POLITECNICO DI MILANO

Scuola di Ingegneria dei Sistemi

Corso di Studi in INGEGNERIA MATEMATICA


Tesi di Laurea Magistrale

# Simulazione della dinamica di imbarcazioni da canottaggio usando geometrie reali e integrazione con la dinamica dei vogatori

Relatore: Prof. Luca Formaggia

Co-Relatore: Prof. Edie Miglio

Simone Palamara

matricola 751475

Anno Accademico 2011-2012

**Sommario**

In questo lavoro di tesi è stato proposto un modello completo per la dinamica di una imbarcazione di canottaggio. Per risolvere la fluidodinamica intorno allo scafo si è scelto un modello 3D multistrato. L'interazione del fluido con lo scafo è stato modellizzata attraverso un vincolo di galleggiamento che impone l'impenetrabilità tra l'imbarcazione e il fluido, risolto con il metodo di Uzawa. Il software sviluppato permette di risolvere il problema fluidodinamico sia per imbarcazioni descritte tramite una funzione della forma $z = z(x, y)$ sia per imbarcazioni qualunque purchè si disponga di una triangolazione della superficie dello scafo. Il modello fluidodinamico è poi accoppiato a un modello per risolvere la dinamica del sistema imbarcazione-vogatori, implementato all'interno del software *Kimè*, sviluppato presso il il *Gruppo di Modellistica e Calcolo Scientifico* (MOX) del Politecnico di Milano. L'accoppiamento è ottenuto tramite una procedura staggered con predittore strutturale. Vengono presentati i risultati delle simulazione relative all'andamento di un'imbarcazione di canottaggio in condizioni di regata per mostrare l'efficacia del modello proposto.

# Indice

# Elenco delle figure

# Elenco delle tabelle

# Introduzione

La tesi presentata si pone come obbiettivo lo studio completo della dinamica di una imbarcazione da canottaggio in condizioni di regata, finalizzata alla implementazione di un software capace di fornire indicazioni utili sia nel design di nuove imbarcazioni sia per lo studio delle tecniche di voga. La difficoltà di questo problema, che si inserisce nella categoria di problemi di interazione fluido-struttura, sta nella complessità della dinamica del moto dello scafo, condizionato sia dalla fluidodinamica che dalla dinamica dei vogatori. Per la parte fluidodinamica il modello matematico scelto si basa sull'equazioni di Navier-Stokes *quasi-3D* con superficie libera. Rispetto alle equazioni di Navier-Stokes classiche, vengono introdotte importanti approssimazioni che sfruttano alcune caratteristiche fisiche del problema in esame. Si assume che il fluido sia incompribile, a densità costante e che i termini viscosi nella direzione orizzontale siano trascurabili rispetto alla direzione verticale, cioè ci si riconduce ad un modello matematico paragonabile a quello delle *Shallow Water* o acque basse, assunzione plausibile nel caso in cui si modelli la fluidodinamica per imbarcazione di canottaggio. Tuttavia, a differenza del modello shallow-water classico, nel nostro modello si tiene conto in modo più completo della dinamica indotta dalle componenti verticali della velocità. Il dominio viene diviso, nella direzione verticale, in diversi strati di spessore fissato, ciascuno parallelo al piano orizzontale, permettendo di discretizzare il dominio orizzontale con una griglia triangolare non strutturata, che viene poi replicata nella direzione verticale. Questa scelta porta alla definizione di un modello 3D multistrato (*3D Multi Layer Shallow Water Equations*), descritto in [EM02] e in [LF08]. Per modellare l'interazione del fluido con lo scafo si è dovuto integrare nel modello fluidodinamico un vincolo di galleggiamento che imponesse l'impenetrabilità tra l'imbarcazione e il fluido. Ci si è così ricondotti a un problema di ottimizzazione vincolata, risolto con il *metodo di Uzawa*, utilizzando il classico approccio con i moltiplicatori di Lagrange e sostituendo il problema vincolato con una successione di problemi non vincolati.

Il modello introdotto è implementato nel codice C++ *Stratos++*, evoluzione del codice Fortran *Stratos*. Questa versione permetteva tuttavia di interfacciarsi solo con scafi rappresentati da funzioni analitiche, e quindi era negata la possibilità di testare il modello con geometrie provenienti da imbarcazione reali. Nella seconda versione *Stratos++V.2*, implementata in questo lavoro di tesi, si è reso possibile utilizzare il modello per analizzare la dinamica di qualunque imbarcazione di canottaggio, purchè si abbia a disposizione una triangolazione della superficie dello scafo. L'utilizzo di superfici triangolate per rappresentare l'imbarcazione ci ha portato a dover definire delle procedure per il calcolo della quota dello scafo e delle forze fluidodinamiche che agiscono su di esso. Quest'ul-

timo punto risulta di fondamentale importanza in quanto le forze fluidodinamiche che agiscono sullo scafo costituiscono una delle informazioni necessarie per risolvere il problema alla struttura. E questo ci porta al secondo importante cambiamento introdotto in *Stratos++V.2*, cioè la possibilità di accoppiarsi con un software che modellizzi la dinamica del sistema imbarcazione-vogatori, che per questo lavoro di tesi sarà il software *Kimè*, un codice sviluppato presso il *Gruppo di Modellistica e Calcolo Scientifico* (MOX) del Politecnico di Milano, che permette di riprodurre la dinamica imposta dal movimento dei vogatori sull'imbarcazione.

Il lavoro è articolato nel seguente modo. Nel capitolo 1 si presenta l'impianto teorico necessario per affrontare un problema di interazione fluido-struttura, presentando in 1.1 il modello implementato in *Stratos++* per risolvere la fluidodinamica intorno allo scafo, introducendo brevemente in 1.2 il modello alla base di *Kimè*, e infine mostrando in 1.3 alcune possibili strategie di accoppiamento dei due modelli. Nel capitolo 2 presentiamo il codice *Stratos++.2*, concentrando l'attenzione sulle nuove funzionalità introdotte. In particolare vogliamo evidenziare il grande impegno profuso per sviluppare dei metodi computazionalmente efficienti per la ricostruzione della quota dello scafo e la valutazione delle forze fluidodinamiche. Per raggiungere questo obbiettivo abbiamo introdotto nuove strutture e classi, le quali utilizzano librerie esterne ottimizzate e strutture ed algoritmi della STL, *Standard Template Library*. Infine nel capitolo 3 vengono esibiti i risultati di alcune simulazioni, introdotte per validare il codice e le modifiche apportate, e per riprodurre la dinamica di una vera imbarcazione di canottaggio.

# Capitolo 1

# Il modello matematico

L'obiettivo di questo studio è analizzare la dinamica di un'imbarcazione da canottaggio che si muove all'interno di un canale, soggetta alle forze imposte dai vogatori, oltre che alle reazioni che la fluidodinamica impone sullo scafo. Si tratta di un tipico problema di interazione fluido-struttura, con la particolarità che il fluido, viscoso e incomprimibile, è libero di muoversi, in modo da determinare autonomamente la posizione della sua superficie. Problemi di questo tipo vengono definiti *a superficie libera*. La risoluzione della fluidodinamica attorno all'imbarcazione è implementata nel software *Stratos++*, che come vedremo nella sezione 1.1, si basa sulle equazioni di Navier-Stokes *quasi-3D* con superficie libera. Nella sezione 1.2, presenteremo invece brevemente il modello utilizzato nel software *Kimè* per la dinamica dell'imbarcazione dovuta alla presenza dei vogatori. Infine nella sezione 1.3, si presenteranno due tecniche per accoppiare in modo numericamente stabile i due modelli introdotti.

## 1.1 Il risolutore fluidodinamico

I fluidi incomprimibili sono tipicamente descritti dalle equazioni di Navier-Stokes:

$$\begin{cases} \dfrac{D\mathbf{v}}{Dt} - \operatorname{div}(\mu \nabla \mathbf{v}) + \dfrac{1}{\rho}\nabla p = 0 \\ \\ \operatorname{div} \mathbf{v} = 0 \end{cases} \tag{1.1}$$

dove le (1.1) esprimono rispettivamente la conservazione del momento e della massa. Tuttavia la presenza di un dominio che varia nel tempo necessita di particolare attenzione, specialmente dal punto di vista numerico. Dato che le simulazioni riguardano un'imbarcazione da canottaggio, nel seguito ipotezzeremo che

- la superficie libera sia cartesiana, cioè che i fronti d'onda non si rompino mai;
- l'estensione del dominio nel piano $xy$ è predominante rispetto all'altezza della superficie libera, assumendo quindi valide le approssimazioni delle *Shallow Water* per semplificare le equazioni di Navier-Stokes.

Indichiamo con $\hat{\Omega}$ il dominio tridimensionale contenente il fluido che, per via della geometria del campo di regata, è possibile immaginare essere un parallelepipedo. Chiamando con $\Omega$ la base di tale dominio, collocata nel piano $xy$, si può definire il riferimento verticale imponendo che il fluido sia contenuto tra il fondo del canale, a quota $z = 0$, e la superficie libera, a quota $z = \eta(x, y, t)$. Quindi il volume occupato dal fluido all'istante di tempo $t$ è definito da:

$$\hat{\Omega}(t) = \{(x, y, z) \,:\, (x, y) \in \Omega,\, z \in (0, \eta(x, y, t))\}$$

Sia inoltre $\Gamma(t)$ il bordo del dominio tridimensionale $\hat{\Omega}(t)$. Per semplicità immaginiamo di partizionare il contorno del canale in tre regioni:

- il *fondo del canale*, invariante nel tempo, è indicato da

$$\Gamma_b = \{(x, y, 0) \,:\, (x, y) \in \Omega\}$$

- la *superficie libera* descritta dalla relazione:

$$\Gamma_s(t) = \{(x, y, \eta(x, y, t)) \,:\, (x, y) \in \Omega\}$$

- la *superficie laterale* di $\Omega$, identificabile con

$$\Gamma_l(t) = \{(x, y, z) \,:\, (x, y) \in \partial\Omega, z \in (0, \eta(x, y, t))\}$$

### 1.1.1 Modello semplificato

Il sistema (1.1) può essere semplificato con approssimazioni simili a quelle usate nella derivazione del modello *Shallow Water* con flussi a superficie libera. L'idea è quella di sfruttare il differente comportamento fisico nelle direzioni orizzontali e verticali, dovute alla diverse scale coinvolte. Iniziamo scomponendo la velocità $\mathbf{v}$ nelle sue componente orizzontale $\mathbf{u} = (u_x, u_y)$ e la sua componente verticale $w$. Ripetiamo la stessa operazione per la viscosità orizzontale $\mu$ e verticale $\nu$, mentre la pressione è scomposta nel suo termine idrostatico $p_h = p_a + g(\eta - z)$ e in una correzione idrodinamica $q$ tale che

$$p = g(\eta - z) + q \qquad \text{in } \hat{\Omega}(t) \tag{1.2}$$

avendo ipotizzato che la pressione atmosferica $p_a$ fosse nulla. Il sistema risultante diventa:

$$\begin{cases} \dfrac{D\mathbf{u}}{Dt} - \mathrm{div}_{xy}(\mu \nabla_{xy} \mathbf{u}) - \dfrac{\partial}{\partial z}\left(\nu \dfrac{\partial \mathbf{u}}{\partial z}\right) + \dfrac{g}{\rho}\mathrm{div}_{xy}(\eta) + \dfrac{1}{\rho}\mathrm{div}_{xy}(q) = 0 \\ \dfrac{Dw}{Dt} - \mathrm{div}_{xy}(\mu \nabla_{xy} w) - \dfrac{\partial}{\partial z}\left(\nu \dfrac{\partial w}{\partial z}\right) + \dfrac{1}{\rho}\dfrac{\partial p}{\partial z} + g = 0 \\ \mathrm{div}_{xy}\mathbf{u} + \dfrac{\partial w}{\partial z} = 0 \end{cases} \tag{1.3}$$

dove $g$ è l'accelerazione di gravità e $\frac{D}{Dt}$ è la derivata lagrangiana. Si osservi come, con il pedice $xy$ si è indicato che l'azione dell'operatore è ristretta alla sola direzione orizzontale. Inoltre è bene precisare che, per via della scelta del

modello, nel seguito assumeremo nulla la viscosità orizzontale, poichè trascurabile rispetto a quella verticale, che viene considerata costante. Le incognite del sistema (1.3) sono $\mathbf{u}$, $w$, $q$ e $\eta$: ci serve quindi un'ulteriore equazione, oltre alle condizioni iniziali e al bordo affinchè si possa chiudere il sistema. Nell'approssimazione delle *Shallow Water* possiamo considerare trascurabile lo spostamento orizzontale delle particelle del fluido sulla superficie libera $\Gamma_s(t)$ e con questa assunzione si ha:

$$\frac{D\eta}{Dt} = \frac{\partial \eta}{\partial t} + \mathbf{u}_s \cdot \nabla_{xy}\eta = w_s \tag{1.4}$$

cioè, la derivata materiale della particella del fluido sulla superficie $\Gamma_s(t)$ deve equiparare la velocità verticale. Con $\mathbf{u}_s$ e $w_s$ indichiamo le componenti della velocità, rispettivamente orizzontali e verticale, in corrispondenza della superficie libera. In realtà, per motivi di discretizzazione numerica, è più conveniente scrivere la condizione (1.4) in forma integrale, pervenendo alla relazione:

$$\frac{\partial \eta}{\partial t} + \mathrm{div}_{xy}\left(\int_0^\eta \mathbf{u} dz\right) = 0 \tag{1.5}$$

Aggiungendo l'equazione (1.5) al sistema (1.3), si ottiene il sistema detto *3D Multi Layer Shallow Water Equations* (3D-ML-SWE):

$$\begin{cases} \dfrac{D\mathbf{u}}{Dt} - \mathrm{div}_{xy}(\mu\nabla_{xy}\mathbf{u}) - \dfrac{\partial}{\partial z}\left(\nu\dfrac{\partial \mathbf{u}}{\partial z}\right) + \dfrac{g}{\rho}\mathrm{div}_{xy}(\eta) + \dfrac{1}{\rho}\mathrm{div}_{xy}(q) = 0 \\ \dfrac{Dw}{Dt} - \mathrm{div}_{xy}(\mu\nabla_{xy}w) - \dfrac{\partial}{\partial z}\left(\nu\dfrac{\partial w}{\partial z}\right) + \dfrac{1}{\rho}\dfrac{\partial p}{\partial z} + g = 0 \\ \mathrm{div}_{xy}\mathbf{u} + \dfrac{\partial w}{\partial z} = 0 \\ \dfrac{\partial \nu}{\partial t} + \mathrm{div}_{xy}\left(\displaystyle\int_0^\eta \mathbf{u} dz\right) = 0 \end{cases} \tag{1.6}$$

La discussione sulle condizioni iniziali e al bordo è rimandata alla sezione 1.1.3.

