2010]. L'interazione di fattori artificiali e naturali, in particolare, ha determinato una evoluzione morfologica evidente, anche rispetto alle condizioni di soli pochi decenni fa.

Un chiaro esempio di tale situazione è costituito dal diffuso abbassamento delle superfici rispetto al medio mare, frutto dell'effetto combinato della subsidenza, di un aumentato livello marino e di un diminuito apporto di sedimenti, causato quest'ultimo dall'allontanamento ad opera della Repubblica Serenissima dei più

importanti corsi d'acqua dal bacino lagunare, per prevenire il suo interrimento [L. D'Alpaos 2010].

Dal punto di vista morfologico, in alcune lagune (e in quelle dell'Alto Adriatico in particolare) si possono distinguere delle forme caratteristiche, riconducibili principalmente a tre diversi ambienti, il cui comportamento idrodinamico si differenzia per il diverso ruolo che vi assumono le forze che governano la propagazione dell'onda di marea [e.g. Straaten 1954]: le *barene*; i *bassifondi* e le *velme*; la rete dei *canali*.

Le barene. Di particolare interesse è il comportamento delle *barene*, caratteristiche formazioni morfologiche solitamente situate ai margini del bacino lagunare, anche se, a volte, come nel caso della Laguna di Venezia, fasce estese di queste strutture si interpongono tra le zone della laguna più prossime alle bocche a mare e gli specchi d'acqua più lontani, posti ai bordi del bacino stesso.

Le barene costituiscono la porzione topograficamente più elevata del bacino, trovandosi a quote superiori a quella del medio mare. Conseguentemente, esse sono periodicamente interessate dalle sole fasi di alta marea, che le sommergono durante la fase di flusso, scoprendole durante la successiva fase di riflusso. Si tratta, generalmente, di strutture piatte, caratterizzate, però, da un andamento altimetrico piuttosto tormentato per la presenza di una rete diffusa di piccoli canali, che le incidono e consentono alle correnti di marea di invaderle in modo regolare e progressivo.

Per la presenza di una ricca varietà di vegetazione alofila, in grado di crescere in zone soggette a prolungati periodi di sommersione e in terreni ad alto contenuto salino, nelle barene si riscontrano importanti interazioni tra i processi idrodinamici e quelli morfodinamici di trasporto e di deposito, insieme a quelli, affatto trascurabili, di produzione di suolo.

Da un punto di vista idrodinamico, la presenza, all'interno di una laguna, di ampie superfici di barena può comportare sensibili ritardi di fase nella propagazione dell'onda di marea ed apprezzabili attenuazioni della sua ampiezza. In particolare, dove la fascia di strutture morfologiche di questo tipo è più estesa e continua, come per Venezia nel caso della Laguna Nord, la marea si presenta ai bordi del bacino lagunare con ritardi di fase di quasi tre ore e con attenuazioni, rispetto al mare, sia sui colmi che sui cavi, che possono superare la decina di centimetri.

I bassifondi e le velme. Quote decisamente inferiori, rispetto alle barene, caratterizzano i *bassifondi* e le *velme*, che sono scoperte dalle acque solo nel caso di maree particolarmente basse. In queste porzioni di bacino, la pressoché continua presenza di acque, durante l'alternarsi delle maree, consente lo sviluppo di scarsa vegetazione (principalmente fanerogame), favorendo invece la presenza, sui fondali, di zone nelle quali la frazione sabbiosa cresce avvicinandosi alla bocca. La mancanza di vegetazione e i tiranti d'acqua più sostenuti fanno sì che, in questo ambiente, il flusso mareale sia controllato anche dai termini inerziali, contrariamente a quanto avviene sulle barene dove l'effetto dovuto alle resistenze al moto è decisamente prevalente.

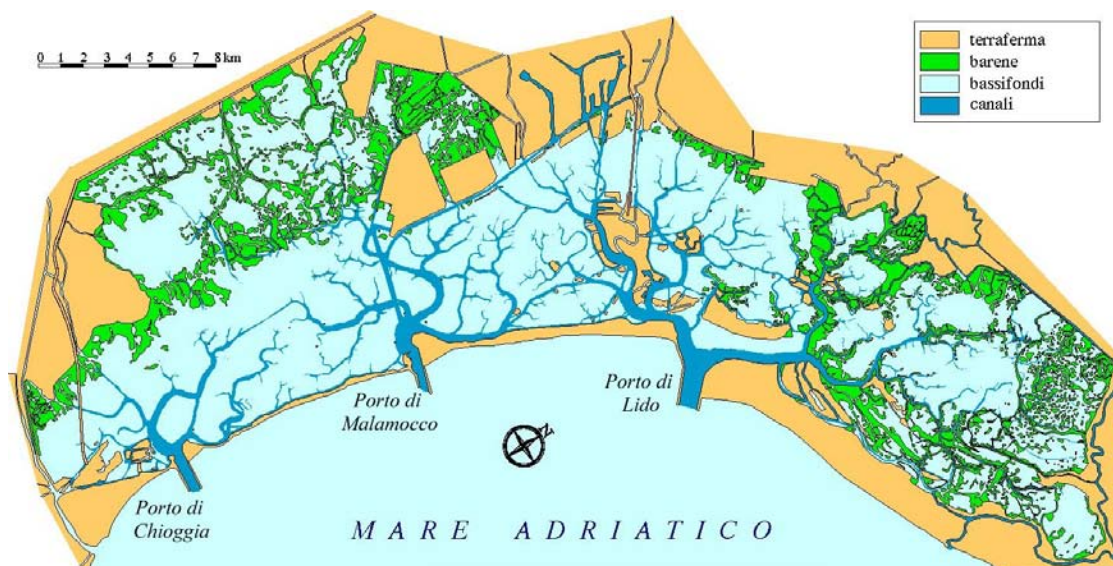


Figura 3.1: La laguna di Venezia.

I canali. Il terzo ambiente è rappresentato dalla rete dei *canali*. Questi, aumentando di sezione liquida mano a mano che ci si avvicina alle bocche di porto della laguna, costituiscono percorsi idrodinamici preferenziali per la propagazione dell'onda di marea, in fase di flusso, e per il drenaggio di barene e bassifondi, in fase di riflusso.

Guardando alle loro caratteristiche morfologiche e al loro ruolo nell'idrodinamica lagunare, i canali sono classificabili in due gruppi.

Il primo gruppo è costituito dai canali principali che si dipartono dalle bocche di porto e, con successive diramazioni, mettono in comunicazione il bacino con il mare. La struttura a rete di questi canali, più o meno articolata, controlla in larga misura la circolazione idrodinamica del sistema, favorendo, tra l'altro, gli scambi di sedimenti tra le barene e i bassifondi, tra i bassifondi e i canali stessi e, attraverso le bocche di porto, tra l'intero bacino lagunare e il mare.

Al secondo gruppo appartengono i canali minori che solcano le barene (i *ghebi*¹), i quali possono estinguersi quando raggiungono il confine con i bassifondi o proseguire fino a ricongiungersi con i canali presenti sui bassifondi stessi. La fitta rete di canali e di vie d'acqua minori, che innervano le barene, rende queste strutture molto "permeabili" al flusso di marea, permettendo la comunicazione diretta tra gli specchi d'acqua adiacenti alle barene, anche quando queste strutture risultano pressoché totalmente emerse.

È importante sottolineare che, come si è detto, la rete dei canali presenti in un bacino a marea determina la presenza di vie preferenziali, sia per il flusso d'acqua, che per quello dei sedimenti. Questi ultimi possono provenire dalle barene in seguito a fenomeni di erosione; essere messi in sospensione, per asportazione dai fondali e dalle scarpate, per l'effetto delle correnti litoranee in prossimità delle bocche di porto e del moto ondoso generato sulla superficie liquida dal vento o dai natanti in navigazione; oppure possono essere introdotti dai corsi d'acqua

¹Ghebi: espressione dialettale veneziana per indicare i canali minori.

affluenti nel bacino o come conseguenza dell'azione dell'uomo.

La Laguna di Venezia, attualmente, è collegata al mare Adriatico tramite le tre bocche di porto di Lido, di Chioggia e di Malamocco (si veda la Figura 3.1), le quali hanno subito, attraverso i secoli, importanti modificazioni per permettere la navigazione [L. D'Alpaos 2010, pp. 65–84, 193–230]. Il regime mareale è semidiurno con escursioni medie dell'ordine del metro. La sua morfologia è caratterizzata da una fitta rete di canali che si dipartono dalle bocche di porto e incidono i bassifondi e le barene. Questi canali presentano caratteristiche diverse l'uno dall'altro, riscontrabili in particolar modo nel grado di irregolarità e nei percorsi in cui curve e controcurve si presentano in posizione ravvicinata. In molti casi l'intervento dell'uomo ha modificato, tramite dragaggi, non solo i fondali dei canali principali, ma anche quelli dei bassifondi, realizzando canali artificiali che connettono tra di loro parti distinte della rete naturale.

3.2 L'area di studio: la Palude Pagliaga

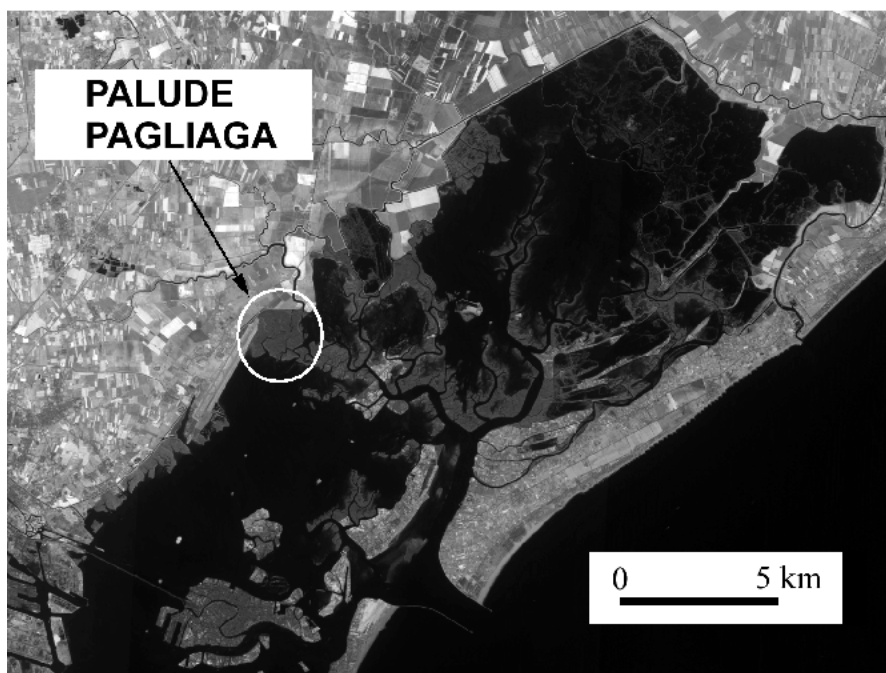
Volendo studiare la forma che assumono le aree di drenaggio afferenti ai vari canali che incidono le barene, una volta definito il modello in grado di determinare univocamente tali aree, è stato scelto di concentrare lo studio in una particolare zona della laguna di Venezia: la Palude di Pagliaga (figure 3.2, 3.3 e 3.4).

Viene denominato Palude di Pagliaga il complesso di piccole isole barenali che si trova nei pressi dell'aeroporto Marco Polo (si veda la Figura 3.2), ai confini nord-occidentali della laguna di Venezia, dove la resistenza al flusso della marea riduce di molto le normali oscillazioni di livello.

Situata, appunto, vicino al confine nord-ovest della laguna di Venezia, come è possibile vedere in Figura 3.4, la Palude Pagliaga è delimitata a nord dal canale Osellino e dallo sbocco del fiume Dese, a est da una zona di laguna denominata Palude di Cona e dalle valli da pesca di Ca' Deriva, a sud-est è chiusa dall'isola di Torcello e dal canale Dese, a sud-ovest è circondata dal bacino di Lido o Palude del Monte, mentre il confine occidentale è rappresentato dall'aeroporto di Venezia.

La palude di Pagliaga si presenta come un insieme di barene solcate da una fitta rete di canali, di cui i principali sono il Fossetta, il Terzo, il canale delle Cape, il ramo Passerini, il Berna e il Canale Dese (Canale Nuovo). L'origine della Palude di Pagliaga è fluviale, dovuta al costante afflusso di sedimenti portati dal fiume Dese, ma nel complesso l'estensione della superficie barenale non ha subito molti cambiamenti nel corso degli ultimi anni, mantenendosi relativamente costante, al contrario della rete di canali che si è ridotta. La sua elevazione varia da 0.10 a 0.45 *m s.m.m.*, mentre i canali che la solcano presentano profondità variabili comprese tra poche decine di centimetri e -1.50 *m s.m.m.* [Rado 2014, pp. 16–17].

All'interno della laguna, la Palude Pagliaga assume un importante ruolo ecologico poiché ospita ampie aree di *Fragmites australis*, una pianta di tipo glicofita, e molte piante alofite tra cui lo *Junctus Maritimum*, *Halimione Portulacoides* e la *Pucciniella Maritima*. Le uniche costruzioni presenti in questa porzione di



LAGUNA NORD DI VENEZIA

Figura 3.2: Immagine satellitare (Satellite SPOT1) della laguna nord di Venezia in cui si evidenzia la posizione della Palude di Pagliaga.



Figura 3.3: Vista da un aereo che atterra all'adiacente aeroporto Marco Polo della Palude di Pagliaga e della laguna di Venezia sullo sfondo. [Fonte <http://www.panoramio.com/photo/1580900>]

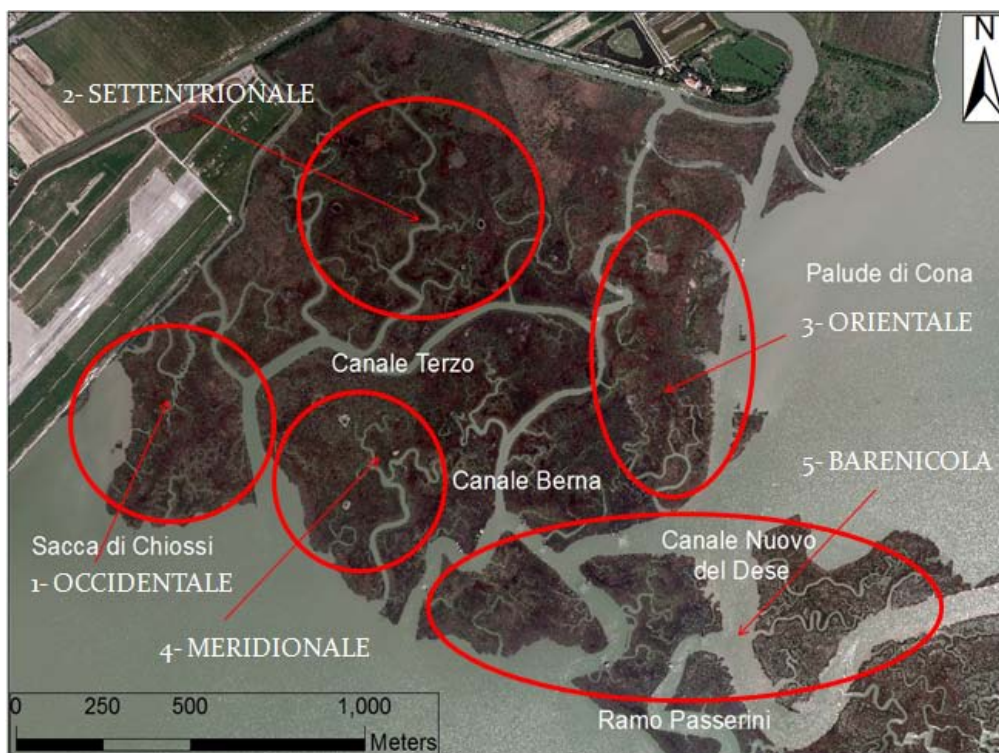


Figura 3.4: Immagine ad alta risoluzione della Palude Pagliaga con la classificazione delle varie aree barenali che la compongono.

laguna sono delle piccole strutture lungo il bordo del canale Berna, provviste di ampie reti (ben visibili in Figura 3.3).

La palude di Pagliaga è stata analizzata per diverse ragioni: innanzitutto, la grande disponibilità fotografica reperibile in rete e i modesti interventi antropici, che la zona ha subito nel corso del tempo, ne permettono un'indagine ampia e ricca, riguardante alcune reti di canali assolutamente naturali, dotate di una notevole complessità morfologica, in continua evoluzione e senza forzanti dovute all'attività umana. Inoltre, questa particolare porzione di laguna, studiata anche in passato da alcuni ricercatori dell'Università di Padova [Fagherazzi et al. 1999 e Marani et al. 2003], di recente, è stata oggetto di studio nell'ambito di una tesi di laurea, svolta con l'obiettivo di studiarne l'evoluzione temporale nel corso degli ultimi 60 anni [Rado 2014]. È, infatti, il lavoro appena citato la fonte per la creazione del file che viene utilizzato per imporre le condizioni al contorno, necessarie alla soluzione numerica dell'equazione (2.23a).

3.3 Condizioni al contorno e generazione del file di input con ArcGis

Nel paragrafo 2.3 è stato mostrato che, se si assume che lungo la rete canalizzata e nei bassifondi l'onda di marea si propaghi istantaneamente [Rinaldo et al. 1999a], la soluzione del problema della determinazione dell'andamento della superficie libera sulle barene risulta essere indipendente dalla profondità

dei canali e dei bassifondi, dipendendo solamente dalla struttura planimetrica della rete e delle barene stesse.

Sulla base della (2.23a), si può affermare che, una volta nota la configurazione planimetrica della superficie barenale che si vuole studiare e una volta imposte le corrette condizioni sul contorno della medesima superficie, la forma dei bacini afferenti ai vari canali incidenti la barena è univocamente determinata. Il problema che si pone è, quindi, quello della costruzione di uno strumento in grado di contenere le informazioni riguardanti la geometria del sito di studio e le relative condizioni al contorno, facilmente ottenibile e trattabile dal punto di vista numerico.

3.3.1 Condizioni al contorno

Per affrontare questa questione Rinaldo et al. [1999a] pensarono di schematizzare la geometria del sistema con una griglia a maglie quadrate, di dimensione costante e sufficientemente piccola da poter descrivere la morfologia dei canali, nella quale, nei vertici delle maglie, assegnare degli specifici valori identificativi a seconda che il *nodo* cadesse sulla barena, su un canale, su una zona di bassifondi o sulla terraferma.

I valori identificativi (*ID*) scelti furono:

- [0] per i punti sulle barene;
- [1] per i punti sulla terraferma o non soggetti ad allagamenti (contorni impermeabili);
- [2] per i punti in corrispondenza dei bassifondi;
- [3] per i punti canale;
- [4] per i punti appartenenti alla bocca di ogni canale².

Sfruttando la regolarità della griglia, per descrivere geometricamente una superficie lagunare è sufficiente memorizzare questo insieme ordinato di valori in una matrice, i cui indici di riga e di colonna fungono anche da coordinate del punto fisico corrispondente (scalate sulla dimensione della maglia).

Un esempio grafico della schematizzazione utilizzata per la Palude Pagliaga è illustrato in Figura 3.5.

In questo studio delle aree di drenaggio è stato scelto di mantenere la convenzione sopra descritta di Figura 3.5.

Utilizzando la schematizzazione planimetrica appena descritta e immaginando che a ogni singolo coefficiente della matrice corrisponda un pixel di un'immagine (Figura 3.5), risulta semplice l'imposizione delle condizioni al contorno:

²La necessità di identificare i punti della prima sezione di ogni canale con un ulteriore valore identificativo è dovuta non tanto all'imposizione delle condizioni al contorno qui trattata, ma all'implementazione del codice per il calcolo delle direzioni di drenaggio lungo la rete dei canali. A tale proposito si veda il paragrafo 4.1.2.

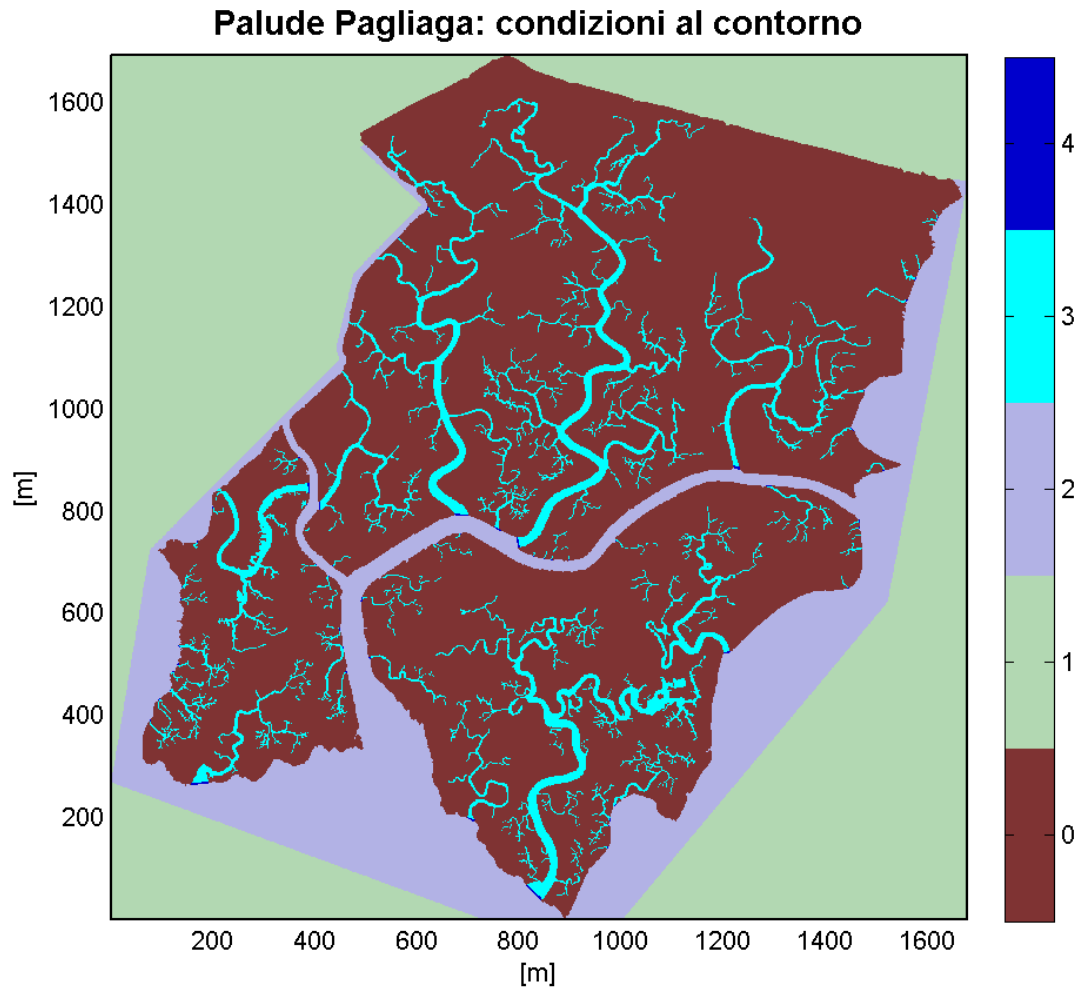


Figura 3.5: Rappresentazione bidimensionale della matrice delle condizioni al contorno per la Palude Pagliaga, determinata con una griglia a maglie quadrate di lato di dimensione costante e pari a 1 m, ottenuta in Matlab.

- tutti gli elementi della matrice contraddistinti dal valore [0] rappresentano i punti in cui si vuole calcolare il valore η_1 della quota della superficie libera (cfr. equazione (2.23a));
- in corrispondenza di tutti i "pixel" contenenti valore [0], posti sul contorno e confinanti con altri pixel rappresentati da numeri [1] si dovrà imporre la condizione di flusso nullo (2.23b) [condizione di *Neumann*];
- in corrispondenza di tutti i "pixel" contenenti valore [0], posti sul contorno e confinanti con altri pixel rappresentati da numeri [2], [3] o [4] si dovrà imporre la condizione $\eta_1 = 0$ (2.23c) [condizione di *Dirichlet*].

3.3.2 Il software ArcGis

Definita la modalità con la quale caratterizzare la struttura planimetrica del sistema lagunare che si vuole studiare, rimane il problema di ottenere, nel modo

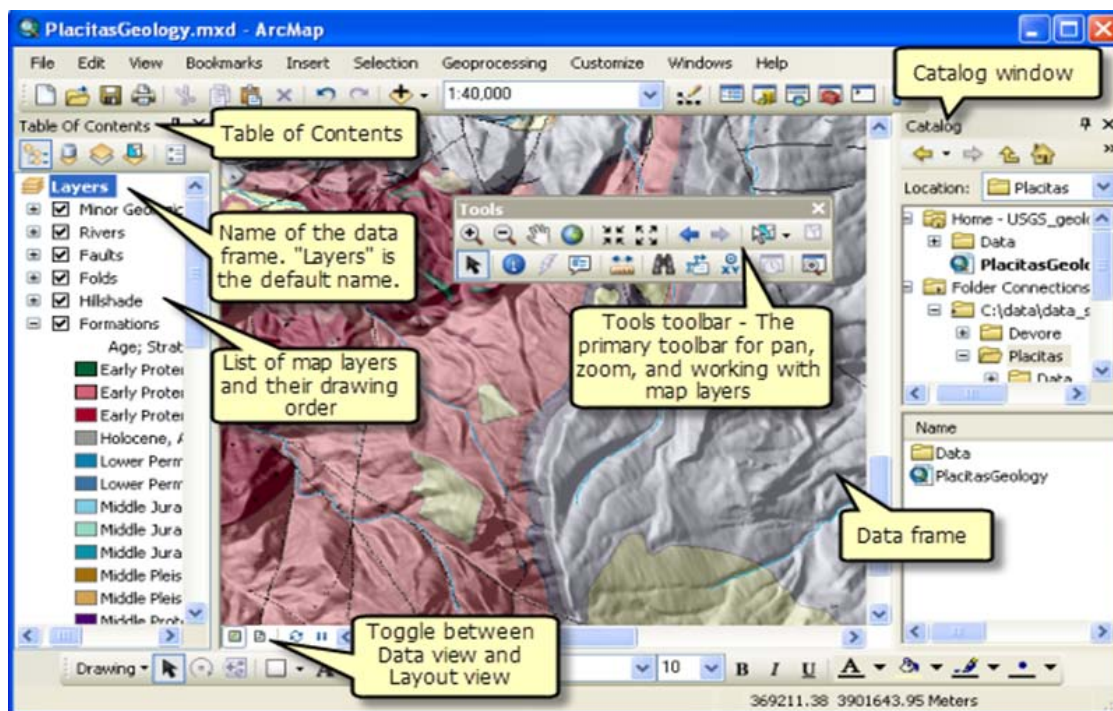


Figura 3.6: Finestra d'interfaccia di ArcMap con evidenziati alcuni dei principali elementi che la contraddistinguono.

più semplice possibile, la matrice necessaria all'implementazione del modello matematico.

Visto il grande sviluppo che ha, negli ultimi anni, contraddistinto il settore dei software di tipo GIS³, per la generazione dei file contenenti le condizioni al contorno, è stato utilizzato uno dei numerosi programmi di questo tipo disponibile in commercio: *ArcGis*.

In generale, un software Gis è un sistema informatico computerizzato che permette l'acquisizione, la registrazione e l'analisi di dati geografici e, di conseguenza, la visualizzazione e la restituzione delle immagini a essi associate. Si tratta, quindi, di un sistema informatico che permette di gestire e analizzare i dati spaziali, associando a ciascun elemento geografico descrizioni di tipo alfanumerico, favorendo l'elaborazione e la manipolazione di dati geometrici georeferenziati.

Un software di questo tipo è, quindi, una commistione tra strumenti informatici, cartografici e statistici: gli elementi cartografici sono le informazioni che permettono la georeferenziazione delle mappe e garantiscono la corrispondenza tra elementi grafici e la loro reale posizione, secondo un preciso riferimento geografico; gli elementi grafici sono determinati da combinazioni di enti geometrici primitivi (punto, linea, area, pixel, simbolo e annotazione grafica); infine, le informazioni statistiche sono ricavabili dagli attributi di ogni elemento, cioè da tutte le informazioni che ne permettono la rappresentazione [Rado 2014].

³GIS: *Geographic Information System* [esri 2014]

ArcGis, in particolare, è una raccolta di applicazioni che permette di elaborare in molti modi i dati topografici georeferenziati che, sempre di più, è possibile reperire gratuitamente in Internet. L'applicazione principale del programma si chiama ArcMap e viene utilizzata per la mappatura, l'editing, l'analisi e la gestione dei dati, utile, soprattutto, per la creazione di file vettoriali per la rappresentazione. Nel caso in esame, viene quindi utilizzato per la rappresentazione di canali e barene.

ArcMap, la cui interfaccia grafica è mostrata in Figura 3.6, è l'ambiente di ArcGis in cui si visualizzano e si esplorano i dati Gis per l'area di studio, si assegnano i simboli e si creano i layout di mappa per la stampa o la pubblicazione. Si presenta come un foglio bianco (che rappresenta la superficie terrestre) su cui è possibile iniziare nuovi progetti o caricare progetti già esistenti.

Tra le diverse funzionalità che l'applicazione fornisce, si ricordano le seguenti:

- visualizzazione delle proprietà della struttura dei dati e dei layer importati;
- creazione, editing o rimozione di layer di varie tipologie, tramite strumenti per il disegno e l'elaborazione degli oggetti geometrici che compaiono nel programma;
- applicazione di numerosi strumenti di elaborazione dei dati, tramite varie *toolbox*, che permettono di modificare, esportare o trasformare in nuovi formati gli elementi contenuti nei layer;
- impostazione del layout di stampa, salvataggio e pubblicazione dei progetti.

Volendo descrivere più nel dettaglio le funzionalità di questo programma, si può dire che in ArcMap ci sono due tipi di visualizzazione: la visualizzazione dell'elenco dei dati e la visualizzazione dei layout generati dai dati stessi. La prima tipologia di visualizzazione è la *Table of Contents* (tabella dei contenuti), visibile sulla sinistra della Figura 3.6. Essa fornisce una mappa della struttura dei dati raccolti come elenco di livelli e permette di scegliere quali di questi dati nascondere, rendere visibili o rimuovere. Nella *Table of Contents* è possibile visualizzare le proprietà dei singoli layer e, in caso, raggrupparli se presentano affinità (e.q., nel caso di rielaborazioni di uno stesso oggetto). L'ordine in cui i layer sono riportati nella tabella dei contenuti influenza anche la visualizzazione del layout; infatti, i layer che stanno sopra nella *Table of Contents* saranno visualizzati in primo piano nella finestra di layout (nota anche come *Data Frame*). Il *Data Frame* consente la visualizzazione di una pagina in cui gli elementi della mappa sono disposti e sviluppati, pronti per la stampa.

In ArcMap è anche possibile editare layer di diverso tipo e, in particolare, quelli che contengono degli *shape file*, ovvero quelli che contengono le forme geometriche disegnabili, quali possono essere linee, polilinee, poligoni ed altri elementi ancora. Con ArcMap, quindi, è possibile importare un layer e lavorarci sopra: questa fase, detta digitalizzazione, consente di disegnare o modificare il layer scelto, salvare i progressi fatti e, ovviamente, visualizzarli. Ad ogni layer vettoriale, all'interno del relativo *shape file*, è associata una tabella di attributi, cioè un insieme di dati che definisce in modo univoco la caratterizzazione geometrica

dell'elemento e che permette al programma di calcolarne alcune grandezze, come le lunghezze delle polilinee o l'area dei poligoni selezionati, fornendo a chi lavora utili informazioni riguardo alle caratteristiche degli elementi che sta utilizzando.

Un'ultima, ma non meno importante, possibilità che si ha lavorando con ArcMap è quella di usufruire delle numerose *toolbox* implementate dagli sviluppatori del software, le quali permettono di effettuare modifiche e rielaborazioni che risulterebbero difficili, se non impossibili, con i classici strumenti per l'editing dei layer. Queste *toolbox*, per la maggior parte già comprese nel programma originale, aumentano di molto le potenzialità dell'ambiente ArcMap.

In questo ambiente di lavoro, Marco Rado [Rado 2014], con l'intento di studiare l'evoluzione temporale delle reti di canali a marea nella laguna di Venezia, ha importato, georeferenziato e digitalizzato numerose immagini aeree della Palude di Pagliaga, rappresentando i contorni delle barene con numerose polilinee. Questa digitalizzazione è qui stata utilizzata come punto di partenza per la procedura di estrazione della matrice delle condizioni al contorno, descritta nel seguente paragrafo.

3.3.3 Imposizione delle condizioni al contorno con ArcMap

Avendo a disposizione la digitalizzazione delle barene della Palude Pagliaga, effettuata utilizzando foto aeree del 2010 (cfr. Figura 3.7 e Rado [2014, pp. 95–132]), si vogliono riutilizzare le polilinee, che definiscono i contorni delle barene, per generare la matrice delle condizioni al contorno descritta nel paragrafo 3.3.1 e rappresentata in Figura 3.5. In questo paragrafo si descrive la procedura utilizzata, sfruttando le potenzialità di ArcMap, per ottenere tale matrice.

Procedendo schematicamente per una maggior chiarezza, i passaggi necessari sono i seguenti.

1. Il primo passo consiste nella revisione e nel completamento delle polilinee ottenute dalla digitalizzazione delle barene, utilizzando i classici strumenti di editing delle forme geometriche (simili a quelli disponibili in molti altri programmi per il disegno tecnico). Confrontando le figure 3.7 e 3.8, si può immediatamente notare quali siano state le modifiche effettuate:
 - sono state eliminate le polilinee che contornavano le barene delle zone a sud-est dell'area di studio analizzata;
 - sono state individuate e chiuse, con due polilinee parallele e ravvicinate, le bocche dei canali che drenano le barene considerate, per l'assegnazione del numero identificativo ($ID = 4$) che indica l'inizio dei canali (cfr. par. 3.3.1 p. 23);
 - è stata disegnata una linea che delimita l'area di studio lungo i contorni est, sud e ovest, per delimitare le zone in cui la barena è in diretta comunicazione con lo specchio liquido lagunare ed in cui si deve assegnare la condizione $ID = 2$;

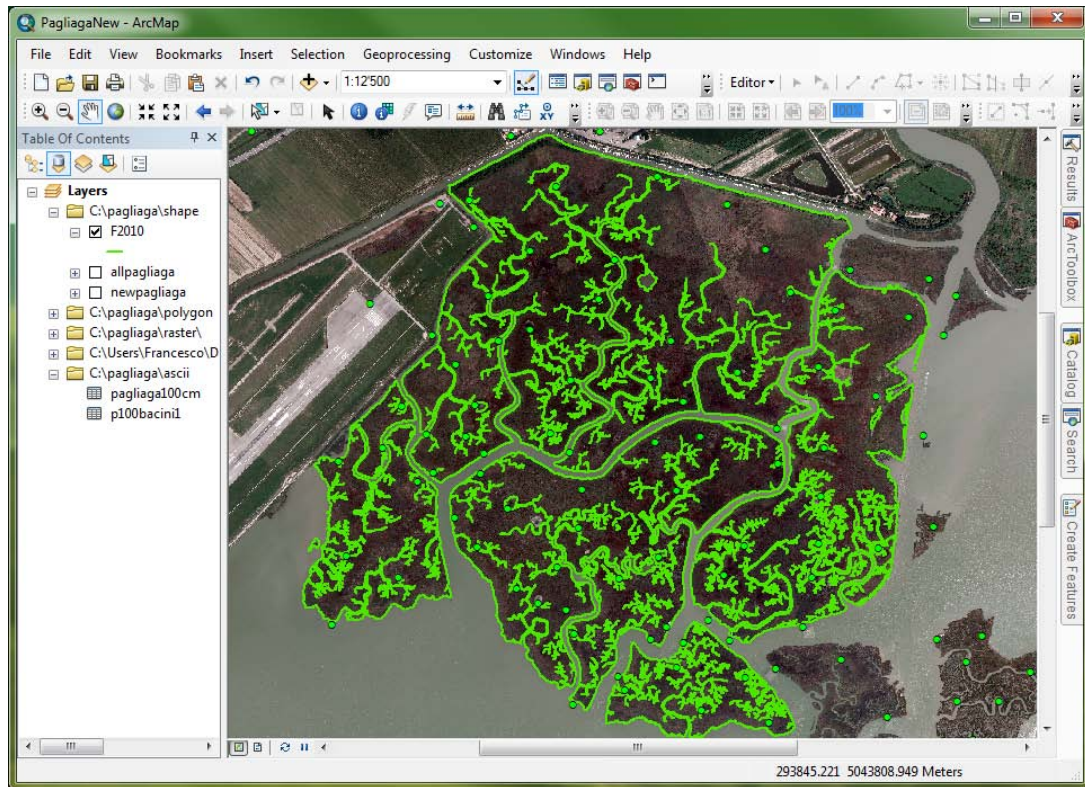


Figura 3.7: Schermata di ArcMap in cui si visualizza la digitalizzazione delle barene della Palude Pagliaga effettuata da Marco Rado [Rado 2014].

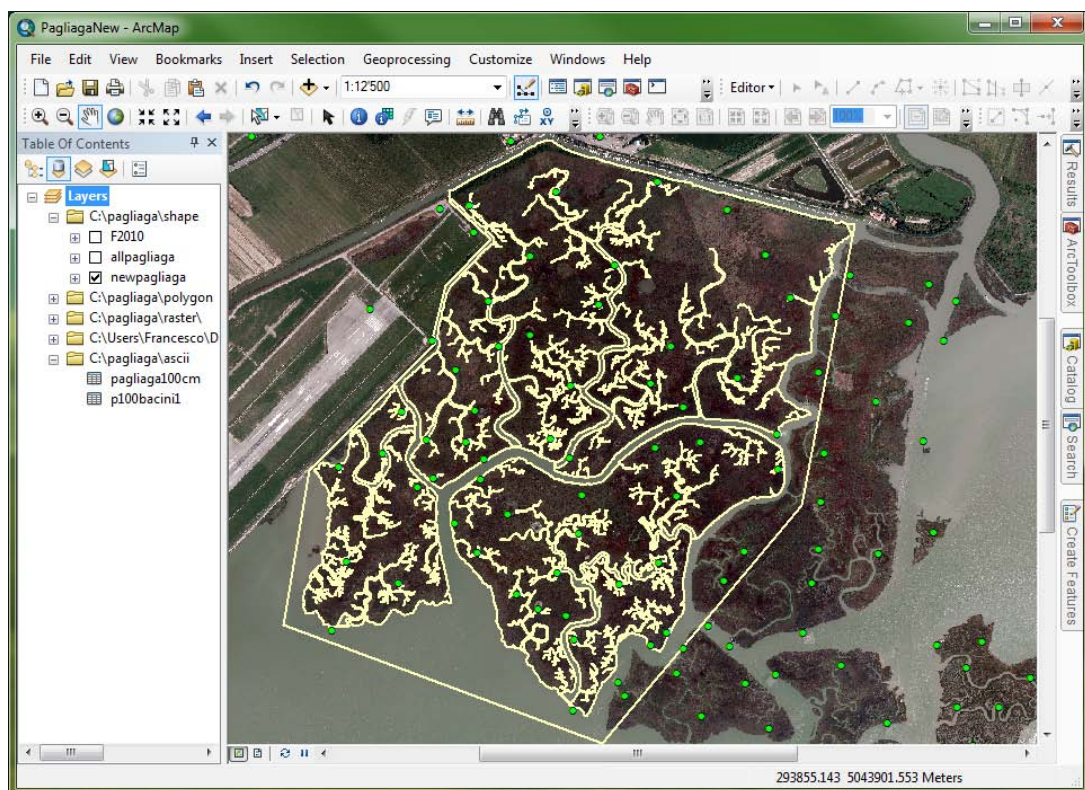


Figura 3.8: Schermata di ArcMap in cui si visualizzano le polilinee modificate per imporre le condizioni al contorno per la soluzione dell'equazione (2.23).

- sono state eliminate e chiuse le polilinee che delimitavano i chiari⁴;
- è stato verificato che tutte le polilinee fossero “chiuse”, ovvero che delimitassero esattamente delle aree definite, senza lasciare spazi aperti. Questo particolare accorgimento è legato al fatto che, come si spiega nel punto 2, queste polilinee devono essere trasformate in poligoni.

2. Ottenute delle polilinee chiuse che delimitano barene, bassifondi e canali, è necessario trasformarle in poligoni, col perimetro coincidente con i confini definiti dalle polilinee. Per farlo è stata utilizzata la funzione *Feature to Polygon* (Figura 3.9), presente nella toolbox **ArcToolbox/Data Management Tools/Features**, scegliendo come input le polilinee e lasciando il valore di default di 0.001 m per l'*XY Tolerance*. In questo modo si ottiene uno *shape file* contenente un insieme di poligoni come quello rappresentato in Figura 3.10.

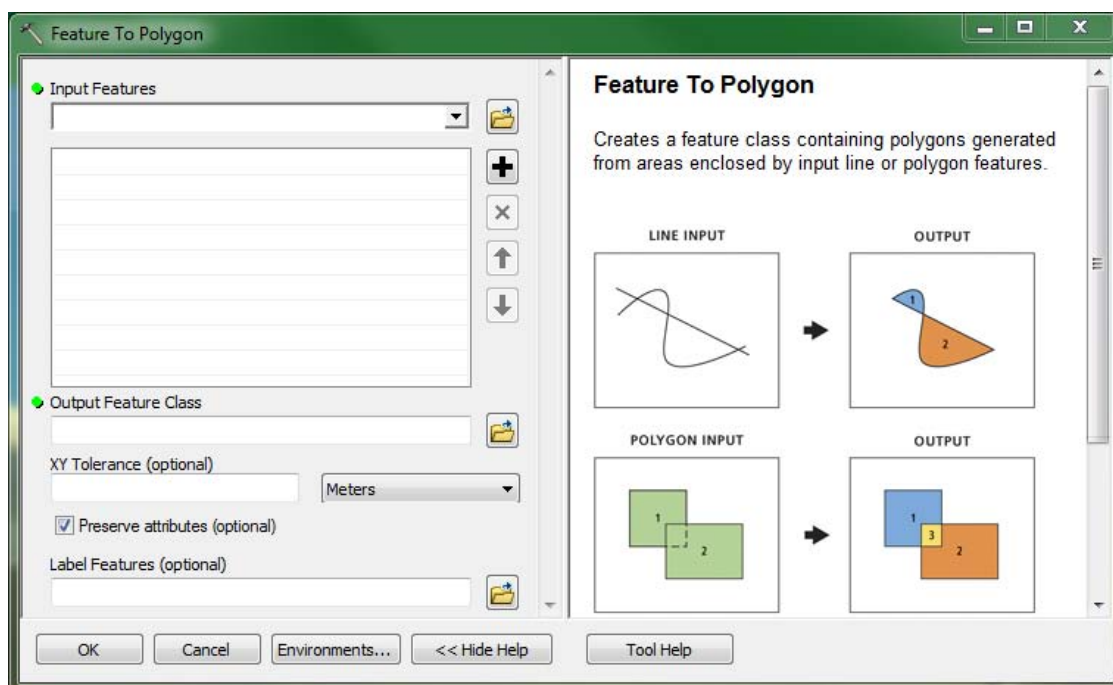


Figura 3.9: Finestra di dialogo di ArcMap per l'esecuzione di *Feature to Polygon*.

3. Ora è necessario trasformare questo insieme di poligoni in un'immagine di tipo *raster*.⁵ Per farlo si ricorre alla funzione *Polygon to Raster* (Figura 3.11), della toolbox **ArcToolbox/Conversion Tools/To Raster**, specificando nella finestra di dialogo le seguenti opzioni:

- *Value field*: FID;
- *Cell assignment type*⁶: CELL_CENTER;

⁴Chiari: piccoli specchi d'acqua piovana o salmastra, imprigionati all'interno delle barene.

⁵Immagine *raster*: immagine composta da una *griglia* regolare di pixel colorati.

⁶Il significato di questo parametro è descritto nella parte destra della Figura 3.11.

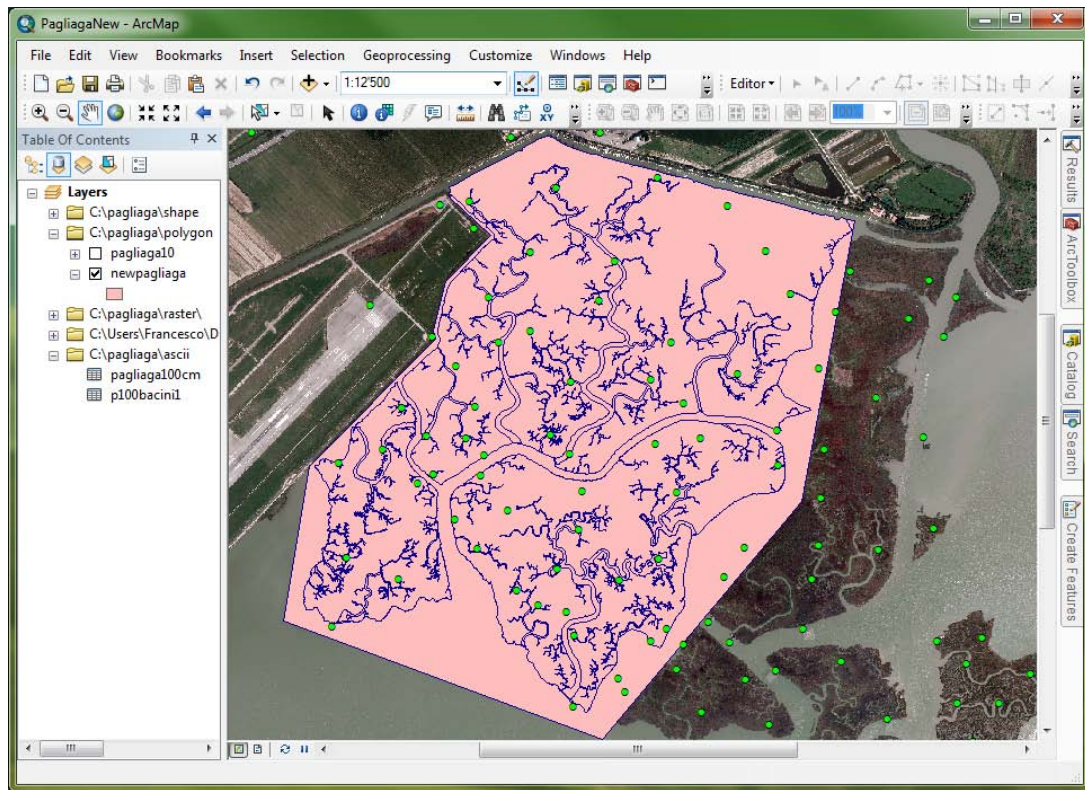


Figura 3.10: Schermata di ArcMap in cui si visualizzano i poligoni ottenuti dalle polilinee di Figura 3.8.

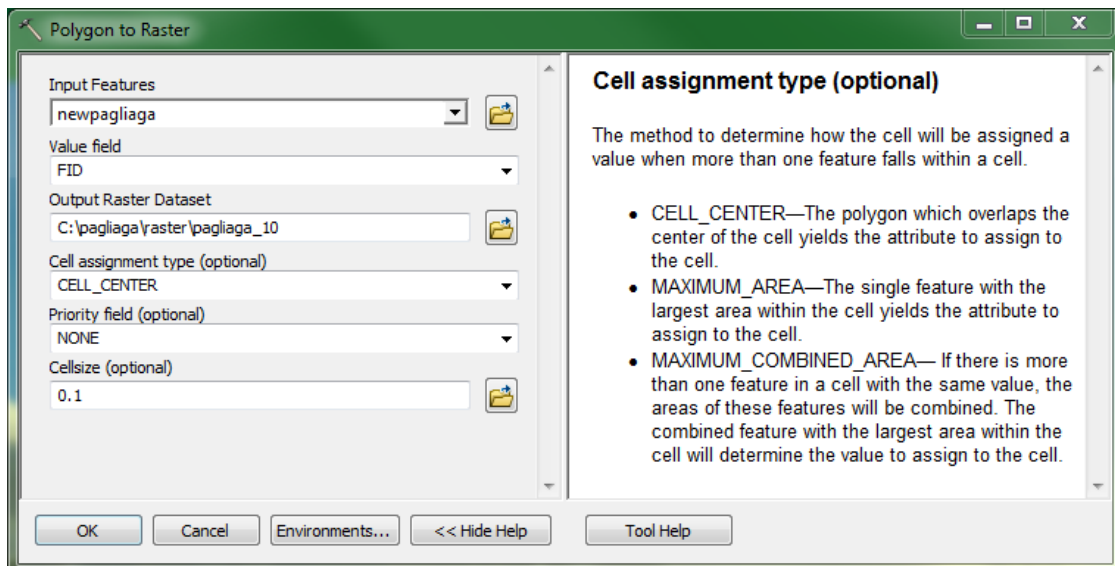


Figura 3.11: Finestra di dialogo di ArcMap per l'esecuzione di *Polygon to Raster*.

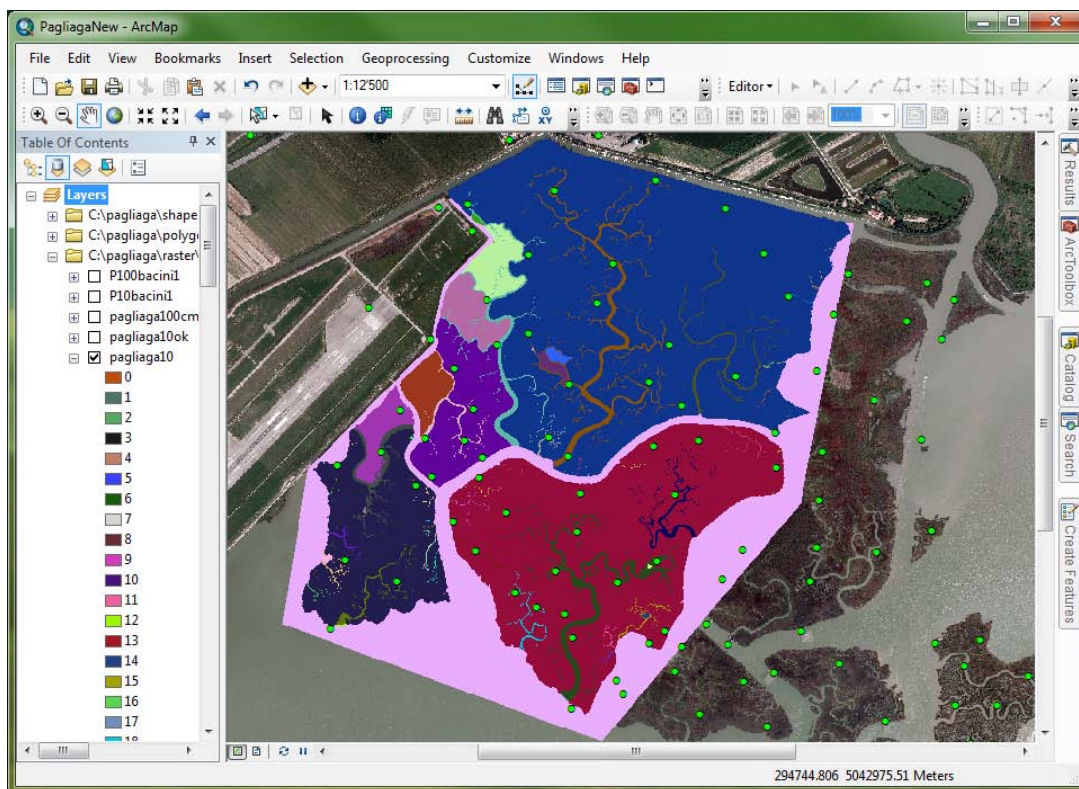


Figura 3.12: Schermata di ArcMap in cui si visualizza l'immagine raster ottenuta dai poligoni di Figura 3.10.

- *Priority field*: NONE;
- *Cellsize*: una misura in metri, sufficientemente piccola da descrivere la configurazione planimetrica dei rami terminali dei canali lagunari. Nel caso in esame, come mostrato in Figura 3.11, è stato scelto di usare 0.1 m.

Il risultato di tale funzione è un'immagine raster in cui, ad ogni poligono dello *shape file* dato come input, viene assegnato un colore e, quindi, un valore numerico diverso. Un esempio lo si può vedere in Figura 3.12.

4. Per ottenere un immagine raster simile a quella di Figura 3.12, con i pixel che la compongono della medesima dimensione, ma con i valori relativi alle condizioni al contorno assegnate, si usa la funzione *Reclassify* (Figura 3.13). Questa, contenuta nella toolbox **ArcToolbox/Spatial Analyst Tools/Reclass**, permette di sostituire i valori associati a ogni singolo pixel di un'immagine raster con altri valori numerici, definendo le volute sostituzioni in un'apposita tabella che compare nella finestra di dialogo. Per eseguire correttamente questa sostituzione è necessario assegnare l'opzione VALUE al campo *Reclass field*, selezionare il pulsante *Unique* e compilare la tabella associando a ogni vecchio valore quello della rispettiva condizione al contorno. Per completare correttamente questa operazione è bene ricordare di:

- assegnare il valore 1, corrispondente alla condizione di flusso nullo, al campo contraddistinto da **NoData** nel raster in input;
- procedere con ordine alla compilazione dei vari campi della tabella. Infatti, come si può vedere in Figura 3.13, può capitare che i campi da riclassificare siano numerosi: per far ciò può essere utile partire dalla fine della tabella ricordando che questa è organizzata in modo tale che, procedendo dalla fine all'inizio, i campi che si incontrano sono quelli visibili nella schermata principale di ArcMap (Figura 3.11), procedendo dall'angolo in alto a sinistra a quello in basso a destra.

Il risultato della riclassificazione appena descritta è riportato in Figura 3.14.

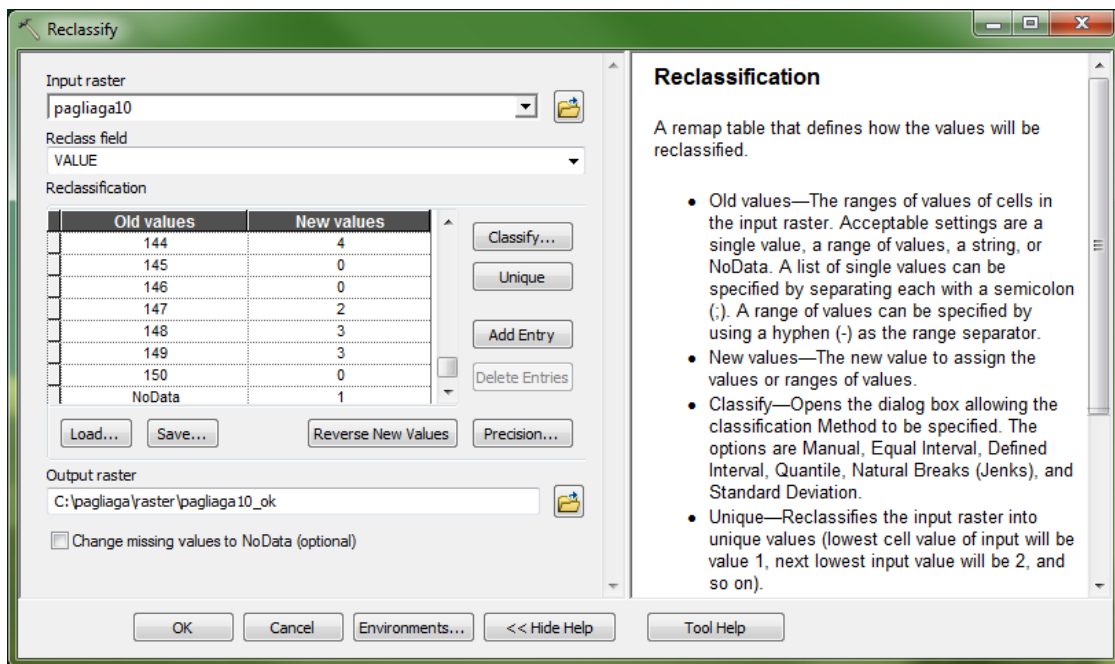


Figura 3.13: Finestra di dialogo di ArcMap per l'esecuzione di *Reclassify*.

5. Prima poter trasformare in file di testo il raster ottenuto (Figura 3.14), è necessario un ulteriore passaggio. Infatti, l'immagine riclassificata pur rappresentando correttamente la griglia delle condizioni al contorno, è caratterizzata da un'alta risoluzione che comporterebbe un eccessivo onere di calcolo, se rapportato al problema che si vuole studiare. Per ridurre la risoluzione della griglia (i.e. aumentare la dimensione del lato della maglia quadrata che la contraddistingue), è possibile ricorrere alla funzione *Aggregate* (Figura 3.15) della toolbox **ArcToolbox/Spatial Analyst Tools/Generalization**, specificando le opzioni:

- *Cell factor*: il numero per il quale moltiplicare la dimensione del pixel dell'immagine raster in input. Questo numero deve essere tale da ridurre il numero di pixel necessari a descrivere l'intera area di studio, mantenendo, però, una dimensione coerente con le dimensioni delle

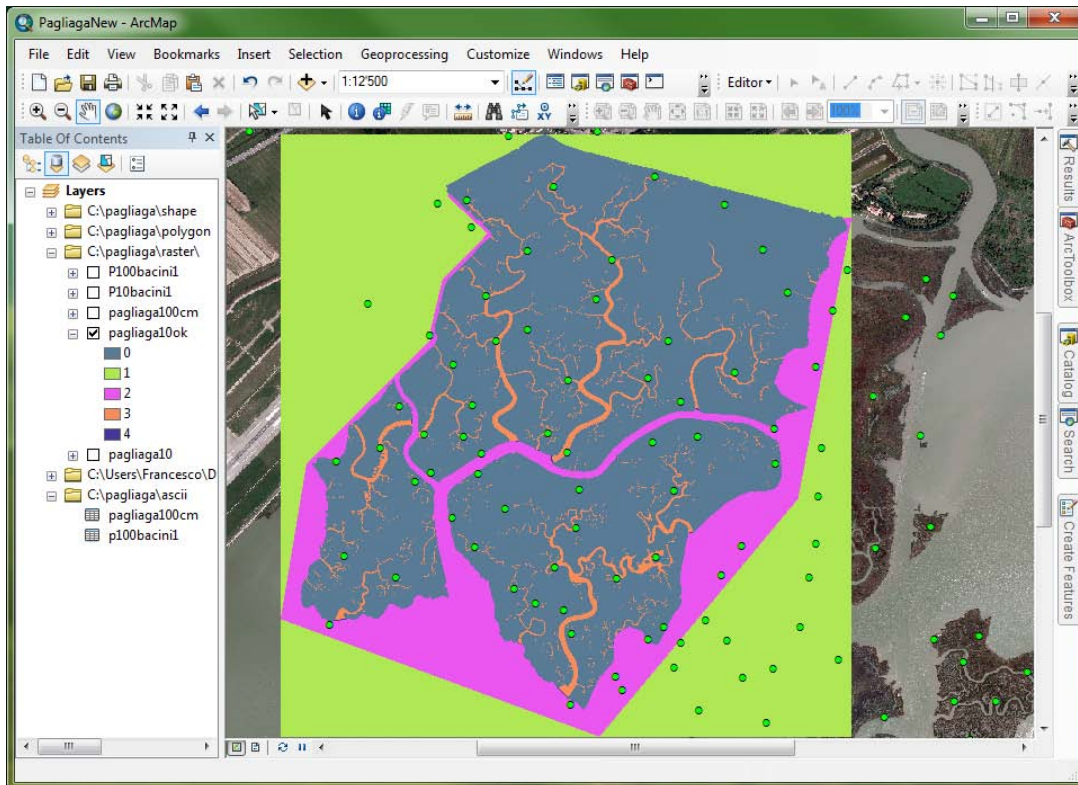


Figura 3.14: Schermata di ArcMap in cui si visualizza l'immagine raster ottenuta dalla riclassificazione del raster di figura 3.12.

parti terminali dei canali che si vogliono discutere. Nel caso in esame è stato scelto di utilizzare un *Cell factor* pari a 10, così da ottenere una griglia di lato 1 m;

- *Aggregation technique: MAXIMUM*. Questa opzione (la cui descrizione è riportata anche nel lato destro di Figura 3.15), definisce come la funzione debba procedere nella determinazione dei valori numerici da assegnare ai pixel della nuova immagine raster. Questa operazione risulta possibile anche grazie alla scelta dei numeri identificativi per l'assegnazione delle condizioni al contorno vista nel paragrafo 3.3.1. Infatti, avendo gli *ID* che individuano i canali i valori maggiori, scegliendo l'opzione *MAXIMUM*, la rete canalizzata risulterà sempre connessa⁷, condizione necessaria per il corretto funzionamento del codice di calcolo con cui viene risolto il modello matematico descritto nel Capitolo 2.

Il risultato che si ottiene con questa funzione, impostando le due opzioni appena descritte, è illustrato in Figura 3.16.

6. Raggiunto il risultato desiderato, è sufficiente trasformare l'immagine di Figura 3.16 in una matrice sfruttando la funzione *Raster to ASCII* della

⁷La rete, per come è stata costruita, sarà sempre connessa: ovvero non ci saranno pixel canale "staccati" da quelli costituenti i canali principali.

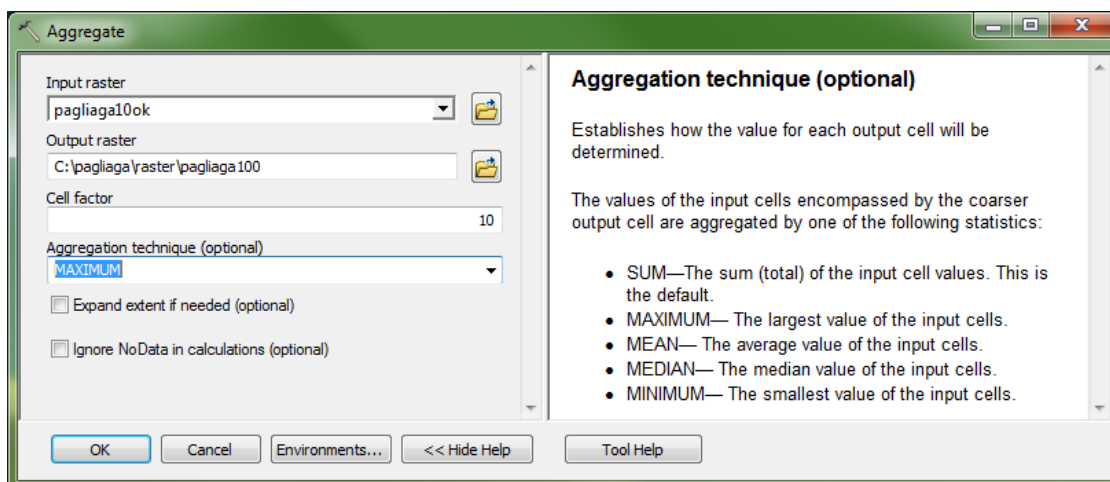


Figura 3.15: Finestra di dialogo di ArcMap per l'esecuzione di *Aggregate*.

