



**UNIVERSITÀ  
DI PARMA**

# UNIVERSITÀ DEGLI STUDI DI PARMA

DIPARTIMENTO DI INGEGNERIA E ARCHITETTURA

## **LINEE GUIDA SULL'UTILIZZO DEL CODICE DI CALCOLO "PARFLOOD"**

**(aggiornato al 16/07/2020)**

Capitolo 1. PROGRAMMI DA INSTALLARE .....	2
Capitolo 2. FILE DI INPUT AL MODELLO .....	2
2.1. Batimetria .....	3
2.2. Scabrezza.....	5
2.3. Condizioni iniziali (di livello) .....	7
2.4. Condizioni iniziali (di velocità).....	8
2.5. Condizioni al contorno .....	8
2.6. Generazione di una breccia in evoluzione .....	12
2.7. File di input .....	14
2.8. Multirisoluzione .....	16
2.9. Scrittura dei risultati con DTOUTPUT variabile.....	17
Capitolo 3. FASE DI CALCOLO.....	19
3.1. Alcune istruzioni utili per lanciare le simulazioni .....	19
Capitolo 4. RISULTATI.....	23
4.1. File di output .....	23
4.2. Decode.....	24
4.3. Zippare i risultati .....	29
Appendice A. UTILIZZO DEL CLUSTER UNIPR .....	30

# Capitolo 1. PROGRAMMI DA INSTALLARE

I programmi necessari per la creazione dei file di input al modello, nonché per la visualizzazione dei file di output sono:

- Global Mapper;
- Surfer;
- Notepad++;
- Cygwin (pacchetto “openssh”);
- WinSCP;
- Matlab (eventuale).

Si elencano inoltre alcuni comandi utili di Surfer (si veda il manuale del programma):

- NEW MAP/ADD TO MAP → ContourMap, Base, Post, ClassedPost, 2-GridVector
- GRID → GridData, Blank/AssignNoData, Math, Slice, Mosaic, GridInfo
- MAP TOOLS → Measure, Digitize.

# Capitolo 2. FILE DI INPUT AL MODELLO

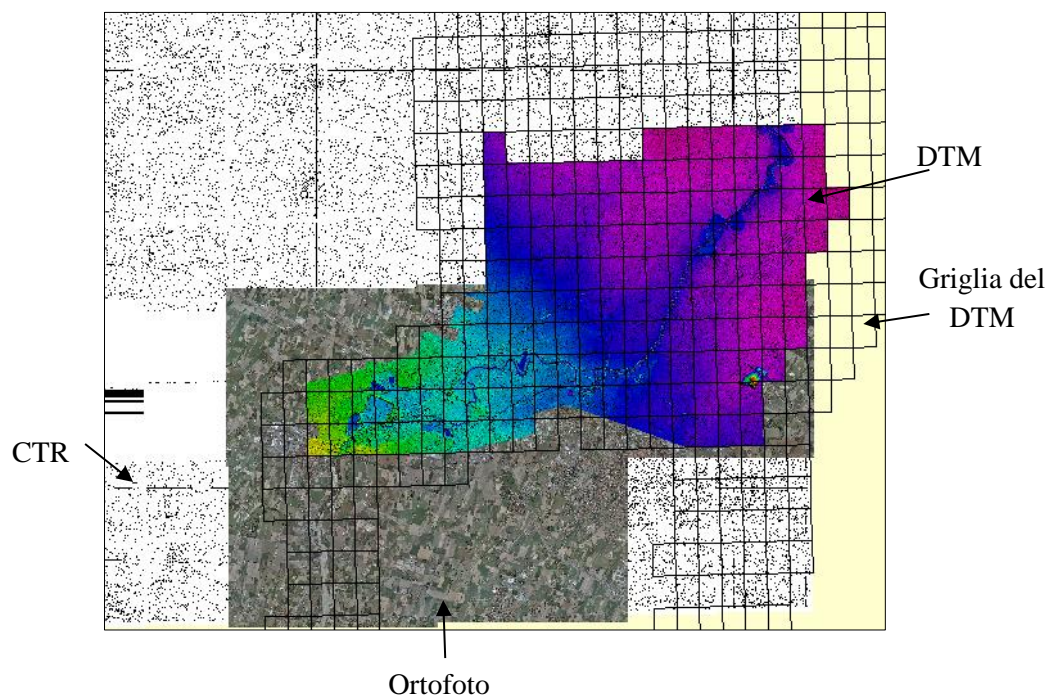
Per risolvere le equazioni alle acque basse il modello necessita di informazioni riguardanti la batimetria e la scabrezza della zona che si intende studiare, nonché le condizioni iniziali e al contorno, richiedendo complessivamente nove file di input:

1. [nome caso].BTM: batimetria;
2. [nome caso].MAN: scabrezza espressa secondo il coefficiente di Manning;
3. [nome caso].INH: condizioni iniziali;
4. [nome caso].BLN: condizioni al contorno;
5. [nome caso].BCC: condizioni al contorno (precisazione del file .BLN);
6. [nome caso].VHX: velocità iniziali lungo l’asse x;
7. [nome caso].VHY: velocità iniziali lungo l’asse y;
8. [nome caso].BRE: creazione di una breccia;
9. [nome caso].PTS: elenco punti per multi-risoluzione.

Se i primi quattro file sopra citati sono indispensabili per qualsiasi simulazione, gli ultimi, in alcuni casi potrebbero anche non comparire: ipotizzando ad esempio di avere una condizione di acqua inizialmente ferma, appare evidente come non esistano condizioni iniziali di velocità e quindi i file .VHX e .VHY non siano richiesti.

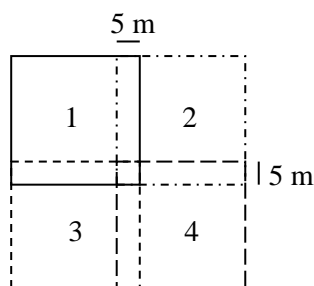
## 2.1. Batimetria

Il file di input .BTM, relativo alla batimetria, contiene la descrizione del territorio che si intende studiare ed ha origine dall'elaborazione di tre tipologie di dati: DTM, ortofoto, CTR. I tre prodotti sopra descritti sono gestiti con l'ausilio del software Global Mapper 13 (Figura 3.1).



*Figura 3.1 Materiale fotogrammetrico utile per la costruzione della batimetria*

Esportando l'intera area di interesse da Global Mapper si otterrebbe un file con dimensioni troppo onerose per essere gestite facilmente nelle successive operazioni di calcolo e per questo motivo è opportuno suddividere la superficie in diversi blocchi, garantendo una sovrapposizione tra le singole aree pari alla dimensione della cella scelta per la griglia (ad esempio sovrapposizione di 5 m per una griglia a passo 5 m, Figura 3.2.).



*Figura 3.2 Sovrapposizione dei primi quattro blocchi*

I blocchi esportati da Global Mapper in formato Arc ASCII Grid, aventi spaziatura di 1 m, devono essere convertiti con Surfer in Surfer 6 Text Grid per poter essere processati con Matlab.

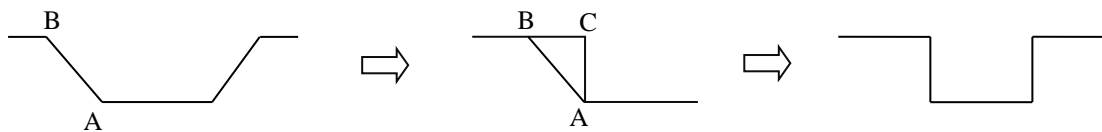
### 2.1.1. Correzione delle arginature

L'estrapolazione di una griglia a passo 5 m oppure 10 m da un DTM a passo 1 m comporta la perdita di informazioni, in quanto si conserva soltanto 1 punto ogni 5 oppure ogni 10.

La problematica più importante è connessa alla mancata conservazione della quota di alcuni tratti arginali e di muretti di contenimento aventi dimensioni minori di quelle della griglia scelta.

Al fine di ripristinare il corretto assetto arginale si ricorre ad uno script di MATLAB, il cui funzionamento può essere così sinteticamente descritto: definito il passo di campionamento  $n$ , lo script costruisce un quadrato prendendo un punto ogni  $n$ , calcola gli scarti e attribuisce al punto centrale il valor massimo tra quelli adiacenti considerati.

Da questa procedura occorre però escludere l'alveo del fiume, per evitare un innalzamento non corrispondente alla realtà delle quote spondali e una conseguente riduzione della portata transitante. Come rappresentato in Figura 3.3 infatti, al punto A verrebbe assegnato il valor massimo più vicino, ossia B, quindi A traslerebbe verso l'alto fino a C. L'alveo fluviale assumerebbe quindi una forma completamente diversa: questo spiega l'esigenza di escluderlo dalla procedura di calcolo svolta dallo script.



*Figura 3.3 Dimostrazione del motivo per cui escludere l'alveo dalla procedura di correzione delle arginature*

Per permettere al programma di procedere con i calcoli occorre posizionare nella stessa cartella che contiene lo script i seguenti file:

- File .grd: blocco della batimetria a passo 1 m che si intende diradare;
- File .bln: porzione dell'alveo fluviale che si intende escludere dal calcolo. Mediante il software Surfer si disegna in senso antiorario un poligono che ricalchi l'alveo fluviale di magra e che quindi resti all'interno delle arginature;
- File .txt: l'elenco dei rilevati da inserire, espressi in termine di coordinate planimetriche e altimetriche. Il file è così strutturato:
  - Numero dei rilevati;
  - Numero dei segmenti impiegati per schematizzare il rilevato;
  - Valore delle coordinate x, y, z di ogni punto.

Dopo aver definito il passo della griglia a bassa e ad alta risoluzione, lo script fornisce come risultato la nuova batimetria corretta del blocco inserito a 1 m. Il procedimento viene ripetuto per tutti i blocchi variando di volta in volta soltanto il primo file .grd.

Terminata la fase di calcolo, con l'ausilio del software Surfer si effettua un "mosaico" dei nuovi blocchi ottenuti al fine di ricavare un'unica griglia.

**I formati compatibili con il modello PARFLOOD sono i .grd seguenti: Surfer 6 Text Grid, Surfer 6 Binary Grid, Surfer 7 Binary Grid.** Nel primo tutti i valori sono scritti in caratteri ascii, mentre negli altri due in binario (float per versione 6 e double per versione 7). Si noti che le due tipologie "binary" hanno dimensioni minori (specialmente la versione 6) e sono più facilmente caricabili/visualizzabili sia da Surfer che da Parflood. Il loro utilizzo è dunque fortemente consigliato per i file di input. Si tenga invece conto che per le elaborazioni in Matlab va sempre usato il formato Surfer 6 Text Grid.

Alla fine, l'estensione del file generato deve essere cambiata, rinominando semplicemente .BTM anzichè .grd.

