# Progetto di Geometria Computazionale: simulazione del movimento ondoso di un fluido utilizzando Kass e Miller

Stefano Ceroni, Sara Toia

Luglio 2011

## 1 Introduzione

Il metodo di Kass e Miller [1] per la simulazione di un moto ondoso di un fluido non ha come scopo una rigorosa simulazione dal punto di vista fisico come quella che si ottiene applicando le equazioni di Navier-Stokes, ma i suoi punti di forza consistono nell'essere veloce, stabile e facilmente implementabile. Tale metodo si fonda su tre semplificazioni:

- il fluido è reso come una superficie in valori di altezza;
- viene ignorata la velocità verticale;
- la velocità orizzontale è costante.

Queste tre assunzioni limitano la simulazione in quanto non è possibile ricreare una corrente verticale e il moto ondoso non genera onde che si infrangono, ma funziona correttamente per la maggior parte dei moti ondosi non troppo burrascosi osservabili in mare aperto.

## 2 Simulazione tramite Kass-Miller

Chiamiamo $b(x)$ l'altezza del suolo, dal livello 0, del nostro bacino d'acqua e $h(x)$ l'altezza dell'acqua, sempre dal livello 0. Quindi la profondità dell'acqua è definita come $d(x) = h(x) - b(x)$. In base alle assunzioni fatte sopra, le equazioni del moto ondoso di un fluido poco profondo possono essere scritte come di seguito:

$$\frac{\partial u}{\partial t} + g\,\frac{\partial h}{\partial x} = 0 \tag{1}$$

$$\frac{\partial h}{\partial t} + d\,\frac{\partial u}{\partial x} = 0 \tag{2}$$

dove $u(x)$ è la velocità orizzontale dell'onda e $g$ è l'accelerazione di gravità. Queste due equazioni esprimono rispettivamente la legge di Newton $F = ma$ e

la conservazione del volume. Differenziando la prima rispetto a $x$ otteniamo:

$$\frac{\partial^2 u}{\partial x\,\partial t} + g\,\frac{\partial^2 h}{\partial x^2} = 0 \qquad \frac{\partial^2 u}{\partial x\,\partial t} = -g\,\frac{\partial^2 h}{\partial x^2}$$

e differenziando la seconda rispetto a $t$ otteniamo:

$$\frac{\partial^2 h}{\partial t^2} + d\,\frac{\partial^2 u}{\partial x\,\partial t} = 0$$

Sostituendo la prima nella seconda

$$\frac{\partial^2 h}{\partial t^2} = dg\,\frac{\partial^2 h}{\partial x^2} \tag{3}$$

si ottiene l'equazione che esprime la legge del moto di un'onda di velocità $v = \sqrt{dg}$. Si tratta quindi di risolvere la 3.

## 2.1 Discretizzazione

Poiché noi trattiamo spazio discreto, è necessario effettuare una discretizzazione dell'equazione 3. Per calcolare le derivate si procede come di seguito:

$$\frac{\partial h_i}{\partial t} = \left(\frac{d_{i-1} + d_i}{2\Delta x}\right) u_{i-1} \; - \; \left(\frac{d_i + d_{i+1}}{2\Delta x}\right) u_i$$

$$\frac{\partial u_i}{\partial t} = -g\,\frac{(h_{i+1} - h_i)}{\Delta x}$$

con $\Delta x$ la larghezza di una singola colonna o la distanza tra una colonna e la successiva. Si può quindi ottenere la discretizzazione dell'equazione 3:

$$\frac{\partial^2 h_i}{\partial t^2} = -\,g\,\left(\frac{d_{i-1} + d_i}{2(\Delta x)^2}\right)(h_i - h_{i-1}) + g\,\left(\frac{d_i + d_{i+1}}{2(\Delta x)^2}\right)(h_{i+1} - h_i) \tag{4}$$