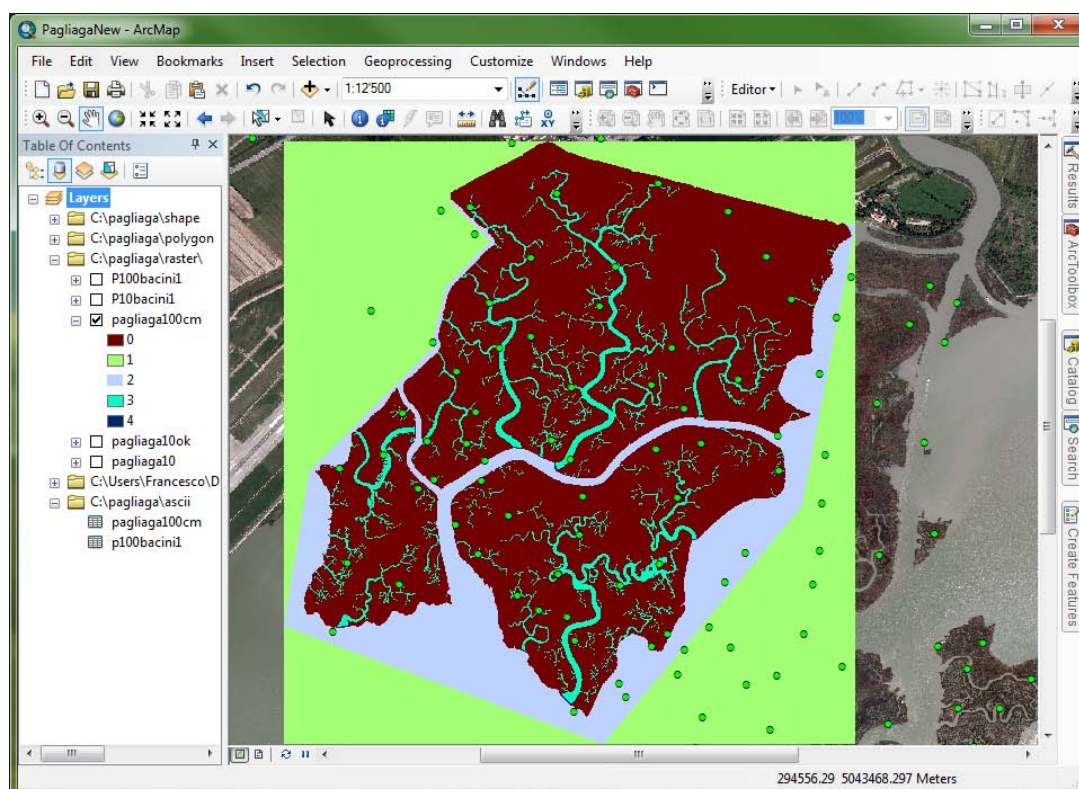


Figura 3.16: Schermata di ArcMap in cui si visualizza l'immagine raster ottenuta dal raster di figura 3.14, con la toolbox *Aggregate*.

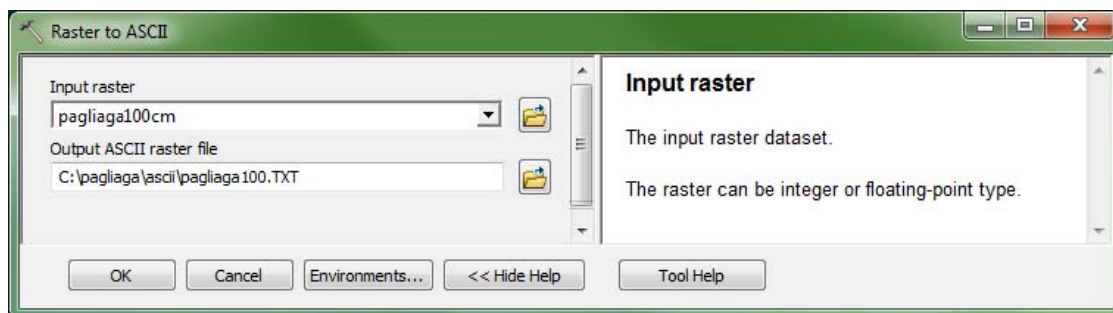


Figura 3.17: Finestra di dialogo di ArcMap per l'esecuzione di *Raster to ASCII*.

toolbox **ArcToolbox/Conversion Tools/From Raster** (Figura 3.17), con la quale si ottiene un file di testo contenente la matrice delle condizioni al contorno, preceduta da sette righe contenenti le informazioni riguardanti le caratteristiche della matrice stessa: numero di colonne, numero di righe, le coordinate spaziali per la georeferenziazione, la dimensione delle celle costituenti la griglia (in metri) e il valore assegnato agli eventuali dati mancanti⁸.

Nel caso in esame è stata ottenuta una matrice di 1769 righe e 1678 colonne, in cui la dimensione (costante) della maglia quadrata è 1 m. Una rappresentazione della matrice in questione è stata riportata nella Figura 3.5.

3.4 Considerazioni conclusive sul procedimento proposto

In questo capitolo, dopo una veloce presentazione dell'ambiente lagunare e dell'area di studio analizzata, è stato trattato il tema dell'assegnazione delle condizioni al contorno del modello matematico, descritto nel Capitolo 2. In particolare, è stato prima descritto il modo in cui le condizioni al contorno sono assegnate nel modello (paragrafo 3.3.1), ed è stata quindi definita una procedura che sfrutta le potenzialità di ArcMap per la creazione della matrice necessaria all'implementazione del modello stesso (paragrafo 3.3.3).

Considerando il risultato ottenuto con la procedura proposta, si possono trarre le seguenti conclusioni.

- Il metodo presuppone la digitalizzazione delle barene. Se questa è già stata eseguita con una buona accuratezza e con una buona organizzazione delle polilinee utilizzate, allora la procedura proposta risulta semplice e di rapida attuazione. La digitalizzazione della rete di canali incidenti le barene può essere fatta seguendo una procedura analoga a quella recentemente eseguita per caratterizzare la Palude Pagliaga e altri due siti della laguna di Venezia [Rado 2014].

⁸Nel file sono chiamati NODATA_value: nel caso in esame è possibile fare un primo controllo di verifica del procedimento assicurandosi che nella matrice risultante non siano presenti dati mancanti.

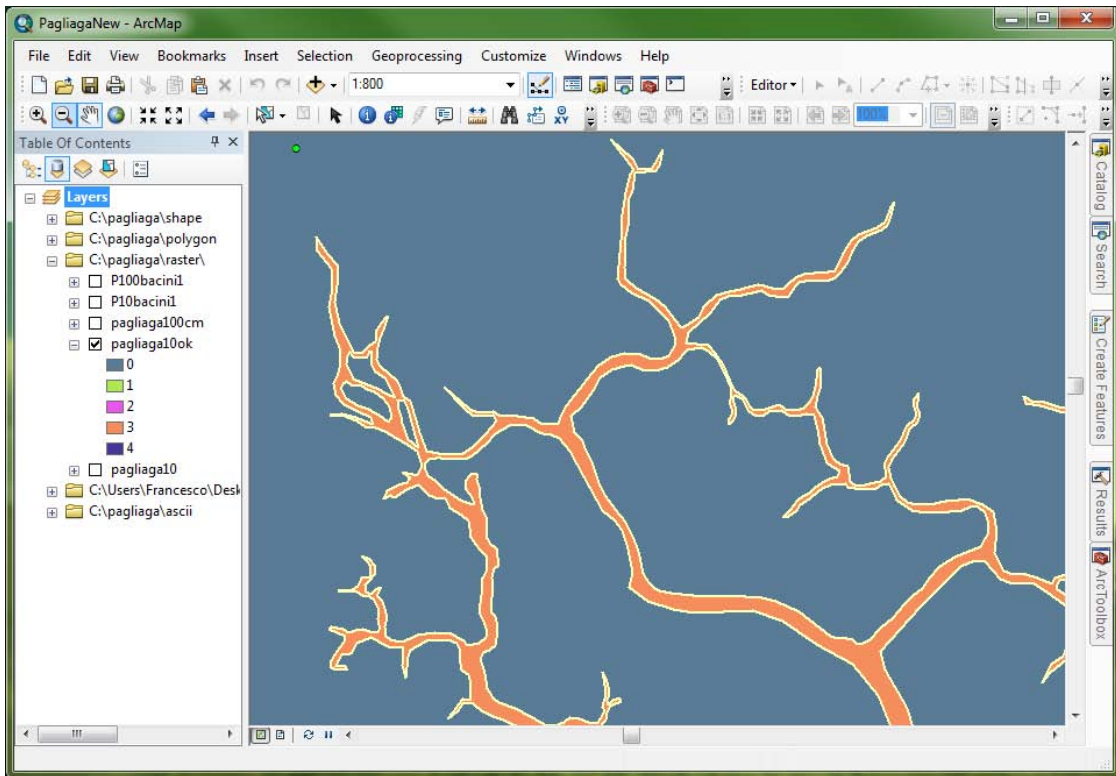


Figura 3.18: Schermata di ArcMap in cui si visualizza un particolare dell'immagine raster di Figura 3.14.

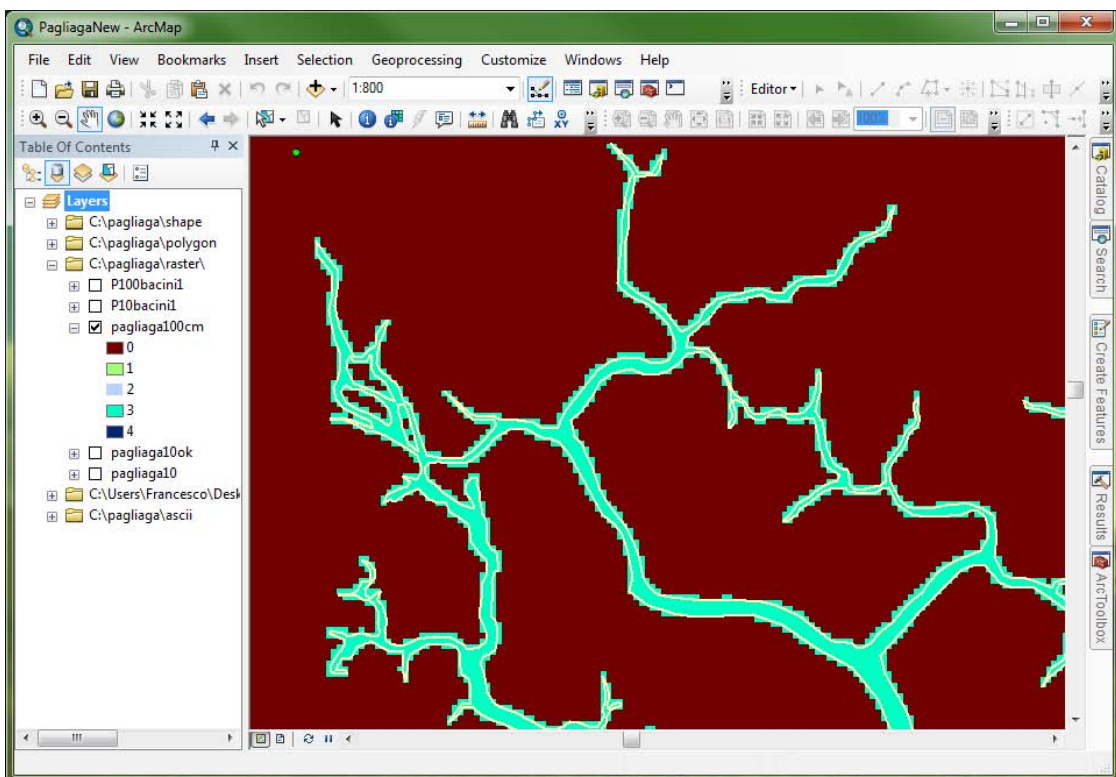


Figura 3.19: Schermata di ArcMap in cui si visualizza un particolare dell'immagine raster di Figura 3.16.

- Applicando la successione di comandi definita nel paragrafo 3.3.3, si ottiene una matrice che può essere utilizzata come file di input del programma che implementa il modello matematico descritto nel Capitolo 2.
- La matrice così ottenuta può essere affetta da piccoli errori di approssimazione, che aumentano d'importanza al diminuire della risoluzione dell'immagine che si ottiene al punto 5 della sequenza descritta nel paragrafo 3.3.3. Un'idea di questi errori d'approssimazione è data dall'analisi dello stesso dettaglio della rete canalizzata, prima della riduzione di risoluzione (Figura 3.18) e dopo tale modifica (Figura 3.19). Dal confronto delle due immagini, rappresentanti una situazione relativamente complessa, si possono notare alcune piccole imperfezioni nella descrizione della rete dei canali dovuta alla diminuzione di risoluzione. Tali imperfezioni, tuttavia, risultano pressoché ininfluenti nella definizione delle aree di drenaggio.

Capitolo 4

L'implementazione del modello in Matlab

Nei capitoli 2 e 3 sono state definite le basi sulle quali si fonda un metodo per l'individuazione univoca della forma delle aree di drenaggio dei canali a marea incidenti le barene di una laguna.

Volendo analizzare quantitativamente la forma dei bacini lagunari della Palude Pagliaga (par. 3.2), in questo capitolo:

- Viene brevemente richiamato il funzionamento del modello matematico sviluppato da Rinaldo et al. [1999a], e successivamente aggiornato da A. D'Alpaos [2001]. In particolare,
 - si descrive come è stata risolta numericamente l'equazione (2.23);
 - si descrive come il modello, sulla base della soluzione numerica ottenuta, determina le direzioni di drenaggio sulle barene e lungo la rete dei canali;
 - si descrive come, note le direzioni di drenaggio, il modello definisce le aree afferenti ai canali.
- Si propone un metodo per importare in Matlab il codice di calcolo (sviluppato in Fortran) che risolve le equazioni del modello matematico. In particolare,
 - si descrivono le motivazioni che hanno spinto all'uso del linguaggio Matlab;
 - si introducono i MEX-File e se ne spiega l'utilità;
 - si descrive come ottenere un MEX-File da una subroutine Fortran esistente;
 - si descrivono le funzioni Matlab implementate, nell'intento di ottenere una toolbox di funzioni per la definizione della forma delle aree di drenaggio afferenti ai vari canali.

4.1 Il modello in Fortran

Il modello matematico esposto nel Capitolo 2 è stato risolto numericamente alle differenze finite utilizzando una griglia di calcolo con maglie quadrate di lato $\Delta x = \Delta y$. Il codice di calcolo che consente di risolvere numericamente l'equazione (2.23) e, quindi, di determinare l'andamento della superficie libera (e, quindi, le direzioni di drenaggio) al di sopra di una determinata barena è stato scritto in linguaggio Fortran [Rinaldo et al. 1999a].

Il codice è costituito da un programma principale e da alcuni sottoprogrammi (*subroutine*¹). Questo insieme di programmi ha costituito il punto di partenza per il lavoro esposto nel seguito di questo capitolo.

Nel programma principale le operazioni che vengono eseguite sono:

- l'inizializzazione del programma;
- la dichiarazione delle variabili;
- l'apertura dei file di input ed output;
- la lettura dei dati iniziali dai file che li contengono;
- un controllo di coerenza dei dati;
- l'inizializzazione delle variabili da determinare nel calcolo;
- la chiamata delle varie subroutine per il calcolo delle variabili di interesse;
- il salvataggio dei file di output.

Come si può vedere da questo elenco, il calcolo delle variabili presenti nel modello è effettuato dalle subroutine. Perciò la prima parte di questo capitolo sarà dedicata alla loro descrizione.

4.1.1 La subroutine *telone*

Tra tutte le subroutine, quella denominata *telone*² è la più importante, ed è quella che richiede il maggior onere di calcolo. Infatti in essa si effettua il calcolo della soluzione dell'equazione (2.23).

In questo paragrafo si illustra il modo di procedere implementato nella subroutine *telone*: prima in termini generali, poi proponendo un esempio di come viene determinata la soluzione, nel caso di una matrice delle condizioni al contorno molto semplice (cfr. matrice (4.4)).

¹Le *subroutine* in Fortran si differenziano da altri sottoprogrammi definiti come *function*. Differenza importante nell'ottica di quanto si espone riguardo ai MEX-File, nel paragrafo 4.2.

²La subroutine che risolve il problema di Poisson, $\nabla^2 \eta = K$, venne denominata *telone* poiché il problema risolto è analogo a quello di un telone gonfiato da sotto, ancorato lungo una particolare porzione del contorno e lasciato libero lungo il restante perimetro.

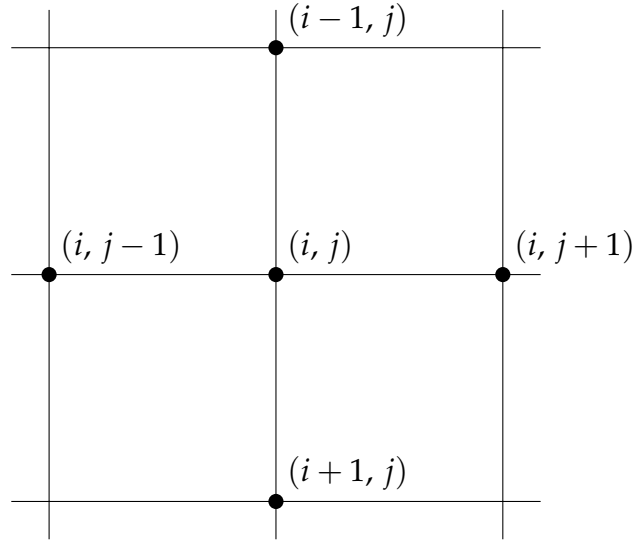


Figura 4.1: Rappresentazione del generico nodo (i, j) nella griglia di calcolo utilizzata per la soluzione alle differenze finite dell'equazione (2.23).

Il problema di Poisson da risolvere è $-\nabla^2\eta = k$, con k costante assegnata opportunamente. Facendo riferimento allo schema illustrato in Figura 4.1, è possibile discretizzare $\nabla^2\eta$ alle differenze finite per il generico nodo (i, j) :

$$-\frac{\eta_{(i,j+1)} - 2\eta_{(i,j)} + \eta_{(i,j-1)}}{\Delta x^2} - \frac{\eta_{(i+1,j)} - 2\eta_{(i,j)} + \eta_{(i-1,j)}}{\Delta y^2} = k \quad (4.1)$$

Essendo la griglia formata da maglie quadrate di dimensione costante, risulta che $\Delta x^2 = \Delta y^2$ e, quindi, l'equazione 4.1 si semplifica:

$$-\eta_{(i,j+1)} - \eta_{(i,j-1)} + 4\eta_{(i,j)} - \eta_{(i+1,j)} - \eta_{(i-1,j)} = \Delta x^2 \cdot k \quad (4.2)$$

Qualora il punto (i, j) considerato si trovasse sul contorno della barena e, in particolare, confinasse con un punto $(i, j + 1)$ in cui è stata imposta la condizione di flusso nullo [$ID = 1$], l'equazione (4.2) si particolarizza come segue. In termini matematici, flusso nullo significa che $\partial\eta/\partial n = 0$, ovvero che η è costante in direzione normale al bordo, perciò si ha che $\eta_{(i,j)} = \eta_{(i,j+1)}$ e l'equazione (4.2) diventa:

$$-\eta_{(i,j)} - \eta_{(i,j-1)} + 4\eta_{(i,j)} - \eta_{(i+1,j)} - \eta_{(i-1,j)} = \Delta x^2 \cdot k$$

cioè:

$$-\eta_{(i,j-1)} + 3\eta_{(i,j)} - \eta_{(i+1,j)} - \eta_{(i-1,j)} = \Delta x^2 \cdot k \quad (4.3)$$

Il coefficiente moltiplicativo dell'incognita relativa al punto (i, j) potrà essere, quindi, al più pari a 4, qualora il punto considerato fosse circondato da punti, a loro volta classificati come incognite; *viceversa* sarà pari ad 1 nel caso particolare in cui il punto (i, j) fosse circondato da tre punti in cui è stata imposta la condizione al contorno di flusso nullo³.

³Non avrebbe senso studiare l'andamento della superficie libera in un punto circondato da quattro pixel classificati come contorno impermeabile, poiché il punto in questione non verrebbe allagato dall'acqua.

Facendo variare gli indici di riga e di colonna i e j , individuando i punti nei quali si deve calcolare la soluzione e facendo attenzione alle condizioni al contorno imposte, l'equazione (4.2) comporta un sistema di equazioni lineari, di ordine pari al numero di pixel classificati come barena.

Si supponga, per esempio, di avere la seguente matrice delle condizioni al contorno:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 3 & 0 & 1 \\ 1 & 0 & 3 & 0 & 1 \\ 1 & 0 & 4 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (4.4)$$

Utilizzando la convenzione vista nel paragrafo 3.3.1, la soluzione dell'equazione (2.23) deve essere calcolata nei punti appartenenti alla barena, cioè nei punti in cui la matrice (4.4) vale 0.

La matrice delle incognite associata al problema schematizzato dalle condizioni al contorno (4.4) è, quindi:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 & 0 \\ 0 & 3 & 0 & 4 & 0 \\ 0 & 5 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (4.5)$$

e le relative incognite del problema sono:

$$\vec{\eta} = \{\eta_1, \eta_2, \eta_3, \eta_4, \eta_5, \eta_6\}^T \quad (4.6)$$

Utilizzando l'equazione (4.2) e imponendo, ove necessario, le condizioni di flusso nullo espresse dalla (4.3), si ottiene il sistema lineare che risolve il problema di Poisson per le condizioni al contorno (4.4):

$$\begin{aligned} - \text{incognita } \eta_1 : & \quad 2\eta_1 - \eta_3 = \Delta x^2 \cdot K + \text{CONDIZIONI}(2,3) \\ - \text{incognita } \eta_2 : & \quad 2\eta_2 - \eta_4 = \Delta x^2 \cdot K + \text{CONDIZIONI}(2,3) \\ - \text{incognita } \eta_3 : & \quad 3\eta_3 - \eta_1 - \eta_5 = \Delta x^2 \cdot K + \text{CONDIZIONI}(3,3) \\ - \text{incognita } \eta_4 : & \quad 3\eta_4 - \eta_2 - \eta_6 = \Delta x^2 \cdot K + \text{CONDIZIONI}(3,3) \\ - \text{incognita } \eta_5 : & \quad 2\eta_5 - \eta_3 = \Delta x^2 \cdot K + \text{CONDIZIONI}(4,3) \\ - \text{incognita } \eta_6 : & \quad 2\eta_6 - \eta_4 = \Delta x^2 \cdot K + \text{CONDIZIONI}(4,3) \end{aligned} \quad (4.7)$$

Il sistema lineare (4.7) scritto in forma matriciale diventa:

$$\underbrace{\begin{bmatrix} 2 & 0 & -1 & 0 & 0 & 0 \\ 0 & 2 & 0 & -1 & 0 & 0 \\ -1 & 0 & 3 & 0 & -1 & 0 \\ 0 & -1 & 0 & 3 & 0 & -1 \\ 0 & 0 & -1 & 0 & 2 & 0 \\ 0 & 0 & 0 & -1 & 0 & 2 \end{bmatrix}}_{\mathcal{H}} \cdot \underbrace{\begin{bmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \\ \eta_4 \\ \eta_5 \\ \eta_6 \end{bmatrix}}_{\vec{\eta}} = \underbrace{\begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ k_4 \\ k_5 \\ k_6 \end{bmatrix}}_{\vec{k}} \Rightarrow \mathcal{H} \cdot \vec{\eta} = \vec{k} \quad (4.8)$$

in cui il termine noto \vec{k} , utilizzando la notazione descritta nel capitolo 2 (cfr. Figura 2.1), risulta:

$$k(i, j) = \Delta x^2 \cdot \frac{\Lambda}{[\eta_0 - z(i, j)]^2} \frac{\partial \eta_0}{\partial t} \quad (4.9)$$

Il termine Λ , che rappresenta un coefficiente efficace d'attrito, può essere stimato con la relazione:

$$\Lambda = \frac{8}{3\pi} \frac{U}{\chi^2} \quad (4.10)$$

assumendo per U e χ i valori riportati in Tabella 2.1.

Osservando il sistema lineare (4.8) si nota che \mathcal{H} è una matrice simmetrica, sparsa, definita positiva e diagonalmente dominante. Per le caratteristiche di \mathcal{H} , il sistema lineare (4.8) ben si presta ad essere risolto utilizzando i metodi iterativi del gradiente, in particolare il metodo del *gradiente coniugato modificato* [Gambolati 2002].

Sfruttando la particolarità di \mathcal{H} , la subroutine *telone*, prima, calcola tale matrice in forma compatta, memorizzandola secondo il metodo CRS⁴, poi risolve il sistema lineare utilizzando il metodo del gradiente coniugato modificato, preconditionando il problema con il fattore incompleto di Cholesky secondo Kershaw [Gambolati 2002; Ferronato 2005].

Risolto il sistema lineare, la subroutine *telone* restituisce una matrice contenente la quota della superficie libera in ogni punto che, nella matrice delle condizioni al contorno, è classificato come barena ($ID = 0$).

In Appendice (cfr. par. A.3, dalla centotrentaduesima riga di comando), è possibile consultare il listato del codice Fortran della subroutine *Telone*, modificato per essere trasformato in una funzione di Matlab.

4.1.2 Definizione delle direzioni di drenaggio

Noto l'andamento della superficie libera in tutto il campo di moto, il modello numerico calcola le direzioni di drenaggio sulle barene e lungo la rete dei canali. Questa fase del calcolo è implementata in tre subroutine distinte:

- *incibarene*, che calcola le direzioni di drenaggio sulla superficie barenale;
- *ampiezza*, che calcola la *Funzione di Ampiezza*⁵ e determina le sezioni trasversali della rete canalizzata;
- *incicanali*, che calcola le direzioni di drenaggio lungo i canali.

⁴CRS: Compressed Row Storage. Una spiegazione del metodo in questione la si può trovare in Massimiliano Ferronato [2005]. *Progetto numerico al calcolatore*. Dispense online per il Corso di Metodi Numerici per l'Ingegneria di Giuseppe Gambolati. Università degli Studi di Padova. URL: <http://dispense.dmsa.unipd.it/gambolati/metodi/gcm/gcm.html>

⁵Si definisce *Funzione di Ampiezza* la distanza minima che intercorre tra ogni pixel appartenente alla rete dei canali e la bocca del canale a cui il pixel considerato appartiene.

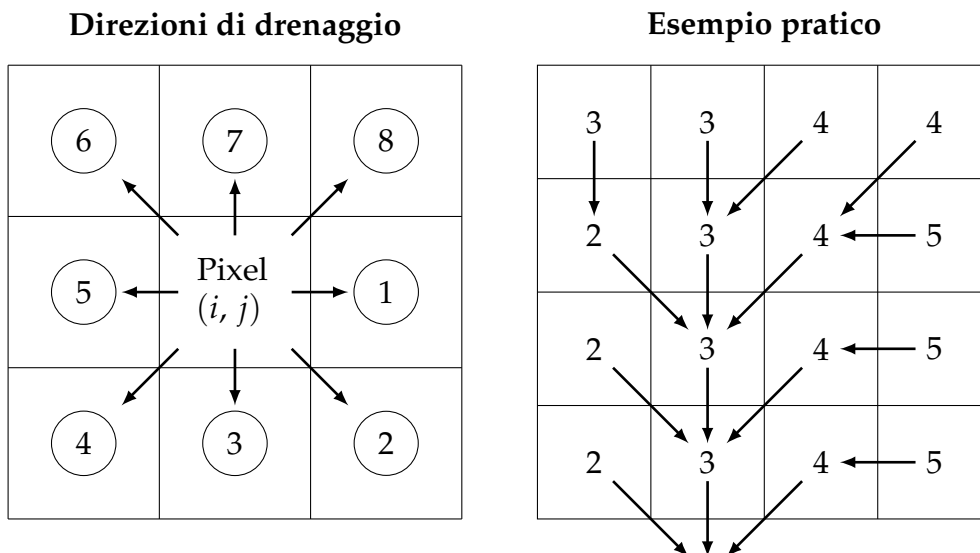


Figura 4.2: Notazione utilizzata per definire le direzioni di drenaggio nelle matrici *incidenza_bar* e *incidenza_rete*, ed esempio pratico del funzionamento della notazione.

La subroutine *incibarene*.

La subroutine *incibarene*, sulla base delle condizioni al contorno e dell'andamento della superficie liquida, che richiede come input, calcola e restituisce in output quattro diverse matrici:

- *incidenza_bar*, contenente le direzioni di drenaggio per tutti i punti della barena, ottenute seguendo la direzione del massimo gradiente della superficie libera;
- *portata_bar*, che, per ogni punto appartenente alla barena, indica il numero di pixel a monte di quello considerato e che drenano verso questo;
- *sorgenti_bar*, che, per ogni punto appartenente alla barena, vale 0 se il punto considerato è definito *sorgente* (i.e. se nessuno degli otto pixel vicini drena verso questo), oppure 1 se riceve contributi dai pixel che lo circondano;
- *lunghezza*, la matrice delle lunghezze di drenaggio al di fuori della rete dei canali.

Tra queste quattro matrici, quella che più interessa lo studio della forma delle aree di drenaggio è la prima, grazie alla quale è possibile identificare il ramo dei canali in cui fluisce il contributo di un qualsiasi pixel della barena.

Per definire le direzioni di drenaggio, come illustrato in Figura 4.2, viene assegnato a ogni pixel barena un numero intero, compreso tra 1 e 8, che individua quale tra gli otto pixel barena confinanti drena quello considerato; al resto della matrice viene assegnato valore nullo.

In Appendice (cfr. par A.5) si trova il listato della subroutine *incibarene*, che (dalla riga di comando 189) riporta commentati, i passaggi necessari a ottenere le matrici sopra descritte che, per brevità, qui non vengono riportati.

La subroutine *ampiezza*.

Data la matrice delle condizioni al contorno, la subroutine *ampiezza* calcola la distanza minima di ogni pixel dalla bocca del canale (i.e. la *Funzione di Ampiezza*) e definisce le sezioni trasversali di ogni canale.

L'algoritmo implementato (cfr. par. A.7), partendo dalla sezione iniziale di un canale, appositamente individuata con un $ID = 4$ (cfr. par. 3.3.1), si sposta verso monte di un pixel e determina, per ogni punto della nuova sezione individuata, la distanza minima dalla bocca del canale. È necessario, qui, prestare attenzione al concetto di distanza minima, poiché l'algoritmo calcola una distanza che, nel caso di canali larghi e di andamento sinuoso, non coincide con la lunghezza misurata lungo l'asse del canale. Nonostante ciò, essendo generalmente la lunghezza dei canali molto più grande della loro larghezza, la distanza che l'algoritmo calcola è una buona approssimazione della lunghezza del canale [Fagherazzi et al. 1999].

In questo modo, è possibile individuare anche le sezioni trasversali dei canali: si definisce sezione ogni insieme di pixel connessi la cui distanza x dalla bocca del canale appartiene all'intervallo $d < x \leq d + P_{sez}$, in cui P_{sez} è il passo che definisce lo "spessore" della sezione (i.e. il numero di pixel, in direzione longitudinale, che appartengono alla medesima sezione).

Come output la subroutine *ampiezza* restituisce le matrici *width* e *width_sez* contenenti, rispettivamente, in corrispondenza di ogni pixel canale, la *Funzione di Ampiezza* del pixel considerato e il numero relativo alla sezione trasversale a cui tale pixel appartiene.

La subroutine *incicanali*.

La subroutine *incicanali*, sulla base delle matrici delle condizioni al contorno e della *Funzione di Ampiezza*, che richiede come input, calcola e restituisce in output tre diverse matrici:

- *incidenza_rete*, contenente le direzioni di drenaggio per tutti i punti della rete dei canali, ottenute seguendo la direzione del massimo gradiente della *Funzione di Ampiezza*;
- *portata_rete*, che, per ogni punto appartenente alla rete canalizzata, indica il numero di pixel a monte di quello considerato e che drenano verso questo;
- *sorgenti_rete*, che, per ogni pixel canale, vale 0 se il punto considerato è definito *sorgente* (i.e. se nessuno degli otto pixel vicini drena verso questo), oppure 1 se riceve contributi dai pixel canale che lo circondano;

Nella matrice *incidenza_rete* le direzioni di drenaggio sono definite nello stesso modo utilizzato per definirle nella matrice *incidenta_bar*: ad ogni pixel canale viene assegnato un numero intero, compreso tra 1 e 8 (cfr. Figura 4.2), che individua quale, tra gli otto pixel canale confinanti, drena quello considerato; al resto della matrice viene assegnato valore nullo.

In Appendice viene riportato il listato della subroutine *incicanali* (cfr. par. A.9) che illustra, ampiamente commentati, i passaggi necessari a ottenere le tre matrici di cui sopra che, per brevità, qui non vengono riportati.

4.1.3 La subroutine *segnarami*

Utilizzando le subroutine esposte nel paragrafo 4.1.2 si determinano le direzioni di drenaggio nel dominio schematizzato dalla matrice delle condizioni al contorno, su tutta l'area barenale e lungo tutta la rete dei canali.

Disponendo, quindi, delle matrici *incidenza_bar* e *incidenza_rete*, per come queste sono state costruite, la determinazione della matrice *incidenza*, contenente le direzioni di drenaggio di tutta l'area di studio, è immediata e consiste nella somma delle due matrici parziali. Conoscendo le direzioni di drenaggio su tutta l'area che si vuole analizzare, è possibile determinare la forma delle aree di drenaggio dei canali incidenti la barena ricorrendo alla subroutine *segnarami*. Questa richiede come input:

- la matrice *contorno*, contenente le condizioni al contorno;
- la matrice *incidenza*, contenente le direzioni di drenaggio;
- la matrice *incidenza_rete*, contenente le direzioni di drenaggio lungo la rete canalizzata;
- la matrice *sezioni*, uguale alla matrice *contorno*, nella quale vengono individuati, con un numero identificativo (*ID*) maggiore di 4, i pixel appartenenti alle sezioni di chiusura dei rami di cui si vuole determinare l'area di drenaggio.

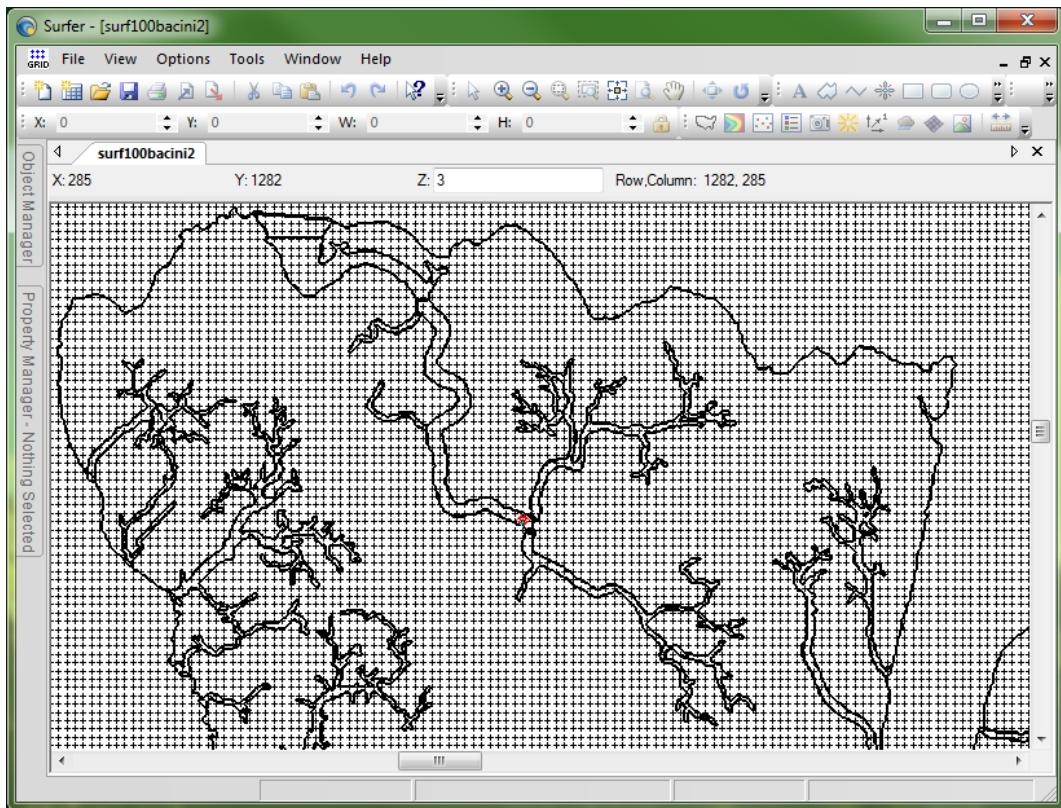
Sulla base di questi quattro input, la subroutine *segnarami*, per tutte le sezioni di chiusura presenti nella matrice *sezioni*, assegna, nella matrice *rami*, lo stesso numero identificativo a tutti i pixel dei canali che drenano verso un punto appartenente a una sezione di chiusura, individuata da un certo *ID*.

Definiti i rami dei canali di cui si vuole determinare l'area di drenaggio, conoscendo le direzione di drenaggio in ogni punto della barena, la subroutine identifica i pixel, il cui contributo fluisce nei canali riportati nella matrice *rami*, e assegna, nella matrice *bacini*, a tutti questi pixel, il medesimo *ID* del canale che li drena.

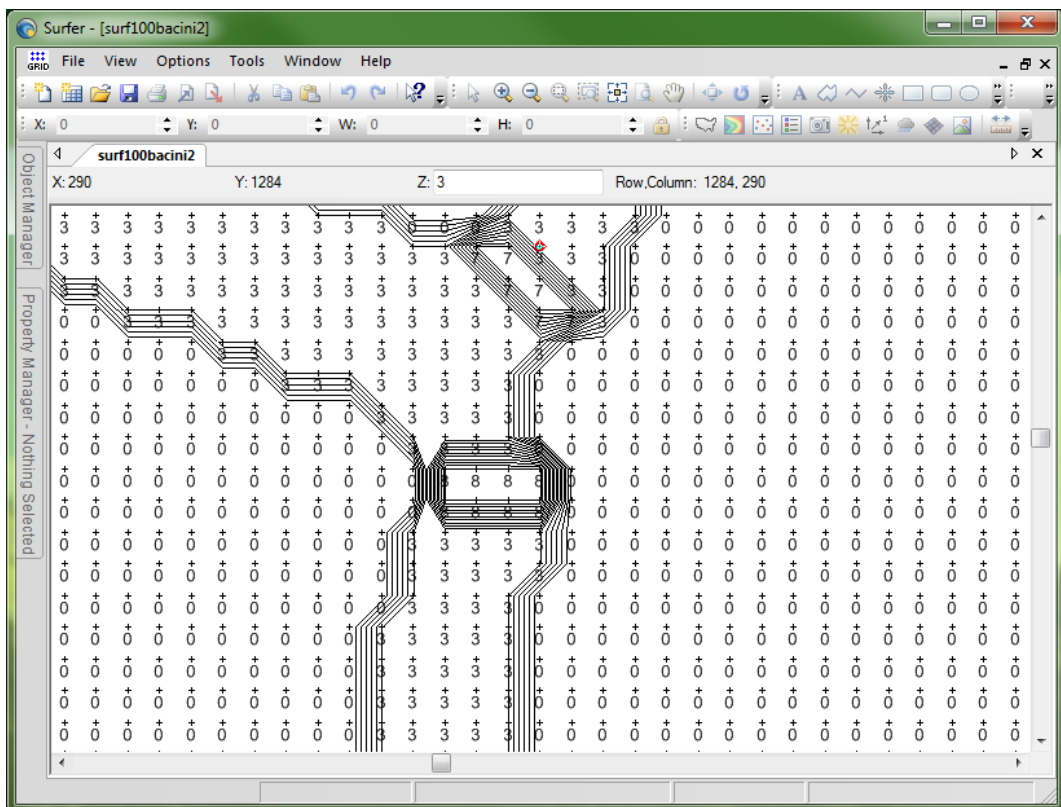
In questo modo, nella matrice *bacini*, vengono identificate, univocamente, le aree drenate da ogni canale chiuso a una determinata sezione, impostata nella matrice *sezioni*. La matrice *bacini* sarà, pertanto, il punto di partenza per lo studio della forma geometrica delle aree di drenaggio dei canali della Palude Pagliaga.

La matrice *sezioni*

Tra le matrici che la subroutine *segnarami* richiede come input, è necessario prestare un po' di attenzione alla matrice *sezioni*: se, concettualmente, il fatto di assegnare un $ID > 4$ alle sezioni di chiusura dei tratti di canale che si vuole studiare è un'operazione semplice, dal punto di vista pratico, non è detto che lo sia. Infatti, spesso, la matrice delle condizioni al contorno è molto grande (dell'ordine di 1000×1000 elementi), perciò individuare manualmente la posizione dei canali



(a) Particolare di un canale di cui si vogliono determinare le aree di drenaggio.



(b) Particolare delle sezioni di chiusura rispetto alle quali si determinano le aree di drenaggio.

Figura 4.3: Particolare dalla matrice sezioni visualizzato con il software Surfer11.

e delle sezioni che devono essere marcate dall'*ID* desiderato non è per niente agevole.

Per completare quest'operazione è consigliabile utilizzare dei particolari editor di testo in grado di dare una rappresentazione grafica del file che si vuole editare. Per questo lavoro è stato utilizzato il software *Surfer11*, prodotto dalla società *Golden Software*⁶.

In Figura 4.3 si riportano due particolari della matrice *sezioni* della Palude Pagliaga, in fase di assegnazione dell'*ID* alle sezioni di chiusura, rispetto alle quali individuare le aree di drenaggio.

4.2 L'implementazione in Matlab

Nella prima parte di questo capitolo è stato trattato il tema dell'implementazione di un modello numerico in grado di risolvere il problema associato all'equazione (2.23) e di utilizzare tale soluzione per determinare la forma dei bacini di drenaggio dei canali lagunari.

Poiché i codici che compongono il modello esaminato nella sezione 4.1 erano già esistenti e scritti in Fortran, volendo implementare lo stesso modello in Matlab, invece di riscrivere da zero l'intero modello, è stato pensato di trasformare in funzioni Matlab i programmi Fortran già esistenti, trasformandoli in dei *MEX-File*⁷.

In questa seconda parte del capitolo, dopo aver elencato le ragioni per cui è conveniente il passaggio dal linguaggio Fortran a quello di Matlab e dopo aver definito cosa sono i *MEX-File*, si espone il lavoro eseguito per trasformare le subroutine della sezione 4.1 in una toolbox di funzioni di Matlab.

4.2.1 I vantaggi della programmazione in Matlab

Negli ultimi vent'anni si è assistito ad una vera e propria rivoluzione nel campo della tecnologia informatica. Il merito principale di questo progresso tecnologico è stato, in un primo momento, quello di fornire calcolatori la cui potenza ha avuto un'incremento di tipo esponenziale e, più di recente, quello di contribuire alla diffusione sempre più capillare di software di supporto a tecnici e ricercatori.

La diffusione a grandissima scala dei nuovi software ha dato origine a una nuova tendenza legata alla semplificazione dei programmi: se, prima, questi strumenti erano prerogativa di poche persone, ora l'obiettivo è quello di creare applicazioni la cui semplicità e immediatezza giocano un ruolo fondamentale.

Questa tendenza, sicuramente positiva in senso generale, si sta facendo sentire anche negli ambienti della ricerca scientifica, infatti, anche in questo campo, vengono apprezzati sempre di più gli strumenti in grado non solo di trovare la soluzione di determinati problemi, ma anche di farlo in modo semplice, dandone poi una rappresentazione grafica chiara ed efficiente. Una conferma di questa affermazione la si può trovare nel fatto che, chi è abituato a lavorare con linguaggi

⁶<http://www.goldensoftware.com/>

⁷MEX-File: *Matlab EXecutable File*

di programmazione simili al Fortran, per l'analisi dei risultati deve ricorrere a programmi specifici che garantiscono una resa grafica di buon livello.

È in questo contesto che si inserisce il lavoro di aggiornamento e semplificazione del *Modello del Telone* ed è qui che entra in gioco Matlab, sia inteso come linguaggio di programmazione più semplice e immediato del Fortran (nonché molto più utilizzato, per non dire "di moda"), ma anche inteso come ambiente di lavoro ricco di interazioni con altri programmi e dotato di strumenti grafici molto potenti.

Volendo riassumere i vantaggi della programmazione in Matlab, piuttosto che in Fortran, si può dire che:

- il primo vantaggio del Matlab rispetto al Fortran è che, essendo un linguaggio *precompilato*, dà la possibilità non solo di scrivere dei programmi di calcolo ed eseguirli tutti in una volta, ma anche di testare singoli passaggi direttamente dalla linea di comando, rendendo estremamente più semplice l'approccio con le varie parti del codice e la successiva fase di verifica e *debug* dello stesso.
- Il secondo vantaggio riguarda la semplificazione delle fasi di creazione delle variabili e di lettura dei dati di input. Se in Fortran, prima di poter iniziare l'elaborazione vera e propria, è necessario:
 1. aprire il programma specificandone il nome;
 2. dichiarare i nomi e la tipologia di tutte le variabili che ci si appresta ad utilizzare;
 3. aprire singolarmente i file di input (i quali devono rispettare criteri abbastanza rigidi per essere fruibili);
 4. leggere singolarmente tutti gli elementi contenuti nei file di input assegnandoli alle rispettive variabili;

in Matlab la creazione delle variabili avviene, quasi in tutte le situazioni, simultaneamente alla lettura stessa dei dati, la quale, a sua volta, è estremamente semplificata, eseguibile in poche righe di programmazione e interfacciabile a diversi tipi di file, quali possono essere i classici file di testo, immagini salvate in diversi formati, ma anche singole parti di un foglio di calcolo elettronico⁸.

- Il terzo vantaggio che si può individuare riguarda la vastità di librerie di funzioni predisposte per essere utilizzate in Matlab. Per quanto anche in Fortran, dopo l'aggiornamento alla versione *Fortran95*, si possano caricare dei pacchetti di funzioni, il loro utilizzo non risulta così agevole come invece è in Matlab, che, per di più, affianca al grande numero di funzioni computazionali un altrettanto grande numero di funzioni grafiche, con

⁸E.g.: con Matlab è possibile importare in una variabile un intervallo di righe e colonne di un foglio qualsiasi, appartenente ad una cartella *Excel*, direttamente e senza la necessità di salvare il file in altri formati.

le quali si possono realizzare grafici (anche tridimensionali) e semplici animazioni salvabili in diversi formati.

- L'ultimo, ma non meno importante, vantaggio riguarda la presenza in Matlab dello strumento *help*: digitando questa istruzione seguita dal nome di una qualsiasi funzione, direttamente dalla linea di comando, si ha accesso immediato alla spiegazione del funzionamento di tale funzione e, se disponibili, ad alcuni esempi di utilizzo. Questo elemento, di fatto, può rimpiazzare totalmente l'uso di scomodi manuali, rendendo il compito molto più semplice a chi sta programmando, sia che si tratti di un utente esperto di programmazione in Matlab, sia che si tratti di qualcuno che si sta avvicinando a questo ambiente per la prima volta.

Se esistono numerosi vantaggi nell'utilizzo del Matlab al posto del Fortran, è bene anche evidenziare che, nella migrazione da un linguaggio all'altro, non esistono solamente aspetti positivi: infatti, tutti i vantaggi esposti in precedenza hanno un costo che si traduce in una velocità di calcolo nettamente inferiore, qualora nel programma Matlab dovessero essere presenti lunghi cicli iterativi, i quali, invece, in linguaggi di programmazione come il Fortran vengono elaborati molto più rapidamente.

Per i motivi fin qui analizzati, si può quindi concludere che l'importazione in Matlab del modello descritto nella sezione 4.1 può sicuramente costituire un vantaggio. Infatti, semplificandone l'utilizzo, si renderà il modello fruibile da un maggior numero di persone e, contemporaneamente, chi dovrà farne uso potrà impiegare molto meno tempo a capire come farlo funzionare, potendosi così concentrare sugli aspetti che contano di più, ovvero sui risultati che questo fornisce.

A quest'affermazione, però, è necessario aggiungere che, se i vantaggi possono essere molti, la trasformazione deve essere tale da non provocare un eccessivo rallentamento nel calcolo dovuto alle caratteristiche del nuovo software: per questo motivo la realizzazione di nuovi programmi non può essere effettuata con una traduzione meccanica dei codici, bensì deve essere realizzata "intelligentemente", traducendo quello che ben si adatta all'implementazione in Matlab e cercando di evitare quanto più possibile i cicli iterativi.

Per risolvere il problema del rallentamento dell'esecuzione del modello, nei paragrafi 4.2.2 e 4.2.3 si espone un metodo generale che permette di godere dei numerosi vantaggi del Matlab, ma senza dover rinunciare alla maggior efficienza dei programmi in Fortran.

Tale metodo, inoltre, permette di sfruttare parti di codice già esistenti, organizzate in *subroutine*, trasformandole in funzioni di Matlab senza doverle trascrivere, risultando particolarmente idoneo al caso in esame, essendo il modello che si vuole utilizzare già completamente esistente ed organizzato in *subroutine*.

4.2.2 I MEX-File: cosa sono e perché utilizzarli

Il motivo della nascita dei MEX-file è da ricercare nelle origini di Matlab: quando fu fondata la società proprietaria del software, nel 1984, il prodotto che stava per essere commercializzato era molto diverso da quello che oggi è utilizzato in tutto il mondo, mentre i linguaggi di programmazione più diffusi e utilizzati negli ambienti scientifici e universitari erano il C e il Fortran.

Con molte meno librerie di funzioni a disposizione, i primi utilizzatori, per poter adoperare questo nuovo strumento, avrebbero dovuto riscrivere in linguaggio Matlab tutti i programmi che avevano già sviluppato: per questo motivo, i fondatori di *The MathWorks* pensarono ad una procedura per rendere utilizzabili parti dei vecchi codici in C e Fortran, senza la necessità della trascrizione manuale di quanto essi contenevano.

Nonostante l'ecosistema di Matlab si sia arricchito, nel corso degli anni, di moltissime funzioni, gli sviluppatori del software hanno ritenuto opportuno non togliere agli utenti la possibilità di creare da soli le proprie librerie di funzioni richiamando i propri codici scritti in C e Fortran.

Oggi, i motivi per cui questa procedura viene ancora considerata utile sono diversi:

- il primo motivo è che per alcuni campi molto specifici, è possibile che non esista ancora in Matlab una libreria di funzioni dedicata;
- una seconda ragione è che molte delle persone abituate a programmare in C e Fortran possono preferire importare i programmi da loro stessi sviluppati, piuttosto che utilizzare delle funzioni *stock* che danno dei risultati, ma che non permettono di controllare i passaggi con cui quest'ultimi vengono determinati;
- un ultimo, ma non meno importante, motivo dell'attuale utilizzo dei MEX-file è da ricercare nella velocità di calcolo. Infatti, per come fu concepito il Matlab, in presenza di numerosi e lunghi cicli iterativi, la velocità di avanzamento di un programma può subire alcuni rallentamenti di non poco conto, se si confronta il tempo complessivo di calcolo con quello ottenibile con i vecchi linguaggi. Utilizzando i MEX-file si può aggirare questo limite⁹.

Dal punto di vista pratico, i MEX-file sono delle *subroutine* prodotte con codici C o Fortran che vengono identificate da Matlab, grazie all'estensione legata alla piattaforma su cui si sta lavorando, come se fossero delle *function* salvate in file con l'estensione *.m*, propria dei programmi Matlab [The MathWorks 2014b].

Nel caso del modello numerico in esame ci sono tutti i presupposti per utilizzare dei MEX-file anziché tradurre da zero i codici originali: infatti, il modello in Fortran è già suddiviso in *subroutine* che ben si prestano a diventare MEX-file; inoltre, dovendo scorrere molte volte i valori contenuti in matrici di grandi dimensioni, le subroutine che compongono il modello presentano numerosi cicli iterativi di cui raramente si può fare a meno.

⁹A dimostrazione di ciò si evidenzia che non è un caso se moltissime toolbox presenti in Matlab, di recente realizzazione, siano basate sui linguaggi C e C++ [Kopecky 2005].

Per questi due motivi è stato scelto di realizzare una toolbox di funzioni richiamabili in Matlab, trasformando le subroutine Fortran in MEX-file e creando delle funzioni Matlab per semplificarne ulteriormente l'uso.

4.2.3 Come ottenere un MEX-File

Si descrive ora come ottenere un MEX-File per l'esecuzione in Matlab di una subroutine sviluppata in Fortran. In particolare, si forniscono le informazioni di base riguardo all'utilizzo delle API¹⁰ che rendono Matlab interfacciabile al Fortran; si descrive la struttura di una *gateway routine* per il passaggio dei dati tra i due linguaggi; si riassumono i passaggi necessari alla compilazione dei file e alla generazione vera e propria dei MEX-File.

L'intento è quello di fornire, a chi legge, le informazioni necessarie a comprendere il funzionamento dei programmi sviluppati nel lavoro esposto in questa tesi. A chi fosse interessato, invece, alla realizzazione di programmi analoghi, per ulteriori approfondimenti, si consiglia di consultare la ricca documentazione riguardante gli argomenti qui accennati, disponibile in rete e nello strumento di supporto di Matlab¹¹.

Le API Routines in Matlab

Tra le varie funzionalità implementate dagli sviluppatori di *The MathWorks*, i MEX-File ricadono tra quelle che usufruiscono delle *API routines*, le quali permettono d'interfacciare i programmi Matlab ad altre applicazioni presenti nei vari sistemi operativi.

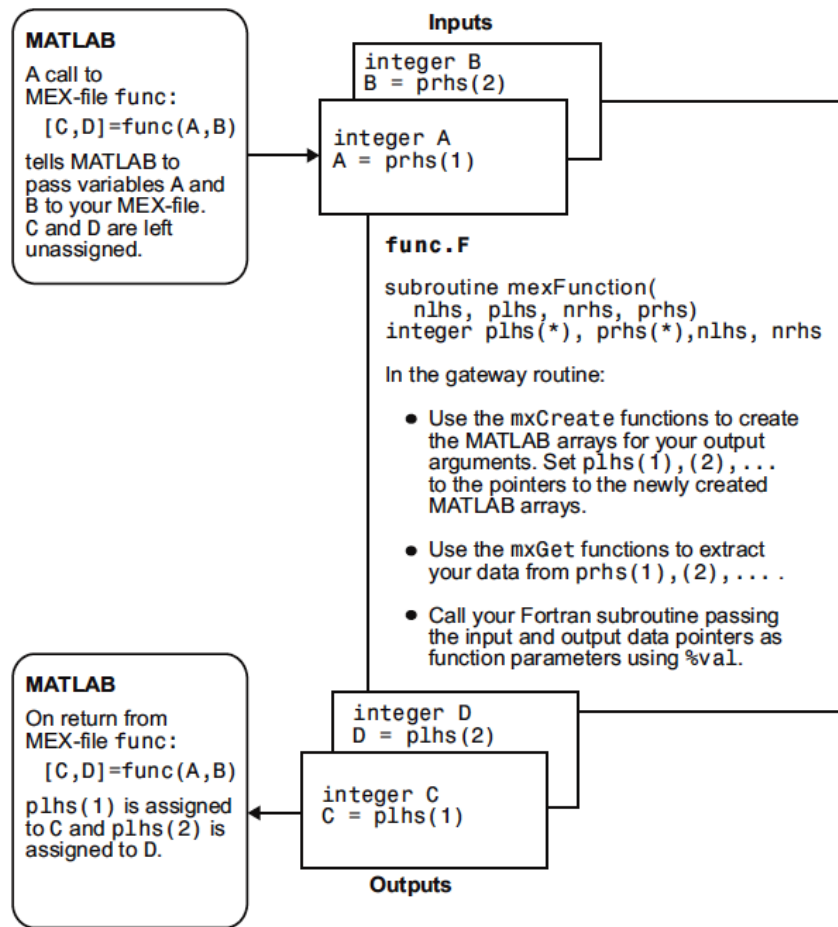
Per avere accesso a queste particolari routine, se si lavora in ambiente *Windows*, può essere richiesta, nel computer che si sta utilizzando, l'installazione di programmi di terze parti. In particolare, nel caso della creazione dei MEX-File, Matlab richiede che, nel computer, sia presente almeno uno tra i compilatori compatibili e, se necessario, uno tra i programmi, creati da *Microsoft* in base al sistema operativo, da cui dipendono le funzionalità del compilatore che si intende utilizzare.

Una classificazione dei programmi mutuamente compatibili, organizzata in base al sistema operativo, per poter richiamare in Matlab i compilatori di altri linguaggi di programmazione, la si può trovare in [The MathWorks 2014c] e nei suoi futuri aggiornamenti.

Nello sviluppo dei programmi, per l'implementazione in Matlab del modello descritto nella sezione 4.1, la realizzazione dei MEX-File è stata effettuata, inizialmente, in un sistema operativo Windows a 32-bit, con la combinazione dei programmi composta da *Matlab R2012a*, *Microsoft Visual Studio 2010* e *Intel Visual Studio EX 2011*; successivamente, i MEX-File così creati sono stati adattati e ricompilati, con la medesima combinazione di software, ma per sistemi operativi Windows a 64-bit.

¹⁰Per API (*Application Programming Interface*) si intende l'insieme di funzioni che specificano come le varie componenti di diversi software devono interagire tra loro.

¹¹Tra la vasta documentazione disponibile sulla creazione dei MEX-File si consiglia di consultare [The MathWorks 2014a,b,c,d]



Fortran MEX Cycle

Figura 4.4: Diagramma di flusso dei dati all'interno di un generico Fortran MEX-File, dovuto alla chiamata da linea di comando di Matlab [The MathWorks 2014b].

La gateway routine

Installati tutti i programmi necessari ad aver accesso alle *API routines*, si può iniziare la creazione della *gateway routine*. Questa, nel caso della compilazione di un Fortran MEX-File, è una particolare subroutine di passaggio dei dati, dalla linea di comando di Matlab, alla subroutine scritta in Fortran.

In Figura 4.4 è illustrato il diagramma di flusso dei dati all'interno di un generico Fortran MEX-File: nella *gateway routine*, che in tutti i file di questo tipo deve essere dichiarata come **subroutine mexFunction**, non avviene nessun tipo di elaborazione dei dati, ma si effettuano solo le operazioni necessarie a rendere compatibili le variabili, di input e output di Matlab, con quelle che si elaborano nella subroutine Fortran, che si desidera trasformare in MEX-File (*computational subroutine*).

Per la generazione dei MEX-File si è tentato di procedere secondo uno schema prefissato: si esamini, ad esempio, il codice della *gateway routine* per la subroutine *telone*, riportata qui di seguito.

```

1: !      The gateway routine: ovvero la subroutine che lega gli oggetti tipici
2: !      di matlab con le classiche variabili integer e real
3:      subroutine mexFunction(nlhs, plhs, nrhs, prhs)
4:      implicit none
5: !-----
6:      integer plhs(*), prhs(*)
7:      integer mxCreateDoubleMatrix
8:      integer contorno_pr,dim1_pr,imax_pr,maglia_pr
9:      integer U_max_pr,dETA0_dt_pr,CHI_pr,ETA0_pr
10:     integer soluzione_pr,iter_pr,toll_pr,z_pr
11:     integer nlhs, nrhs
12:     integer n1, n2, dim1, dim2, size_cont, imax
13:     integer mxGetM, mxGetN, mxGetPr, mxIsNumeric
14:     REAL*8,allocatable :: soluzione(:, :)
15:     REAL*8,allocatable :: contorno(:, :)
16:     REAL*8,allocatable :: z(:, :)
17:     real*8 dim1_t,imax_t,iter,maglia,U_max,dETA0_dt
18:     REAL*8 CHI,ETA0,toll
19: !-----
20: ! controllo che gli input siano in numero appropriato.
21:     if (nrhs .ne. 10) then
22:         call mexErrMsgTxt('10 inputs required.')
23:     endif
24: ! controllo che gli output siano in numero appropriato.
25:     if (nlhs .ne. 2) then
26:         call mexErrMsgTxt('2 output required.')
27:     endif
28: ! controllo che gli input siano matrici di numeri.
29:     if (mxIsNumeric(prhs(1)) .ne. 1) then
30:         call mexErrMsgTxt('Input #1 is not a numeric array.')
31:     endif
32:     if (mxIsNumeric(prhs(2)) .ne. 1) then
33:         call mexErrMsgTxt('Input #2 is not a numeric.')
34:     endif
35:     if (mxIsNumeric(prhs(3)) .ne. 1) then
36:         call mexErrMsgTxt('Input #3 is not a numeric.')
37:     endif
38:     if (mxIsNumeric(prhs(4)) .ne. 1) then
39:         call mexErrMsgTxt('Input #4 is not a numeric.')
40:     endif
41:     if (mxIsNumeric(prhs(5)) .ne. 1) then
42:         call mexErrMsgTxt('Input #5 is not a numeric.')
43:     endif
44:     if (mxIsNumeric(prhs(6)) .ne. 1) then
45:         call mexErrMsgTxt('Input #6 is not a numeric.')
46:     endif
47:     if (mxIsNumeric(prhs(7)) .ne. 1) then
48:         call mexErrMsgTxt('Input #7 is not a numeric.')
49:     endif
50:     if (mxIsNumeric(prhs(8)) .ne. 1) then
51:         call mexErrMsgTxt('Input #8 is not a numeric.')
52:     endif
53:     if (mxIsNumeric(prhs(9)) .ne. 1) then
54:         call mexErrMsgTxt('Input #9 is not a numeric.')
55:     endif
56:     if (mxIsNumeric(prhs(10)) .ne. 1) then
57:         call mexErrMsgTxt('Input #10 is not a numeric array.')
58:     endif
59:

```

```

60: ! rilevo le dimensioni degli input assegnati alla funzione
61:     n1 = mxGetM(prhs(1))
62:     n2 = mxGetN(prhs(1))
63:     allocate(contorno(n1,n2))
64:     allocate(soluzione(n1,n2))
65:     allocate(z(n1,n2))
66:     size_cont = n1*n2
67:
68: ! creo una matrice per i risultati di output.
69:     plhs(1) = mxCreateDoubleMatrix(n1, n2, 0)
70:     plhs(2) = mxCreateDoubleMatrix(1, 1, 0)
71:
72: ! assegno i parametri di input e output a delle variabili.
73:     contorno_pr = mxGetPr(prhs(1))
74:     dim1_pr = mxGetPr(prhs(2))
75:     imax_pr = mxGetPr(prhs(3))
76:     maglia_pr = mxGetPr(prhs(4))
77:     U_max_pr = mxGetPr(prhs(5))
78:     dETA0_dt_pr = mxGetPr(prhs(6))
79:     CHI_pr = mxGetPr(prhs(7))
80:     ETA0_pr = mxGetPr(prhs(8))
81:     toll_pr = mxGetPr(prhs(9))
82:     z_pr = mxGetPr(prhs(10))
83:
84:     soluzione_pr = mxGetPr(plhs(1))
85:     iter_pr = mxGetPr(plhs(2))
86:
87: ! carico i dati di input in variabili fortran.
88:     call mxCopyPtrToReal8(contorno_pr, contorno, size_cont)
89:     call mxCopyPtrToReal8(dim1_pr, dim1_t, 1)
90:     call mxCopyPtrToReal8(imax_pr, imax_t, 1)
91:     call mxCopyPtrToReal8(maglia_pr, maglia, 1)
92:     call mxCopyPtrToReal8(U_max_pr, U_max, 1)
93:     call mxCopyPtrToReal8(dETA0_dt_pr, dETA0_dt, 1)
94:     call mxCopyPtrToReal8(CHI_pr, CHI, 1)
95:     call mxCopyPtrToReal8(ETA0_pr, ETA0, 1)
96:     call mxCopyPtrToReal8(toll_pr, toll, 1)
97:     call mxCopyPtrToReal8(z_pr, z, size_cont)
98:
99:     dim1 = INT(dim1_t)
100:    imax = INT(imax_t)
101:    dim2 = dim1*3
102:
103: ! richiamo la computational subroutine.
104:    call telone(contorno,n1,n2,dim1,dim2,imax,maglia,U_max,
105:    $ dETA0_dt,CHI,ETA0,soluzione,iter,toll,z)
106:
107: ! carico i dati di output nella MATLAB array.
108:    call mxCopyReal8ToPtr(soluzione, soluzione_pr, size_cont)
109:    call mxCopyReal8ToPtr(iter, iter_pr, 1)
110:
111:    deallocate(contorno)
112:    deallocate(soluzione)
113:    deallocate(z)
114:
115:    return
116:    contains
117:
118: !=====

```

Per tutte le subroutine descritte nella sezione 4.1, la *gateway routine* è stata realizzata secondo il seguente schema.