### 1.1.2 Vincolo di galleggiamento

Il modo più semplice per considerare l'interazione tra la superficie libera $\Gamma_s(t)$ e l'oggetto galleggiante, è imporre una condizione di incompenetrabilità sulla variabile $\eta$, attraverso moltiplicatori di Lagrange. Si assuma che la posizione dell'oggetto galleggiante sia un dato del problema, rappresentata attraverso la funzione $\phi(x, y, t)$. Di conseguenza il galleggiamento è rappresentato dalla disuguaglianza:

$$\eta(x, y, t) - \phi(x, y, t) \leq 0 \qquad \forall (x, y) \in \Omega \qquad \forall t \in [0, T] \tag{1.7}$$

L'equazione (1.7), nota come *vincolo di galleggiamento*, ha senso sotto l'assunzione che la geometria dello scafo (la parte del corpo immersa nell'acqua) sia abbastanza semplice cosicchè la superficie all'interfaccia possa essere descritto dal grafico di una funzione, cioè i punti dello scafo sono del tipo $(x, y, \phi(x, y))$. Dal punto di vista fisico, la presenza del corpo nell'acqua si comporta come un

campo di pressione esterno applicato sulla superficie. Il vincolo può perciò essere trattato introducendo una condizione dinamica nella regione dell'interfaccia:

$$p = p_a + \lambda, \qquad \text{su } \Gamma_s(t) \tag{1.8}$$

dove $\lambda$ è la normale al campo di forze applicata dallo scafo al fluido. La pressione all'interno del fluido può essere quindi riscritta come:

$$p(\mathbf{x}, z, t) = \lambda(\mathbf{x}, t) + g(\eta(\mathbf{x}, t) - z) + q(\mathbf{x}, z, t) \tag{1.9}$$

Nel contesto di imbarcazioni galleggianti, o più in generale di oggetti che si muovono durante la simulazione, è spesso conveniente cambiare il sistema di riferimento per evitare inutili oneri computazionali. Operiamo quindi un cambio di sistema di riferimento prendendone uno solidale con l'imbarcazione. Se rinominiamo il sistema di coordinate precedente come $(\overline{x}, \overline{y}, \overline{z})$, si ha

$$\begin{aligned} x &= \overline{x} - \overline{X}(t) \\ y &= \overline{y} \\ z &= \overline{z} \end{aligned} \tag{1.10}$$

dove $\overline{X}(t)$ è la traiettoria del baricentro della barca nel sistema di coordinate $(\overline{x}, \overline{y}, \overline{z})$. In modo analogo, nelle nuove coordinate, la velocità sarà espressa come:

$$\begin{aligned} \mathbf{u}(\mathbf{x}, z) &= \overline{\mathbf{u}}(\overline{\mathbf{x}}, \overline{z}) - \dot{\overline{X}}(t)\mathbf{e}_x, \\ w(\mathbf{x}, z) &= \overline{w}(\overline{\mathbf{x}}, \overline{z}), \end{aligned}$$

dove $\mathbf{e}_x = (1, 0)^T$, cioè il versore relativo all'asse $x$. Ricordiamo che la pressione è galileana invariante e, pertanto, lo sono anche le seguenti grandezze ad essa associate:

$$\begin{aligned} \eta(\mathbf{x}) &= \overline{\eta}(\overline{\mathbf{x}}), \\ \lambda(\mathbf{x}) &= \overline{\lambda}(\overline{\mathbf{x}}), \\ q(\mathbf{x}, z) &= \overline{q}(\overline{\mathbf{x}}, \overline{z}). \end{aligned}$$

Il sistema finale di equazioni di *Shallow Water*, con *vincolo di galleggiamento* e sistema di riferimento solidale con l'imbarcazione, diventa:

$$\begin{cases} \dfrac{D\mathbf{u}}{Dt} - \dfrac{\partial}{\partial z}\left(\nu \dfrac{\partial \mathbf{u}}{\partial z}\right) + g\,\mathrm{div}_{xy}\,\eta + \boxed{\nabla_{xy}\lambda} + \mathrm{div}_{xy}\,q + \boxed{\ddot{\overline{X}}(t)\mathbf{e}_x} = 0 \\ \dfrac{Dw}{Dt} - \dfrac{\partial}{\partial z}\left(\nu \dfrac{\partial w}{\partial z}\right) + \dfrac{\partial q}{\partial z} = 0 \\ \mathrm{div}_{xy}\,\mathbf{u} + \dfrac{\partial w}{\partial z} = 0 \\ \boxed{\lambda(\eta - \Psi) = 0, \qquad \lambda \geq 0, \qquad \eta \leq \Psi} \end{cases} \tag{1.11}$$

con condizioni iniziali:

$$\begin{aligned} \mathbf{u}(x, y, z, 0) &= \mathbf{u}_0(x, y, z) && \forall (x, y, z) \in \hat{\Omega}(0) \\ w(x, y, z, 0) &= w_0(x, y, z) && \forall (x, y, z) \in \hat{\Omega}(0) \\ q(x, y, z, 0) &= q_0(x, y, z) && \forall (x, y, z) \in \hat{\Omega}(0) \\ \eta(x, y, 0) &= \eta_0(x, y) && \forall (x, y) \in \Omega \end{aligned} \tag{1.12}$$

Il sistema di equazioni nel nuovo sistema di riferimento non varia eccetto nel bilancio del momento orizzontale. In quest'equazione si ha infatti il contributo della forza apparente nel sistema non inerziale che è dovuta all'accelerazione longitudinale della barca.

### 1.1.3 Condizioni al bordo

Sulla superficie libera $\Gamma_s(t)$ e sulla superficie del fondale $\Gamma_b$ sono applicate delle classiche condizioni di Neumann, che impongono la continuità dello stress attraverso le due superfici, con condizioni di equilibrio dinamico. In particolare si ha

$$\begin{aligned} \nu\frac{\partial \mathbf{u}}{\partial z} &= f_a(\mathbf{W}) = K|\mathbf{W}|\mathbf{W} \\ q &= p_a = 0 \end{aligned} \tag{1.13}$$

dove $f_a(\mathbf{W})$ è la forza esercitata sulla superficie dal vento, e $p_a$ è la pressione atmosferica di riferimento (che è posta a zero). Invece, sul fondo del canale, la componente verticale del campo di velocità deve necessariamente essere nulla ($w = 0$) mentre, per quanto riguarda quella orizzontale, viene simulato uno strato limite grazie alla condizione:

$$\nu\frac{\partial \mathbf{u}}{\partial z} = \frac{g|\mathbf{u}|}{C_D^2}\mathbf{u} \qquad \text{su } \Gamma_b(t)$$

dove $C_D$ è il coefficiente di Chezy, che permette di includere nel modello l'effetto dell'attrito con il fondale. Sulla superficie laterale del dominio, è possibile assegnare diverse condizioni:

- flusso nullo, che rispecchia l'impenetrabilità delle eventuale coste presenti;

$$\mathbf{u} \cdot \mathbf{n} = 0 \tag{1.14}$$

- condizione di *Dirichlet* sul campo di velocità per simulare per esempio una velocità di inflow e outflow di un canale;

$$(\mathbf{u}, w) = (\mathbf{u}_d, w_d) \tag{1.15}$$

- condizione di *Dirichlet* sul campo di elevazione $\eta$ per simulare ad esempio delle onde entranti;

$$\eta = \eta_d \tag{1.16}$$

- condizione *non riflettente* o *radiativa* sulla variabile $\eta$ per simulare il troncamento del dominio infinito. É disegnata per minimizzare la riflessione delle onde incidenti ed è basato sulle condizioni al bordo di *Sommerfeld* per problemi elettromagnetici.

$$\frac{\partial \eta}{\partial t} + (\dot{\overline{X}} + \sqrt{g(\eta + h)n}) \cdot \nabla\eta = 0 \tag{1.17}$$

  Si tratta di una relazione approssimata, ma accettabile se il bordo laterale del dominio è sufficientemente distante dalla imbarcazione.

### 1.1.4 Discretizzazione temporale

Consideriamo una discretizzazione dell'intervallo $[0, T]$ in sottointervalli $[t^n, t^{n+1}]$ con $\Delta t = t^{n+1} - t^n$ per $n = 0, \ldots, N-1$, con il seguente schema di avanzamento semi-implicito: per ogni $n \geq 0$, cerchiamo $(\mathbf{u}^{n+1}, w^{n+1}, q^{n+1}, \eta^{n+1}\lambda^{n+1})$ tale che:

$$\begin{cases} \dfrac{\mathbf{u}^{n+1} - \mathbf{u}^n(\mathbf{X})}{\Delta t} - \dfrac{\partial}{\partial z}\left(\nu \dfrac{\partial \mathbf{u}^{n+1}}{\partial z}\right) + g\nabla_{xy}\eta^{n+1} + \nabla_{xy}q^{n+1} + \nabla_{xy}\lambda^{n+1} = 0 \\ \dfrac{\eta^{n+1} - \eta^n}{\Delta t} + \operatorname{div}_{xy}\left(\displaystyle\int_0^{\eta^n} \mathbf{u}^{n+1} dz\right) = 0 \\ \operatorname{div}_{xy}\mathbf{u}^{n+1} + \dfrac{\partial w^{n+1}}{\partial z} = 0 \\ \dfrac{w^{n+1} - w^n(\mathbf{X})}{\Delta t} - \dfrac{\partial}{\partial z}\left(\nu \dfrac{\partial w^{n+1}}{\partial z}\right) + \dfrac{\partial q^{n+1}}{\partial z} = 0 \\ \lambda^{n+1}(\eta^{n+1} - \phi^{n+1}) = 0 \quad \lambda \geq 0, \quad \eta^{n+1} - \phi^{n+1} \leq 0 \end{cases} \tag{1.18}$$

dove le quantità al tempo $t^n$ sono note e $\mathbf{X}(s; (\mathbf{x}, z), t)$ è la soluzione del seguente problema differenziale:

$$\begin{cases} \dfrac{d\mathbf{X}(s; (\mathbf{x}, z), t)}{ds} = \mathbf{v}(\mathbf{X}(s; (\mathbf{x}, z), t), s), \quad \text{per } s \in (0, t) \\ \mathbf{X}(t, (\mathbf{x}, z), t) = (\mathbf{x}, z) \end{cases}$$

Da un punto di vista geometrico $\mathbf{X}(\cdot) = \mathbf{X}(\cdot; (\mathbf{x}, z), t)$ è la rappresentazione parametrica delle linee di flusso del campo di velocità, dunque $\mathbf{X}(s; (\mathbf{x}, z), t)$ è la posizione al tempo $s$ della particella, sottoposta al campo di velocità $\mathbf{v}$, che al tempo $t > s$ occupa la posizione $(\mathbf{x}, z)$. Considerando la discretizzazione dell'intervallo temporale precedentemente introdotta, è possibile discretizzare la derivata totale nel modo seguente:

$$\frac{D\mathbf{v}}{Dt}((\mathbf{x}, z), t^{n+1}) \simeq \frac{\mathbf{v}((\mathbf{x}, z), t^{n+1}) - \mathbf{v}(\mathbf{X}(t^n; (\mathbf{x}, z), t^{n+1}), t^n)}{\Delta t}$$

Per maggiori dettagli si veda [Alt09].

### 1.1.5 Formulazione debole

Per derivare la formulazione debole del problema in oggetto è necessario introdurre alcuni spazi funzionali che risulteranno utili nel seguito. Risulterà indispensabile avvalersi dello spazio delle funzioni a quadrato sommabile sul dominio bidimensionale

$$L^2(\Omega) = \left\{ \phi \, : \, \int_\Omega \phi^2 d\Omega < \infty \right\}$$

oltre allo spazio di Sobolev delle funzioni a quadrato sommabile, dotate di gradiente anch'esso quadrato sommabile:

$$H^1(\Omega) = \{\phi \in L^2(\Omega) \,:\, \nabla\phi \in L^2(\Omega)\}$$

In aggiunta alle due ambientazioni classiche appena introdotte, sarà necessario utilizzare anche lo spazio di funzioni vettoriali

$$H_{0,\Gamma}(div_{xy};\Omega) = \{\tau \,:\, \tau \in [L^2(\Omega)]^2,\, \mathrm{div}_{xy}\tau \in L^2(\Omega),\, \tau\cdot\mathbf{n} = 0 \text{su } \Gamma_b \cup \Gamma_s\}$$

dove, come descritto in precedenza, $\Gamma_b$ e $\Gamma_s$ sono i bordi del dominio in corrispondenza del fondale e della superficie libera. Per ogni passo temporale $t^{n+1}$ cerchiamo una soluzione $\mathbf{v} = (\mathbf{u}, w)^T \in \mathcal{U} = \mathcal{U}_u \times \mathcal{U}_w$ dove

$$\begin{aligned} \mathcal{U}_u &= H_{0,\Gamma}(div_{xy};\Omega) \times H^1([-h, \eta^n(x,y)]), \\ \mathcal{U}_w &= L^2(\Omega) \times H^1([0, \eta^n(x,y)]) \end{aligned} \tag{1.19}$$

Inoltre cerchiamo la pressione idrodinamica $q$, l'elevazione $\eta$ e il moltiplicatore $\lambda$ in $L^2(\Omega)$. Possiamo allora riscrivere il problema in forma debole: Trovare $\mathbf{v} \in \mathcal{U}$, e $\eta, \lambda, q \in L^2(\Omega)$ tale che

$$\begin{cases} a(\mathbf{v},\boldsymbol{\theta}) + gb(\boldsymbol{\theta},\eta) + b(\boldsymbol{\theta},\lambda) + d(\boldsymbol{\theta},q) = F(\boldsymbol{\theta}) & \forall \boldsymbol{\theta} \in \mathcal{U} \\ d(\mathbf{v},r) = 0 & \forall r \in L^2(\Omega) \\ b(\mathbf{v},t) \le G(r) & \forall r \in L^2(\Omega) \\ (\eta, r)_\Omega = (\eta^n, r) + \Delta t b(\mathbf{v}, r) & \forall r \in L^2(\Omega) \\ \text{con } \lambda \ge 0 \text{ e tale che } (\lambda, \eta - \phi) = 0 \end{cases} \tag{1.20}$$

dove $\boldsymbol{\theta} = (\boldsymbol{\Phi}, \gamma) \in \mathcal{U}$ e le forme bilineari $a(\cdot,\cdot)$, $b(\cdot,\cdot)$ e $d(\cdot,\cdot)$ sono così definite:

$$\begin{aligned} a(\boldsymbol{\psi},\boldsymbol{\theta}) &= \int_\Omega \int_{-h}^{\eta^n(x,y)} \left( \frac{1}{\Delta t}\boldsymbol{\psi}\cdot\boldsymbol{\theta} + \nu \frac{\partial \boldsymbol{\psi}}{\partial z}\cdot\frac{\partial \boldsymbol{\theta}}{\partial z}\right) d\Omega dz \\ b(\boldsymbol{\psi},\chi) &= -\int_\Omega \chi \left( \nabla\cdot \int_h^{\eta^n(x,y)} \boldsymbol{\varphi} dz \right) d\Omega \\ d(\boldsymbol{\psi}, q) &= -\int_\Omega \int_{-h}^{\eta^n(x,y)} q\nabla\cdot\boldsymbol{\psi} d\Omega dz \end{aligned} \tag{1.21}$$

dove si è posto $\boldsymbol{\psi} = (\boldsymbol{\varphi}, \upsilon) \in \mathcal{U}$ e $q, \chi \in L^2(\Omega)$. I termini noti $F$ e $G$ derivano dai termini relativi alle condizioni al bordo sulla superficie libera e sul fondale e dai contributi della precedente iterazione temporale

$$\begin{aligned} F(\boldsymbol{\theta}) &= \int_\Omega \left( K|\mathbf{W}|\mathbf{W}\cdot\boldsymbol{\theta} - g\frac{|\mathbf{u}^n|\mathbf{u}^n}{C_D^2}\cdot\boldsymbol{\Phi}\right) d\Omega + \frac{1}{\Delta t}\int_\Omega \int_{-h}^{\eta^n(x,y)} \mathbf{v}^n\cdot\boldsymbol{\theta} d\Omega dz \\ G(\chi) &= \left( \frac{\phi - \eta^n}{\Delta t}, \chi \right)_\Omega \end{aligned} \tag{1.22}$$

Per approfondire le tecniche matematiche che consentono di ottenere la formulazione debole di un problema variazionale e per i passaggi che portano alla formulazione debole si rimanda a [Sal08] e [LT11].