## 2.2 Integrazione

Per integrare i risultati, poiché la simulazione verrà scandita da un tempo discreto, utilizziamo un metodo del primo ordine, implicito:

$$\dot{h}(n) = \frac{h(n) - h(n-1)}{\Delta t} \tag{5}$$

$$\ddot{h}(n) = \frac{\dot{h}(n) - \dot{h}(n-1)}{\Delta t} \tag{6}$$

con $n$ che rappresenta il tempo discreto e $\Delta t$ il passo di integrazione. Per ricavare $h(n)$ dalle equazioni descritte sopra (5 e 6) le giriamo e le mettiamo a sistema:

$$\begin{cases} h(n) = h(n-1) + \Delta t\,\dot{h}(n) \\ \dot{h}(n) = \dot{h}(n-1) + \Delta t\,\ddot{h}(n) \end{cases}$$

e ricaviamo:

$$h(n) = h(n-1) \,+\, \Delta t \,\left(\dot{h}(n-1) \,+\, \Delta t\, \ddot{h}(n)\right)$$

$$= h(n-1) \,+\, \Delta t\, \dot{h}(n-1) \,+\, (\Delta t)^2\, \ddot{h}(n) \tag{7}$$

Mettendo a sistema la 7 con la 5:

$$\begin{cases} h(n) = h(n-1) \,+\, \Delta t\, \dot{h}(n-1) \,+\, (\Delta t)^2\, \ddot{h}(n) \\ \\ \dot{h}(n-1) = \dfrac{h(n-1) - h(n-2)}{\Delta t} \end{cases}$$

si ottiene:

$$h(n) = \, h(n-1) \,+\, \cancel{\Delta t} \left(\frac{h(n-1) \,-\, h(n-2)}{\cancel{\Delta t}}\right) \,+\, (\Delta t)^2\, \ddot{h}(n)$$

$$= \, 2h(n-1) \,-\, h(n-2) \,+\, (\Delta t)^2\, \ddot{h}(n) \tag{8}$$

A questo punto possiamo sostituire l'ultimo termine rimasto, $\ddot{h}(n)$, che rappresenta la derivata seconda, con l'equazione 4:

$$\begin{cases} \dfrac{\partial^2 h_i}{\partial t^2} = -\, g \,\left(\dfrac{d_{i-1} + d_i}{2(\Delta x)^2}\right)(h_i - h_{i-1}) + g \,\left(\dfrac{d_i + d_{i+1}}{2(\Delta x)^2}\right)(h_{i+1} - h_i) \\ \\ h(n) = \, 2h(n-1) \,-\, h(n-2) \,+\, (\Delta t)^2\, \ddot{h}(n) \end{cases}$$

Sostituendo la prima nella seconda otteniamo:

$$\begin{aligned} h_i(n) \;&= 2\, h_i(n-1) \,-\, h_i(n-2) \\ &\quad -\, g\,(\Delta t)^2 \left(\frac{d_{i-1} + d_i}{2(\Delta x)^2}\right)(h_i(n) - h_{i-1}(n)) \\ &\quad +\, g\,(\Delta t)^2 \left(\frac{d_i + d_{i+1}}{2(\Delta x)^2}\right)(h_{i+1}(n) - h_i(n)) \end{aligned} \tag{9}$$

Questa equazione è non lineare poiché $d$ dipende da $h$. Per renderla tale, al fine di semplificare i calcoli, consideriamo, ad ogni iterazione, $d$ come costante. Grazie a questa linearizzazione possiamo riformulare tutto come un sistema lineare:

$$A\, h(n) = 2\, h(n-1) - h(n-2) \tag{10}$$

in cui $A$ è una matrice quadrata tridiagonale simmetrica:

$$A \,=\, \begin{bmatrix} e_0 & f_0 & & & & & \\ f_0 & e_1 & f_1 & & & & \\ & f_1 & e_2 & f_2 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & f_{n-3} & e_{n-2} & f_{n-2} & \\ & & & & f_{n-2} & e_{n-1} & \end{bmatrix} \tag{11}$$

e gli elementi sono definiti come segue:

$$f_i \;=\; -\,g\,(\Delta t)^2\,\left(\frac{d_{i+1}+d_i}{2(\Delta x)^2}\right)$$

$$e_0 \;=\; 1 \;+\; g(\Delta t)^2\,\left(\frac{d_0+d_1}{2(\Delta x)^2}\right)$$

$$e_i \;=\; 1 \;+\; g(\Delta t)^2\,\left(\frac{d_{i-1}+2d_i+d_{i+1}}{2(\Delta x)^2}\right)$$

$$e_{n-1} \;=\; 1 \;+\; g(\Delta t)^2\,\left(\frac{d_{n-2}+d_{n-1}}{2(\Delta x)^2}\right) \tag{12}$$