## 2.2. Scabrezza

Il file di ingresso al modello .MAN racchiude, come suggerisce il nome stesso, le informazioni di scabrezza nella formulazione di Manning.

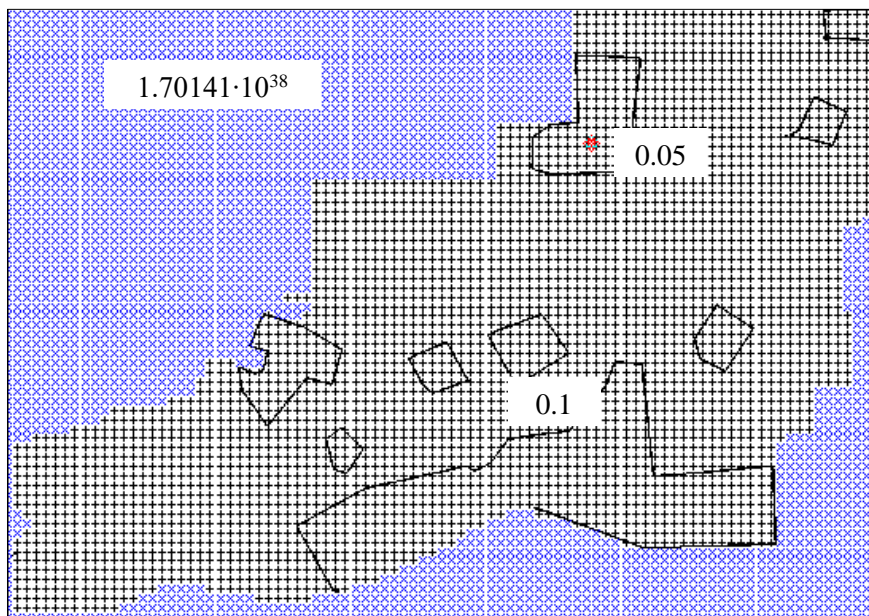
Le dimensioni del file .MAN devono essere **identiche a quelle della batimetria precedentemente costruita**: ipotizzato un file .BTM di 1000 righe per 2000 colonne, il file .MAN dovrà avere dimensioni 1000 x 2000. Se nel file della batimetria ciascun punto della griglia è individuato dalle coordinate planimetriche x ed y della stessa griglia cartesiana e dalla coordinata z rappresentativa dell'altimetria, nel file della scabrezza le coordinate x ed y restano immutate mentre alla z viene associato il coefficiente di Manning n. Tale analogia permette di ottenere il file .MAN trasformando semplicemente le coordinate z del file .BTM secondo quanto segue:

1. Apertura del file prova.BTM con Surfer;
2. Comando Grid-Math:
  - Add Grid (carica file desiderato): prova.BTM;
  - Digitare l'operazione che si vuole sia svolta affinché i diversi valori delle quote z diventino un unico valore di scabrezza n. Tra le diverse operazioni possibili si riporta a titolo esemplificativo la seguente:  $\text{if } (A > 5000, A, n)$   
Nel comando si afferma che se i valori z della matrice A (prova.BTM) sono superiori a 5000, nella nuova griglia resteranno i valori z di A, in caso contrario nella nuova griglia i valori z saranno posti uguali a n (sia esso 0.1, 0.5, ...)

- Output grid file: nome del file contenente le scabrezze che viene costruito, salvato in uno dei formati GRD di Surfer compatibili;
3. Si rinomi il file ottenuto, cambiando l'estensione da .grd a .MAN, ottenendo prova.MAN

Nel caso in cui si volessero utilizzare diversi valori della scabrezza, ad esempio per distinguere le zone urbane, oppure l'alveo di magra del corso d'acqua o altri elementi, la procedura da seguire sarebbe la seguente:

1. Con l'ausilio di ortofoto o della batimetria si traccino diversi poligoni al fine di delimitare le aree di interesse (ad esempio un centro abitato);
2. Sbiancatura del centro abitato (resta solo il centro abitato);
3. Mosaic dei diversi centri abitati, dando alla nuova griglia le dimensioni 1000 x 2000;
4. Attribuzione del valore di scabrezza  $n_1$  alla griglia dei centri abitati (come visto sopra);
5. Creazione di una griglia con ovunque scabrezza  $n_2$  (come visto sopra);
6. Mosaic della griglia dei centri abitati con scabrezza  $n_1$  con quella  $n_2$ , scegliendo l'opzione opportuna per fare in modo che nelle zone di sovrapposizione prevalga la scabrezza dei centri abitati;
7. Si rinomi il file ottenuto, cambiando l'estensione da .grd a .MAN, ottenendo prova.MAN



*Figura 3.6 Esempio di un file .MAN*

## 2.3. Condizioni iniziali (di livello)

### 2.3.1. Alveo asciutto

Nel caso in cui si voglia propagare un'onda di piena in ingresso ad un tratto fluviale, partendo da una condizione di alveo asciutto, occorre prestare attenzione al fatto che è comunque necessaria almeno qualche cella “bagnata” (ossia in cui la quota dell'INH è superiore a quella della BTM) in corrispondenza della stringa della condizione al contorno di monte per poter permettere l'ingresso di portata. In Surfer, si può ad esempio:

1. Usare il comando Grid>Math sulla BTM (matrice A), sfruttando l'operazione:  $A+h$  (dove h è un numero che rappresenta la profondità idrica, es. 1 m);
2. Tracciare col comando Map>Digitize, un poligono attorno all'alveo inciso nella sezione di monte, e salvare il file digitalizzato in formato.bln.;
3. Modificare l'intestazione del file.bln sostituendo l'indicazione della prima riga da “N,1” a “N,0” (dove N è il numero dei punti del poligono);
4. Usare il comando Grid>Blank (o AssignNoData nelle versioni più recenti di Surfer) per “sbiancare” tutti i valori della griglia al di fuori del poligono. La griglia così ottenuta può fungere da file INH per la simulazione preliminare (le dimensioni della griglia saranno già identiche a quelle della BTM).

### 2.3.2. Condizioni iniziali risultanti da una simulazione preliminare

In alcuni casi il file .INH può essere ottenuto dopo aver eseguito una simulazione preliminare con alveo asciutto (vedi punto 2.3.1), convertendo semplicemente un file .WSE risultante, e opportunamente decodificato, in .INH. Assunto ad esempio come riferimento l'istante temporale 0010, il file .INH sarà generato:

1. Decodificando con l'opzione decode il file risultante ...-0010.WSE, così da poter ottenere una griglia leggibile in Surfer;
2. Controllando che le dimensioni del file ...-0010.WSE coincidano con quelle del file .BTM, ed in caso contrario effettuando un'operazione di mosaic per aggiustare le dimensioni della griglia decodificata .WSE;
3. Rinominando il file ottenuto ai punti precedenti apponendo l'estensione .INH.

## **2.4. Condizioni iniziali (di velocità)**

### **2.4.1. File .VHX**

Effettuata una simulazione preliminare in cui non compaiono i file .VHX e .VHY, si individua l'istante temporale che sarà la condizione iniziale di una nuova simulazione. Ipotezzando di scegliere l'istante 0010, disponendo dei file di output decodificati ...-0010.DEP e ...-0010.VVX, il file VHX si ottiene moltiplicando in Surfer (con la funzione Math) i due file precedenti.

Eventualmente può essere utile azzerare la velocità nelle celle quasi asciutte (ossia con livelli idrici molto bassi, es.  $< 1$  mm). Per fare questo, nella predisposizione del file tramite la funzione Math di Surfer occorre specificare una condizione del tipo:  $\text{if}(A > 0.001, A * B, 0)$ , avendo importato la griglia .DEP come A e la griglia .VVX come B.

### **2.4.2. File .VHY**

Analogamente a quanto effettuato per il file .VHX, il file .VHY si ottiene moltiplicando (scelto l'istante 0010) i file decodificati ...-0010.DEP e ...-0010.VVY.

## **2.5. Condizioni al contorno**

### **2.5.1. File .BLN**

Il file .BLN, che racchiude le condizioni al contorno, è un file di testo strutturato nel modo seguente:

- Numero di punti;
- Valore delle coordinate x, y di ogni punto e della condizione al contorno.

Il terzo valore che segue le coordinate è l'indicatore della condizione al contorno da applicare al segmento compreso tra il punto in oggetto e il successivo e può essere:

- 21: portata (onda di piena in ingresso lungo la sezione definita dal segmento);
- 1: muro (l'acqua non può proseguire oltre, in quanto il segmento funge da muro insormontabile);
- 2: portata specifica;
- 3: livello (si impone un valore di tirante idrico);
- 4: farfield (l'acqua si allontana senza disturbo, come se vi fosse un terreno pianeggiante indefinito e privo di ostacoli);
- 5: scala di deflusso (sulla sezione rappresentata dal segmento è nota la relazione che fornisce l'andamento delle portate in funzione del tirante idrico).



Con l'ausilio del software Notepad++ il file .BLN risulterà così strutturato (N è il numero di punti):

N

X<sub>A</sub> Y<sub>A</sub> CC<sub>A</sub>

X<sub>B</sub> Y<sub>B</sub> CC<sub>B</sub>

...

X<sub>N</sub> Y<sub>N</sub> CC<sub>N</sub>

In Figura 3.8 si riporta un esempio di file .BLN utilizzato in una simulazione.

Il file è costituito da 25 punti, le condizioni al contorno sono tutte di muro eccetto la condizione al contorno di monte (condizione 21, ossia onda di piena sul primo segmento) e due condizioni di valle (condizione 3, cioè livelli sul segmento 6, e condizione 5, cioè scala di deflusso sul segmento 17).

La costruzione di un file .BLN si articola dunque nel modo seguente:

1. Surfer: tracciamento in senso antiorario del poligono che delimita l'area di calcolo con "digitize". Tale poligono deve rimanere all'interno dei limiti della batimetria: si preferisce comunque non posizionare i punti sul bordo, per evitare problemi di instabilità numerica durante il processo di simulazione;
2. Notepad ++: si importa il file generato da Surfer e si attribuiscono le condizioni al contorno (tra cui le condizioni al contorno di monte e di valle). Si aggiunge in testa il numero complessivo dei punti. Vanno eliminate tutte le virgole.