1. Si crea la subroutine **mexFunction** (*nlhs*, *plhs*, *nrhs*, *prhs*) e si dichiara l'opzione **implicit none**.
2. Si dichiarano come **integer** le variabili utilizzate come *puntatori*¹² per il passaggio delle variabili Matlab.
3. Si dichiarano le variabili Fortran, specificando, per le matrici, l'opzione **allocatable** così da assegnare manualmente lo spazio necessario ad accogliere i dati di input e output della subroutine Fortran.
4. Si verifica che i dati in input siano, in numero e tipo, adeguati al funzionamento della subroutine.
5. Si leggono, con le funzioni **mxGetM** e **mxGetN**, rispettivamente, il numero di righe e di colonne delle matrici in input e si alloca lo spazio per le matrici da passare alla subroutine.
6. Si creano le variabili in output da restituire a Matlab.
7. Si assegnano ai *puntatori*, dichiarati al punto 2, le rispettive variabili Matlab di input e output.
8. Si trasferiscono i dati contenuti nelle variabili di input, provenienti da Matlab, a quelle di tipo Fortran da passare alla subroutine, con la funzione **mxCopyPtrToReal8**.
9. Si richiama la subroutine scritta in Fortran.
10. Si trasferiscono i dati di output da restituire al software Matlab, contenuti nelle variabili Fortran, nelle rispettive variabili Matlab utilizzando la funzione **mxCopyReal8ToPtr**.
11. Si *dealloca* lo spazio assegnato alle matrici Fortran.
12. Si ritornano al software Matlab le variabili in output con i risultati dell'esecuzione della subroutine Fortran e si chiude la subroutine della *gateway routine*.

In questo modo sono state create tutte le funzioni d'interfaccia che permettono di eseguire le subroutine Fortran direttamente dalla linea di comando di Matlab.

In Appendice (cfr. par. A.3, A.5, A.7, A.9, A.11) si riportano per intero i listati dei codici di tutti i MEX-File implementati per la determinazione della forma delle aree di drenaggio dei canali a marea.

¹²Nel passaggio dalla versione a 32-bit a quella a 64-bit cambia la dichiarazione di queste variabili da **integer** a degli appositi formati implementati per sfruttare le caratteristiche delle piattaforme a 64-bit. [The MathWorks 2014d]

Compilazione dei MEX-File

Al fine di ottenere dai file descritti in questo capitolo i MEX-File veri e propri, è necessario compilare in Matlab i file contenenti le subroutine e le relative *gateway routine*.

Per completare quest'operazione dalla linea di comando di Matlab si devono compiere due passaggi:

1. impostare come predefinito il compilatore scelto per processare le subroutine create;
2. compilare i codici elaborati (uno alla volta o tutti assieme), verificando che il compilatore non segnali la presenza di errori.

La prima di queste due operazioni, una volta che si ha installato correttamente il compilatore ed eventuali altri programmi richiesti dal sistema operativo che si sta utilizzando [The MathWorks 2014c], si compie, direttamente dalla linea di comando della *Comand Window* di Matlab, digitando l'espressione:

```
mex -setup
```

Come illustrato in Figura 4.5, a tale comando il software avvia la procedura per il settaggio del compilatore chiedendo in serie:

- se si desidera che la funzione *mex* individui i compilatori compatibili installati;
- se è presente più di un compilatore, quale tra quelli rilevati deve essere impostato;
- se il compilatore è effettivamente disponibile (i.e., se nel computer sono presenti tutti i programmi necessari), viene chiesto di confermare la scelta effettuata e si chiude la procedura.

Se l'operazione va a buon fine, Matlab dispone di tutto il necessario per processare le subroutine scritte in Fortran.

La seconda delle due operazioni, la compilazione, viene eseguita anch'essa direttamente dalla *Comand Window* di Matlab: per compilare un file contenente una *gateway routine* e una o più *computational subroutine* è sufficiente eseguire il comando:

```
mex -v -largeArrayDims nomefile.f90
```

in cui l'opzione `-v` serve per visualizzare a video i risultati dei vari passaggi della compilazione ed eventuali errori di sintassi effettuati, mentre l'opzione `-largeArrayDims` attiva le API dedicate alle matrici di grandi dimensioni.

In Figura 4.6 è illustrato un esempio di quanto viene visualizzato nella *Comand Window* in seguito alla compilazione del file `F_telone.f90` (cfr. par A.3).

Attuando la procedura qui proposta, nella cartella in cui si trovano i file compilati, vengono creati dei file con il medesimo nome dei quelli processati, ma con l'estensione `MEXW32` (o `MEXW64` in base al sistema operativo). Tramite questi particolari file le subroutine Fortran divengono delle function di Matlab.

```
>> mex -setup

Welcome to mex -setup. This utility will help you set up
a default compiler. For a list of supported compilers, see
http://www.mathworks.com/support/compilers/R2012a/win32.html

Please choose your compiler for building MEX-files:

Would you like mex to locate installed compilers [y]/n? y

Select a compiler:
[1] Intel Visual Fortran 12.0 (with Microsoft Visual C++ 2010 linker) in C:\Prog
[2] Lcc-win32 C 2.4.1 in C:\PROGRA~1\MATLAB\R2012a\sys\lcc
[3] Microsoft Visual C++ 2010 in C:\Program Files\Microsoft Visual Studio 10.0

[0] None

Compiler: 1

Please verify your choices:

Compiler: Intel Visual Fortran 12.0
Location: C:\Program Files\Intel\ComposerXE-2011\

Are these correct [y]/n? y

Trying to update options file: C:\Users\Francesco\AppData\Roaming\MathWorks\MATL
From template: C:\PROGRA~1\MATLAB\R2012a\bin\win32\mexopts\intelf12

Done . . .

*****
Warning: The MATLAB C and Fortran API has changed to support MATLAB
variables with more than 2^32-1 elements. In the near future
you will be required to update your code to utilize the new
API. You can find more information about this at:
http://www.mathworks.com/help/techdoc/matlab\_external/bsflnue-1.html
Building with the -largeArrayDims option enables the new API.
*****

fx >> |
```

Figura 4.5: Command Window di Matlab: schermata che si ottiene nel settaggio del compilatore con cui processare i MEX-File.

```

Command Window
File Edit Debug Desktop Window Help
>> mex -v -largeArrayDims F_telone.f90
-> Default options filename found in C:\Users\Francesco\AppData\Roaming\MathWorks\
-----
-> Options file           = C:\Users\Francesco\AppData\Roaming\MathWorks\MATLAB
MATLAB                   = C:\Program Files\MATLAB\R2012a
-> COMPILER               = ifort
-> Compiler flags:
  COMPFLAGS               = /fpp /Qprec "/IC:\Program Files\MATLAB\R2012a/exter
  OPTIMFLAGS              = /O2 /DNDEBUG
  DEBUGFLAGS              = /Z7
  arguments                =
  Name switch              = /Fo
-> Pre-linking commands  =
-> LINKER                 = link
-> Link directives:
  LINKFLAGS                = /dll /export:MEXFUNCTION /LIBPATH:"C:\Program Files
  LINKDEBUGFLAGS           = /debug /PDB:"F_telone.mexw32.pdb"
  LINKFLAGSPOST            =
  Name directive            = /out:"F_telone.mexw32"
  File link directive      =
  Lib. link directive      =
  Rsp file indicator       = @
-> Resource Compiler      = rc /fo "mexversion.res"
-> Resource Linker        =
-----

--> ifort /fpp /Qprec "/IC:\Program Files\MATLAB\R2012a/extern/include" -c -nolog

  Contents of C:\Users\FRANCE~1\AppData\Local\Temp\mex_k4oYrX\mex_tmp.rsp:
  C:\Users\FRANCE~1\AppData\Local\Temp\mex_k4oYrX\F_telone.obj

--> link /out:"F_telone.mexw32" /dll /export:MEXFUNCTION /LIBPATH:"C:\Program File

  Creazione della libreria C:\Users\FRANCE~1\AppData\Local\Temp\mex_k4oYrX\templi

--> del "C:\Users\FRANCE~1\AppData\Local\Temp\mex_k4oYrX\templib.x" "C:\Users\FRAN

--> mt -outputresource:"F_telone.mexw32";2 -manifest "F_telone.mexw32.manifest"

Microsoft (R) Manifest Tool version 5.2.3790.2076
Copyright (c) Microsoft Corporation 2005.
All rights reserved.

--> del "F_telone.mexw32.manifest"

--> del "F_telone.mexw32.map"
fx
OVR

```

Figura 4.6: Command Window di Matlab: schermata che si ottiene alla compilazione di un MEX-File.

4.2.4 Realizzazione delle funzioni di supporto ai MEX-File

Nell'ottica di creare una toolbox di funzioni per la determinazione in Matlab della forma delle aree di drenaggio dei canali a marea, disponendo di programmi già sviluppati in Fortran (cfr. sez. 4.1), nei paragrafi 4.2.2 e 4.2.3, è stato esposto un modo per rendere eseguibili in Matlab tali programmi senza doverli tradurre.

I MEX-File così ottenuti, nonostante siano perfettamente funzionanti dal punto di vista pratico, se confrontati con le altre funzioni disponibili in Matlab, presentano ancora qualche limite: nel chiamare una funzione contenuta in un MEX-File è strettamente necessario assegnare rigorosamente tutti gli input nel corretto ordine e predisporre tutti gli output previsti, altrimenti la subroutine non viene eseguita e il software Matlab diventa molto instabile. Inoltre, alle subroutine così implementate non è possibile associare la spiegazione caratteristica delle funzioni Matlab, visualizzabile nella *Command Window*, sfruttando il comando `help`.

Al fine di trasformare l'insieme delle funzioni fin qui descritte in una vera e propria toolbox, accessibile anche a chi non ha sviluppato le funzioni stesse, sono state realizzate delle `function` di supporto ai MEX-File. Queste, scritte in linguaggio Matlab¹³, fungono da tramite tra la linea di comando e i MEX-File per garantire che questi vengano sempre richiamati nel modo corretto, migliorando sia la facilità d'uso delle funzioni che la stabilità complessiva del software. Inoltre, inserendo al loro interno, prima dell'inizializzazione della `function`, dei commenti con la spiegazione del funzionamento del MEX-File associato si attiva automaticamente la funzionalità del comando `help`. In Figura 4.7 è riportato un esempio della schermata che si ottiene chiedendo l'`help` per la `function` `telone`.

La gestione dei dati di input

La creazione delle `function` di supporto ai MEX-File permette di rendere questi molto più versatili e semplici da utilizzare, rendendoli simili alle funzioni implementate dagli sviluppatori di *The MathWorks*.

Una delle caratteristiche che si vuole implementare in queste `function` riguarda la flessibilità dei dati in ingresso al MEX-File: poiché alcune delle subroutine implementate richiedono in input molti parametri che ben si prestano ad essere standardizzati¹⁴, è possibile, sfruttando l'apposita funzione `inputParser` associata al comando `varargin`, creare una struttura a oggetti per la gestione delle variabili di input.

Tale struttura si genera dichiarando tra le variabili in input, nell'inizializzazione della `function`, solamente quelle che si considerano strettamente necessarie per l'implementazione, seguite dal comando `varargin`.

Si consideri ad esempio il caso della `function` `telone` (i.e., la `function` di supporto al MEX-File che esegue la subroutine `telone`, cfr. par. 4.1.1), assumendo

¹³Le `function` di supporto ai MEX-File sono interamente riportate in Appendice: cfr. par. A.2, A.4, A.6, A.8, A.10.

¹⁴E.g. si pensi ai valori di prima approssimazione della velocità e del coefficiente di scabrezza secondo *Chezy* sulle barene (cfr. Tabella 2.1).

```

Command Window
File Edit Debug Desktop Window Help
>> help telone
[sol,iter]=telone(contorno,n_marsh,imax,toll,maglia,U_max,dETA0_dt,CHI,ETA0);
=====
Questo e' il programma telone originale che non tiene conto delle condizioni
al CONTORNO nei punti canale, divenuto subroutine e poi mex-file.
Su tutti i punti canale l'elevazione e' posta pari a zero.
L'unica modifica apportata, rispetto al modello originale consiste nel
risolvere l'equazione '-laplaciano di eta = KAPPA', anziche' =1
=====
INPUT:
- contorno: (INPUT OBBLIGATORIO) matrice contenente le caratteristiche
dell'area di studio (i.e. matrice delle condizioni al contorno):
LA CONVENZIONE SUI PUNTI DELLA LAGUNA E' LA SEGUENTE:
0 APPARTIENE ALLO SPECCHIO D'ACQUA
1 CONDIZIONE AL contorno DI DIRICHLET
2 CONDIZIONE AL contorno DI NEUMANN PER PUNTI NON CANALE
3 CONDIZIONE AL contorno DI NEUMANN PER PUNTI CANALE
4 PUNTI ANCORA CANALE MA DI INIZIO PER LA FUNZIONE DI AMPIEZZA
- n_marsh: (INPUT FACOLTATIVO) n di pixel barena presenti nella matrice
contorno, se non specificato la funzione lo calcola autonomamente
cercando gli elementi della matrice pari a zero.
- imax: (INPUT FACOLTATIVO) n massimo di iterazioni consentite per la
soluzione del sistema lineare col metodo del gradiente coniugato,
precondizionato con la decomposta incompleta di Cholesky;
(DEFAULT => imax = 10^4).
- toll: (INPUT FACOLTATIVO) tolleranza ammessa per la soluzione approssimata,
determinata con l'algoritmo del gradiente coniugato,
precondizionato con la decomposta incompleta di Cholesky.
(DEFAULT => toll = 10^-5).

PARAMETRI FISICI DEL PROBLEMA (OPZIONALI):
- maglia: dimensione del lato della maglia quadrata del reticolo di calcolo
(DEFAULT => maglia = 1 [m]).
- U_max: valore della velocita' massima sulla barena
(DEFAULT => U_max = 0.15 [m/s]).
- dETA0_dt: derivata dell'elevazione media istantanea rispetto al tempo
(DEFAULT => dETA0_dt = 7.5*10^-5).
- CHI: valore del coefficiente di scabrezza relativa di Chezy
(DEFAULT => CHI = 10 [m^.5/s]).
- ETA0: valore dell'elevazione media istantanea della marea
(DEFAULT => ETA0 = 0.35 [m]).

OUTPUT (OBBLIGATORI):
- sol: matrice in cui si descrive l'andamento della soluzione calcolata in
tutti i pixel barena.
- iter: n d'iterazioni dell'algoritmo del GCP.

NOTA1: e' possibile assegnare i parametri di input in tre modi diversi:
- assegnando solo la matrice 'contorno' la funzione calcolera'
autonomamente i valori di default per gli altri parametri e li
fx
OVR

```

Figura 4.7: Command Window di Matlab: esempio della schermata che si ottiene digitando il comando *help* seguito dal nome di una *function* di supporto ai MEX-File.

che l'unico parametro di input imprescindibile per l'esecuzione di questa funzione sia la matrice delle condizioni al contorno del problema, l'inizializzazione della **function** di supporto può avvenire così:

```
1 function [sol,iter]=telone(contorno,varargin);
```

In questo modo, quando si chiamerà la funzione dalla linea di comando, il primo input specificato verrà assegnato alla variabile `contorno`, mentre quelli successivi saranno memorizzati nella struttura **varargin**.

Al fine di poter assegnare dei valori predefiniti, qualora questi non siano specificati nella chiamata della **function**, la soluzione proposta in Matlab prevede la creazione di una variabile di tipo *oggetto*, le cui *proprietà*¹⁵ rappresentano tutte le variabili che possono essere definite nella chiamata della funzione. La creazione dell'*oggetto* finalizzato alla gestione delle variabili in input deve essere eseguita con la funzione **inputParser**:

```
2 % creo l'oggetto in cui organizzare i dai di input
3 in = inputParser;
```

L'oggetto `in` così creato è vuoto e necessita di essere strutturato in base al tipo di input che si vuole gestire.

Per dare una struttura all'oggetto `in` si scelgono i valori di *default* per le grandezze che possono non venire assegnate nella chiamata della **function** e si memorizzano in delle apposite variabili:

```
4 % scelgo i valori di default per gli input facoltativi
5 defaultN_marsh = length(contorno(contorno==0));
6 defaultImax = 10000;
7 defaultToll = 10(-5);
8 defaultMaglia = 1;
9 defaultU_max = 0.15;
10 defaultDETA0_dt = 0.000075;
11 defaultCHI = 10;
12 defaultETA0 = 0.35;
13 defaultZ = zeros(size(contorno));
```

Con la funzione **addRequired** si creano le proprietà dell'oggetto legate agli elementi di input che devono essere definiti obbligatoriamente nella chiamata della **function**, specificando il nome dell'oggetto in cui creare la proprietà, il nome da assegnare alla proprietà stessa e la tipologia di input atteso per questa classe:

```
14 % creo i parametri obbligatori
15 addRequired(in,'contorno',@isnumeric);
```

In modo del tutto analogo, con la funzione **addOptional** si creano le proprietà dell'oggetto legate agli elementi di input che possono essere definiti facolta-

¹⁵Nella programmazione ad oggetti, si definisce *proprietà* il nome che si assegna ad ogni campo, al quale si può assegnare una variabile di qualsiasi dimensione e tipologia, contenuto in un determinato oggetto.

tivamente, specificandone il valore di *default* da assegnare in caso di mancata assegnazione nella chiamata della **function**:

```
16 % creo i parametri facoltativi
17 addOptional(in,'n_marsh',defaultN_marsh,@isnumeric);
18 addOptional(in,'imax',defaultImax,@isnumeric);
19 addOptional(in,'toll',defaultToll,@isnumeric);
20 addOptional(in,'maglia',defaultMaglia,@isnumeric);
21 addOptional(in,'U_max',defaultU_max,@isnumeric);
22 addOptional(in,'dETA0_dt',defaultDETA0_dt,@isnumeric);
23 addOptional(in,'CHI',defaultCHI,@isnumeric);
24 addOptional(in,'ETA0',defaultETA0,@isnumeric);
25 addOptional(in,'z',defaultZ,@isnumeric);
```

Infine, si esegue la funzione **parse**, assegnando a questa: l'oggetto *in*, appositamente creato per la gestione dell'input; le variabili obbligatorie specificate nell'inizializzazione della **function**, rispettandone l'ordine; il comando **varargin** che rappresenta i parametri che possono venire definiti esternamente alla funzione in base alle scelte di chi programma.

```
26 % riempio l'oggetto in con gli appropriati valori
27 parse(in,contorno,varargin{:});
```

La funzione **parse** scorre le variabili assegnate in input nella chiamata della **function** e, con i valori di input che trova in **varargin**, compila un nuovo oggetto chiamato *in.Result*, caratterizzato dalla medesima struttura dell'oggetto *in*, assegnando alle proprietà i valori trovati nelle variabili in input; se la funzione **parse** non trova tra gli input valori riferiti ad alcune delle proprietà dell'oggetto *in*, allora assegna alle rispettive proprietà dell'oggetto *in.Result* i valori di *default* previsti.

A questo punto è possibile eseguire il MEX-File associato alla funzione, avendo la certezza di disporre di tutti i valori necessari all'esecuzione della subroutine in Fortran:

```
28 % si richiama il mex_file creato tramite F_telone.f90
29 [sol,iter]=F_telone(in.Results.contorno,...
30                   in.Results.n_marsh,...
31                   in.Results.imax,...
32                   in.Results.maglia,...
33                   in.Results.U_max,...
34                   in.Results.dETA0_dt,...
35                   in.Results.CHI,...
36                   in.Results.ETA0,...
37                   in.Results.toll,...
38                   in.Results.z);
```

La sintassi prevista dalla funzione **inputParser** per l'assegnazione dei parametri di input, riportata e spiegata nella descrizione delle **function** visualizzabile col comando **help**, consente a chi programma di specificare, quindi,

anche uno solo tra i parametri opzionali prestabiliti semplicemente digitando tra due apici il nome del parametro che si desidera assegnare, seguito dal valore numerico che si vuole trasmettere alla funzione (o dalla variabile contenente il valore o la matrice desiderati).

Un esempio pratico sarà più efficace della descrizione appena eseguita. Se si desidera utilizzare la funzione *telone* per la soluzione dell'equazione 2.23, ma in prima approssimazione si è interessati solamente a specificare, oltre alla matrice delle condizioni al contorno, la dimensione della maglia quadrata che questa matrice schematizza ed, eventualmente, l'elevazione dell'onda di marea, è sufficiente digitare il seguente comando:

```
% esecuzione della funzione telone
% pagliaga = matrice delle condizioni al contorno
% lato della maglia quadrata pari a 15 m
% elevazione di marea pari a 1 m
soluzione = telone(pagliaga, 'maglia', 15, 'ETA0', 1);
```

Così facendo la funzione **inputParser** assegnerà:

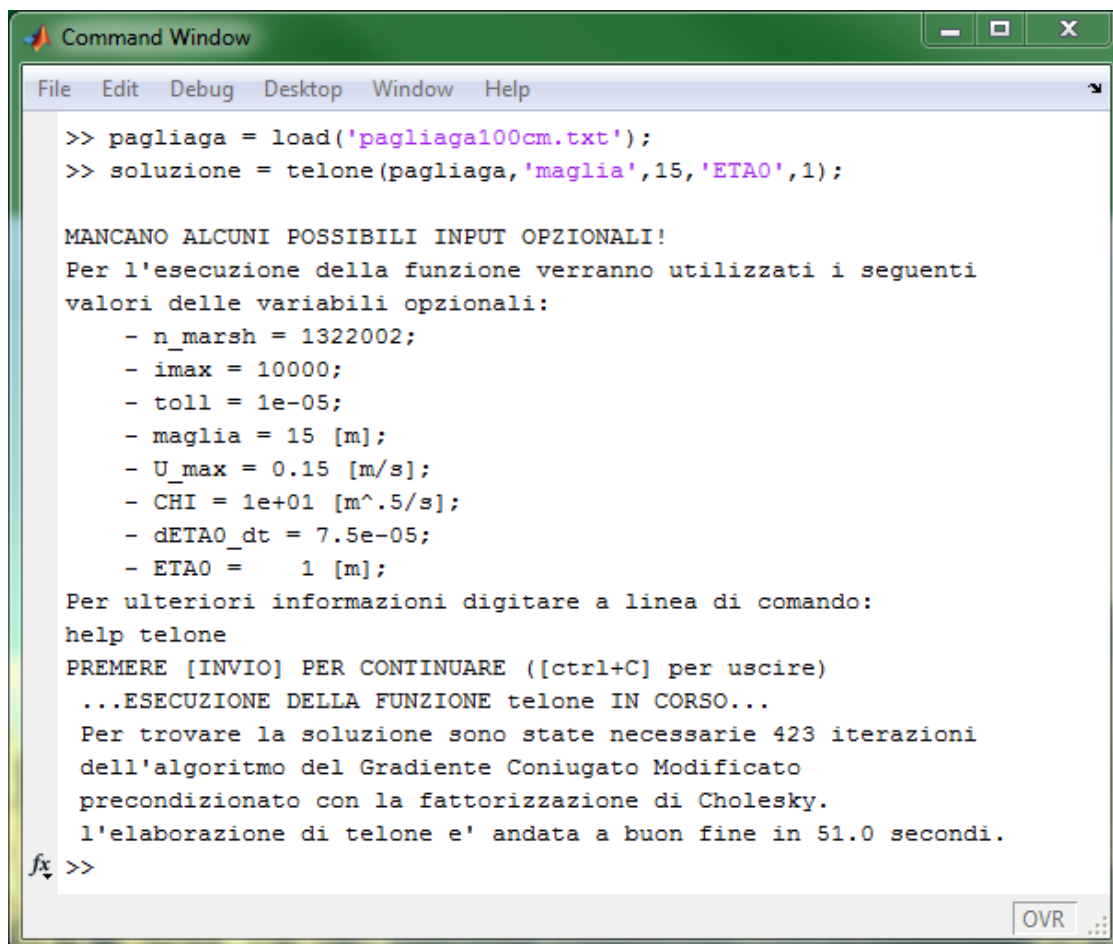
- alla proprietà `contorno` dell'oggetto `in.Results` la matrice `pagliaga` delle condizioni al contorno;
- alla variabile `in.Results.maglia` la dimensione della maglia quadrata del reticolo di calcolo di 15 m;
- alla variabile `in.Results.ETA0` un'elevazione dell'onda di marea pari a $\eta_0 = 1$ m;
- a tutte le altre proprietà dell'oggetto `in.Results` i valori di default previsti.

Al fine di evitare errori di assegnazione dei dati, come illustrato nel listato completo della **function** `telone` (cfr. appendice A.2), prima di eseguire il MEX-File `F_telone.mexw32`¹⁶, la funzione esegue una verifica sui dati di input assegnati e, se – come nel caso di questo esempio – i dati assegnati non corrispondono a tutti quelli previsti, scrive a video i valori che verranno utilizzati per l'implementazione del MEX-File, chiedendo conferma all'utente.¹⁷ In Figura 4.8 è illustrata la schermata della Command Window di Matlab che si ottiene eseguendo l'esempio appena descritto.

In questo modo sono state ottenute delle funzioni che ben si integrano con l'ambiente di programmazione del software Matlab e che garantiscono un utilizzo semplice ed immediato del modello numerico in esame, soprattutto se paragonate all'implementazione dello stesso modello in Fortran da cui si è partiti.

¹⁶Oppure `F_telone.mexw64` a seconda del sistema in cui si sta operando.

¹⁷La medesima procedura di controllo è stata implementata in tutte le **function** di supporto ai MEX-File realizzati (cfr. par. A.4, A.6, A.8, A.10).



```
Command Window
File Edit Debug Desktop Window Help
>> pagliaga = load('pagliaga100cm.txt');
>> soluzione = telone(pagliaga,'maglia',15,'ETA0',1);

MANCANO ALCUNI POSSIBILI INPUT OPZIONALI!
Per l'esecuzione della funzione verranno utilizzati i seguenti
valori delle variabili opzionali:
  - n_marsh = 1322002;
  - imax = 10000;
  - toll = 1e-05;
  - maglia = 15 [m];
  - U_max = 0.15 [m/s];
  - CHI = 1e+01 [m^.5/s];
  - dETA0_dt = 7.5e-05;
  - ETA0 = 1 [m];
Per ulteriori informazioni digitare a linea di comando:
help telone
PREMERE [INVIO] PER CONTINUARE ([ctrl+C] per uscire)
...ESECUZIONE DELLA FUNZIONE telone IN CORSO...
Per trovare la soluzione sono state necessarie 423 iterazioni
dell'algorithmo del Gradiente Coniugato Modificato
precondizionato con la fattorizzazione di Cholesky.
l'elaborazione di telone e' andata a buon fine in 51.0 secondi.
fx >>
```

Figura 4.8: Command Window di Matlab: esempio dell'assegnazione dei dati di input alla **function** `telone`, sfruttando le funzionalità del comando `inputParser`.

Capitolo 5

Analisi della forma delle aree di drenaggio dei canali

Nel Capitolo 4 è stato trattato il metodo utilizzato per importare in Matlab il codice di calcolo sviluppato per risolvere alle differenze finite l'equazione 2.23.

Sfruttando il lavoro esposto nel Capitolo 4, il modello matematico è stato applicato allo studio della porzione di laguna appartenente alla Palude Pagliaga (cfr. Capitolo 3).

Nella prima parte di questo capitolo si descrivono i risultati ottenuti dall'esecuzione dei codici presentati nel Capitolo 4 e riportati in Appendice A. Nella seconda parte si prendono in considerazione tutte le 63 aree di drenaggio afferenti ai canali lagunari considerati, al fine di definire, in modo oggettivo un metodo per la caratterizzazione delle forme che queste assumono e proporre una classificazione. Infine, sfruttando questa classificazione, è stato scelto un insieme di aree di drenaggio ed è stato eseguito un confronto tra le caratteristiche da esse presentate.

5.1 I risultati del modello per la Palude Pagliaga

Dall'esecuzione delle funzioni descritte nel Capitolo 4 si ottengono una serie di matrici che, visualizzate utilizzando la funzione `surface`, rappresentano una mappa della Palude Pagliaga. In tale mappa si illustrano schematicamente i valori assegnati nei nodi del reticolo di calcolo utilizzato per la discretizzazione della barena.

I passaggi della programmazione in Matlab per la rappresentazione grafica di queste matrici, che qui vengono omessi per brevità, sono consultabili nel listato del codice principale scritto in Matlab, riportato in Appendice A.1.

Come illustrato in Figura 5.1, nella quale si riporta la *Command Window* ottenuta eseguendo il file `AreeDiDrenaggio2.2.m`, la prima funzione che viene richiamata dal programma principale è la `function` `telone` (cfr. A.2). Questa, per le condizioni al contorno di Figura 3.5, impiega 314 iterazioni dell'algoritmo del gradiente coniugato, preconditionato con la fattorizzazione di *Cholewsky*, per la soluzione dell'equazione (2.23).

```

Command Window
File Edit Debug Desktop Window Help
=====
===== PROVA TOOLBOX TELONE =====
=====
I dati che verranno utilizzati in questa simulazione sono:
- numero di righe_N1 = 1692;
- numero di colonne_N2 = 1680;
- num. di incognite nel modello del telone = 1.322e+06;
- dimensione del lato della maglia quadrata = 1.000 {m};
- valore del coefficiente di Chezy = 10.000 {m^0.5/s};
- valore della velocita' massima sulla barena = 0.150{m/s};
- valore dell'elevazione media istantanea = 0.350;
- derivata dell'elevazione media istantanea
  rispetto al tempo, nell'istante considerato = 0.000075.
=====
INIZIO DELL'ELABORZIONE DEI DATI:
=====
----- function telone -----
...ESECUZIONE DELLA FUNZIONE telone IN CORSO...
Per trovare la soluzione sono state necessarie 314 iterazioni
dell'algoritmo del Gradiente Coniugato Modificato
precondizionato con la fattorizzazione di Cholesky.
l'elaborazione di telone e' andata a buon fine in 37.1 secondi.
=====
----- incidenza barene -----
l'elaboraz. di incidenza_barene e' andata a buon fine in 0.9 secondi
=====
----- function ampiezza -----
l'elaboraz. di ampiezza e' andata a buon fine in 3 secondi
=====
----- incidenza canali -----
l'elaboraz. di incidenza_canali e' andata a buon fine in 0.6 secondi
=====
----- incidenza -----
calcolo la matrice "incidenza" delle direzioni di drenaggio:
incidenza = incidenza_rete + incidenza_bar
=====
----- segna rami -----
...ESECUZIONE DELLA FUNZIONE segnarami IN CORSO...
l'elaborazione di segnarami e' andata a buon fine in 47.2 secondi.
=====
fx >> |
OVR

```

Figura 5.1: Command Window di Matlab rappresentante l'output che viene visualizzato a schermo dalle **function** esposte nel Capitolo 4.

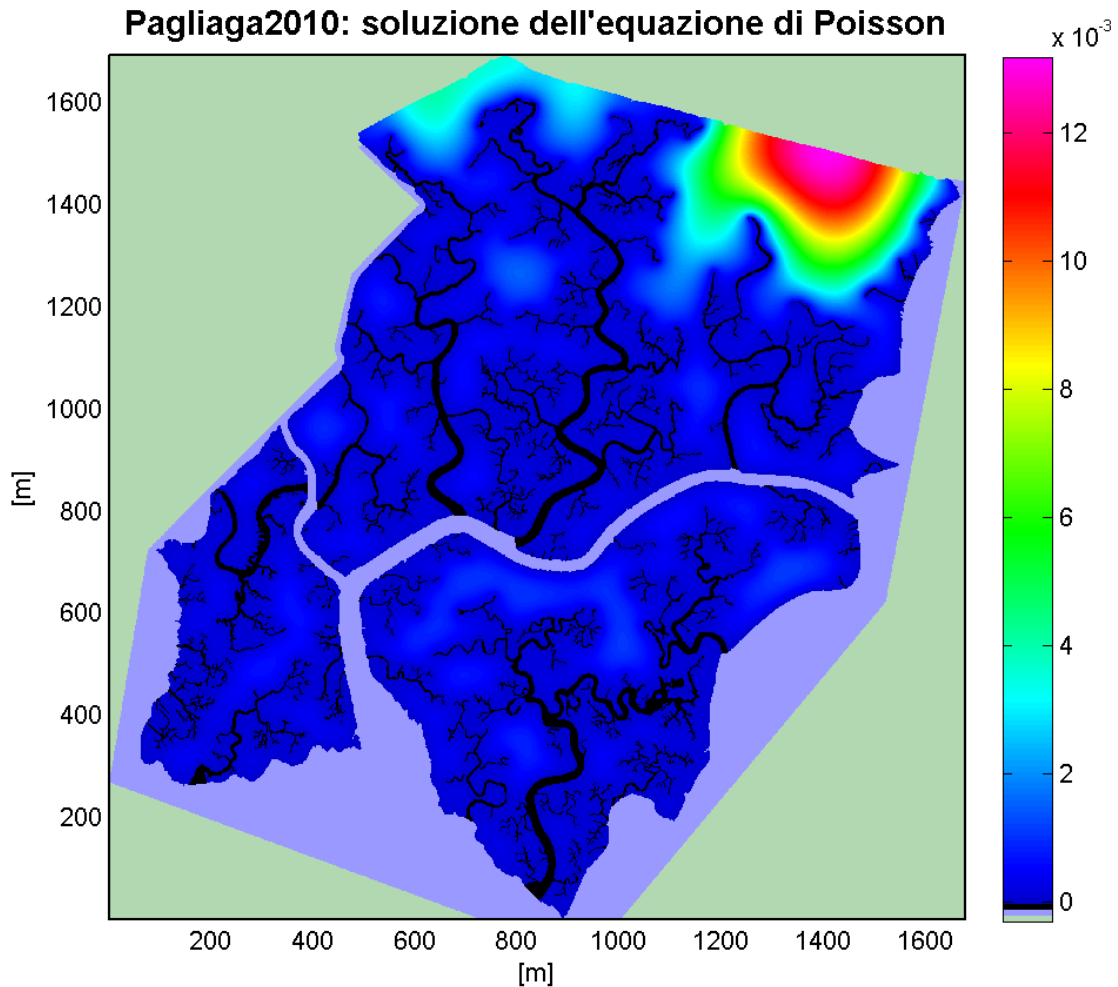


Figura 5.2: Soluzione dell'equazione (2.23) per le condizioni al contorno della Paluge Pagliaga, illustrate in Figura 3.5.

La soluzione calcolata viene quindi memorizzata in una matrice contenente la quota (in metri) della superficie libera, per tutti i pixel definiti "barena" nella matrice delle condizioni al contorno (i.e, con $ID = 0$ nella matrice `contorno`, cfr Figura 3.5). Tale soluzione è illustrata in Figura 5.2 (cfr. par. 4.1.2).

Dato l'andamento della superficie libera, la `function` `incidenzabarene` calcola le direzioni di drenaggio su tutta la superficie delle barene, seguendo la direzione del massimo gradiente della superficie libera di (cfr. par. 4.1.29).

Le direzioni di drenaggio risultanti, definite secondo la notazione di Figura 4.2, sono riportate in Figura 5.3. Si nota come le direzioni di drenaggio sulle barene puntino nella direzione dei canali, dando un'idea visiva di quali siano i confini delle aree drenate da ciascun canale. Inoltre, osservando le direzioni di drenaggio lungo il contorno che separa le barene dai bassifondi, è possibile verificare che il drenaggio avviene direttamente verso la laguna, esattamente come ci si aspettava che fosse.

In modo analogo a quanto fatto per l'andamento della superficie libera e per

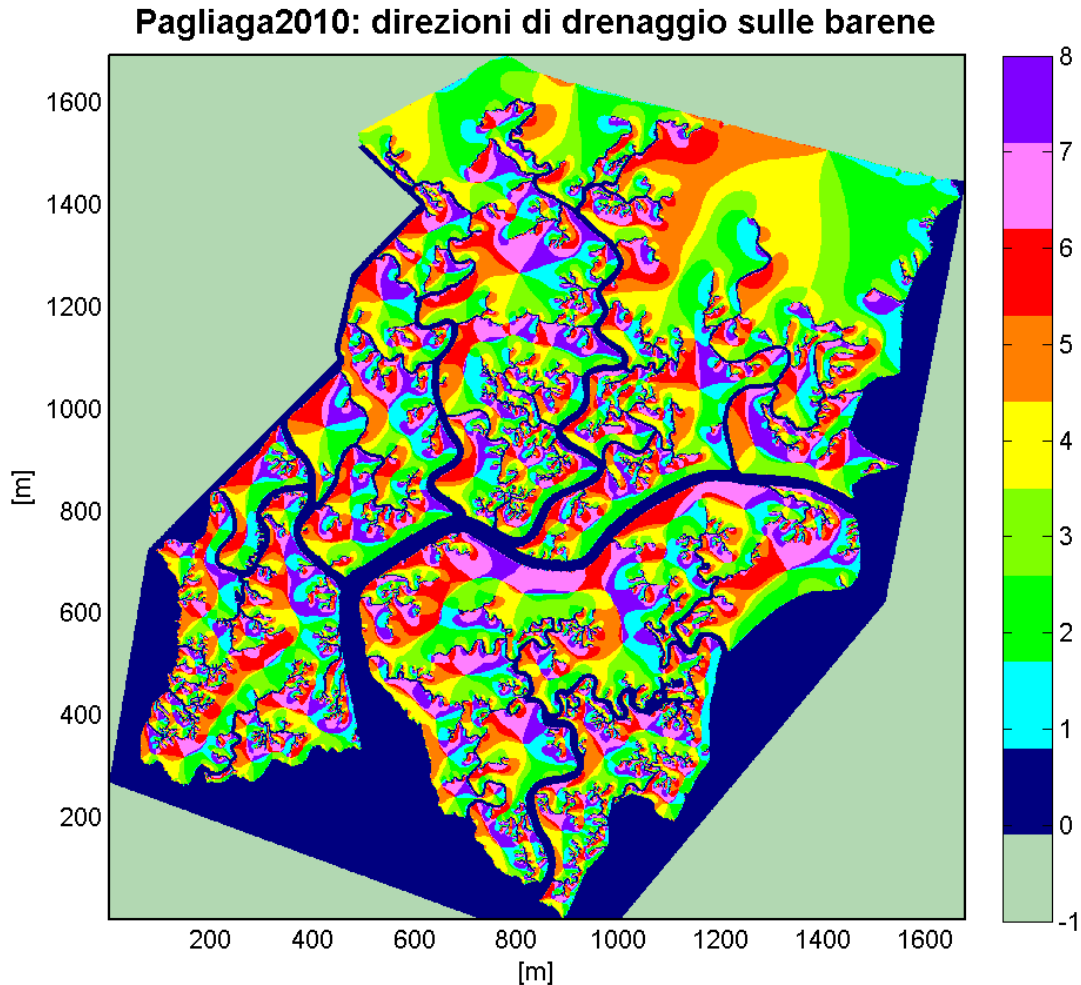


Figura 5.3: Rappresentazione delle direzioni di drenaggio sulle barene della Palude Pagliaga, calcolate dall'andamento della superficie libera illustrato in Figura 5.2, definite secondo la notazione di Figura 4.2.

le direzioni di drenaggio, con le **function** `ampiezza` e `incidenzacanali` si calcolano, rispettivamente, la *Funzione di Ampiezza* e le direzioni di drenaggio, lungo la rete dei canali che incidono le barene della Palude Pagliaga. Il risultato che si ottiene dall'esecuzione di queste due funzioni è riportato nelle Figure 5.4 e 5.5.

Con le matrici `incidenza_bar` e `incidenza_rete`, avendo creato una matrice `sezioni` (cfr. par. 4.1.3), nella quale sono state identificate con dei numeri identificativi (*ID*), compresi tre 9 e 71, le sezioni immediatamente a monte della sezione iniziale di ogni canale, è possibile eseguire la **function** `segnarami` per individuare i bacini di drenaggio afferenti a ogni canale. In Figura 5.6 sono illustrate le aree afferenti ai vari canali colorate in base al numero identificativo con il quale è stata individuata la sezione di chiusura del canale.

Visto il numero consistente di canali, per aumentare la leggibilità dell'immagine risultante, è stato scelto di ricolorare le aree con una sequenza casuale di

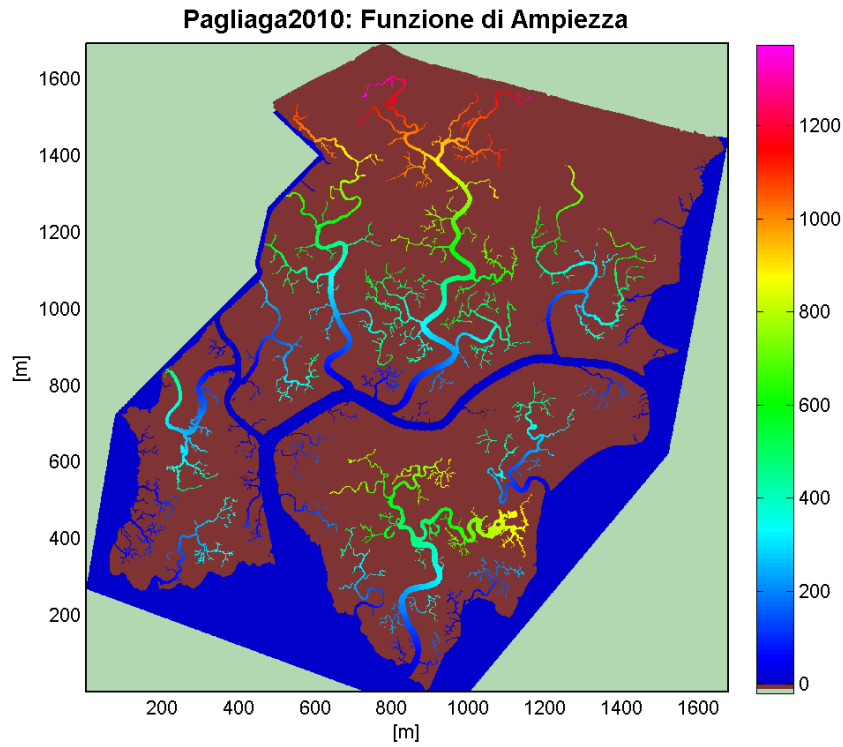


Figura 5.4: Andamento della *Funzione di Ampiezza* lungo la rete di canali della Palude Pagliaga.

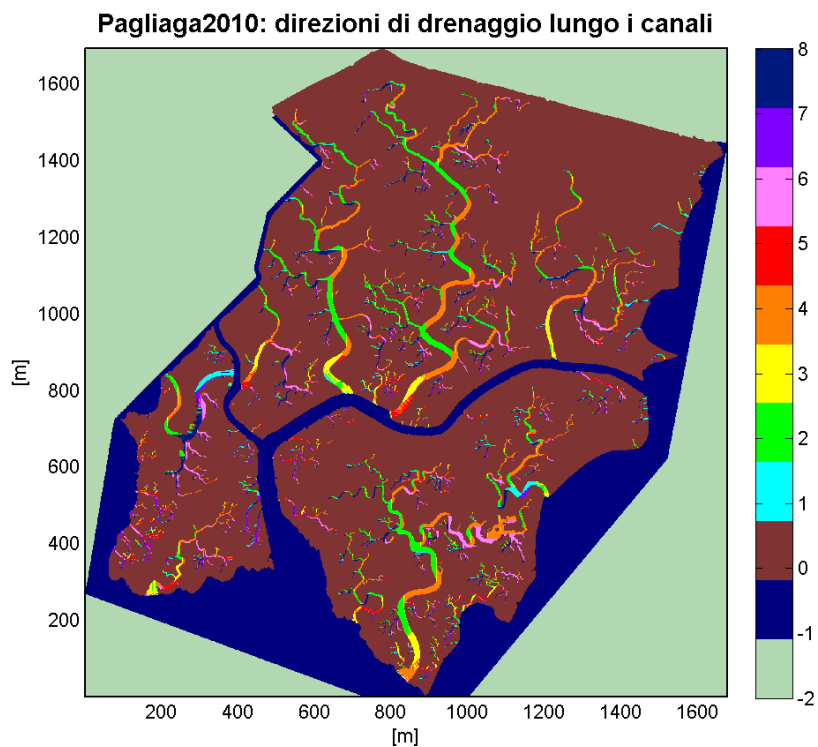


Figura 5.5: Rappresentazione delle direzioni di drenaggio lungo la rete di canali della Palude Pagliaga, calcolate dall'andamento della *Funzione di Ampiezza* illustrato in Figura 5.4, definite secondo la notazione di Figura 4.2.

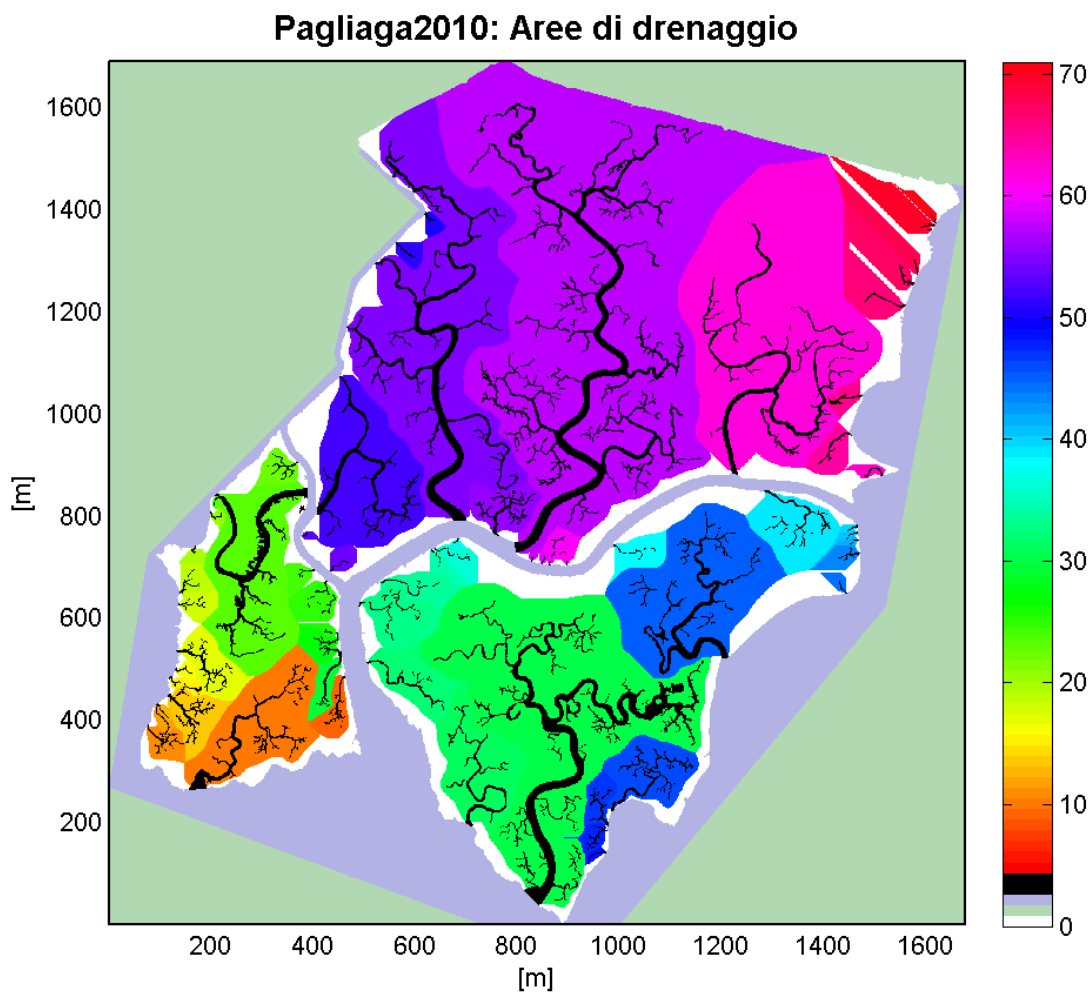


Figura 5.6: Aree di drenaggio dei canali a marea solcanti la Palude Pagliaga, colorate in base al numero identificativo *ID* con il quale sono state identificate le sezioni di chiusura dei canali.

colori. Il risultato di quest'operazione è illustrato in Figura 5.7.

Analizzando queste due figure risulta chiaro come le aree di drenaggio dei canali a marea, pur all'interno della medesima barena, assumano forme e dimensioni estremamente variabili. Tali forme dipendono non solo dalla morfologia del canale che le drena, ma anche dalla presenza o meno di altri canali e dall'esistenza di eventuali contorni impermeabili che delimitino le barene.

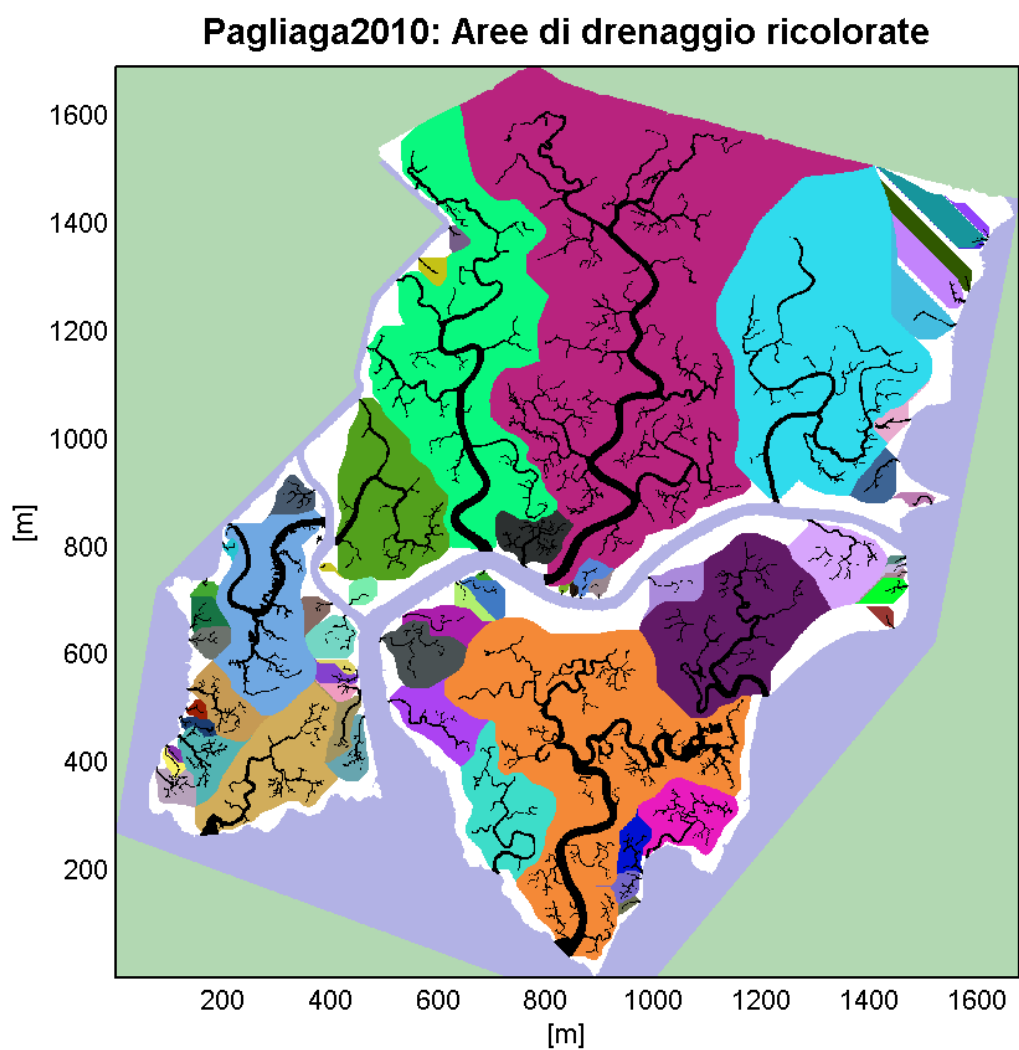


Figura 5.7: Aree di drenaggio dei canali a marea solcanti la Palude Pagliaga, ricolorate per rendere più facilmente distinguibile le aree adiacenti.

5.2 Studio e classificazione della forma delle aree di drenaggio

Per dare una caratterizzazione il più possibile oggettiva della forma che assumono le aree di drenaggio dei canali a marea, si è cercato innanzitutto di capire se esiste una similarità tra le forme che queste assumono. Si è pertanto considerato un comune sistema di riferimento, che renda più agevole la comparazione tra i bacini lagunari individuati.

Per eseguire questa analisi è stato inserito nel programma principale del codice di calcolo, un algoritmo per l'individuazione dei perimetri che delimitano tutti i bacini lagunari presenti nella matrice illustrata in Figura 5.6.

Tali perimetri sono stati quindi rappresentati in un sistema di riferimento cartesiano, centrato nel baricentro delle rispettive aree di drenaggio. Un esempio di alcune delle forme delle aree di drenaggio che sono state così ricavate, in relazione alla morfologia del canale che le drena, è illustrato in Figura 5.8. Per questioni di spazio, dato il gran numero di forme analizzate, è stato scelto di non riportare per intero tutte le immagini così sviluppate.

Dall'analisi delle immagini contenenti i perimetri è possibile verificare l'effettiva esistenza di una direzione predominante nello sviluppo delle forme che i bacini lagunari assumono: questa particolare direzione può essere approssimata con quella individuata dalla retta passante per il centro della sezione iniziale del canale e il baricentro dell'area che questo drena.

È possibile quindi, tramite la semplice rotazione del sistema di riferimento cartesiano, rappresentare con il medesimo orientamento tutte le forme delle aree di drenaggio dei canali solcanti la Palude Pagliaga. In Figura 5.9 sono riportati i perimetri illustrati in Figura 5.8, rappresentati rispetto al nuovo sistema di riferimento (n, s) . La direzione s è quella individuata dalla retta passante per il centro della sezione di chiusura del canale e il baricentro della corrispondente area di drenaggio; mentre n è la direzione di ortogonale a s .

Dall'analisi dei grafici riportati in Figura 5.9 si nota come il nuovo sistema di riferimento individuati, in quasi tutti i casi analizzati la direzione di drenaggio privilegiata dalla morfologia del canale. Il sistema di coordinate (n, s) ha infatti il pregio di rappresentare la forma delle aree di drenaggio secondo un'orientamento individuabile in modo semplice e oggettivo, comune a tutti i bacini di drenaggio considerati. Questo permette di poter confrontare direttamente i parametri di forma che, altrimenti, avrebbero valenza solo per uno specifico bacino di drenaggio.

Ricordando che il sistema di riferimento così determinato ha l'origine nel baricentro delle aree di drenaggio (i.e., la media delle coordinate lungo s ed n è nulla), per ognuno dei bacini lagunari illustrati in Figura 5.6 sono stati calcolati i seguenti parametri:

- La deviazione standard, calcolata lungo le direzioni s ed n , per descrivere

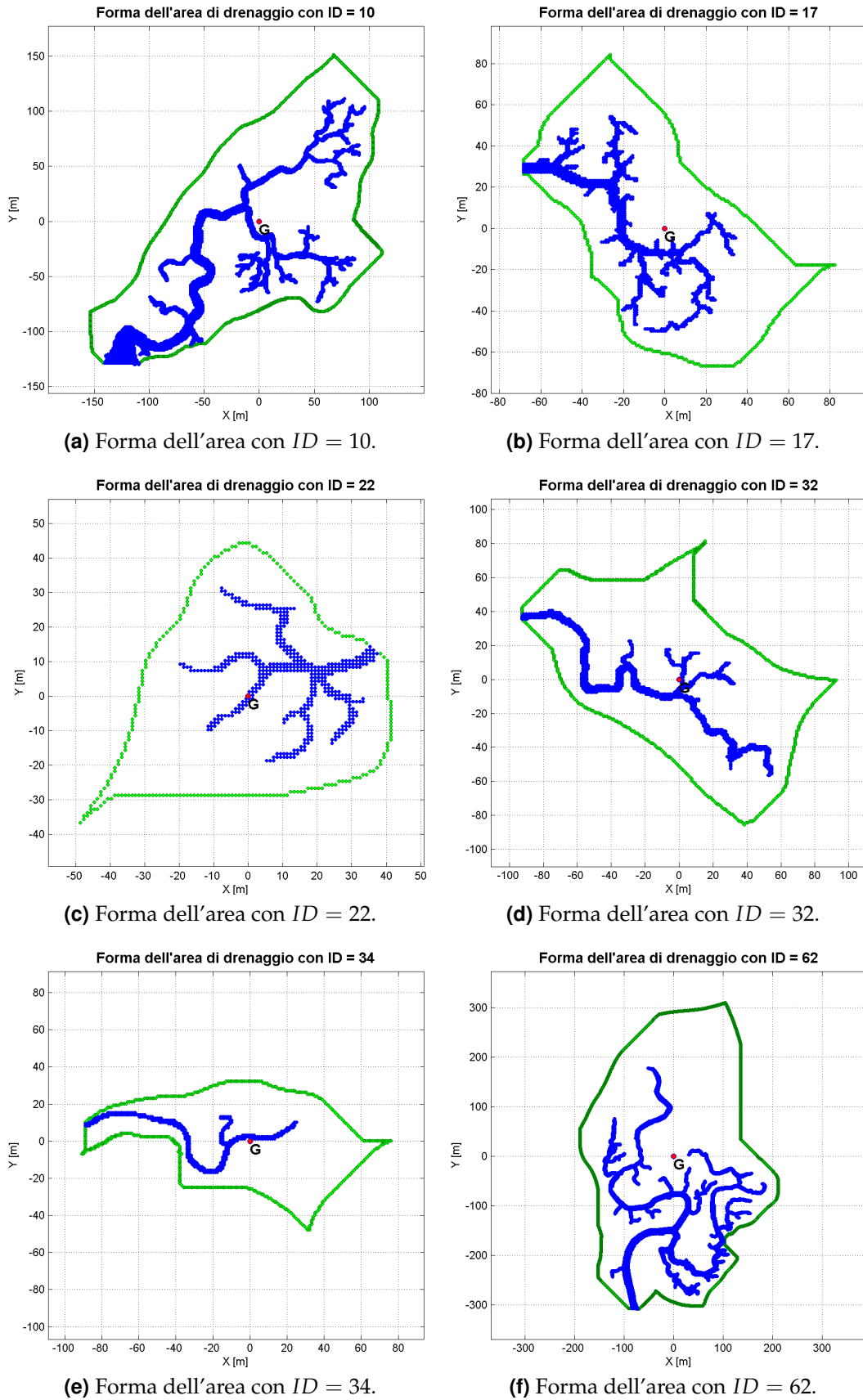
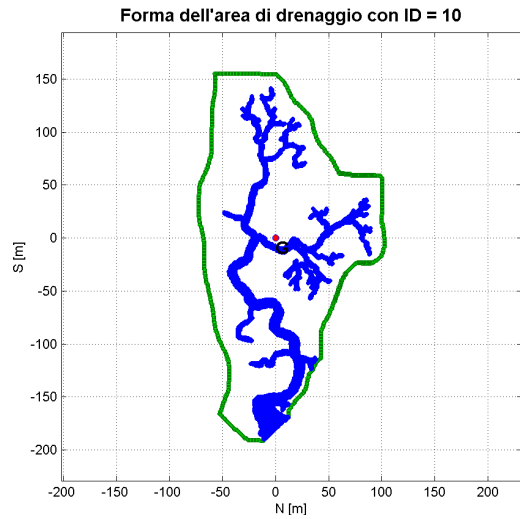
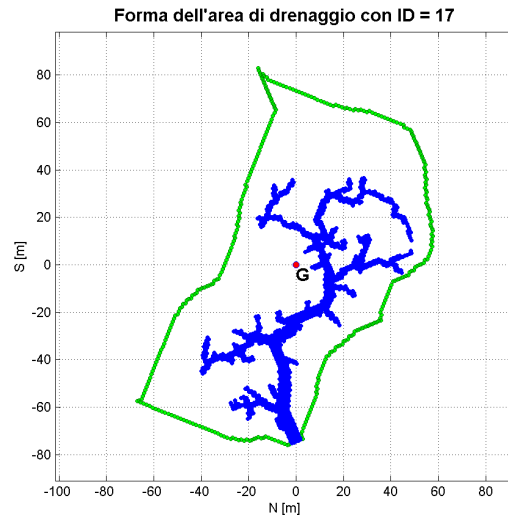


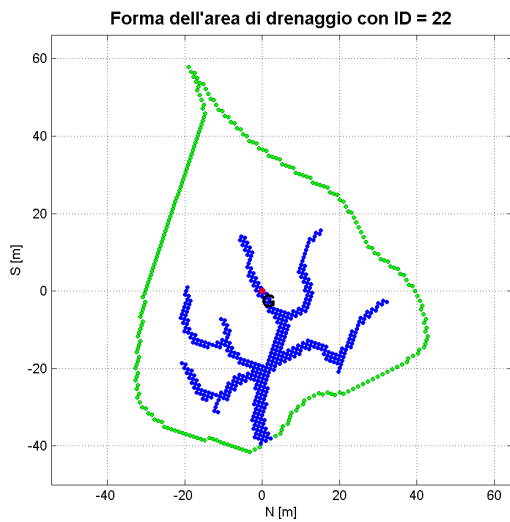
Figura 5.8: Forma di sei aree di drenaggio ottenute dalla matrice illustrata in Fig. 5.6.



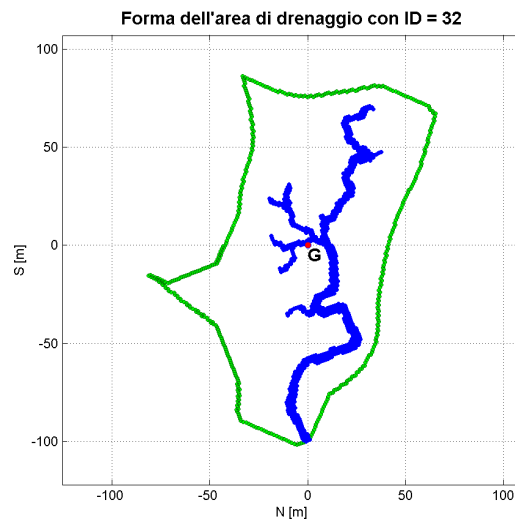
(a) Forma dell'area con $ID = 10$.



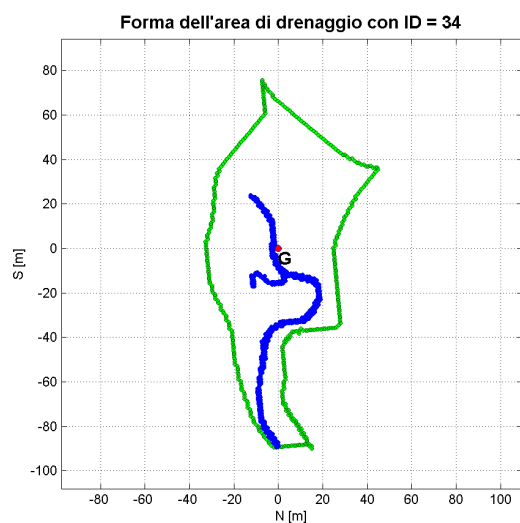
(b) Forma dell'area con $ID = 17$.



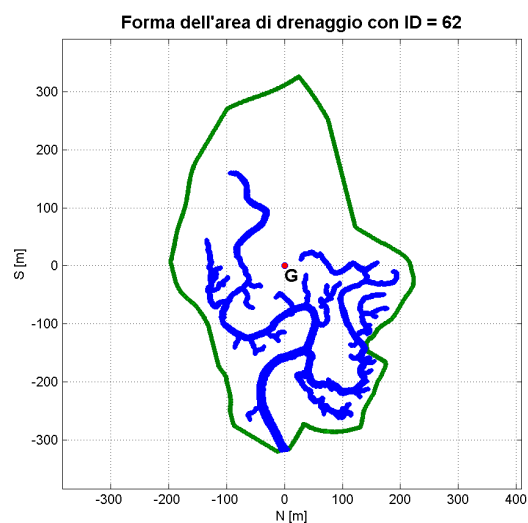
(c) Forma dell'area con $ID = 22$.



(d) Forma dell'area con $ID = 32$.



(e) Forma dell'area con $ID = 34$.



(f) Forma dell'area con $ID = 62$.

Figura 5.9: Forma delle sei aree di drenaggio di Fig. 5.8 rispetto al riferimento (n, s) .

l'estensione dell'area di drenaggio considerata, rispetto agli assi baricentrici:

$$\sigma_s = \sqrt{\frac{1}{N} \sum_{k=1}^N n_k^2} \quad (5.1)$$

$$\sigma_n = \sqrt{\frac{1}{N} \sum_{k=1}^N s_k^2} \quad (5.2)$$

in cui N è il numero di punti che appartengono all'area di drenaggio considerata.