### 1.1.6 Il metodo di Uzawa

La formulazione debole (1.20) può essere interpretata come un problema di ottimizzazione vincolato

$$\begin{aligned} &\text{Trovare } \mathbf{v}^* : \mathcal{J} = \min_{\mathbf{v}\in\mathcal{U}} \mathcal{J}(\mathbf{v}) \\ &\text{s.t. } \nabla \cdot \mathbf{v}^* = 0 \\ &\text{e } \frac{\eta - \phi}{\Delta t} \leq 0 \end{aligned} \tag{1.23}$$

dove

$$\mathcal{J}(\mathbf{v}) = \frac{1}{2} a(\mathbf{v}, \mathbf{v}) - F(\mathbf{v}) \tag{1.24}$$

Associato a (1.23), detto *problema primale*, possiamo definire il funzionale Lagrangiano:

$$\mathcal{L}(\mathbf{v}, p, \lambda) = \mathcal{J}(\mathbf{v}) + d(\mathbf{v}, p) + \int_\Omega \left( \frac{\eta(\mathbf{v} - \phi)}{\Delta t} \right) d\Omega \tag{1.25}$$

Si può dimostrare che la soluzione di (1.23) corrisponde a un punto di sella di $\mathcal{L}$. L'esistenza del punto di sella, dimostrata in [LT11], ci permette di scrivere il problema primale nel modo equivalente

$$\text{trovare } (\mathbf{v}^*, p^*, \lambda^*) : \mathcal{L}(\mathbf{v}^*, p^*, \lambda^*) = \inf_{\mathbf{v}\in\mathcal{U}} \sup_{p\in L^2(\Omega)} \sup_{\lambda\in\Lambda} \mathcal{L}(\mathbf{v}^*, p^*, \lambda^*) \tag{1.26}$$

Osserviamo che potrebbe esistere una soluzione di (1.26) persino quando non esiste un punto di sella di$\mathcal{L}$, ma se $(\mathbf{v}^*, p^*, \lambda^*)$ è un punto di sella, allora per definizione $(\mathbf{v}^*, p^*, \lambda^*)$ è anche soluzione di (1.26), e segue che possiamo cambiare l'ordine degli operatori sup e min, perciò

$$\inf_{\mathbf{v}\in\mathcal{U}} \sup_{p\in L^2(\Omega)} \sup_{\lambda\in\Lambda} \mathcal{L}(\mathbf{v}^*, p^*, \lambda^*) = \mathcal{L}(\mathbf{v}^*, p^*, \lambda^*) = \sup_{\lambda\in\Lambda} \sup_{p\in L^2(\Omega)} \inf_{\mathbf{v}\in\mathcal{U}} \mathcal{L}(\mathbf{v}^*, p^*, \lambda^*) \tag{1.27}$$

Guardando al lato destro di (1.27) è naturale definire la *funzione duale* come

$$\mathcal{D}(\lambda) = \sup_{p\in L^2(\Omega)} \inf_{\mathbf{v}\in\mathcal{U}} \mathcal{L}(\mathbf{v}, p, \lambda) \tag{1.28}$$

e il *problema duale* come

$$\text{Trovare } \lambda^* : \mathcal{D}(\lambda^*) = \sup_{\lambda\in\Lambda} \mathcal{D}(\lambda) \tag{1.29}$$

La funzione duale $\mathcal{D}$ associa ad ogni profilo di pressione sul corpo $\lambda$, la soluzione del corrispondente problema di Navier-Stokes a superficie libera. Per risolvere il problema utiliziamo l'*algoritmo di Uzawa* che è una metodo iterativo che permette di costruire una successione $(\mathbf{v}_{(k)})$ che converge all'unica soluzione del problema primale (1.23). La procedura iterativa, assegnate due tolleranze $\epsilon_1 > 0$ e $\epsilon_2 > 0$ e $\lambda_0$s, può essere riassunto nei seguenti passi:

1. calcolare $\eta_{(k)} \in L^2(\Omega)$, $\mathbf{v}_{(k)} \in \mathcal{U}$ e $q \in L^2(\Omega)$ tale che

$$\begin{aligned} &a(\mathbf{v}_{(k)}, \boldsymbol{\theta}) + b(\boldsymbol{\theta}, \lambda_{(k)}) + gb(\boldsymbol{\theta}, \eta_{(k)}) + d(\boldsymbol{\theta}, q_{(k)}) = F(\boldsymbol{\theta}), && \forall \boldsymbol{\theta} \in \mathcal{U} \\ &d(\mathbf{v}, r) = 0, && \forall r \in L^2(\Omega) \\ &(\eta_{(k)}, r)_\Omega = (\eta^n, r)_\Omega - \Delta t b(\mathbf{v}_{(k)}, r), && \forall r \in L^2(\Omega) \end{aligned} \tag{1.30}$$

2. fissiamo il valore $\alpha_{(k)}$ e aggiorniamo $\lambda$ secondo la formula

$$\lambda_{(k+1)} = \max\left(\lambda_{(k)} + \alpha_{(k)} \frac{\eta_{(k)} - \phi}{\Delta t}, 0\right) \tag{1.31}$$

Ripetiamo i passi fino al raggiungimento di un criterio di convergenza, che si basa sul soddisfacimento delle classiche condizione di Karush-Khun-Tucker

$$\begin{aligned} &|(\lambda_{(k)}, \eta_{(k)} - \phi)_\Omega| \leq \epsilon_1 \\ &\phi - \eta_{(k)} > -\epsilon_2 \end{aligned} \tag{1.32}$$

Si verifica sperimentalmente che il parametro di convergenza ottimale $\alpha_{(k)}$ è proprio l'accelerazione di gravità $g$, e ciò rispecchia l'interpretazione fisica del moltiplicatore di Lagrange, che rappresenta il termine di pressione dovuto alla presenza dello scafo. Nelle simulazioni si è poi scelto come parametri di convergenza rispettivamente i valori $10^{-6}$ e $10^{-9}$.

### 1.1.7 Discretizzazione spaziale

Il dominio fisico tridimensionale $\hat{\Omega}$ è immerso nel parallelepipedo composto di $N$ strati di base $\Omega$. Lo spessore del lato $k$ è indicato con $\delta z_k$. Per discretizzare il dominio $\Omega$ si considera una mesh triangolare non strutturata. La mesh è posizionata nei punti medi di ogni lato ed è replicata per ogni strato. La velocità orizzontale è approssimata combinando gli elementi finiti di ordine più basso di Raviart-Thomas $RT_0$ nel piano $xy$, con gli elementi finiti $\mathcal{P}^1$ lungo la direzione verticale

$$\mathbf{u}_h^*(\mathbf{x}, z) = \sum_{k=1}^{\mathcal{K}} \sum_{l=1}^{N_{ed}} J_{lk} \tau_l(\mathbf{x}) \phi_k(z), \qquad \text{con } \tau_l \in \mathbb{Q}_h \text{ e } \phi_k \in W_{\delta z}$$

dove $N_{ed}$ è il numero totale di lati presenti nella triangolazione $\mathcal{T}_h$. I gradi di libertà associati alla velocità orizzontale sono collocati nei punti medi dei lati del triangolo posto a metà altezza dell'elemento prismatico. Ciò significa che, per definizione, $J_{lk}$ è il flusso normale che attraversa il lato $\mathbf{e}_l$, appartenente alla griglia $\mathcal{T}_h$, posta in corrispondenza del punto medio dello strato $k$-esimo:

$$J_{lk} = \int_{\mathbf{e}_l} \mathbf{u}_k \cdot \mathbf{n} d\sigma \qquad \text{con } l = 1, \ldots, N_{ed}$$

Per quanto riguarda la componente verticale del campo di velocità, essa sarà approssimata combinando elementi finiti costanti a tratti nel piano $xy$ con

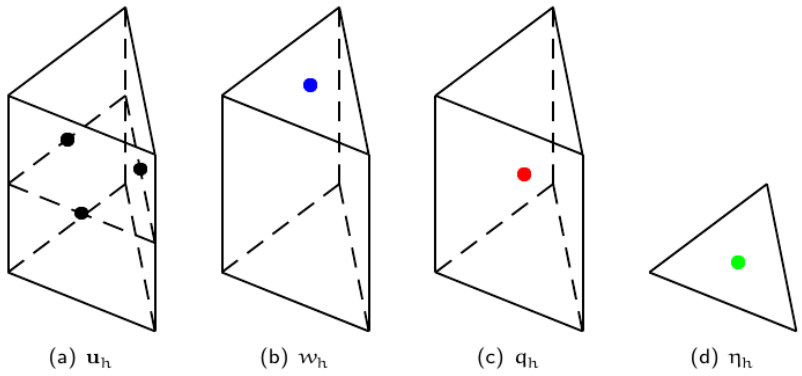



**Figura 1.1:** Rappresentazione dei gradi di libertà per le variabili discretizzate. In nero sono riportati i nodi per la componente orizzontale del campo di velocità e in blu quello per la componente verticale, in rosso il nodo per la pressione idrodinamica e in verde quello per l'elevazione della superficie libera.

elementi lineari lungo la direzione verticale:

$$w_h^*(\mathbf{x}, z) = \sum_{k=1}^{\mathcal{K}} \sum_{i=1}^{N_{el}} \omega_{ik} \psi_i(\mathbf{x}) \phi_k(z), \qquad \text{con } \psi_i \in \mathcal{U}_h \text{ e } \phi_k \in \mathcal{W}_{\delta z}$$

dove $N_{el}$ è il numero totale di elementi presenti nella triangolazione $\mathcal{T}_h$ . Qui è necessario porre attenzione poichè, a differenza di quanto accadeva in precedenza, i gradi di libertà per la velocità verticale sono collocati nei circocentri dei due triangoli di base dell'elemento prismatico. Questo significa che la griglia è sfalsata rispetto a quella per la componente orizzontale del campo di velocità. Dunque, nonostante per ora siano state identificate con lo stesso simbolo, in realtà le funzioni di base $\phi_k$, nei due casi, saranno collocate in nodi differenti. Pertanto i coefficienti $\omega_{ik}$ rappresenteranno la velocità verticale delle particelle contenute nell'$i$-esimo triangolo della griglia posta in corrispondenza della faccia superiore del $k$-esimo strato. É bene sottolineare che, in questo modo, la condizione al contorno omogenea sul fondale verrà imposta in maniera automatica, semplicemente grazie alla scelta opportuna dello spazio a elementi finiti. Approssimiamo invece l'elevazione della superficie libera con funzioni costanti a tratti su ogni singolo elemento:

$$\eta_h^*(\mathbf{x}) = \sum_{i=1}^{N_e l} \xi_i \psi_i(\mathbf{x}), \qquad \text{con } \psi_i \in \mathcal{U}_h.$$

In questo caso, dato che l'elevazione non dipende dalla quota $z$, $\xi_i$ non è nient'altro che il valore di $\eta$ in corrispondenza dell'$i$-esimo elemento della triangolazione $\mathcal{T}_h$.

In 1.1 sono rappresentati i gradi di libertà per le diverse variabili, in corrispondenza delle rispettive posizioni sui prismi del dominio discretizzato. Per discretizzare la componente idrodinamica della pressione si sceglie un sotto-spazio

di funzioni costanti su ciascun prisma della griglia tridimensionale:

$$X_p = \left\{ \chi \in L^2(\hat{\Omega}) \,:\, \chi|_P \in \mathbb{P}_0(P), \forall P \right\}.$$

In questo modo la pressione, rappresentata nel sotto-spazio finito dimensionale appena definito, assume la forma

$$q_h^{n+1}(\mathbf{x}, z) = \sum_{p=1}^{N_P} \zeta_p \xi_p(\mathbf{x}, z), \qquad \text{con } \zeta_p \in X_p.$$

dove $N_P$ è il numero totale di prismi contenuti nel dominio $\hat{\Omega}$ al passo temporale attuale. Ciò significa che, per via della scelta delle funzioni di forma, stiamo assegnando un unico valore di pressione a ciascun prisma $p = 1, \ldots, N_P$ . Il fatto di scegliere delle funzioni di base costanti, chiaramente, non permette di discretizzare la pressione idrodinamica in modo molto accurato. Per quanto riguarda il moltiplicatore di Lagrange $\lambda$ associato al vincolo di galleggiamento, dato che si tratta di un contributo in pressione, la scelta deve essere coerente con quanto fatto in precedenza. Per questo motivo si seleziona una base costante a tratti sugli elementi della triangolazione $\mathcal{T}_h$:

$$\lambda_h^{n+1}(\mathbf{x}) = \sum_{l=1}^{N_{el}} \mu_i \psi_i(\mathbf{x}), \qquad \text{con } \psi_i \in \mathcal{U}_h$$

Ciò significa che le incognite dell'algoritmo di Uzawa saranno i valori dell'elevazione $\xi_i$ e quelli del moltiplicatore $\mu_i$, entrambi collocati nei circocentri dei triangoli $T_i \in \mathcal{T}_h$.

### 1.1.8 Il Sistema Algebrico

Combinando la discretizzazione in tempo e in spazio appena presentata, possiamo riscrivere il sistema (1.11) ad ogni passo temporale $t^{n+1}$ come il seguente sistema algebrico:

$$\begin{bmatrix} \frac{1}{\Delta t} M_u + k_u & g D_{xy}^T & 0 & D_{xy} \\ E & \frac{1}{\Delta t} M_\nu & 0 & 0 \\ 0 & 0 & \frac{1}{\Delta t} M_\Omega + k_w & D_z^T \\ D_{xy} & 0 & D_z & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}^{n+1} \\ \eta^{n+1} \\ w^{n+1} \\ q^{n+1} \end{bmatrix} = \frac{1}{\Delta t} \begin{bmatrix} M_u \mathbf{u}^n + \mathbf{F}_{u,(k)} - g D_{xy}^T \lambda_k \\ M_\eta \eta^n + \mathbf{F}_{\eta,(k)} \\ M_w w^n + \mathbf{F}_{w,(k)} \\ 0 \end{bmatrix} \tag{1.33}$$

dove con il termine $\mathbf{F}_{(k)} = (\mathbf{F}_{u,(k)}, \mathbf{F}_{\eta,(k)}, \mathbf{F}_{w,(k)})$ si è tenuto conto delle condizioni al bordo. Poniamo

$$A = \begin{bmatrix} \frac{1}{\Delta t} M_u + k_u & g D_{xy}^T & 0 \\ E & \frac{1}{\Delta t} M_\nu & 0 \\ 0 & 0 & \frac{1}{\Delta t} M_\Omega + k_w \end{bmatrix} \tag{1.34}$$

e

$$D = \begin{bmatrix} D_{xy} & 0 & D_z \end{bmatrix} \tag{1.35}$$

dove abbiamo indicato con $M_u$, $M_\eta$ e $M_w$ le matrice di massa; $k_u$, $k_w$ e $k_\eta$ le matrici di stiffness verticale, e $D$ la controparte algebrica dell'operatore completo di divergenza in tre dimensioni. Segue quindi che $D_{xy}$ indica l'operatore

divergenza limitato sul piano $xy$ mentre $D_z$ è l'operatore derivata nella direzione verticale. Infine $E$ tiene conto del termine integrale della divergenza nell'equazione dell'elevazione. Se con il termine $\hat{\mathbf{F}}_{(k)}$ indichiamo il termine noto derivante dalle condizioni al bordo, dai termini espliciti delle derivate temporali e dal termine del moltiplicatore di Lagrange, possiamo riscrivere il sistema nel seguente modo:

$$\begin{bmatrix} A & D^T \\ D & 0 \end{bmatrix} \begin{bmatrix} \mathbf{V}^{n+1} \\ q^{n+1} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{F}}_{(k)} \\ 0 \end{bmatrix} \tag{1.36}$$

dove $\mathbf{V}^{n+1} = (\mathbf{u}^{n+1}, \eta^{n+1}, w^{n+1})$.