É possibile ottenere degli effetti di viscosità del liquido modificando la 10 come segue:

$$A\,h(n) = h(n-1) \;+\; (1-\tau)\,(h(n-1) - h(n-2)) \tag{13}$$

in cui $0 \leq \tau \leq 1$ indica il grado di viscosità.

## 2.3 Risoluzione del sistema lineare

Per poter ricavare le altezze della superficie del fluido è necessario risolvere il sistema lineare 10. Di seguito vengono descritti rapidamente due metodi di risoluzione. Il primo, il Metodo di Jacobi, è un metodo iterativo di risoluzione, cioè ottiene la soluzione avvicinandosi passo dopo passo, mentre il secondo, l'algoritmo di Thomas, giunge alla soluzione per eliminazione a blocchi ed è specifico per i sistemi lineari con matrice tridiagonale. Nell'applicazione da noi sviluppata, abbiamo utilizzato il secondo metodo.

### 2.3.1 Il Metodo di Jacobi

Il Metodo di Jacobi [2] si prefigge di risolvere un sistema lineare in modo iterativo. Per risolvere un sistema lineare

$$Ax = b$$

si decompone $A$ in

$$A = D - L - U$$

con $D$ che rappresenta la diagonale, $-L$ la triangolare inferiore e $-U$ la triangolare superiore. In Jacobi si pone

$$M = D$$

$$N = L + U$$

Quindi si itera:

$$x^{(k+1)} = D_{-1}\,(L+U)\,x^{(k)} + D^{-1}b$$

cioè

$$x_i^{(k+1)} = -\frac{1}{a_{i,i}} \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{i,j} x_j^{(k)} + \frac{b_i}{a_{i.i}}$$

calcolato per ogni $i$. Tale metodo ha una complessità di $3n^2+2n$ ed è apprezzato per la sua facile parallelizzazione.

### 2.3.2 L'algoritmo di Thomas

L'algoritmo di Thomas [3] è un metodo diretto di risoluzione di un sistema lineare la cui matrice dei coefficienti è tridiagonale:

$$\begin{bmatrix} d_1 & u_1 & & & \\ l_1 & d_2 & u_2 & & \\ & \ddots & \ddots & \ddots & \\ & & l_{n-2} & d_{n-1} & u_{n-1} \\ & & & l_{n-1} & d_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}$$

Il primo passo è quello di trasformare la matrice del sistema in una triangolare superiore procedendo nel seguente modo:

$$m = \frac{l_i}{d_i}$$

$$d_{i+1} = d_{i+1} - m\, u_i$$

$$b_{i+1} = b_{i+1} - m\, b_i$$

calcolato per ogni $i = 1, 2, \ldots, n$. Otteniamo quindi un sistema del tipo:

$$\begin{bmatrix} d'_1 & u_1 & & & \\ 0 & d'_2 & u_2 & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & d'_{n-1} & u_{n-1} \\ & & & 0 & d'_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b'_1 \\ b'_2 \\ \vdots \\ b'_{n-1} \\ b'_n \end{bmatrix}$$

A questo punto si ottengono i valori del vettore delle incognite procedendo a ritroso:

$$x_n = \frac{b'_n}{d'_n}$$

$$x_{n-1} = \frac{b'_{n-1} - u_{n-1} x_n}{d'_{n-1}}$$

$$x_{n-2} = \frac{b'_{n-2} - u_{n-2} x_{n-1}}{d'_{n-2}}$$

$$\vdots$$

$$x_0 = \frac{b'_0 - u_0 x_1}{d'_0}$$

Questo metodo è in grado di risolvere il sistema in tempo lineare ma risulta difficilmente parallelizzabile.