- Il rapporto tra le deviazioni standard rispetto all'asse n e s , per definire se l'area di drenaggio ha una forma più o meno allungata lungo s , in relazione al suo sviluppo lungo l'asse n :

$$R_\sigma = \frac{\sigma_n}{\sigma_s} \quad (5.3)$$

- Il momento di terzo ordine $\mu_s^{[3]}$ (la *Skewness*) rispetto all'asse s , per definire da quale parte dell'asse è situata la maggior parte dei punti che appartengono al bacino di drenaggio:

$$\mu_s^{[3]} = \frac{1}{N} \sum_{k=1}^N n_k^3 \quad (5.4)$$

- Il momento di quarto ordine $\mu_s^{[4]}$ (la *Kurtosis*) rispetto all'asse s , per definire l'importanza della dispersione dei punti dell'area rispetto all'asse s :

$$\mu_s^{[4]} = \frac{1}{N} \sum_{k=1}^N n_k^4 \quad (5.5)$$

I risultati ottenuti per tutte le aree di drenaggio dei canali lagunari presenti nella Palude Pagliaga sono riportati nelle Tabelle 5.1 e 5.2; mentre i grafici delle forme delle aree di drenaggio, per la gran quantità di spazio che richiedono, vengono riportati nell'Appendice B.

L'analisi delle Tabelle 5.1 e 5.2, supportata dallo studio dei grafici della forma delle aree di drenaggio (e.g. Figura 5.9; cfr. Appendice B), ha posto in evidenza che, tra i parametri calcolati, quelli che forniscono più informazioni riguardanti la forma dei bacini lagunari sono: le deviazioni standard σ_s , σ_n e il valore assunto da R_σ .

Una prima analisi riguarda l'andamento delle deviazioni standard σ_s e σ_n in funzione dell'estensione area di drenaggio. La distribuzione risultante, illustrata in Figura 5.10, evidenzia come, all'aumentare dell'estensione del bacino, σ_s e σ_n tendano entrambe ad aumentare. Questo è significativo del fatto che, all'interno della porzione di laguna analizzata, non c'è un valore limite di larghezza o di

Tabella 5.1: Parametri di forma calcolati, per le aree di drenaggio con ID compreso tra 9 e 55, per il sistema di riferimento (n, s) .

ID	Area (m^2)	Perimetro (m)	σ_s (m)	σ_n (m)	R_σ	$\mu_s^{[3]}$ (m^3)	$\mu_s^{[4]}$ (m^3)
9	4324	349.6	16.52	32.69	1.98	7.24e+02	1.84e+05
10	33522	921.7	41.44	81.91	1.98	3.24e+04	7.05e+06
11	3725	291.3	17.74	22.08	1.24	-7.39e+02	2.31e+05
12	931	163.9	8.23	14.16	1.72	-3.33e+02	1.10e+04
13	513	102.6	5.03	9.83	1.95	9.83e+00	1.29e+03
14	8010	494.4	31.99	30.30	0.95	1.88e+03	2.42e+06
15	1335	198.2	8.14	18.81	2.31	2.89e+01	9.03e+03
16	914	124.5	9.49	9.56	1.01	-2.08e+02	1.63e+04
17	9203	471.0	28.61	38.99	1.36	-8.79e+02	1.48e+06
18	3151	221.0	15.53	18.41	1.19	6.64e+02	1.17e+05
19	3055	259.4	19.51	15.25	0.78	5.71e+02	3.35e+05
20	1073	151.9	13.20	7.63	0.58	6.52e+02	6.15e+04
21	658	117.1	9.01	7.47	0.83	-4.29e+01	1.39e+04
22	3781	286.9	18.55	20.59	1.11	2.12e+03	2.53e+05
23	43311	1118.7	63.44	92.39	1.46	-1.20e+04	3.41e+07
24	2356	227.0	15.53	15.36	0.99	4.93e+02	1.24e+05
25	5475	289.5	21.88	22.20	1.01	-1.36e+03	4.75e+05
26	933	183.8	5.73	19.12	3.34	2.88e+01	2.33e+03
27	2048	231.7	10.34	20.28	1.96	2.71e+02	2.35e+04
28	1493	204.0	9.49	17.01	1.79	-1.01e+02	1.70e+04
29	3901	458.3	17.03	34.93	2.05	6.94e+03	6.36e+05
30	138953	2369.1	124.23	150.20	1.21	3.67e+05	6.40e+08
31	26026	831.2	47.56	60.05	1.26	9.16e+03	9.34e+06
32	13140	576.3	28.14	47.47	1.69	1.68e+03	1.44e+06
33	12147	470.5	29.38	37.18	1.27	4.91e+03	1.60e+06
34	6139	434.0	16.39	36.41	2.22	9.40e+02	1.64e+05
35	2188	239.2	9.88	24.68	2.50	4.07e+02	2.35e+04
36	3543	264.2	14.85	23.53	1.58	1.24e+02	1.02e+05
37	332	76.8	7.23	4.21	0.58	-3.12e+01	5.87e+03
38	5959	399.8	24.45	26.67	1.09	2.85e+03	9.88e+05
39	17247	554.5	37.02	43.36	1.17	-1.34e+04	3.92e+06
40	391	95.9	4.20	8.81	2.09	3.99e-01	6.03e+02
41	664	256.0	8.00	23.98	3.00	-6.72e+02	1.98e+04
42	565	242.5	7.10	27.27	3.84	-3.94e+02	9.53e+03
43	2750	245.6	11.45	22.02	1.92	2.11e+02	3.62e+04
44	1123	149.0	8.35	13.12	1.57	1.00e+02	9.78e+03
45	62165	1135.6	92.51	66.77	0.72	-4.68e+04	1.36e+08
46	15597	606.8	39.28	41.32	1.05	2.63e+03	4.63e+06
47	4110	329.9	20.05	25.42	1.27	-3.08e+02	3.33e+05
48	2043	249.8	13.26	15.89	1.20	-2.23e+02	5.80e+04
49	642	120.1	6.12	12.05	1.97	-2.00e+01	2.85e+03
50	1264	145.5	10.04	11.57	1.15	2.28e+02	2.08e+04
51	2036	198.9	13.11	14.71	1.12	-7.29e+02	6.94e+04
52	45447	961.4	77.36	61.53	0.80	-1.20e+05	7.53e+07
53	445	118.7	6.07	10.20	1.68	1.28e+02	4.31e+03
54	2259	180.5	12.57	15.24	1.21	-2.97e+01	5.16e+04
55	131196	2552.9	77.17	201.45	2.61	-5.90e+04	8.10e+07

Tabella 5.2: Parametri di forma calcolati, per le aree di drenaggio con ID compreso tra 56 e 71, per il sistema di riferimento (n, s) .

ID	Area (m^2)	Perimetro (m)	σ_s (m)	σ_n (m)	R_σ	$\mu_s^{[3]}$ (m^3)	$\mu_s^{[4]}$ (m^3)
56	8530	462.2	37.90	25.85	0.68	-1.05e+04	3.75e+06
57	315232	3397.4	154.78	235.89	1.52	-1.13e+06	1.37e+09
58	255	78.7	4.78	5.88	1.23	-2.17e+01	9.96e+02
59	290	69.1	4.25	6.42	1.51	2.06e+01	6.18e+02
60	2386	242.2	13.97	19.07	1.36	9.84e+02	8.88e+04
61	1044	160.8	10.58	11.32	1.07	-7.34e+01	2.54e+04
62	155648	1784.8	91.71	153.95	1.68	7.94e+04	1.60e+08
63	1276	184.4	7.95	17.86	2.25	-3.78e+01	8.69e+03
64	5314	309.5	18.40	27.62	1.50	-1.53e+03	2.27e+05
65	2610	334.0	21.03	20.05	0.95	5.97e+03	6.80e+05
66	10024	594.5	32.89	44.49	1.35	3.25e+04	4.70e+06
67	8330	472.7	14.89	50.05	3.36	-2.50e+02	8.94e+04
68	6664	558.1	10.69	65.75	6.15	1.35e+02	3.16e+04
69	300	537.6	17.81	78.04	4.38	4.64e+03	2.84e+05
70	8100	532.8	15.01	55.23	3.68	-1.18e+03	1.15e+05
71	1449	238.1	6.02	23.99	3.99	-1.25e+01	2.76e+03

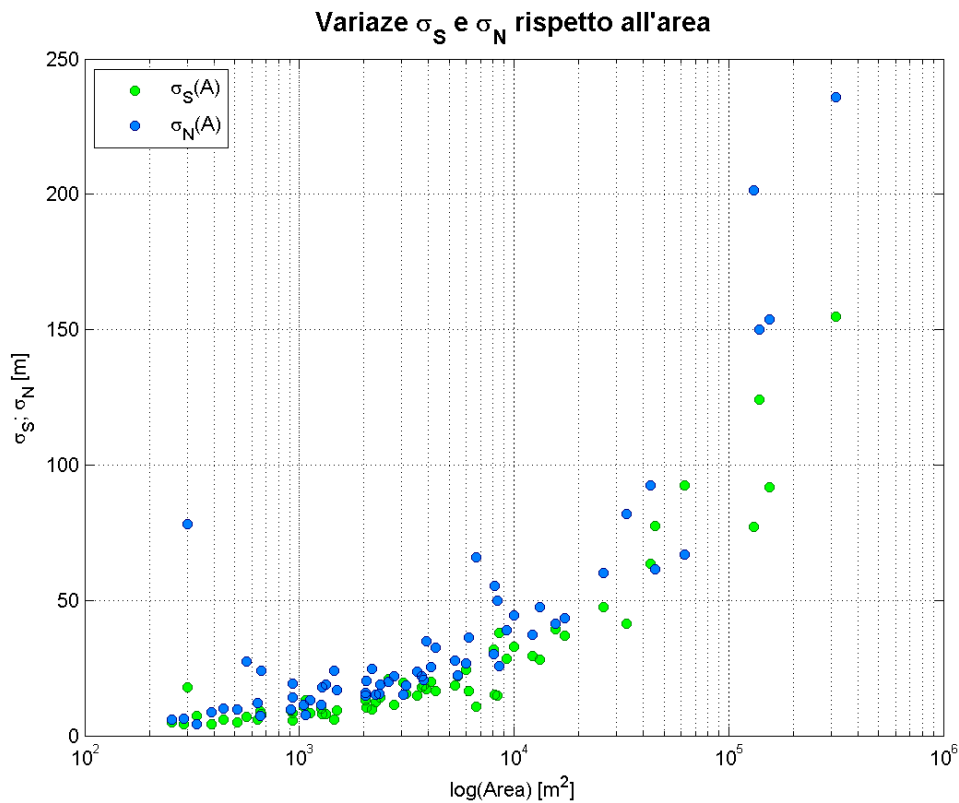


Figura 5.10: Distribuzione delle varianze σ_n (5.2) e σ_s (5.1), in funzione dell'estensione della superficie drenata.

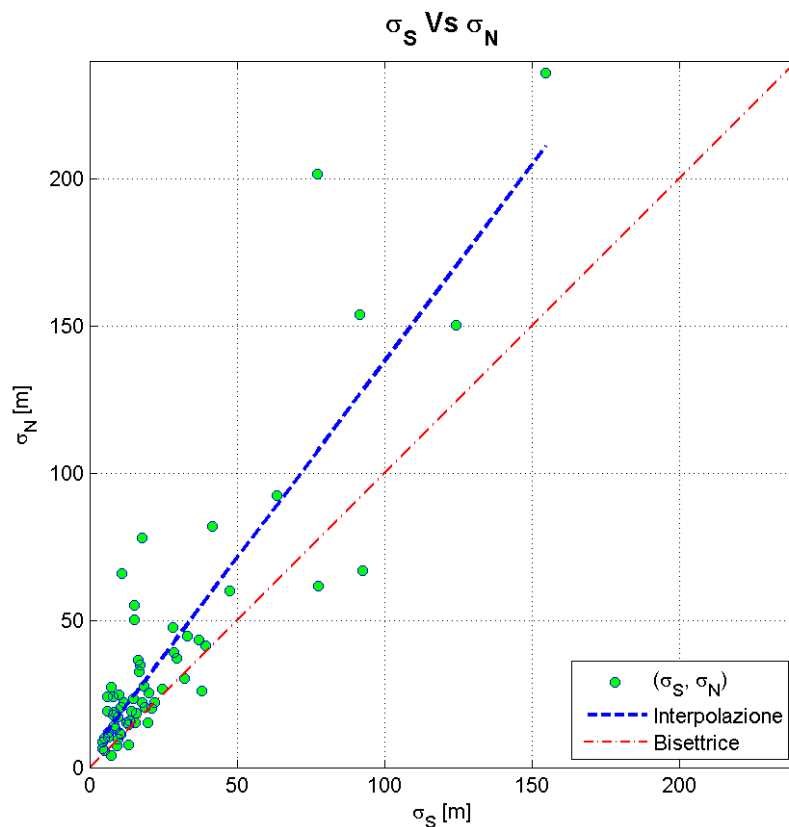


Figura 5.11: Confronto diretto delle varianze σ_n e σ_s , per ognuno dei bacini lagunari di Figura 5.6, riportante nelle Tabelle 5.1 e 5.2.

lunghezza oltre il quale questi valori si stabilizzano, perciò la forma che i bacini lagunari assumono risulta indipendente della grandezza dell'area di drenaggio.

Inoltre, in Figura 5.10, si nota che i valori assunti da σ_n risultano generalmente più grandi di quelli calcolati per σ_s . Questo, in termini di forma, significa che, indipendentemente dalla grandezza dei bacini lagunari, questi tenderanno ad essere più allungati nella direzione della coordinata s .

Una conferma dell'allungamento delle forme delle aree di drenaggio nella direzione individuata dal centro della sezione iniziale del canale e dal baricentro dell'area, la si ha dal confronto diretto delle varianze calcolate per ogni singolo bacino lagunare, illustrato in Figura 5.11. Da questo grafico, si osserva che la distribuzione dei punti, al pari dell'andamento della retta interpolatrice indicano una predominanza dei valori di σ_n rispetto a quelli di σ_s , a conferma della tendenza delle aree di drenaggio ad allungarsi l'ungo l'asse s .

Per definire più precisamente la relazione che intercorre tra le deviazioni standard σ_s e σ_n , si è studiata la distribuzione statistica che assumono i valori di R_σ calcolati per tutte le aree di drenaggio di Figura 5.6.

Il risultato ottenuto è illustrato in Figura 5.12 e conferma ulteriormente la tendenza dei bacini ad avere una forma allungata in direzione s . Il grafico di Figura 5.12 indica che una distribuzione di probabilità di tipo *log-normale* è in grado di descrivere con una certa approssimazione l'andamento della distribuzione di R_σ fintanto che i valori del rapporto delle varianze restano inferiori a 3.25.

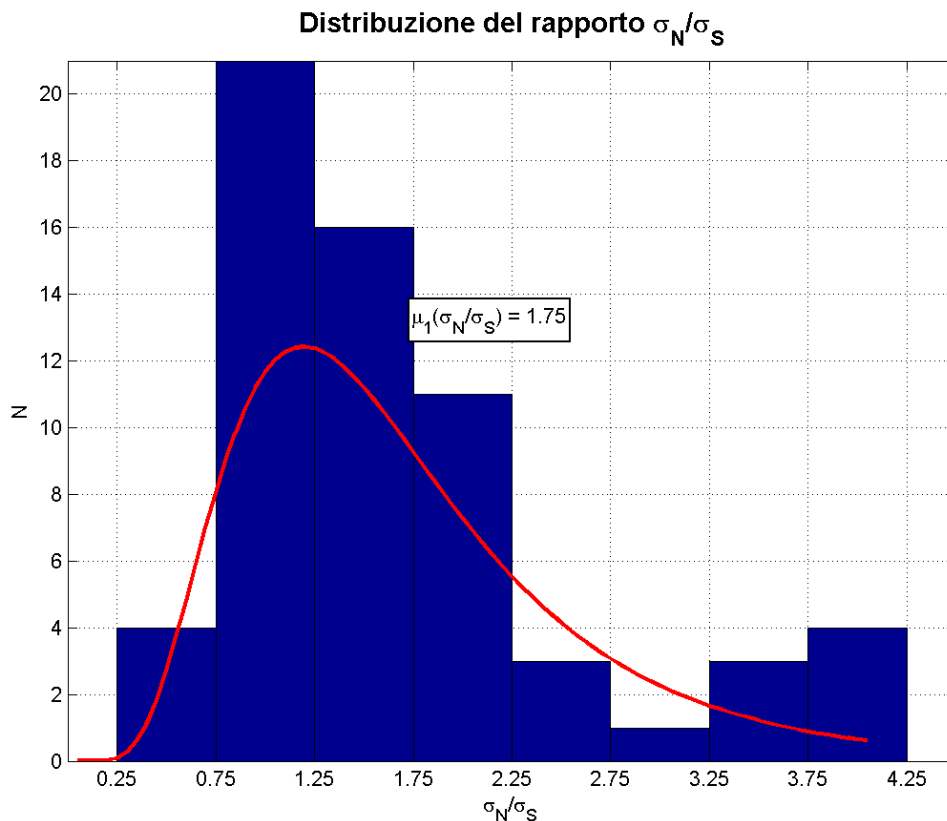


Figura 5.12: Distribuzione statistica assunta dai valori di R_σ , ottenuti per tutti i bacini lagunari di Figura 5.6 e riportanti nelle Tabelle 5.1 e 5.2.

Guardando, infatti, ai valori più elevati di R_σ , nel grafico di Figura 5.12, si nota la presenza di alcuni bacini lagunari per i quali il rapporto in questione sembra indicare che le forme delle loro aree abbiano caratteristiche diverse, rispetto alla maggior parte di quelle presenti nel campione analizzato.

Se si verifica, nelle Tabelle 5.1 e 5.2, quali sono le aree ad avere un valore di R_σ più elevato di 3.25, si nota che queste aree sono quelle identificate con gli ID 26, 42, 67, 68, 69, 70 e 71. Cinque su sette di questi bacini, come illustrato in Figura 5.6, si trovano al confine nord-orientale della Palude Pagliaga, dove la barena viene confinata da un contorno impermeabile composto da un rialzo del suolo, coperto da piante e arbusti visibili dalle immagini aeree.

Le aree di drenaggio con $ID \geq 67$ rappresentano, quindi, un caso particolare dovuto alle caratteristiche della zona barenale che occupano, che giustificano il comportamento apparentemente in controtendenza rispetto a quello delle restanti forme analizzate.

Escludendo, quindi, le aree di drenaggio contraddistinte da un $ID \geq 67$ si ottiene il grafico riportato in Figura 5.13, nel quale si vede che la funzione di distribuzione *log-normale* è effettivamente in grado di ben approssimare la distribuzione assunta dai valori di R_σ .

Nonostante l'esclusione di un numero non trascurabile di valori molto elevati di R_σ , il valore medio del rapporto tra le varianze standardizzate si attesta ancora ampiamente al di sopra dell'unità e pari a 1.53, confermando definitivamente la

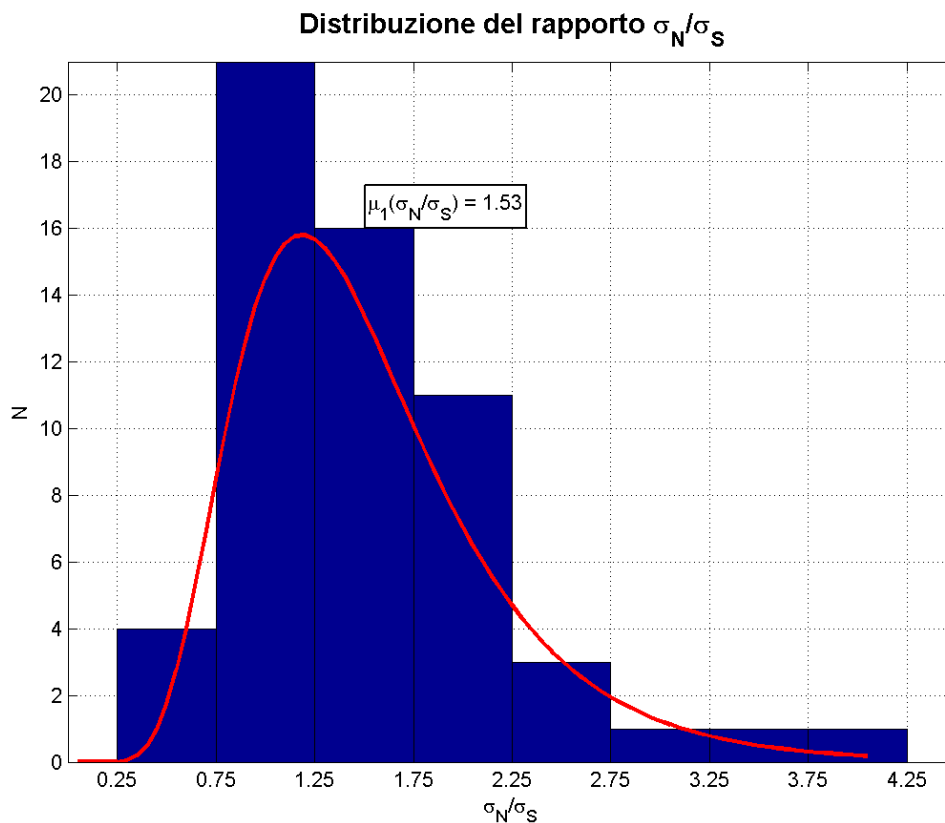


Figura 5.13: Distribuzione statistica assunta dai valori di R_σ , ottenuti per tutti i bacini lagunari, esclusi quelli contraddistinti da un $ID \geq 67$ per la particolarità dovute alla zona barenale che occupano (cfr. Figura 5.6).

tendenza delle aree di drenaggio ad assumere una forma allungata nella direzione definita dall'asse s .

Sulla base dei risultati ottenuti si propone quindi la seguente classificazione delle forme delle aree di drenaggio dei bacini a marea, basata sul valore assunto da R_σ .

- Classe 1:** $R_\sigma \leq 0.8 \Rightarrow$ forma del bacino *schacciata*;
- Classe 2:** $0.8 < R_\sigma \leq 1.3 \Rightarrow$ forma del bacino *quasi-circolare*;
- Classe 3:** $1.3 < R_\sigma \leq 2.0 \Rightarrow$ forma del bacino *allungata*;
- Classe 4:** $R_\sigma > 2.0 \Rightarrow$ forma del bacino *molto allungata*;

5.3 Confronto tra le classi di aree individuate

La classificazione proposta nel precedente paragrafo ha consentito di organizzare le aree di drenaggio ricavate all'interno della Palude Pagliaga e confrontare le forme che queste assumono, sulla base della classe di appartenenza e delle dimensioni.

Per effettuare i confronti, per ogni classe individuata, si è considerata una serie di bacini i cui perimetri sono stati sovrapposti in una medesima immagine, sovrapponendo l'origine del sistema di riferimento sull'asse della sezione terminale del canale.

Successivamente, per ognuno dei bacini rappresentati nella medesima immagine, è stata calcolata l'andamento della larghezza del bacino (B_t) lungo la direzione n , al variare della coordinata s . I profili con gli andamenti della larghezza sono stati quindi rappresentati in un secondo grafico, con la stessa scala di colori scelta per il confronto delle forme dei bacini.

Classe 1: $R_\sigma \leq 0.8$

Le aree di drenaggio che ricadono nella **Classe 1** sono quelle meno frequenti: nel campione di 63 bacini lagunari considerati solo 6 presentano un valore di $R_\sigma \leq 0.8$. Questi sono caratterizzati dall'aver una forma che si sviluppa, principalmente, lungo la direzione n .

In Figura 5.14a sono state sovrapposte le forme dei bacini lagunari identificati con gli ID pari a 19, 45, 52 e 56: il grafico ottenuto mostra come, pur avendo dimensioni molto diverse tra loro, le aree considerate (in particolare la 45 e la 56) presentino uno sviluppo planimetrico relativamente simile.

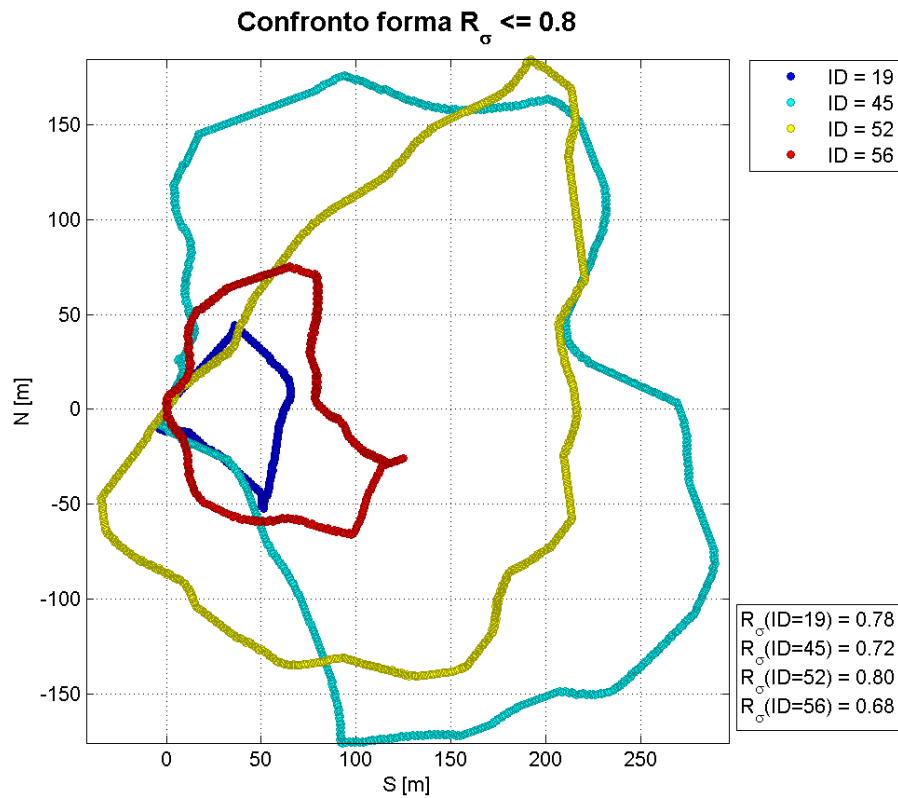
Se si considera, invece, l'andamento di B_t illustrato in Figura 5.14b per le medesime aree di drenaggio, si nota come tutti e quattro i bacini lagunari considerati si sviluppino a partire da una larghezza pressoché nulla. Questa, dopo una prima fase di allargamento repentino, tende a stabilizzarsi e a mantenersi su valori confrontabili. Infine, in prossimità della parte terminale del bacino lagunare, la larghezza subisce un crollo improvviso che riporta i valori di B_t in prossimità dello zero.

Analizzando quindi i due grafici riportati in Figura 5.14, è possibile affermare che le aree di drenaggio considerate, oltre ad avere valori confrontabili del parametro R_σ , si sviluppano in larghezza secondo uno schema comune. Questo fatto rafforza la validità della supposizione fondata sui valori dai parametri di forma calcolati, ovvero che le aree di drenaggio con un valore di R_σ minore o pari a 0.8 presentano caratteristiche di forma simili, indipendentemente della grandezza assoluta dell'area di drenaggio.

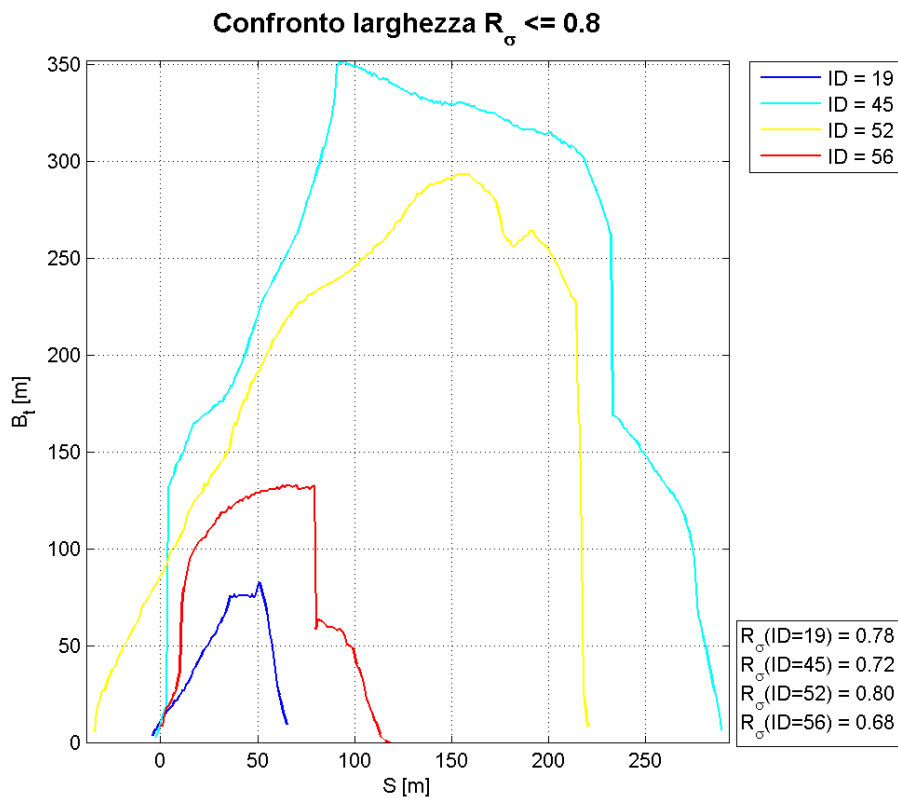
Classe 2: $0.8 < R_\sigma \leq 1.3$

Le aree di drenaggio che ricadono nella **Classe 2** sono molto frequenti e sono caratterizzate dall'aver una forma che non presenta una direzione prevalente di sviluppo. I bacini di drenaggio contraddistinti da un valore di R_σ compreso tra 0.8 e 1.3, sono 22 su 63. In questa categoria ricadono le aree di drenaggio che, solitamente, hanno la forma più regolare, alle volte molto vicina a quella di un cerchio.

In Figura 5.15a sono state sovrapposte le forme dei bacini lagunari identificati con gli ID pari a 11, 18, 22, 24, 25 e 54. Dal grafico risultante si può apprezzare come tutte le forme rappresentate presentino un allargamento e un restringimen-

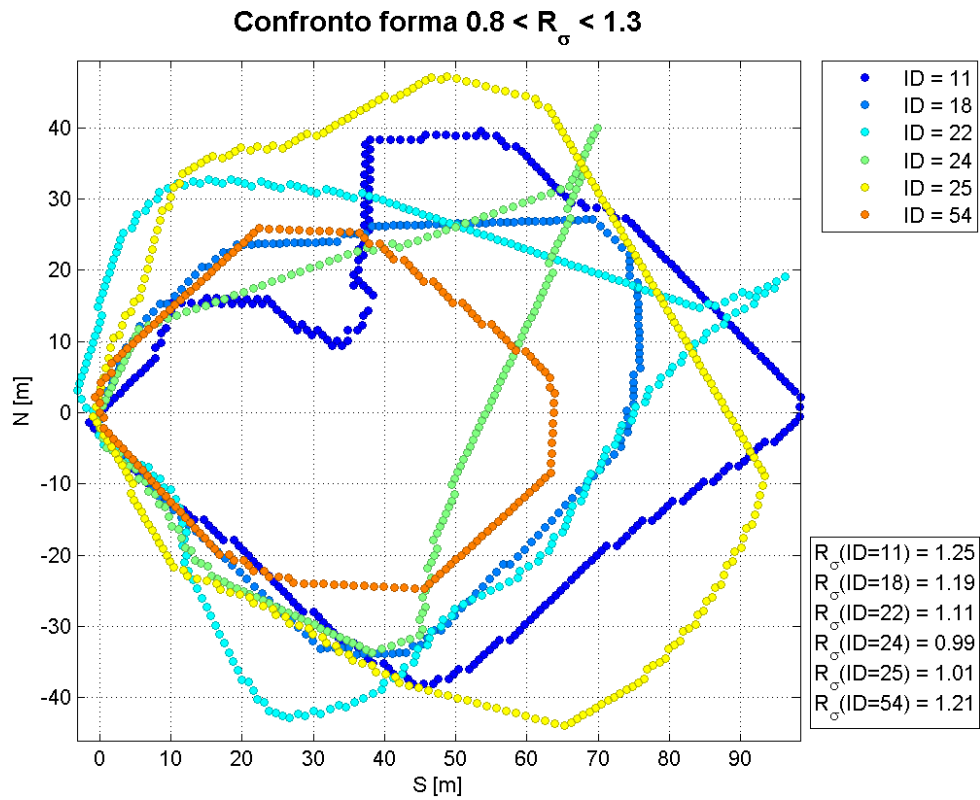


(a) Confronto della forma delle aree di drenaggio.

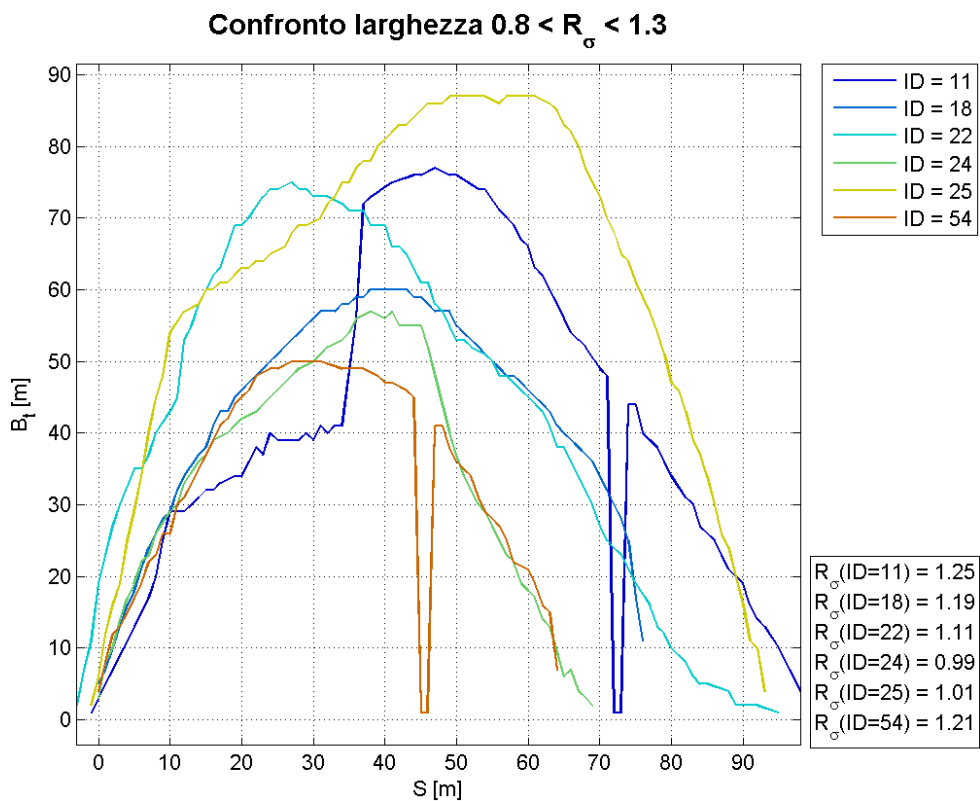


(b) Andamento della larghezza in funzione di s .

Figura 5.14: Confronto della forma delle aree di drenaggio e andamento della larghezza in funzione di s per i bacini appartenenti alla **Classe 1** (i.e. $R_{\sigma} \leq 0.8$).



(a) Confronto della forma delle aree di drenaggio.



(b) Andamento della larghezza in funzione di s .

Figura 5.15: Confronto della forma delle aree di drenaggio e andamento della larghezza in funzione di s per i bacini appartenenti alla **Classe 2** (i.e. $0.8 < R_{\sigma} \leq 1.3$).

to graduale, che, quasi sempre, si realizza in egual misura per valori positivi e negativi della coordinata n (e.g., per i bacini 18, 22, 25 e 54).

Una conferma della gradualità delle variazioni della larghezza la si ha osservando l'andamento di B_t riportato in Figura 5.15b. A differenza di quanto illustrato in Figura 5.14b, per valori di R_σ compresi tra 0.8 e 1.3 i profili che descrivono le variazioni della larghezza presentano un andamento molto più regolare, con variazioni graduali. Anche in questo caso, i bacini lagunari si sviluppano a partire da una larghezza pressoché nulla; allontanandosi dalla bocca, B_t aumenta gradualmente fino a raggiungere un punto di massimo, dopo il quale inizia, altrettanto gradualmente, a ridursi, tornando ad assumere valori prossimi allo zero nei punti più distanti dalla sezione di chiusura.

Una particolarità di questa tipologia di profili di B_t è, dunque, quella di non avere tratti in cui la larghezza tende a mantenersi costante. Questa caratteristica è stata riscontrata, in modo così chiaro, solamente per le aree di drenaggio appartenenti a questa classe ed è tanto più evidente quanto più R_σ è prossimo ad uno.

Nonostante la quasi totalità delle aree di drenaggio di questa categoria tenda ad avere una forma quasi-circolare, tra i 22 bacini individuati sono presenti tre eccezioni (cfr. Figure B.4d, B.5f e B.10c). Tali casi particolari, pur avendo un valore di R_σ prossimo a uno, sono caratterizzate da aree di forma estremamente irregolare, in cui, tuttavia, la particolare conformazione della rete canalizzata fa sì che il bacino abbia uno sviluppo confrontabile nelle direzioni (n, s).

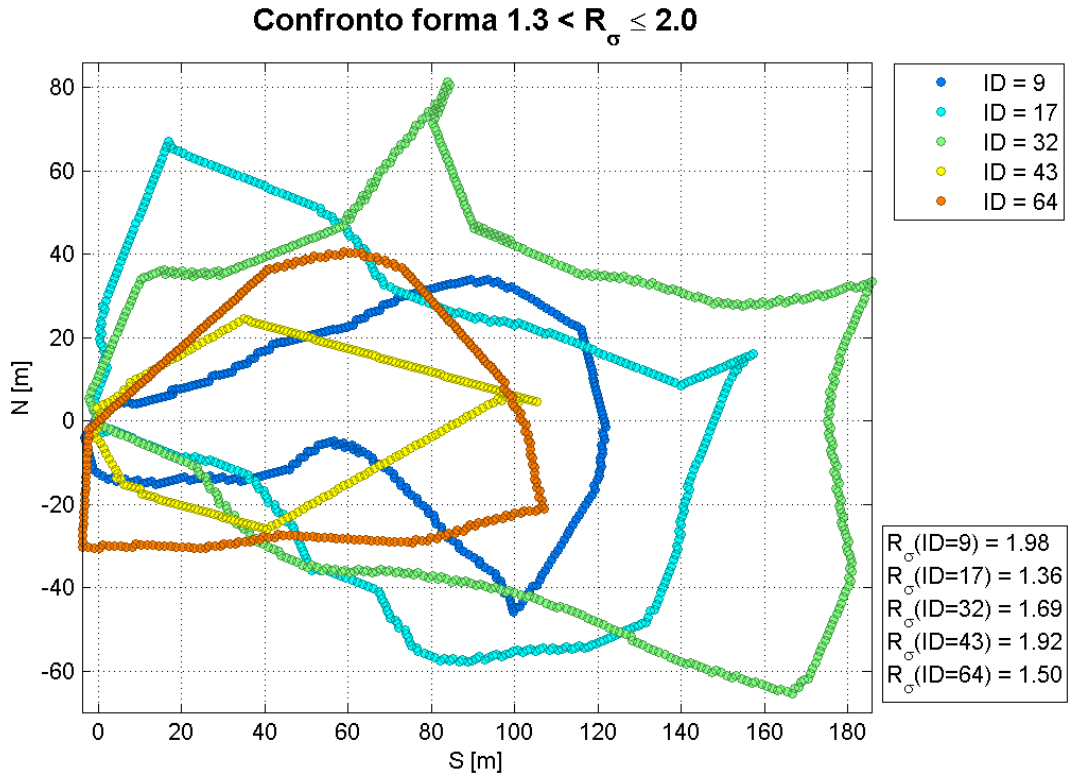
Dall'analisi delle forme delle aree di drenaggio caratterizzate da $0.8 < R_\sigma \leq 1.3$ si può affermare che i bacini lagunari che ricadono nella Classe 2 presentano, a meno di poche eccezioni, caratteristiche di forma simili (i.e., molto compatta e tendente quasi al circolare), così da poter essere accomunati gli uni agli altri.

Classe 3: $1.3 < R_\sigma \leq 2.0$

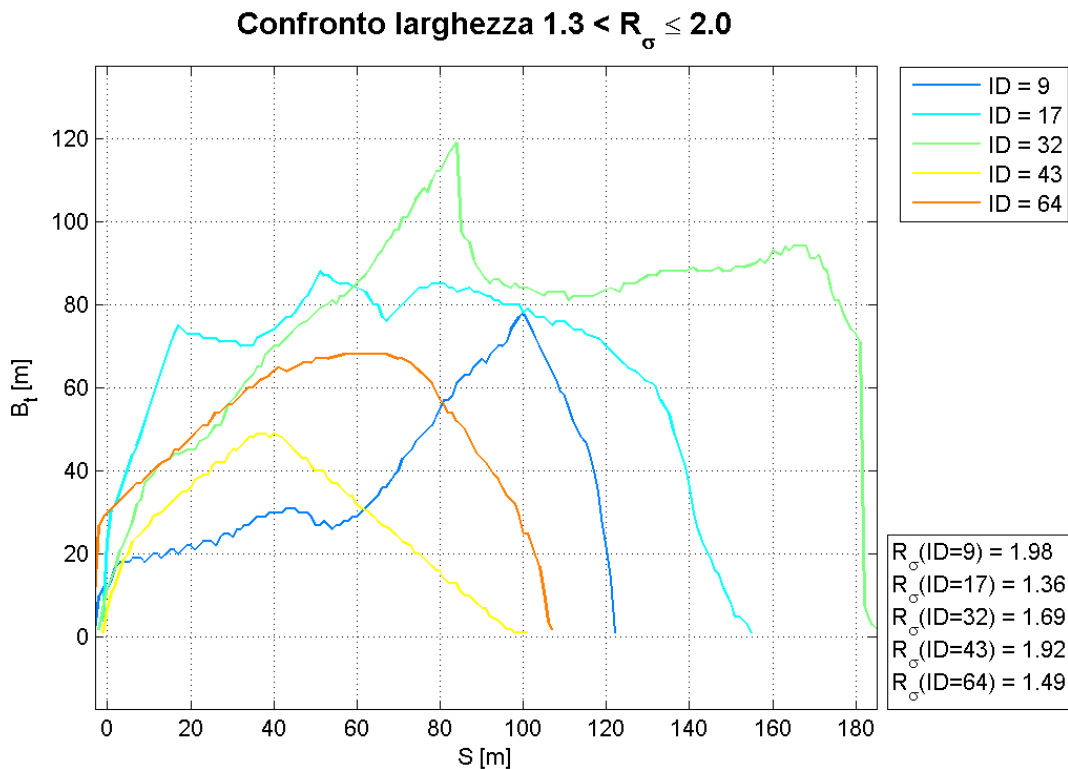
Tra i 63 considerati, sono stati individuati 20 bacini lagunari con un valore di R_σ compreso tra 1.3 e 2.0; questi sono caratterizzati dall'aver una forma che tende ad allungarsi lungo s . In particolare, l'estensione lungo s è sempre maggiore di quella lungo n , essendo questa mediamente pari alla metà della lunghezza del bacino.

In Figura 5.16a sono state sovrapposte le forme dei bacini lagunari identificati con gli ID pari a 9, 17, 32, 43 e 64: dal grafico risultante si può apprezzare come le forme rappresentate, al contrario di quanto illustrato in Figura 5.15a, abbiano una forma allungata lungo s . Questo comportamento può essere legato al fatto che i canali che drenano queste aree, rispetto ai canali drenanti bacini lagunari meno allungati, spesso hanno una struttura più semplice, ovvero sono contraddistinti da un canale principale ben individuabile, poco sinuoso e con poche ramificazioni di rilievo.

Se si analizza l'andamento di B_t delle aree di drenaggio 9, 17, 32, 43 e 64, illustrato in Figura 5.16b, risulta chiara la differenza con quanto visto in Figura 5.15b: se per le aree di Classe 2 B_t ha un andamento che cresce e decresce in modo continuo e regolare, per i bacini di Classe 3 si ha una crescita iniziale



(a) Confronto della forma delle aree di drenaggio.



(b) Andamento della larghezza in funzione di s .

Figura 5.16: Confronto della forma delle aree di drenaggio e andamento della larghezza in funzione di s per i bacini appartenenti alla **Classe 3** (i.e. $1.3 < R_{\sigma} \leq 2.0$).

molto rapida, seguita da tratti in cui le variazioni di larghezza avvengono in modo graduale, i quali si concludono con una veloce diminuzione di B_t verso la fine dell'area di drenaggio. Lungo i tratti in cui la larghezza è meno variabile, tanto più evidenti quanto più è grande il valore di R_σ , l'ordine di grandezza dei valori che assume B_t è circa la metà della lunghezza totale dei bacini, a meno di allargamenti localizzati in brevi tratti.

Tra le aree appartenenti alla Classe 3 un caso particolare è quello del bacino 43, il quale presenta un valore di R_σ pari a 1.92 (i.e. prossimo al valore limite 2) e, pur essendo effettivamente di forma allungata, presenta una forma particolarmente regolare rispetto a quelle tipiche di questa classe. A meno di indagini più approfondite, il caso rappresentato da questa particolare area di drenaggio può essere considerato come un'eccezione.

Classe 4: $R_\sigma > 2.0$

Alla Classe 4 appartengono le aree di drenaggio che presentano una forma molto allungata: nella Palude Pagliaga ci sono 15 bacini lagunari che appartengono a questa categoria, considerando anche le aree di drenaggio con $ID \geq 67$ che risultano molto allungate per la particolare posizione in cui si trovano.

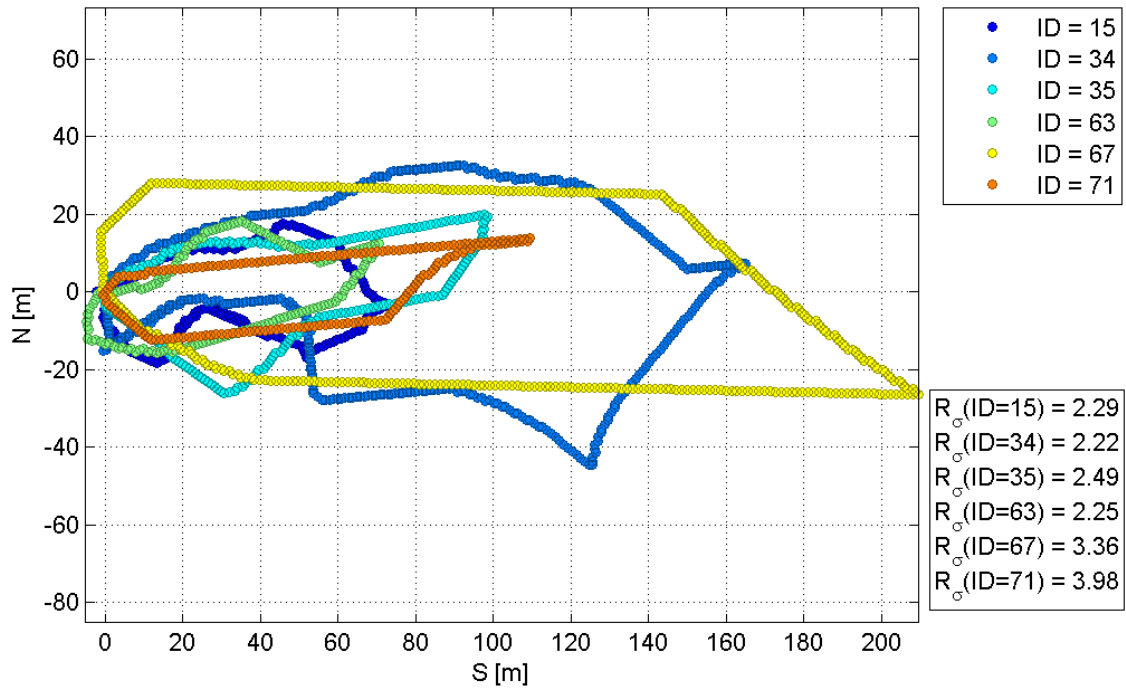
Se si confronta la Classe 4 con la Classe 3, le caratteristiche che contraddistinguono le aree di drenaggio appartenenti a queste classi risultano molto simili, ma nella Classe 4 ($R_\sigma > 2$) l'allungamento in relazione alla larghezza del bacino risulta molto più accentuato. Il limite posto pari a $R_\sigma = 2$ è stato scelto in modo arbitrario, prendendo come riferimento il fatto che la larghezza delle aree di drenaggio che superano questo limite sarà sempre inferiore alla metà della lunghezza. Si può quindi affermare che alla Classe 4 appartengono le aree di drenaggio in cui la forma è prettamente allungata; mentre la Classe 3 rappresenta un livello intermedio tra le Classi 2 e 4, in cui le forme tendono ad allungarsi senza esserlo in modo univoco.

In Figura 5.17a sono state sovrapposte le forme dei bacini lagunari identificati con gli ID pari a 15, 34, 35, 63, 67 e 71: dal grafico che ne risulta si osserva che, a eccezione di pochi tratti in cui si realizzano dei piccoli allargamenti, le aree di drenaggio hanno uno sviluppo prevalente lungo la direzione dell'asse s . Un'altra caratteristica che presentano queste aree di drenaggio è la gradualità con la quale si restringono, nella fase terminale del bacino.

Quest'ultima peculiare caratteristica si evidenzia chiaramente anche nel grafico illustrato in Figura 5.17b: dall'andamento di B_t si nota che, dopo un tratto di allargamento iniziale, relativamente rapido, la larghezza delle aree di drenaggio rimane pressoché costante per lunghi tratti, attorno a valori dell'ordine di grandezza di un quarto della lunghezza totale dei bacini, senza mai raggiungere valori prossimi alla metà della stessa. Infine, giunti nel tratto conclusivo, le aree di drenaggio tendono a restringersi fino a raggiungere valori di B_t molto piccoli o nulli, in modo lento e graduale.

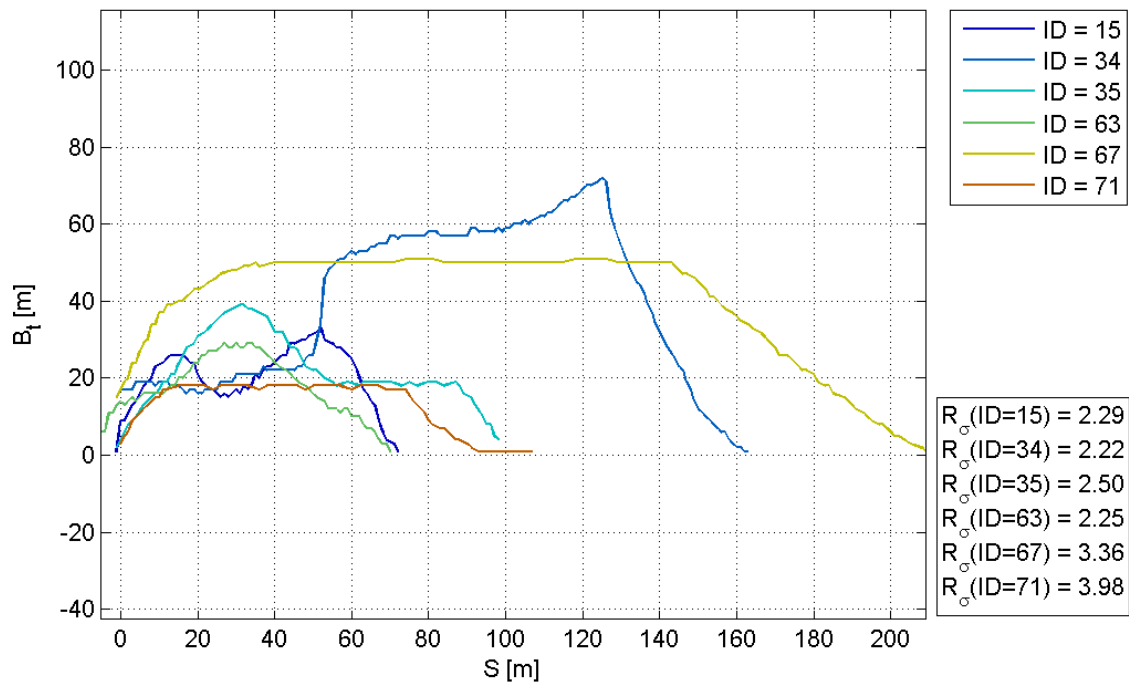
Dai confronti effettuati è possibile affermare che la classificazione proposta, fondata sul valore che assume R_σ , è in grado di fornire una prima suddivisione della forma delle aree di drenaggio, dividendole in quattro distinte categorie.

Confronto forma $R_{\sigma} > 2.0$



(a) Confronto della forma delle aree di drenaggio.

Confronto larghezza $R_{\sigma} > 2.0$



(b) Andamento della larghezza in funzione di s .

Figura 5.17: Confronto della forma delle aree di drenaggio e andamento della larghezza in funzione di s per i bacini appartenenti alla **Classe 4** (i.e. $R_{\sigma} > 2.0$).

Questa suddivisione risulta valida in generale, anche se, in alcuni casi specifici, le caratteristiche presentate da alcuni bacini lagunari esulano da quelle proprie della classe di appartenenza: chiaramente, per poter verificare definitivamente la validità della classificazione proposta, è necessario estendere a ulteriori barene le indagini qui illustrate ed effettuate con riferimento alla Palude Pagliaga.

Capitolo 6

Conclusioni

In questa tesi è stata studiata la forma delle aree di drenaggio afferenti ai canali a marea. In particolare, è stato realizzato un programma di calcolo, fondato sul modello matematico sviluppato da Rinaldo et al. [1999a]. Tale programma risolve, alle differenze finite, l'equazione di Poisson (2.23) risultante dalle semplificazione delle equazioni delle onde lunghe in acque basse, che fornisce la distribuzione spaziale della superficie dell'acqua su una barena. È stato così possibile studiare le caratteristiche delle aree drenate dai canali lagunari solcanti la Palude Pagliaga (posta al confine nord-occidentale della laguna di Venezia, cfr. par. 3.2).

Questa particolare porzione di laguna, studiata anche in passato Fagherazzi et al. [1999] e Marani et al. [2003], è stata recentemente oggetto di studio nell'ambito di una tesi di laurea [Rado 2014], svolta con l'obiettivo di studiarne l'evoluzione temporale nel corso degli ultimi sessanta anni. I dati raccolti da Rado [2014] hanno consentito di digitalizzare i contorni delle barene presenti nella Palude Pagliaga e costituire, quindi, il file contenente le condizioni al contorno necessarie alla soluzione numerica del modello matematico sopracitato.

Nel Capitolo 3 è stata studiata la procedura per generare, utilizzando il software commerciale ArcGis, il file contenente le condizioni al contorno. In particolare, la procedura proposta nel Paragrafo 3.3.3 ha consentito di trasformare i contorni delle barene, digitalizzati usando ArcMap, nel file contenente la matrice delle condizioni al contorno per la Palude Pagliaga. Secondo lo schema illustrato nel Paragrafo 3.3.1, la distribuzione planimetrica di canali, barene e bassifondi presenti nell'area di studio è stata discretizzata con una risoluzione spaziale di $1\text{ m} \times 1\text{ m}$. La matrice delle condizioni al contorno risultante è composta da 1690 righe e 1678 colonne, e rappresenta con sufficiente dettaglio le caratteristiche della rete di canali lagunari della Palude Pagliaga (cfr. par. 3.4).

Nel Capitolo 4 è stato esposto come il modello numerico per la soluzione dell'equazione di Poisson, originariamente sviluppato in Fortran [A. D'Alpaos 2001], è stato implementato in Matlab. Sfruttando le API, sono stati realizzati i MEX-Files [The MathWorks 2014b] che consentono di tradurre le subroutines Fortran in funzioni di Matlab, aggirando i problemi di rallentamento che si sarebbero avuti traducendo direttamente in Matlab l'originario codice Fortran in cui sono presenti lunghi cicli iterativi. I codici di calcolo per la trasformazione delle subroutines Fortran in MEX-Files sono interamente riportati in Appendice A.11;

il procedimento necessario per la loro compilazione è illustrato nel Paragrafo 4.2.3. Al fine di rendere più semplice l'utilizzo delle funzioni risultanti, sono state realizzate delle funzioni di Matlab per la gestione dei dati entranti e uscenti dai MEX-Files, rendendoli simili alle funzioni che vengono sviluppate da *The MathWorks*, la società proprietaria del software (cfr. par. 4.2.4). In questo modo è stata ottenuta una toolbox di funzioni Matlab per l'individuazione della forma delle aree di drenaggio dei canali a marea, note le matrici delle condizioni al contorno e quella che definisce le sezioni di chiusura dei canali di cui si vuole determinare la forma del bacino di drenaggio (cfr. par. 4.1.3). L'utilizzo dei MEX-Files, supportato dalle funzioni per la gestione dell'input e dell'output, ha permesso di realizzare un programma di calcolo la cui efficienza è paragonabile a quella del programma sviluppato in Fortran, ma molto più semplice da utilizzare. È stato raggiunto quindi l'obiettivo che aveva motivato la migrazione da un software all'altro.

Nel Capitolo 5 il programma di calcolo realizzato è stato utilizzato per analizzare la forma di 63 bacini a marea, estratti sulla base della soluzione dell'equazione di Poisson (cfr. Figura 5.7). Tali bacini sono stati rappresentati singolarmente rispetto a un nuovo sistema di riferimento cartesiano, di coordinate (n, s) , centrato nel baricentro dell'area di drenaggio. La direzione s è individuata dalla retta passante per il centro della sezione di sbocco del canale e il baricentro stesso dell'area (cfr. Appendice B); la direzione n è normale a s .

Tale sistema di riferimento permette di rappresentare in modo oggettivo e con la medesima orientazione tutte le aree di drenaggio. Sono stati quindi calcolati i seguenti parametri di forma: le deviazioni standard rispetto agli assi (n, s) ; il loro rapporto R_σ ; la *skewness* e la *kurtosis* rispetto al solo asse s (cfr. Tabelle 5.1 e 5.2).

Dalle analisi effettuate è stato riscontrato che, mediamente, la forma delle aree di drenaggio tende ad essere allungata nella direzione individuata dall'asse s . Tra i parametri calcolati, quello in grado di sintetizzare maggiormente la forma che assumono i bacini di drenaggio analizzati è risultato essere R_σ . Nella Sezione 5.2 è stata pertanto proposta una classificazione delle forme delle aree di drenaggio, fondata sul rapporto R_σ , definita da quattro classi:

- Classe 1:** $R_\sigma \leq 0.8 \Rightarrow$ forma del bacino *schacciata*;
- Classe 2:** $0.8 < R_\sigma \leq 1.3 \Rightarrow$ forma del bacino *quasi-circolare*;
- Classe 3:** $1.3 < R_\sigma \leq 2.0 \Rightarrow$ forma del bacino *allungata*;
- Classe 4:** $R_\sigma > 2.0 \Rightarrow$ forma del bacino *molto allungata*.

Nella Sezione 5.3 la classificazione proposta è stata verificata sovrapponendo in un medesimo grafico i perimetri delle aree di drenaggio appartenenti alla stessa classe, facendo coincidere l'origine del sistema di riferimento con il punto centrale della sezione di sbocco dei canali (Figure 5.14a, 5.15a, 5.16a e 5.17a). Inoltre, per le stesse aree di drenaggio, in un secondo grafico è stato riportato l'andamento lungo s delle larghezze B_t , valutate in direzione n (Figure 5.14b, 5.15b, 5.16b e 5.17b).

I dati analizzati indicano che la classificazione proposta è in grado di individuare, con buona approssimazione, quattro categorie distinte di bacini di drenaggio, fornendo quindi un metodo per organizzare in modo razionale le aree di drenaggio ai fini di uno studio più approfondito delle loro caratteristiche di forma.

Possibili sviluppi futuri

Di particolare interesse è l'individuazione dei parametri in grado di fornire una rappresentazione adimensionale della forma delle aree di drenaggio, così da ottenere una scala di confronto uniforme e indipendente dalle dimensioni molto variabili che assumono i bacini lagunari. Una rappresentazione di questo tipo fornirebbe la base per un'indagine più ampia rispetto a quella eseguita per la Palude Pagliaga, consentendo di approfondire ulteriormente lo studio delle aree di drenaggio afferente i canali lagunari. Tali caratteristiche sono di cruciale importanza per la determinazione del profilo longitudinale e della sezione di equilibrio dei canali a marea.

Sarebbe inoltre interessante indagare lo sviluppo nel tempo della forma delle aree di drenaggio in relazione allo sviluppo della rete dei canali. Questa particolare analisi, per la quale la Palude Pagliaga risulterebbe particolarmente adatta vista la lunga serie di foto aeree disponibile a partire dal 1954, consentirebbe di approfondire la conoscenza del legame che intercorre tra l'idrodinamica del sistema e lo sviluppo morfologico dei canali lagunari.

Appendice A

Listati dei codici implementati

In questa Appendice vengono riportati i listati dei codici di calcolo realizzati per individuare la forma delle aree di drenaggio dei canali a marea.

Il programma principale, scritto in linguaggio Matlab, viene riportato per primo; a questo seguono per esteso i codici per la traduzione delle *subroutines* in MEX-Files, organizzati in ordine di chiamata effettuata dal file principale.

Per la descrizione del contenuto dei programmi implementati si rimanda alla descrizione effettuata nei Capitoli [4](#) e [5](#).