Risolvere il sistema (1.36) a ogni iterazione del metodo di Uzawa può essere piuttosto costoso per i problemi affrontati, dove il numero dei gradi di libertà è elevato. Presentiamo ora la ben nota scomposizione di *Chorin-Temam* per i problemi di Navier-Stokes, che permette un efficace implementazione delle iterazioni del metodo di Uzawa. La scelta di questa scomposizione nasce dall'osservazione che la regola di aggiornamento (**??**) dipende solo da $\eta$, quindi tutte le altre variabili calcolate ad ogni iterazione di Uzawa non sono necessarie. Per questo motivo, scindiamo la soluzione dell'equazioni di Navier-Stokes in un passo idrostatico e una correzione idrodinamica. Nel passo idrostatico calcoliamo l'elevazione $\eta$ e un'approssimazione della velocità, mentre nella correzione idrodinamica calcolando la pressione idrodinamica $q$ e modificando il valore di $\mathbf{v}$, mentre nessuna modifica è necessaria per l'elevazione $\eta$ calcolata al passo precedente. Perciò, il passo idrostatico è l'unico richiesto per le iterazioni di Uzawa, con grossi risparmi computazionali. La procedura è descritta dal seguente algoritmo:

**Step idrostatico** Calcoliamo $\tilde{\mathbf{u}}_{(k)}$, $\tilde{\eta}_{(k)}$ e $\lambda_k$ tale che

$$\begin{cases} \left(\frac{1}{\Delta t} M_u + k_u\right) \tilde{\mathbf{u}}_{(k)} + g D_{xy}^T \tilde{\eta}_{(k)} = \frac{1}{\Delta t} M_u \mathbf{u}^n(X) - g D_{xy}^T \lambda_k \\ -E\tilde{\mathbf{u}}_{(k)} + \frac{1}{\Delta t} M_\eta \tilde{\eta}_{(k)} = \frac{1}{\Delta t} M_\eta \eta^n \end{cases} \tag{1.37}$$

e aggiorniamo $\lambda_k$ secondo la formula (**??**) fino a che non sia soddisfatto il criterio di convergenza (1.32).

**Valutazione componente velocità verticale intermedia** Calcoliamo $\tilde{w}$ risolvendo il sistema:

$$\left(\frac{1}{\Delta t} M_w + k_w\right) \tilde{w} = \frac{1}{\Delta t} M_w w^n(\mathbf{X}) \tag{1.38}$$

**Step idrodinamico** Calcoliamo il valore finale della pressione idrodinamica $q$ risolvendo il sistema:

$$-\Delta t (D_{xy} M_u^{-1} D_z^T + D_z M_w^{-1} D_z^T) \tilde{q}^{n+1} = -(D_{xy}\tilde{\mathbf{u}} + D_z \tilde{w}) \tag{1.39}$$

**Correzione idrodinamica sulle componenti della velocità** Calcoliamo il valore finale per le velocità $\mathbf{u}^{n+1}$, $w^{n+1}$ secondo le equazioni:

$$\begin{cases} \mathbf{u}^{n+1} = \tilde{U} - \Delta t M_u^{-1} D_{xy}^T q^{n+1} \\ \eta^{n+1} = \tilde{\eta} \\ w^{n+1} = \tilde{w} - \Delta t M_w^{-1} D_z^T q^{n+1} \end{cases} \tag{1.40}$$

### 1.1.9 La descrizione geometrica dello scafo

Come accennato in precedenza, la descrizione della superficie esterna dell'imbarcazione è effettuata sia per mezzo di funzioni analitiche, sia attraverso mesh triangolate. In particolare, qualunque sia la forma dello scafo, essa è sempre riferita a una terna ortogonale, con l'origine posizionata in corrispondenza del centro di massa della barca. Una delle problematiche del vincolo di galleggiamento sulla superficie libera è certamente la descrizione delle rotazioni che agiscono su di essa. In *Stratos++* si possono simulare due rotazioni: il moto di beccheggio e quello di rollio. Il beccheggio si riferisce all'oscillazione della barca attorno al suo asse trasversale, mentre con rollio si intende la rotazione intorno al suo asse longitudinale. Per poter applicare il vincolo ruotato in maniera corretta, sarà necessario applicare opportune trasformazioni che, come vedremo, comporteranno anche la risoluzione di problemi non lineari.

#### Configurazione rototraslata

Nel seguito della trattazione supporremmo di poter associare una funzione $f(x, y)$ anche per superficie triangolate, delle quali ovviamente non conosciamo la funzione che la ha generata. Questa scelta rispecchia in parte la filosofia anche usata all'interno del codice, dove la classe implementata si comporta come una funzione che date le coordinate $(x, y)$, va a restituire la coordinata $z$ dell'imbarcazione. La geometria dell'oggetto galleggiante va inoltre estesa opportunamente in modo che $\phi$ sia definita su tutto $\Omega$. Per imporre il vincolo di galleggiamento sulla superficie libera, dunque, si applica una rototraslazione alla funzione che descrive lo scafo, in modo da porla nella posizione corretta, a seconda delle diverse configurazioni che può assumere istante per istante. Per semplificare al massimo l'impianto algebrico, si definisce la superficie, nella configurazione di riferimento, per via parametrica:

$$\mathbf{f} = \begin{bmatrix} x \\ y \\ z(x,y) \end{bmatrix} \tag{1.41}$$

Ora, definendo con $(X, Y, Z)$ la posizione reale del baricentro della barca, si può riferire lo scafo a questo punto dello spazio tridimensionale per mezzo di una traslazione. Dal punto di vista algebrico questa trasformazione è bene rappresentata dalla somma vettoriale:

$$T\mathbf{f} = \mathbf{f} + \mathbf{t} = \begin{bmatrix} x \\ y \\ z(x,y) \end{bmatrix} + \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x+X \\ y+Y \\ z(x,y)+Z \end{bmatrix} \tag{1.42}$$

Definendo con $\theta$ l'angolo di beccheggio e con $\varphi$ l'angolo di rollio, si ricavano le due matrici che rappresentano le rotazioni in chiave algebrica:

$$B = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \qquad R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi) & \sin(\varphi) \\ 0 & -\sin(\varphi) & \cos(\varphi) \end{bmatrix}$$

dove $B$ è ovviamente relativa alla rotazione lungo l'asse trasversale, mentre $R$ a quella diretta lungo l'asse longitudinale della barca. La combinazione delle

rotazioni lungo i due assi, ortogonali e passanti per il centro di massa dello scafo, può essere analiticamente espressa dal prodotto delle matrici relative alle singole rotazioni. In questo modo si ottiene un'unica struttura algebrica che tiene conto, allo stesso tempo, di beccheggio e rollio:

$$BR = \begin{bmatrix} \cos(\theta) & \sin(\theta)\sin(\phi) & -\sin(\theta)\cos(\varphi) \\ 0 & \cos(\varphi) & \sin(\varphi) \\ \sin(\theta) & -\cos(\theta)\sin(\varphi) & \cos(\theta)\cos(\varphi) \end{bmatrix} \tag{1.43}$$

É possibile quindi combinare la (1.42) con la (1.43), per poi applicare la trasformazione risultante alla descrizione della superficie dello scafo nella configurazione di riferimento. Ne risulta la relazione:

$$TBR\mathbf{f} = \begin{bmatrix} \cos(\theta)x + \sin(\theta)\sin(\varphi)y - \sin(\theta)\cos(\varphi)z(x,y) + X \\ \cos(\varphi)y + \sin(\varphi)z(x,y) + Y \\ \sin(\theta)x - \cos(\theta)\sin(\varphi)y + \cos(\theta)\cos(\varphi)z(x,y) + Z \end{bmatrix} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} \tag{1.44}$$

che porta i punti $(x, y, z)$, relativi alla configurazione riferita al centro del sistema cartesiano, in corrispondenza dei punti $(\hat{x}, \hat{y}, \hat{z})$, che si trovano sulla configurazione rototraslata. Il problema che si presenta al momento di dover applicare il vincolo sulla superficie libera è il seguente. A partire dalla conoscenza delle coordinate $(\hat{x}, \hat{y})$, si vuole ricavare la quota $\hat{z}$ a cui si trova la chiglia della barca nella configurazione attuale. Di fatto per soddisfare questa richiesta è necessario ricavare la trasformazione inversa che, nel caso in cui la descrizione $z(x, y)$ dello scafo non sia elementare, richiede la risoluzione di un problema non lineare. Per ulteriori dettagli si rimanda a [Alt09].

### Geometria analitica

Le geometrie descritte tramite funzioni analitiche $z = z(x, y)$ implementate in *Stratos++* sono essenzialmente due: una forma ellissoidale e il Wigley-Hull. Tuttavia, come si vedrà nella presentazione del codice, integrarne di nuove risulta di semplice implementazione. La forma ellissoidale è descritta dalla seguente espressione analitica:

$$z(x, y) = -H\sqrt{1 - \frac{4x^2}{L^2} - \frac{4y^2}{W^2}} + G \tag{1.45}$$

dove $L$, $W$ e $H$ sono, rispettivamente, la lunghezza (*length*), la larghezza (*width*) e l'altezza (*height*) dello scafo. Inoltre, poichè sarà necessario descrivere le rotazioni attorno al centro di massa della barca, è bene che la descrizione analitica ponga il baricentro dell'imbarcazione in corrispondenza dell'origine degli assi. Per questo si indica con $G$ la distanza tra la coperta e il centro di massa, in modo da porre l'equazione (1.45) nel riferimento corretto.

Un'alternativa alla descrizione ellissoidale dello scafo è quella fornita dal *Wigley-Hull*. Questo tipo di geometria è decisamente più realistica della precedente, ed è stata ampiamente utilizzata in ingegneria per lo studio prestazionale di imbarcazioni da gara. La sua espressione analitica è la seguente:

$$z(x, y) = -H\sqrt{1 - \frac{\frac{2|y|}{W}}{1 - (\frac{2x}{L})^4}} + G$$

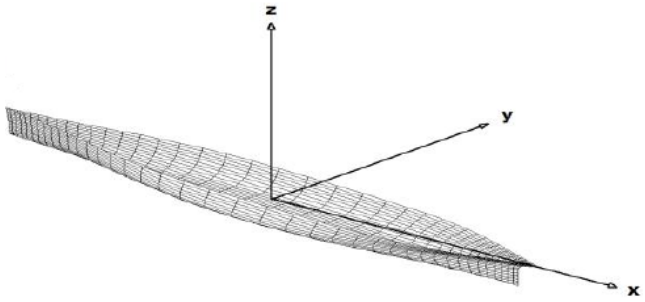



**Figura 1.2:** Imbarcazione nella configurazione di riferimento.

Anche in questo caso la terna $(L, W, H)$ si riferisce alle dimensioni caratteristiche della barca, mentre il valore $G$ alla distanza del baricentro dalla coperta.

### Geometria triangolata

Uno degli aspetti più limitanti del codice *Stratos++V.1.0* era sicuramente la impossibilità di interfacciarsi con geometrie non analitiche. Questa mancanza non permetteva di testare il codice con imbarcazioni reali e di confrontare i risultati delle simulazioni con dati sperimentali. Sono quindi stati integrati nel codice i metodi necessari per interfacciarsi con scafi triangolati. L'unica assunzione che è stata fatta riguardo allo scafo è che la sua superficie sia convessa, ipotesi che, come vedremo nel seguito, risulta fondamentale nella determinazione della coordinata $z$ dell'imbarcazione, e che comunque non si rivela eccessivamente vincolante, in quanto le imbarcazioni di canottaggio sono in generale caratterizzate da scafi convessi. L'utilizzo di geometrie non analitiche ha introdotto due problematiche:

- determinare la coordinata $z$ della superficie dello scafo date le coordinate $x$ e $y$;
- integrare il valore del moltiplicatore di Lagrange $\lambda$ dovuto al vincolo di galleggiamento sull'imbarcazione per ricavare la forza che agisce su di esso.

La seconda problematica verrà analizzata nella sezione successiva. Per risolvere invece il primo punto, possiamo osservare che dall'analisi fatta nella sezione precedente, è necessaria una procedura che date le coordinate $x$ e $y$ nella configurazione rototraslata, ci restituisca la coordinata $z$. In questa configurazione la barca è centrata rispetto al baricentro e con angoli di beccheggio e rollio nulli, come mostrato in 1.2. Quindi, sfruttando l'ipotesi di convessità della superficie, possiamo affermare che dato il punto $P = (x, y, 0)$ e la retta parallela all'asse $z$ e passante per il punto $P$, di equazione parametrica $r(t) = P + t\mathbf{d}$, esiste un unico punto di intersezione $\overline{P}$ tra la retta $r$ e la superficie $\mathcal{B}$ dell'imbarcazione. La coordinata $z$ di $\overline{P}$ sarà la quota dell'imbarcazione date le coordinate le coordinate $x$ e $y$ nella configurazione di riferimento. A questo punto, grazie alla relazione (1.44), possiamo ottenere le coordinate $(\hat{x}, \hat{y}, \hat{z})$ nella configurazione rototraslata. La risoluzione del problema di intersezione tra la retta $r$ e la superficie $\mathcal{B}_t$ dello scafo verrà presentata in 2.4, dove veranno introdotte tutte le

tecniche usate per minimizzare l'impatto di queste operazioni sulle performance del codice.

### 1.1.10 Il calcolo delle forze sull'imbarcazione

Ad ogni passo temporale dopo aver risolto numericamente la fluidodinamica attorno allo scafo, è necessario utilizzare le informazioni ricavate per l'interazione con la dinamica dell'imbarcazione. In particolare, si vogliono conoscere le forze idrodinamiche che agiscono sullo scafo, sotto l'ipotesi che sia un corpo rigido. Questo punto risulta di fondamentale importanza, anche in vista dell'accoppiamento fluido-struttura, in quanto saranno le forze fluidodinamiche le informazioni richieste dal risolutore della struttura. L'ipotesi di corpo rigido fa si che quello ci interessa conoscere non sia l'intera distribuzione delle forze, bensì solo la forza $\mathbf{F}$ e il momento $\mathbf{M}$ agenti sul centro di massa dell'imbarcazione. Queste grandezze possono essere calcolate a partire dal profilo di pressione che agisce sulla superficie di interfaccia tra l'acqua e la chiglia. Per come è stato costruito il modello fisico-matematico, tale profilo è rappresentato da $p_\Psi = \rho\lambda$.

Nel caso di geometrie analitiche i valori di $p_\Psi$ sui nodi della griglia venivano utilizzati direttamente per il calcolo delle forze e dei momenti. Quando invece si usano geometrie triangolate, questo non è più vero, in quanto i nodi della triangolazione che descrive la superficie dello scafo non corrispondono ai nodi del domino $\Omega_h$. É stato quindi neccessario introdurre un operatore *interpolante* $\Pi_h$ che data $p_\Psi$ su $\Omega_h$, mi restituisca una funzione $p^s_\Psi$ definita sullo $\mathcal{B}_h$. Nella sezione 2.5 presenteremo le possibili scelte per l'operatore $\Pi_h$ e la tecnica di interpolazione adottata. Per ottenere le forze sullo scafo, si è dovuto quindi integrare il profilo di pressione $p^s_\Psi$ sulla superficie bagnata $\Gamma_w$ dell'imbarcazione:

$$\begin{cases} \mathbf{F} = -\displaystyle\int_{\Gamma_w} p^s_\Psi(\mathbf{x})\mathbf{n}d\Gamma \\ \mathbf{M} = -\displaystyle\int_{\Gamma_w} p^s_\Psi(\mathbf{x})(\mathbf{x}-\mathbf{x}_{cm})\times\mathbf{n}d\Gamma \end{cases} \tag{1.46}$$

dove $\mathbf{x}_{cm}$ è la posizione del centro di massa della barca, mentre $\mathbf{n}$ è la normale uscente alla superficie esterna dello scafo.