PO_gamma_Casalm.BLN				
1	25			
2	611703.79738226	4981894.1744775	21	
3	611157.96924291	4981402.5124894	1	
4	610034.54878664	4980459.1414875	1	
5	610964.95028140	4977168.1543052	1	
6	612006.35411737	4973967.2077778	1	
7	629080.95864274	4972738.2732118	3	
8	629923.59678939	4973652.1499334	1	
9	633314.72576215	4974044.0042109	1	
10	637269.15272718	4986517.8456432	1	
11	652759.64323715	4987808.7198524	1	
12	655411.16863976	4990355.5797786	1	
13	657355.41007163	4990608.2547255	1	
14	691876.95180365	4973384.4756789	1	
15	693284.12763963	4972412.0410486	1	
16	705295.44297596	4972622.7219031	1	
17	705907.97666152	4973801.4905946	1	
18	705948.48311265	4973786.2871454	5	
19	705987.91815558	4974053.5684545	1	
20	701455.98026292	4981157.8265563	1	
21	692201.08149387	4984801.0248019	1	
22	685678.58140902	4990559.6284804	1	
23	681682.81559127	4994349.7298811	1	
24	671487.73662982	4994384.7749292	1	
25	664642.04960382	4994367.2524051	1	
26	655769.09903795	4994320.3492501	1	

Figura 3.8 Esempio di file .BLN

## 2.5.2. File .BCC

Assegnando i valori 1 o 4 non occorre specificare altre condizioni, mentre negli altri casi, al file .BLN si aggiunge il file .BCC, il quale racchiude gli andamenti delle onde di piena o delle scale di deflusso, menzionate nel file .BLN. Il file BCC è un file di testo, strutturato secondo quanto riportato schematicamente in Tabella 3.1.

Da sottolineare come le condizioni 21 e 3 siano simili, con la differenza che nel primo caso il modello considera soltanto i valori di portata e trascura quelli di livello, mentre nel secondo valuta solo i livelli a discapito delle portate. Infine non necessariamente dovranno essere presenti contemporaneamente le tre condizioni, alcune potranno ripetersi, altre addirittura non comparire.

Il file .BLN di Figura 3.8 presenta una condizione 21, una 3 ed una 5 che necessitano di un file .BCC (Figura 3.9).

Il valore 3 sottintende le tre condizioni da inserire, la prima delle quali, costituita da 11 elementi è applicata al segmento 1, coerentemente con quanto stabilito nel file .BLN. Trattandosi di una condizione 21 la prima colonna rappresenta il tempo, la seconda i livelli idrici (elemento che

trattandosi di una condizione 21 non sarà letto), la terza la portata e la quarta contiene il valore 0 (questa colonna viene inserita solo per compatibilità con versioni precedenti del codice). Terminata la prima condizione inizia la descrizione della seconda, applicata al segmento 6 e costituita da 2 soli elementi: sono presenti quattro colonne (tempo, livello, due colonne di 0). Infine, la terza condizione (scala di deflusso) è applicata al segmento 17 e costituita da 23 elementi. La prima colonna rappresenta la quota idrica, la seconda la portata, la terza contiene il valore 0 (per compatibilità con versioni precedenti del codice).

Si noti che la scansione temporale con cui si impongono le condizioni di portata/livello non deve essere necessariamente la medesima: l'onda di piena è descritta con intervalli di 1 ora mentre i livelli con intervalli di 10 ore. Occorre però rispettare gli istanti iniziali e finali: entrambe le condizioni partono da 0 s e terminano dopo 36000 s.

Il file .BCC si costruisce nel modo seguente:

1. Excel: calcolo di tempi, livelli e portate;
2. Notepad++: si incolla quanto effettuato in excel e si aggiungono le righe di intestazione.

Numero di condizioni complessive (portate, livelli idrici, scale di deflusso)			
Numero del segmento cui imporre la condizione (letto sul file .BLN)	Numero di valori che descrivono l'andamento dell' <b>onda di piena</b>		
Tempo [s]	Quota idrica [m s.l.m.]	Q [m <sup>3</sup> /s]	0
Numero del segmento cui imporre la condizione (letto sul file .BLN)	Numero di valori che descrivono l'andamento dei <b>livelli</b>		
Tempo [s]	Quota idrica [m s.l.m.]	Q [m <sup>3</sup> /s]	0
Numero del segmento cui imporre la condizione (letto sul file .BLN)	Numero di valori che descrivono la <b>scala di deflusso</b>		
Quota idrica [m s.l.m.]	Q [m <sup>3</sup> /s]	0	////

*Tabella 3.1 Struttura del file .BCC*

```

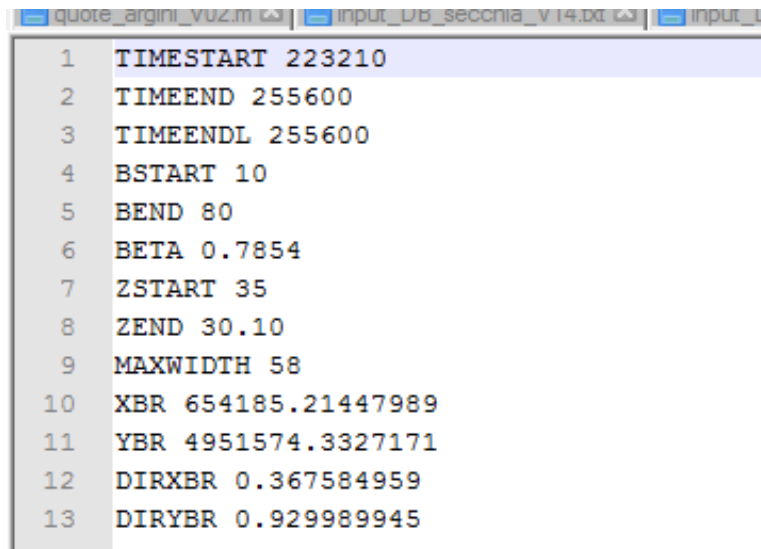
1 3
2 1 11
3 0 25 500 0
4 3600 25 500.00725 0
5 7200 25 500.1112659 0
6 10800 25 500.5402958 0
7 14400 25 501.6379134 0
8 18000 25 503.8356184 0
9 21600 25 507.6289501 0
10 25200 25 513.5567731 0
11 28800 25 522.1834206 0
12 32400 25 534.0834069 0
13 36000 25 549.8284478 0
14 6 2
15 0 12.32 0 0
16 36000 12.32 0 0
17 17 23
18 0 0 0
19 1.12 500.7771379 0
20 1.22 516.0843253 0
21 1.32 532.4837844 0
22 1.42 549.9755152 0
23 1.52 568.5595176 0
24 1.62 588.2357918 0
25 1.72 609.0043376 0
26 1.82 630.8651551 0
27 1.92 653.8182443 0
28 2.02 677.8636051 0
29 2.12 703.0012377 0
30 2.22 729.2311419 0
31 2.32 756.5533178 0
32 2.42 784.9677654 0
33 2.52 814.4744847 0
34 2.62 845.0734756 0
35 2.72 876.7647382 0
36 2.82 909.5482725 0
37 2.92 943.4240785 0
38 3.02 978.3921562 0
39 3.12 1014.452506 0
40 3.22 1051.605127 0

```

*Figura 3.9 Esempio di file .BCC*

## 2.6. Generazione di una breccia in evoluzione

La creazione di una breccia in un tratto arginale, che evolve sia temporalmente che spazialmente, è resa possibile utilizzando il file .BRE, strutturato come l'esempio riportato in Figura 3.11 e realizzato con l'ausilio di Notepad++:



```

1  Timestart 223210
2  Timeend 255600
3  Timeendl 255600
4  Bstart 10
5  Bend 80
6  Beta 0.7854
7  Zstart 35
8  Zend 30.10
9  Maxwidth 58
10 XBR 654185.21447989
11 YBR 4951574.3327171
12 DIRXBR 0.367584959
13 DIRYBR 0.929989945

```

Figura 3.11 Esempio di file .BRE

Devono quindi essere definiti i seguenti parametri:

1. Istante in cui inizia la formazione della breccia (in secondi);
2. Istante in cui termina l'abbassamento (in secondi);
3. Istante in cui termina l'allargamento (in secondi);
4. Larghezza iniziale (in metri);
5. Larghezza finale (in metri);
6. Angolo di inclinazione;
7. Quota dell'argine da cui parte la fessurazione (in metri slm);
8. Quota finale che raggiunge la breccia (ad esempio il piano campagna) (in metri slm);
9. Larghezza massima in direzione ortogonale a quella dell'argine (in metri);
10. Coordinata x del punto da cui si origina la breccia;
11. Coordinata y del punto da cui si origina la breccia;
12. Versore x della breccia;
13. Versore y della breccia.

Per definire i versori si devono individuare due punti lungo l'argine aventi coordinate  $x_1$ ,  $y_1$ ,  $x_2$ ,  $y_2$ .

Si calcolano le differenze:

$$a = x_1 - x_2$$

$$b = y_1 - y_2$$

Si calcola la lunghezza del segmento:

$$l = \text{radq}(a^2 + b^2)$$

Versore x: a/l

Versore y: b/l

### 2.6.1. Chiusura della breccia

È possibile simulare anche la chiusura della breccia stessa (ad esempio per simulare interventi provvisori in fase di evento). Per farlo, occorre aggiungere, in coda al file .BRE, alcuni parametri:

14. Istante di inizio chiusura della breccia (in secondi);
15. Istante di fine chiusura “verticale” (rialzo del fondo) (in secondi);
16. Istante di fine chiusura “laterale” (restringimento) (in secondi);
17. Larghezza della chiusura in direzione ortogonale all’argine (in metri).

Il file sarà dunque simile a quello riportato nella Figura 3.12.

```
TIMESTART 367200
TIMEEND 378000
TIMEENDL 378000
BSTART 10
BEND 100
BETA 0.7854
ZSTART 22.4
ZEND 16.3
MAXWIDTH 35
XBR 675615.8288
YBR 4964774.056
DIRXBR -0.961105965
DIRYBR 0.276179875
CLOSESTART 378010
CLOSEEND 421200
CLOSEENDL 421200
CLOSEWIDTH 10
```

*Figura 3.12 Esempio di file .BRE con parametri per la chiusura della breccia*

## 2.7. File di input (parametri generali della simulazione)

In Figura 3.13 si riporta l’esempio di un file di input al modello, descrivendo in seguito il significato dei principali parametri coinvolti.

```

1 PO_gamma_Casalm ! file name BASE with input files for initial conditions, bathymetry, output and boundary conditions
2 START 0. ! t0, beginning time (hours)
3 END 10. ! tlast, simulation time
4 DTOUTPUT 1. !in hours
5 CR 0.9 ! CR, courant number
6 LIMITER 3 ! limiter, (1 van leer, 2 van albeda, 3 minbee, 4 superbee)
7 YEPS 1.E-5 ! yeps, water depth threshold for wet/dry cell
8 VELEPS 0.20 !threshold to limit fluxes for very shallow cells
9 MUSCL -1 ! R, MUSCL reconstruction -1 UPWIND, 1 CENTERED
10 IBINARY 0 ! IBINARY, input/output (1 for ASCII, 0 for IBINARY)
11 AL 2 ! AL 0 SGM puro, 1 DGM puro, 2 WSDGM
12 SR 1 ! SR, 1 strang splitting formulation for friction source term, 0 for godunov formulation
13 DTSOGLIA 0.001 ! dtsoglia, threshold for the time step (if dt<dtsoglia then WARINING is plotted)
14 EXPON 1.667 ! expon, for discharge redistribution at inflow boundary conditions
15 AW 0. ! A_WSDGM, froude number for the beginning of the DGM trsition
16 BW 2.0 ! B_WSDGM, end of the DGM transition
17 METODO 1 ! metodo, 2 for HLLC fluxes, 1 for slic fluxes
18 NITER 3 ! niter, iteration number for correction procedure at wetting/drying fronts
19 PENDFARFIELD 0.1 ! slope for farfield boundary condition