In particolare, la trattazione di come le *subroutines*, originariamente sviluppate in Fortran, sono state modificate per essere trasformate in MEX-Files e poi compilate all'interno del software Matlab la si può consultare nella Sezione [4.2.3](#).


```

1: %=====
2: % ===== MODELLO PER LO STUDIO DELLE AREE DI DRENAGGIO DEI CANALI LAGUNARI =====
3: %=====
4: % Inizio della realizzazione del codice 18/03/2014
5: % autore: Carraro Francesco
6: % TOOLBOX necessaria: TELONE_TBOX (compatibile con WINDOWS 32-64 bit)
7: % se non è stata installata la toolbox è possibile richiamarla col comando
8: % 'userpath'
9: %=====
10: close all
11: clear all
12: clc
13: %=====
14: % seleziono la cartella contenente le funzioni
15: userp = 'C:\Users\Francesco\Dropbox\TESI\ModelloTelone_Toolbox';
16: userpath([userp '\TELONE_TBOX'])
17: %=====
18: % LETTURA DEI DATI CHE ESCONO DA ARCGIS PER L'ANALISI DELLA PALUDE PAGLIAGA
19: % ASSEGNAZIONE E LETTURA DEI DATI:
20: % Assegno il nome dell'area di studio
21: name = 'Pagliaga2010';
22: % Assegno la dimensione della maglia del reticolo di calcolo {m}:
23: maglia = 1;
24: % Assegno il valore del coefficiente di Chezy {m^0.5/s}:
25: CHI = 10;
26: % Assegno il valore della velocità massima sulla barena {m/s}:
27: U_max = 0.15;
28: % Assegno il elevazione media istantanea della marea:
29: ETA0 = 0.35;
30: % Assegno il valore della derivata dell'elevazione media istantanea rispetto al
31: % tempo, nell'istante considerato:
32: dETA0_dT = 7.5*10^(-5);
33: % Assegno il valore che permette di definire le sezioni dei canali. Se considero
34: % la direzione dell'asse, "passo_sez" pixels in quella direzione formano
35: % una sezione:
36: passo_sez = 1;
37: % Imposto il numero massimo d'iterazioni per il GCM:
38: imax=10000;
39: % Imposto la tolleranza per la soluzione con GCM
40: toll=10^(-3);
41: %-----
42: % Leggo la matrice delle condizioni al contorno:
43: % legenda:
44: %     0 => pixel di barena (INCOGNITE);
45: %     1 => pixel contorno a flusso nullo;
46: %     2 => pixel di specchio liquido lagunare (bassifondi);
47: %     3 => pixel canale;
48: %     4 => pixel inlet canale.
49: cont=load([userp '\pagliaga100cm.txt']);
50: %-----
51: % Leggo la matrice 'sezioni' contenenti le sezioni di chiusura dei canali
52: % rispetto alle quali determinare le aree di drenaggio dei canali
53: sez=load([userp '\sezioni_9-71.txt']);
54: %=====
55: % Per poter utilizzare le matrici uscenti da argis è necessario aggiungere alle
56: % matrici stesse una cornice di 1:
57: contorno = ones(length(cont(:,1))+2,length(cont(1,:))+2);
58: contorno(2:end-1,2:end-1)=cont;
59: clear cont
    
```

```
60: sezioni = ones(length(sez(:,1))+2,length(sez(1,:))+2);
61: sezioni(2:end-1,2:end-1)=sez;
62: clear sez
63: %-----
64: % Inverto l'ordine delle righe della matrice contorno in modo che le immagini
65: % risultino più coerenti alla realtà:
66: contorno=contorno(end:-1:1,:);
67: % Faccio la stessa operazione per la matrice sezioni:
68: sezioni= sezioni(end:-1:1,:);
69: %=====
70: % INIZIO L'ELABORAZIONE DEI DATI. DETERMINO:
71: % - n1 = n di righe; n2 = n di colonne della matrice contorno:
72:     [n1,n2]=size(contorno);
73: % - n di incognite del problema:
74:     dim1=length(contorno(contorno==0)); n_marshall = dim1;
75: % - numero di pixel canale alle bocche:
76:     n_inlet=length(contorno(contorno==4));
77: % - n di pixel canale:
78:     n_chann = length(contorno(contorno>=3));
79: % - n di pixel che identificano le sezioni di chiusura dei bacini lagunari:
80:     n_segnati = length(sezioni(sezioni>4));
81: %-----
82: % scrivo a video i dati che verranno utilizzati
83: fprintf('=====\n')
84: fprintf(' ===== PROVA TOOLBOX TELONE ===== \n')
85: fprintf('=====\n')
86: fprintf('I dati che verranno utilizzati in questa simulazione sono:\n')
87: fprintf(' - numero di righe_N1 = %d;\n',n1)
88: fprintf(' - numero di colonne_N2 = %d;\n',n2)
89: fprintf(' - num. di incognite nel modello del telone = %g;\n',dim1)
90: fprintf(' - dimensione del lato della maglia quadrata = %5.3f {m};\n', maglia)
91: fprintf(' - valore del coefficiente di Chezy = %5.3f {m^0.5/s};\n', CHI)
92: fprintf(' - valore della velocità massima sulla barena = %5.3f{m/s};\n',U_max)
93: fprintf(' - valore dell'elevazione media istantanea = %5.3f;\n', ETA0)
94: fprintf(' - derivata dell'elevazione media istantanea\n')
95: fprintf(' rispetto al tempo, nell'istante considerato = %f.\n', dETA0_dT)
96: fprintf('=====\n')
97: %=====
98: % ===== CHIAMATA DELLE FUNCTION DALLA TOOLBOX =====
99: %=====
100: fprintf('INIZIO DELL'ELABORAZIONE DEI DATI:\n')
101: fprintf('=====\n')
102: fprintf(' ----- function telone ----- \n')
103: %-----
104: [sol,iter]=telone(contorno,n_marshall,imax,toll,maglia,U_max,dETA0_dT,CHI,ETA0);
105: %-----
106: fprintf('=====\n')
107: fprintf(' ----- incidenza barene ----- \n')
108: %-----
109: [incidenza_bar,portata_bar,sorgenti_bar,L_bar]=incidenzabarene(contorno,sol);
110: %-----
111: fprintf('=====\n')
112: fprintf(' ----- function ampiezza ----- \n')
113: %-----
114: [width,width_sez]=ampiezza(contorno,n_chann,n_inlet,passo_sez);
115: %-----
116: fprintf('=====\n')
117: fprintf(' ----- incidenza canali ----- \n')
118: %-----
```

```

119: [incidenza_rete,portata_rete,sorgenti_rete]=incidenzacanali(contorno,width);
120: %-----
121: fprintf('=====\n')
122: fprintf(' ----- incidenza ----- \n')
123: %-----
124: fprintf('calcolo la matrice "incidenza" delle direzioni di drenaggio:\n')
125: fprintf('\t incidenza = incidenza_rete + incidenza_bar \n');
126: incidenza=incidenza_rete+incidenza_bar;
127: %-----
128: fprintf('=====\n')
129: fprintf(' ----- segna rami ----- \n')
130: [rami,bacini,contnew]= segnarami(contorno,incidenza_rete,incidenza,...
131:                               sezioni,n_segnati,n_chann);
132: fprintf('=====\n')
133: %-----
134: % ===== ANALISI DELLA FORMA DELLE AREE DI DRENAGGIO =====
135: %-----
136: %creo le matrici in cui analizzare i rami e i bacini
137: A_rami = zeros(size(rami));
138: A_rami(rami>4) = floor(rami(rami>4));
139:
140: A_bacini = zeros(size(bacini));
141: A_bacini(bacini>4) = floor(bacini(bacini>4));
142:
143: AA_ramibacini= A_bacini+A_rami;
144:
145: A_width = zeros(size(width));
146: A_width(rami>4) = width(rami>4);
147:
148: % determino il perimetro di tutti i bacini considerati
149: Stats = regionprops(AA_ramibacini,'Perimeter');
150: PK = cat(1,Stats.Perimeter);
151: % DEFINISCO IL SET DI CANALI SU CUI EFFETTUARE LO STUDIO PIU' APPROFONDITO
152: KK = find(PK>0);
153: %KK = 9:66;
154: %KK = [9:25 27 28 30:40 43:64 66 67];
155: % creo le variabili necessarie a memorizzare tutti i parametri delle aree
156: nK =length(KK); AK = zeros(nK,1); dK = zeros(nK,1); xCK = zeros(nK,1);
157: yCK = zeros(nK,1); Rc = zeros(nK,1); Ru = zeros(nK,1); LK = zeros(nK,1);
158: FK = zeros(nK,1); EK = zeros(nK,1); sigma2N = zeros(nK,1);
159: sigma2S = zeros(nK,1); mu3S = zeros(nK,1); mu4S = zeros(nK,1);
160: % apro la tabella in cui salvare le grandezze relative alla forma dei bacini
161: TABname = fopen([userp '\FormaBacini_ALL.txt'],'w');
162: % area & perimetro & sigma_s & sigma_n & sigma_n/sigma_s & skewness & kurtosis
163: FSpec = ['%2d \t & \t %7.0f \t & \t %5.1f \t & \t %5.2f \t & '...
164:         '\t %5.2f \t & \t %3.2f \t & \t %5.2e \t & \t %5.2e \t \\ \n'];
165: tic
166: for K = 1:nK
167:     % calcolo le coordinate di tutti i punti appartenenti al bacino
168:     [yK,xK]=find(A_bacini==KK(K));
169:     % calcolo le coordinate del rispettivo canale
170:     [ychann,xchann]=find(A_rami==KK(K));
171:     % calcolo l'area del bacino
172:     AK(K)=length(xK);
173:     % calcolo il diametro equivalente
174:     dK(K)= sqrt(4*AK(K)/pi);
175:     % calcolo le coordinate del baricentro
176:     xCK(K) = mean(xK);
177:     yCK(K) = mean(yK);

```

```

178: % calcolo le coordinate cartesiane del perimetro
179: XYP = [];
180: for xx = min(xK):max(xK)
181:     ymin = find(AA_ramibacini(:,xx)==KK(K),1,'first');
182:     ymax = find(AA_ramibacini(:,xx)==KK(K),1,'last');
183:     XYP = [XYP; ymin xx; ymax xx];
184:     ymin1 = find(AA_ramibacini(ymin:ymax,xx)~=KK(K),1,'first');
185:     ymax1 = find(AA_ramibacini(ymin:ymax,xx)~=KK(K),1,'last');
186:     if length(ymin1)>0, XYP = [XYP; ymin+ymin1-1 xx]; end
187:     if length(ymax1)>0, XYP = [XYP; ymin+ymax1+1 xx]; end
188: end
189: for yy = min(yK):max(yK)
190:     xmin = find(AA_ramibacini(yy,:)==KK(K),1,'first');
191:     xmax = find(AA_ramibacini(yy,:)==KK(K),1,'last');
192:     XYP = [XYP; yy xmin; yy xmax];
193:     xmin1 = find(AA_ramibacini(yy,xmin:xmax)~=KK(K),1,'first');
194:     xmax1 = find(AA_ramibacini(yy,xmin:xmax)~=KK(K),1,'last');
195:     if length(xmin1)>0, XYP = [XYP; yy xmin+xmin1-1]; end
196:     if length(xmax1)>0, XYP = [XYP; yy xmin+xmax1+1]; end
197: end
198: XYP = unique(XYP,'rows');
199: % trovo le coordinate cartesiane baricentriche
200: XC = XYP(:,2) - xCK(K);
201: YC = XYP(:,1) - yCK(K);
202: % trasformo le coordinate cartesiane baricentriche in coordinate polari
203: [theta,rho] = cart2pol(XC, YC);
204: % ordino le coordinate in base a theta
205: [theta, ind]=sort(theta);
206: XYP = XYP(ind,:); XC = XC(ind); YC = YC(ind); rho = rho(ind);
207: % memorizzo le coordinate cartesiane del perimetro nella struttura 2PP_CP
208: ID = ['nK' num2str(KK(K))];
209: P2P.CP.(ID) = [theta rho(ind)];
210: % memorizzo le coordinate cartesiane del perimetro nella struttura 2PP_CC
211: P2P.CC.(ID) = XYP;
212: % calcolo il fattore di circolarità
213: Rc(K) = 4*pi*AK(K)/PK(KK(K))^2;
214: % calcolo il rapporto di uniformità
215: Ru(K) = PK(KK(K))/(pi*dK(K));
216: % calcolo la lunghezza del canale K-esimo
217: LK(K) = max(A_width(A_rami==KK(K)))-min(A_width(A_rami==KK(K)));
218: % calcolo il fattore di forma
219: FK(K) = AK(K)/LK(K);
220: % calcolo il rapporto di allungamento
221: EK(K) = 2*sqrt(AK(K))/(LK(K)*sqrt(pi));
222: %=====
223: % ANALISI DELLA FORMA DELLE AREE DI DRENAGGIO 2.0
224: % Determino la direzione principale dell'area di drenaggio coincidente con la
225: % retta passante per per il centro dell'inlet e il baricentro dell'area:
226: % - trovo le coordinate baricentriche dell'inlet
227: [yy_in,xx_in] = find(sezioni == KK(K));
228: y_in = yy_in(round(length(yy_in)/2));
229: x_in = xx_in(round(length(yy_in)/2));
230: x_in = x_in - xCK(K); y_in = y_in - yCK(K);
231: % - trovo l'angolo che forma la retta passante per G e l'inlet con l'asse
232: % baricentrico verticale
233: theta1 = atan2(y_in,x_in);
234: theta2 = -0.5*pi-theta1;
235: [t_in,r_in]=cart2pol(x_in,y_in);
236: t_in = t_in+theta2;

```

```
237: [N_in,S_in]=pol2cart(t_in,r_in);
238: % - ruoto le coordinate polari baricentriche del perimetro dell'angolo theta2 e
239: % le ritrasformo in coordinate cartesiane per ottenere le coordinate S ed N
240: % rispettivamente lungo la direzione principale dell'area di drenaggio
241: % e lungo la direzione normale a questa:
242: theta = theta+theta2;
243: [N,S] = pol2cart(theta,rho);
244: P2P.NS.(ID) = [N S];
245: % - eseguo la stessa trasformazione per le coordinate del canale e per quelle
246: % dell'area che si vuole analizzare
247: [theta_chann,rho_chann] = cart2pol(xchann-xCK(K),ychann-yCK(K));
248: [theta_bar,rho_bar] = cart2pol(xK-xCK(K),yK-yCK(K));
249: theta_chann = theta_chann + theta2;
250: theta_bar = theta_bar + theta2;
251: [N_chann,S_chann] = pol2cart(theta_chann,rho_chann);
252: [NK,SK] = pol2cart(theta_bar,rho_bar);
253: % - calcolo la varianza in direzione N ed S:
254: sigma2N(K) = mean(SK.^2);
255: sigma2S(K) = mean(NK.^2);
256: % - calcolo i momenti di terzo e quarto ordine rispetto all'asse principale
257: mu3S(K) = mean(NK.^3);
258: mu4S(K) = mean(NK.^4);
259: %=====
260: % ===== CALCOLO LA DISTRIBUZIONE DELLE LARGHEZZE LUNGO S =====
261: % traslo il perimetro in modo da avere nell'origine l'inlet del canale
262: S = S-S_in;
263: P2P.NSI.(ID) = [N S];
264: % ordino in modo crescente le coordinate del perimetro
265: [S,ind] = sort(S);
266: N = N(ind);
267: % arrotondo all'intero piu' vicino le coordinate dei punti in modo che siano
268: % piu' facilmente confrontabili computazionalmente
269: N = round(N); S = round(S);
270: % inizializzo la variabile per la memorizzazione della larghezza
271: BBt = [];
272: % calcolo la larghezza in funzione di S
273: for ss=min(S):max(S)
274:     N_ss = N(S==ss);
275:     if length(N_ss) > 1
276:         BB = max(N_ss) - min(N_ss);
277:         BBt = [BBt; ss BB];
278:     end
279: end
280: Bt.(ID) = BBt(1,:);
281: for ii=2:length(BBt(:,1))-1
282:     if BBt(ii,2) > 0.9*BBt(ii-1,2) | BBt(ii,2) > 0.9*BBt(ii+1,2)
283:         Bt.(ID) = [Bt.(ID); BBt(ii,:)];
284:     end
285: end
286: Bt.(ID) = [Bt.(ID); BBt(end,:)];
287: %-----
288: % SALVO LA FIGURA RAPPRESENTANTE IL PERIMETRO DEI SINGOLI CANALI ANALIZZATI
289: figura = figure('Units','centimeters','OuterPosition',[3 1 18 19]);
290: grid on
291: box on
292: hold on
293: xmin=xmin-xCK(K); xmax=xmax-xCK(K); ymin=ymin-yCK(K); ymax=ymax-yCK(K);
294: titolo = ['Forma dell'area di drenaggio con ID = ' num2str(KK(K))];
295: title(titolo,'FontWeight','bold',...
```

```

296:     'FontSize',14,...
297:     'FontName','Arial');
298: xlabel('N [m]');ylabel('S [m]');
299: xlim(round([min(P2P.NS.(ID)(:,1))*1.2 ...
300:           max(P2P.NS.(ID)(:,1))-0.2*min(P2P.NS.(ID)(:,1)]));
301: ylim(round([min(P2P.NS.(ID)(:,2))*1.2 ...
302:           max(P2P.NS.(ID)(:,2))-0.2*min(P2P.NS.(ID)(:,2)]));
303: plot(P2P.NS.(ID)(:,1),P2P.NS.(ID)(:,2),...
304:      'MarkerFaceColor',[0 1 0],'MarkerEdgeColor',[0 0.5 0],...
305:      'MarkerSize',3,...
306:      'Marker','o',...
307:      'LineStyle','none',...
308:      'Color',[0 0 1]);
309: plot(N_chann,S_chann,...
310:      'MarkerFaceColor',[0 0 1],'MarkerEdgeColor',[0 0 1],...
311:      'MarkerSize',3,...
312:      'Marker','o',...
313:      'LineStyle','none',...
314:      'Color',[0 0 1]);
315: plot(0,0,'MarkerFaceColor',[1 0 0],'MarkerEdgeColor',[0 0 1],...
316:      'MarkerSize',5,...
317:      'Marker','o',...
318:      'LineStyle','none',...
319:      'Color',[0 0 1]);
320: axis equal
321: text(0,0,'G','VerticalAlignment','top','FontWeight','bold',...
322:      'FontSize',14,...
323:      'FontName','Arial');
324: str(1)={'L_{c}' = num2str(LK(K),'%5.1f') ' m'}};
325: str(2)={'A = ' num2str(AK(K),'%6.0f') ' m^2'}};
326: str(3)={'\sigma_N = ' num2str(sqrt(sigma2N(K)),'%5.2f') ' m'}};
327: str(4)={'\sigma_S = ' num2str(sqrt(sigma2S(K)),'%5.2f') ' m'}};
328: str(5)={'\mu_{3S} = ' num2str(mu3S(K),'%5.2f') ' m^3'}};
329: str(6)={'\mu_{4S} = ' num2str(mu4S(K),'%5.1g') ' m^4'}};
330: yl=get(gca,'ylim');xl=get(gca,'xlim');
331: ylim([yl(1)*1.2 yl(2)]);
332: xl=get(gca,'xlim');
333: text(xl(2)*0.99,yl(1)*1.19,str,...
334:      'VerticalAlignment','bottom',...
335:      'HorizontalAlignment','right',...
336:      'BackgroundColor',[1 1 1],...
337:      'FontSize',11,...
338:      'EdgeColor',[0 0 0],...
339:      'LineWidth',1,...
340:      'LineStyle','- ',...
341:      'FontName','Arial');
342: rapp={'\sigma_N/\sigma_S = ' num2str(sigma2N(K)^.5/sigma2S(K)^.5,'%3.2f')});
343: text(xl(1)*0.97,yl(2)*0.97,rapp,...
344:      'VerticalAlignment','top',...
345:      'HorizontalAlignment','left',...
346:      'BackgroundColor',[1 1 1],...
347:      'FontSize',11,...
348:      'EdgeColor',[0 0 0],...
349:      'LineWidth',1,...
350:      'LineStyle','- ',...
351:      'FontName','Arial');
352: % salvo l'immagine
353: set(gcf, 'PaperUnits', 'centimeters');
354: set(gcf, 'PaperSize', [18 19]);

```

```

355: set(gcf, 'PaperPositionMode','manual');
356: fig_png = [userp '\Immagini\NS_' name '_AreaID_' num2str(KK(K)) '.png'];
357: print('-dpng',figura,fig_png);
358: close figure 1; clear figura str
359: % salvo le grandezze gel k-esimo bacino di drenaggio nell'apposita tabella
360: fprintf(TABname,FSpec, KK(K),AK(K),PK(KK(K)),sqrt(sigma2S(K)),sqrt(sigma2N(K)),
...
361: sqrt(sigma2N(K))/sqrt(sigma2S(K)),mu3S(K),mu4S(K));
362: end
363: toc
364: fclose(TABname);
365: %=====
366: % ===== ELABORAZIONE GRAFICA DEI RISULTATI OTTENUTI =====
367: %=====
368: % Creo i vettori per definire gli assi cartesiani in metri:
369: xx = (1:n2).*maglia; yy=(1:n1).*maglia;
370: % Leggo la scala dei colori da utilizzare nei grafici
371: load([userp '\rainbow2.mat']);
372: %-----
373: % FIGURA 1: MATRICE DELLE CONDIZIONI AL CONTORNO
374: cont=contorno;
375: cont(contorno==4)=5;
376: figural = figure('Units','centimeters','OuterPosition',[3 1 18 19]);
377: % Create axes
378: axes1 = axes('Parent',figural,'XLim',[1 n2*maglia],'YLim',[1 n1*maglia],...
379: 'box','on','LineWidth',1,'FontName','Arial','DataAspectRatio',[1 1 1]);
380: xlabel('[m]');ylabel('[m]');
381: titolo = [name ': condizioni al contorno'];
382: title(titolo,'FontWeight','bold',...
383: 'FontSize',14,...
384: 'FontName','Arial');
385: hold(axes1,'all');
386: % Create surface
387: surface('Parent',axes1,'ZData',cont,'YData',yy,'XData',xx,...
388: 'LineStyle','none',...
389: 'EdgeColor','none','FaceLighting','phong',...
390: 'CData',cont);
391: colormap([0.5 0.2 0.2; 0.7 0.85 0.7;0.7 0.7 0.9; 0 1 1; 0 0 0.8]);
392: colorbar('peer',axes1,'YTickLabel',{'0','1','2','3','4'},...
393: 'YTick',[0.5 1.5 2.5 3.5 4.5],...
394: 'YLim',[0 5]);
395: % salvo l'immagine
396: set(gcf, 'PaperUnits', 'centimeters');
397: set(gcf, 'PaperSize', [18 19]);
398: set(gcf, 'PaperPositionMode','manual');
399: fig_png = [userp '\Immagini\1' name '_' num2str(maglia) 'm.png'];
400: print('-dpng',figural,fig_png);
401: close figure 1; clear figural cont;
402: %-----
403: % FIGURA 2: SOLUZIONE DELL'EQUAZIONE DI POISSON
404: soluzione = sol;
405: soluzione(contorno == 1)=-0.0003;
406: soluzione(contorno == 2)=-0.0002;
407: soluzione(contorno >= 3)=-0.0001;
408: figura2 = figure('Units','centimeters','OuterPosition',[3 1 18 19]);
409: % Create axes
410: axes1 = axes('Parent',figura2,'XLim',[1 n2*maglia],'YLim',[1 n1*maglia],...
411: 'box','on','LineWidth',1,'FontName','Arial','DataAspectRatio',[1 1 1]);
412: xlabel('[m]');ylabel('[m]');

```

```
413: titolo = [name ': soluzione dell''equazione di Poisson'];
414: title(titolo,'FontWeight','bold',...
415:       'FontSize',14,...
416:       'FontName','Arial');
417: hold(axes1,'all');
418: % Create surface
419: surface('Parent',axes1,'ZData',soluzione,'YData',yy,'XData',xx,...
420:        'LineStyle','none',...
421:        'EdgeColor','none','FaceLighting','phong',...
422:        'CData',soluzione);
423: % Create colorbar
424: colormap([0.7 0.85 0.7;0.6 0.6 1; 0 0 0; map]);
425: colorbar('peer',axes1);
426: % salvo l'immagine
427: set(gcf, 'PaperUnits', 'centimeters');
428: set(gcf, 'PaperSize', [18 19]);
429: set(gcf, 'PaperPositionMode','manual');
430: fig_png = [userp '\Immagini\2' name '_poisson_' num2str(maglia) 'm.png'];
431: print('-dpng',figura2,fig_png);
432: close figure 1; clear figura2 soluzione
433: %-----
434: % FIGURA 3: MATRICE DELLE DIREZIONI DI DRENAGGIO SULLA BARENA
435: incidenzabar = incidenza_bar;
436: incidenzabar(contorno == 1) = -1;
437: figura3 = figure('Units','centimeters','OuterPosition',[3 1 18 19]);
438: % Create axes
439: axes1 = axes('Parent',figura3,'XLim',[1 n2*maglia],'YLim',[1 n1*maglia],...
440:             'box','on','LineWidth',1,'FontName','Arial','DataAspectRatio',[1 1 1]);
441: xlabel('[m]');ylabel('[m]');
442: titolo = [name ': direzioni di drenaggio sulle barene'];
443: title(titolo,'FontWeight','bold',...
444:       'FontSize',14,...
445:       'FontName','Arial');
446: hold(axes1,'all');
447: % Create surface
448: surface('Parent',axes1,'ZData',incidenzabar,'YData',yy,'XData',xx,...
449:        'LineStyle','none',...
450:        'EdgeColor','none','FaceLighting','phong',...
451:        'CData',incidenzabar);
452: % Create colorbar
453: colormap([0.7 0.85 0.7;...
454:          0 0 0.5;...
455:          0 1 1;...
456:          0 1 0;...
457:          0.5 1 0;...
458:          1 1 0;...
459:          1 0.5 0;...
460:          1 0 0;...
461:          1 0.5 1;...
462:          0.5 0 1]);
463: colorbar('peer',axes1);
464: % salvo l'immagine
465: set(gcf, 'PaperUnits', 'centimeters');
466: set(gcf, 'PaperSize', [18 19]);
467: set(gcf, 'PaperPositionMode','manual');
468: fig_png = [userp '\Immagini\3' name '_incibarena_' num2str(maglia) 'm.png'];
469: print('-dpng',figura3,fig_png);
470: close figure 1; clear figura3 incidenzabar
471: %-----
```



```
472: % FIGURA 4: MATRICE DELLA FUNZIONE DI AMPIEZZA
473: width_stamp=width;
474: width_stamp(contorno==1)=-20/maglia;
475: width_stamp(contorno==0)=-10/maglia;
476: figura4 = figure('Units','centimeters','OuterPosition',[3 1 18 19]);
477: % Create axes
478: axes1 = axes('Parent',figura4,'XLim',[1 n2*maglia],'YLim',[1 n1*maglia],...
479: 'box','on','LineWidth',1,'FontName','Arial','DataAspectRatio',[1 1 1]);
480: xlabel('[m]');ylabel('[m]');
481: titolo = [name ': Funzione di Ampiezza'];
482: title(titolo,'FontWeight','bold',...
483: 'FontSize',14,...
484: 'FontName','Arial');
485: hold(axes1,'all');
486: % Create surface
487: surface('Parent',axes1,'ZData',width,'YData',yy,'XData',xx,...
488: 'LineStyle','none',...
489: 'EdgeColor','none','FaceLighting','phong',...
490: 'CData',width_stamp);
491: % Create colorbar
492: colormap([0.7 0.85 0.7; 0.5 0.2 0.2; map]);
493: colorbar('peer',axes1);
494: % salvo l'immagine
495: set(gcf, 'PaperUnits', 'centimeters');
496: set(gcf, 'PaperSize', [18 19]);
497: set(gcf, 'PaperPositionMode','manual');
498: fig_png = [userp '\Immagini\4' name '_sezioni_' num2str(maglia) 'm.png'];
499: print('-dpng',figura4,fig_png);
500: close figure 1; clear figura4 width_stamp;
501: %-----
502: % FIGURA 5: MATRICE DELLE DIREZIONI DI DRENAGGIO LUNGO I CANALI
503: incidenzarete=incidenza_rete;
504: incidenzarete(contorno==2)=-1;
505: incidenzarete(contorno==1)=-2;
506: incidenzarete(contorno==0)=0;
507: figura5 = figure('Units','centimeters','OuterPosition',[3 1 18 19]);
508: % Create axes
509: axes1 = axes('Parent',figura5,'XLim',[1 n2*maglia],'YLim',[1 n1*maglia],...
510: 'box','on','LineWidth',1,'FontName','Arial','DataAspectRatio',[1 1 1]);
511: xlabel('[m]');ylabel('[m]');
512: titolo = [name ': direzioni di drenaggio lungo i canali'];
513: title(titolo,'FontWeight','bold',...
514: 'FontSize',14,...
515: 'FontName','Arial');
516: hold(axes1,'all');
517: % Create surface
518: surface('Parent',axes1,'ZData',incidenzarete,'YData',yy,'XData',xx,...
519: 'LineStyle','none',...
520: 'EdgeColor','none','FaceLighting','phong',...
521: 'CData',incidenzarete);
522: % Create colorbar
523: colormap([0.7 0.85 0.7;...
524: 0 0 0.5;...
525: 0.5 0.2 0.2;...
526: 0 1 1;...
527: 0 1 0;...
528: 1 1 0;...
529: 1 0.5 0;...
530: 1 0 0;...]
```

```
531:         1 0.5 1;...
532:         0.5 0 1;...
533:         0 0.1 0.5]);
534:     colorbar('peer',axes1);
535:     % salvo l'immagine
536: set(gcf, 'PaperUnits', 'centimeters');
537: set(gcf, 'PaperSize', [18 19]);
538: set(gcf, 'PaperPositionMode','manual');
539: fig_png = [userp '\Immagini\5' name '_incicanali_' num2str(maglia) 'm.png'];
540: print('-dpng',figura5,fig_png);
541: close figure 1; clear figura5 incidenzarete
542: %-----
543: % FIGURA 6: MATRICE DELLE SORGENTI SULLA BARENA
544: sorgenti=sorgenti_bar;
545: sorgenti(contorno>=2)=3;
546: sorgenti(contorno==1)=2;
547: figura6 = figure('Units','centimeters','OuterPosition',[3 1 18 19]);
548: % Create axes
549: axes1 = axes('Parent',figura6,'XLim',[1 n2*maglia],'YLim',[1 n1*maglia],...
550:     'box','on','LineWidth',1,'FontName','Arial','DataAspectRatio',[1 1 1]);
551: xlabel('[m]');ylabel('[m]');
552: titolo = [name ': Sorgenti'];
553: title(titolo,'FontWeight','bold',...
554:     'FontSize',14,...
555:     'FontName','Arial');
556: hold(axes1,'all');
557: % Create surface
558: surface('Parent',axes1,'ZData',sorgenti,'YData',yy,'XData',xx,...
559:     'LineStyle','none',...
560:     'EdgeColor','none','FaceLighting','phong',...
561:     'CData',sorgenti);
562: % Create colorbar
563: colormap([0.8 0 0; 1 1 1; 0.7 0.85 0.7; 0 0 1]);
564: % salvo l'immagine
565: set(gcf, 'PaperUnits', 'centimeters');
566: set(gcf, 'PaperSize', [18 19]);
567: set(gcf, 'PaperPositionMode','manual');
568: fig_png = [userp '\Immagini\6' name '_sorgentiBAR_' num2str(maglia) 'm.png'];
569: print('-dpng',figura6,fig_png);
570: close figure 1; clear figura6 sorgenti;
571: %-----
572: % FIGURA 7: MATRICE DELLE DIREZIONI DI DRENAGGIO
573: incidenza_fig = incidenza;
574: incidenza_fig(contorno == 1) = -1;
575: figura7 = figure('Units','centimeters','OuterPosition',[3 1 18 19]);
576: % Create axes
577: axes1 = axes('Parent',figura7,'XLim',[1 n2*maglia],'YLim',[1 n1*maglia],...
578:     'box','on','LineWidth',1,'FontName','Arial','DataAspectRatio',[1 1 1]);
579: xlabel('[m]');ylabel('[m]');
580: titolo = [name ': Direzioni di drenaggio'];
581: title(titolo,'FontWeight','bold',...
582:     'FontSize',14,...
583:     'FontName','Arial');
584: hold(axes1,'all');
585: % Create surface
586: surface('Parent',axes1,'ZData',incidenza_fig,'YData',yy,'XData',xx,...
587:     'LineStyle','none',...
588:     'EdgeColor','none','FaceLighting','phong',...
589:     'CData',incidenza_fig);
```

```

590: % Create colorbar
591: colormap([0.7 0.85 0.7;...
592:         0 0 0.5;...
593:         0 1 1;...
594:         0 1 0;...
595:         0.5 1 0;...
596:         1 1 0;...
597:         1 0.5 0;...
598:         1 0 0;...
599:         1 0.5 1;...
600:         0.5 0 1]);
601: colorbar('peer',axes1);
602: % salvo l'immagine
603: set(gcf, 'PaperUnits', 'centimeters');
604: set(gcf, 'PaperSize', [18 19]);
605: set(gcf, 'PaperPositionMode','manual');
606: fig_png = [userp '\Immagini\7' name '_' num2str(maglia) 'm_inci.png'];
607: print('-dpng',figura7,fig_png);
608: close figure 1; clear figura7 incidenza_fig;
609: %-----
610: % FIGURA 8: MATRICE DELLE SEZIONI DI CHIUSURA DEI BACINI LAGUNARI
611: figura8 = figure('Units','centimeters','OuterPosition',[3 1 18 19]);
612: % Create axes
613: axes1 = axes('Parent',figura8,'XLim',[1 n2*maglia],'YLim',[1 n1*maglia],...
614:             'box','on','LineWidth',1,'FontName','Arial','DataAspectRatio',[1 1 1]);
615: xlabel('[m]');ylabel('[m]');
616: titolo = [name ': ' num2str(maglia) ' [m] sezioni1'];
617: title(titolo,'FontWeight','bold',...
618:       'FontSize',14,...
619:       'FontName','Arial');
620: hold(axes1,'all');
621: % Create surface
622: surface('Parent',axes1,'ZData',sezioni,'YData',yy,'XData',xx,...
623:        'LineStyle','none',...
624:        'EdgeColor','none','FaceLighting','phong',...
625:        'CData',sezioni);
626: colormap(map);
627: colorbar('peer',axes1);
628: % salvo l'immagine
629: set(gcf, 'PaperUnits', 'centimeters');
630: set(gcf, 'PaperSize', [18 19]);
631: set(gcf, 'PaperPositionMode','manual');
632: fig_png = [userp '\Immagini\8' name '_' num2str(maglia) 'm_sezion1.png'];
633: print('-dpng',figura8,fig_png);
634: close figure 1; clear figura8;
635: %-----
636: % FIGURA 9: MATRICE DELLE AREE DI DRENAGGIO
637: figura9 = figure('Units','centimeters','OuterPosition',[3 1 18 19]);
638: % Create axes
639: axes1 = axes('Parent',figura9,'XLim',[1 n2*maglia],'YLim',[1 n1*maglia],...
640:             'box','on','LineWidth',1,'FontName','Arial','DataAspectRatio',[1 1 1]);
641: xlabel('[m]');ylabel('[m]');
642: titolo = [name ': Aree di drenaggio'];
643: title(titolo,'FontWeight','bold',...
644:       'FontSize',14,...
645:       'FontName','Arial');
646: hold(axes1,'all');
647: % Create surface
648: surface('Parent',axes1,'ZData',bacini,'YData',yy,'XData',xx,...

```

```

649:         'LineStyle','none',...
650:         'EdgeColor','none','FaceLighting','phong',...
651:         'CData',bacini);
652:     colormap([1 1 1; 0.7 0.85 0.7;0.7 0.7 0.9; 0 0 0; 0 0 0];...
653:             hsv(length(PK)+5));
654:     colorbar('peer',axes1);
655:     % salvo l'immagine
656:     set(gcf, 'PaperUnits', 'centimeters');
657:     set(gcf, 'PaperSize', [18 19]);
658:     set(gcf, 'PaperPositionMode','manual');
659:     fig_png = [userp '\Immagini\9' name '_' num2str(maglia) 'm_bacini.png'];
660:     print('-dpng',figura9,fig_png);
661:     close figure 1; clear figura9;
662: %-----
663: % FIGURA 10: MATRICE DEI RAMI RELATIVI ALLE AREE DI DRENAGGIO
664: figura10 = figure('Units','centimeters','OuterPosition',[3 1 18 19]);
665: % Create axes
666: axes1 = axes('Parent',figura10,'XLim',[1 n2*maglia],'YLim',[1 n1*maglia],...
667:             'box','on','LineWidth',1,'FontName','Arial','DataAspectRatio',[1 1 1]);
668: xlabel('[m]');ylabel('[m]');
669: titolo = [name ': Canali analizzati'];
670: title(titolo,'FontWeight','bold',...
671:       'FontSize',14,...
672:       'FontName','Arial');
673: hold(axes1,'all');
674: % Create surface
675: surface('Parent',axes1,'ZData',rami,'YData',yy,'XData',xx,...
676:         'LineStyle','none',...
677:         'EdgeColor','none','FaceLighting','phong',...
678:         'CData',rami);
679:     colormap([1 1 1; 0.7 0.85 0.7;0.7 0.7 0.9; 0 0 0; 0 0 0];...
680:             hsv(length(PK)+5));
681:     colorbar('peer',axes1);
682:     % salvo l'immagine
683:     set(gcf, 'PaperUnits', 'centimeters');
684:     set(gcf, 'PaperSize', [18 19]);
685:     set(gcf, 'PaperPositionMode','manual');
686:     fig_png = [userp '\Immagini\10' name '_' num2str(maglia) 'm_rami1.png'];
687:     print('-dpng',figura10,fig_png);
688:     close figure 1; clear figura10;
689: %-----
690: % FIGURA 9-bis: MATRICE DELLE AREE DI DRENAGGIO RICOLORATE
691: load([userp '\randALL.mat']);
692: figura9 = figure('Units','centimeters','OuterPosition',[3 1 18 19]);
693: % Create axes
694: axes1 = axes('Parent',figura9,'XLim',[1 n2*maglia],'YLim',[1 n1*maglia],...
695:             'box','on','LineWidth',1,'FontName','Arial','DataAspectRatio',[1 1 1]);
696: xlabel('[m]');ylabel('[m]');
697: titolo = [name ': Aree di drenaggio ricolorate'];
698: title(titolo,'FontWeight','bold',...
699:       'FontSize',14,...
700:       'FontName','Arial');
701: hold(axes1,'all');
702: % Create surface
703: surface('Parent',axes1,'ZData',bacini,'YData',yy,'XData',xx,...
704:         'LineStyle','none',...
705:         'EdgeColor','none','FaceLighting','phong',...
706:         'CData',bacini);
707: % utilizzo dei colori casuali per descrivere le aree di drenaggio
    
```

```

708: colormap([1 1 1; 0.7 0.85 0.7;0.7 0.7 0.9; 0 0 0; 0 0 0; cmap1]);
709: % salvo l'immagine
710: set(gcf, 'PaperUnits', 'centimeters');
711: set(gcf, 'PaperSize', [18 19]);
712: set(gcf, 'PaperPositionMode','manual');
713: fig_png = [userp '\Immagini\11' name '_' num2str(maglia) 'm_baciniRCOL.png'];
714: print('-dpng',figura9,fig_png);
715: close figure 1; clear figura9;
716: %-----
717: % FIGURA 10 bis: MATRICE DEI RAMI RELATIVI ALLE AREE DI DRENAGGIO RICOLORATI
718: figura10 = figure('Units','centimeters','OuterPosition',[3 1 18 19]);
719: % Create axes
720: axes1 = axes('Parent',figura10,'XLim',[1 n2*maglia],'YLim',[1 n1*maglia],...
721: 'box','on','LineWidth',1,'FontName','Arial','DataAspectRatio',[1 1 1]);
722: xlabel('[m]');ylabel('[m]');
723: titolo = [name : 'Canali analizzati ricolorati'];
724: title(titolo,'FontWeight','bold',...
725: 'FontSize',14,...
726: 'FontName','Arial');
727: hold(axes1,'all');
728: % Create surface
729: surface('Parent',axes1,'ZData',rami,'YData',yy,'XData',xx,...
730: 'LineStyle','none',...
731: 'EdgeColor','none','FaceLighting','phong',...
732: 'CData',rami);
733: % utilizzo dei colori casuali per descrivere le aree di drenaggio
734: colormap([1 1 1; 0.7 0.85 0.7;0.7 0.7 0.9; 0 0 0; 0 0 0; cmap1]);
735: % salvo l'immagine
736: set(gcf, 'PaperUnits', 'centimeters');
737: set(gcf, 'PaperSize', [18 19]);
738: set(gcf, 'PaperPositionMode','manual');
739: fig_png = [userp '\Immagini\12' name '_' num2str(maglia) 'm_ramiRCOL.png'];
740: print('-dpng',figura10,fig_png);
741: close figure 1; clear figura10 camp1;
742: %-----
743: % SALVO I GRAFICI DELL'ANALISI DELLA FORMA DEI BACINI
744: %-----
745: % VARIANZE Vs AREA
746: figura = figure('Units','centimeters','OuterPosition',[3 1 18 19]);
747: hold on
748: semilogx(AK,sigma2S.^5,...
749: 'MarkerFaceColor',[0 1 0],...
750: 'MarkerEdgeColor',[0 0.5 0],...
751: 'MarkerSize',5,...
752: 'Marker','o',...
753: 'LineStyle','none',...
754: 'Color',[0 0 1]);
755: semilogx(AK,sigma2N.^5,...
756: 'MarkerFaceColor',[0 0.5 1],...
757: 'MarkerEdgeColor',[0 0 0.5],...
758: 'MarkerSize',5,...
759: 'Marker','o',...
760: 'LineStyle','none',...
761: 'Color',[0 0 1]);
762: titolo = ['Varianze \sigma_S e \sigma_N rispetto all''area'];
763: title(titolo,'FontWeight','bold',...
764: 'FontSize',14,...
765: 'FontName','Arial');
766: xlabel('log(Area) [m^2]');ylabel('\sigma_S; \sigma_N [m]');
    
```

```

767: set(gca,'XScale','log');
768: grid on
769: box on
770: legend1 = legend('\sigma_S(A)','\sigma_N(A)');
771: set(legend1,'Units','centimeters','Location','NorthWest');
772: % salvo l'immagine
773: set(gcf,'PaperUnits','centimeters');
774: set(gcf,'PaperSize',[18 19]);
775: set(gcf,'PaperPositionMode','manual');
776: fig_png = [userp '\Immagini\ZZ1_SigmaVsArea.png'];
777: print('-dpng',figura,fig_png);close figure 1
778: %-----
779: % DISTRIBUZIONE DELLE VARIANZE LUNGO S
780: figura = figure('Units','centimeters','OuterPosition',[3 1 18 19]);
781: hist(sigma2S.^.5);
782: titolo = ['Distribuzione della varianza \sigma_S'];
783: title(titolo,'FontWeight','bold',...
784:     'FontSize',14,...
785:     'FontName','Arial');
786: xlabel('\sigma_S [m]);ylabel('N');
787: ylim([0 45]);
788: grid on
789: box on
790: % salvo l'immagine
791: set(gcf,'PaperUnits','centimeters');
792: set(gcf,'PaperSize',[18 19]);
793: set(gcf,'PaperPositionMode','manual');
794: fig_png = [userp '\Immagini\ZZ3_DitrSigmaS.png'];
795: print('-dpng',figura,fig_png);close figure 1
796: %-----
797: % DISTRIBUZIONE DELLE VARIANZE LUNGO N
798: figura = figure('Units','centimeters','OuterPosition',[3 1 18 19]);
799: hist(sigma2N.^.5);
800: titolo = ['Distribuzione della varianza \sigma_N'];
801: title(titolo,'FontWeight','bold',...
802:     'FontSize',14,...
803:     'FontName','Arial');
804: xlabel('\sigma_N [m]);ylabel('N');
805: ylim([0 45]);
806: grid on
807: box on
808: % salvo l'immagine
809: set(gcf,'PaperUnits','centimeters');
810: set(gcf,'PaperSize',[18 19]);
811: set(gcf,'PaperPositionMode','manual');
812: fig_png = [userp '\Immagini\ZZ4_DitrSigmaN.png'];
813: print('-dpng',figura,fig_png);close figure 1
814: %-----
815: % CONFRONTO TRA SIGMA_S E SIGMA_N
816: mm = polyfit(sigma2S.^.5,sigma2N.^.5,1);
817: ff = polyval(mm,sort(sigma2S.^.5));
818: figura = figure('Units','centimeters','OuterPosition',[3 1 18 19]);
819: hold on
820: axis equal
821: plot(sigma2S.^.5,sigma2N.^.5,...
822:     'MarkerFaceColor',[0 1 0],...
823:     'MarkerEdgeColor',[0 0 1],...
824:     'MarkerSize',5,...
825:     'Marker','o',...

```

```
826:     'LineStyle','none',...
827:     'Color',[0 0 1]);
828: plot(sort(sigma2S.^.5),ff,'Color',[0 0 1],'linewidth',2,'LineStyle','--');
829: plot([0 300],[0 300],'Color',[1 0 0],'linewidth',1.2,'LineStyle','-');
830: titolo = ['\sigma_S Vs \sigma_N'];
831: title(titolo,'FontWeight','bold',...
832:       'FontSize',14,...
833:       'FontName','Arial');
834: xlabel('\sigma_S [m]);ylabel('\sigma_N [m]);
835: ylim([0 240]);xlim([0 240]);
836: grid on
837: box on
838: legend1 = legend('\sigma_S, \sigma_N','Interpolazione','Bisettrice');
839: set(legend1,'Units','centimeters','Location','SouthEast');
840: % salvo l'immagine
841: set(gcf, 'PaperUnits', 'centimeters');
842: set(gcf, 'PaperSize', [18 19]);
843: set(gcf, 'PaperPositionMode','manual');
844: fig_png = [userp '\Immagini\ZZ5_SigmaSVsSigmaN.png'];
845: print('-dpng',figura,fig_png);close figure 1
846: %-----
847: % DISTRIBUZIONE DEL RAPPORTO TRA LE VARIANZE
848: RS = sigma2N.^.5./sigma2S.^.5;
849: NRS = length(RS(RS<0.75));
850: NRS = [NRS; length(RS(RS>=0.75 & RS<1.25))];
851: NRS = [NRS; length(RS(RS>=1.25 & RS<1.75))];
852: NRS = [NRS; length(RS(RS>=1.75 & RS<2.25))];
853: NRS = [NRS; length(RS(RS>=2.25 & RS<2.75))];
854: NRS = [NRS; length(RS(RS>=2.75 & RS<3.25))];
855: NRS = [NRS; length(RS(RS>=3.25 & RS<3.75))];
856: NRS = [NRS; length(RS(RS>=3.75))];
857: PNRS = NRS./length(RS);
858: parameter= fitdist(RS,'lognormal');
859: PDFRS = pdf('lognorm',0.05:0.05:4.05,parameter.mu,parameter.sigma);
860: figura = figure('Units','centimeters','OuterPosition',[3 1 18 19]);
861: hold on
862: bar(0.5:0.5:4,NRS,'hist')
863: plot(0.05:0.05:4.05,PDFRS.*max(NRS),'r','LineWidth',2)
864: titolo = ['Distribuzione del rapporto \sigma_N/\sigma_S'];
865: title(titolo,'FontWeight','bold',...
866:       'FontSize',14,...
867:       'FontName','Arial');
868: set(gca,'XTick',0.25:0.5:4.25);
869: ylim([0 max(NRS)]);
870: xlabel('\sigma_N/\sigma_S');ylabel('N');
871: grid on
872: box on
873: testo = [{'\mu_1(\sigma_N/\sigma_S) = ' num2str(mean(RS),'%3.2f')'}];
874: text(mean(RS),max(PDFRS)*max(NRS)*1.02,testo,...
875:      'VerticalAlignment','bottom',...
876:      'HorizontalAlignment','left',...
877:      'BackgroundColor',[1 1 1],...
878:      'FontSize',11,...
879:      'EdgeColor',[0 0 0],...
880:      'LineWidth',1,...
881:      'LineStyle','- ',...
882:      'FontName','Arial');
883: % salvo l'immagine
884: set(gcf, 'PaperUnits', 'centimeters');
```

```

885: set(gcf, 'PaperSize', [18 19]);
886: set(gcf, 'PaperPositionMode', 'manual');
887: fig_png = [userp '\Immagini\ZZ6_DitrSigmaNsuSigmaS1.png'];
888: print('-dpng', figura, fig_png); close figure 1
889: %-----
890: % GRAFICI PER IL CONFRONTO DIRETTO DELLE AREE SUDDIVISE IN CLASSI
891: KClass = [15 34 35 63 67 71];
892: figura = figure('Units', 'centimeters', 'OuterPosition', [3 1 20 16]);
893: hold on
894: box on
895: grid on
896: color = jet(length(KClass));
897: for kk=1:length(KClass)
898:     ID = ['nK' num2str(KClass(kk))];
899:     plot(P2P.NSI.(ID)(:,2), -P2P.NSI.(ID)(:,1), ...
900:         'MarkerFaceColor', color(kk,:), ...
901:         'MarkerEdgeColor', color(kk,:).*0.6, ...
902:         'MarkerSize', 4, ...
903:         'Marker', 'o', ...
904:         'LineStyle', 'none');
905:     id(kk) = {'ID = ' num2str(KClass(kk))};
906:     kkk = find(KK == KClass(kk));
907:     tabRS(kk) = {'R_{\sigma}(ID=' num2str(KClass(kk)) ') = ' ...
908:                 num2str(RS(kkk), '%3.2f')'];
909: end
910: axis equal
911: titolo = ['Confronto forma R_{\sigma} > 2.0'];
912: title(titolo, 'FontWeight', 'bold', ...
913:       'FontSize', 14, ...
914:       'FontName', 'Arial');
915: xlabel('S [m]'); ylabel('N [m]');
916: yl=get(gca, 'ylim'); xl=get(gca, 'xlim');
917: text(xl(2)*1.02, yl(1)*0.95, tabRS, ...
918:       'VerticalAlignment', 'bottom', ...
919:       'HorizontalAlignment', 'left', ...
920:       'BackgroundColor', [1 1 1], ...
921:       'FontSize', 10, ...
922:       'EdgeColor', [0 0 0], ...
923:       'LineWidth', 0.5, ...
924:       'LineStyle', '-', ...
925:       'FontName', 'Arial');
926: legend1 = legend(id);
927: set(legend1, 'Units', 'centimeters', 'Location', 'NorthEastOutside');
928: % salvo l'immagine
929: set(gcf, 'PaperUnits', 'centimeters');
930: set(gcf, 'PaperSize', [20 16]);
931: set(gcf, 'PaperPositionMode', 'manual');
932: fig_png = [userp '\Immagini\ZZZ_C4_1.png'];
933: print('-dpng', figura, fig_png); close figure 1; clear id tabRS;
934: %-----
935: figura = figure('Units', 'centimeters', 'OuterPosition', [3 1 20 16]);
936: hold on
937: box on
938: grid on
939: color = jet(length(KClass));
940: for kk=1:length(KClass)
941:     ID = ['nK' num2str(KClass(kk))];
942:     plot(Bt.(ID)(:,1), Bt.(ID)(:,2), ...
943:         'Color', color(kk,:).*0.75, 'LineWidth', 1);

```



```

944:     id(kk) = {'ID = ' num2str(KClass(kk))}];
945:     kkk = find(KK == KClass(kk));
946:     tabRS(kk) = {'R_{\sigma}(ID=' num2str(KClass(kk)) ') = '...
947:                 num2str(RS(kkk), '%3.2f')]}];
948: end
949: axis equal
950: titolo = ['Confronto larghezza R_{\sigma} > 2.0'];
951: title(titolo, 'FontWeight', 'bold', ...
952:       'FontSize', 14, ...
953:       'FontName', 'Arial');
954: xlabel('S [m]'); ylabel('B_t [m]');
955: yl=get(gca, 'ylim'); xl=get(gca, 'xlim');
956: text(xl(2)*1.02, yl(1)*0.95, tabRS, ...
957:      'VerticalAlignment', 'bottom', ...
958:      'HorizontalAlignment', 'left', ...
959:      'BackgroundColor', [1 1 1], ...
960:      'FontSize', 10, ...
961:      'EdgeColor', [0 0 0], ...
962:      'LineWidth', 0.5, ...
963:      'LineStyle', '-', ...
964:      'FontName', 'Arial');
965: legend1 = legend(id);
966: set(legend1, 'Units', 'centimeters', 'Location', 'NorthEastOutside');
967: % salvo l'immagine
968: set(gcf, 'PaperUnits', 'centimeters');
969: set(gcf, 'PaperSize', [20 16]);
970: set(gcf, 'PaperPositionMode', 'manual');
971: fig_png = [userp '\Immagini\ZZZ_C4_1Larghezza.png'];
972: print('-dpng', figura, fig_png); close figure 1; clear id tabRS;
973: %-----
974: %-----
975: KClass = [19 45 52 56];
976: figura = figure('Units', 'centimeters', 'OuterPosition', [3 1 16 18]);
977: hold on
978: box on
979: grid on
980: color = jet(length(KClass));
981: for kk=1:length(KClass)
982:     ID = ['nK' num2str(KClass(kk))];
983:     plot(P2P.NSI.(ID)(:,2), -P2P.NSI.(ID)(:,1), ...
984:          'MarkerFaceColor', color(kk,:), ...
985:          'MarkerEdgeColor', color(kk,:).*0.6, ...
986:          'MarkerSize', 4, ...
987:          'Marker', 'o', ...
988:          'LineStyle', 'none');
989:     id(kk) = {'ID = ' num2str(KClass(kk))}];
990:     kkk = find(KK == KClass(kk));
991:     tabRS(kk) = {'R_{\sigma}(ID=' num2str(KClass(kk)) ') = '...
992:                 num2str(RS(kkk), '%3.2f')]}];
993: end
994: axis equal
995: titolo = ['Confronto forma R_{\sigma} <= 0.8'];
996: title(titolo, 'FontWeight', 'bold', ...
997:       'FontSize', 14, ...
998:       'FontName', 'Arial');
999: xlabel('S [m]'); ylabel('N [m]');
1000: yl=get(gca, 'ylim'); xl=get(gca, 'xlim');
1001: text(xl(2)*1.02, yl(1)*0.95, tabRS, ...
1002:      'VerticalAlignment', 'bottom', ...

```

```

1003:     'HorizontalAlignment','left',...
1004:     'BackgroundColor',[1 1 1],...
1005:     'FontSize',10,...
1006:     'EdgeColor',[0 0 0],...
1007:     'LineWidth',0.5,...
1008:     'LineStyle','- ',...
1009:     'FontName','Arial');
1010: legend1 = legend(id);
1011: set(legend1,'Units','centimeters','Location','NorthEastOutside');
1012: % salvo l'immagine
1013: set(gcf, 'PaperUnits', 'centimeters');
1014: set(gcf, 'PaperSize', [20 16]);
1015: set(gcf, 'PaperPositionMode','manual');
1016: fig_png = [userp '\Immagini\ZZZ_Cl_1.png'];
1017: print('-dpng',figura,fig_png);close figure 1;clear id tabRS;
1018: %-----
1019: figura = figure('Units','centimeters','OuterPosition',[3 1 16 18]);
1020: hold on
1021: box on
1022: grid on
1023: color = jet(length(KClass));
1024: for kk=1:length(KClass)
1025:     ID = ['nK' num2str(KClass(kk))];
1026:     plot(Bt.(ID)(:,1),Bt.(ID)(:,2),...
1027:         'Color',color(kk,:), 'LineWidth',1);
1028:     id(kk) = {'ID = ' num2str(KClass(kk))}];
1029:     kkk = find(KK == KClass(kk));
1030:     tabRS(kk) = {'R_{\sigma}(ID=' num2str(KClass(kk)) ') = '...
1031:                 num2str(RS(kkk), '%3.2f')}]];
1032: end
1033: axis equal
1034: titolo = ['Confronto larghezza R_{\sigma} <= 0.8'];
1035: title(titolo,'FontWeight','bold',...
1036:     'FontSize',14,...
1037:     'FontName','Arial');
1038: xlabel('S [m]);ylabel('B_t [m]);
1039: yl=get(gca,'ylim');xl=get(gca,'xlim');
1040: text(xl(2)*1.02,yl(1)*0.95,tabRS,...
1041:     'VerticalAlignment','bottom',...
1042:     'HorizontalAlignment','left',...
1043:     'BackgroundColor',[1 1 1],...
1044:     'FontSize',10,...
1045:     'EdgeColor',[0 0 0],...
1046:     'LineWidth',0.5,...
1047:     'LineStyle','- ',...
1048:     'FontName','Arial');
1049: legend1 = legend(id);
1050: set(legend1,'Units','centimeters','Location','NorthEastOutside');
1051: % salvo l'immagine
1052: set(gcf, 'PaperUnits', 'centimeters');
1053: set(gcf, 'PaperSize', [20 16]);
1054: set(gcf, 'PaperPositionMode','manual');
1055: fig_png = [userp '\Immagini\ZZZ_Cl_1Larghezza.png'];
1056: print('-dpng',figura,fig_png);close figure 1;clear id tabRS;
1057: %-----
1058: %-----
1059: KClass = [11 18 22 24 25 54];
1060: figura = figure('Units','centimeters','OuterPosition',[3 1 18 18]);
1061: hold on
    
```

```

1062:  box on
1063:  grid on
1064:  color = jet(length(KClass));
1065:  for kk=1:length(KClass)
1066:      ID = ['nK' num2str(KClass(kk))];
1067:      plot(P2P.NSI.(ID)(:,2),-P2P.NSI.(ID)(:,1),...
1068:          'MarkerFaceColor',color(kk,:),...
1069:          'MarkerEdgeColor',color(kk,:).*0.7,...
1070:          'MarkerSize',4,...
1071:          'Marker','o',...
1072:          'LineStyle','none');
1073:      id(kk) = {'ID = ' num2str(KClass(kk))}];
1074:      kkk = find(KK == KClass(kk));
1075:      tabRS(kk) = {'R_{\sigma}(ID=' num2str(KClass(kk)) ') = '...
1076:                  num2str(RS(kkk),'%3.2f')]}];
1077:  end
1078:  axis equal
1079:  titolo = ['Confronto forma 0.8 < R_{\sigma} < 1.3'];
1080:  title(titolo,'FontWeight','bold',...
1081:        'FontSize',14,...
1082:        'FontName','Arial');
1083:  xlabel('S [m]');ylabel('N [m]');
1084:  yl=get(gca,'ylim');xl=get(gca,'xlim');
1085:  text(xl(2)*1.02,yl(1)*0.95,tabRS,...
1086:        'VerticalAlignment','bottom',...
1087:        'HorizontalAlignment','left',...
1088:        'BackgroundColor',[1 1 1],...
1089:        'FontSize',10,...
1090:        'EdgeColor',[0 0 0],...
1091:        'LineWidth',0.5,...
1092:        'LineStyle','- ',...
1093:        'FontName','Arial');
1094:  legend1 = legend(id);
1095:  set(legend1,'Units','centimeters','Location','NorthEastOutside');
1096:  % salvo l'immagine
1097:  set(gcf, 'PaperUnits', 'centimeters');
1098:  set(gcf, 'PaperSize', [20 16]);
1099:  set(gcf, 'PaperPositionMode','manual');
1100:  fig_png = [userp '\Immagini\ZZZ_C2_1.png'];
1101:  print('-dpng',figura,fig_png);close figure 1;clear id tabRS;
1102:  %-----
1103:  figura = figure('Units','centimeters','OuterPosition',[3 1 18 18]);
1104:  hold on
1105:  box on
1106:  grid on
1107:  color = jet(length(KClass));
1108:  for kk=1:length(KClass)
1109:      ID = ['nK' num2str(KClass(kk))];
1110:      plot(Bt.(ID)(:,1),Bt.(ID)(:,2),...
1111:          'Color',color(kk,:).*0.8,'LineWidth',1);
1112:      id(kk) = {'ID = ' num2str(KClass(kk))}];
1113:      kkk = find(KK == KClass(kk));
1114:      tabRS(kk) = {'R_{\sigma}(ID=' num2str(KClass(kk)) ') = '...
1115:                  num2str(RS(kkk),'%3.2f')]}];
1116:  end
1117:  axis equal
1118:  titolo = ['Confronto larghezza 0.8 < R_{\sigma} < 1.3'];
1119:  title(titolo,'FontWeight','bold',...
1120:        'FontSize',14,...