La superficie bagnata $\Gamma_w$ dello scafo è identificata dagli elementi di $\mathcal{B}_h$ nei quali $p^s_\Psi$ è non nullo, cioè solo nella regione in cui agisce il vincolo. A questo punto, per mostrare in che modo viene integrato il profilo di pressione, si pone l'attenzione su uno specifico triangolo $T \in \mathcal{B}_h$, per il quale il valore del moltiplicatore discreto nel baricentro dell'elemento sia non nullo. Questo triangolo è completamente determinato dai suoi 3 vertici $A = (x_A, y_A, z_A)$, $B = (x_B, y_B, z_B)$ e $C = (x_C, y_C, z_C)$, a partire dei quali è possibile calcolare la normale:

$$\mathbf{n} = \vec{AB}\times\vec{AC} = det\begin{bmatrix} \mathbf{e}_x & \mathbf{e}_y & \mathbf{e}_z \\ \mathbf{x}_B-\mathbf{x}_A & \mathbf{y}_B-\mathbf{y}_A & \mathbf{z}_B-\mathbf{z}_A \\ \mathbf{x}_C-\mathbf{x}_A & \mathbf{y}_C-\mathbf{y}_A & \mathbf{z}_C-\mathbf{z}_A \end{bmatrix}$$

dove i vettori $\mathbf{e}_i$ non sono altro che i versori degli assi coordinati. Una volta nota la normale all'elemento, è immediato procedere all'integrazione del profilo in pressione, noti i valori del moltiplicatore di Lagrange in corrispondenza dei

vertici del triangolo. Poichè abbiamo scelto un'interpolante lineare del profilo, il valore dell'integrale sul triangolo $\hat{T}$ è ottenuto attraverso la formula di quadratura del trapezio sui triangoli. Ripetendo la procedura per tutti i triangoli attivi della griglia dello scafo, possiamo concludere che la forza agente sullo scafo può essere :

$$\mathbf{F} = -\sum_{i=1}^{N_w} \frac{1}{3}\left[\sum_{l=1}^{3} p^s_{\Psi,V_i^l}\right]|T_i|\mathbf{n}_i \tag{1.47}$$

dove la somma viene effettuata solo sugli $N_w$ elementi bagnati e $V_i^l$ indica l'$l$-esimo vertice del triangolo $T_i \in \mathcal{B}_h$, con $l = 1, \ldots, 3$.

Per finire, il calcolo del momento totale agente sullo scafo può essere effettuato ricavando i momenti che agiscono sui singoli triangoli, e poi sommando tutti i contributi. In pratica, si assume che la forza agisca nel baricentro del triangolo, in modo da calcolare il braccio come la distanza tra quel punto e il centro di massa dello scafo, punto intorno al quale vengono compiute le rotazioni. In conclusione, secondo la notazione introdotta, il calcolo del momento totale si può esprimere attraverso la relazione:

$$\mathbf{M} = -\sum_{i=1}^{N_w} \frac{1}{3}\left[\sum_{l=1}^{3} p^s_{\Psi,V_i^l}\right]|T_i|(\mathbf{x}_i - \mathbf{x}_{cm}) \times \mathbf{n}_i$$

dove con $\mathbf{x}_i$ si intendono le coordinate del baricentro del triangolo $T_i$.

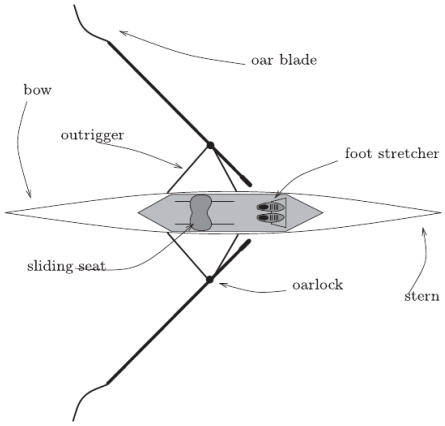



**Figura 1.3:** Scafo singolo con le componenti principali, immagine presentata in [FMMM09]

## 1.2 Risolutore del sistema imbarcazione-vogatori

Prima di presentare il modello associato alla struttura formata dall'insieme imbarcazione e vogatori, descriviamo brevemente le principali componenti dell'imbarcazione di canottaggio, mostrate in 1.3. In questo tipo di imbarcazioni, i vogatori sono seduti nella parte centrale dello scafo allungato, con la loro schiena che punta nella direzione di moto della canoa. I seggiolini scorrono su rotaie, mentre i piedi dei vogatori sono assicurate alla pedana. I remi sono fissati all'imbarcazione tramite gli scalmi (oarlock) montati su supporti laterali detti stabilizzatori. Nel modello implementato nel software *Kimè*, gli scalmi sono rappresentati come perfetti giunti sferici. Esistono differenti tipi di scafi, singoli , doppi ,a quattro posti e a otto posti, i quali possono essere con o senza timoniere. Nel modello implementato in *Kimè*, la presenza del timoniere può essere considerata aggiungendo una massa addizionale. Per semplicità, nel seguito considereremo la derivazione del modello per uno scafo senza timoniere.

La velocità istantanea del punto di una barca può essere scomposta in due componenti: una velocità media $\overline{\mathbf{v}}$ e dei moti secondari $\mathbf{u}$. La velocità media $\overline{\mathbf{v}}$ è la media della velocità orizzontale della imbarcazione in un periodo di vogata (cadenza). Si ha quindi

$$\mathbf{v}(\mathbf{x}, t) = \overline{\mathbf{v}}(\mathbf{X}, t) + \mathbf{u}(\mathbf{x}, t) \tag{1.48}$$

in ogni punto della barca e per ogni $t > 0$. I moti secondari sono quelli indotti dai movimenti dei vogatori e dall'azione dei remi, e sono assunti periodici con periodo uguale alla cadenza dell'azione dei vogatori. Se noi trascuriamo le fasi iniziali, possiamo assumere che $\overline{\mathbf{v}}$ sia praticamente costante.

Considereremo il moto nel piano simmetrico della barca e in 1.4 mostriamo i tre moti secondari considerati in questo lavoro. Il primo è un moto orizzontale lineare lungo l'asse longitudinale della barca. É principalmente originata dalla trazione intermittente e dallo spostamento orizzontale dei vogatori. Ci riferiremo a esso come *moto orizzontale secondario*. Un secondo moto, detto *moto di affondamento*, si ha lungo l'asse verticale, e può essere pensato come una fluttuazione intorno alla posizione di equilibrio idrodinamica. Esso è generato

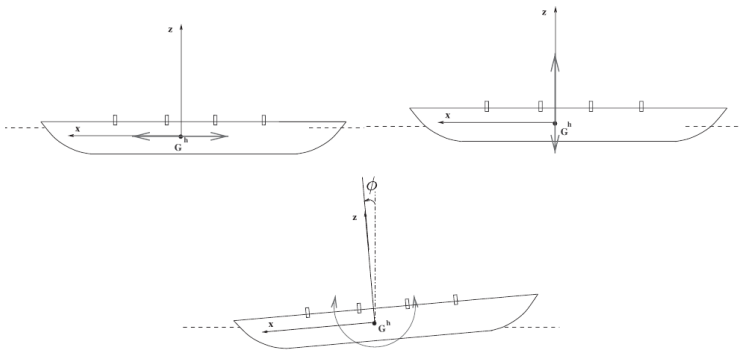



**Figura 1.4:** Movimenti secondari prodotti dall'azione dei vogatori.

dai vogatori durante la cosìdetta fase *drive* della vogata, quando essi spingono contro la pedana e tirano i remi per forzare la lama del remo attraverso l'acqua. Questa azione produce una forza con componente verticale rilevante. Infine il centro di massa dei vogatori si muove a causa della rotazione delle spalle e dallo scorrere dei seggiolini. La combinazione di questi effetti conduce a un cambiamento nell'inerzia angolare della barca che è controbilanciata dal moto rotazionale, che porta al *moto di beccheggio*. Per semplificare l'analisi della dinamica della canoa si è considerato il caso in cui i moti secondari nella direzione che non sono parallele al piano di simmetria dello scafo siano trascurabili. Il problema può essere pensato come costituito da tre diversi sottosistemi che interagiscono tra di loro, cioè:

1. Lo *scafo*. Come ipotizzato nel modello fluidodinamico, si assume sia un corpo rigido di massa e momento di inerzia noto, considerando i seggiolini scorrevoli di massa trascurabile. Il suo centro di massa sarà indicato con $\mathbf{G}^h$ e la sua posizione è una delle incognite per il modello della "struttura".

2. I *vogatori*. Nella maggior parte dei casi la loro massa prevale largamente quella dello scafo. In *Kimè* viene ipotizzato che ogni vogatore possa essere approssimato da un insieme di masse puntiformi corrispondenti alle parti principali del corpo che si muovono secondo il modello di moto dei vogatori.

3. I *remi*. I remi sono assunti agire come leve perfette con massa trascurabile e costituite da lame perfette. Quindi il fulcro della leva è posizionata sulla pala del remo.

### La dinamica dello scafo

La dinamica dello scafo può essere descritto dal seguente sistema di equazioni:

$$\begin{aligned} M\ddot{\mathbf{G}}^h = &\quad \textstyle\sum_{j=1}^{n} \mathbf{F}_{o_j} + \sum_{j=1}^{n} \mathbf{F}_{s_j} + \sum_{j=1}^{n} \mathbf{F}_{f_j} + M\mathbf{g} + \mathbf{F}_w \\ I_{yy}\ddot{\phi} = &\quad \textstyle\sum_{j=1}^{n} (\mathbf{X}_{o_j} - \mathbf{G}^h) \times \mathbf{F}_{o_j} + \sum_{j=1}^{n} (\mathbf{X}_{s_j} - \mathbf{G}^h) \times \mathbf{F}_{s_j} \\ &+ \textstyle\sum_{j=1}^{n} (\mathbf{X}_{f_j} - \mathbf{G}^h) \times \mathbf{F}_{f_j} + \mathbf{M}_w \end{aligned} \tag{1.49}$$

dove $I_{YY}$ è il momento di inerzia dello scafo nella direzione $Y$ e relativa al suo centro di massa, $\mathbf{g}$ è l'accelerazione di gravità e $M$ è la massa dello scafo. Sul lato destro abbiamo le forze esterne e il momento angolare applicato alla imbarcazione e in particolare le due interazioni con il fluido circostante sono indicate con $\mathbf{F}^W$ e $\mathbf{M}^W$ e saranno le forze e momenti calcolate da *Stratos++*. Con $\mathbf{F}_{o_j}$, $\mathbf{F}_{s_j}$, $\mathbf{F}_{f_j}$ si indicano invece le forze esterne esercitate dal $j$-esimo vogatore sugli scalmi, seggiolini e pedane, rispettivamente. Esse possono essere ottenute dalle equazioni che governano la dinamica dei vogatori.

### 1.2.1 Equazioni del moto per i vogatori

La distribuzione di massa di un atleta di date caratteristiche viene data suddividendo il corpo in $p = 12$ parti, la cui massa $m_{ij}$ è ricavata da una tabella anatomica. Ogni parte è poi considerata come concentrata nel suo centro di massa $\mathbf{X}_{ij}$ trascurando il momento di inerzia. L'equazione del momento per il $j$-esimo vogatore è poi data dal seguente sistema:

$$\begin{aligned} \textstyle\sum_{i=1}^{p} m_{ij}(\ddot{\mathbf{X}}_{ij} - \mathbf{g}) = \quad & \mathbf{F}_{h_j} + \mathbf{F}_{s_j} + \mathbf{F}_{f_j} \\ \textstyle\sum_{i=1}^{p} m_{ij}(\mathbf{X}_{ij} - \mathbf{G}^h) \times (\ddot{\mathbf{X}}_{ij} - \mathbf{g}) = \quad & (\mathbf{X}_{h_j} - \mathbf{G}^h) \times \mathbf{F}_{h_j} + (\mathbf{X}_{s_j} - \mathbf{G}^h) \times \mathbf{F}_{s_j} \\ & +(\mathbf{X}_{f_j} - \mathbf{G}^h) \times \mathbf{F}_{f_j} \end{aligned} \tag{1.50}$$

dove con $\mathbf{F}_{h_j}$ si è indicata la forza esercitata dalla mano del $j$-esimo vogatore, mentre con $\mathbf{X}_{h_j}$,$\mathbf{X}_{s_j}$ e $\mathbf{X}_{f_j}$ si sono indicate rispettivamente le posizioni delle mani, seggiolini e pedana. Si osservi inoltre come il momento angolare sia stato calcolato rispetto al baricentro dello scafo $\mathbf{G}^h$.

### 1.2.2 La dinamica dei remi

All'interno del codice di *Kimè* si fa l'ipotesi che il remo sia una leva ideale, con una massa trascurabile e rigidità infinita, cosìcchè tutte le forzi e i momenti che agiscono siano sempre bilanciati. Sotto queste ipotesi, i valori per le forze delle mani $\mathbf{F}_h$ e degli scalmi $\mathbf{F}_o$ sono proporzionali e il coefficiente di proporzionalità dipende dalla configurazione geometrica della barca. Le forze che agiscono su ogni remo sono $-\mathbf{F}_o$, $-\mathbf{F}_h$ e $-\mathbf{F}_w$ che corrispondono all'azione esercitata dall'acqua sulla lama del remo. Dal bilancio delle forze e del momento si ha

$$\mathbf{F}_h = -\frac{L - r_h}{L}\mathbf{F}_o \tag{1.51}$$

Chiaramente, questo semplice modello potrebbe essere migliorato avendo una descrizione più dettagliata della lama.