## 2.4 Conservazione del volume

Nonostante l'equazione 10 sia stata ricavata dalla 2 che esprime la legge di conservazione del volume, essa non è garantita poiché è possibile che si possa verificare una situazione in cui si ha $h_i < b_i$ per qualche $i$, cioè il livello del fluido può essere inferiore al livello del suolo. In questa situazione avremmo $d_i = h_i - b_i < 0$ reso poi $d_i = 0$. Questo porterebbe il sistema a compensare questa mancanza aggiungendo fluido nelle altre colonne, aumentando a tutti gli effetti il livello globale del fluido nel sistema. Si rende necessaria quindi una correzione. Chiamiamo

$$V(n) = \sum_{i=1}^{K} d_i(n)$$

il volume di tutte le $K$ colonne al tempo $n$. La correzione consiste in:

$$h_i(n) = h_i(n) + \frac{V(n) - V(n-1)}{K} \tag{14}$$

per ogni $i = 1, \ldots, K$.

## 2.5 L'algoritmo

Qui di seguito viene illustrato l'algoritmo:

```
1  {inizializza b, h, h_old}
2
3  begin
4
5      Volume = getVolume();
6
7      while !fine do begin
8          {perturbazione del volume d'acqua}
9          d = h - b;
10
11         if d<0 then d := 0;
12
13         {risolvo}
14         {A h = h + (1-tau)(h - h_old)}
15         {e trovo quindi h(j)}
16
17         V = getVolume();
18         h = h + (Volume - V) * 1/K;
19
20         if h < b then begin
21             h = b - eps;
22             h_old = b - eps;
23         end;
24     end;
25 end;
```

# 3 Superficie liquida in 3D

Finora abbiamo visto come implementare un sistema in grado di simulare il comportamento di una sezione laterale di un liquido. Noi vogliamo estendere il metodo descritto sopra in modo da poter descrivere il comportamento di una intera superficie. In [1] si trova una breve spiegazione di come procedere in questo senso. Si tratta di sostituire nell'equazione 3 al posto della derivata, il laplaciano. Ricordando che l'operatore di Laplace è un operatore differenziale del secondo ordine ed è definito in questo modo:

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

l'equazione 3 diventa:

$$\frac{\partial^2 h}{\partial t^2} = dg \left( \frac{\partial^2 h}{\partial x^2} + \frac{\partial^2 h}{\partial y^2} \right)$$

Per risolvere tale equazione procediamo utilizzando il metodo delle direzioni alternate, ovvero dividiamo il calcolo dell'equazione in due sottoiterazioni separando le due componenti. Nella prima risolviamo l'equazione

$$\frac{\partial^2 h}{\partial t^2} = dg \frac{\partial^2 h}{\partial x^2} \tag{15}$$

mentre nella seconda iterazione risolviamo l'equazione

$$\frac{\partial^2 h}{\partial t^2} = dg \frac{\partial^2 h}{\partial y^2} \tag{16}$$

In ogni sottoiterazione dobbiamo risolvere il sistema lineare tridiagonale per ogni sezione della superficie. L'algoritmo quindi diventa:

```
1  {inizializza b, h, h_old}
2
3  begin
4
5      Volume = getVolume();
6
7      while !fine do begin
8          {perturbazione del volume d'acqua}
9          d = h - b;
10
11         if d<0 then d = 0;
12
13         {risolvo il sistema
14         A h = h + (1-tau)(h - h_old)
15         sulle righe e trovo quindi h}
16
17         {risolvo il sistema con i nuovi valori di h
18         A h = h + (1-tau)(h - h_old)
19         sulle colonne e trovo quindi h}
20
21         V = getVolume();
22         h = h + (Volume - V) * 1/K;
23
24         if h < b then begin
25             h = b - eps;
26             h_old = b - eps;
27         end;
28     end;
29 end;
```

## Riferimenti bibliografici


[1] M. Kass, G. Miller
Rapid, Stable Fluid Dynamics for Computer Graphics
Computer Graphics, 1990

[2] Wikipedia: Metodo di Jacobi
`http://it.wikipedia.org/wiki/Metodo_di_Jacobi`

[3] Wikipedia: Tridiagonal matrix algorithm
`http://en.wikipedia.org/wiki/Tridiagonal_matrix_algorithm`