# GENERAZIONE DEGLI SCENARI DI INONDAZIONE

## Dati

- raster dei puntatori del dominio (possibilmente a risoluzione idraulica, altrimenti a risoluzione idrologica)
- hazard maps nominali (= mappe di hazard generate con simulazioni idrauliche per un numero limitato di tempi di ritorno, a risoluzione idraulica)
- un insieme di sezioni sul reticolo (in un file di testo identico a quello usato da Continuum per le sezioni), che sono quelle dove operativamente saranno note le portate

## Procedura

### *1. Generazione delle aree di competenza*

(da fare solo una volta, deve essere ripetuta se cambiano le sezioni)

Dati necessari:

- raster dei puntatori
- sezioni sul reticolo

Si deve generare una mappa di "aree di competenza". Un'area di competenza è l'area a monte di una data sezione da cui vengono eliminate le aree di monte di eventuali altre sezioni a monte della sezione stessa. La mappa finale è quindi un mosaico di queste aree, ognuna delle quali avrà come valore un identificativo unico corrispondente alla propria sezione (per esempio il codice della sezione stessa).
In pratica, l'area di competenza di una sezione è SOLO l'area in cui l'acqua viene drenata direttamente dalla sezione stessa, senza passare per eventuali sezioni di monte. Idealmente il mosaico dovrebbe coprire l'intero bacino, ma con le sezioni con idrometri non succede praticamente mai, quindi il resto o viene riempito con un nearest neighbor (sconsigliato) oppure va assegnato a mano espandendo in maniera sensata (se c'è) le aree già definite.
La mappa finale deve essere esattamente con la stessa estensione e risoluzione delle mappe di hazard, quindi se il raster dei puntatori non era disponibile sulla griglia idraulica, la mappa delle aree di competenza andrà rigrigliata sulla griglia idraulica, con metodo nearest neighbor.

### *2. Interpolazione delle Hazard Maps*

(da fare solo una volta)

Dati necessari:

- hazard maps nominali
- script: `interpolazione_hazardmaps.m`

Date le hazard maps nominali (quindi quelle iniziali corrispondenti a un numero ridotto di tempi di ritorno), vanno interpolate le hazard maps per tutti gli altri tempi di ritorno necessari, tipicamente da 2 a 1000 anni. Per questo c'è l'algoritmo Virtual Lake (interpolazione_hazardmaps.m) che genera, con una procedura abbastanza complicata, tutte le mappe di hazard.
Una descrizione dettagliata dell'algoritmo è in fondo a questo documento.

***3. Generazione degli scenari di inondazione***
(operativa)

Dati necessari:

- hazard maps interpolate (per tempi di ritorno da 2 a 1000 anni)
- mappa aree di competenza a risoluzione idraulica
- n-upla di tempi dei ritorno delle portate verificatisi nelle n sezioni nell'istante di interesse
- script: `generazione_flood.m`

Per generare la mappa di inondazione in un dato istante in cui siano note le portate nelle varie sezioni del dominio, le portate vanno convertite in tempi di ritorno e fornite nello stesso ordine con cui compaiono nel file iniziale delle sezioni. Lo script `generazione_flood.m` carica la mappa delle aree di competenza (i cui valori sono i codici delle sezioni, è presente anche una variabile con i codici stessi che mantengono sempre lo stesso ordine delle sezioni) e produce quindi una mappa di inondazione come mosaico: per ogni area di competenza viene usato il frammento di mappa di hazard corrispondente al tempo di ritorno verificato nella sezione data.

## Algoritmo di interpolazione delle Hazard Maps

*Nota*: in entrambi gli algoritmi, vecchio e nuovo (Virtual Lake), le mappe di hazard nominali DEVONO essere strettamente crescenti in ogni pixel col tempo di ritorno, anche di pochissimo. Gli algoritmi stessi, quando le leggono, se necessario le riordinano e forzano una crescita stretta per ogni pixel.

### *VECCHIO ALGORITMO:*

Nella vecchia versione, quella di cui abbiamo scoperto i problemi, l'interpolazione delle mappe di hazard avviene come segue.
Anzitutto si prendono le mappe di hazard nominali (T= 25, 50, 100, 200, 500 e 1000), si aggiunge una mappa completamente nulla all'inizio, e si interpolano due curve: una che descrive l'andamento dei volumi (somma dei valori di battente di tutti i pixel) col tempo di ritorno, e l'altra che descrive l'andamento delle aree inondate (numero dei pixel non nulli). L'interpolazione avviene con polinomi hermitiani (pchip) che garantiscono la monotonicità tra un punto e l'altro.
Dato un tempo di ritorno, per esempio 60, si individuano le due mappe di hazard nominali immediatamente sotto e sopra (quindi 50 e 100), vengono individuati i pixel che sono

presenti nella 100 ma NON nella 50 (quindi quelli che "si asciugano" andando verso quella inferiore, chiamiamo questi pixel "corona", mentre i pixel comuni alle due nominali li chiamiamo "nucleo"), e i corrispondenti battenti vengono ordinati per valore crescente.