### 1.2.3 Equazioni complessiva

Sostituendo nelle equazioni (1.49) i valori di $\mathbf{F}_{s_i} + \mathbf{F}_{f_i}$ ottenuti da (1.50)a e i valori di $(\mathbf{X}_{s_i} - \mathbf{G}^h) \times \mathbf{F}_{s_i} + (\mathbf{X}_{f_i} - \mathbf{G}^h) \times \mathbf{F}_{f_i}$ ottenute dall'equazioni (1.50)b

si ha:

$$\begin{aligned} M(\ddot{G}^h - \mathbf{g}) = \quad & \tfrac{r_h}{L}\sum_{j=1}^{n}\mathbf{F}_{o_j} - \sum_{j=1}^{n}\sum_{i=1}^{p} m_{ij}(\ddot{\mathbf{X}}_{ij} - \mathbf{g}) + \mathbf{F}^w \\ I_{YY}\ddot{\phi} = \quad & \sum_{j=1}^{n}\left[(\mathbf{X}_{o_j} - \mathbf{G}^h) - \tfrac{L-r_h}{L}(\mathbf{X}_{h_j} - \mathbf{G}^h)\right] \times \mathbf{F}_{o_j} \\ & - \sum_{j=1}^{n}\sum_{i=1}^{p}(\mathbf{X}_{ij} - \mathbf{G}^h) \times m_{ij}(\ddot{\mathbf{X}}_{ij} - \mathbf{g}) + \mathbf{M}^w \end{aligned} \tag{1.52}$$

Esprimendo le posizioni e accelerazioni delle parti del corpo dei vogatori nel sistema di riferimento solidale allo scafo, otteniamo il sistema finale di equazioni ordinarie differenziali per le incognite $\mathbf{G}^h$ e $\phi$, dove con $M^r = \sum_{i,j} m_{ij}$ e $M^t = M + M^r$ indichiamo la massa totale dei vogatori e quello dello scafo, che comprende sia i vogatori che lo scafo, rispettivamente. Cioè:

$$\begin{aligned} & M^t\ddot{\mathbf{G}}^h + \tfrac{d\mathcal{R}}{d\psi}(\psi)\textstyle\sum_{i,j} m_{ij}\mathbf{x}_{ij}\ddot{\psi} + 2\tfrac{d\mathcal{R}}{d\psi}(\psi)\sum_{i,j} m_{ij}\dot{\mathbf{x}}_{ij}\dot{\psi} - \mathcal{R}(\psi)\left(\sum_{i,j} m_{ij}\mathbf{x}_{ij}\right)\dot{\psi}^2 \\ & \qquad = -\mathcal{R}(\psi)\textstyle\sum_{ij} m_{ij}\ddot{\mathbf{x}}_{ij} + \tfrac{r_h}{L}\sum_{j=1}^{n}\mathbf{F}_{o_j} + M^t\mathbf{g} + \mathbf{F}^w \end{aligned} \tag{1.53}$$

e

$$\begin{aligned} & \mathcal{R}(\psi)\textstyle\sum_{i,j} m_{ij}\mathbf{x}_{ij} \times \ddot{\mathbf{G}}^h + \left(I_{YY} + \sum_{i,j} m_{ij}|\mathbf{x}_{ij}|^2\right) + 2\sum_{i,j} m_{ij}\mathcal{R}(\phi)\mathbf{x}_{ij} \times \mathcal{O}(\phi)\dot{\mathbf{x}}\dot{\phi} \\ & = -\mathcal{R}(\phi)\textstyle\sum_{i,j} m_{i,j}\mathbf{x}_{i,j} \times \mathcal{R}(\phi)\ddot{\mathbf{x}}_{ij} + \mathcal{R}(\phi)\sum_i \left(x_{o_i} - x_{h_i} + \tfrac{r_h}{L}x_{h_i}\right) \times F_{o_i} \\ & + \mathcal{R}(\phi)\textstyle\sum_{i,j} m_{ij}\mathbf{x}_{ij} \times \mathbf{g} + \mathbf{M}^W \end{aligned} \tag{1.54}$$

Per chiudere le equazioni è necessario fornire dei modelli adeguati per la legge di moto dei vogatori e le forze agli scalmi. Questi modelli sono ricavati da dati sperimentali, e per ulteriori dettagli si rimanda a [FMMM09]. All'interno del software *Kimè* viene data anche la possibilità di risolvere la fluidodinamica intorno allo scafo attraverso un modello a potenziale, nel nostro caso invece invece le forze e i momenti fluidodinamici $\mathbf{F}^W$ e $\mathbf{M}^W$ verranno ricavati, come già detto, utilizzando il software *Stratos++*.

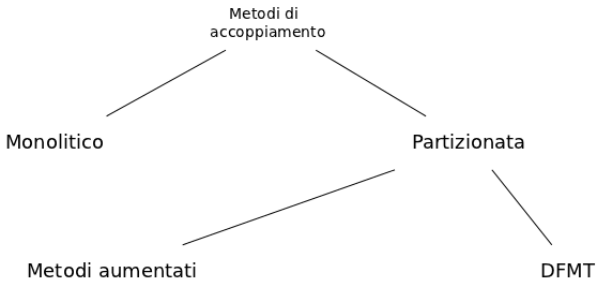



**Figura 1.5:** Possibili strategie di accoppiamento fluido-struttura.

## 1.3 Accoppiamento fluido-struttura

La simulazione numerica di fenomeni di interazioni fluido-struttura (FSI, *fluid structure interaction*), si incontra spesso in molte applicazioni ingegneristiche, per esempio nella dinamica di un imbarcazione, nelle oscillazioni di un profilo alare, o nell'interazioni del vento con ponti sospesi. Presentiamo ora brevemente alcune strategie attraverso le quali sia possibile risolvere il problema dell'accoppiamento, concentrando poi la nostra attenzione su due metodi di tipo *Conventional Serial Staggered*(CSS), e un metodo *Improved Serial Staggered* (ISS).

In figura (1.5) vengono mostrate le principali tecniche con cui possono essere accoppiati due modelli. Osserviamo come la prima distinzione possibile si ha tra una strategia *Monolitica* e una strategia *Partizionata*. Nella prima, il problema accoppiato è introdotto e risolto come se fosse un unico modello, usando solitamente le stesse tecniche di discretizzazione per entrambi i problemi accoppiati. Questo ci porta ad ottenere dei risolutori spesso efficienti ma computazionalmente onerosi. Inoltre la diversa natura del problema fluidodinamico e del problema strutturale porta a volte ad utilizzare tecniche di risoluzione diverse nei due sottosistemi. Per questo una seconda possibilità sta nel mantenere l'indipendenza tra i due risolutori. Con questo approccio, ogni risolutore è basato su una differente tecnica di discretizzazione, potendo impiegare schemi di integrazionee perfino passi temporali diversi. In particolare concentreremo la nostra attenzione sulle strategie di soluzione partizionate.

### 1.3.1 Strategia partizionata

L'approccio di tipo *Partizionato* è stato introdotto per la prima volta in [FdR77] per modellizzare l'impatto della cavitazione acustica sui sottomarini, ed era principalmente motivato dalla mancanza di accesso al codice usato per risolvere uno dei campi. Una delle problematiche fondamentali in questo tipo di strategia, già noto nelle prime pubblicazioni, sta nella difficoltà di ottenere un algoritmo di partizionamento stabile. Oltre alla sincronizzazione temporale, una seconda difficoltà dovuta a un approccio di tipo partizionato è quello di trasmettere nel modo più accurato possibile le informazioni all'interfaccia dei sottoproblemi sia per discretizzazioni spaziali conformi che non conformi. Questa seconda problematica risulta rilevante nel caso in cui la struttura non fosse rigida.

I metodi partizionati possono essere divisi in due grandi categorie:

- *Direct Force Motion Transfer*(DFMT). Essi sono i primi metodi introdotti per risolvere FSI in un modo partizionato. Nel DFMT, non si ha l'aggiunta di nuove incognite all'interfaccia, ma le quantità primali (spostamento e velocità) di un sottoproblema e le quantità duali dell'altro (pressione o forza) sono scambiate in modo implicito o esplicito, iterativo o non iterativo. La condizione all'interfaccia non è mai rispettata in modo stretto fino a convergenza dei metodi. Di conseguenza la disuguaglianza è rispettata a una certa tolleranza. Nel caso di accoppiamento esplicito, si accetta un errore addizionale dovuto all'accoppiamento e ci si deve assicurare che non introduca instabilità.

- *Metodi Aumentati*, detti anche *Localized Lagrange Multipliers*(LMM). L'idea generale è di introdurre un campo di moltiplicatori di Lagrange, e di esprimere la condizione di accoppiamento introducendo dei vincoli all'interfaccia.

In molti casi i metodi basati su LLM mostrano migliori proprietà di convergenza rispetto ai metodi DFMT, permettendo, per esempio, l'uso di finestre temporali più ampie senza perdita di stabilità. Per quanto riguarda la notazione qui utilizzata ci siamo ispirati al lavoro di [Kas] e di [Pip97]. Indicheremo con il pedice $f$ le variabili riferite al problema del fluido, mentre con il pedice $s$ le variabili associate al problema sulla struttura.

Le condizioni da imporre all'interfaccia riflettono due classici principi di meccanica:

- *Continuità dello spostamento*, cioè

$$\mathbf{u}_s = \mathbf{u}_f \qquad \text{su } \Gamma \times (0,T) \tag{1.55}$$

  Si osservi come questa condizione implichi

$$\mathbf{v}_s = \mathbf{v}_f \qquad \text{su } \Gamma \times (0,T) \tag{1.56}$$

  dove $\mathbf{v} = \dot{\mathbf{u}}$.

- *Continuità degli sforzi*:

$$\sigma(\mathbf{v}_f, p) \cdot \mathbf{n} = \sigma(\mathbf{v}_s) \cdot \mathbf{n} \qquad \text{su } \Gamma \times (0,T) \tag{1.57}$$

  dove con $\mathbf{n}$ si è indicato il versore normale e con $\sigma$ il tensore degli sforzi di Cauchy sull'interfaccia.

Tuttavia, nei problemi di interazione con fluidi a superficie libera le condizione di interfaccia 1.56 e 1.59 risultano problematiche in quanto non permettono di modificare la linea di galleggiamento. Quindi si preferisce utilizzare un accoppiamento di tipo *slip*, imponendo:

- condizione di *impenetrabilità*

$$\mathbf{v}_s \cdot \mathbf{n} = \mathbf{v}_f \cdot \mathbf{n} \qquad \text{su } \Gamma \times (0,T) \tag{1.58}$$

- condizione dinamica

$$\mathbf{n}^T \sigma(\mathbf{v}_f, p) \cdot \mathbf{n} = \mathbf{n}^T \sigma(\mathbf{v}_s) \cdot \mathbf{n} \qquad \text{su } \Gamma \times (0,T) \tag{1.59}$$

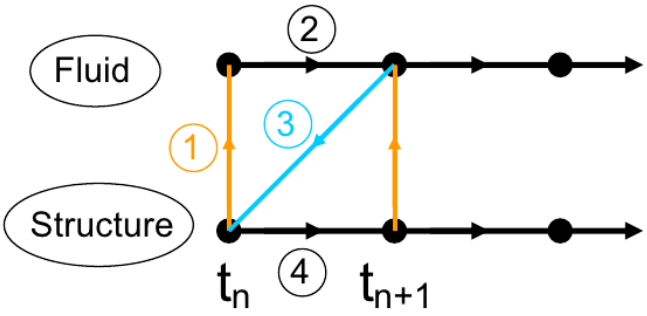



**Figura 1.6:** Procedura di accoppiamento CSS

La condizione dinamica può essere quindi aumentata con una condizione sulla componente tangenziale degli sforzi di Cauchy all'interfaccia per tenere conto dell'attrito causato dal fluido sull'imbarcazione, come mostrato in [LT11]. Notiamo che nel nostro caso la componente normale dello sforzo di Cauchy all'interfaccia è proprio il moltiplicatore di Lagrange $\lambda$.

In entrambi i casi, la condizione dinamica può essere riformulata introducendo due operatori $\mathcal{S}_f$ e $\mathcal{S}_s$ associati al rispettivamente al problema fluidodinamico e a quello alla struttura, che dato lo spostamento della struttura ne restituiscono le forze agenti sullo scafo. Il principio di azione e reazione può quindi essere riscritto come segue:

$$\begin{aligned} &\text{Trovare } \mathbf{u} \text{ su } \Gamma \times [0,T] \text{t.c.} \\ &\mathcal{S}_f(\mathbf{u}) + \mathcal{S}_s(\mathbf{u}) = 0 \end{aligned} \tag{1.60}$$

Usando l'operatore inverso di $\mathcal{S}_s$ possiamo scrivere l'equilibrio della quantità duale come segue:

$$\begin{aligned} &\text{Trovare } \mathbf{u} \text{ su } \Gamma \times [0,T] \text{t.c.} \\ &\mathbf{u} = \mathcal{S}_s^{-1}\left(-\mathcal{S}_f(\mathbf{u})\right) = 0 \end{aligned} \tag{1.61}$$

L'equazione (1.61) corrisponde a determinare i *punti fissi* dell'applicazione $\Psi = S_s^{-1} \circ S_f$. Quindi per trovare la soluzione di (1.61) sarebbe necessario introdurre un metodo iterativo, che assegnato $\mathbf{u}^0$, calcoli $\mathbf{u}^{k+1}$ come

$$\mathbf{u}^{k+1} = \Psi(\mathbf{u}^k) \tag{1.62}$$

procedura che deve essere ripetuta fino a quando non sia verificato un criterio di convergenza, cioè $\|\mathbf{u}^{k+1} - \mathbf{u}^k\| \leq \epsilon$, secondo una particolare norma, con $\epsilon$ valore di tolleranza. Tuttavia una strategia di questo tipo risulta computazionalmente onerosa, in quanto ad ogni iterazione temporale è necessario risolvere più volte il problema alla struttura e alla fluidodinamica fino al raggiungimento di convergenza. Per questo motivo, in [PFL95] e [PF01] vengono presentate alcune strategie per evitare questa procedura e garantire stabilità numerica. Nelle prossime due sezioni esporremo brevemente due possibili strategie.

**Conventional Serial Staggered**

Iniziamo con la più semplice procedura di tipo *Conventional Serial Staggered* (CSS), schematizzata in 1.6. Per ogni ciclo integrazione $\Delta t_s$ si eseguono le seguenti operazioni:

1. Risolvere il problema fluidodinamico, eventualmente con più sottocicli nel caso in cui $\Delta t_f < \Delta t_s$, considerando la posizione dell'imbarcazione costante e pari a quella assegnata dal problema alla struttura al passo temporale precedente.

2. Risolvere il problema alla struttura con passo di integrazione $\Delta t_s$ ponendo $\mathbf{F}_s^{n+1} = \tilde{\mathbf{F}}_f^{n+1}$, dove $\tilde{\mathbf{F}}_f^{n+1}$ è funzione delle forze calcolate nelle iterazioni precedenti dal risolutore del problema fluidodinamico.

3. Assegnare la posizione della struttura $\mathbf{u}_s^{n+1}$ al problema fluidodinamico.

Osserviamo che l'equazione di continuità per lo spostamento (1.55) è forzata alla fine del passo temporale globale, mentre l'equazione di continuità della velocità (1.58) in generale non vale più, e può essere recuperata solo introducendo, come vedremo nel seguito, una procedura *staggered* sfalsata.

Parecchie scelte sono possibili per le forze $\tilde{\mathbf{F}}_f^{n+1}$ del sottopasso 2. Una possibilità è prendere come $\tilde{\mathbf{F}}_f^{n+1} = \mathbf{F}_f^n$, oppure una media temporale

$$\tilde{\mathbf{F}}_f^{n+1} = \overline{\mathbf{F}}^{n+1/2} = \frac{1}{\Delta t_S} \sum_{i=1}^{n_{\mathbf{F}/S}} \Delta t_{\mathbf{F}_k} \mathbf{F}_k^n \tag{1.63}$$

dove la somma è esteso su tutti i sottocicli, $\mathbf{F}_k^{n-1}$ è la forza di pressione calcolata al $k$-esimo sottociclo e $n_{F/S}$ indica il numero di passi necessari per raggiungere il passo temporale $\Delta t_S$. Infine è possibile scegliere anche una correzione della forza media del tipo:

$$\tilde{\mathbf{F}}_f^{n+1} = 2\overline{\mathbf{F}}^{n+1/2} - \tilde{\mathbf{F}}_f^n \tag{1.64}$$

Recentemente in [PFL95] sono state introdotte delle procedure *staggered* che usano un predittore strutturale. In breve, esse permettono di ridurre in modo considerevole l'errore di conservazione dell'energia. L'idea generale è semplice: assumiamo di poter costruire con buon accuratezza una predizione dello stato della struttura alla fine del prossimo passo temporale. A questo punto si può riscrivere la nuova procedura staggered come segue:

1. Calcolare una predizione $\tilde{\mathbf{u}}^{n+1}$ per lo spostamento della struttura alla fine del corrente passo temporale.

2. Risolvere tanti passi del risolutore fluidodinamico necessari per completare il passo temporale $\Delta t_s$, tenendo costante la posizione della struttura a $\tilde{\mathbf{u}}^{n+1}$.