```

Figura 3.13 Esempio del file di input

- NOME STRINGA: nome della parte che precede l'estensione dei file;
- START xxx: tempo di inizio della simulazione (inizio alle ore xxx);
- END xxx: tempo in cui termina la simulazione (fine alle ore xxx);
- DTOUTPUT xxx: tempo in cui salvare i risultati (risultati salvati e stampati su file ogni xxx ore); esiste l'opzione di dt\_output variabile nel tempo: in questo caso va posto un valore negativo in questo file di input e va fornito un file aggiuntivo con estensione .DTO (si veda oltre);
- CR xxx: numero di Courant utilizzato per la definizione del passo di calcolo;
- YEPS xxx: soglia che distingue le celle bagnate da quelle asciutte (valore del livello< xxx la cella è considerata asciutta). Una valore basso della soglia YEPS comporta errori ridotti ma espone a un rischio maggiore di instabilità numerica;
- VELEPS xxx: soglia per limitare i flussi scambiati tra celle molto piccole, inferiori al valore fissato xxx;
- DTSOGLIA xxx: soglia temporale per il passo di calcolo. Il modello è esplicito ed il passo di calcolo  $dt$  viene calcolato mediante la seguente relazione:

$$dt = CR \cdot \min \frac{dx}{\sqrt{gh} + V}$$

La soglia temporale funge da parametro di allarme; nel caso in cui durante la simulazione in una cella si raggiungessero velocità  $V$  non fisiche, il denominatore tenderebbe a zero,  $dt < xxx$  e il modello interromperebbe istantaneamente il processo di calcolo segnalando un valore di  $dt$  troppo piccolo;

- PENDFARFIELD xxx: si fissa pari a xxx la pendenza del terreno nella cella adiacente (ma esterna al dominio di calcolo) a quella sulla quale è imposta la condizione al contorno di farfield. Mettere il segno +/- in base all'inclinazione.

## 2.8. Multirisoluzione

Quanto finora descritto è valido per effettuare simulazioni utilizzando una griglia di calcolo cartesiana: il dominio di calcolo è discretizzato con una griglia di celle quadrate (o rettangolari) di dimensioni predefinite e costanti ovunque. Per domini molto estesi, questa discretizzazione si traduce in una griglia di calcolo con un numero molto elevato di celle (dell'ordine di  $10^6$ - $10^7$  celle). Per ridurre l'onere computazionale, è possibile ricorrere ad una griglia multi-risoluzione, ossia in cui le dimensioni delle celle non sono costanti ovunque, ma risultano più piccole ove occorra un dettaglio maggiore e più grandi nelle zone di minor interesse.

Nel codice PARFLOOD, per generare una griglia multi-risoluzione occorre comunque fornire in input le mappe .BTM, .MAN, ecc... alla massima risoluzione (con celle alla dimensione minima consentita). Va poi fornito un file di testo con estensione .PTS in cui è contenuta la lista dei punti con risoluzione prefissata. Il file ha quattro colonne: le prime due rappresentano le coordinate x e y dei vari punti (ottenuti manualmente in Surfer con il comando “digitize”), la terza riporta la risoluzione richiesta (da 1 a 4, con 1=risoluzione più fitta  $\Delta x_{min}$ , 2=risoluzione pari a  $2\Delta x_{min}$ , 3=risoluzione pari a  $4\Delta x_{min}$ , 4=risoluzione più larga pari a  $8\Delta x_{min}$ ) e la quarta riporta uno zero (eventualmente si può usare per l'imposizione di “tappi”, con 0=no tappo, 1=tappo). Un esempio è mostrato in Figura 3.14.

Oltre alle restrizioni fornite dal file .PTS, è necessario imporre una risoluzione predefinita anche sul contorno del dominio (ossia lungo il .BLN). Questa operazione è fatta da riga di comando al momento del lancio della simulazione (vedi paragrafo 3.1), tramite la stringa **-bcc-level=n** (dove  $n=1,2,3,4$ ). È possibile inoltre imporre una risoluzione predefinita (tipicamente la massima) in corrispondenza della breccia (**-bre-level=n**).

Infine, è possibile imporre il numero di livelli di risoluzione con la stringa **-levels=n** (dove  $n$  è il numero di livelli di risoluzione, di default=4).

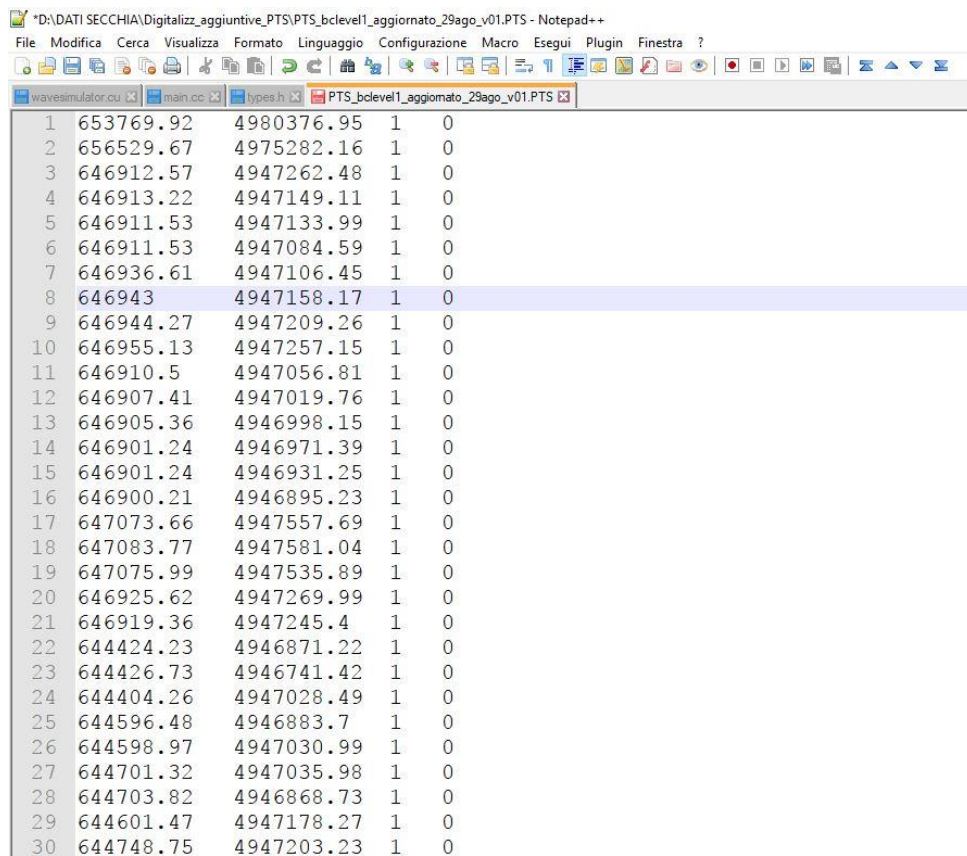
Qualora non si fornisca in input il file .PTS, il modello assume automaticamente ovunque la risoluzione più grande (4).

Molte informazioni sulla generazione della griglia multi-risoluzione sono scritte nel file di output, inclusi eventuali errori e conflitti che richiedono una modifica del file .PTS.

È sempre consigliabile verificare la griglia multi-risoluzione generata dal modello prima di effettuare le simulazioni. Per farlo, è possibile utilizzare il file con estensione .PMAP fornito in output. Si tratta di un file di testo con l'elenco di tutte le celle del dominio e la relativa risoluzione (tre colonne: x, y, risoluzione 0-3, che corrisponde a 1-4 per il numero di livelli di default). Per



visualizzarlo è possibile importarlo in Surfer come Post Map o come Classed Post Map (eventualmente riducendo la frequenza con cui sono visualizzati i punti).



1	653769.92	4980376.95	1	0
2	656529.67	4975282.16	1	0
3	646912.57	4947262.48	1	0
4	646913.22	4947149.11	1	0
5	646911.53	4947133.99	1	0
6	646911.53	4947084.59	1	0
7	646936.61	4947106.45	1	0
8	646943	4947158.17	1	0
9	646944.27	4947209.26	1	0
10	646955.13	4947257.15	1	0
11	646910.5	4947056.81	1	0
12	646907.41	4947019.76	1	0
13	646905.36	4946998.15	1	0
14	646901.24	4946971.39	1	0
15	646901.24	4946931.25	1	0
16	646900.21	4946895.23	1	0
17	647073.66	4947557.69	1	0
18	647083.77	4947581.04	1	0
19	647075.99	4947535.89	1	0
20	646925.62	4947269.99	1	0
21	646919.36	4947245.4	1	0
22	644424.23	4946871.22	1	0
23	644426.73	4946741.42	1	0
24	644404.26	4947028.49	1	0
25	644596.48	4946883.7	1	0
26	644598.97	4947030.99	1	0
27	644701.32	4947035.98	1	0
28	644703.82	4946868.73	1	0
29	644601.47	4947178.27	1	0
30	644748.75	4947203.23	1	0

Figura 3.14 Esempio del file .PTS

## 2.9. Scrittura dei risultati con DTOUTPUT variabile

È possibile scrivere i file di output ad intervalli variabili nel tempo, anziché ad intervalli fissi. In questo caso, nel file di input deve essere scritto un valore negativo alla voce “DTOUTPUT” (ad esempio -1). Inoltre deve essere fornito un file aggiuntivo .DTO che contenga le informazioni sugli intervalli di scrittura desiderati.

Il file .DTO deve essere formattato come segue (in questo caso tutti i tempi sono espressi in secondi):

N\_intervalli

T\_iniz1 Dt\_output1

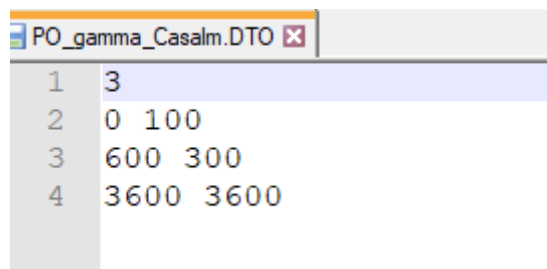
T\_iniz2 Dt\_output2

T\_iniz3 Dt\_output3

...