```

```

1121:     'FontName','Arial');
1122: xlabel('S [m]');ylabel('B_t [m]');
1123: yl=get(gca,'ylim');xl=get(gca,'xlim');
1124: text(xl(2)*1.02,yl(1)*0.95,tabRS,...
1125:     'VerticalAlignment','bottom',...
1126:     'HorizontalAlignment','left',...
1127:     'BackgroundColor',[1 1 1],...
1128:     'FontSize',10,...
1129:     'EdgeColor',[0 0 0],...
1130:     'LineWidth',0.5,...
1131:     'LineStyle','- ',...
1132:     'FontName','Arial');
1133: legend1 = legend(id);
1134: set(legend1,'Units','centimeters','Location','NorthEastOutside');
1135: % salvo l'immagine
1136: set(gcf, 'PaperUnits', 'centimeters');
1137: set(gcf, 'PaperSize', [20 16]);
1138: set(gcf, 'PaperPositionMode','manual');
1139: fig_png = [userp '\Immagini\ZZZ_C2_1Larghezza.png'];
1140: print('-dpng',figura,fig_png);close figure 1;clear id tabRS;
1141: %-----
1142: %-----
1143: KClass = [31 33 39 46];
1144: figura = figure('Units','centimeters','OuterPosition',[3 1 18 18]);
1145: hold on
1146: box on
1147: grid on
1148: color = jet(length(KClass));
1149: for kk=1:length(KClass)
1150:     ID = ['nK' num2str(KClass(kk))];
1151:     plot(P2P.NSI.(ID)(:,2),-P2P.NSI.(ID)(:,1),...
1152:         'MarkerFaceColor',color(kk,:),...
1153:         'MarkerEdgeColor',color(kk,:).*0.6,...
1154:         'MarkerSize',4,...
1155:         'Marker','o',...
1156:         'LineStyle','none');
1157:     id(kk) = {'ID = ' num2str(KClass(kk))};
1158:     kkk = find(KK == KClass(kk));
1159:     tabRS(kk) = {'R_{\sigma}(ID=' num2str(KClass(kk)) ') = '...
1160:                 num2str(RS(kkk),'%3.2f')];
1161: end
1162: axis equal
1163: titolo = ['Confronto forma 0.8 < R_{\sigma} < 1.3'];
1164: title(titolo,'FontWeight','bold',...
1165:     'FontSize',14,...
1166:     'FontName','Arial');
1167: xlabel('S [m]');ylabel('N [m]');
1168: yl=get(gca,'ylim');xl=get(gca,'xlim');
1169: text(xl(2)*1.02,yl(1)*0.95,tabRS,...
1170:     'VerticalAlignment','bottom',...
1171:     'HorizontalAlignment','left',...
1172:     'BackgroundColor',[1 1 1],...
1173:     'FontSize',10,...
1174:     'EdgeColor',[0 0 0],...
1175:     'LineWidth',0.5,...
1176:     'LineStyle','- ',...
1177:     'FontName','Arial');
1178: legend1 = legend(id);
1179: set(legend1,'Units','centimeters','Location','NorthEastOutside');
    
```

```
1180: % salvo l'immagine
1181: set(gcf, 'PaperUnits', 'centimeters');
1182: set(gcf, 'PaperSize', [20 16]);
1183: set(gcf, 'PaperPositionMode', 'manual');
1184: fig_png = [userp '\Immagini\ZZZ_C2_2.png'];
1185: print('-dpng', figura, fig_png); close figure 1; clear id tabRS;
1186: %-----
1187: figura = figure('Units', 'centimeters', 'OuterPosition', [3 1 18 18]);
1188: hold on
1189: box on
1190: grid on
1191: color = jet(length(KClass));
1192: for kk=1:length(KClass)
1193:     ID = ['nK' num2str(KClass(kk))];
1194:     plot(Bt.(ID)(:,1), Bt.(ID)(:,2), ...
1195:         'Color', color(kk,:), 'LineWidth', 1);
1196:     id(kk) = {'ID = ' num2str(KClass(kk))}';
1197:     kkk = find(KK == KClass(kk));
1198:     tabRS(kk) = {'R_{\sigma}(ID=' num2str(KClass(kk)) ') = ' ...
1199:                 num2str(RS(kkk), '%3.2f')}';
1200: end
1201: axis equal
1202: titolo = ['Confronto larghezza 0.8 < R_{\sigma} < 1.3'];
1203: title(titolo, 'FontWeight', 'bold', ...
1204:     'FontSize', 14, ...
1205:     'FontName', 'Arial');
1206: xlabel('S [m]'); ylabel('B_t [m]');
1207: yl=get(gca, 'ylim'); xl=get(gca, 'xlim');
1208: text(xl(2)*1.02, yl(1)*0.95, tabRS, ...
1209:     'VerticalAlignment', 'bottom', ...
1210:     'HorizontalAlignment', 'left', ...
1211:     'BackgroundColor', [1 1 1], ...
1212:     'FontSize', 10, ...
1213:     'EdgeColor', [0 0 0], ...
1214:     'LineWidth', 0.5, ...
1215:     'LineStyle', '-', ...
1216:     'FontName', 'Arial');
1217: legend1 = legend(id);
1218: set(legend1, 'Units', 'centimeters', 'Location', 'NorthEastOutside');
1219: % salvo l'immagine
1220: set(gcf, 'PaperUnits', 'centimeters');
1221: set(gcf, 'PaperSize', [20 16]);
1222: set(gcf, 'PaperPositionMode', 'manual');
1223: fig_png = [userp '\Immagini\ZZZ_C2_2Larghezza.png'];
1224: print('-dpng', figura, fig_png); close figure 1; clear id tabRS;
1225: %-----
1226: %-----
1227: KClass = [16 21 50 51 61];
1228: figura = figure('Units', 'centimeters', 'OuterPosition', [3 1 18 18]);
1229: hold on
1230: box on
1231: grid on
1232: color = jet(length(KClass));
1233: for kk=1:length(KClass)
1234:     ID = ['nK' num2str(KClass(kk))];
1235:     plot(P2P.NSI.(ID)(:,2), -P2P.NSI.(ID)(:,1), ...
1236:         'MarkerFaceColor', color(kk,:), ...
1237:         'MarkerEdgeColor', color(kk,:).*0.6, ...
1238:         'MarkerSize', 4, ...
```

```
1239:     'Marker','o',...
1240:     'LineStyle','none');
1241:     id(kk) = {'ID = ' num2str(KClass(kk))}];
1242:     kkk = find(KK == KClass(kk));
1243:     tabRS(kk) = {'R_{\sigma}(ID=' num2str(KClass(kk)) ') = '...
1244:                 num2str(RS(kkk), '%3.2f')]}];
1245: end
1246: axis equal
1247: titolo = ['Confronto forma 0.8 < R_{\sigma} < 1.3'];
1248: title(titolo,'FontWeight','bold',...
1249:       'FontSize',14,...
1250:       'FontName','Arial');
1251: xlabel('S [m]);ylabel('N [m]');
1252: yl=get(gca,'ylim');xl=get(gca,'xlim');
1253: text(xl(2)*1.02,yl(1)*0.95,tabRS,...
1254:      'VerticalAlignment','bottom',...
1255:      'HorizontalAlignment','left',...
1256:      'BackgroundColor',[1 1 1],...
1257:      'FontSize',10,...
1258:      'EdgeColor',[0 0 0],...
1259:      'LineWidth',0.5,...
1260:      'LineStyle','- ',...
1261:      'FontName','Arial');
1262: legend1 = legend(id);
1263: set(legend1,'Units','centimeters','Location','NorthEastOutside');
1264: % salvo l'immagine
1265: set(gcf, 'PaperUnits', 'centimeters');
1266: set(gcf, 'PaperSize', [20 16]);
1267: set(gcf, 'PaperPositionMode','manual');
1268: fig_png = [userp '\Immagini\ZZZ_C2_3.png'];
1269: print('-dpng',figura,fig_png);close figure 1;clear id tabRS;
1270: %-----
1271: figura = figure('Units','centimeters','OuterPosition',[3 1 18 18]);
1272: hold on
1273: box on
1274: grid on
1275: color = jet(length(KClass));
1276: for kk=1:length(KClass)
1277:     ID = ['nK' num2str(KClass(kk))];
1278:     plot(Bt.(ID)(:,1),Bt.(ID)(:,2),...
1279:          'Color',color(kk,:), 'LineWidth',1);
1280:     id(kk) = {'ID = ' num2str(KClass(kk))}];
1281:     kkk = find(KK == KClass(kk));
1282:     tabRS(kk) = {'R_{\sigma}(ID=' num2str(KClass(kk)) ') = '...
1283:                 num2str(RS(kkk), '%3.2f')]}];
1284: end
1285: axis equal
1286: titolo = ['Confronto larghezza 0.8 < R_{\sigma} < 1.3'];
1287: title(titolo,'FontWeight','bold',...
1288:       'FontSize',14,...
1289:       'FontName','Arial');
1290: xlabel('S [m]);ylabel('B_t [m]');
1291: yl=get(gca,'ylim');xl=get(gca,'xlim');
1292: text(xl(2)*1.02,yl(1)*0.95,tabRS,...
1293:      'VerticalAlignment','bottom',...
1294:      'HorizontalAlignment','left',...
1295:      'BackgroundColor',[1 1 1],...
1296:      'FontSize',10,...
1297:      'EdgeColor',[0 0 0],...
```

```
1298:     'LineWidth',0.5,...
1299:     'LineStyle','-',...
1300:     'FontName','Arial');
1301: legend1 = legend(id);
1302: set(legend1,'Units','centimeters','Location','NorthEastOutside');
1303: % salvo l'immagine
1304: set(gcf, 'PaperUnits', 'centimeters');
1305: set(gcf, 'PaperSize', [20 16]);
1306: set(gcf, 'PaperPositionMode','manual');
1307: fig_png = [userp '\Immagini\ZZZ_C2_3Larghezza.png'];
1308: print('-dpng',figura,fig_png);close figure 1;clear id tabRS;
1309: %-----
1310: %-----
1311: KClass = [9 17 32 43 64];
1312: figura = figure('Units','centimeters','OuterPosition',[3 1 18 16]);
1313: hold on
1314: box on
1315: grid on
1316: color = jet(length(KClass));
1317: for kk=1:length(KClass)
1318:     ID = ['nK' num2str(KClass(kk))];
1319:     plot(P2P.NSI.(ID)(:,2),-P2P.NSI.(ID)(:,1),...
1320:         'MarkerFaceColor',color(kk,:),...
1321:         'MarkerEdgeColor',color(kk,:).*0.6,...
1322:         'MarkerSize',4,...
1323:         'Marker','o',...
1324:         'LineStyle','none');
1325:     id(kk) = {'ID = ' num2str(KClass(kk))}];
1326:     kkk = find(KK == KClass(kk));
1327:     tabRS(kk) = {'R_{\sigma}(ID=' num2str(KClass(kk)) ') = '...
1328:                 num2str(RS(kkk),'%3.2f')]}];
1329: end
1330: axis equal
1331: titolo = ['Confronto forma 1.3 < R_{\sigma} \leq 2.0'];
1332: title(titolo,'FontWeight','bold',...
1333:       'FontSize',14,...
1334:       'FontName','Arial');
1335: xlabel('S [m]');ylabel('N [m]');
1336: yl=get(gca,'ylim');xl=get(gca,'xlim');
1337: text(xl(2)*1.02,yl(1)*0.95,tabRS,...
1338:      'VerticalAlignment','bottom',...
1339:      'HorizontalAlignment','left',...
1340:      'BackgroundColor',[1 1 1],...
1341:      'FontSize',10,...
1342:      'EdgeColor',[0 0 0],...
1343:      'LineWidth',0.5,...
1344:      'LineStyle','-',...
1345:      'FontName','Arial');
1346: legend1 = legend(id);
1347: set(legend1,'Units','centimeters','Location','NorthEastOutside');
1348: % salvo l'immagine
1349: set(gcf, 'PaperUnits', 'centimeters');
1350: set(gcf, 'PaperSize', [20 16]);
1351: set(gcf, 'PaperPositionMode','manual');
1352: fig_png = [userp '\Immagini\ZZZ_C3_1.png'];
1353: print('-dpng',figura,fig_png);close figure 1;clear id tabRS;
1354: %-----
1355: figura = figure('Units','centimeters','OuterPosition',[3 1 18 16]);
1356: hold on
```

```

1357:  box on
1358:  grid on
1359:  color = jet(length(KClass));
1360:  for kk=1:length(KClass)
1361:      ID = ['nK' num2str(KClass(kk))];
1362:      plot(Bt.(ID)(:,1),Bt.(ID)(:,2),...
1363:          'Color',color(kk,:), 'LineWidth',1);
1364:      id(kk) = {'ID = ' num2str(KClass(kk))};
1365:      kkk = find(KK == KClass(kk));
1366:      tabRS(kk) = {'R_{\sigma}(ID=' num2str(KClass(kk)) ' ) = '...
1367:                  num2str(RS(kkk), '%3.2f')'];
1368:  end
1369:  axis equal
1370:  titolo = ['Confronto larghezza 1.3 < R_{\sigma} \leq 2.0'];
1371:  title(titolo, 'FontWeight', 'bold',...
1372:        'FontSize',14,...
1373:        'FontName', 'Arial');
1374:  xlabel('S [m]);ylabel('B_t [m]');
1375:  yl=get(gca, 'ylim');xl=get(gca, 'xlim');
1376:  text(xl(2)*1.02,yl(1)*0.95,tabRS,...
1377:        'VerticalAlignment', 'bottom',...
1378:        'HorizontalAlignment', 'left',...
1379:        'BackgroundColor', [1 1 1],...
1380:        'FontSize',10,...
1381:        'EdgeColor', [0 0 0],...
1382:        'LineWidth',0.5,...
1383:        'LineStyle', '-',...
1384:        'FontName', 'Arial');
1385:  legend1 = legend(id);
1386:  set(legend1, 'Units', 'centimeters', 'Location', 'NorthEastOutside');
1387:  % salvo l'immagine
1388:  set(gcf, 'PaperUnits', 'centimeters');
1389:  set(gcf, 'PaperSize', [20 16]);
1390:  set(gcf, 'PaperPositionMode', 'manual');
1391:  fig_png = [userp '\Immagini\ZZZ_C3_1Larghezza.png'];
1392:  print('-dpng', figura, fig_png);close figure 1;clear id tabRS;
1393:  %-----
1394:  %-----
1395:  KClass = [23 31 45 55 62];
1396:  figura = figure('Units', 'centimeters', 'OuterPosition', [3 1 18 16]);
1397:  hold on
1398:  box on
1399:  grid on
1400:  color = jet(length(KClass));
1401:  for kk=1:length(KClass)
1402:      ID = ['nK' num2str(KClass(kk))];
1403:      plot(P2P.NSI.(ID)(:,2),-P2P.NSI.(ID)(:,1),...
1404:          'MarkerFaceColor',color(kk,:),...
1405:          'MarkerEdgeColor',color(kk,:).*0.9,...
1406:          'MarkerSize',2,...
1407:          'Marker', 'o',...
1408:          'LineStyle', 'none');
1409:      id(kk) = {'ID = ' num2str(KClass(kk))};
1410:      kkk = find(KK == KClass(kk));
1411:      tabRS(kk) = {'R_{\sigma}(ID=' num2str(KClass(kk)) ' ) = '...
1412:                  num2str(RS(kkk), '%3.2f')'];
1413:  end
1414:  axis equal
1415:  titolo = ['Confronto forma tra le varie classi'];
    
```



```
1416: title(titolo,'FontWeight','bold',...
1417:     'FontSize',14,...
1418:     'FontName','Arial');
1419: xlabel('S [m]');ylabel('N [m]');
1420: yl=get(gca,'ylim');xl=get(gca,'xlim');
1421: text(xl(2)*1.02,yl(1)*0.95,tabRS,...
1422:     'VerticalAlignment','bottom',...
1423:     'HorizontalAlignment','left',...
1424:     'BackgroundColor',[1 1 1],...
1425:     'FontSize',10,...
1426:     'EdgeColor',[0 0 0],...
1427:     'LineWidth',0.5,...
1428:     'LineStyle','- ',...
1429:     'FontName','Arial');
1430: legend1 = legend(id);
1431: set(legend1,'Units','centimeters','Location','NorthEastOutside');
1432: % salvo l'immagine
1433: set(gcf, 'PaperUnits', 'centimeters');
1434: set(gcf, 'PaperSize', [20 16]);
1435: set(gcf, 'PaperPositionMode','manual');
1436: fig_png = [userp '\Immagini\ZZZ_C_mista.png'];
1437: print('-dpng',figura,fig_png);close figure 1;clear id tabRS;
1438: %-----
1439: figura = figure('Units','centimeters','OuterPosition',[3 1 18 16]);
1440: hold on
1441: box on
1442: grid on
1443: color = jet(length(KClass));
1444: for kk=1:length(KClass)
1445:     ID = ['nK' num2str(KClass(kk))];
1446:     plot(Bt.(ID)(:,1),Bt.(ID)(:,2),...
1447:         'Color',color(kk,:), 'LineWidth',1);
1448:     id(kk) = {'ID = ' num2str(KClass(kk))}];
1449:     kkk = find(KK == KClass(kk));
1450:     tabRS(kk) = {'R_{\sigma}(ID=' num2str(KClass(kk)) ') = '...
1451:                 num2str(RS(kkk),'%3.2f')}]];
1452: end
1453: axis equal
1454: titolo = ['Confronto larghezza tra le varie classi'];
1455: title(titolo,'FontWeight','bold',...
1456:     'FontSize',14,...
1457:     'FontName','Arial');
1458: xlabel('S [m]');ylabel('B_t [m]');
1459: yl=get(gca,'ylim');xl=get(gca,'xlim');
1460: text(xl(2)*1.02,yl(1)*0.95,tabRS,...
1461:     'VerticalAlignment','bottom',...
1462:     'HorizontalAlignment','left',...
1463:     'BackgroundColor',[1 1 1],...
1464:     'FontSize',10,...
1465:     'EdgeColor',[0 0 0],...
1466:     'LineWidth',0.5,...
1467:     'LineStyle','- ',...
1468:     'FontName','Arial');
1469: legend1 = legend(id);
1470: set(legend1,'Units','centimeters','Location','NorthEastOutside');
1471: % salvo l'immagine
1472: set(gcf, 'PaperUnits', 'centimeters');
1473: set(gcf, 'PaperSize', [20 16]);
1474: set(gcf, 'PaperPositionMode','manual');
```

```
1475: fig_png = [userp '\Immagini\ZZZ_Cmista_Larghezza.png'];  
1476: print('-dpng',figura,fig_png);close figure 1;clear id tabRS;  
1477: %-----  
1478:
```

```

1: % [sol,iter]=telone(contorno,n_marsh,imax,toll,maglia,U_max,dETA0_dt,CHI,ETA0);
2: %=====
3: % Questo e' il programma telone originale che non tiene conto delle condizioni
4: % al CONTORNO nei punti canale, divenuto subroutine e poi mex-file.
5: % Su tutti i punti canale l'elevazione e' posta pari a zero.
6: % L'unica modifica apportata, rispetto al modello originale consiste nel
7: % risolvere l'equazione '-laplaciano di eta = KAPPA', anziche' =1
8: %=====
9: % INPUT:
10: % - contorno: (INPUT OBBLIGATORIO) matrice contenente le caratteristiche
11: % dell'area di studio (i.e. matrice delle condizioni al contorno):
12: % LA CONVENZIONE SUI PUNTI DELLA LAGUNA E' LA SEGUENTE:
13: % 0 APPARTIENE ALLO SPECCHIO D'ACQUA
14: % 1 CONDIZIONE AL contorno DI DIRICHLET
15: % 2 CONDIZIONE AL contorno DI NEUMANN PER PUNTI NON CANALE
16: % 3 CONDIZIONE AL contorno DI NEUMANN PER PUNTI CANALE
17: % 4 PUNTI ANCORA CANALE MA DI INIZIO PER LA FUNZIONE DI AMPIEZZA
18: % - n_marsh: (INPUT FACOLTATIVO) n di pixel barena presenti nella matrice
19: % contorno, se non specificato la funzione lo calcola autonomamente
20: % cercando gli elementi della matrice pari a zero.
21: % - imax:(INPUT FACOLTATIVO) n massimo di iterazioni consentite per la
22: % soluzione del sistema lineare col metodo del gradiente coniugato,
23: % preconditionato con la decomposta incompleta di Cholesky;
24: % (DEFAULT => imax = 10^4).
25: % - toll:(INPUT FACOLTATIVO) tolleranza ammessa per la soluzione approssimata,
26: % determinata con l'algoritmo del gradiente coniugato,
27: % preconditionato con la decomposta incompleta di Cholesky.
28: % (DEFAULT => toll = 10^-5).
29: %
30: % PARAMETRI FISICI DEL PROBLEMA (OPZIONALI):
31: % - maglia: dimensione del lato della maglia quadrata del reticolo di calcolo
32: % (DEFAULT => maglia = 1 [m]).
33: % - U_max: valore della velocita' massima sulla barena
34: % (DEFAULT => U_max = 0.15 [m/s]).
35: % - dETA0_dt: derivata dell'elevazione media istantanea rispetto al tempo
36: % (DEFAULT => dETA0_dt = 7.5*10^-5).
37: % - CHI: valore del coefficiente di scabrezza relativa di Chezy
38: % (DEFAULT => CHI = 10 [m^.5/s]).
39: % - ETA0: valore dell'elevazione media istantanea della marea
40: % (DEFAULT => ETA0 = 0.35 [m]).
41: %
42: % OUTPUT(OBBLIGATORI):
43: % - sol: matrice in cui si descrive l'andamento della soluzione calcolata in
44: % tutti i pixel barena.
45: % - iter: n d'iterazioni dell'algoritmo del GCP.
46: %
47: % NOTA1: e' possibile assegnare i parametri di input in tre modi diversi:
48: % - assegnando solo la matrice 'contorno' la funzione calcolera'
49: % autonomamente i valori di default per gli altri parametri e li
50: % scrivera' a video chiedendo conferma all'utente;
51: % - assegnando tutti i parametri richiesti RISPETTANDO STRETTAMENTE
52: % L'ORDINE (contorno,n_canali,n_bocche,passo_sezione), la funzione
53: % eseguirà il calcolo segnalando a video solo eventuali errori.
54: %
55: % !!!NB:E' FONDAMENTALE IN QUESTO CASO IL RISPETTO DELL'ORDINE!!!
56: %
57: % - assegnando solo alcuni dei parametri facoltativi specificandoli,
58: % dopo l'assegnazione della matrice 'contorno', in qualsiasi ordine
59: % assegnandoli dopo aver posto tra apici il nome del parametro che si

```

```

60: %           desidera imporre.
61: %
62: %           ESEMPIO:
63: %           Se si vuole imporre solo alcuni dei parametri opzionali
64: %           e' possibile farlo col comando:
65: %           [sol,iter]=telone(contorno,'imax',100,'toll',10^(-3));
66: %           In questo modo la funzione scrivera' ugualmente a video i parame-
67: %           tri facoltativi e chiederà conferma all'utente prima di eseguire
68: %           il calcolo.
69: %
70: % NOTA2: questa funzione e' composta in parte da codice matlab e in parte da un
71: % mex_file. La matrice contorno puo' essere al massimo di 2000x2000 ele-
72: % menti, anche se l'effettivo funzionamento della funzione dipende dal
73: % numero di pixel barena.
74: %
75: % NB: LA FUNZIONE E' STATA TESTATA FINO AD UN NUMERO MASSIMO DI PIXEL BARENA
76: % CIRCA PARI A 4'000'000. E' SCONSIGLIABILE L'UTILIZZO DI QUESTA FUNZIONE
77: % SE I PIXEL SU CUI SI DEVE CALCOLARE LA SOLUZIONE ECCEDONO QUESTO NUMERO
78: %
79: %=====
80: % PER L'ESECUZIONE E' RICHIESTA LA TOOLBOX TELONE_TBOX
81: %=====
82: % data di creazione: 23/04/2014
83: % ultima modifica: 17/06/2014
84: % autore: Francesco Carraro
85: %=====
86: function [sol,iter]=telone(contorno,varargin);
87: % creo la struttura di input: preparo i parametri da dare alla mex_function,
88: %                               impostando i valori di default per i parametri
89: %                               opzionali.
90: % creo la struttura
91: in = inputParser;
92: % scelgo i valori di default
93: defaultN_marsh = length(contorno(contorno==0));
94: defaultImax = 10000;
95: defaultToll = 10^(-5);
96: defaultMaglia = 1;
97: defaultU_max = 0.15;
98: defaultDETA0_dt = 0.000075;
99: defaultCHI = 10;
100: defaultETA0 = 0.35;
101: defaultZ = zeros(size(contorno));
102: % creo i parametri obbligatori
103: addRequired(in,'contorno',@isnumeric);
104: % creo i parametri facoltativi
105: addOptional(in,'n_marsh',defaultN_marsh,@isnumeric);
106: addOptional(in,'imax',defaultImax,@isnumeric);
107: addOptional(in,'toll',defaultToll,@isnumeric);
108: addOptional(in,'maglia',defaultMaglia,@isnumeric);
109: addOptional(in,'U_max',defaultU_max,@isnumeric);
110: addOptional(in,'dETA0_dt',defaultDETA0_dt,@isnumeric);
111: addOptional(in,'CHI',defaultCHI,@isnumeric);
112: addOptional(in,'ETA0',defaultETA0,@isnumeric);
113: addOptional(in,'z',defaultZ,@isnumeric);
114: % creo il file struttura risultante con gli input corretti per il mex
115: parse(in,contorno,varargin{:});
116:
117: % nel caso non siano stati assegnati tutti i parametri richiesti, scrivo a video
118: % i valori che la funzione ha scelto per l'esecuzione della mex_function

```

```
119: num = 0;
120: for iii=1:length(varargin),num=num+isnumeric(varargin{iii});end
121: if (length(varargin) < 18) && num < 8
122:     fprintf('\nMANCANO ALCUNI POSSIBILI INPUT OPZIONALI!\n')
123:     fprintf('Per l''esecuzione della funzione verranno utilizzati i seguenti\n')
124:     fprintf('valori delle variabili opzionali:\n')
125:     fprintf('\t- n_marsh = %d;\n',in.Results.n_marsh)
126:     fprintf('\t- imax = %d;\n',in.Results.imax)
127:     fprintf('\t- toll = %4.3g;\n',in.Results.toll)
128:     fprintf('\t- maglia = %g [m];\n',in.Results.maglia)
129:     fprintf('\t- U_max = %4.3g [m/s];\n',in.Results.U_max)
130:     fprintf('\t- CHI = %3.1g [m^.5/s];\n',in.Results.CHI)
131:     fprintf('\t- dETA0_dt = %4.3g;\n',in.Results.dETA0_dt)
132:     fprintf('\t- ETA0 = %4.3g [m];\n',in.Results.ETA0)
133:     fprintf('Per ulteriori informazioni digitare a linea di comando:\n')
134:     fprintf('help telone\n')
135:     fprintf('PREMERE [INVIO] PER CONTINUARE ([ctrl+C] per uscire)\n')
136:     pause
137: end
138:     fprintf(' ...')
139:     fprintf('ESECUZIONE DELLA FUNZIONE telone IN CORSO...\n')
140:
141: tic;
142: % si richiama il mex_file creato tramite F_telone.f90
143: [sol,iter]=F_telone(in.Results.contorno,...
144:                   in.Results.n_marsh,...
145:                   in.Results.imax,...
146:                   in.Results.maglia,...
147:                   in.Results.U_max,...
148:                   in.Results.dETA0_dt,...
149:                   in.Results.CHI,...
150:                   in.Results.ETA0,...
151:                   in.Results.toll,...
152:                   in.Results.z);
153: t = toc;
154: fprintf(...
155: ' Per trovare la soluzione sono state necessarie %d iterazioni\n',iter)
156: fprintf(...
157: ' dell''algoritmo del Gradiente Coniugato Modificato\n')
158: fprintf(' preconditionato con la fattorizzazione di Cholesky.\n')
159: fprintf(...
160: ' l''elaborazione di telone e'' andata a buon fine in %4.1f secondi. \n',t)
161:
162:
```



```

1: !      The gateway routine: ovvero la subroutine che lega gli oggetti tipici
2: !      di matlab con le classiche variabili integer e real
3:      subroutine mexFunction(nlhs, plhs, nrhs, prhs)
4:      implicit none
5: !-----
6:      integer plhs(*), prhs(*)
7:      integer mxCreateDoubleMatrix
8:      integer contorno_pr,dim1_pr,imax_pr,maglia_pr
9:      integer U_max_pr,dETA0_dt_pr,CHI_pr,ETA0_pr
10:     integer soluzione_pr,iter_pr,toll_pr,z_pr
11:     integer nlhs, nrhs
12:     integer n1, n2, dim1, dim2, size_cont, imax
13:     integer mxGetM, mxGetN, mxGetPr, mxIsNumeric
14:     REAL*8,allocatable :: soluzione(:, :)
15:     REAL*8,allocatable :: contorno(:, :)
16:     REAL*8,allocatable :: z(:, :)
17:     real*8 dim1_t,imax_t,iter,maglia,U_max,dETA0_dt
18:     REAL*8 CHI,ETA0,toll
19: !-----
20: ! controllo che gli input siano in numero appropriato.
21:     if (nrhs .ne. 10) then
22:         call mexErrMsgTxt('10 inputs required.')
23:     endif
24: ! controllo che gli output siano in numero appropriato.
25:     if (nlhs .ne. 2) then
26:         call mexErrMsgTxt('2 output required.')
27:     endif
28: ! controllo che gli input siano matrici di numeri.
29:     if (mxIsNumeric(prhs(1)) .ne. 1) then
30:         call mexErrMsgTxt('Input #1 is not a numeric array.')
31:     endif
32:     if (mxIsNumeric(prhs(2)) .ne. 1) then
33:         call mexErrMsgTxt('Input #2 is not a numeric.')
34:     endif
35:     if (mxIsNumeric(prhs(3)) .ne. 1) then
36:         call mexErrMsgTxt('Input #3 is not a numeric.')
37:     endif
38:     if (mxIsNumeric(prhs(4)) .ne. 1) then
39:         call mexErrMsgTxt('Input #4 is not a numeric.')
40:     endif
41:     if (mxIsNumeric(prhs(5)) .ne. 1) then
42:         call mexErrMsgTxt('Input #5 is not a numeric.')
43:     endif
44:     if (mxIsNumeric(prhs(6)) .ne. 1) then
45:         call mexErrMsgTxt('Input #6 is not a numeric.')
46:     endif
47:     if (mxIsNumeric(prhs(7)) .ne. 1) then
48:         call mexErrMsgTxt('Input #7 is not a numeric.')
49:     endif
50:     if (mxIsNumeric(prhs(8)) .ne. 1) then
51:         call mexErrMsgTxt('Input #8 is not a numeric.')
52:     endif
53:     if (mxIsNumeric(prhs(9)) .ne. 1) then
54:         call mexErrMsgTxt('Input #9 is not a numeric.')
55:     endif
56:     if (mxIsNumeric(prhs(10)) .ne. 1) then
57:         call mexErrMsgTxt('Input #10 is not a numeric array.')
58:     endif
59:

```

```

60: ! rilevo le dimensioni degli input assegnati alla funzione
61:     n1 = mxGetM(prhs(1))
62:     n2 = mxGetN(prhs(1))
63:     allocate(contorno(n1,n2))
64:     allocate(soluzione(n1,n2))
65:     allocate(z(n1,n2))
66:     size_cont = n1*n2
67:
68: !     creo una matrice per i risultati di output.
69:     plhs(1) = mxCreateDoubleMatrix(n1, n2, 0)
70:     plhs(2) = mxCreateDoubleMatrix(1, 1, 0)
71:
72: !     assegno i parametri di input e output a delle variabili.
73:     contorno_pr = mxGetPr(prhs(1))
74:     dim1_pr = mxGetPr(prhs(2))
75:     imax_pr = mxGetPr(prhs(3))
76:     maglia_pr = mxGetPr(prhs(4))
77:     U_max_pr = mxGetPr(prhs(5))
78:     dETA0_dt_pr = mxGetPr(prhs(6))
79:     CHI_pr = mxGetPr(prhs(7))
80:     ETA0_pr = mxGetPr(prhs(8))
81:     toll_pr = mxGetPr(prhs(9))
82:     z_pr = mxGetPr(prhs(10))
83:
84:     soluzione_pr = mxGetPr(plhs(1))
85:     iter_pr = mxGetPr(plhs(2))
86:
87: !     carico i dati di input in variabili fortran.
88:     call mxCopyPtrToReal8(contorno_pr, contorno, size_cont)
89:     call mxCopyPtrToReal8(dim1_pr, dim1_t, 1)
90:     call mxCopyPtrToReal8(imax_pr, imax_t, 1)
91:     call mxCopyPtrToReal8(maglia_pr, maglia, 1)
92:     call mxCopyPtrToReal8(U_max_pr, U_max, 1)
93:     call mxCopyPtrToReal8(dETA0_dt_pr, dETA0_dt, 1)
94:     call mxCopyPtrToReal8(CHI_pr, CHI, 1)
95:     call mxCopyPtrToReal8(ETA0_pr, ETA0, 1)
96:     call mxCopyPtrToReal8(toll_pr, toll, 1)
97:     call mxCopyPtrToReal8(z_pr, z, size_cont)
98:
99:     dim1 = INT(dim1_t)
100:    imax = INT(imax_t)
101:    dim2 = dim1*3
102:
103: !     richiamo la computational subroutine.
104:    call telone(contorno,n1,n2,dim1,dim2,imax,maglia,U_max,
105: $ dETA0_dt,CHI,ETA0,soluzione,iter,toll,z)
106:
107: !     carico i dati di output nella MATLAB array.
108:    call mxCopyReal8ToPtr(soluzione, soluzione_pr, size_cont)
109:    call mxCopyReal8ToPtr(iter, iter_pr, 1)
110:
111:    deallocate(contorno)
112:    deallocate(soluzione)
113:    deallocate(z)
114:
115:    return
116:    contains
117:
118: !=====

```



```

119:      SUBROUTINE telone(contorno,n1,n2,dim1,dim2,imax,maglia,
120:      $ U_max,dETA0_dt,CHI,ETA0,soluzione,iter,toll,z)
121:      ! Questo il programma telone originale che non tien conto delle condizioni
122:      ! al CONTORNO nei punti canale, divenuto ora subroutine. Su tutti i punti
123:      ! canale l'elevazione posta pari a zero. L'unica modifica apportata consiste
124:      ! nel risolvere l'equazione '-laplaciano di eta = KAPPA', anzich =1
125:      ! *****
126:      ! LA CONVENZIONE SUI PUNTI DELLA LAGUNA E' LA SEGUENTE:
127:      ! 0 APPARTIENE ALLO SPECCHIO D'ACQUA
128:      ! 1 CONDIZIONE AL contorno DI DIRICHLET
129:      ! 2 CONDIZIONE AL contorno DI NEUMANN PER PUNTI NON CANALE
130:      ! 3 CONDIZIONE AL contorno DI NEUMANN PER PUNTI CANALE
131:      ! 4 PUNTI ANCORA CANALE MA DI INIZIO PER LA FUNZIONE DI AMPIEZZA
132:      ! *****
133:      !-----
134:      IMPLICIT NONE
135:      !----- Dichiarazione delle variabili -----
136:      integer n1,n2,dim1,n_bar,dim2
137:      real*8 :: contorno(:, :)
138:      real*8 :: soluzione(:, :)
139:      real*8 :: z(:, :)
140:      real*8 maglia,U_max,dETA0_dt,CHI,ETA0,iter
141:      REAL*8 diag,alfa,PI,a,zero
142:      parameter (zero=0.0)
143:      PARAMETER(PI = 3.141593) !valore della costante Pi_greco
144:      INTEGER imax ! del sistema lineare
145:      INTEGER k,nn,mm,i,j,m,ii,kk,k1,i1,j1,k2! indici dei cicli
146:      INTEGER n_ca,n_iat,n_diag,ind3 !
147:      real*8 toll, norm, beta, num, den,res,resmin
148:      real*8 normb
149:
150:      ! VARIABILI CON ALLOCAZIONE DINAMICA DELLO SPAZIO
151:      INTEGER,ALLOCATABLE :: incognite(:, :) ! matrice identificatrice dei
152:      ! punti barena (incognite)
153:      INTEGER,ALLOCATABLE :: iac(:)
154:      INTEGER,ALLOCATABLE :: iat(:)
155:      REAL*8,ALLOCATABLE :: ca(:)
156:      REAL*8,ALLOCATABLE :: b(:)
157:      ! xk = approssimazione del vettore incognito
158:      ! soluzione
159:      REAL*8,ALLOCATABLE :: xk(:)
160:      REAL*8,ALLOCATABLE :: R(:) ! R inizializzato dopo PROD
161:      REAL*8,ALLOCATABLE :: P(:) ! R inizializzato prima di KAPPA (primo)
162:      REAL*8,ALLOCATABLE :: KR(:) ! KR e' inizializzato in KAPPA (secondo)
163:      REAL*8,ALLOCATABLE :: Y(:) !Y e' inizializzato in KAPPA
164:      REAL*8,ALLOCATABLE :: CU(:) ! CU inizializzato qui sotto
165:      REAL*8,ALLOCATABLE :: AP(:) ! AP inizializzato in PROD
166:
167:
168:      INTEGER :: status
169:      ALLOCATE (incognite(n1,n2), STAT=status)
170:      ALLOCATE (xk(dim1), STAT=status)
171:      ALLOCATE (AP(dim1), STAT=status)
172:      ALLOCATE (iac(dim2),STAT=status)
173:      ALLOCATE (iat(dim1+1),STAT=status)
174:      ALLOCATE (ca(dim2),STAT=status)
175:      ALLOCATE (b(dim1),STAT=status)
176:      ALLOCATE (R(dim1),STAT=status)
177:      ALLOCATE (P(dim1),STAT=status)

```

```

178:      ALLOCATE (KR(dim1),STAT=status)
179:      ALLOCATE (Y(dim1),STAT=status)
180:      ALLOCATE (CU(dim2),STAT=status)
181:
182:      !----- Calcolo -----
183:      !costruisce per righe la matrice INCOGNITE. Agli elementi (I,J) della
184:      !matrice delle INCOGNITE che corrispondono agli elementi (I,J) nulli
185:      !della matrice CONTORNO, e' assegnato un numero crescente che identifica
186:      !l'incognita. Sono considerate incognite solo i punti con valore 0 nella
187:      !matrice CONTORNO e cioe' solo i punti della BARENA.
188:      n_bar=0          ! numero delle incognite, cio  numero dei punti barena
189:      DO j=1,n2
190:        DO i=1,n1
191:          IF(contorno(i,j)==0.) THEN
192:            n_bar=n_bar+1
193:            incognite(i,j)=n_bar
194:          ELSE
195:            incognite(i,j)=0.
196:          END IF
197:        END DO
198:      END DO
199:
200:      !----- costruzione della triangolare alta -----
201:      !Costruisce la triangolare alta (CA) della matrice delle incognite (INCOGNITE)
202:      n_ca=0
203:      n_iat=0
204:      DO j=1,n2
205:        DO i=1,n1
206:          IF(contorno(i,j).eq.0) THEN
207:
208:            n_ca=n_ca+1    ! C'E' UN ELEMENTO IN PIU' (ogni volta che trova 1
209:                          ! elemento del CONTORNO pari a zero, significa
210:                          ! che ha trovato una nuova incognita, cio lnuovo
211:                          ! elemento da aggiungere a CA e ovviamente a IAC)
212:            n_iat=n_iat+1 ! C'E' UNA RIGA IN PIU' NELLA MATRICE (ogni volta
213:                          ! che trova l'elemento del contorno pari a zero e cio
214:                          ! lnuova incognita significa che ha trovato lnuovo
215:                          ! elemento da aggiungere a IAT, ovvero una nuova riga
216:                          ! della matrice))
217:
218:            iat(n_iat)=n_ca ! SCRIVE il nuovo valore di iat
219:
220:            n_diag=n_ca    ! MEMORIZZA la posizione dell'elemento diagonale
221:            diag=4.
222:
223:            ! Controlla se i due elementi vicini che vengono visualizzati nella triangolare
224:            ! alta sono incognite. il controllo e' fatto solo sugli elementi posti a est e
225:            ! a sud dell'elemento (i,j) considerato.
226:            IF(contorno(i,j+1).eq.0) THEN
227:              n_ca=n_ca+1
228:              ca(n_ca)=-1.
229:              iac(n_ca)=incognite(i,j+1)
230:            ENDIF
231:
232:            IF(contorno(i+1,j).eq.0) THEN
233:              n_ca=n_ca+1
234:              ca(n_ca)=-1.
235:              iac(n_ca)=incognite(i+1,j)
236:            ENDIF

```

```

237:
238: ! Controlla se tra i quattro vicini ci sono elementi di contorno. in questo modo
239: ! semplifica l'equazione perche' se un vicino appartiene al bordo, si sfrutta la
240: ! condizione di flusso nullo e diag (che il coefficiente che moltiplica
241: ! l'incognita che sto considerando, e che vale al max 4.0) diminuisce di uno ad
242: ! ogni vicino incognito trovato.
243:
244:         IF(contorno(i-1,j).eq.1) THEN
245:             diag=diag-1
246:         ENDIF
247:
248:         IF(contorno(i+1,j).eq.1) THEN
249:             diag=diag-1
250:         ENDIF
251:
252:         IF(contorno(i,j-1).eq.1) THEN
253:             diag=diag-1
254:         ENDIF
255:
256:         IF(contorno(i,j+1).eq.1) THEN
257:             diag=diag-1
258:         ENDIF
259:
260:         ca(n_diag)=diag
261:         iac(n_diag)=incognite(i,j)
262:
263:     ENDIF
264:
265: END DO
266: END DO
267:
268: n_iat=n_iat+1
269: iat(n_iat)=n_ca+1
270:
271: !----- costruzione del vettore dei termini noti -----
272: ! Costruisce il vettore dei termini noti:
273: ! Attenzione alla z!
274: !=====
275: !inizialmente pongo z tutto nullo, poi si pu pensare di assegnarlo da fuori
276: !=====
277:     alfa = (maglia**2)*(8/(3*PI))*U_max*dETA0_dt/(CHI**2)
278:     DO j=1,n2
279:         DO i=1,n1
280:             IF(contorno(i,j).eq.0) THEN
281:                 IF((eta0-z(i,j)).gt.0) THEN
282:                     b(incognite(i,j))= alfa/(ETA0-z(i,j))**2
283:                 END IF
284:             ENDIF
285:         END DO
286:     END DO
287:
288: !=====
289: !----- Risoluzione del sistema lineare -----
290: !=====
291:     ! inizializzo le variabili per la soluzione del sistema
292:     ! lineare con il metodo del gradiente coniugato modificato
293:     normb=0.
294:     alfa=0.
295:

```

```

296:      do ii=1,n_bar
297:          normb=normb+(b(ii))**2
298:      enddo
299:      normb=SQRT(normb)
300:
301:      if(normb.eq.0) then
302:          do ii=1,n_bar
303:              xk(ii)=0.
304:          enddo
305:      endif
306:
307:  !=====
308:  !   calcolo della matrice di preconditionamento
309:  !   decomposta incompleta di Cholesky secondo Kershaw
310:  !   che ha la stessa topologia di CA
311:      do k=1,n_ca
312:          CU(k) = zero
313:      end do
314:
315:      do kk=1,dim1-1
316:
317:          k = iat(kk)
318:          a = ca(k) - CU(k)
319:          if(a.le.zero) then
320:              a = (CU(iat(kk-1)))**2
321:          end if
322:          CU(K) = sqrt(a)
323:
324:          i = iat(kk) + 1
325:          j = iat(kk+1) - 1
326:
327:
328:          do k1 = i,j
329:              CU(k1) = (ca(k1)-CU(k1))/CU(k)
330:          end do
331:
332:          do k2 = i,j-1
333:
334:              j1 = iat(iac(k2))
335:              CU(j1) = CU(j1) + CU(k2)**2
336:              i1 = k2 + 1
337:              j1 = j1 + 1
338:              do while(j1.lt.iat(iac(k2)+1).and.i1.le.j)
339:                  if(iac(j1).eq.iac(i1)) then
340:                      CU(j1) = CU(j1) + CU(k2)*CU(i1)
341:                      i1 = i1 + 1
342:                      j1 = j1 + 1
343:                  else if (iac(j1).lt.iac(i1))then
344:                      j1 = j1 + 1
345:                  else if (iac(j1).gt.iac(i1))then
346:                      i1 = i1 + 1
347:                  end if
348:              end do
349:
350:          end do
351:
352:          if(j.ge.i) CU(iat(iac(j)))=CU(iat(iac(j)))+CU(k2)**2
353:
354:      end do

```

```

355:
356:     k = iat(dim1)
357:     a = ca(k)-CU(k)
358:     if(a.le.zero) then
359:         a = (CU(iat(dim1-1)))**2
360:     end if
361:     CU(k) = sqrt(a)
362: !=====
363: !   inizializzo la soluzione xk(0)=K^-1*b
364:     do k = 1,dim1
365:         xk(k) = zero
366:     end do
367:     do k = 1,dim1
368:         i = iat(k)
369:         j = iat(k+1) - 1
370:         xk(k) = (b(k) - xk(k))/CU(i)
371:         do m = i+1,j
372:             xk(iac(m)) = xk(iac(m)) + CU(m)*xk(k)
373:         end do
374:     end do
375:     do k = 1,dim1
376:         nn = dim1-k+1
377:         a = zero
378:         i = iat(nn)
379:         j = iat(nn+1) - 1
380:         do m = i,j-1
381:             mm = j-m+i
382:             a = a + CU(mm)*xk(iac(mm))
383:         end do
384:         xk(nn) = (xk(nn) - a)/CU(i)
385:     end do
386: !=====
387: !   inizializzo lo scarto R(0) = b - A*x(1)
388:     AP = 0.0
389:     DO i =1,dim1
390:         ind3 = iat(i)
391:         AP(i) = AP(i) + ca(ind3)*xk(iac(ind3))
392:         DO j = ind3+1, iat(i+1)-1
393:             AP(i) = AP(i) + ca(j)*xk(iac(j))
394:             AP(iac(j)) = AP(iac(j)) + ca(j)*xk(i)
395:         END DO
396:     END DO
397: !=====
398:     R = b - AP
399:
400: !   P(0) = K^-1*R
401:     P=0.
402: !=====
403:     do k = 1,dim1
404:         P(k) = zero
405:     end do
406:     do k = 1,dim1
407:         i = iat(k)
408:         j = iat(k+1) - 1
409:         P(k) = (R(k) - P(k))/CU(i)
410:         do m = i+1,j
411:             P(iac(m)) = P(iac(m)) + CU(m)*P(k)
412:         end do
413:     end do

```

```

414:      do k = 1,dim1
415:          nn = dim1-k+1
416:          a = zero
417:          i = iat(nn)
418:          j = iat(nn+1) - 1
419:          do m = i,j-1
420:              mm = j-m+i
421:              a = a + CU(mm)*P(iac(mm))
422:          end do
423:          P(nn) = (P(nn) - a)/CU(i)
424:      end do
425:  !=====
426:  !   inizializzo le variabili rimanenti
427:      res = 2.*toll
428:      iter = 0
429:      beta = 0.0
430:      alfa = 0.0
431:      resmin=1.e+05
432:
433:  !   inizio il ciclo risolutivo
434:      DO WHILE ( (ABS(res) .gt. toll) .and. (iter .lt. imax) )
435:          num = 0.0
436:          den = 0.0
437:  !   Inizia il calcolo di alfa
438:  !   Trovo A*P(k)
439:  !=====
440:          AP = 0.0 ! inizializzo
441:          DO i =1,dim1
442:              ind3 = iat(i)
443:              AP(i) = AP(i) + ca(ind3)*P(iac(ind3))
444:              DO j = ind3+1, iat(i+1)-1
445:                  AP(i) = AP(i) + ca(j)*P(iac(j))
446:                  AP(iac(j)) = AP(iac(j)) + ca(j)*P(i)
447:              END DO
448:          END DO
449:  !=====
450:  !   Calcolo numeratore (P(k)*R(k)) e denominatore (P(k)*AP(k))
451:          DO ii = 1,n_bar
452:              num = num + P(ii)*R(ii)
453:              den = den + P(ii)*AP(ii)
454:          ENDDO
455:          alfa = num / den
456:  !   Calcolo l' iterata successiva: x(k+1) = x(k) + alfa(k)*P(k)
457:          DO ii = 1,n_bar
458:              xk(ii) = xk(ii) + alfa*P(ii)
459:          END DO
460:  !   R(k+1) = R(k) - alfa(k)*AP(k)
461:          DO ii = 1, n_bar
462:              R(ii) = R(ii) - alfa*AP(ii)
463:          END DO
464:  !   Inizia il calcolo di beta
465:  !   Trovo K^-1 * R(k+1)
466:  !=====
467:          do k = 1,dim1
468:              KR(k) = zero
469:          end do
470:          do k = 1,dim1
471:              i = iat(k)
472:              j = iat(k+1) - 1

```

```

473:         KR(k) = (R(k) - KR(k))/CU(i)
474:         do m = i+1,j
475:             KR(iac(m)) = KR(iac(m)) + CU(m)*KR(k)
476:         end do
477:     end do
478:     do k = 1,dim1
479:         nn = dim1-k+1
480:         a = zero
481:         i = iat(nn)
482:         j = iat(nn+1) - 1
483:         do m = i,j-1
484:             mm = j-m+i
485:             a = a + CU(mm)*KR(iac(mm))
486:         end do
487:         KR(nn) = (KR(nn) - a)/CU(i)
488:     end do
489: !=====
490: !     numeratore      (K^-1*R(k+1))*(AP(k))
491: !     denominatore   (P(k))*(AP(k))
492:     num = 0.0
493:     DO ii = 1,n_bar
494:         num = num + KR(ii)*AP(ii)
495:     END DO
496:     beta = - num / den
497: !     Delta x = x(k+1) - x(k)
498: !     P(k+1) = K^-1*R(k+1) + beta(k)*P(k)
499: !     x(corrente) = x(k+1)
500:     DO ii = 1,n_bar
501:         P(ii) = KR(ii) + beta*P(ii)
502:     END DO
503:     norm=0.
504:     do ii=1,n_bar
505:         norm=norm+(AP(ii))**2.
506:     enddo
507:     norm=norm*alfa**2.
508:     if (norm.ne.0.) then
509:         norm=SQRT(norm)
510:     else
511:         norm=0.
512:     endif
513:     res=0.
514:     do ii=1,n_bar
515:         res=res+R(ii)**2.
516:     enddo
517:     res=SQRT(res)
518:     res=res/normb ! residuo relativo
519:
520: !     check
521: !     if(res.LT.resmin) resmin=res !perch ??
522:     if (xk(1).GT.1.E+03) iter=imax-1
523:     iter = iter + 1
524:     END DO
525:
526: !=====
527: !----- Costruzione della soluzione -----
528: ! Costruisce la matrice con la soluzione
529:     DO j=1,n2
530:         DO i=1,n1
531:             IF(contorno(i,j).eq.0) THEN

```

```
532:         soluzione(i,j)=xk(incognite(i,j))
533:         END IF
534:         IF(contorno(i,j).ge.2) THEN
535:         soluzione(i,j)=0.
536:         END IF
537:         IF(contorno(i,j).eq.1) THEN
538:         soluzione(i,j)=0.
539:         END IF
540:     END DO
541: END DO
542: !=====
543: !----- libero lo spazio utilizzato -----
544: !=====
545:     DEALLOCATE (incognite, STAT=status)
546:     DEALLOCATE (AP, STAT=status)
547:     DEALLOCATE (iac,STAT=status)
548:     DEALLOCATE (iat,STAT=status)
549:     DEALLOCATE (ca,STAT=status)
550:     DEALLOCATE (b,STAT=status)
551:     DEALLOCATE (xk,STAT=status)
552:     DEALLOCATE (R,STAT=status)
553:     DEALLOCATE (P,STAT=status)
554:     DEALLOCATE (KR,STAT=status)
555:     DEALLOCATE (Y,STAT=status)
556:     DEALLOCATE (CU,STAT=status)
557:
558:     RETURN
559: END subroutine telone
560: !=====
561: end subroutine mexfunction
562:
```



```

1: % [incidenza_bar,portata_bar,sorgenti_bar]=incidenzabarene(contorno,soluzione);
2: %=====
3: % FUNZIONE CHE TROVA LE DIREZIONI DI DRENAGGIO SULLA BARENA
4: % Data la matrice contenente la GEOMETRIA della rete (CONTORNO) e data la matr-
5: % ice contenente l'andamento del PELO LIBERO (SOLUZIONE) su tutti i punti della
6: % barena, questa funzione calcola:
7: % 1)la MATRICE 'incidenza_bar';
8: % 2)la MATRICE 'portata_bar';
9: % 3)la MATRICE 'sorgenti_bar'.
10: %=====
11: % INPUT:
12: % - contorno: (INPUT OBBLIGATORIO) matrice contenente le caratteristiche
13: % dell'area di studio (i.e. matrice delle condizioni al contorno):
14: % LA CONVENZIONE SUI PUNTI DELLA LAGUNA E' LA SEGUENTE:
15: % 0 APPARTIENE ALLO SPECCHIO D'ACQUA
16: % 1 CONDIZIONE AL contorno DI DIRICHLET
17: % 2 CONDIZIONE AL contorno DI NEUMANN PER PUNTI NON CANALE
18: % 3 CONDIZIONE AL contorno DI NEUMANN PER PUNTI CANALE
19: % 4 PUNTI ANCORA CANALE MA DI INIZIO PER LA FUNZIONE DI AMPIEZZA
20: % - soluzione: (INPUT OBBLIGATORIO) matrice che contiene l'andamento della
21: % superficie libera calcolato dalla funzione che esegue il
22: % modello del telone.
23: %
24: % OUTPUT(OBBLIGATORI):
25: % - incidenza_bar: matrice contenente le direzioni di drenaggio x tutti i pun-
26: % ti della barena, ottenute seguendo la direzione del massimo
27: % gradiente del pelo libero:
28: % - portata_bar: matrice che cumula secondo le direzioni di drenaggio e si
29: % ferma non appena incontra un canale, dando cos , per ogni
30: % pixel appartenente alle barene, il numero di pixel a monte
31: % dello stesso che drenano verso questo;
32: % - sorgenti_bar: per tutti i punti della barena assegna valore 0 nei puntiti
33: % sorgenti, cio nei pixel che non ricevono contributi da nes-
34: % suno dei vicini, e 1 nei pixel che ricevono i contributi dei
35: % pixel vicini.
36: %
37: % NOTA: questa funzione composta in parte da codice matlab e in parte da un
38: % mex_file. Eventuali errori nella computazione della funzione
39: % vengono segnalati a video: in particolare, se ci sono pixel canale
40: % alla quale viene assegnata una direzione di drenaggio,
41: % questi vengono individuati e reimpostati pari a 0.
42: %
43: %=====
44: % PER L'ESECUZIONE E' RICHIESTA LA TOOLBOX TELONE_TBOX
45: %=====
46: % data di creazione: 19/05/2014
47: % ultima modifica: 17/06/2014
48: % autore: Francesco Carraro
49: %=====
50: function [incidenza_bar,portata_bar,sorgenti_bar,L_bar]=...
51:         incidenzabarene(contorno,soluzione);
52: % calcolo il numero di pixel barena nella matrice contorno
53: n_marsh = length(contorno(contorno==0));
54:
55: tic;
56: % si richiama il mex_file creato tramite F_incibarene.f90
57: [incidenza_bar,sorgenti_bar,portata_bar,L_bar,err_bar]=...
58:         F_incibarene(contorno,soluzione,n_marsh);
59: t = toc;
    
```

```
60:
61: fprintf(...)
62: 'l''elaboraz. di incidenzabarene e'' andata a buon fine in %4.1g secondi\n',t)
63:
64: % si scrivono a video eventuali errori che sono stati riscontrati nel calcolo
65: if err_bar > 0
66:     fprintf(' ===== SONO STATI TROVATI ALCUNI ERRORI =====\n')
67:     fprintf(' ## NB:ci sono %4g punti canale in cui e'' stata calcolata ###\n'...
68:     ,err_bar)
69:     fprintf(' ##     erroneamente una direzione di drenaggio e in cui   ###\n')
70:     fprintf(' ##     stato poi reimposto valore 0                       ###\n')
71:     fprintf(' =====\n')
72:     fprintf('PREMERE [INVIO] PER CONTINUARE ([ctrl+C] per uscire)\n')
73:     pause
74: end
75:
```

```

1: !      The gateway routine: ovvero la subroutine che lega gli oggetti tipici
2: !      di matlab con le classiche variabili integer e real
3:      subroutine mexFunction(nlhs, plhs, nrhs, prhs)
4:      implicit none
5: !-----
6:      integer plhs(*), prhs(*)
7:      integer mxCreateDoubleMatrix
8:      integer contorno_pr, lunghezza_pr, bar_err_pr
9:      integer incidenza_bar_pr, portata_bar_pr, sorgenti_bar_pr
10:     integer n_marsh_pr, soluzione_pr
11:     integer nlhs, nrhs
12:     integer n1, n2, n_marsh, size_cont
13:     integer mxGetM, mxGetN, mxGetPr, mxIsNumeric
14:     REAL*8,allocatable :: contorno(:, :)
15:     REAL*8,allocatable :: incidenza_bar(:, :)
16:     REAL*8,allocatable :: sorgenti_bar(:, :)
17:     REAL*8,allocatable :: portata_bar(:, :)
18:     REAL*8,allocatable :: soluzione(:, :)
19:     REAL*8,allocatable :: lunghezza(:, :)
20:     real*8 n_marsh_t
21:     REAL*8 bar_err
22: !-----
23: ! controllo che gli input siano in numero appropriato.
24:     if (nrhs .ne. 3) then
25:         call mexErrMsgTxt('Four inputs required.')
26:     endif
27: ! controllo che gli output siano in numero appropriato.
28:     if (nlhs .ne. 5) then
29:         call mexErrMsgTxt('Four output required.')
30:     endif
31: ! controllo che gli input siano matrici di numeri.
32:     if (mxIsNumeric(prhs(1)) .ne. 1) then
33:         call mexErrMsgTxt('Input #1 is not a numeric array.')
34:     endif
35:     if (mxIsNumeric(prhs(2)) .ne. 1) then
36:         call mexErrMsgTxt('Input #2 is not a numeric.')
37:     endif
38:     if (mxIsNumeric(prhs(3)) .ne. 1) then
39:         call mexErrMsgTxt('Input #3 is not a numeric array.')
40:     endif
41:
42: ! rilevo le dimensioni degli input assegnati alla funzione
43:     n1 = mxGetM(prhs(1))
44:     n2 = mxGetN(prhs(1))
45:     size_cont = n1*n2
46:
47:     allocate(incidenza_bar(n1,n2))
48:     allocate(soluzione(n1,n2))
49:     allocate(lunghezza(n1,n2))
50:     allocate(contorno(n1,n2))
51:     allocate(sorgenti_bar(n1,n2))
52:     allocate(portata_bar(n1,n2))
53:
54: C      creo una matrice per i risultati di output.
55:     plhs(1) = mxCreateDoubleMatrix(n1, n2, 0)
56:     plhs(2) = mxCreateDoubleMatrix(n1, n2, 0)
57:     plhs(3) = mxCreateDoubleMatrix(n1, n2, 0)
58:     plhs(4) = mxCreateDoubleMatrix(n1, n2, 0)
59:     plhs(5) = mxCreateDoubleMatrix(1, 1, 0)

```

```

60:
61: C   assegno i parametri di input e output a delle variabili.
62:     contorno_pr = mxGetPr(prhs(1))
63:     soluzione_pr = mxGetPr(prhs(2))
64:     n_marsh_pr = mxGetPr(prhs(3))
65:
66:     incidenza_bar_pr = mxGetPr(plhs(1))
67:     sorgenti_bar_pr = mxGetPr(plhs(2))
68:     portata_bar_pr = mxGetPr(plhs(3))
69:     lunghezza_pr = mxGetPr(plhs(4))
70:     bar_err_pr = mxGetPr(plhs(5))
71:
72: C   carico i dati di input in variabili fortran.
73:     call mxCopyPtrToReal8(contorno_pr, contorno, size_cont)
74:     call mxCopyPtrToReal8(n_marsh_pr, n_marsh_t, 1)
75:     call mxCopyPtrToReal8(soluzione_pr, soluzione, size_cont)
76:
77:     n_marsh = INT(n_marsh_t)
78:
79: C   richiamo la computational subroutine.
80:     call inci_barene(n1,n2,contorno,soluzione,n_marsh,
81: $incidenza_bar,sorgenti_bar,portata_bar,lunghezza,bar_err)
82:
83: C   carico i dati di output nella MATLAB array.
84:     call mxCopyReal8ToPtr(incidenza_bar,incidenza_bar_pr,
85: $size_cont)
86:     call mxCopyReal8ToPtr(sorgenti_bar,sorgenti_bar_pr,
87: $size_cont)
88:     call mxCopyReal8ToPtr(portata_bar,portata_bar_pr,
89: $size_cont)
90:     call mxCopyReal8ToPtr(lunghezza, lunghezza_pr, size_cont)
91:     call mxCopyReal8ToPtr(bar_err, bar_err_pr, 1)
92:
93:     deallocate(incidenza_bar)
94:     deallocate(soluzione)
95:     deallocate(lunghezza)
96:     deallocate(contorno)
97:     deallocate(sorgenti_bar)
98:     deallocate(portata_bar)
99:
100:    return
101:    contains
102: ! =====
103: ! ===== SUBROUTINE LARGH =====
104: ! Calcola la LARGHEZZA dei canali e la scrive per sezioni
105: ! =====
106:    ! Computational subroutine
107:    SUBROUTINE inci_barene(n1,n2,contorno,soluzione,n_marsh,
108: $incidenza_bar,sorgenti_bar,portata_bar,lunghezza,bar_err)
109: ! Data la matrice contenente la GEOMETRIA della rete (CONTORNO) e data la
110: ! matrice contenente l'andamento del PELO LIBERO (SOLUZIONE) su tutti i punti
111: ! della barena, questa subroutine calcola:
112: ! 1)la MATRICE 'incidenza_bar'(contenente le direzioni di drenaggio x tutti i
113: ! punti della barena, ottenute seguendo la direzione del massimo gradiente
114: ! del pelo libero);
115: ! 2)la MATRICE 'portata_bar' (che cumula secondo le direzioni di drenaggio e si
116: ! ferma non appena incontra un canale, dando cos , per ogni pixel
117: ! appartenente alle barene, il numero di pixel a monte dello stesso che
118: ! drenano verso questo);

```

```

119: ! 3)la MATRICE delle 'sorgenti_bar' x tutti i punti della barena (che vale 0
120: ! nei pti sorgenti, cioè nei pixel che non ricevono contributi da nessuno dei
121: ! vicini, e 1 nei pixel che ricevono i contributi dei pixel vicini).
122: !-----
123:      IMPLICIT NONE
124: !----- Dichiarazione delle variabili -----
125:      REAL*8 diri(8),dirj(8)          ! Puntatore delle direzioni
126:      REAL*8 slope(8),cslope(8),area,ran
127:      REAL*8 :: lung, somma, media
128:      INTEGER :: kmin(1),j,i,k,il,jl,m_1,n_1,m
129:      INTEGER :: rnd,krand,kminimo
130:      INTEGER :: count
131:      REAL*8 bar_err          ! Variabile indicatrice dell'errore
132:      INTEGER n1,n2,n_marsh,n_bar
133:      REAL*8 :: incidenza_bar(:, :)
134:      REAL*8 :: sorgenti_bar(:, :)
135:      REAL*8 :: portata_bar(:, :)
136:      REAL*8 :: contorno(:, :)
137:      REAL*8 :: soluzione(:, :)
138:      REAL*8 :: lunghezza(:, :)
139:      integer, allocatable :: xmarsh(:)
140:      integer, allocatable :: ymarsh(:)
141:      REAL*8, allocatable :: vettorelung(:) ! Vettore che contiene i valori delle
142:      ! lunghezze fuori rete per tutti i punti della barena
143:
144: !----- Inizializzazione delle variabili -----
145:      allocate(vettorelung(n_marsh))
146:      allocate(xmarsh(n_marsh))
147:      allocate(ymarsh(n_marsh))
148:
149:      bar_err = 0.
150:      incidenza_bar = 0. !Inizializzazione della matrice di incidenza barene
151:      sorgenti_bar = 0. !Inizializzazione della matrice delle sorgenti barene
152:      portata_bar = 0. !Inizializzazione della matrice delle portate delle
153:      ! barene che x ogni pixel di barena contiene il numero
154:      ! di pixel drenati da questo pixel
155:      lunghezza = 0. !Inizializzazione della matrice lunghezze fuori rete
156:      vettorelung = 0. !Inizializzazione del vettore delle lunghezze fuori rete
157:      xmarsh = 0
158:      ymarsh = 0
159:      ! Puntatore delle direzioni
160:      DATA diri /0,-1,-1,-1,0,1,1,1/ ! Puntatore delle direzioni lungo "i"
161:      DATA dirj /1,1,0,-1,-1,-1,0,1/ ! Puntatore delle direzioni lungo "j"
162:
163:      n_bar = 0
164:      do i=1,n1
165:         do j=1,n2
166:            if(contorno(i,j).eq.0) then
167:               n_bar = n_bar+1
168:               xmarsh(n_bar) = j
169:               ymarsh(n_bar) = i
170:            endif
171:         enddo
172:      enddo
173:
174: !----- Calcolo -----
175: ! Calcola 'INCIDENZA_BAR' e 'SORGENTI_BAR'
176: ! Calcola la MATRICE 'incidenza_bar' x tutti i punti della barena: calcola cioè
177: ! la direzione di drenaggio x ciascun pixel. Quando trova un punto che

```

```

178: ! appartiene al contorno impermeabile esce dal ciclo. Poiche' la direzione di
179: ! drenaggio coincide con la direzione del massimo gradiente del pelo libero,
180: ! x ogni pixel calcola il gradiente della superficie libera considerando gli
181: ! otto vicini e individua dove qto pixel scarica la propria portata.
182:   loop1:DO m=1,n_marsh ! esegue il ciclo per tutti i punti della barena
183:     j = xmarsh(m) ! memorizza le coordinate dei
184:     i = ymarsh(m) ! pixel barena in 'i' e 'j'
185:
186:     DO k=1,8 !inizia il ciclo in tutte le direzioni cioe' su
187:       !ogniuno degli 8 vicini del pixel considerato.
188: ! per tutti i pixels calcola la il gradiente della superficie libera
189:   IF((contorno(i+diri(k),j+dirj(k))).ne.1)THEN
190:     slope(k)=(soluzione(i,j)-soluzione(i+diri(k),j+
191: $dirj(k)))/SQRT(ABS(diri(k))+ABS(dirj(k)))
192:   ELSE ! se uno dei vicini e' un pixel in cui e' imposta la
193:     slope(k)=0 ! condizione di flusso nullo pone pari a 0 il
194:   END IF ! gradiente del pelo libero
195:           ! (come fisicamente è giusto che sia)
196:
197:   END DO
198: ! Trasla il vettore delle pendenze di un numero casuale da 1 a 8 e trova la
199: ! posizione minima in questo vettore. In questo modo se ci sono piu' di
200: ! due minimi, ne sceglie uno a caso
201:   rnd=rnd+1
202:   krand=INT(RAN(rnd)*8)
203:   cslope=CSHIFT(slope,krand)
204:   kmin=MAXLOC(cslope)
205:   kminimo=kmin(1)
206:   kminimo=MOD(kmin(1)-1+krand,8)+1
207:
208:   incidenza_bar(i,j)=kminimo
209:   sorgenti_bar(i+diri(kminimo),j+dirj(kminimo))= 1
210:
211:   END DO loop1
212: !-----
213: ! Calcola 'PORTATE_BAR'
214: ! Parte da una SORGENTE (dove cioe' la matrice 'incidenza_bar' e' nulla) e segue
215: ! il percorso di massimo gradiente assegnando le portate (assegnando cioe' le
216: ! aree drenate da ciascun pixel).
217:   loop2:DO m=1,n_marsh ! esegue il ciclo per tutti i punti della barena
218:     j = xmarsh(m) ! memorizza le coordinate dei
219:     i = ymarsh(m) ! pixel barena in 'i' e 'j'
220:     !write(*,*) i, j
221:     ! ad ogni punto di barena che sia un punto sorgente
222:     IF ((sorgenti_bar(i,j).eq.0).and.(contorno(i,j).eq.0)
223: $.and.(soluzione(i,j).ne.0)) THEN
224:
225:       area=1
226:       portata_bar(i,j)=area ! assegna area pari a 1
227:       il=i ! memorizza le coordinate del punto
228:       jl=j
229:           ! va verso il pixel in cui drena il
230: 222 m_1=diri(incidenza_bar(il,jl))! pixel (i,j)memorizza il valore dei
231: n_1=dirj(incidenza_bar(il,jl))! puntatori diri e dirj relativi
232: ! al pixel (i,j)
233:
234:       il=il+m_1 ! va verso il pixel (il+m_1,jl+n_1)
235:       jl=jl+n_1
236: ! se questo pixel non e' un pixel canale e non ha ancora ricevuto i contributi

```

```

237: ! dei pixels upstream ad esso connessi, allora incrementa area di 1
238:         IF((portata_bar(il,jl).eq.0).and.(contorno(il,jl)
239:         $.ne.3).and.(contorno(il,jl).ne.4)) THEN
240:             area=area+1
241:             END IF
242:
243:         portata_bar(il,jl)=portata_bar(il,jl)+area
244:         IF(incidenza_bar(il,jl).ne.0) GOTO 222
245:     END IF
246: END DO loop2
247: !-----
248: ! CONTROLLO
249: ! Controlla di non aver attribuito valori di incidenza_bar a pixel
250: ! che non appartengono alla barena (cio a pixel tali che contorno(i,j).ne.0)
251:     DO i=1,n1
252:         DO j=1,n2
253:             IF ((contorno(i,j).ne.0).and.(incidenza_bar(i,j).ne.0))
254: $THEN
255:                 bar_err=bar_err+1
256:                 incidenza_bar(i,j) = 0.
257:             END IF
258:         END DO
259:     END DO
260: !-----
261: ! Calcola la matrice delle 'LUNGHEZZE' fuori rete
262:     loop4:DO m=1,n_marsh ! esegue il ciclo per tutti i punti della barena
263:         j = xmarsh(m) ! memorizza le coordinate dei
264:         i = ymarsh(m) ! pixel barena in 'i' e 'j'
265:         ! x ogni punto di barena
266:         IF ((incidenza_bar(i,j) /= 0).and.(contorno(i,j) == 0)
267: $) THEN
268:
269:             lung=0. ! assegna lunghezza pari a 0.
270:             il=i ! memorizza le coordinate del punto
271:             jl=j
272:
273:             ! va verso il pixel in cui drena il
274:             m_1=diri(incidenza_bar(il,jl))! pixel (i,j) memorizza il valore dei
275:             n_1=dirj(incidenza_bar(il,jl))! puntatori diri e dirj relativi al
276:             ! pixel (i,j)
277:             lung=lung+SQRT(FLOAT(m_1**2+n_1**2)) !assegna il valore a lunghezza
278:
279:             il=il+m_1 ! va verso il pixel (il+m_1,jl+n_1)
280:             jl=jl+n_1
281:
282:             IF((incidenza_bar(il,jl).ne.0).and.(contorno(il,jl)
283: $< 3)) GOTO 223
284:                 lung=lung+lunghezza(il,jl)
285:                 lunghezza(i,j)=lung
286:             END IF
287:         END DO loop4
288:
289:         deallocate(vettorelung)
290:         deallocate(xmarsh)
291:         deallocate(ymarsh)
292:         RETURN
293: !-----
294:     end subroutine inci_barene
295: end subroutine mexfunction

```



```

1: % [width,width_sez]=ampiezza(contorno,n_canali,n_bocche,passo_sezione);
2: %=====
3: % FUNZIONE PER IL CALCOLO DELLA FUNZIONE DI AMPIEZZA, o WIDTH FUNCTION:
4: % per ogni pixel della rete di canali calcola la sua distanza dalla bocca,
5: % seguendo la rete.
6: %=====
7: % INPUT:
8: %   - contorno: (INPUT OBBLIGATORIO) matrice contenente le caratteristiche
9: %   dell'area di studio (i.e. matrice delle condizioni al contorno):
10: %   LA CONVENZIONE SUI PUNTI DELLA LAGUNA E' LA SEGUENTE:
11: %   0 APPARTIENE ALLO SPECCHIO D'ACQUA
12: %   1 CONDIZIONE AL contorno DI DIRICHLET
13: %   2 CONDIZIONE AL contorno DI NEUMANN PER PUNTI NON CANALE
14: %   3 CONDIZIONE AL contorno DI NEUMANN PER PUNTI CANALE
15: %   4 PUNTI ANCORA CANALE MA DI INIZIO PER LA FUNZIONE DI AMPIEZZA
16: %   - n_canali: (INPUT FACOLTATIVO) n di pixel canale presenti nella matrice
17: %   contorno, se non specificato la funzione lo calcola autonomamente
18: %   cercando gli elementi della matrice che contengono numeri maggi-
19: %   ori o uguali a 3.
20: %   - n_bocche: (INPUT FACOLTATIVO) n di pixel inlet-canale presenti nella mat-
21: %   rice contorno, se non specificato la funzione lo calcola autonom-
22: %   amente cercando gli elementi della matrice che contengono il 4.
23: %   - passo_sezione: permette di definire le sezioni dei canali. Se considero la
24: %   direzione dell'asse, "passo_sezione" pixels in quella direzione
25: %   formano una sezione; (INPUT FACOLTATIVO) se non assegnato il
26: %   valore di default posto pari a 1.
27: %
28: % OUTPUT(OBBLIGATORI):
29: %   - width: matrice contenente la funzione ampiezza, per ogni pixel della rete
30: %   di canali riporta la distanza dalla bocca.
31: %   - width_sez: matrice che definisce la funzione AMPIEZZA per sezioni, in base
32: %   al passo scelto.
33: %
34: % NOTA1: e' possibile assegnare i parametri di input in tre modi diversi:
35: %   - assegnando solo la matrice 'contorno' la funzione calcolera'
36: %   autonomamente i valori di default per gli altri parametri e li
37: %   scrivera' a video chiedendo conferma all'utente;
38: %   - assegnando tutti i parametri richiesti RISPETTANDO STRETTAMENTE
39: %   L'ORDINE (contorno,n_canali,n_bocche,passo_sezione), la funzione
40: %   eseguirà il calcolo segnalando a video solo eventuali errori.
41: %
42: %   !!!NB:E' FONDAMENTALE IN QUESTO CASO IL RISPETTO DELL'ORDINE.!!!
43: %
44: %   - assegnando solo alcuni dei parametri facoltativi specificandoli,
45: %   dopo l'assegnazione della matrice 'contorno', in qualsiasi ordine
46: %   assegnandoli dopo aver posto tra apici il nome del parametro che si
47: %   desidera imporre.
48: %   ESEMPIO. Se si vuole imporre solo 'passo_sezione' e' possibile farlo
49: %   col comando:
50: %       [w,w_sez]=ampiezza(cont,'passo_sezione',p_sez);
51: %   In questo modo la funzione scrivera' ugualmente a video i parametri
52: %   facoltativi e chiederà conferma all'utente prima di eseguire il
53: %   calcolo.
54: %
55: % NOTA2: questa funzione e' composta in parte da codice matlab e in parte da un
56: %   mex_file. Nel caso ci fossero errori nella computazione della funzione
57: %   ampiezza, questi vengono segnalati a video: in particolare, se ci sono
58: %   pixel canale alla quale non viene assegnato alcun valore di ampiezza,
59: %   questi vengono individuati in width da un penalty negativo[-999].