3. Calcolare le forze fluidodinamiche $\mathbf{F}_f^{n+1}$ di input per il problema sulla struttura.

4. Eseguire un integrazione strutturale in tempo sul passo $\Delta t_S$ con $\mathbf{F}_s^{n+1} = \mathbf{F}_f^{n+1}$.

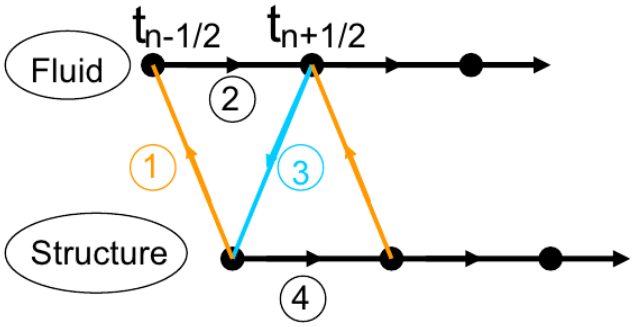



**Figura 1.7:** Procedura di accoppiamento ISS.

Alla fine del passo temporale, l'equazione di continuità dello spostamento della struttura (1.55) è *a priori* non soddisfatta a meno che il predittore strutturale non sia perfetto, tuttavia introducendo un predittore della posizione, a ottenere un risultato più vicino alla posizione reale, rispetto alle più accurate procedure staggered senza predittore. Il primo predittore al quale si può pensare è un predittore del primo ordine per lo spostamento della struttura:

$$\tilde{\mathbf{u}}^{n+1} = \mathbf{u}^n + \Delta t_s \mathbf{v}^n \tag{1.65}$$

oppure predittori del secondo ordine come i seguenti:

$$\tilde{\mathbf{u}}^{n+1} = \mathbf{u}^n + \Delta t_S (1.5\mathbf{v}^n - 0.5\mathbf{v}^{n-1}) \tag{1.66}$$

$$\tilde{\mathbf{u}}^{n+1} = \mathbf{u}^n + \Delta t_S \mathbf{v}^n + 0.5\Delta t_S \mathbf{a}^n \tag{1.67}$$

mentre per le forze di input $\mathbf{F}_f^{n+1}$ dello step 3, possiamo usare una delle approssimazioni precedentemente introdotte.

### Improved Serial Staggered

Questo tipo di algoritmo, presentanto in [PF01] per la soluzione di problemi aeroelastici, permette di ottenere un accoppiamento più accurato. Infatti, a differenza della procedura CSS, non introduce errori di predizione dello scambio di energia tra il fluido e la struttura sull'interfaccia $\Gamma$. Ci riferiremo a questo metodo come *Improved Serial Staggered* (ISS) dato che è capace di calcolare in modo accurato le soluzioni usando un passo di tempo che è comparabile a quello che ci si può permettere per uno schema monolitico completamente implicito e per metodi fortemente accoppiati.

La descrizione del metodo *ISS*, mostrato in 1.7, è la seguente:

1. Data qualche condizione iniziale $\mathbf{F}^0$, $\mathbf{u}^0$ e $\dot{\mathbf{u}}^0$, inizializziamo la dinamica del fluido come segue
$$\mathbf{u}_f^{-1/2} = \mathbf{u}^0 - \frac{\Delta t}{2}\dot{\mathbf{u}}^0 \tag{1.68}$$
2. poniamo
$$\dot{\mathbf{u}}_f^n = \dot{\mathbf{u}}_s^n \tag{1.69}$$

e aggiornare la posizione della struttura per il problema fluidodinamico come segue:

$$\mathbf{u}_f^{n+1/2} = \mathbf{u}_f^{n-1/2} + \Delta t \dot{\mathbf{u}}_f{}^n \tag{1.70}$$

3. risolvere il problema fluidodinamico ottenendo $\mathbf{F}_{n+1/2}$.

4. avanzare il sistema strutturale usando in input le forze $\mathbf{F}_{n+1/2}$.

Inoltre, come mostrato in [FL00], questo algoritmo forza sia l'equazione di continuità (1.55) che la *Legge di Conservazione Geometrica*, una condizione che richiede che uno schema numerico riproduce una soluzione costante in modo esatto, nel caso venga scelto uno schema di integrazione di tipo punto medio per la parte solida.

Nonostante le ottime proprietà di accuratezza presentate nella procedura ISS si è scelto di implementare CSS con un predittore del primo ordine. La motivazione che sta alla base di questa scelta è da ricercasi nello schema di integrazione temporale usato nel software *Kim*è. Infatti lo schema di avanzamento in tempo usato è lo schema di Runge-Kutta-Fehelberg 4-5 adattivo, scelto per seguire in modo efficace la dinamica dei vogatori, e non una regola del punto medio, che permetterebbe di sfruttare le potenzialità di un accoppiamento di tipo ISS.

# Capitolo 2

# Aspetti implementativi

## 2.1 Organizzazione del codice

Il codice *Stratos++* è organizzato come in figura 2.1. Si può osservare come tutto il codice sia scomponibile in cinque blocchi:

1. *LinearAlgebra*: si dedica alla gestione delle strutture per la risoluzione di sistemi lineari;
2. *Problem*: si occupa della risoluzione del problema fluidodinamico;
3. *TriangulatedHull*: integra tutte le funzionalità per interfacciarsi con scafi triangolati;
4. *Interp*: permette di calcolarne le forze su una geometria triangolata attraverso alcune tecniche di interpolazione;
5. *FSI*: si prende carico di accoppiare il modello per il fluido con quello della struttura e di interfacciarsi con il codice *Kimè*.

Uno degli obbiettivi che ci si è posti durante l'implementazione della nuova versione del codice è stata quella di modificare il meno possibile *Stratos++V.1.0*. In figura 2.1 evidenziamo in giallo le classi già implementate in *Stratos++V.1.0* che non si è dovuto modificare, in blu le classi già implementate ma che è stato necessario modificare, mentre in rosso le nuovi classi introdotte.
Le classi modificate sono quelle che poi si devono interfacciare con la geometria triangolata, cioè la classe `Geometry` e la classe `Scull`, mentre non è stato necessario modificare la parte legata al risolutore fluidodinamico.
Nelle sezioni successive presenteremo le classi introdotte e le modifiche apportate, mentre per la descrizione della classi non modificate si rimanda a [Alt09].
All'interno del codice, particolare attenzione è stata rivolta alle prestazioni, grazie all'impiego massiccio delle strutture e degli algoritmi forniti dalla Standard Template Library del C++, oltre che di librerie esterne altamente ottimizzate. Ad esempio, per la gestione di piccole strutture algebriche, come vettori e matrici a 2 o 3 dimensioni, vengono utilizzati i metodi forniti da *Eigen*, una potente Template Library, impiegata anche dall'Agenzia Spaziale Europea per

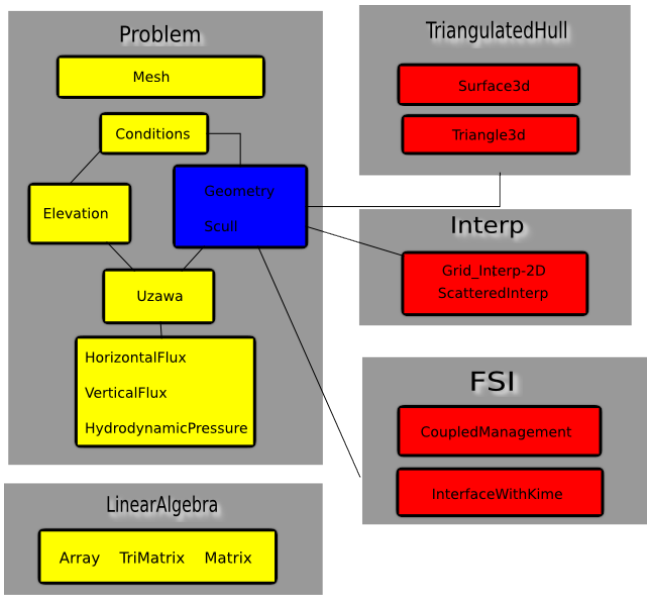



**Figura 2.1:** Organizzazione codice *Stratos++V.2*.

alcuni progetti[1] e già usata nella precedente versione di *Stratos++*. Inoltre per la risoluzione di equazioni non lineari ci si è appoggiati alla libreria *GSL* (*GNU Scientific Library*)[2], una libreria numerica che fornisce diverse strumenti utili, da generatori di numeri casuali a risolutori di equazioni differenziali.

## 2.2 La classe `Boat`

Costituisce un interfaccia pubblica per le classi che vanno a descrivere la geometria dello scafo. Nel caso si volesse implementare una nuovo scafo definito da una funzione analitica o nel caso ci si volesse interfacciare con superfici provenienti da file CAD, basterà derivare pubblicamente la nuova geometria da questa classe e implementare i seguenti metodi:

- `Domain` che date le coordinate $x$ e $y$ del punto determina se cade sulla superficie dello scafo;
- `Shape` che date le coordinate $x$ e $y$ del punto mi restituisce la coordinata $z$ del punto se questo è interno alla superficie, altrimenti restituisce la coordinata $z$ del baricentro.

Da questa classe derivano pubblicamente le classi `WigleyHull` e `Elliptic` che implementano lo scafo Wigley-Hull e lo scafo ellittico, già supportati dalla precedente versione di *Stratos++*. Esiste inoltre una terza classe, la quale anch'essa

[1] L'ESA (European Space Agency ) si avvale della libreria Eigen per il suo codice di calcolo delle traiettorie nello spazio, all'interno del progetto STA (*Space Trajectory Analysis*). Per tutti i dettagli relativi alla libreria si rimanda alla pagina web del progetto: `http://eigen.tuxfamily.org/`.

[2] Per ulteriori dettagli si veda la pagina web `http://www.gnu.org/s/gsl/`.

deriva pubblicamente da `Boat`, cioè la classe `TriangulatedHull`, introdotta per la gestione di geometrie triangolate. Membro principale contenuto all'interno di questa classe è `_grid`, un oggetto di tipo `Surface3d` che contiene la mesh dello scafo e implementa i metodi e le strutture introdotte per determinare la quota dello scafo.

## 2.3 La classe `Scull`

La classe `Scull` si occupa di costruire e gestire la superficie esterna della barca. In particolare, per ogni punto appartenente al dominio bidimensionale, essa deve essere in grado di fornire la quota a cui si colloca la chiglia dello scafo, in funzione della posizione del suo centro di massa e degli angoli caratteristici. Infatti, a seconda delle condizioni del fluido in cui si trova a navigare, l'imbarcazione è soggetta a forze che le fanno variare gli angoli di beccheggio e di rollio, istante per istante. In particolare, è stato necessario introdurre due nuovi membri:

- `_hull`: un puntatore a un oggetto del tipo della classe virtuale `Boat`, da cui derivano pubblicamente, come visto, sia le classi per la rappresentazioni di scafi analitici, sia la classe per scafi triangolati.
- `_keel`: un funtore che può essere inizializzato con `keelTriangulate` o `keelAnalitic`, che permette di ottenere la posizione della barca nella configurazione rototraslata.

Queste modifiche sono state necessarie per rendere il codice adatto a interfacciarsi sia con geometrie descritte tramite funzioni analitiche $z = z(x, y)$, sia che con geometrie triangolate, senza dover sconvolgere completamente l'organizzazione di *Stratos++*.
Per quanto riguarda i metodi di `Scull`, la principale modifica riguarda il metodo `getKeel`. Esso richiama una funzione di tipo `keel` che viene assegnata nel costruttore di `Scull`, e che, come detto, può essere `KeelAnalitic` o `KeelTriangulate`. Nella versione analitica, la risoluzione dell'equazione non lineare per la ricerca delle coordinate del generico punto $P$ nella configurazione rototraslata viene fatta per ogni nodo del dominio $\Omega_h$, e risolvendo l'equazione non lineare con un metodo di bisezione. Dopodichè si testa se il punto ottenuto sia all'interno dell'imbarcazione, e nel caso questa condizione sia verificata, se ne restituisce la posizione sempre nella configurazione rototraslata.
In linea di principio, questa stessa implementazione potrebbe essere usata per il caso di scafi triangolati, in quanto all'interno di questo metodo si chiamano le funzioni `Shape` e `Domain` di `Boat` che, sfruttando il polimorfismo e l'ereditarietà, può rappresentare una geometria analitica o triangolata. Tuttavia computazionalmente questa procedura risulterebbe troppo costosa nel caso di scafi triangolati, questo perchè valutare una funzione di cui si ha l'espressione analitica non ha lo stesso costo che ricercare per ogni punto, la coordinata $z$ data dall'intersezione tra la retta $r$ e la superficie $\mathcal{B}_h$ che descrive lo scafo. Una prima operazione che viene fatta è testare a priori se il punto nella configurazione rototraslata sarà esterno allo scafo, senza dover necessariamente risolvere l'equazione non lineare. Per fare ciò si sfrutta il fatto che il sistema di riferimento sia solidale con lo scafo e che i possibili moti simulati dal codice sono quello di beccheggio, quello di rollio, di affondamento ma non di imbardata. Sotto queste

ipotesi, se un punto non verifica:

$$\begin{array}{c} |x - x_b| \leq L/2 \\ |y - y_b| \leq W \end{array} \tag{2.1}$$

allora, anche nella configurazione rototraslata, sarà sicuramente esterno allo scafo, per qualunque angolo di beccheggio o rollio. Nel caso ci fosse anche un angolo di imbardata diverso da zero, questo test funziona ancora se l'angolo è piccolo, altrimenti bisognerebbe in generale aumentare il secondo limite. L'ipotesi di angolo di imbardata piccolo o nullo risulta inoltre sensata nel caso in cui la canoa sia del tipo *Quad Scull*, o se i vogatori siano professionisti e quindi estremamente sincronizzati.
Inoltre per rendere più veloce la risoluzione dell'equazione non lineare ci si è appoggiati al *metodo di Brent-Dekker*, implementato nelle librerie *GSL*, che combina una strategia di interpolazione con un algoritmo di bisezione. Questo produce una procedura di risoluzione veloce e robusto. Ad ogni iterazione il metodo di Brent approssima la funzione usando una curva interpolante. Alla prima iterazione, questa è una interpolazione lineare dei due estremi. Per ogni successiva iterazione, l'algoritmo usa un'interpolazione inversa quadratica passante per gli ultimi tre punti, per maggior accuratezza. L'intercetta della curva interpolante con l'asse $x$ è presa come possibile radice dell'equazione non lineare. Se questo punto giace all'interno del corrente intervallo, allora il punto di interpolazione è accettato e usato per generare un intervallo di interpolazione più piccolo. Se il punto interpolante non è accettato allora l'algoritmo ripiega su un ordinario passo di bisezione. La migliore stima della radice è presa dalla più recente interpolazione o bisezione.

## 2.4 La classe `Surface3d`

Una delle modifiche più importanti introdotte nella nuova versione di *Stratos++* è sicuramente la possibilità di interfacciarsi con superficie di scafi triangolati, il cui compito è delegato alla classe `Surface3d`, la quale, oltre a contenere la mesh, implementa il metodo che permette di determinare la quota dello scafo date le coordinate $(x, y)$.
I possibili file mesh che possono essere importati sono i seguenti:

- **.msh**: Il formato di file MSH ASCII, formato diffuso per la rappresentazione di mesh in uno spazio bidimensionale e tridimensionale. Nel caso più semplice, che corrisponde a quello considerato, il file conterrà prima l'elenco dei nodi e successivamente l'elenco degli indici ai nodi che vanno a costruire ogni singolo elemento. Un esempio di software che genera mesh in questo formato è *Gmsh*.
- **.neu**: GAMBIT neutral file, sono file ASCII che possono essere usati per importare o esportare mesh, condizioni al bordo o soluzioni su nodi o celle. In particolare questo è il formato che viene supportato da *Kimè*;

In generale se si volessero importare file di altri formati, è necessario che il file contenga l'elenco dei nodi e gli indici dei vertici dei triangoli della mesh. Infatti date queste informazioni, è possibile inizializzare tutte le strutture della classe. Le strutture fondamentali contenute in `Surface3d` sono le seguenti:

- `_Nodes` un vettore che contiene le coordinate dei vertici degli elementi;
- `_element2element` un vettore che nella posizione $i$ contiene tutti gli elementi adiacenti allo $i$-esimo elemento;
- `_elem2nodes` un vettore che nella posizione $i$ contiene gli indici delle coordinate dei vertici dello $i$-esimo elemento;
- `_node2elem` un vettore che nella posizione $i$ contiene gli indici di tutti gli elementi che hanno in comune lo $i$-esimo vertice;
- `_Elements`: un vettore che contiene i puntatori a oggetti di tipo *Triangle3d*.
- i vettori denominati `_partitionX` e `_partitionY` e la lista `_indexPath`, le cui funzionalità saranno chiare più avanti.