Nel file il primo valore rappresenta il numero dei diversi intervalli temporali che si prevede di utilizzare. Successivamente sono elencati i valori degli istanti temporali in cui iniziare a scrivere con un determinato DTOUTPUT (nella prima colonna) e il DTOUTPUT stesso (seconda colonna). Si veda ad esempio la Figura 3.15: in questa simulazione verranno scritti i risultati inizialmente (da  $t=0$ ) ogni 100 s, poi a partire da 600 s (10 min) si scriverà ogni 300 s (5 min); infine, da 3600 s (1 h) i risultati saranno restituiti ogni ora (3600 s) fino alla fine della simulazione (si noti che non è necessario specificare l'istante finale!).

I frames stampati saranno comunque numerati in ordine crescente, per l'istante temporale corrispondente si dovrà far riferimento al file in uscita chiamato [Nome caso]\_output.txt (si veda il paragrafo 4.1).



1	3
2	0 100
3	600 300
4	3600 3600

*Figura 3.15 Esempio del file .DTO*

## Capitolo 3. FASE DI CALCOLO

Il modello necessita di coerenza tra tutti i nomi dei file di input, ossia la stringa che precede l'estensione dei file deve essere univoca all'interno della simulazione. I file inoltre devono essere collocati all'interno di un'unica cartella. La cartella *prova* ad esempio conterrà: *prova.BTM*, *prova.MAN*, *prova.INH*, *prova.BLN*, *prova.BCC*, *prova.BRE*, ...

Le griglie di calcolo generate dovranno inoltre riportare le stesse dimensioni ed essere salvate nel formato Surfer 6 Text Grid.

### 3.1. Alcune istruzioni utili per lanciare le simulazioni

Il software Cygwin consente di lanciare il modello di calcolo sfruttando il linguaggio LINUX, mentre WinSCP permette di gestire le cartelle di input e output con un'interfaccia grafica.

Prima di procedere però occorre disporre della cartella *prova* contenente tutti i file di ingresso al modello e di avere accesso alla scheda video di un idoneo calcolatore mediante il collegamento remoto ssh.

#### WinSCP

Alla prima apertura di WinSCP occorre specificare:

- Protocollo file: SCP;
- Nome server
- Nome utente: nome dell'account fornito;
- Password: password fornita.

Al termine della compilazione dei campi si consiglia di salvare la posizione, al fine di evitare di ricompilare tutte le voci al successivo accesso.

All'interno della cartella con il proprio account sarà creata una cartella *swe*, a sua volta ripartita in due rami:

*Nome account*

- Swe
  - Bin
  - Res

La cartella *prova* deve essere copiata all'interno di *res*.

## Cygwin

Tramite Cygwin ci si connette alla porta della scheda video remota mediante il comando `ssh`. Ad esempio, volendo connettersi al calcolatore “4K40” occorre digitare:

**`ssh nomeutente@160.78.64.48`**

verrà poi richiesta la **password**.

Per navigare all'interno delle cartelle in LINUX si usa il comando **`cd`** (change directory). Per entrare in una sottocartella si digita:

**`cd nome_cartella`**

mentre per risalire alla cartella di livello superiore si digita:

**`cd ..`**

Entrati nella propria cartella, ossia *nomeutente*, si deve andare all'interno della cartella *res* dove è contenuta la cartella *prova* ed entrare in quest'ultima:

**`cd swe`**

**`cd res`**

**`cd prova`**

oppure semplicemente **`cd swe/res/prova`** (si possono combinare i percorsi).

La simulazione *prova* viene lanciata con il comando:

**`nohup ../../bin/swegpu input_prova.txt -order=1 -gpu=0 (+eventuali altre condizioni) > output_prova.txt &`**

(la riga di comando è unica, qui è spezzata in due solo per motivi di formattazione)

La serie di comandi ha il seguente significato:

1. *nohup* permette che la simulazione prosegua nonostante la chiusura di Cygwin.
2. Lancio dell'eseguibile: dalla cartella *prova* si risale alla posizione dell'eseguibile, ossia si sale di un livello (si arriva in *res*), poi di un altro (si arriva in *swe*), poi si entra in *bin*, infine si lancia l'eseguibile *swegpu*.
3. *input\_prova.txt* è il file di input principale della simulazione.
4. Si specificano alcune condizioni per la simulazione:
  - a) `-order=1` (alternativa 2) → specifica l'ordine di accuratezza della simulazione;
  - b) `-gpu=0` → specifica quale GPU verrà utilizzata per la simulazione;
  - c) `-debug=1` (di default è 0) → si richiede che i risultati siano stampati per ogni passo di calcolo e che vengano fornite informazioni aggiuntive;

- d) -multi=1 (oppure 0, oppure hi) → specifica se la simulazione è effettuata in multirisoluzione (1), con griglia cartesiana (0), o con “finta” multirisoluzione (hi);
  - e) -maxmaps=1 → se viene richiesta la stampa finale delle mappe MAXVEL, MAXHTOT, TIMEARR;
  - f) -checkvol=1 → per attivare il controllo sul bilancio di volume;
  - g) -levels=n → numero livelli di multirisoluzione (default=4);
  - h) -bcc-level=1 (oppure 2,3,4) → livello risoluzione sulle condizioni al contorno;
  - i) -bre-level=1 (oppure 2,3,4) → livello risoluzione sulla breccia.
5. Nella cartella *prova*, da cui lanciamo il comando, verrà scritto il file di output.txt.
  6. & permette di riavere la schermata libera mantenendo la simulazione in background. Qualora si dimenticasse di aggiungere “&” alla riga di comando, per tornare alla schermata libera senza interrompere la simulazione è possibile digitare: **ctrl+z** in modo tale da liberare la schermata e poi scrivere **bg** affinché la simulazione possa continuare in background.

### Come usare l’help

Per ricordare le varie opzioni da aggiungere alla riga di comando, è possibile usare la seguente stringa di comando:

**../bin/swegpu -help**

che stampa a video tutte le opzioni di simulazione, nonché le opzioni di decode (vedi paragrafo 4.2).

### Come terminare una simulazione

Nel caso in cui si volesse terminare una simulazione corrente basterebbe digitare:

**top**

così da vedere i lavori in corso e poi

**k** (*kill*) seguito dal numero dell’operazione da fermare ed eliminare, e poi invio.

Per uscire da top basta premere **ctrl+c**.

### Come lanciare una simulazione con resume

Si ipotizzi di aver terminato una simulazione *prova1* di 10 ore, in cui i risultati sono stati salvati con passo orario. All’interno della cartella saranno quindi presenti tutti i file di output per gli istanti: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, last.

Si ipotizzi ora di simulare (*prova2*) un dam-break, oppure un generico collasso, oppure un cambio di scabrezza o di condizioni al contorno che avvenga alla decima ora della precedente simulazione (*prova1*). L’obiettivo è quindi di prolungare la simulazione precedente variando alcune

condizioni, in altri termini partire da una condizione iniziale coincidente con i risultati della decima ora. Sarà sufficiente aggiungere, rinominandoli correttamente, tutti i file di output relativi all'istante 9 nella cartella *prova2* e lanciare *prova2* in modalità resume, ossia scegliendo di ripartire dall'istante 9.

```
nohup ../../bin/swegpu input_prova.txt -order=1 -gpu=0 (+eventuali altre condizioni)  
-resume=0009 > output_prova2.txt &
```

La modalità resume non è vincolata all'ultimo istante temporale, restando nel caso precedente, si potrebbe infatti lanciare in resume partendo da qualsiasi istante compreso tra 0 e 9.

Una precisazione deve essere fatta in merito al tempo di calcolo. Ipotizzando di voler simulare un periodo di 50 ore, di cui 10 ore simulate in *prova1* e le restanti 40 ore in *prova2*, il file di input in *prova2* dovrebbe contenere l'indicazione di 50 ore e non di 40 ore (il tempo cui fare riferimento è quello complessivo).

Per insistere sul concetto di tempo si riportano ora due esempi atti a giustificare la decisione di eseguire una simulazione in resume:

1. Sostituzione del file .BTM. La simulazione *prova2*, rispetto a *prova1*, contiene un diverso file .BTM (ad esempio crollo/innalzamento di un tratto arginale o di un manufatto). Si assuma una durata della simulazione pari a 50 ore e si ipotizzi la necessità di sostituire il file alla decima ora. Il file di input di *prova1* presenterà una durata di 10 ore, mentre quello di *prova2* di 50 ore. Il file .BCC sarà invece il medesimo per entrambe le simulazioni; nonostante il .BCC contenga le condizioni per 50 ore, la simulazione *prova1*, coerentemente con la durata scritta nel file di input, terminerà dopo 10 ore.
2. Sostituzione del file .BCC. Si ipotizzi per un qualsiasi motivo di voler cambiare le condizioni al contorno a partire dalla decima ora. Il file .BCC di *prova2*, ovviamente diverso da quello di *prova1*, pur descrivendo soltanto 40 ore, dovrà iniziare non dall'istante 0 s ma da 36000 s (10 ore).

A partire da questo istante nella cartella *prova* in res compariranno i primi risultati, i quali potranno poi essere copiati tramite WinSCP sul proprio computer. Si presti attenzione a copiare i risultati soltanto dopo che il codice abbia effettivamente terminato di scriverli (semplicemente si copino ad esempio i risultati dell'istante 0 quando inizieranno a comparire quelli di 1).

#### **Caso particolare: -resume=-1**

Qualora si volesse rilanciare una simulazione in multi-risoluzione nella quale sono stati modificati alcuni parametri dei file di input (ma NON le mappe .BTM, .MAN, né le condizioni iniziali, né il

.PTS né il .BLN), è possibile usare l'opzione `--resume=-1`, che permette di saltare le operazioni di pre-processing e iniziare direttamente i calcoli, previa lettura degli 8 file prova..JUMP-xxx (dove xxx sono: PREPROCESSINFO, OFSBLOCKS, ecc..) e prova-INITIAL.WSE, .VVX ecc.. (tutti derivanti dalla simulazione già effettuata).

## Capitolo 4. RISULTATI

### 4.1. File di output

Il programma di calcolo restituisce ad intervalli di tempo definiti nel file di input, i seguenti file:

- .WSE: quote idriche (water surface elevation);
- .DEP: profondità idriche (depth);
- .VVX: componente lungo l'asse x delle velocità;
- .VVY: componente lungo l'asse y delle velocità;
- Il file [Nome caso]\_output.txt: contiene tante righe quanti sono gli istanti temporali salvati e riassume i tempi di calcolo e di salvataggio; è importante specialmente nel caso di dtoutput variabile;
- Il file Output\_[nome caso].txt: descrive in maniera esaustiva l'intero procedimento seguito dal modello di calcolo. Dalla lettura di questo documento si comprendono:
  - Le dimensioni dei file di input caricati;
  - Le coordinate delle sezioni;
  - La rasterizzazione dei segmenti che costituiscono le sezioni;
  - Le condizioni al contorno applicate;
  - Per ogni istante di calcolo e non solo per quelli di salvataggio si individuano le celle (i,j) che vincolano il passo di calcolo dt, il valore di dt, le velocità lungo x ed y;
  - Per ciascuna cella derivante dalla rasterizzazione del segmento delle sezioni si specificano le componenti di velocità memorizzate (Uh, Vh);
  - Gli errori nella simulazione, ad esempio file aventi griglie di diverse dimensioni (segnalato con la scritta "dimensioni diverse.."), oppure il raggiungimento di velocità non fisiche (segnalato con la scritta "dt troppo piccolo");
- Nei file denominati debug.N, debug.S, debug.E, debug.W (solo per `--debug=1`) sono riportati i tipi di condizioni al contorno assegnati a ciascuna cella, rispettivamente lungo le direzioni Nord, Sud, Est, Ovest. Importando tali file in Surfer, si visualizza la presenza

di eventuali errori connessi proprio alle condizioni al contorno, ad esempio l'esistenza di un muro (condizione 1) in un punto in cui si era posto un valore di livello (condizione 3).