```

```

60: %
61: %=====
62: % PER L'ESECUZIONE E' RICHIESTA LA TOOLBOX TELONE_TBOX
63: %=====
64: % data di creazione: 21/03/2014
65: % ultima modifica: 17/06/2014
66: % autore: Francesco Carraro
67: %=====
68: function [width,width_sez]= ampiezza(contorno,varargin);
69: % creo la struttura di input: preparo i parametri da dare alla mex_function,
70: %                                impostando i valori di default per i parametri
71: %                                opzionali.
72: % creo la struttura
73: in = inputParser;
74:     % scelgo i valori di default
75:     defaultN_canali = length(contorno(contorno>=3));
76:     defaultN_bocche = length(contorno(contorno==4));
77:     defaultP_sezione = 1;
78:     % creo il parametro obbligatorio
79:     addRequired(in,'contorno',@isnumeric);
80:     % creo i parametri facoltativi
81:     addOptional(in,'n_canali',defaultN_canali,@isnumeric);
82:     addOptional(in,'n_bocche',defaultN_bocche,@isnumeric);
83:     addOptional(in,'passo_sezione',defaultP_sezione,@isnumeric);
84:     % creo il file struttura risultante con gli input corretti per il mex
85:     parse(in,contorno,varargin{:});
86:
87: % si esegue un controllo su n_canali e n_inlet
88: if in.Results.n_canali < in.Results.n_bocche
89:     fprintf(' \nERRORE NELL'ASSEGNAZIONE DEI PARAMETRI OPZIONALI\n')
90:     fprintf('Ci sono meno pixel canale di quanti non siano quelli degli inlet!\n')
91:     pause
92:     width = -1000;
93:     width_sez = -1000;
94:     return
95: end
96:
97: % nel caso non siano stati assegnati tutti i parametri richiesti, scrivo a video
98: % i valori che la funzione ha scelto per l'esecuzione della mex_function
99: num = 0;
100: for iii=1:length(varargin),num=num+isnumeric(varargin{iii});end
101: if (length(varargin) < 6) && num < 3
102:     fprintf(' \nMANCANO ALCUNI POSSIBILI INPUT OPZIONALI!\n')
103:     fprintf('Per l''esecuzione della funzione verranno utilizzati i seguenti\n')
104:     fprintf('valori delle variabili opzionali:\n')
105:     fprintf('n_canali = %d;\n',in.Results.n_canali)
106:     fprintf('n_bocche = %d;\n',in.Results.n_bocche)
107:     fprintf('passo_sezione = %d.\n\n',in.Results.passo_sezione)
108:     fprintf('Per ulteriori informazioni digitare a linea di comando:\n')
109:     fprintf('help ampiezza\n')
110:     fprintf('PREMERE [INVIO] PER CONTINUARE ([ctrl+C] per uscire)\n')
111:     pause
112:     fprintf('...\n')
113:     fprintf('ESECUZIONE DELLA FUNZIONE ampiezza IN CORSO\n')
114: end
115:
116: tic;
117: % si richiama il mex_file creato tramite F_ampiezza.f90
118: [width,width_sez,w_err,ww_err]=F_ampiezza(in.Results.contorno,...

```

```
119:                                     in.Results.n_canali,...
120:                                     in.Results.n_bocche,...
121:                                     in.Results.passo_sezione);
122: t = toc;
123: fprintf(' l''elaboraz. di ampiezza e'' andata a buon fine in %4.1g secondi\n',t)
124:
125: % si scrivono a video eventuali errori che sono stati riscontrati nel calcolo
126: if w_err > 0 || ww_err >0
127:     fprintf(' ===== SONO STATI TROVATI ALCUNI ERRORI =====\n')
128:     fprintf(' - sono stati corretti %d punti non canale in cui era stato\n',w_err)
129:     fprintf('   determinato un valore di ampiezza non nullo!\n\n')
130:     fprintf(' - ci sono %d punti canale di ampiezza nulla,\n',ww_err)
131:     fprintf('   in cui   stato posto width = -999!\n\n')
132:     fprintf(' =====\n')
133:     fprintf('PREMERE [INVIO] PER CONTINUARE ([ctrl+C] per uscire)\n')
134:     pause
135: end
136:
```



```

1: !      The gateway routine: ovvero la subroutine che lega gli oggetti tipici
2: !      di matlab con le classiche variabili integer e real
3:      subroutine mexFunction(nlhs, plhs, nrhs, prhs)
4:      implicit none
5: !-----
6:      integer plhs(*), prhs(*)
7:      integer mxCreateDoubleMatrix
8:      integer contorno_pr, x_channel_pr, y_channel_pr
9:      integer passo_sez_pr, width_pr, width_sez_pr, ww_err_pr
10:     integer n_inlet_pr, n_channel_pr, ITER_pr, w_err_pr
11:     integer nlhs, nrhs
12:     integer n1, n2, n_inlet, n_channel, size_cont, n, m
13:     integer mxGetM, mxGetN, mxGetPr, mxIsNumeric, size_chann
14:     REAL*8 passo_sez
15:     REAL*8, allocatable :: width(:, :)
16:     REAL*8, allocatable :: contorno(:, :)
17:     REAL*8, allocatable :: width_sez(:, :)
18:     real*8 n_inlet_t, n_channel_t
19:     REAL*8 werror, wwerror
20: !-----
21: ! controllo che gli input siano in numero appropriato.
22:     if (nrhs .ne. 4) then
23:         call mexErrMsgTxt('Four inputs required.')
24:     endif
25: ! controllo che gli output siano in numero appropriato.
26:     if (nlhs .ne. 4) then
27:         call mexErrMsgTxt('Four output required.')
28:     endif
29: ! controllo che gli input siano matrici di numeri.
30:     if (mxIsNumeric(prhs(1)) .ne. 1) then
31:         call mexErrMsgTxt('Input #1 is not a numeric array.')
32:     endif
33:     if (mxIsNumeric(prhs(2)) .ne. 1) then
34:         call mexErrMsgTxt('Input #2 is not a numeric.')
35:     endif
36:     if (mxIsNumeric(prhs(3)) .ne. 1) then
37:         call mexErrMsgTxt('Input #3 is not a numeric.')
38:     endif
39:     if (mxIsNumeric(prhs(4)) .ne. 1) then
40:         call mexErrMsgTxt('Input #4 is not a numeric.')
41:     endif
42: ! rilevo le dimensioni degli input assegnati alla funzione
43:     n1 = mxGetM(prhs(1))
44:     n2 = mxGetN(prhs(1))
45:     allocate(contorno(n1, n2))
46:     allocate(width(n1, n2))
47:     allocate(width_sez(n1, n2))
48:     size_cont = n1*n2
49:
50: C      creo una matrice per i risultati di output.
51:     plhs(1) = mxCreateDoubleMatrix(n1, n2, 0)
52:     plhs(2) = mxCreateDoubleMatrix(n1, n2, 0)
53:     plhs(3) = mxCreateDoubleMatrix(1, 1, 0)
54:     plhs(4) = mxCreateDoubleMatrix(1, 1, 0)
55:
56: C      assegno i parametri di input e output a delle variabili.
57:     contorno_pr = mxGetPr(prhs(1))
58:     n_channel_pr = mxGetPr(prhs(2))
59:     n_inlet_pr = mxGetPr(prhs(3))

```

```

60:      passo_sez_pr = mxGetPr(prhs(4))
61:
62:      width_pr = mxGetPr(plhs(1))
63:      width_sez_pr = mxGetPr(plhs(2))
64:      w_err_pr = mxGetPr(plhs(3))
65:      ww_err_pr = mxGetPr(plhs(4))
66:
67: C      carico i dati di input in variabili fortran.
68:      call mxCopyPtrToReal8(contorno_pr, contorno, size_cont)
69:      call mxCopyPtrToReal8(n_channel_pr, n_channel_t, 1)
70:      call mxCopyPtrToReal8(n_inlet_pr, n_inlet_t, 1)
71:      call mxCopyPtrToReal8(passo_sez_pr, passo_sez, 1)
72:
73:      n_channel = INT(n_channel_t)
74:      n_inlet = INT(n_inlet_t)
75:
76: C      richiamo la computational subroutine.
77:      call ampiezza(n1,n2,contorno,n_channel,n_inlet,
78: $passo_sez,width,width_sez,werror,wwerror)
79:
80:      if (werror.ge.999999) then
81:          call mexErrMsgTxt('n_channel is not the correct number
82: $of channelized pixels')
83:      endif
84:
85:
86: C      carico i dati di output nella MATLAB array.
87:      call mxCopyReal8ToPtr(width, width_pr, size_cont)
88:      call mxCopyReal8ToPtr(width_sez, width_sez_pr, size_cont)
89:      call mxCopyReal8ToPtr(werror, w_err_pr, 1)
90:      call mxCopyReal8ToPtr(wwerror, ww_err_pr, 1)
91:
92:      deallocate(contorno)
93:      deallocate(width)
94:      deallocate(width_sez)
95:
96:      return
97:      contains
98: !=====
99: ! ===== SUBROUTINE AMPIEZZA =====
100: ! Calcola la FUNZIONE DI AMPIEZZA, o WIDTH FUNCTION:
101: ! per ogni pixel della rete di canali calcola la sua distanza
102: ! dalla bocca, seguendo la rete.
103: !=====
104: ! Computational subroutine
105:      subroutine ampiezza(n1,n2,contorno,n_channel,n_inlet,
106: $passo_sez,width,width_sez,werror,wwerror)
107:          implicit none
108: !----- Dichiarazione delle variabili -----
109:          INTEGER n_elementi
110: !Al primo passo il numero di elementi che costituiscono la sezione
111: !iniziale: bocca del canale.
112: !N.B.:potrebbero esserci pi sezioni iniziali
113:
114:          INTEGER xc(n_channel),yc(n_channel),xchann(n_channel)
115:          INTEGER ychann(n_channel), n_canali
116: !Al primo passo contengono le coordinate dei pixels che
117: !costituiscono la sezione iniziale o le sezioni iniziali.
118:

```

```

119:      INTEGER k,i,j,m      !indici dei cicli
120:      REAL*8 rad2, w, a
121:      REAL*8 diri(8),dirj(8)
122:      !puntatori delle direzioni lungo "i" e "j"
123:
124:      REAL*8 werror, wwerror
125:      !variabile indicatrice di errore
126:
127:      INTEGER n1,n2,n_channel,n_inlet
128:      REAL*8 passo_sez
129:      REAL*8 :: width(:, :)
130:      REAL*8 :: contorno(:, :)
131:      REAL*8 :: width_sez(:, :)
132:
133:      !----- Inizializzazione delle variabili -----
134:      rad2=SQRT(2.)
135:      werror=0
136:      wwerror=0
137:      do i=1,n_channel
138:          xc(i)=0
139:          yc(i)=0
140:      enddo
141:      DATA diri /0,-1,-1,-1,0,1,1,1/      ! Puntatore delle direzioni lungo "i"
142:      DATA dirj /1,1,0,-1,-1,-1,0,1/      ! Puntatore delle direzioni lungo "j"
143:      !- Inizializza la sezione di partenza o le sezioni di partenza -
144:      ! pone la funzione di ampiezza pari a zero, ovunque
145:      do j=1,n2
146:          do i=1,n1
147:              width(i,j)=0.
148:              width_sez(i,j)=0.
149:          enddo
150:      enddo
151:      ! inizializza a zero il numero di pixels appartenenti alle bocche
152:      n_elementi=0
153:      n_canali = 0
154:      !Scorre su tutte le righe e su tutte le colonne
155:      DO j=1,n2
156:          DO i=1,n1
157:              IF(contorno(i,j).ge.3.) THEN !trovo le coordinate dei punti canalizzati
158:                  n_canali=n_canali+1      !secondo la notazione del fortran
159:                  xchann(n_canali) = j
160:                  ychann(n_canali) = i
161:              END IF
162:              IF(contorno(i,j).eq.4) THEN !memorizza in xc e yc le coordinate dei
163:                  !pixel il cui valore nella matrice contorno è 4
164:                  ! sono in pixel di inizio per la funzione
165:                  !di ampiezza, sono cioe' i pixels appartenenti alle bocche.
166:                  n_elementi=n_elementi+1 !ha trovato 1 nuovo pixel che appartiene alla
167:                  !sezione iniziale aggiunge un elemento ai vettori xc e yc
168:                  xc(n_elementi)=j      !memorizza il valore della colonna in xc
169:                  yc(n_elementi)=i      !memorizza il valore della riga in yc
170:                  width(i,j)=1          !width vale 1 in qti pixel: sono i pixel della bocca
171:
172:              END IF
173:          END DO
174:      END DO
175:      ! controllo che il numero di pixel canale sia uguale ad n_channel
176:      IF(n_canali .ne. n_channel) THEN
          werror = 999999
      END IF

```

```

177:      ENDIF
178: !----- Inizia il ciclo su tutti gli n_elementi connessi -----
179:      w=1.
180:      !WRITE(*,*) 'Inizia il ciclo'
181:      loop1:DO WHILE (n_elementi.ne.0)
182:
183:          loop2:DO k=1,n_elementi !inizia il ciclo su tutti i pixel canale.
184:              !Alla la iterazione il ciclo viene fatto su
185:              !tutti gli elementi appartenenti alle bocche
186:              loop3:DO m=1,8 !inizia il ciclo in tutte le direzioni cioe' su ognuno
187:                  !degli 8vicini del pixel attualmente considerato.
188:                  !calcola un valore della funzione di ampiezza per il vicino considerato,
189:                  !sommando alla width del pixel (xc,yc) la distanza tra qto pixel ed il
190:                  !vicino considerato
191:                  a=width(yc(k),xc(k))+SQRT(ABS(diri(m))+ABS(dirj(m)))
192:
193:                  !Parte dal pixel di coordinate (xc,yc). Guarda gli otto vicini(loop3).
194:                  ! Ciclo IF esterno: se il vicino considerato e' un pixel canale
195:                  ! (contorno(...)==3) e se:
196:                  ! 1)qto pixel ha un valore di width==0 (perche' x qto pixel non e' ancora stata
197:                  ! calcolata la funzione di ampiezza e quindi width e' ancora 0, valore iniziale);
198:                  ! 2)la sua width e' > di quella del pixel di coordinate (xc,yc) (e cioe' guarda
199:                  ! solo i pixel upstream); allora ciclo IF interno:
200:                  ! se la width del vicino considerato e' > a oppure e' =0(perche' x qto pixel
201:                  ! non e' ancora stata calcolata la width function e quindi width e' ancora 0,
202:                  ! valore iniziale) allora pone width=a (questo perche' x ogni pixel si calc-
203:                  ! ola la minima distanza che lo congiunge alla sezione iniziale)
204:                  IF((contorno(yc(k)+diri(m),xc(k)+dirj(m)).eq.3).and.
205:                  $(width(yc(k)+diri(m),xc(k)+dirj(m)).eq.0).or.
206:                  $(width(yc(k)+diri(m),xc(k)+dirj(m)).gt.width(yc(k),xc(k)))
207:                  $)) THEN
208:                      IF((width(yc(k)+diri(m),xc(k)+dirj(m)).gt.a).or.
209:                      $(width(yc(k)+diri(m),xc(k)+dirj(m)).eq.0)) THEN
210:                          width(yc(k)+diri(m),xc(k)+dirj(m))=a
211:                      END IF
212:                  END IF
213:                  END DO loop3
214:              END DO loop2
215:
216:          ! Seleziona il nuovo insieme di partenza
217:          w=w+1
218:          n_elementi=0 !sta selezionando il nuovo insieme di partenza, perciò
219:          !azzerà il numero degli elementi dell'insieme.
220:          !Il nuovo insieme di partenza e' costituito da tutti i pixel canale x i quali
221:          !si ha w-2.< width <= w. Più' in generale questo vale x tutti i pixel x i quali
222:          !si ha w-OTT.< width <= w, con w che si incrementa di POTT ad ogni ciclo
223:          !(w=w+POTT). OTT e' il numero di sezioni su cui ottimizzare. POTT e' il passo
224:          !su cui ottimizzare. All'aumentare di OTT aumenta il numero di elementi
225:          !che formano il nuovo insieme di partenza.
226:          DO m=1,n_canali
227:              IF((width(ychann(m),xchann(m)).gt.w-2.).and.
228:              $(width(ychann(m),xchann(m)).le.w)) THEN
229:                  n_elementi=n_elementi+1
230:                  xc(n_elementi)=xchann(m)
231:                  yc(n_elementi)=ychann(m)
232:              END IF
233:          END DO
234:
235:      END DO loop1

```



```
236:
237: !Arrotonda i valori della funzione d'ampiezza e li calcola ogni 'passo_sez'.
238: !In questo modo costruisce delle sezioni individuate da pixel, appartenenti ad
239: !un ramo della rete, caratterizzati dall'aver la medesima distanza dalla bocca!
240:     DO j=1,n2
241:         DO i=1,n1
242:             ! controllo che non ci siano errori nel calcolo ad esempio
243:             ! pixel non canalizzati con un valore di ampiezza diverso da zero,
244:             ! oppure pixel canalizzati con valori di ampiezza nulli
245:             IF((contorno(i,j).lt.3).and.(width(i,j).ne.0)) THEN
246:                 werror=werror+1
247:                 width(i,j)=0.
248:             END IF
249:             IF((contorno(i,j).ge.3).and.(width(i,j).eq.0)) THEN
250:                 wwerror=wwerror+1
251:                 width(i,j)=-999.
252:             ENDIF
253:             width_sez(i,j) = width(i,j) / passo_sez
254:             IF (width(i,j).gt.0) THEN ! cioe' solo sui canali
255:                 width_sez(i,j)=INT(width_sez(i,j)) ! da verificare!
256:             END IF
257:         END DO
258:     END DO
259:
260:     return
261: end subroutine ampiezza
262: end subroutine mexfunction
263:
```



```

1: % [incidenza_rete,portata_rete,sorgenti_rete]=incidenzacanali(contorno,width);
2: %=====
3: % FUNZIONE CHE TROVA LE DIREZIONI DI DRENAGGIO SUI CANALI
4: % Data la matrice contenente la GEOMETRIA della rete (CONTORNO) e data la matri-
5: % ce contenente la 'FUNZIONE DI AMPIEZZA'(width) per i punti canale, questa fun-
6: % zione calcola:
7: % 1)la MATRICE 'incidenza_rete';
8: % 2)la MATRICE 'sorgenti_rete';
9: % 3)la MATRICE 'portata_rete'.
10: %=====
11: % INPUT:
12: % - contorno: (INPUT OBBLIGATORIO) matrice contenente le caratteristiche
13: % dell'area di studio (i.e. matrice delle condizioni al contorno):
14: % LA CONVENZIONE SUI PUNTI DELLA LAGUNA E' LA SEGUENTE:
15: % 0 APPARTIENE ALLO SPECCHIO D'ACQUA
16: % 1 CONDIZIONE AL contorno DI DIRICHLET
17: % 2 CONDIZIONE AL contorno DI NEUMANN PER PUNTI NON CANALE
18: % 3 CONDIZIONE AL contorno DI NEUMANN PER PUNTI CANALE
19: % 4 PUNTI ANCORA CANALE MA DI INIZIO PER LA FUNZIONE DI AMPIEZZA
20: % - width: (INPUT OBBLIGATORIO) matrice contenente la funzione ampiezza,
21: % per ogni pixel della rete di canali riporta la distanza dalla
22: % bocca del canale. Questa matrice viene calcolata dalla funzione
23: % 'ampiezza'
24: %
25: % OUTPUT(OBBLIGATORI):
26: % - incidenza_rete: matrice contenente le direzioni di drenaggio x tutti i
27: % punti che appartengono alla rete dei canali, ottenute
28: % seguendo la direzione del massimo gradiente della
29: % 'FUNZIONE DI AMPIEZZA'(width). Per i canali non
30: % possibile procedere come per la barena, perch lungo i
31: % canali la marea si propaga istantaneamente e quindi non
32: % si puo' utilizzare il gradiente della superficie libera
33: % per determinare le direzioni di drenaggio. Procedendo in
34: % questo modo le direzioni di drenaggio sono determinate
35: % seguendo il minimo percorso che porta alla sezione iniziale;
36: % - portata_rete: matrice che cumula secondo le direzioni di drenaggio lungo
37: % la rete dei canali, dando cos , per ogni pixel canale, il
38: % numero di pixel a monte dello stesso che drenano verso questo;
39: % - sorgenti_rete: matrice in cui per tutti i punti canale pari 0 nei punti
40: % sorgenti, cio nei pixel che non ricevono contributi da
41: % nessuno dei vicini, e 1 nei pixel che ricevono i contributi
42: % dei pixel vicini.
43: %
44: % NOTA: questa funzione composta in parte da codice matlab e in parte da un
45: % mex_file. Eventuali errori nella computazione della funzione
46: % vengono segnalati a video: in particolare, se ci sono pixel canale
47: % alla quale viene assegnata una direzione di drenaggio,
48: % questi vengono individuati e reimpostati pari a 0.
49: %
50: %=====
51: % PER L'ESECUZIONE E' RICHIESTA LA TOOLBOX TELONE_TBOX
52: %=====
53: % data di creazione: 21/05/2014
54: % ultima modifica: 17/06/2014
55: % autore: Francesco Carraro
56: %=====
57: function [incidenza_rete,portata_rete,sorgenti_rete]=...
58: % incidenzacanali(contorno,width);
59: % calcolo il numero di pixel barena nella matrice contorno

```

```
60: n_chann = length(contorno(contorno>=3));
61:
62: tic;
63: % si richiama il mex_file creato tramite F_incibarene.f90
64: [incidenza_rete,sorgenti_rete,portata_rete,err_rete]=...
65:                                     F_incicanali(contorno,width,n_chann);
66: t = toc;
67:
68: fprintf(...
69: 'l''elaboraz. di incidenzacanali e'' andata a buon fine in %4.1g secondi\n',t)
70:
71: % si scrivono a video eventuali errori che sono stati riscontrati nel calcolo
72: if err_rete > 0
73: fprintf(' ===== SONO STATI TROVATI ALCUNI ERRORI =====\n')
74: fprintf(' ### NB: ci sono %4g punti barena in cui e'' stato calcolata ###\n'...
75:         ,err_rete)
76: fprintf(' ###      erroneamente una direzione di drenaggio e in cui      ###\n')
77: fprintf(' ###      stato poi reimposto valore 0                          ###\n')
78: fprintf(' =====\n')
79: fprintf('PREMERE [INVIO] PER CONTINUARE ([ctrl+C] per uscire)\n')
80: pause
81: end
82:
```

```

1: !      The gateway routine: ovvero la subroutine che lega gli oggetti tipici
2: !      di matlab con le classiche variabili integer e real
3:      subroutine mexFunction(nlhs, plhs, nrhs, prhs)
4:      implicit none
5: !-----
6:      integer plhs(*), prhs(*)
7:      integer mxCreateDoubleMatrix
8:      integer contorno_pr, rete_err_pr
9:      integer incidenza_rete_pr, portata_rete_pr, sorgenti_rete_pr
10:     integer n_chann_pr, width_pr
11:     integer nlhs, nrhs
12:     integer n1, n2, n_chann, size_cont
13:     integer mxGetM, mxGetN, mxGetPr, mxIsNumeric
14:     REAL*8,allocatable :: contorno(:, :)
15:     REAL*8,allocatable :: incidenza_rete(:, :)
16:     REAL*8,allocatable :: sorgenti_rete(:, :)
17:     REAL*8,allocatable :: portata_rete(:, :)
18:     REAL*8,allocatable :: width(:, :)
19:     real*8 n_chann_t
20:     REAL*8 rete_err
21: !-----
22: ! controllo che gli input siano in numero appropriato.
23:     if (nrhs .ne. 3) then
24:         call mexErrMsgTxt('Four inputs required.')
25:     endif
26: ! controllo che gli output siano in numero appropriato.
27:     if (nlhs .ne. 4) then
28:         call mexErrMsgTxt('Four output required.')
29:     endif
30: ! controllo che gli input siano matrici di numeri.
31:     if (mxIsNumeric(prhs(1)) .ne. 1) then
32:         call mexErrMsgTxt('Input #1 is not a numeric array.')
33:     endif
34:     if (mxIsNumeric(prhs(2)) .ne. 1) then
35:         call mexErrMsgTxt('Input #2 is not a numeric array.')
36:     endif
37:     if (mxIsNumeric(prhs(3)) .ne. 1) then
38:         call mexErrMsgTxt('Input #3 is not a numeric.')
39:     endif
40: ! rilevo le dimensioni degli input assegnati alla funzione
41:     n1 = mxGetM(prhs(1))
42:     n2 = mxGetN(prhs(1))
43:     size_cont = n1*n2
44:
45:     allocate(incidenza_rete(n1,n2))
46:     allocate(width(n1,n2))
47:     allocate(contorno(n1,n2))
48:     allocate(sorgenti_rete(n1,n2))
49:     allocate(portata_rete(n1,n2))
50:
51: C      creo una matrice per i risultati di output.
52:     plhs(1) = mxCreateDoubleMatrix(n1, n2, 0)
53:     plhs(2) = mxCreateDoubleMatrix(n1, n2, 0)
54:     plhs(3) = mxCreateDoubleMatrix(n1, n2, 0)
55:     plhs(4) = mxCreateDoubleMatrix(1, 1, 0)
56:
57: C      assegno i parametri di input e output a delle variabili.
58:     contorno_pr = mxGetPr(prhs(1))
59:     width_pr = mxGetPr(prhs(2))

```

```

60:      n_chann_pr = mxGetPr(prhs(3))
61:
62:      incidenza_rete_pr = mxGetPr(plhs(1))
63:      sorgenti_rete_pr = mxGetPr(plhs(2))
64:      portata_rete_pr = mxGetPr(plhs(3))
65:      rete_err_pr = mxGetPr(plhs(4))
66:
67: C      carico i dati di input in variabili fortran.
68:      call mxCopyPtrToReal8(contorno_pr, contorno, size_cont)
69:      call mxCopyPtrToReal8(n_chann_pr, n_chann_t, 1)
70:      call mxCopyPtrToReal8(width_pr, width, size_cont)
71:
72:      n_chann = INT(n_chann_t)
73:
74:
75: C      richiamo la computational subroutine.
76:      call inci_rete(n1,n2,contorno,width,n_chann,
77: $incidenza_rete,sorgenti_rete,portata_rete,rete_err)
78:
79: C      carico i dati di output nella MATLAB array.
80:      call mxCopyReal8ToPtr(incidenza_rete,incidenza_rete_pr,
81: $size_cont)
82:      call mxCopyReal8ToPtr(sorgenti_rete,sorgenti_rete_pr,
83: $size_cont)
84:      call mxCopyReal8ToPtr(portata_rete,portata_rete_pr,
85: $size_cont)
86:      call mxCopyReal8ToPtr(rete_err, rete_err_pr, 1)
87:
88:      deallocate(incidenza_rete)
89:      deallocate(width)
90:      deallocate(contorno)
91:      deallocate(sorgenti_rete)
92:      deallocate(portata_rete)
93:
94:      return
95:      contains
96:
97: ! =====
98: ! ===== SUBROUTINE LARGH =====
99: ! Calcola la LARGHEZZA dei canali e la scrive per sezioni
100: ! =====
101:      ! Computational subroutine
102:      SUBROUTINE inci_rete(n1,n2,contorno,width,n_channel,
103: $incidenza_rete,sorgenti_rete,portata_rete,rete_err)
104: !Data la matrice contenente la GEOMETRIA della rete (CONTORNO) e data la matrice
105: ! della 'FUNZIONE DI AMPIEZZA'(width) per i punti canale.
106: ! Questa subroutine calcola:
107: ! 1)la MATRICE 'incidenza_rete'(contenente le direzioni di drenaggio x tutti
108: ! i punti che appartengono alla rete dei canali, ottenute seguendo la direzione
109: ! del massimo gradiente della 'FUNZIONE DI AMPIEZZA'(width). Infatti x i canali
110: ! non e' possibile procedere come per la barena, perche' lungo i canali la
111: ! marea si propaga istantaneamente e quindi non si pu' utilizzare il gradiente
112: ! del pelo libero per determinare le direzioni di drenaggio. Procedendo in q.to
113: ! modo le direzioni di drenaggio sono determinate seguendo il minimo percorso
114: ! che porta alla sezione iniziale;
115: ! 2)la MATRICE 'portata_rete' (che cumula secondo le direzioni di drenaggio lun-
116: ! go la rete dei canali, dando cosi', per ogni pixel canale, il numero di
117: ! pixel a monte dello stesso che drenano verso questo);
118: ! 3)la MATRICE delle 'sorgenti_rete' x tutti i punti canale (che vale 0 nei pti

```

```

119: ! sorgenti, cioe' nei pixel che non ricevono contributi da nessuno dei vicini,
120: ! e 1 nei pixel che ricevono i contributi dei pixel vicini).
121: !-----
122:      IMPLICIT NONE
123: !----- Dichiarazione delle variabili -----
124:      REAL*8 diri(8),dirj(8)           ! Puntatore delle direzioni
125:      REAL*8 slope(8),cslope(8),area,ran
126:      REAL*8 :: lung, somma, media
127:      INTEGER :: kmin(1),j,i,k,il,jl,m_1,n_1,m
128:      INTEGER :: rnd,krand,kminimo
129:      REAL*8 rete_err                 ! Variabile indicatrice dell'errore
130:      INTEGER n1,n2,n_channel,n_rete
131:      REAL*8 :: incidenza_rete(:, :)
132:      REAL*8 :: sorgenti_rete(:, :)
133:      REAL*8 :: portata_rete(:, :)
134:      REAL*8 :: contorno(:, :)
135:      REAL*8 :: width(:, :)
136:      integer,allocatable :: xchann(:)
137:      integer,allocatable :: ychann(:)
138:
139: !----- Inizializzazione delle variabili -----
140:      allocate(xchann(n_channel))
141:      allocate(ychann(n_channel))
142:
143:      rete_err = 0.
144:      incidenza_rete = 0. !Inizializzo la matrice di incidenza delle barene
145:      sorgenti_rete = 0. !Inizializzo la matrice delle sorgenti delle barene
146:      portata_rete = 0. !Inizializzo la matrice delle portate delle barene che
147:                        !x ogni pixel di barena contiene il numero di pixel
148:                        !drenati da questo pixel
149:      xchann = 0
150:      ychann = 0
151:      ! Puntatore delle direzioni
152:      DATA diri /0,-1,-1,-1,0,1,1,1/   ! Puntatore delle direzioni lungo "i"
153:      DATA dirj /1,1,0,-1,-1,-1,0,1/   ! Puntatore delle direzioni lungo "j"
154:
155:      n_rete = 0
156:      do i=1,n1
157:         do j=1,n2
158:            if(contorno(i,j).ge.3) then
159:               n_rete = n_rete+1
160:               xchann(n_rete) = j
161:               ychann(n_rete) = i
162:            endif
163:         enddo
164:      enddo
165:
166: !----- Calcolo -----
167: ! Calcola 'INCIDENZA_RETE' e 'SORGENTI_RETE'
168: ! Calcola la MATRICE 'incidenza_rete' per tutti i punti canale: calcola cioe' la
169: ! direzione di drenaggio per ciascun pixel. Poiche' la direzione di drenaggio
170: ! coincide con la direzione lungo la quale e' minima la distanza dalla sezione
171: ! iniziale, x ogni pixel calcola il gradiente della 'FUNZIONE DI AMPIEZZA'
172: ! (width) e individua dove qto pixel scarica la propria portata.
173:
174:      loop1:DO m=1,n_channel ! esegue il ciclo per tutti i pixel canale
175:         j = xchann(m)      ! memorizza le coordinate dei
176:         i = ychann(m)      ! pixel canale in 'i' e 'j'
177:

```

```

178:         DO k=1,8  !inizia il ciclo in tutte le direzioni cioe' su ognuno
179:                 !degli 8 vicinidel pixel attualmente considerato.
180:                 slope(k)=(width(i,j)-width(i+diri(k),j+dirj(k)))/
181: $ SQR(T(ABS(diri(k))+ABS(dirj(k))))
182:                 IF(width(i+diri(k),j+dirj(k)).eq.0) THEN!se non e' un punto canale
183:                 slope(k)=0 !allora pone pendenza = 0.
184:                 END IF
185:             END DO
186: ! Trasla il vettore delle pendenze di un numero casuale da uno a otto e trova la
187: ! posizione minima in questo vettore. In questo modo se ci sono piu' di due mi-
188: ! nimi, ne sceglie uno a caso
189:         rnd=rnd+1
190:         krand=INT(RAN(rnd)*8)
191:         cslope=CSHIFT(slope,krand)
192:         kmin=MAXLOC(cslope)
193:         kminimo=kmin(1)
194:         kminimo=MOD(kmin(1)-1+krand,8)+1
195:
196:         incidenza_rete(i,j)=kminimo
197:         sorgenti_rete(i+diri(kminimo),j+dirj(kminimo))= 1
198:     END DO loop1
199:
200: !-----
201: ! Calcola 'PORTATA_RETE'
202: ! Parte da una SORGENTE (dove cioe' la matrice 'incidenza_rete' e' nulla) e
203: ! segue il percorso di massimo gradiente assegnando le portate, (assegnando
204: ! cioe' le aree drenate da ciascun pixel).
205:     loop2:DO m=1,n_channel ! esegue il ciclo per tutti i pixel canale
206:         j = xchann(m) ! memorizza le coordinate dei
207:         i = ychann(m) ! pixel canale in 'i' e 'j'
208:         IF((sorgenti_rete(i,j)==0).and.(contorno(i,j)==3)) THEN
209:
210:             area=1
211:             portata_rete(i,j)=area
212:             il=i
213:             jl=j
214:
215: 222             m_1=diri(incidenza_rete(il,jl))
216:                 n_1=dirj(incidenza_rete(il,jl))
217:
218:             il=il+m_1
219:             jl=jl+n_1
220:
221:             IF((portata_rete(il,jl)==0).and.
222: $ (contorno(il,jl)==3)) THEN
223:                 area=area+1
224:             END IF
225:
226:             portata_rete(il,jl)=portata_rete(il,jl)+area
227:
228:             IF(contorno(il,jl)/=4) GOTO 222
229:         END IF
230:     END DO loop2
231: !-----
232: ! CONTROLLO
233: ! Controlla di non aver attribuito valori di incidenza_rete a pixels
234: ! che non appartengono alla rete (cio a pixels tali che contorno(i,j) < 3)
235:     DO i=1,n1
236:         DO j=1,n2

```



```
237:         IF((contorno(i,j) < 3).and.(incidenza_rete(i,j).ne.0)
238: $.and.(portata_rete(i,j).ne.0)) THEN
239:         rete_err=rete_err+1
240:         incidenza_rete(i,j)=0
241:         portata_rete(i,j)=0
242:         END IF
243:     END DO
244: END DO
245:
246: deallocate(xchann)
247: deallocate(ychann)
248: RETURN
249: !=====
250: end subroutine inci_rete
251: end subroutine mexfunction
252:
```



```

1: % [rami,bacini,contnew]= segnarami(contorno,incidenza_rete,incidenza,...
2: %
3: %=====
4: % FUNZIONE PER LA DEFINIZIONE DELLA FORMA DELLE AREE DI DRENAGGIO
5: % La funzione segnarami permette di marcare col medesimo valore (sezindex(..))
6: % tutti i pixel canale che afferiscono ad una determinata sezione di
7: % coordinate(xsez(..),ysez(..)) marcata con lo stesso valore.
8: %=====
9: % INPUT:
10: % - contorno: (INPUT OBBLIGATORIO) matrice contenente le caratteristiche
11: % dell'area di studio (i.e. matrice delle condizioni al contorno):
12: % LA CONVENZIONE SUI PUNTI DELLA LAGUNA E' LA SEGUENTE:
13: % 0 APPARTIENE ALLO SPECCHIO D'ACQUA
14: % 1 CONDIZIONE AL contorno DI DIRICHLET
15: % 2 CONDIZIONE AL contorno DI NEUMANN PER PUNTI NON CANALE
16: % 3 CONDIZIONE AL contorno DI NEUMANN PER PUNTI CANALE
17: % 4 PUNTI ANCORA CANALE MA DI INIZIO PER LA FUNZIONE DI AMPIEZZA
18: % - incidenza_rete:(INPUT OBBLIGATORIO) matrice contenente le direzioni di
19: % drenaggio x tutti i punti che appartengono alla rete dei
20: % canali, ottenute seguendo la direzione del massimo gradiente
21: % della 'FUNZIONE DI AMPIEZZA'(width).
22: % - incidenza:(INPUT OBBLIGATORIO) matrice contenente le direzioni di
23: % drenaggio in tutto il campo di moto considerato;
24: % - sezioni: (INPUT OBBLIGATORIO) e' la matrice contorno nella quale sono
25: % stati colorati con un certo valore (sezindex) i pixels apparte-
26: % nenti alle sezioni di chiusura di alcuni rami;
27: % - n_segnati:(INPUT FACOLTATIVO) n° di pixel appartenenti alle sezioni marca-
28: % te da un valore di sezindex nella matrice sezioni, se non speci-
29: % ficato la funzione lo calcola autonomamente cercando gli elementi
30: % di tale matrice maggiori di 4;
31: % - n_chann: (INPUT FACOLTATIVO) n° di pixel canale presenti nella matrice
32: % contorno, se non specificato la funzione lo calcola autonomamente
33: % cercando gli elementi della matrice >= 3.
34: %
35: % OUTPUT:
36: % - rami: matrice in cui si marcano con i valori di sezindex i rami dei canali
37: % chiusi dalle sezioni rispetto alle quali determinare le aree di
38: % drenaggio;
39: % - bacini: matrice in cui si marcano con i valori di sezindex le aree drenate
40: % dai rispettivi rami riportati nella matrice rami;
41: % - contnew: matrice delle condizioni al contorno per i rami analizzati e le
42: % relative aree di drenaggio;
43: %
44: % NOTA1: e' possibile assegnare i parametri di input in tre modi diversi:
45: % - assegnando solo le matrici obbligatorie la funzione calcolera'
46: % autonomamente i valori di default per gli altri parametri e li
47: % scrivera' a video chiedendo conferma all'utente;
48: % - assegnando tutti i parametri richiesti RISPETTANDO STRETTAMENTE
49: % L'ORDINE (contorno,incidenza_rete,incidenza,sezioni,n_segnati,
50: % n_chann), la funzione eseguirà il calcolo segnalando a
51: % video solo eventuali errori.
52: %
53: % !!!NB:E' FONDAMENTALE IN QUESTO CASO IL RISPETTO DELL'ORDINE.!!!
54: %
55: % - assegnando solo alcuni dei parametri facoltativi specificandoli,
56: % dopo l'assegnazione della matrice 'contorno', in qualsiasi ordine
57: % assegnandoli dopo aver posto tra apici il nome del parametro che si
58: % desidera imporre.
59: %

```

```

60: % NOTA2: questa funzione e' composta in parte da codice matlab e in parte da un
61: %      mex_file.
62: %=====
63: % PER L'ESECUZIONE E' RICHIESTA LA TOOLBOX TELONE_TBOX
64: %=====
65: % data di creazione: 17/06/2014
66: % ultima modifica: 17/06/2014
67: % autore: Francesco Carraro
68: %=====
69: function [rami,bacini,contnew]=segnarami(contorno,incidenza_rete,incidenza,...
70:      sezioni,varargin);
71: % creo la struttura di input: preparo i parametri da dare alla mex_function,
72: %      impostando i valori di default per i parametri
73: %      opzionali.
74: % creo la struttura
75: in = inputParser;
76: % scelgo i valori di default
77: defaultN_chann = length(contorno(contorno>=3));
78: defaultN_segnati = length(sezioni(sezioni>4));
79: % creo i parametri obbligatori
80: addRequired(in,'contorno',@isnumeric);
81: addRequired(in,'incidenza_rete',@isnumeric);
82: addRequired(in,'incidenza',@isnumeric);
83: addRequired(in,'sezioni',@isnumeric);
84: % creo i parametri facoltativi
85: addOptional(in,'n_chann',defaultN_chann,@isnumeric);
86: addOptional(in,'n_segnati',defaultN_segnati,@isnumeric);
87: % creo il file struttura risultante con gli input corretti per il mex
88: parse(in,contorno,incidenza_rete,incidenza,sezioni,varargin{:});
89:
90: % nel caso non siano stati assegnati tutti i parametri richiesti, scrivo a video
91: % i valori che la funzione ha scelto per l'esecuzione della mex_function
92: num = 0;
93: for iii=1:length(varargin),num=num+isnumeric(varargin{iii});end
94: if (length(varargin) < 4) && num < 2
95:     fprintf('\nMANCANO ALCUNI POSSIBILI INPUT OPZIONALI!\n')
96:     fprintf('Per l''esecuzione della funzione verranno utilizzati i seguenti\n')
97:     fprintf('valori delle variabili opzionali:\n')
98:     fprintf('\t- n_chann = %d;\n',in.Results.n_chann)
99:     fprintf('\t- n_segnati = %d;\n',in.Results.n_segnati)
100:    fprintf('Per ulteriori informazioni digitare a linea di comando:\n')
101:    fprintf('help segnarami\n')
102:    fprintf('PREMERE [INVIO] PER CONTINUARE ([ctrl+C] per uscire)\n')
103:    pause
104: end
105:    fprintf(' ...')
106:    fprintf('ESECUZIONE DELLA FUNZIONE segnarami IN CORSO...\n')
107:
108: tic;
109: % si richiama il mex_file creato tramite F_telone.f90
110: [rami,bacini,contnew]=F_segna_rami(in.Results.contorno,...
111:      in.Results.incidenza_rete,...
112:      in.Results.incidenza,...
113:      in.Results.sezioni,...
114:      in.Results.n_chann,...
115:      in.Results.n_segnati);
116: t = toc;
117: fprintf(...
118: ' l''elaborazione di segnarami e'' andata a buon fine in %4.1f secondi.\n',t)

```

```

1: !      The gateway routine: ovvero la subroutine che lega gli oggetti tipici
2: !      di matlab con le classiche variabili integer e real
3:      subroutine mexFunction(nlhs, plhs, nrhs, prhs)
4:      implicit none
5: !-----
6:      integer plhs(*), prhs(*)
7:      integer mxCreateDoubleMatrix
8:      integer contorno_pr,sezioni_pr,incidenza_pr, rami_pr
9:      integer incidenza_rete_pr,bacini_pr,contornonew_pr
10:     integer n_segna_t,canali_tot_pr
11:     integer nlhs, nrhs
12:     integer n1, n2, size_cont
13:     integer mxGetM, mxGetN, mxGetPr, mxIsNumeric
14:     REAL*8,allocatable :: contorno(:, :)
15:     REAL*8,allocatable :: incidenza(:, :)
16:     REAL*8,allocatable :: incidenza_rete(:, :)
17:     REAL*8,allocatable :: sezioni(:, :)
18:     REAL*8,allocatable :: rami(:, :)
19:     REAL*8,allocatable :: bacini(:, :)
20:     REAL*8,allocatable :: contornonew(:, :)
21:     REAL*8 :: n_segna_t, canali_tot_t
22:     INTEGER:: n_segna_t, canali_tot
23: !-----
24: ! controllo che gli input siano in numero appropriato.
25:     if (nrhs .ne. 6) then
26:         call mexErrMsgTxt('Four inputs required.')
27:     endif
28: ! controllo che gli output siano in numero appropriato.
29:     if (nlhs .ne. 3) then
30:         call mexErrMsgTxt('Four output required.')
31:     endif
32:
33: ! controllo che gli input siano matrici di numeri.
34:     if (mxIsNumeric(prhs(1)) .ne. 1) then
35:         call mexErrMsgTxt('Input #1 is not a numeric array.')
36:     endif
37:     if (mxIsNumeric(prhs(2)) .ne. 1) then
38:         call mexErrMsgTxt('Input #2 is not a numeric array.')
39:     endif
40:     if (mxIsNumeric(prhs(3)) .ne. 1) then
41:         call mexErrMsgTxt('Input #3 is not a numeric array.')
42:     endif
43:     if (mxIsNumeric(prhs(4)) .ne. 1) then
44:         call mexErrMsgTxt('Input #4 is not a numeric array.')
45:     endif
46:     if (mxIsNumeric(prhs(5)) .ne. 1) then
47:         call mexErrMsgTxt('Input #5 is not a numeric.')
48:     endif
49:     if (mxIsNumeric(prhs(6)) .ne. 1) then
50:         call mexErrMsgTxt('Input #6 is not a numeric.')
51:     endif
52: ! rilevo le dimensioni degli input assegnati alla funzione
53:     n1 = mxGetM(prhs(1))
54:     n2 = mxGetN(prhs(1))
55:     size_cont = n1*n2
56:
57:     allocate(incidenza(n1,n2))
58:     allocate(incidenza_rete(n1,n2))
59:     allocate(sezioni(n1,n2))

```

```

60:      allocate(contorno(n1,n2))
61:      allocate(rami(n1,n2))
62:      allocate(bacini(n1,n2))
63:      allocate(contornonew(n1,n2))
64:
65: C    creo una matrice per i risultati di output.
66:      plhs(1) = mxCreateDoubleMatrix(n1, n2, 0)
67:      plhs(2) = mxCreateDoubleMatrix(n1, n2, 0)
68:      plhs(3) = mxCreateDoubleMatrix(n1, n2, 0)
69:
70: C    assegno i parametri di input e output a delle variabili.
71:      contorno_pr = mxGetPr(prhs(1))
72:      incidenza_rete_pr = mxGetPr(prhs(2))
73:      incidenza_pr = mxGetPr(prhs(3))
74:      sezioni_pr = mxGetPr(prhs(4))
75:      n_segnati_pr = mxGetPr(prhs(5))
76:      canali_tot_pr = mxGetPr(prhs(6))
77:
78:      rami_pr = mxGetPr(plhs(1))
79:      bacini_pr = mxGetPr(plhs(2))
80:      contornonew_pr = mxGetPr(plhs(3))
81:
82: C    carico i dati di input in variabili fortran.
83:      call mxCopyPtrToReal8(contorno_pr, contorno, size_cont)
84:      call mxCopyPtrToReal8(incidenza_rete_pr, incidenza_rete,
85: $ size_cont)
86:      call mxCopyPtrToReal8(incidenza_pr, incidenza, size_cont)
87:      call mxCopyPtrToReal8(sezioni_pr, sezioni, size_cont)
88:      call mxCopyPtrToReal8(n_segnati_pr, n_segnati_t, 1)
89:      call mxCopyPtrToReal8(canali_tot_pr, canali_tot_t, 1)
90:
91:      n_segnati = INT(n_segnati_t)
92:      canali_tot = INT(canali_tot_t)
93:
94: C    richiamo la computational subroutine.
95:      call segna_rami(n1,n2,contorno,incidenza_rete,incidenza,
96: $ sezioni,n_segnati,canali_tot,rami,bacini,contornonew)
97:
98:
99: C    carico i dati di output nella MATLAB array.
100:     call mxCopyReal8ToPtr(rami,rami_pr,size_cont)
101:     call mxCopyReal8ToPtr(bacini,bacini_pr,size_cont)
102:     call mxCopyReal8ToPtr(contornonew,contornonew_pr,
103: $size_cont)
104:
105:     deallocate(incidenza)
106:     deallocate(incidenza_rete)
107:     deallocate(sezioni)
108:     deallocate(contorno)
109:     deallocate(rami)
110:     deallocate(bacini)
111:     deallocate(contornonew)
112:
113:     return
114:     contains
115:
116: !=====
117: ! ===== SUBROUTINE SEGNA_RAMI =====
118: !=====

```

```

119:      subroutine segna_rami(n1,n2,contorno,incidenza_rete,
120:      $ incidenza,sezioni,n_segnati,canali_tot,rami,bacini,
121:      $ contornonew)
122:      ! Permette di marcare col medesimo valore (sezindex(..)) tutti i pixel canale
123:      ! che afferiscono ad una determinata sezione di coordinate (xsez(..),ysez(..))
124:      ! marcata con lo stesso valore.
125:      IMPLICIT NONE
126:      ! Dichiarazione delle variabili
127:      CHARACTER(len=90) filedat, filemat, fileinc, filesez      ! files di input

128:      REAL*8 :: contorno(:, :) !matrice che definisce la configuraz. planimetrica
129:      !della rete. Essa contiene le condizioni al contorno necessarie per risolvere il
130:      !problema LAPLACIANO(eta1)=cost
131:      !-----
132:      ! La convenzione sui punti della laguna e' la seguente:
133:      ! 0 appartiene allo specchio d'acqua (punti barena o bassofondale)
134:      ! 1 condizione al contorno di dirichlet (flusso nullo)
135:      ! 2 condizione al contorno di neumann per punti non canale
136:      ! 3 condizione al contorno di neumann per punti canale (eta1=0)
137:      ! 4 punti ancora canale ma di inizio per la funzione di ampiezza
138:      !-----
139:      REAL*8 :: rami(:, :)
140:      REAL*8 :: sezioni(:, :)
141:      !-----
142:      ! SEZIONI e' la matrice contorno nella quale sono stati colorati con un certo
143:      ! valore i pixels appartenenti alle sezioni di chiusura di alcuni rami
144:      !-----

145:      REAL*8 :: bacini(:, :)
146:      REAL*8 :: contornonew(:, :)
147:      REAL*8 :: incidenza_rete(:, :) ! matrice delle incidenze dei canali
148:      REAL*8 :: incidenza(:, :)      ! matrice delle incidenze per l'intera laguna
149:      REAL*8, ALLOCATABLE :: sezindex(:)
150:      INTEGER, allocatable :: xsez(:), ysez(:)
151:      INTEGER, allocatable :: xcanali(:), ycanali(:) !coordinate dei pixel canale
152:      INTEGER, allocatable :: xmarsh(:), ymarsh(:)  !coordinate dei pixel barena
153:      INTEGER :: diri(8), dirj(8)      !puntatori delle direzioni lungo "i" e "j"

154:      INTEGER :: n1, n2      !numero di righe e di colonne
155:      INTEGER :: i, j, k, m, m_1, n_1, i1, j1      !indici dei cicli
156:      INTEGER :: n_segnati !numero di pixel marcati appartenenti alle sezioni
157:      INTEGER :: n_sez      !numero di pixel marcati appartenenti alle sezioni scelte
158:      !controllato scorrendo la matrice per righe e per colonne
159:      INTEGER :: canali_tot !numero di punti canale (totale dei pixel canale)
160:      INTEGER :: n_canali !numero di punti canale
161:      INTEGER :: n_marsh !numero di punti barena
162:      ALLOCATE(xsez(n_segnati))
163:      ALLOCATE(ysez(n_segnati))
164:      ALLOCATE(sezindex(n_segnati))
165:      ALLOCATE(xcanali(canali_tot))
166:      ALLOCATE(ycanali(canali_tot))
167:      ALLOCATE(xmarsh(n1*n2))
168:      ALLOCATE(ymarsh(n1*n2))
169:      !-----
170:      !----- Inizializzazione delle variabili -----
171:      DATA diri /0,-1,-1,-1,0,1,1,1/      ! Puntatore delle direzioni lungo "i"
172:      DATA dirj /1,1,0,-1,-1,-1,0,1/      ! Puntatore delle direzioni lungo "j"
173:      xcanali = 0
174:      ycanali = 0

```

```

175:      xsez      = 0
176:      ysez      = 0
177:      n_sez     = 0
178:      n_canali  = 0
179: !----- Calcolo -----
180: !memorizza le coordinate relative alle sezioni esaminate e il valore dell'indice
181: !in quei pixels (cio del valore assegnato a quei pixels)
182:      DO i=1,n1
183:          DO j=1,n2
184:              IF(sezioni(i,j) > 4) THEN
185:                  n_sez = n_sez + 1
186:              ! IF (n_sez>n_segnati) THEN
187:              ! WRITE(*,*) 'errore: n_sez > n_segnati'
188:              ! WRITE(*,*) 'n_sez=',n_sez
189:              ! WRITE(*,*) 'n_segnati=',n_segnati
190:              ! CYCLE
191:              ! END IF
192:              xsez(n_sez) = i
193:              ysez(n_sez) = j
194:              sezindex(n_sez) = sezioni(i,j)
195:          END IF
196:      END DO
197:  END DO
198:
199: ! memorizza le coordinate dei pixels canale
200:      DO i=1,n1
201:          DO j=1,n2
202:              IF(contorno(i,j)>=3) THEN
203:                  n_canali=n_canali+1
204:              ! IF (n_canali>canali_tot) THEN
205:              ! WRITE(*,*) 'errore: n_canali > canali_tot'
206:              ! WRITE(*,*) 'n_canali=',n_canali
207:              ! WRITE(*,*) 'canali_tot=',canali_tot
208:              ! CYCLE
209:              ! END IF
210:              xcanali(n_canali) = i
211:              ycanali(n_canali) = j
212:          END IF
213:      END DO
214:  END DO
215: !----- inizia elaborazione -----
216: ! Assegna a tutti i pixel appartenenti ad un ramo la cui bocca ha valore
217: ! "sezindex" lo stesso valore "sezindex"! Ho quindi una matrice dei rami
218: ! nella quale ogni ramo ha un diverso valore.
219: ! Per far questo:
220: ! (1) copia la matrice CONTORNO su RAMI
221:      rami = contorno
222: ! (2) in RAMI colora tutte le sezioni con un certo valore di "sezindex"
223:      DO m=1,n_sez
224:          rami(xsez(m),ysez(m)) = sezindex(m)
225:      END DO
226: ! (3) assegna a tutti i pixel canale che drenano in un pixel appartenente
227: ! alle bocche e caratterizzato da un certo valore di "sezindex(:)"
228: ! quel valore di "sezindex"
229:      DO j=1,n_canali
230:
231:          il=xcanali(j)
232:          jl=ycanali(j)
233:

```



```

234: 222          m_1=diri(INT(incidenza_rete(il,j1)))
235:             n_1=dirj(INT(incidenza_rete(il,j1)))
236:
237:             il=il+m_1
238:             j1=j1+n_1
239:
240:             IF(((contorno(il,j1) == 4)).and.((sezioni(il,j1)
241: $ <= 4))) CYCLE
242:             IF(sezioni(il,j1) >= minval(sezindex)) THEN
243:                 rami(xcanali(j),ycanali(j)) =sezioni(il,j1)
244:             ELSE
245:                 IF (incidenza_rete(il,j1)/=0) GOTO 222
246:             END IF
247:
248:         END DO
249:
250: !-----
251: ! adesso assegna a tutti i pixel barena che drenano in un determinato
252: ! ramo il valore di quel ramo, cosi' colora in modi diversi sottobacini
253: ! afferenti a rami diversi
254: ! Questa operazione la fa solo sui pixel barena, quindi prima memorizza
255: ! le coordinate dei pixel barena
256:     xmarsh = 0
257:     ymarsh = 0
258:     n_marsh = 0
259: ! memorizza le coordinate dei pixels barena
260:     DO i=1,n1
261:         DO j=1,n2
262:             IF(contorno(i,j) == 0.) THEN
263:                 n_marsh = n_marsh + 1
264:                 xmarsh(n_marsh) = i
265:                 ymarsh(n_marsh) = j
266:             END IF
267:         END DO
268:     END DO
269: ! assegna a tutti i pixel barena che drenano in un ramo caratterizzato da un
270: ! certo valore di "sezindex(:)" quel valore di "sezindex".
271: ! Per i pixel canale lo ha gia' fatto e quindi copia la matrice rami su bacini
272:     bacini = contorno
273: !bacini = rami
274:     DO j=1, n_marsh
275:         il=xmarsh(j)
276:         j1=ymarsh(j)
277:
278: 223          m_1=diri(INT(incidenza(il,j1)))
279:             n_1=dirj(INT(incidenza(il,j1)))
280:
281:             il=il+m_1
282:             j1=j1+n_1
283:
284:             IF(((contorno(il,j1) == 4)).and.((rami(il,j1)
285: $ <= 4))) CYCLE
286:
287:             IF(rami(il,j1) >= 4.1) THEN
288:                 bacini(xmarsh(j),ymarsh(j)) = rami(il,j1)
289:             ELSE
290:                 IF (incidenza(il,j1)/=0) GOTO 223
291:             END IF
292:         END DO

```

```
293:
294: ! crea la matrice contornonew
295: ! la pone tutta uguale a 1 cosi' ha gia' imposto la condizione di flusso nullo
296:   contornonew = 1.
297:   DO i=1,n1
298:     DO j=1,n2
299:       IF(bacini(i,j) > 4) THEN
300:         contornonew(i,j)=0.
301:       END IF
302:       IF(rami(i,j) > 4) THEN
303:         contornonew(i,j)=3.
304:       END IF
305:     END DO
306:   END DO
307:
308:   DEALLOCATE(xsez)
309:   DEALLOCATE(ysez)
310:   DEALLOCATE(sezindex)
311:   DEALLOCATE(xcanali)
312:   DEALLOCATE(ycanali)
313:   DEALLOCATE(xmarsh)
314:   DEALLOCATE(ymarsh)
315:   RETURN
316:
317: !=====
318:   end subroutine segna_rami
319:   end subroutine mexfunction
```

Appendice B

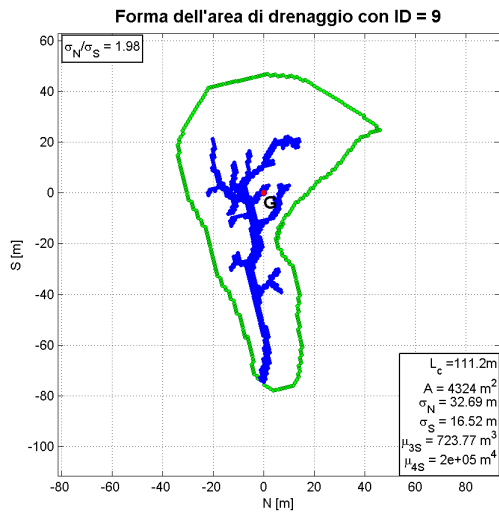
Grafici della forma delle aree di drenaggio analizzate

In questa Appendice si riportano i grafici della forma delle 63 aree di drenaggio dei canali a marea presenti all'interno dell'area di studio (cfr. par. 3.2).

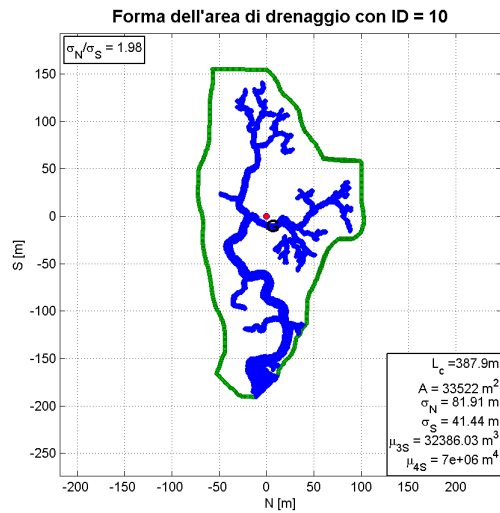
La forma di ognuno dei bacini di drenaggio, estratta dalla matrice rappresentata in Figura 5.6, viene qui illustrata in funzione del canale lagunare che la drena ovvero, rispetto al sistema di riferimento baricentrico (n, s) definito nel Capitolo 5, in cui s è la direzione individuata dalla retta passante per il centro della sezione di sbocco del canale e il baricentro dell'area di drenaggio da esso drenata.