In base alle due curve di cui sopra, sono noti il volume (V60) e il numero di pixel "bagnati" totali (N60) che dovrà avere la mappa a 60 anni: l'algoritmo consiste nell'eliminare, dalla nominale superiore (100) abbastanza pixel di corona (eliminati in ordine crescente di battente) da arrivare al numero di pixel giusto, N60 appunto, dopodiché ciò che rimane (pixel di nucleo + pixel di corona non azzerati) viene moltiplicato tutto per un fattore costante in modo da raggiungere il volume V60.
Qui di seguito c'è il codice (lo script completo contiene in più solo parti di gestione dei file, percorsi, ecc..):

```
% calcolo area e volume delle nominali
TR_add=1:1000;
TR=[25 50 100 200 500 1000];
[V,N]=deal(NaN(1,length(TR)));
for t=1:length(TR)
    load([inpath_hazmaps,'/Hazmap_',sprintf('%04.0f',TR(t)),'.mat']);
    mappa=single(mappa_h);
    if t==1
        [nrows,ncols]=size(mappa);
        HazMaps_3d=zeros(nrows,ncols,length(TR)+1);
    end
    HazMaps_3d(:,:,t+1)=mappa;
    V(t)=nansum(mappa(:));
    N(t)=nansum(mappa(:)>0);
end


% Funzioni di area e volume
tttt=0:1000;
vvvv=pchip([0,TR],[0,V],tttt);
nnnn=pchip([0,V],[0,N],vvvv);
T=[0,TR];
V=[0,V];
N=[0,N];


% CICLO SUI TEMPI DI RITORNO
for t=1:length(TR_add)

    TT=TR_add(t);

    % Tempi di ritorno nominali di riferimento
    i2=find((TT-T)<0,1,'first');
```

```
    i1=i2-1;

    % Interpolazione area e volume
    Tresiduo=TT;
    Vresiduo=interp1(tttt,vvvv,double(Tresiduo));
    Nresiduo=ceil(interp1(vvvv,nnnn,Vresiduo));
    V_TT=Vresiduo;
    N_TT=Nresiduo;
    Npixel_da_elim=floor(N(i2)-N_TT);

    % Calcolo mappa con tempo di ritorno TT
    h1=squeeze(HazMaps_3d(:,:,i1));
    h2=squeeze(HazMaps_3d(:,:,i2));
    indici1_2=setdiff(find(h2),find(h1));
    valori1_2=h2(indici1_2);
    [val_sort,indici_sort]=sort(valori1_2);
    mappa_h_nominale=h2;
    mappa_h_nominale(indici1_2(indici_sort(1:Npixel_da_elim)))=0;
    V_mappa=nansum(mappa_h_nominale(:));
    mappa_h=single(mappa_h_nominale)*(V_TT/V_mappa);

    % salvataggio mappa
    save([outpah,'Hazmap__T',sprintf('%04.0f',TT)],'-v7.3','mappa_h');

end
```

Questo metodo però crea problemi perchè sostanzialmente fa un'assunzione forte, che si è rivelata non vera a causa dei pattern spaziali e cioè che le mappe nominali seguano la legge di riscalatura che viene usata dall'algoritmo. Precisamente l'algoritmo presuppone che nel passaggio da una mappa nominale a quella inferiore, i pixel che scompaiono siano tutti quelli con i battenti più bassi o, detto in altri termini, che i battenti del nucleo siano tutti uguali o maggiori a quelli della corona. Invece non è così (le aree difese contribuiscono a volte al problema, ma anche senza quelle ci sono moltissimi pixel in questa situazione anche nelle mappe indifese) ed è in effetti normale: le mappe vengono da un modello idraulico su un dem complesso e quindi può verificarsi benissimo questa situazione, anche molto spesso, come in effetti avviene.
Applicando quindi questo algoritmo, man mano che ci si avvicina alla nominale inferiore, le mappe prodotte sono sempre più diverse da essa e quindi si creano forti discontinuità locali. Inoltre, siccome l'algoritmo forza determinati valori dei volumi, il fattore moltiplicativo è molto irregolare e può benissimo essere maggiore di 1, da cui i vari problemi osservati di inversione dell'andamento dei livelli al variare del tempo di ritorno.
Un esempio semplice può essere il seguente: supponiamo che le mappe siano costituite da soli 4 pixel allineati e la 100 e la 50 siano le seguenti:

`mappa_100=[0 60 100 30];` (N100=3, V100=190)
`mappa_50= [0  0  80 20];` (N50=2, V50=100)

Supponiamo inoltre che N60=2 (quindi devono rimanere solo due pixel non nulli) e che V60=150.
Seguendo l'algoritmo: il pixel 2 costituisce tutta la corona. Per arrivare a N60 quindi va eliminato.
Rimane quindi la mappa, da riscalare, `[0  0 100 30]`, il cui volume è 130. Per arrivare a V60=150, il fattore moltiplicativo è quindi 150/130 che è maggiore di 1, e quindi si ottengono, nella 60, valori maggiori della mappa 100...
Tutto questo succede perchè nel nucleo c'è un pixel (il 4) che è minore del pixel di corona (il 2). Con corone più estese e mappe più complesse si ottengono aree sia sotto che sopra i valori che dovrebbero avere.
Di questi problemi purtroppo ce ne siamo accorti solo controllando molto in dettaglio alcuni degli scenari generati per Sao Tomè, perchè aveva un dominio abbastanza piccolo da notare queste anomalie, in quelle più grandi si confondono meglio, ma abbiamo verificato che ci sono e sono molto estese.

*NUOVO ALGORITMO (VIRTUAL LAKE):*

Per risolvere il problema ho implementato il nuovo algoritmo, che è inventato da zero e completamente diverso (e più complicato). L'algoritmo consiste nel trasformare la mappa in una sorta di invaso, un "lago virtuale" con un dem del fondale costruito apposta per poter simulare un processo di svuotamento.

Come prima, si identificano le due mappe nominali (50 e 100) "intorno" al tempo di ritorno per cui ricavare la mappa (60).
A questo punto si suddividono di nuovo i pixel della mappa nominale superiore (100) in corona e nucleo, e si crea il dem virtuale: nella zona di nucleo si prende il valore del battente massimo della mappa nominale inferiore (50), e gli si sottraggono tutti gli altri battenti del nucleo inferiore stesso, ottenendo le quote del "fondale" (in pratica il fondale del nucleo è tale che la nominale inferiore lo riempie perfettamente fino all'orlo con una quota del livello idrico virtuale pari appunto al battente massimo). Si assume che la quota 0 corrisponda al punto più basso del fondale del nucleo. Nella corona, il dem del fondale è ottenuto sommando il massimo dei battenti del nucleo della nominale inferiore (lo stesso valore appena usato) al massimo dei battenti della corona della nominale superiore (100) a cui vengono sottratti tutti i battenti sempre della nominale superiore.

Definito il dem del fondale totale (nucleo+corona), si definisce una mappa di fattori di scaling: questi fattori valgono 1 in tutti i pixel della corona, mentre nel nucleo sono tali che, per ogni pixel, la differenza tra i battenti del nucleo superiore e di quello inferiore (cioè di quanto cala l'acqua tra la 100 e la 50) viene "espansa/compressa" fino a riempire totalmente ed esattamente il lago fino alla quota massima che è, per definizione, uguale al battente massimo del nucleo inferiore PIÙ il battente massimo della corona.

In questo modo il lago ha due livelli idrici estremi per definizione: il minimo è alla quota "battente massimo del nucleo inferiore" e i battenti corrispondenti sono esattamente quelli della nominale inferiore, mentre il massimo è alla quota "battente massimo del nucleo inferiore + battente massimo della corona" e i battenti corrispondenti, una volta riscalati con i fattori di scaling di prima, ricostruiscono esattamente la nominale superiore. I battenti "virtuali" (=non ancora riscalati) del lago sono definiti in ogni pixel come la differenza tra la

quota del livello idrico e la quota corrispondente del dem, dove questa differenza non sia negativa.

Facendo aumentare il livello idrico tra queste due quote limite, per costruzione vengono generate mappe per cui:
- l' area (numero di pixel con battente non nullo) è non decrescente
- il volume è strettamente crescente
- i battenti sono strettamente crescenti per ogni pixel (sia nel nucleo che nella corona)
- il tutto segue una geometria compatibile con le due nominali
- ai livelli estremi ricostruisce esattamente entrambe le nominali
- il cambiamento di tutte le caratteristiche (area, volume, battenti locali) è graduale

Per fare in modo che il campionamento delle mappe sia almeno approssimativamente corretto (cioè cresca in maniera compatibile con le mappe nominali), viene re-interpolata la stessa curva dei volumi del vecchio algoritmo. Poi vengono generare le mappe corrispondenti a parecchi livelli idrici virtuali, si calcolano i corrispettivi volumi reali e si inverte la curva livelli virtuali- volumi reali per ottenere i livelli idrici che permettono di rispettare il vincolo della curva dei volumi suddetta.