La numerazione degli istanti temporali inizia da 0 e termina con LAST, file contenente gli ultimi risultati prima del termine della simulazione o di una sua improvvisa interruzione. Si ipotizzi, ad esempio, di voler simulare per un tempo complessivo di tre ore, salvando ogni ora, i file di output saranno quattro: 0 (prima ora), 1 (seconda ora), 2 (terza ora), LAST (terza ora). La numerazione sarà comunque crescente da 0 a LAST anche nel caso di dtoutput variabile, ma in tal caso sarà necessario fare attenzione alla corrispondenza tra numero del frame e istante temporale di stampa (si può far riferimento al file [Nome caso]\_output.txt).

Tra i file LAST, oltre ai già citati .WSE, .DEP, .VVX, .VY, compare anche .MAXWSE: al suo interno è memorizzato l'involuppo dei massimi valori di livello raggiunti dalle singole celle durante l'intera simulazione. Se si è richiesta in input la stampa delle mappe del massimi (-maxmaps=1), compaiono inoltre i file LAST .MAXVEL, .MAXHTOT, .TIMEARR.

Se si simula una breccia (si dà in input un file .BRE), sono inoltre stampate le mappe .BTM a partire dal frame immediatamente successivo all'istante di apertura della breccia, fino al LAST.

Nel rispetto della coerenza tra tutti i titoli dei file di input e output, l'estensione dei file sopra descritti è preceduta dal nome del caso test e dal numero dell'istante temporale, ad esempio prova-output-0010.WSE, prova-output-0010.VVX, ...

I file .WSE, .DEP, .VVX, .VY riportano per ciascun istante temporale l'andamento delle quote idriche, delle profondità, nonché i vettori velocità per definire la direzione seguita dalla corrente.

Tutti i file di output sono scritti in linguaggio binario e non sono immediatamente visualizzabili. Per poterli visualizzare con Surfer, è necessario "decodificarli" (vedi paragrafo successivo).

## 4.2. Decode

Sono tre le tipologie fondamentali di dati che si possono ottenere decodificando i file di output:

1. Mappe di livelli, quote idriche, velocità, ecc...;
2. Portate nelle sezioni;
3. Livello nei punti.

In tutti i casi il comando da lanciare da riga di comando è del tipo seguente:

**../bin/swegpu -decode nome\_file\_da\_decodificare nome\_file\_decodificato opzioni\_decode**

Le opzioni di decode sono riassunte anche nell'help (vedi paragrafo 3.1).



### 4.2.1. Decode mappe

È possibile convertire una mappa dei risultati della simulazione da file binario a grd di Surfer per poterla visualizzare. I formati disponibili in output sono Surfer 6 Text Grid e Surfer 6 Binary Grid (opzione **-binary=b**, con  $b=0$  per ascii e  $b=1$  per binario: quest'ultima è l'opzione di default ed è consigliata).

L'operazione può riguardare l'intera mappa oppure solo una sua porzione. Nel primo caso, l'opzione da utilizzare è **-all**, nel secondo caso **-range xmin xmax ymin ymax** (dove *xmin*, *xmax*, *ymin*, *ymax* sono gli estremi della porzione di dominio che si vuole decodificare).

Se si dispone di una lunga serie di frames, per decodificarli tutti non è necessario ripetere l'operazione molte volte, ma si può aggiungere l'opzione **-frames frame\_start frame\_end** (dove *frame\_start* *frame\_end* sono gli estremi dell'intervallo di frame da decodificare, es. da 0 a 9). Come nome del file da decodificare si specifica quello del frame 0000, come nome del file decodificato si omette l'estensione, che verrà attribuita in automatico con la numerazione progressiva.

Infine, si può (e si deve) specificare la risoluzione a cui decodificare la mappa, tramite l'opzione **-res=n** (con  $n=1,2,3,4$ ).

Facciamo un esempio. Vogliamo decodificare le mappe intere della profondità idrica dal frame 0 al frame 9, alla massima risoluzione e in formato ascii, per una simulazione chiamata "prova". Dovremo scrivere:

```
../bin/swegpu -decode prova-output-0000.DEP prova-outputascii- -all -frames 0 9  
-binary=0 -res=1
```

che genererà le mappe: prova-outputascii-0000.DEP, prova-outputascii-0001.DEP, ... prova-outputascii-0009.DEP. E' utile chiamare i file decodificati con l'appendice outputascii perché sono scritti in formato ASCII e possono quindi essere immediatamente riconosciuti da quelli binari non decodificati.

Se invece ci interessasse solo una porzione della mappa (es. tra  $x=100$  e  $x=200$  e tra  $y=100$  e  $y=200$ ) scriveremmo:

```
../bin/swegpu -decode prova-output-0000.DEP prova-outputascii- -range 100 200 100 200  
-frames 0 9 -binary=0 -res=1
```

Per le mappe LAST si procede in modo simile, ma non occorre mai specificare i frames. Ad esempio per la mappa LAST-MAXWSE scriveremmo:

```
../bin/swegpu -decode prova-output-LAST.MAXWSE  
prova-outputbin-LAST.MAXWSE -range 100 200 100 200 -res=1
```

Non avendo specificato l'opzione -binary, otterremo di default la mappa in formato binario.

Alcune opzioni particolari:

- a) **-createmaxwse** calcola la massima WSE tra i frames specificati e si lancia come segue:

```
../bin/swegpu -decode prova-output-0000.WSE prova-outputascii-MAXWSE  
-createmaxwse -all -frames frame_start frame_end -res=n
```

È utile per calcolare il MAXWSE quando si sono effettuati dei resume della simulazione e non è dunque disponibile l'output LAST dell'intera simulazione.

- b) **-createmaxvelmagn** calcola il modulo massimo della velocità tra i frames specificati e si lancia come segue:

```
../bin/swegpu -decode prova-output-0000.VVX prova-outputascii-MAXVVM  
-createmaxvelmagn -all -frames frame_start frame_end -res=n
```

- c) **-createvelmagn** calcola il modulo delle velocità per ogni frame specificato e si lancia come segue:

```
../bin/swegpu -decode prova-output-0000.VVX prova-outputascii- -createvelmagn  
-all -frames frame_start frame_end -res=n
```

e fornisce in output dei file chiamati prova-output-0000.VVM, ecc..

- d) **-createfroude** calcola il numero di Froude per ogni frame specificato e si lancia come segue:

```
../bin/swegpu -decode prova-output-0000.VVX prova-outputascii- -createfroude  
-all -frames frame_start frame_end -res=n
```

e fornisce in output dei file chiamati prova-output-0000.FRO, ecc...

- e) **-createmaxtothead** calcola il carico totale massimo tra i frame specificati e si lancia come segue:

```
../bin/swegpu -decode prova-output-0000.VVX prova-out-MAXTOTHEAD  
-createmaxtothead -all -frames frame_start frame_end -res=n
```

- f) **-createmaxtotdepth** calcola la profondità totale massima tra i frame specificati e si lancia come segue:

```
../bin/swegpu -decode prova-output-0000.VVX prova-out-MAXHTOT  
-createmaxtotdepth -all -frames frame_start frame_end -res=n
```

- g) **-createmaxintensity** calcola la intensità (= altezza idrica\*velocità contemporanee) massima tra i frame specificati e si lancia come segue:

```
../bin/swegpu -decode prova-output-0000.VVX prova-out-MAXINTENS  
-createmaxintensity -all -frames frame_start frame_end -res=n
```

- h) **-createtimemaxwse** calcola il tempo di arrivo della massima quota idrica in ogni cella tra i frame specificati e usando il *dt\_output* (in ore) fornito e si lancia come segue:

```
../bin/swegpu -decode prova-output-0000.WSE prova-out-TIME_MAXWSE  
-createtimemaxwse dt_output -all -frames frame_start frame_end -res=n
```

- i) **-createtimearr** calcola il tempo di arrivo dell'allagamento in ogni cella tra i frame specificati e usando il *dt\_output* (in ore) fornito e si lancia come segue:

```
../bin/swegpu -decode prova-output-0000.WSE prova-out-TIMEARR  
-createtimearr dt_output -all -frames frame_start frame_end -res=n
```

- j) **-createtimdepth** calcola il tempo di arrivo di una prefissata altezza idrica *h\_fixed* (in m) in ogni cella tra i frame specificati e usando il *dt\_output* (in ore) fornito e si lancia come segue:

```
../bin/swegpu -decode prova-output-0000.WSE prova-out-TIMEDEPTH1  
-createtimdepth h_fixed dt_output -all -frames frame_start frame_end -res=n
```

Tutti gli esempi precedenti possono ovviamente essere fatti anche specificando un range.

#### 4.2.2. Decode portata in una sezione e livello in un punto

##### Portata in una sezione

Per estrarre la portata che transita in una sezione predefinita (che va dal punto *x1,y1* al punto *x2,y2*), si usa l'opzione **-sez *x1 y1 x2 y2***. I valori *x,y* degli estremi della sezione si ottengono da Surfer usando lo strumento "digitize". Il comando è il seguente:

```
../bin/swegpu -decode prova-output-0000.WSE prova-nome_sezione -sez x1 y1 x2 y2  
-frames frame_start frame_end -res=1
```

Si ottengono tre file: *prova-nome\_sezione.SEZX*, *prova-nome\_sezione.SEZY*, *prova-nome\_sezione.SEZ*. I primi due forniscono le componenti della portata lungo *x* e lungo *y* rispettivamente, mentre il file *.SEZ* riassume il risultato su tre colonne: *Qx*, *Qy*, *Qtot*. Si presti attenzione al fatto che *Qtot* è ottenuta come somma/differenza di *Qx* e *Qy* (a seconda dell'orientamento della sezione) e non con il teorema di Pitagora.

Non è riportata la colonna dei tempi, che è però facilmente ottenibile in Excel se è noto l'intervallo di frames e il *dt\_output*.