All'interno di ogni grafico, in alto a sinistra, si riporta il valore del parametro R_σ (rapporto delle deviazioni standard rispetto agli assi n e s), assunto come valore di riferimento per la classificazione delle aree di drenaggio nel paragrafo 5.2; in basso a destra, è invece riportata la legenda di riepilogo degli parametri di interesse calcolati per ogni singola area di drenaggio, ovvero:

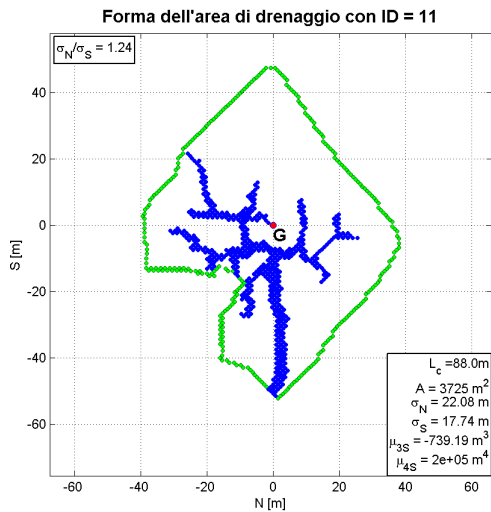
- L_c : lunghezza del canale;
- A : estensione dell'area drenata;
- σ_n : deviazione standard rispetto all'asse n (5.2);
- σ_s : deviazione standard rispetto all'asse s (5.1);
- $\mu_s^{[3]}$: *skewness* rispetto all'asse s (5.4);
- $\mu_s^{[4]}$: *kurtosis* rispetto all'asse s (5.5).



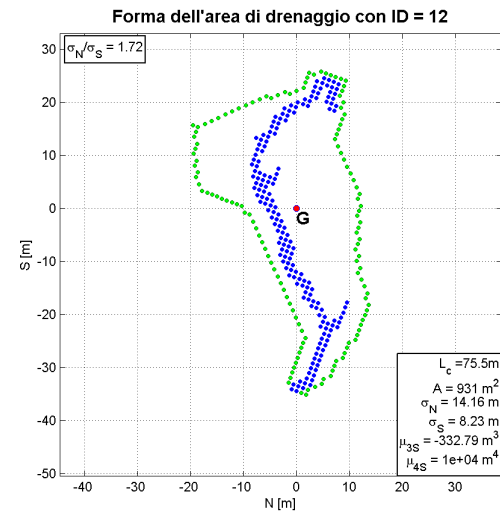
(a) Forma dell'area con $ID = 9$.



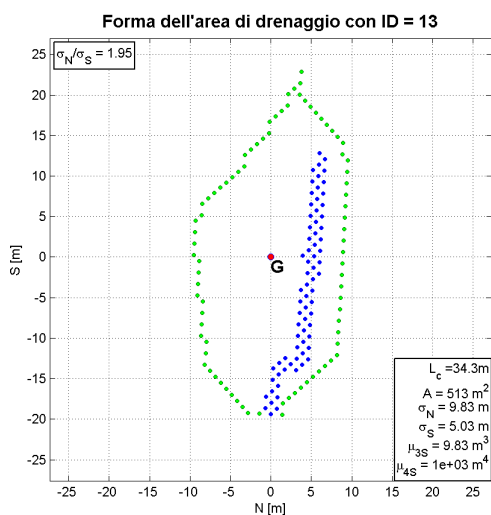
(b) Forma dell'area con $ID = 10$.



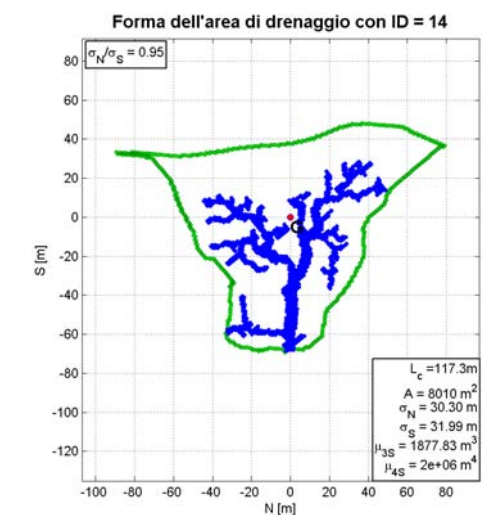
(c) Forma dell'area con $ID = 11$.



(d) Forma dell'area con $ID = 12$.

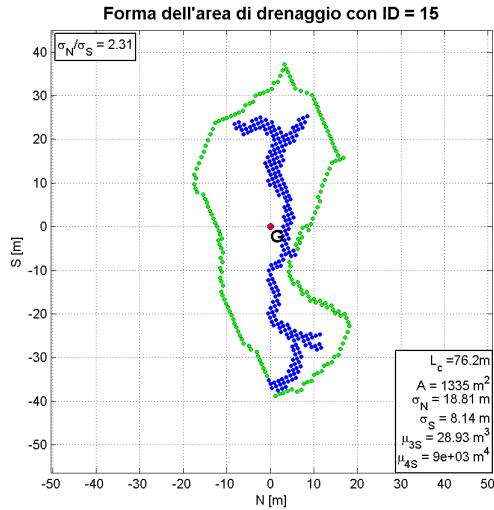


(e) Forma dell'area con $ID = 13$.

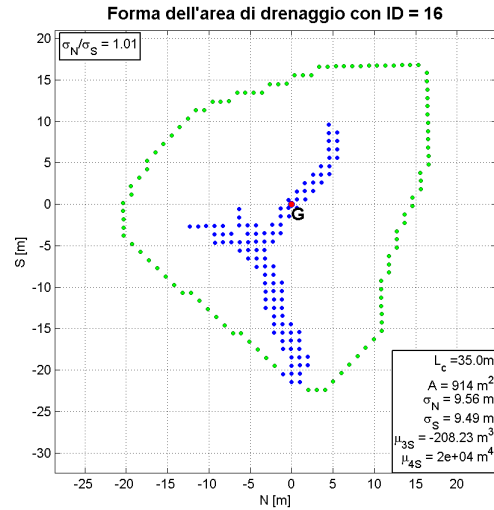


(f) Forma dell'area con $ID = 14$.

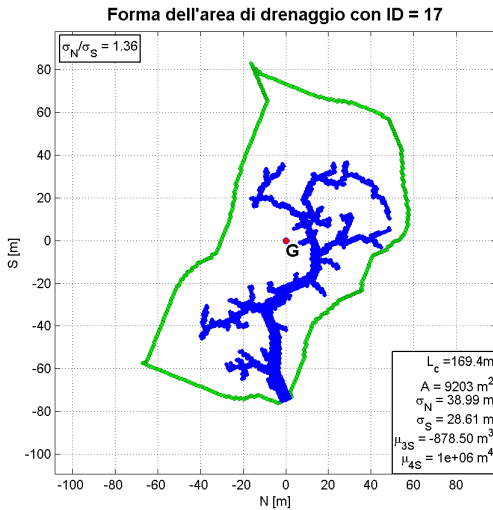
Figura B.1: Forma delle aree di drenaggio di $9 \leq ID \leq 14$, rispetto al riferimento (n, s) .



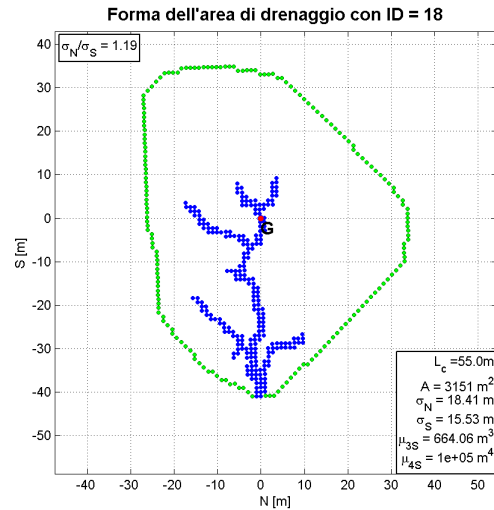
(a) Forma dell'area con ID = 15.



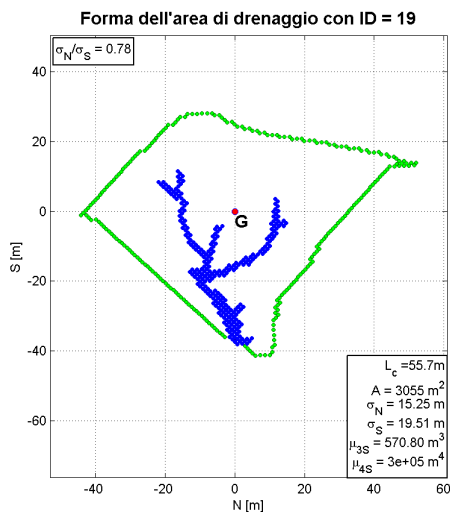
(b) Forma dell'area con ID = 16.



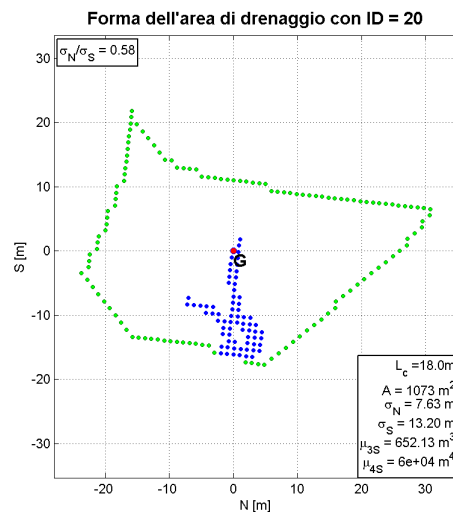
(c) Forma dell'area con ID = 17.



(d) Forma dell'area con ID = 18.

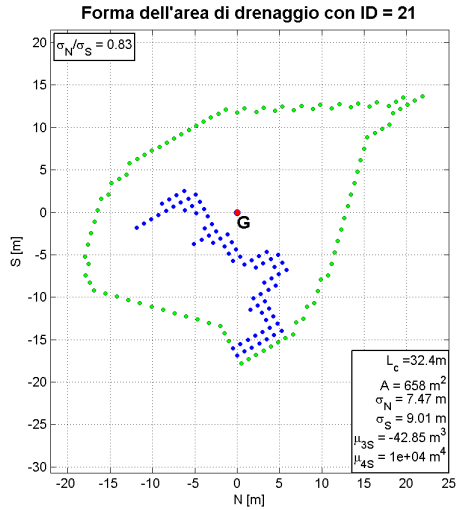


(e) Forma dell'area con ID = 19.

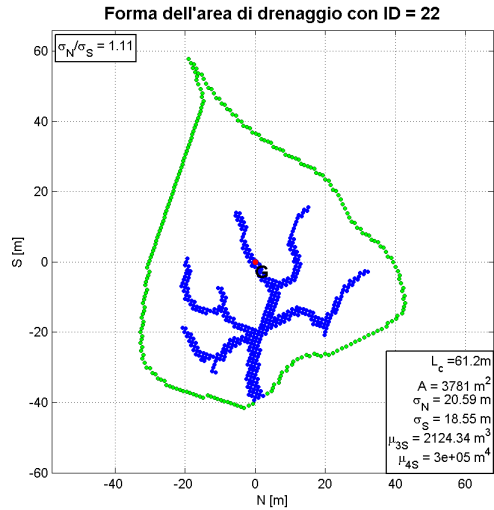


(f) Forma dell'area con ID = 20.

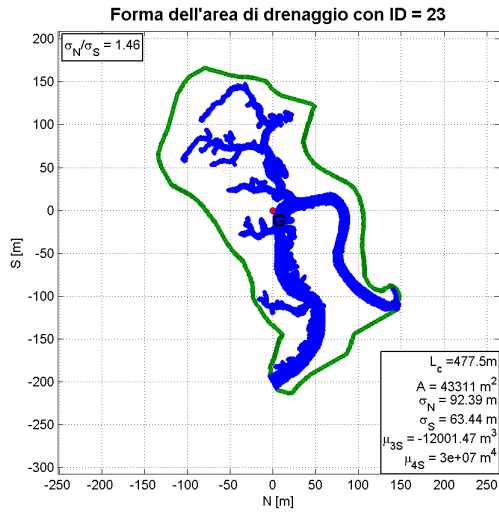
Figura B.2: Forma delle aree di drenaggio di $15 \leq ID \leq 20$, rispetto al riferimento (n, s) .



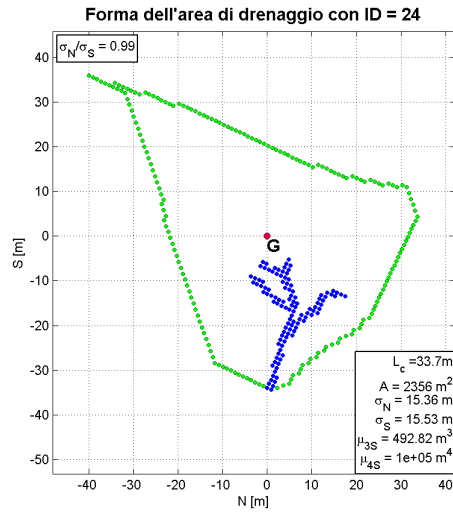
(a) Forma dell'area con $ID = 21$.



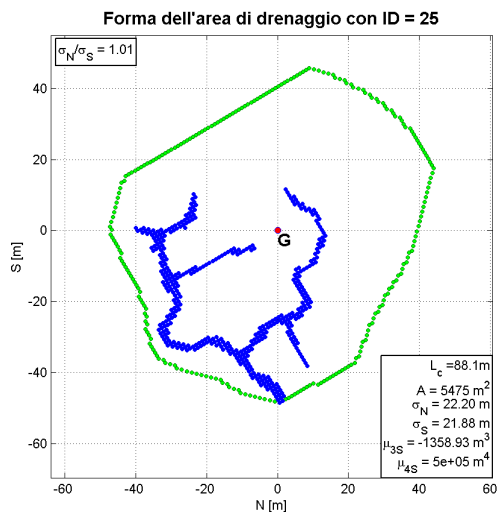
(b) Forma dell'area con $ID = 22$.



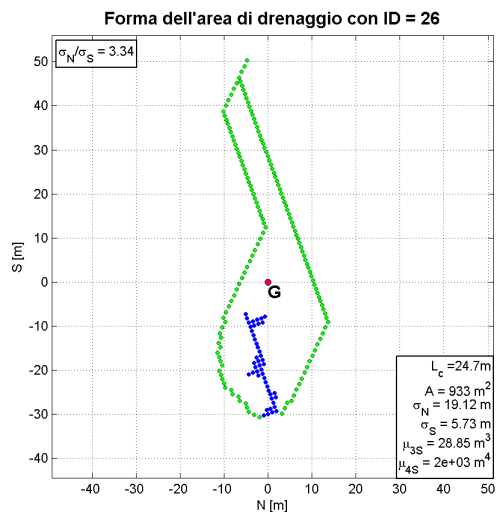
(c) Forma dell'area con $ID = 23$.



(d) Forma dell'area con $ID = 24$.

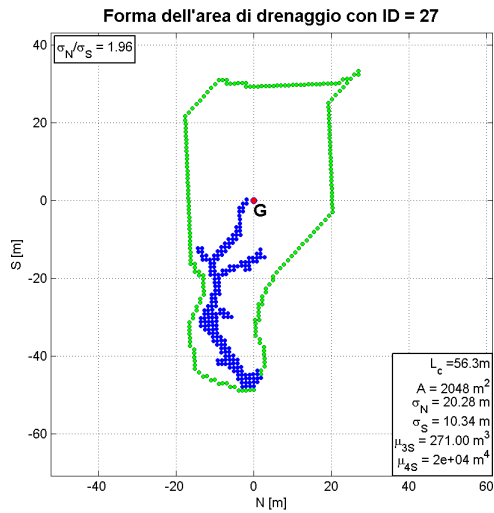


(e) Forma dell'area con $ID = 25$.

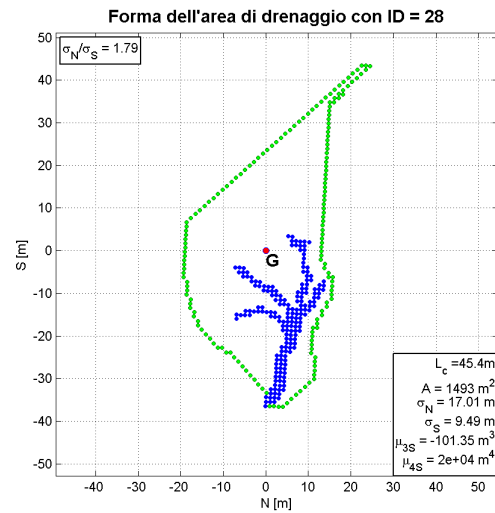


(f) Forma dell'area con $ID = 26$.

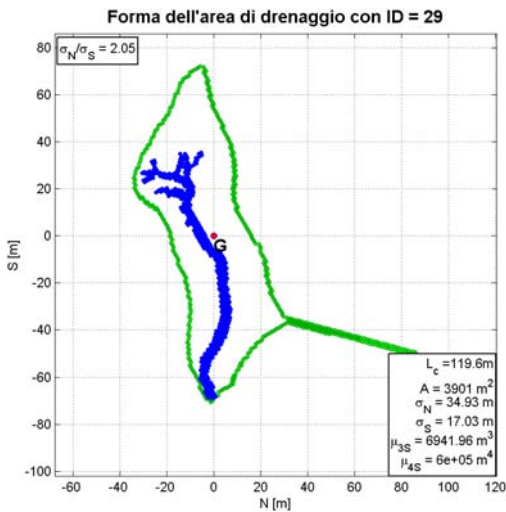
Figura B.3: Forma delle aree di drenaggio di $21 \leq ID \leq 26$, rispetto al riferimento (n, s) .



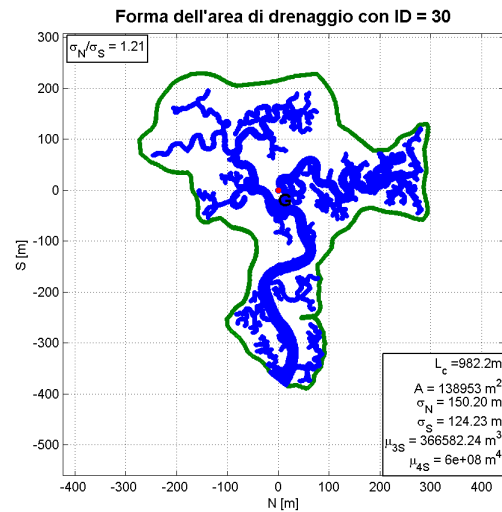
(a) Forma dell'area con ID = 27.



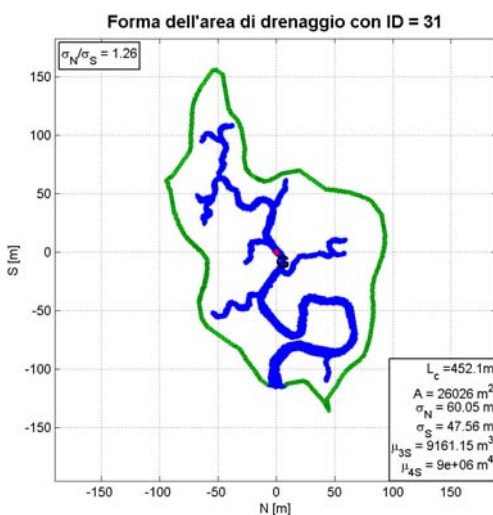
(b) Forma dell'area con ID = 28.



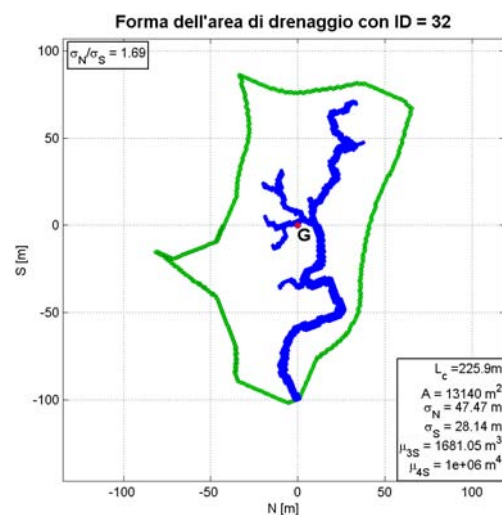
(c) Forma dell'area con ID = 29.



(d) Forma dell'area con ID = 30.

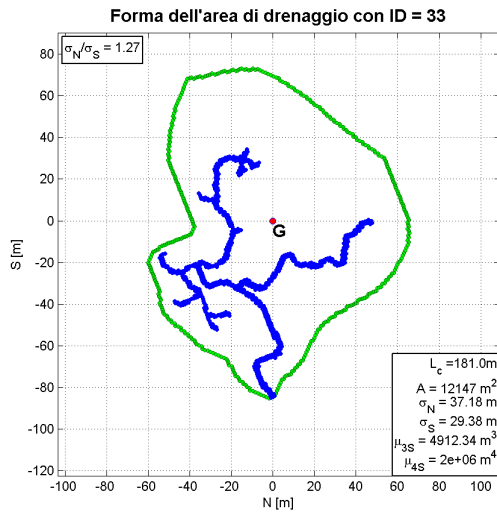


(e) Forma dell'area con ID = 31.

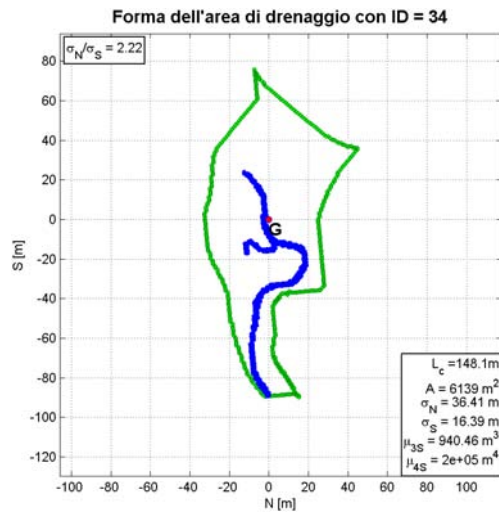


(f) Forma dell'area con ID = 32.

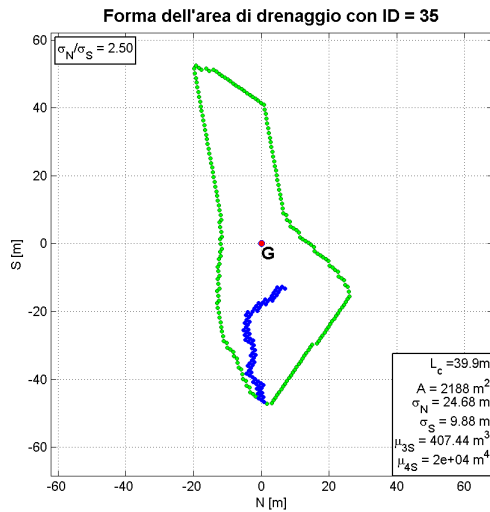
Figura B.4: Forma delle aree di drenaggio di $27 \leq ID \leq 32$, rispetto al riferimento (n, s) .



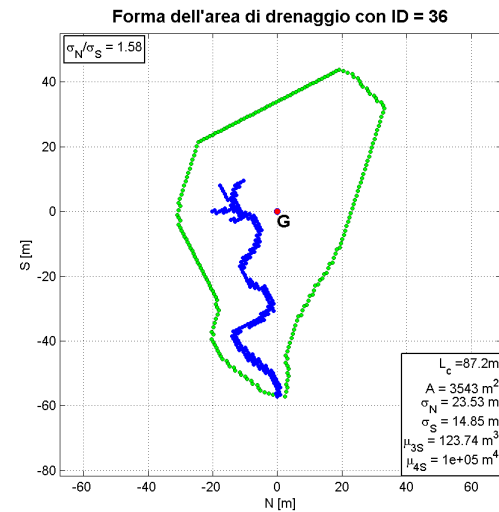
(a) Forma dell'area con ID = 33.



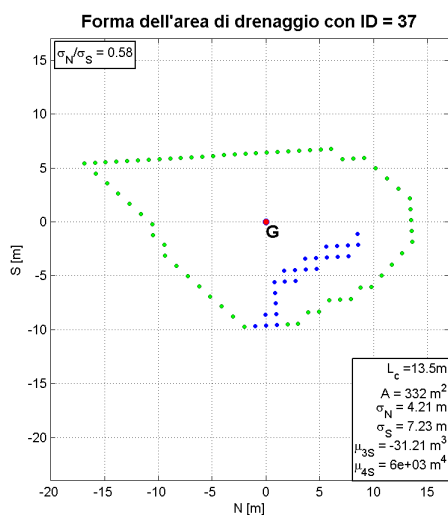
(b) Forma dell'area con ID = 34.



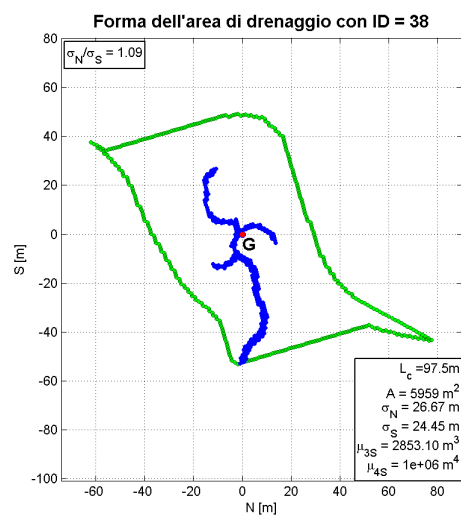
(c) Forma dell'area con ID = 35.



(d) Forma dell'area con ID = 36.

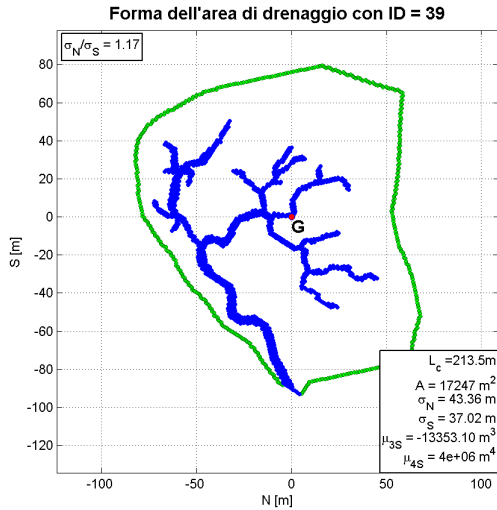


(e) Forma dell'area con ID = 37.

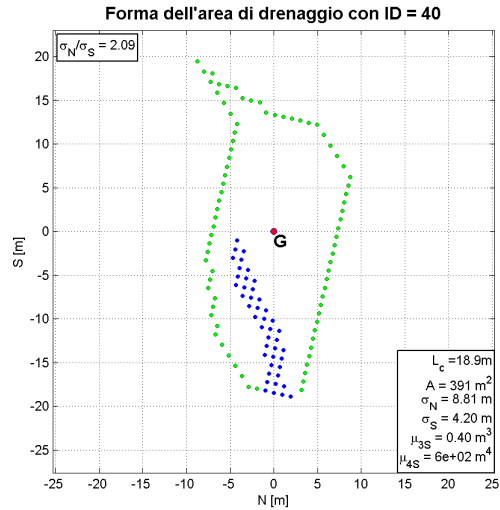


(f) Forma dell'area con ID = 38.

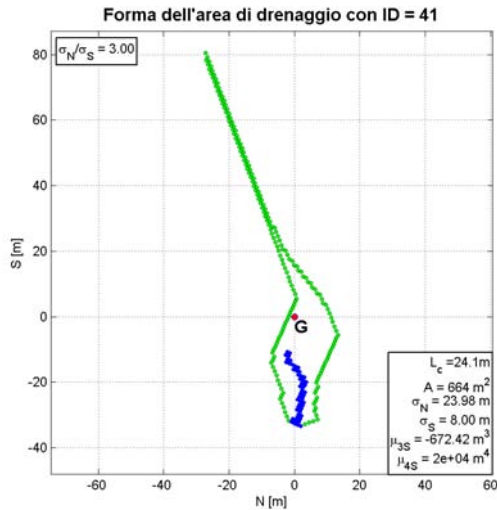
Figura B.5: Forma delle aree di drenaggio di $33 \leq ID \leq 38$, rispetto al riferimento (n, s) .



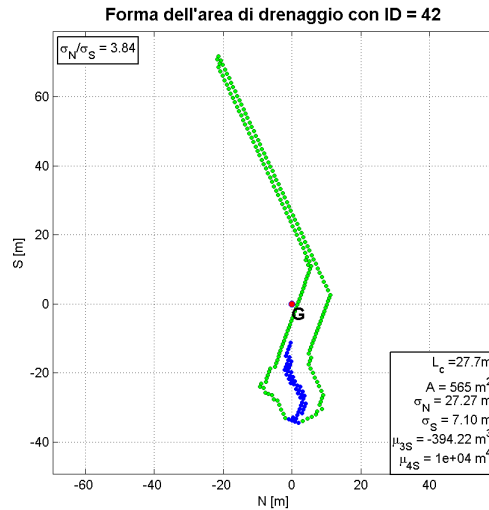
(a) Forma dell'area con ID = 39.



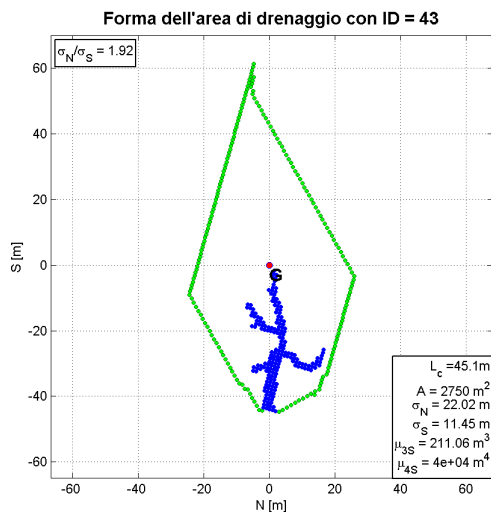
(b) Forma dell'area con ID = 40.



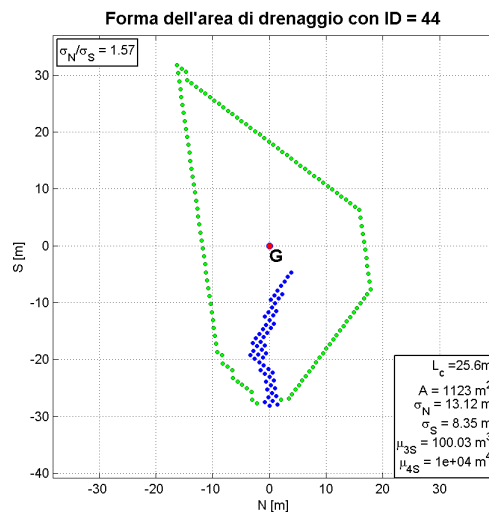
(c) Forma dell'area con ID = 41.



(d) Forma dell'area con ID = 42.

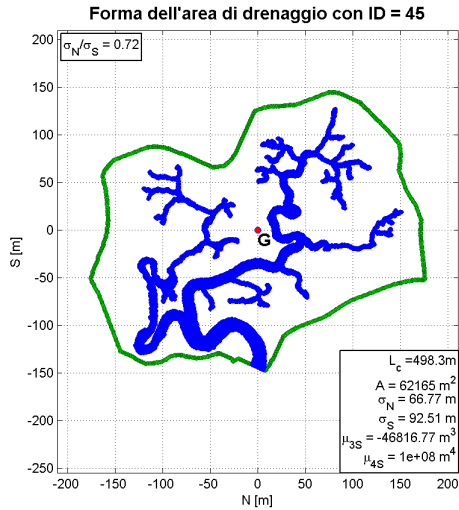


(e) Forma dell'area con ID = 43.

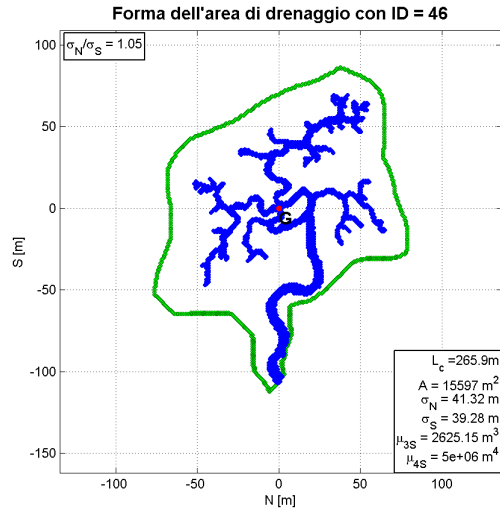


(f) Forma dell'area con ID = 44.

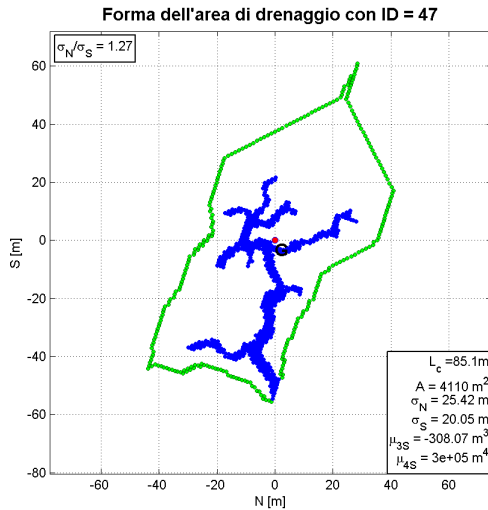
Figura B.6: Forma delle aree di drenaggio di $39 \leq ID \leq 44$, rispetto al riferimento (n, s) .



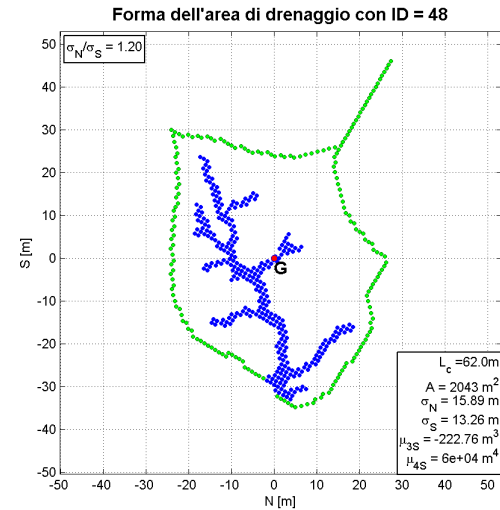
(a) Forma dell'area con ID = 45.



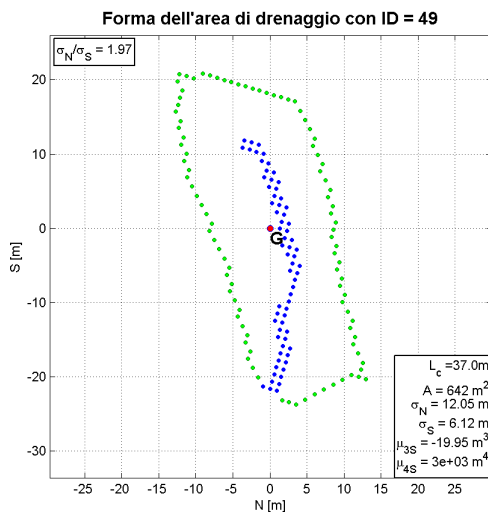
(b) Forma dell'area con ID = 46.



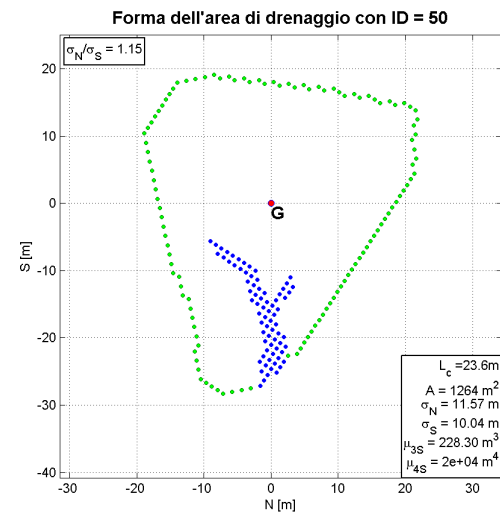
(c) Forma dell'area con ID = 47.



(d) Forma dell'area con ID = 48.

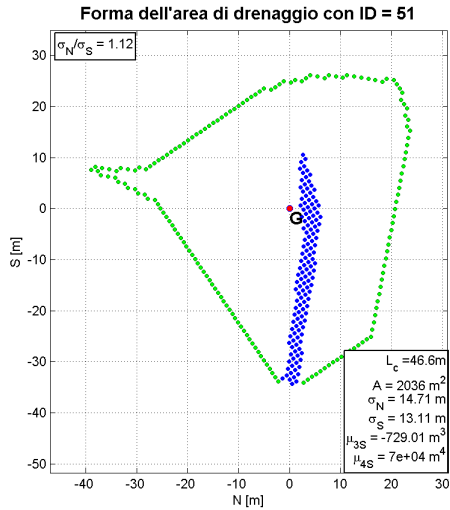


(e) Forma dell'area con ID = 49.

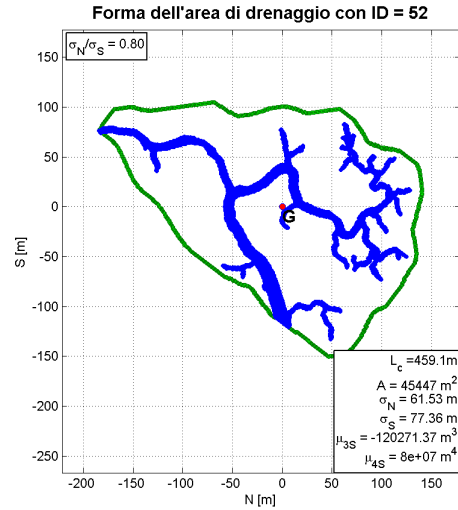


(f) Forma dell'area con ID = 50.

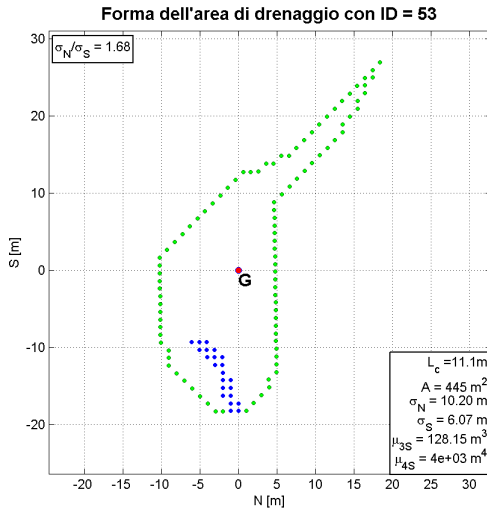
Figura B.7: Forma delle aree di drenaggio di $45 \leq ID \leq 50$, rispetto al riferimento (n, s) .



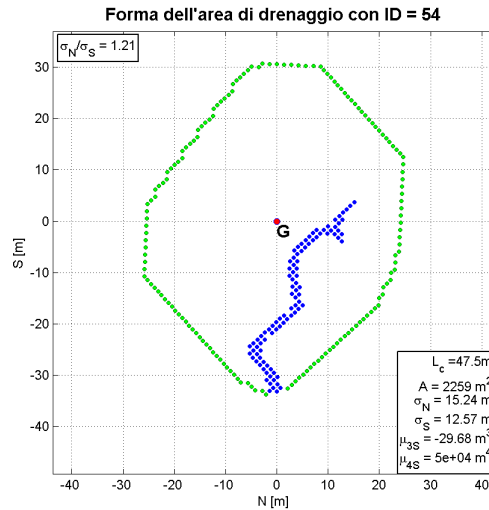
(a) Forma dell'area con ID = 51.



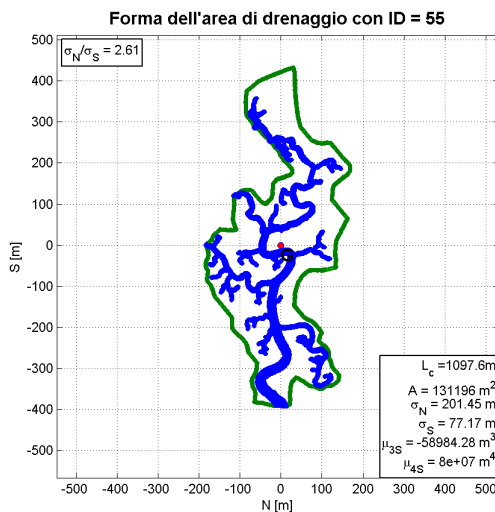
(b) Forma dell'area con ID = 52.



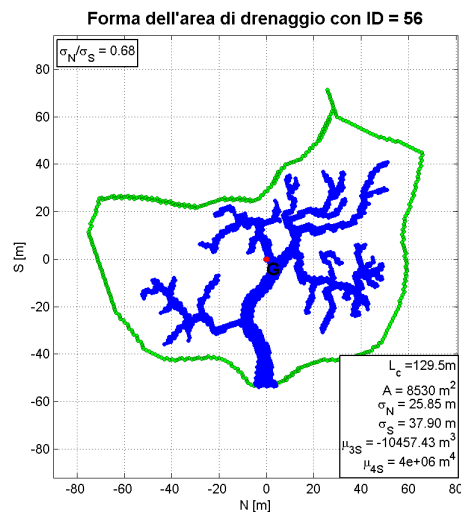
(c) Forma dell'area con ID = 53.



(d) Forma dell'area con ID = 54.

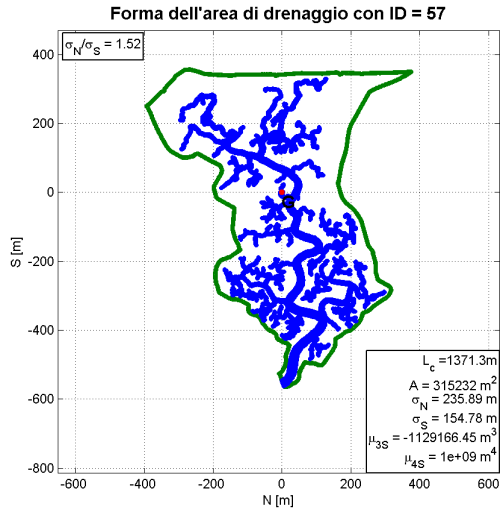


(e) Forma dell'area con ID = 55.

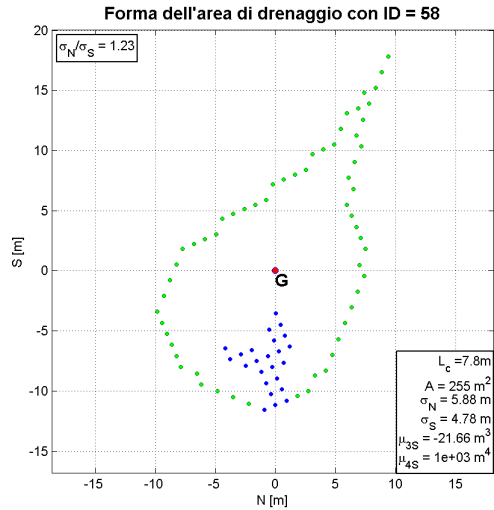


(f) Forma dell'area con ID = 56.

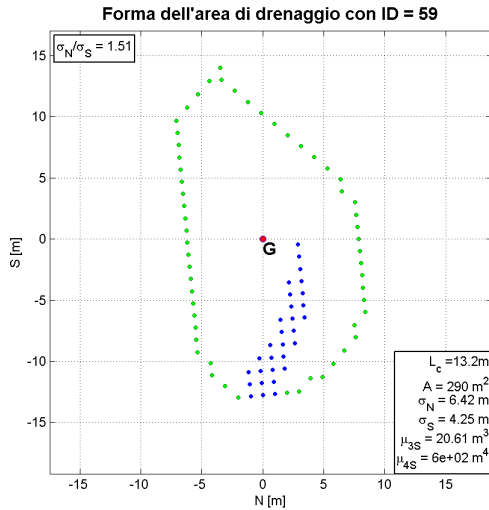
Figura B.8: Forma delle aree di drenaggio di $51 \leq ID \leq 56$, rispetto al riferimento (n, s) .



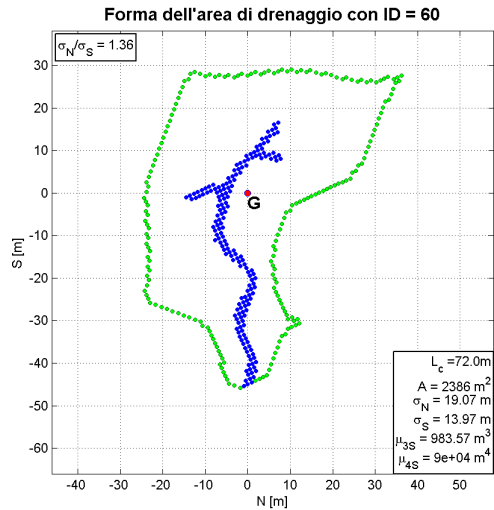
(a) Forma dell'area con ID = 57.



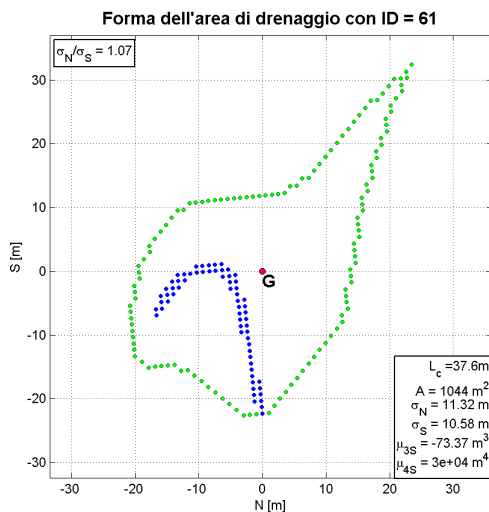
(b) Forma dell'area con ID = 58.



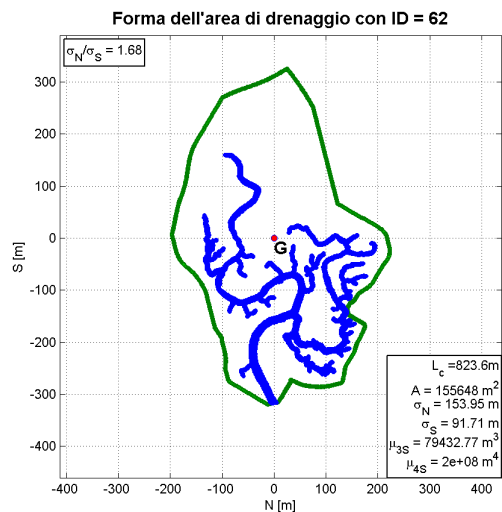
(c) Forma dell'area con ID = 59.



(d) Forma dell'area con ID = 60.

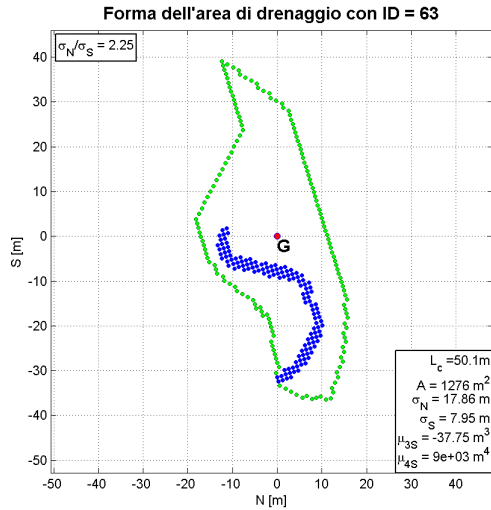


(e) Forma dell'area con ID = 61.

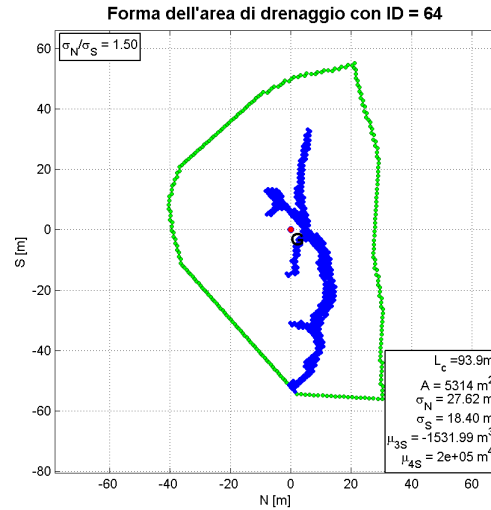


(f) Forma dell'area con ID = 62.

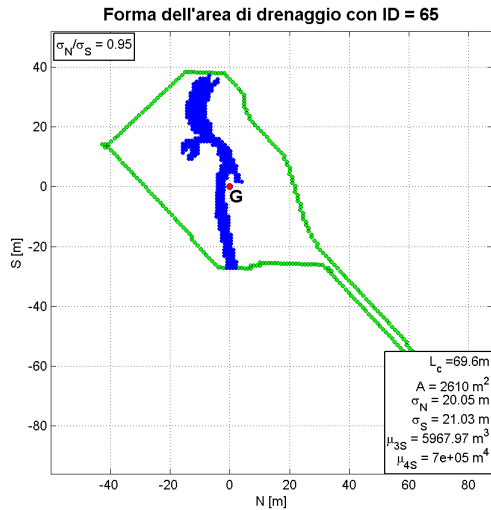
Figura B.9: Forma delle aree di drenaggio di $57 \leq ID \leq 62$, rispetto al riferimento (n, s) .



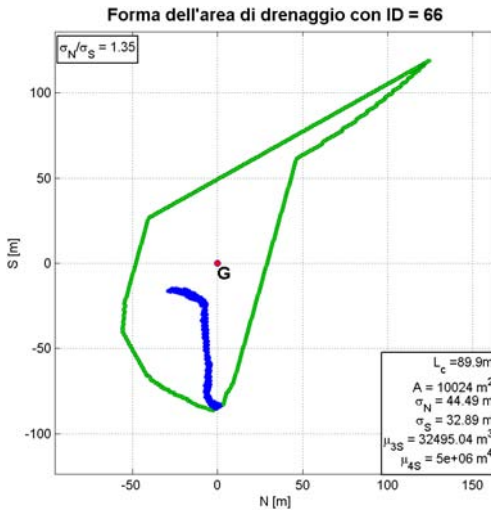
(a) Forma dell'area con ID = 63.



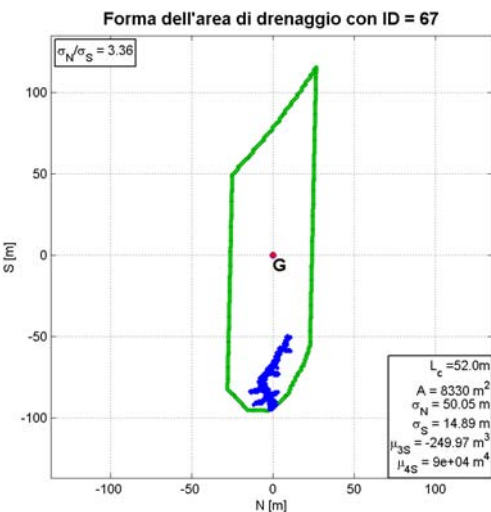
(b) Forma dell'area con ID = 64.



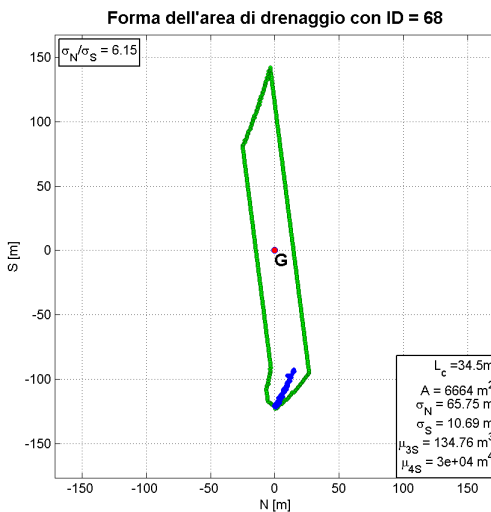
(c) Forma dell'area con ID = 65.



(d) Forma dell'area con ID = 66.

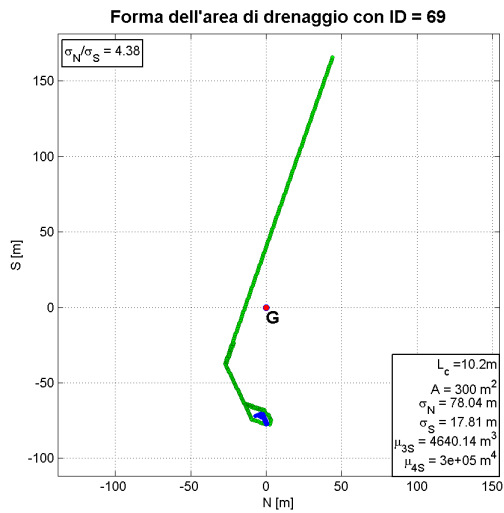


(e) Forma dell'area con ID = 67.

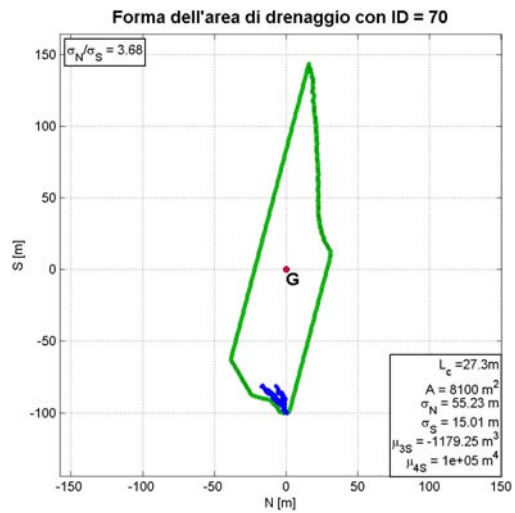


(f) Forma dell'area con ID = 68.

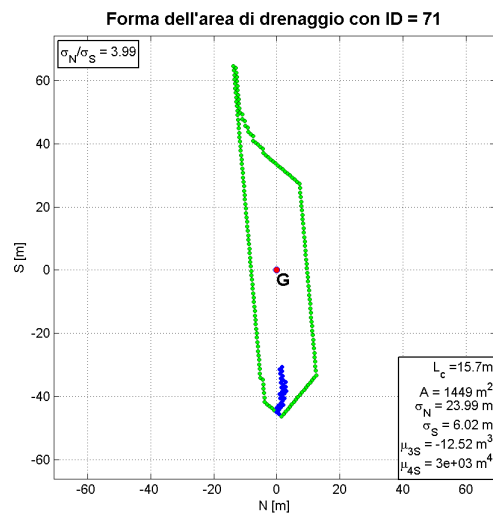
Figura B.10: Forma delle aree di drenaggio $63 \leq ID \leq 68$, rispetto al riferimento (n, s) .



(a) Forma dell'area con ID = 69.



(b) Forma dell'area con ID = 70.



(c) Forma dell'area con ID = 71.

Figura B.11: Forma delle aree di drenaggio di $69 \leq ID \leq 71$, rispetto al riferimento (n, s) .

Bibliografia

- Bayliss-Smith, T.P. et al. (1979). «Tidal flows in salt marsh creeks». In: *Estuarine and Coastal Marine Science* 9.3, pp. 235–255. ISSN: 0302-3524.
- Boon, John D. (1975). «Tidal discharge asymmetry in a salt marsh drainage system». In: *Limnology and Oceanography* 20.1, pp. 71–80.
- D’Alpaos, Andrea (2001). «Contributi allo studio idrodinamico e morfodinamico di lagune a marea». Tesi di dottorato. Padova: Dipartimento IMAGE, Università degli Studi di Padova.
- D’Alpaos, Luigi (2010). *Fatti e misfatti di idraulica lagunare: la laguna di Venezia dalla diversione dei fiumi alle nuove opere alle bocche di porto*. Memorie. Istituto veneto di scienze, lettere ed arti (Classe di scienze fisiche, matematiche e naturali). ISBN: 9788895996219.
- Dronkers, J. J. (1964). *Tidal Computations in Rivers and Coastal Waters*. Amsterdam: North Holland Publishing Company.
- esri, cur. (2014). *Overview Geographic Information System*. URL: http://www.esri.com/what-is-gis/overview#overview_panel.
- Fagherazzi, Sergio et al. (1999). «Tidal networks: 1. Automatic network extraction and preliminary scaling features from digital terrain maps». In: *Water Resources Research* 35.12, pp. 3891–3904. ISSN: 1944-7973.
- Ferronato, Massimiliano (2005). *Progetto numerico al calcolatore*. Dispense online per il Corso di Metodi Numerici per l’Ingegneria di Giuseppe Gambolati. Università degli Studi di Padova. URL: <http://dispense.dmsa.unipd.it/gambolati/metodi/gcm/gcm.html>.
- French, J. R. e D. R. Stoddart (1992). «Hydrodynamics of salt marsh creek systems: Implications for marsh morphological development and material exchange». In: *Earth Surface Processes and Landforms* 17.3, pp. 235–252. ISSN: 1096-9837.
- Friedrichs, Carl T. e Ole S. Madsen (1992). «Nonlinear diffusion of the tidal signal in frictionally dominated embayments». In: *Journal of Geophysical Research: Oceans* 97.C4, pp. 5637–5650. ISSN: 2156-2202.
- Gambolati, Giuseppe (2002). *Lezioni di metodi numerici per ingegneria e scienze applicate*. Seconda edizione riveduta e corretta. Padova: Cortina. ISBN: 9788877841865.
- Healey, R. G. et al. (1981). «Velocity variations in salt marsh creeks, Norfolk, England». In: *Estuarine, Coastal and Shelf Science* 13.5, pp. 535–545.
- Kopecky, Karen A. (2005). *Calling C and Fortran Programs from MATLAB*. Lezioni di C++. Department of Economics, University of Rochester. URL: <http://www.karenkopecky.net/Teaching/Cclass/MatlabCallsC.pdf>.

- Lanzoni, Stefano e Giovanni Seminara (1998). «On tide propagation in convergent estuaries». In: *Journal of Geophysical Research: Oceans* 103.C13, pp. 30793–30812. ISSN: 2156-2202.
- LeBlond, Paul H. (1978). «On tidal propagation in shallow rivers». In: *Journal of Geophysical Research: Oceans* 83.C9, pp. 4717–4721. ISSN: 2156-2202.
- Lorenz, H. A. (1926). *Verslag Staatcommissie Zuiderzee 1918–1926*. Report of the Government Zuiderzee Commission. Algemeene Landsdrukkerij.
- Lynn, Leonard A, Albert C. Hine e Mark E. Luther (1995). «Surficial sediment transport and deposition processes in a *Juncus roemerianus* marsh, west-central Florida». In: *Journal of Coastal Research*, pp. 322–336.
- Marani, Marco et al. (2003). «On the drainage density of tidal networks». In: *Water Resources Research* 39.2, n/a–n/a. ISSN: 1944-7973.
- Montgomery, David R. e William E. Dietrich (1988). «Where do channels begin?» In: *Nature* 336 (6196), pp. 232–234.
- (1992). «Channel initiation and the problem of landscape scale». In: *Science* 255.5046, pp. 826–830.
- Myrick Robert M.; Leopold, Luna Bergere (1963). *Hydraulic geometry of a small tidal estuary*. Professional Paper 422-B. U.S. Govt. Print. Off.
- Pethick, J. S. (1980). «Velocity surges and asymmetry in tidal channels». In: *Estuarine and Coastal Marine Science* 11.3, pp. 331–345. ISSN: 0302-3524.
- Rado, Marco (2014). «Evoluzione temporale delle reti di canali a marea». Tesi di laurea magistrale. Padova: Dipartimento ICEA, Università degli Studi di Padova.
- Rinaldo, Andrea, William E. Dietrich et al. (1995). «Geomorphological signatures of varying climate». In: *Nature* 374.6523, pp. 632–635.
- Rinaldo, Andrea et al. (1999a). «Tidal networks: 2. Watershed delineation and comparative network morphology». In: *Water Resources Research* 35.12, pp. 3905–3917. ISSN: 1944-7973.
- (1999b). «Tidal networks: 3. Landscape-forming discharges and studies in empirical geomorphic relationships». In: *Water Resources Research* 35.12, pp. 3919–3929. ISSN: 1944-7973.
- Rodriguez-Iturbe, Ignacio e Andrea Rinaldo (1997). *Fractal River Basins: Chance and Self-Organization*. New York: Cambridge University Press.
- Shi, Z., J. S. Pethick e K. Pye (1995). «Flow structure in and above the various heights of a saltmarsh canopy: a laboratory flume study». In: *Journal of Coastal Research*, pp. 1204–1209.
- Shuttelaars, H.M. e H.E. De Swart (1996). «An idealized long-term morphodynamic model of a tidal embayment». In: *Eur.J. Mech. B/Fluids* 15.1, pp. 55–80.
- Straaten, L.M.J.U. van (1954). *Composition and Structure of Recent Marine Sediments in the Netherlands*. Leidse Geologische Medelingen. Rijksgeologisch-mineralogisch museum.
- The MathWorks, cur. (2014a). *Create Fortran Source MEX-File*. Documentation Center. URL: http://www.mathworks.it/it/help/matlab/matlab_external/create-fortran-source-mex-file.html.
- cur. (2014b). *Matlab External Interfaces*. URL: http://www.mathworks.com/help/pdf_doc/matlab/apiext.pdf.

- cur. (2014c). *Supported and Compatible Compilers – Release 2014a*. Documentation Center. URL: <http://www.mathworks.it/support/compilers/R2014a/index.html>.
 - cur. (2014d). *Upgrade MEX-Files to Use 64-Bit API*. Documentation Center. URL: http://www.mathworks.it/it/help/matlab/matlab_external/upgrading-mex-files-to-use-64-bit-api.html.
- Van Dongeren, A.R. e H.J. De Vriend (1994). «A model of morphological behaviour of tidal basins». In: *Coastal Engineering* 22.3, pp. 287–310.
- Zimmerman, JTF (1982). «On the Lorentz linearization of a quadratically damped forced oscillator». In: *Physics Letters A* 89.3, pp. 123–124.

Ringraziamenti

Desidero ringraziare il prof. Stefano Lanzoni per la fiducia accordatami, per la pazienza dimostrata nel correggere le bozze di questa tesi e per la grande disponibilità dimostrata nei miei confronti.

Desidero ringraziare il prof. Andrea D'Alpaos per la gentilezza e per il tempo dedicato a rispondere alle mie numerose domande sul funzionamento del programma per la soluzione del modello matematico.

Ringrazio tutti i colleghi studenti che mi hanno accompagnato in questi anni universitari, dividendo con me le gioie e le fatiche legate allo studio, hanno contribuito a trasformare questa esperienza in un ricordo che mi accompagnerà nel resto della mia vita.

Un grazie va anche a tutti quei professori che, con il lavoro di ogni giorno, tentano di trasmettere la loro passione agli studenti. In particolar modo vorrei ricordare il prof. Luigi D'Alpaos e l'ing. Carlo Salmaso che hanno aperto la porta delle proprie case per ospitare noi studenti, generando un momento di aggregazione che ha contribuito a rafforzare il già forte legame instaurato tra noi studenti.

Ringrazio i nonni Maria e Sergio (nel ricordo dei nonni Bruna e Antonio) per la sempre pronta ospitalità nei momenti in cui lo studio era più intenso. Loro mi hanno insegnato a non mollare mai e, se sono giunto al termine di questa esperienza, lo devo anche ai loro insegnamenti.

Ringrazio Chiara per aver letto e corretto, *volontariamente*, le primissime bozze di questa tesi.

Ringrazio mio fratello Thomas per tutto il supporto morale e materiale fornitomi per tutta la durata dei miei studi universitari.

Un grazie speciale va a Jessica per l'affetto e la pazienza dimostrati nei miei confronti in tutti questi anni in cui ha scelto di stare al mio fianco.

Infine, un grazie immenso va ai miei genitori, Carla e Alfonso: a loro devo tutto e senza di loro non sarebbe mai stato possibile raggiungere questo traguardo.

Padova, 16/07/2014.

Francesco Carraro