##### Livello in un punto

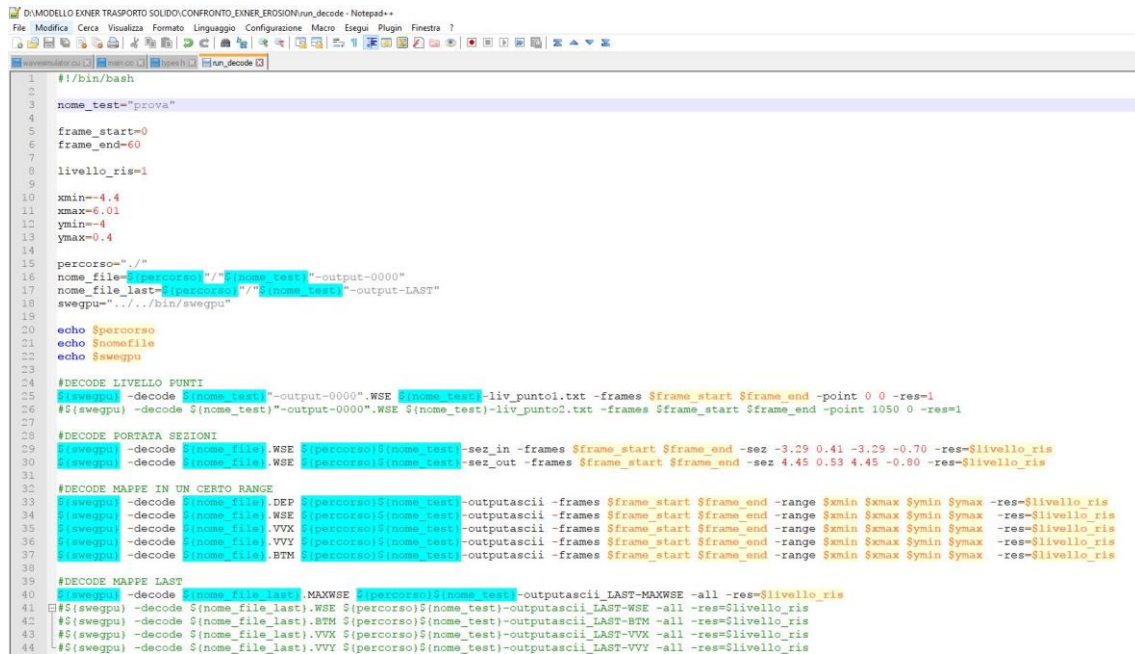
In maniera analoga, si può estrarre la quota idrica in un punto predefinito *xp,yp* (digitalizzato da Surfer), usando l'opzione **-point *xp yp***. Si usa il comando seguente:

```
../bin/swegpu -decode prova-output-0000.WSE prova-livelloP.txt -point xp yp  
-frames frame_start frame_end -res=1
```

Anche in questo caso non è riportata la colonna dei tempi, ma solo la colonna del livello.

### 4.2.3. Utilizzo di uno script per il decode

Conviene predisporre uno script riassuntivo per il decode che contenga già tutte le opzioni possibili. Un esempio è riportato nella Figura 4.1.



```
1 #!/bin/bash
2
3 nome_test="prova"
4
5 frame_start=0
6 frame_end=60
7
8 livello_ris=1
9
10 xmin=-4.4
11 xmax=6.01
12 ymin=-4
13 ymax=0.4
14
15 percorsore="/"
16 nome_file=${percorsore}/${nome_test}-output-0000"
17 nome_file_last=${percorsore}/${nome_test}-output-LAST"
18 svegpu=".../bin/svegpu"
19
20 echo $percorsore
21 echo $nome_file
22 echo $svegpu
23
24 #DECODE LIVELLO PUNTI
25 $svegpu -decode $nome_test -output-0000".WSE $nome_test -liv_punto1.txt -frames $frame_start $frame_end -point 0 0 -res=1
26 $svegpu -decode $nome_test -output-0000".WSE $nome_test -liv_punto2.txt -frames $frame_start $frame_end -point 1050 0 -res=1
27
28 #DECODE PORTATA SEZIONI
29 $svegpu -decode $nome_file.WSE $percorsore$nome_test -sez_in -frames $frame_start $frame_end -sez -3.29 0.41 -3.29 -0.70 -res=$livello_ris
30 $svegpu -decode $nome_file.WSE $percorsore$nome_test -sez_out -frames $frame_start $frame_end -sez 4.45 0.53 4.45 -0.80 -res=$livello_ris
31
32 #DECODE MAPPE IN UN CERTO RANGE
33 $svegpu -decode $nome_file.DEP $percorsore$nome_test -outputasci -frames $frame_start $frame_end -range $xmin $xmax $ymin $ymax -res=$livello_ris
34 $svegpu -decode $nome_file.WSE $percorsore$nome_test -outputasci -frames $frame_start $frame_end -range $xmin $xmax $ymin $ymax -res=$livello_ris
35 $svegpu -decode $nome_file.VVX $percorsore$nome_test -outputasci -frames $frame_start $frame_end -range $xmin $xmax $ymin $ymax -res=$livello_ris
36 $svegpu -decode $nome_file.VVY $percorsore$nome_test -outputasci -frames $frame_start $frame_end -range $xmin $xmax $ymin $ymax -res=$livello_ris
37 $svegpu -decode $nome_file.BTH $percorsore$nome_test -outputasci -frames $frame_start $frame_end -range $xmin $xmax $ymin $ymax -res=$livello_ris
38
39 #DECODE MAPPE LAST
40 $svegpu -decode $nome_file_last.MAXWSE $percorsore$nome_test -outputasci LAST-MAXWSE -all -res=$livello_ris
41 $svegpu -decode $nome_file_last.WSE $percorsore$nome_test -outputasci LAST-WSE -all -res=$livello_ris
42 $svegpu -decode $nome_file_last.BTH $percorsore$nome_test -outputasci LAST-BTH -all -res=$livello_ris
43 $svegpu -decode $nome_file_last.VVX $percorsore$nome_test -outputasci LAST-VVX -all -res=$livello_ris
44 $svegpu -decode $nome_file_last.VVY $percorsore$nome_test -outputasci LAST-VVY -all -res=$livello_ris
```

Figura 4.1 Esempio di script per il decode

Si possono predisporre dei valori per il nome del caso test, il livello di risoluzione, i frame iniziali e finali, ecc.. che poi sono richiamati direttamente nelle varie righe di comando.

Predisposto lo script, è necessario renderlo eseguibile. Per prima cosa, va copiato nella cartella che contiene tutti i file di input e output della simulazione. Poi occorre digitare:

**chmod u+x nome\_script**

e fare invio.

A questo punto, per lanciare lo script, basta scrivere il comando:

**./nome\_script**

Si noti che in Figura 4.1 alcune righe sono scritte in colore verde, perché iniziano con “#”. Tale carattere serve per “commentare” una riga, ossia renderla non eseguibile. In questo modo, attivando o disattivando le righe con le varie operazioni, è possibile selezionare quali operazioni compiere di volta in volta.

Uno script simile può essere creato anche per il lancio della simulazione.

### 4.3. Zippare i risultati

Per limitare lo spazio occupato dai risultati di una simulazione, qualora si desideri conservarli, si consiglia di zipparli. Questa operazione può essere effettuata sul proprio PC usando WinZip, WinRar o software analoghi, oppure direttamente sul calcolatore remoto con il comando “zip”.

Per zippare l'intera cartella “prova”, basta spostarsi nella cartella *res* e lanciare il comando:

**zip -r nome\_cartella\_prova.zip prova**

Per un gruppo di file (ad esempio tutti quelli di output, il cui nome inizia con “prova-output-”), ci si deve spostare nella cartella “prova” e digitare:

**zip nome\_file\_zippato.zip prova-output-\***

In alternativa va specificato l'elenco dei file. Ad esempio, per i file di input:

**zip file\_input\_zippati.zip prova.BTM prova.MAN prova.INH altri\_file input\_prova.txt**

Una volta zippati, i file o le cartelle originali si possono cancellare. In alternativa, aggiungendo l'opzione -m tali file/cartelle si cancellano in automatico.

Infine, per estrarre i file all'interno di un archivio zippato, si usa il comando “unzip”:

**unzip nome\_file\_zippato.zip**

In alternativa, si può usare il comando “tar”. Per zippare si digiti:

**tar -czvf nome\_file\_zippato.tar.gz prova**

e per estrarre i file dall'archivio si digiti:

**tar -xzvf nome\_file\_zippato.tar.gz**

#### NOTA BENE

Fare attenzione allo spazio occupato sul disco! Le simulazioni possono occupare anche molti GB di memoria. Al termine di ogni simulazione, si consiglia dunque di:

- effettuare tutte le operazioni di decode necessarie, scaricando sul proprio PC solo i file così ottenuti;
- zippare la cartella della simulazione direttamente sulla macchina remota e poi cancellarla, mantenendo solo l'archivio compresso;
- eventualmente scaricare sul proprio PC l'archivio compresso come backup per le simulazioni considerate “definitive”;
- i file di input devono essere archiviati sul proprio PC.

## Appendice A. UTILIZZO DEL CLUSTER UNIPR

In alternativa alla “4K40”, gli utenti abilitati possono effettuare le simulazioni sul cluster di ateneo. Per connettersi, digitare in Cygwin:

**ssh nome.cognome@login.hpc.unipr.it**

Verrà poi richiesta la **password**, che coincide con la password della mail istituzionale/studenti.

Al primo login, devono essere eseguiti 3 comandi preliminari (si veda al link seguente: [https://www.hpc.unipr.it/dokuwiki/doku.php?id=calcoloscientifico:userguide#password-less\\_access\\_among\\_nodes](https://www.hpc.unipr.it/dokuwiki/doku.php?id=calcoloscientifico:userguide#password-less_access_among_nodes)).

La guida completa per l'utilizzo del cluster è disponibile online (<https://www.hpc.unipr.it/dokuwiki/doku.php?id=calcoloscientifico:userguide>), nel seguito si richiamano solamente alcune informazioni utili per usare Parflood.

Sul cluster di ateneo, sono disponibili due cartelle di lavoro, **home** e **scratch**.

La *home* ha una capienza massima di 50 GB ed è dotata di backup; *scratch*, invece, non ha (al momento) limitazioni di spazio, ma non è disponibile il backup. All'interno della cartella *home*, si consiglia di creare una sottocartella *swe*, che a sua volta contenga *bin* (con l'eseguibile del codice) e *res* (per i file di lavoro). La cartella *scratch* serve invece per contenere i file di output delle simulazioni.

Per muoversi tra le cartelle, si utilizza sempre il comando *cd*. Le scorciatoie per raggiungere *home* e *scratch* da qualsiasi percorso sono:

**cd \$HOME** oppure **cd \$SCRATCH**

Poiché il cluster è condiviso da numerosi utenti, esso è dotato di un “gestore di code” (Slurm) che controlla in automatico l'esecuzione dei *job* in base a vari criteri per stabilirne la priorità, al fine di garantire un uso razionale ed equo delle risorse di calcolo.

Un *job* lanciato da un utente può quindi:

- cominciare immediatamente (se sono disponibili GPU/CPU non utilizzate da altri utenti);
- essere provvisoriamente messa “in coda”, in attesa che termini almeno uno dei *job* che tengono impegnate le risorse di calcolo.

In realtà esistono diverse code (*bdw*, *knl*, *gpu*, *vrt*), ma quella di maggiore interesse per le simulazioni con Parflood è la coda **gpu**, che ha alcune restrizioni:

- la scheda video non può essere occupata dallo stesso *job* per più di 24 ore (al termine delle quali il *job* viene interrotto);
- un singolo utente non può occupare contemporaneamente più di 6 schede.

Per lanciare un *job*, va utilizzato il comando:

**sbatch nome\_script.sh**

dove *nome\_script.sh* è il nome dello script che contiene le istruzioni sulle risorse di calcolo richieste e i comandi per il lancio della simulazione/decode.

Infine, per interrompere un *job* in corso, si digita:

**scancel job\_id**

dove *job\_id* è il numero identificativo del *job* da interrompere.

Per visualizzare l'elenco dei *job* in coda, con i relativi codici identificativi (*job\_id*), si digita:

**squeue**

oppure per i *job* sulla coda *gpu*:

**squeue -p gpu**

oppure per i *job* di un singolo utente:

**squeue -u nome.cognome**

A titolo di esempio, si riporta in Figura A.1 una schermata ottenuta dal comando *squeue* per la coda *gpu*. L'elenco riporta (da sinistra a destra) per ciascun *job*: il codice identificativo del *job* stesso (JOBID), la coda su cui si trova (PARTITION), il nome del *job* (NAME), l'utente che lo ha lanciato (USER), lo stato (ST: R=in corso; PD=in coda), da quanto tempo il *job* è in corso (TIME), il numero di nodi di calcolo usati (NODES), il nome del nodo usato (NODELIST) oppure la motivazione per cui il *job* è in coda (REASON).

```
[susanna.dazzi@ui01 ~]$ squeue -p gpu
  JOBID PARTITION   NAME       USER ST       TIME  NODES NODELIST(REASON)
  934374      gpu  deep_has federico PD        0:00        1 (Resources)
  934434      gpu  lancia_s  leonardo PD        0:00        1 (Priority)
  933934      gpu  dabna1-b rama.dha  R    4:51:20        1 wn41
  933933      gpu  dabna1-b rama.dha  R   10:06:09        1 wn42
  934384      gpu  lancia_s  leonardo R    2:14:59        1 wn41
  934381      gpu  lancia_s  leonardo R    2:23:24        1 wn41
  934388      gpu  KLW_SA1_ laura.sc R    2:04:28        1 wn41
  934371      gpu    npt_1  laura.sc R    3:51:00        1 wn41
```

Figura A.1 Esempio di schermata da *squeue* per la coda *gpu*.

Un altro comando utile per controllare lo stato di utilizzo delle schede video è **gtop**.

Come detto, per lanciare una simulazione, è necessario predisporre uno script apposito, di cui sono riportati due esempi nelle Figure A.2 e A.3. Ovviamente questo può essere personalizzato in base alle esigenze.

Nella prima parte sono riportate le richieste per la coda (precedute da #SBATCH), in particolare: il tempo di calcolo *--time* (nell'esempio 11 ore e 59 minuti), la quantità di memoria *--mem*

(nell'esempio 32 GB), la coda `--partition` (gpu), il numero di schede video richieste `--gres` (nell'esempio 1), il nome del job `--job-name`, l'eventuale richiesta di file di output e di errore. Si noti che il tempo di calcolo richiesto per la coda gpu non può superare le 24 ore.

Nello script, sono poi definiti alcuni parametri, in particolare i nomi delle cartelle di lavoro e dei file di input della simulazione, nonché il nome dell'eseguibile del codice. Nell'ultima riga, si trova il comando per il lancio vero e proprio della simulazione (in cui andranno specificate le varie opzioni, si veda il capitolo 3). Si noti che la voce `-gpu=0` deve essere lasciata invariata.

In sintesi, ci sono due possibilità per eseguire le simulazioni:

- Si può creare dentro `home/swe/res` una cartella per la simulazione, nella quale vanno per prima cosa copiati i file di input. Nella stessa cartella va posizionato lo script simile a quello di Figura A.2, opportunamente corretto, da lanciare col comando `sbatch`. Allo stesso tempo, verrà creata automaticamente un'altra cartella col nome corrispondente dentro `scratch`, in cui verranno via via scritti i file di output.
- In alternativa, si può lavorare completamente in `scratch`, usando uno script analogo a quello di Figura A.3. Deve essere creata una cartella apposita in `scratch` con i file di input e lo script, da lanciare. Nella stessa cartella verranno via via scritti i file di output.

Nella cartella in cui è presente lo script compariranno anche due file chiamati `job_id.e` (in cui vengono stampati eventuali errori, ad esempio se il `job` si dovesse interrompere) e `job_id.o` (con eventuali output).

#### AGGIORNAMENTO DAL 2020:

Sul cluster è disponibile un nuovo nodo di calcolo con 2 nuove GPU, dotata di una coda apposita e riservata (gpu\_hylab). Per utilizzarla, vanno sostituiti i comandi iniziali dello script come segue:

```
#SBATCH --partition=gpu_hylab
#SBATCH --gres=gpu:v100:1
```

Analogamente, anche per le operazioni di decode deve essere predisposto un apposito script, da posizionare nella cartella dei file di input e da lanciare anch'esso con il comando `sbatch`. Anche i file decodificati appariranno nella cartella corrispondente in `scratch`. In questo caso, si farà riferimento alla coda `vrt` (anziché `gpu`). Le risorse richieste saranno ad esempio:

```
#SBATCH -p vrt -N1 -n1
#SBATCH --time 0-00:30:00
#SBATCH --mem=16G
```



```

1  #!/bin/sh
2
3  #SBATCH --time=0-11:59:00
4  #SBATCH --ntasks-per-node=1
5  #SBATCH --mem=32G
6  #SBATCH --partition=gpu
7  #SBATCH --gres=gpu:tesla:1
8  #SBATCH --job-name=lancia_simu_slurm.sh
9  #SBATCH --output=%x.o%j
10 #SBATCH --error=%x.e%j
11
12 export PATH=/hpc/share/tools/cuda-8.0.61/bin:$PATH
13 export LD_LIBRARY_PATH=/hpc/share/tools/cuda-8.0.61/lib64:$LD_LIBRARY_PATH
14
15 #=====
16 #INIZIO DEFINIZIONE PARAMETRI
17 #=====
18
19 #nome utente
20 nome_utente=$USER
21 #directory contenente input files
22 DIR_TEST="nome_cartella"
23 #directory in cui scrive i file di output
24 DIR_OUTPUT="nome_cartella"
25 #nome file con parametri (tempo iniziale, dt output, CFL number etc...)
26 input_file="input_esempio.txt"
27 #nome delle mappe di input
28 input_map="esempio"
29 #nome del file di output (in cui scrive time, dt, volumi etc...)
30 output_file="output_esempio.txt"
31 #nome file binario
32 swegpu=/hpc/home/$nome_utente/swe/bin/swegpu_single_tesi_cluster
33
34 #=====
35 #FINE DEFINIZIONE PARAMETRI
36 #=====
37
38 INPUT_DIR="/hpc/home/$nome_utente/swe/res/$DIR_TEST"
39 OUTPUT_DIR="/hpc/scratch/$nome_utente/$DIR_OUTPUT"
40
41 echo "HOSTNAME           : $HOSTNAME"
42 echo "INPUT_DIR           : $INPUT_DIR"
43 echo "OUTPUT_DIR          : $OUTPUT_DIR"
44 echo "input_file          : $input_file"
45 echo "output_file         : $output_file"
46 echo "input_map           : $input_map"
47 echo "swegpu              : $swegpu"
48 echo "CUDA_VISIBLE_DEVICES: $CUDA_VISIBLE_DEVICES"
49 echo '===== '
50 echo
51
52 rm -r "${OUTPUT_DIR}"
53 mkdir -p "${OUTPUT_DIR}"
54
55 cd "${OUTPUT_DIR}"
56
57 ln -s "${INPUT_DIR}/${input_file}"
58
59 for f in $INPUT_DIR/$input_map.???; do
60     ln -s "$f"
61 done
62
63 ${swegpu} ${input_file} -order=1 -multi=hi -gpu=0 > ${output_file}
64

```

Figura A.2 Esempio di script per lanciare una simulazione sul cluster (input in home, output in scratch)..

```

1  #!/bin/sh
2
3  #SBATCH --time=0-01:00:00
4  #SBATCH --ntasks-per-node=1
5  #SBATCH --mem=8G
6  #SBATCH --partition=gpu
7  #SBATCH --gres=gpu:tesla:1
8  #SBATCH --job-name=lancia_simu_slurm.sh
9  #SBATCH --output=%x.o%j
10 #SBATCH --error=%x.e%j
11
12 export PATH=/hpc/share/tools/cuda-8.0.61/bin:$PATH
13 export LD_LIBRARY_PATH=/hpc/share/tools/cuda-8.0.61/lib64:$LD_LIBRARY_PATH
14
15 #=====
16 #INIZIO DEFINIZIONE PARAMETRI
17 #=====
18
19 #nome utente
20 nome_utente=$USER
21
22 #directory contenente files
23 DIR_TEST="parma_cassa_5m"
24
25 #nome delle mappe di input
26 NOME_TEST="parma_cassa_5m"
27
28 #nome file binario
29 swegpu=/swe/bin/swegpu_single_tesi7_cluster
30
31 #=====
32 #FINE DEFINIZIONE PARAMETRI
33 #=====
34
35 #input e output nella stessa cartella
36 INPUT_DIR="/$SCRATCH/$DIR_TEST"
37 OUTPUT_DIR="/$SCRATCH/$DIR_TEST"
38
39 #nome file con parametri (tempo iniziale, dt output, CFL number etc...)
40 input_file=input_${NOME_TEST}.txt
41 #nome del file di output (in cui scrive time, dt, volumi etc...)
42 output_file=output_${NOME_TEST}.txt
43
44
45 echo "HOSTNAME           : $HOSTNAME"
46 echo "INPUT_DIR           : $INPUT_DIR"
47 echo "OUTPUT_DIR          : $OUTPUT_DIR"
48 echo "input_file          : $input_file"
49 echo "output_file         : $output_file"
50 echo "input_map           : $NOME_TEST"
51 echo "swegpu              : $swegpu"
52 echo "CUDA_VISIBLE_DEVICES: $CUDA_VISIBLE_DEVICES"
53 echo '===== '
54 echo
55
56 #riga di comando con opzioni
57 ${swegpu} ${input_file} -order=1 -multi=hi -gpu=0 > ${output_file}
58

```

Figura A.3 Esempio di script per lanciare una simulazione sul cluster (input e output in scratch).