Per inizializzare questi membri sono stati implementati i rispettivi metodi `Set`. Il metodo principale di questa classe è quello denominato `Locate`, che determina la quota dello scafo. Dato che la superficie è triangolata, il processo di determinazione della quota può essere riassunto nei seguenti passi:

1. dato il punto $P = (x, y, 0)$ costruisco la retta $r$ passante per $P$ e parallela all'asse $z$;
2. attraverso un algoritmo di ricerca determino quale triangolo della mesh interseca la retta $r$.
3. calcolo il punto di intersezione tra il triangolo e la retta, la cui coordinata $z$ è la quota cercata.

Il nodo fondamentale di questo algoritmo è il metodo usato per determinare il triangolo che interseca la retta. Infatti all'interno del codice *Stratos++* il metodo `getKeel` della classe `Scull` andrà a chiamare il metodo `Locate` su tutti i circocentri dei triangoli e su tutti i nodi della mesh che descrive $\Omega$, costituita nel nostro caso da circa 175000 elementi e 88000 nodi. Per massimizzare le performance del codice si sono attuate le seguenti scelte implementative:

- cercare di ridurre al minimo il numero di ricerche, determinando con dei test a priori se un punto sia esterno alla superficie dello scafo.
- nel caso in cui il punto fosse interno, trovare il triangolo con un metodo di ricerca computazionalmente efficiente.
- sfruttare le ricerche già svolte per poter velocizzare le future chiamate alla funzione.

### 2.4.1 Riduzione del numero di ricerche

Per scremare il numero di punti su cui applicare l'algoritmo di ricerca del triangolo si può cercare qualche condizione necessaria affinchè le coordinate del punto $P$ siano interne alla proiezione dello scafo sul piano $xy$. Una condizione necessaria affinchè ciò sia vero è

$$\begin{aligned} x_m \leq P_x \leq x_M \\ y_m \leq P_y \leq y_M \end{aligned} \tag{2.2}$$

dove con $(x_m, y_m)$ e $(x_M, y_M)$ sono stati indicati gli estremi dello scafo. Si sta cioè testando se le coordinate siano interne al rettangolo i cui estremi sono le massime e minime coordinate nella direzione $x$ e $y$. Questa condizione vale per qualunque forma, anche per geometrie non convesse. In particolare si evidenzia come sia solo una condizione necessaria, quindi ci assicura solo che i punti che non la verificano sono sicuramente esterni. Ciò nonostante questa è gia un ottima scrematura che permette di ridurre notevolmente il numero di ricerche da effettuare.
Tuttavia ciò non è abbastanza e si è quindi cercato qualche criterio che ci possa dare anche delle condizioni sufficienti per sapere se il punto sia interno. Per semplicità riconduciamoci al caso monodimensionale di ricerca dell'intervallo che contiene un punto, per poi estenderlo al caso di mesh bidimensionale. In questo caso la griglia si riduce a una partizione $\mathcal{P}$ di un segmento, che pensiamo di estremi $[a, b]$. Nel caso di mesh uniforme, dato il passo di griglia, l'intervallo che contiene il punto si determina in modo semplice attraverso la formula

$$i = \left\lfloor \frac{(x-a)}{h} \right\rfloor \tag{2.3}$$

dove $a$ è l'estremo inferiore della griglia e $h$ è il passo di discretizzazione spaziale. Nel caso di griglia non uniforme, non abbiamo una formula che ci permetta di calcolare la posizione dell'intervallo che contiene il punto. Quello che allora si fa è creare una nuova partizione $\mathcal{P}_u$ uniforme dell'intervallo $[a, b]$ con passo di discretizzazione $h_M$, che è la lunghezza del più grande intervallo della griglia, e costruire un vettore, denominato $\mathbf{v}_x$, che contiene nella posizione $i$ l'indice di tutti gli intervalli che hanno il punto medio $\overline{x}$ appartenente allo $i$-esimo intervallo della partizione $\mathcal{P}_u$, cioè

$$a + ih_M \leq \overline{x} < a + (i+1)h_M \tag{2.4}$$

A questo punto, quando si deve ricercare l'intervallo contenente un punto $P$, si costruisce l'insieme $S$ dei candidati intervalli che contengono il punto nel seguente modo:

$$S = \bigcup_{i \in \mathcal{G}} \mathcal{I}_i \tag{2.5}$$

dove l'insieme $\mathcal{G}$ è dato dall'insieme degli indici degli intervalli contenuti nella partizione $\mathcal{P}_u$ tra l'indice $\overline{i}$ e l'indice $\overline{s}$, calcolati secondo la formula (2.3), rispettivamente con $x = P - h_M$ e $x = P + h_M$. In questo modo stiamo cercando l'intervallo tra il sottoinsieme di intervalli per i quali $\overline{x} \in [P - h_M, P + h_M]$. Ovviamente questa procedura riduce notevolmente il numero di intervalli su cui ricercare se la partizione non è eccessivamente anisotropa, altrimenti una possibile soluzione è dividere inizialmente l'intervallo in regioni all'interno delle quali la griglia non sia troppo sbilanciata e applicare a ogni singola regione la procedura appena introdotta.
Cerchiamo ora di generalizzare al caso bidimensionale. Come visto, nel caso monodimensionale come punto di riferimento per "ordinare" i vari intervalli veniva usato il punto medio. Nel caso bidimensionale invece come riferimento si può usare il baricentro o il circocentro dei triangoli della mesh. La scelta è legata alla regolarità della mesh: infatti nel caso in cui la mesh fosse regolare, siamo sicuri che il circocentro sia interno al triangolo e possiamo usarlo come riferimento

prendendo come lunghezza massima (corrisponde a $h_M$ del caso monodimensionale) dell'elemento il massimo del raggio circoscritto $\overline{R}$ della mesh; mentre nel caso in cui la mesh non fosse regolare, il circocentro può trovarsi all'esterno del triangolo, e quindi è preferibile usare come riferimento il baricentro che invece è sicuramente interno, e come lunghezza massima il doppio di $\overline{R}$.
La partizione uniforme $\mathcal{P}_u$ che si andrà a creare sarà ora una griglia di elementi quadrati di lato $\overline{R}$, e la ricerca dell'elemento non viene fatto su tutti i triangoli della mesh, ma solo sugli elementi tali che il baricentro (o il circocentro se la mesh è regolare) appartenga all'intervallo $S \equiv [x-\overline{R}, x+\overline{R}] \times [y-\overline{R}, y+\overline{R}]$ dove con $x$ e $y$ indichiamo le coordinate del punto. Per questo motivo, nel costruttore di `Surface3d`, si vanno a inizializzare i vettori `_partitionX` e `_partitionY` la cui funzionalità è analoga al vettore $\mathbf{v}_x$ solo che, essendo due le dimensioni in gioco, abbiamo una partizione $\mathcal{P}_u^x$ in direzione $x$ e una partizione $\mathcal{P}_u^y$ in direzione $y$ che mi genera poi la partizione $\mathcal{P}_u$ a elementi quadrati.
Per trovare il sottoinsieme $S$ all'interno del quale cercare il triangolo che interseca la retta $r$, si fa quindi l'intersezione tra l'insieme degli indici degli elementi tali che la coordinata $x$ del baricentro sia nell'intervallo $[x - R, x + R]$, sfruttando la struttura `_partitionX`, e l'insieme degli indici degli elementi tali che la coordinata $y$ del baricentro sta nell'intervallo $[y - R, y + R]$, sfruttando la struttura`_partitionY`. Se $S$ ha dimensione nulla, allora il punto su cui si vuole interpolare è esterno alla mesh. Nel caso in cui invece $S$ non abbia dimensione nulla, allora, come già evidenziato, il triangolo che interseca la retta $r$ è sicuramente appartenente a $S$. In questo modo non solo abbiamo introdotto una condizione sufficiente che ci garantisce che il punto sia interno, ma abbiamo anche trovato un sottoinsieme più piccolo rispetto a tutti i triangoli che appartengono alla mesh (nelle simulazioni lo scafo contiene circa 20000 elementi) su cui applicare l'algoritmo di ricerca.

### 2.4.2 Algoritmo di intersezione retta-triangolo

Selezionando il primo elemento di $S$ come elemento iniziale da cui partire con la ricerca, l'algoritmo di ricerca può essere così riassunto:

1. testo se il punto è interno all'elemento attuale, se sì ho trovato l'elemento, altrimenti

2. scelgo come elemento successivo nella ricerca un elemento che non ho già visitato, che appartiene a $S$ e tale che sia il più vicino possibile al punto. Questa funzionalità è svolta dal metodo `NextElement`.

3. se non esiste un tale elemento, allora aggiorno la lista degli elementi visitati, contenuta in `_indexPath`, con quello corrente, torno all'ultimo elemento visitato e riparto da 2.

Per capire quale tra gli elementi adiacenti all'elemento corrente è il più vicino, si calcola la distanza tra i punti medi di ogni lato, proiettati sul piano $xy$, e il punto in questione.
In ultima istanza, è necessario definire un algoritmo che mi permetta di stabilire se una retta interseca un triangolo e di trovarne il punto di intersezione. Una retta $R(t)$ è definita attraverso un origine $O$ e una direzione normalizzata $D$ come

$$R(t) = O + tD \tag{2.6}$$

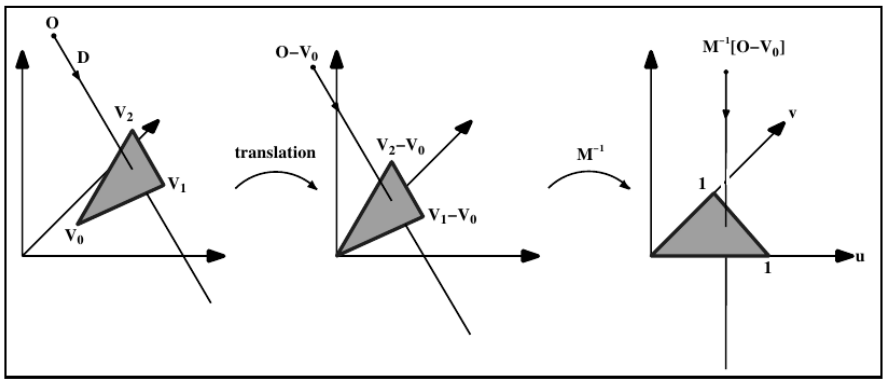



**Figura 2.2:** Traslazione e cambiamento di base dell'origine della retta

e un triangolo è definito attraverso tre vertici $V_0$,$V_1$ e $V_2$. In [Are88] si propone di risolvere il problema calcolando prima l'intersezione tra la retta e il piano nel quale il triangolo giace, e poi testare se il punto di intersezione è interno al triangolo. Noi abbiamo invece scelto di implementare l'algoritmo presentato in [MT05], che usa la minor quantità di memoria possibile (infatti per questo algoritmo necessita solo di memorizzare i vertici del triangolo) e non necessita di nessun preprocessing. Per una mesh triangolata, la memoria risparmiata è significativa, variando tra il 25% e 50%, che dipende dalla quantità di vertici condivisi. In questo algoritmo, una trasformazione è costruita e applicata all'origine della retta. Questa trasformazione ci porta a un vettore contenente la distanza $t$ dall'intersezione e le coordinate, $(u, v)$ della stessa. In questo modo si evita di calcolare l'intersezione retta-piano. Un punto, $T(u, v)$, su un triangolo è dato da

$$T(u, v) = (1 - u - v)V_0 + uV_1 + vV_2 \tag{2.7}$$

dove $(u, v)$ sono le coordinate baricentriche, e affinchè sia interno al triangolo deve soddisfare $u \geq 0$, $v \geq 0$ e $u + v \leq 1$. Calcolare l'intersezione tra la retta, $R(t)$, e il triangolo, $T(u, v)$, è equivalente a risolvere il sistema:

$$O + tD = (1 - u - v)V_0 + uV_1 + vV_2 \tag{2.8}$$

Riarragiando i termini si ha

$$\begin{bmatrix} -D, & V_1 - V_0, & V_2 - V_0 \end{bmatrix} \begin{bmatrix} t \\ u \\ v \end{bmatrix} = O - V_0 \tag{2.9}$$

Quindi per determinare le coordinate baricentriche e la distanza dall'origine della retta al punto di intersezione è necessario risolvere il sistema lineare (2.9). Il sistema sopra può essere pensato geometricamente come traslazione del triangolo nell'origine, posta nel vertice $V_0$ del triangolo, e una trasformazione dello stesso nel triangolo unitario nel piano $y - z$ e con retta allineata all'asse $x$, come mostrato in (2.2) (dove $M = [-D,\ V_1 - V_0,\ V_2 - V_0$ è la matrice in (2.9)). Indicando con $E_1 = V_1 - V_0$, $E_2 = V_2 - V_0$ e $T = O - V_0$, la soluzione all'equazione (2.9) è ottenuta direttamente tramire inversione della matrice.

| Caso test | Scafo analitico | Scafo triangolato | Scafo triangolato ottimizzato |
|---|---|---|---|
| Moto di affondamento | $67s$ | $6h$ e $17m$ | $239s$ |
| Moto di beccheggio | $74s$ | n.d. | $317s$ |

**Tabella 2.1:** Tempi di simulazione per iterazione nel caso di simulazioni idrostatiche.

### 2.4.3 Sfruttare le ricerche già fatte

Come già detto l'algoritmo di ricerca verrà applicato per tutti gli elementi della mesh che discretizza il dominio $\Omega$ ad ogni iterazione temporale. É quindi naturale pensare di sfruttare le ricerche già fatte per velocizzare le future chiamate. Dato che l'insieme di punti su cui si testa è sempre lo stesso, per ogni punto si memorizza l'indice dell'elemento che lo conteneva all'iterazione precedente e, si usa questo indice come elemento da cui partire per ricercare il nuovo elemento nella successiva iterazione temporale. Il vettore che contiene gli indici dell'elemento trovato all'iterazione precedenti per ogni punto, si trova nella classe `Mesh`, in quanto come già esposto, sono questi i punti con cui chiamiamo il metodo `Locate`.
Se l'imbarcazione non si è eccessivamente mossa dalla posizione all'istante precedente, è molto probabile che partendo dall'elemento trovato nella configurazione precedente, in una o poche iterazioni si trovi l'elemento che contiene il punto nella nuova configurazione. Se poi ciò non si verifica, allora si andrà a costruire l'insieme $S$ come visto in precedenza. Una volta che ho trovato l'elemento che contiene il punto nella nuova iterazione, è plausibile assumere che tutti i punti vicini o siano contenuti nello stesso elemento o comunque ad elementi subito adiacenti, per questo aggiorno gli indici dei punti vicini con quello che ho appena trovato.
Questa serie di accortezze permettono di ottenere tempi di simulazione paragonabili a quelli ottenuti usando scafi analitici, come mostrato in 2.1. In questa tabella mostriamo il tempo medio di simulazione per iterazioni nel caso di simulazioni idrostatiche, confrontando le prestazioni delle simulazioni su uno scafo analitico con simulazioni con geometrie triangolate (scafo descritto da circa 20000 elementi e 9000 nodi) usando le tecniche di ottimizzazione implementate oppure con una ricerca senza nessuna ottimizzazione. Come evidenziato in 2.1, i tempi di simulazioni si riducono drasticamente, e addirittura nel caso di moto di beccheggio erano talmente lunghi che non si è potuto completare un'iterazione. Abbiamo inoltre osservato che l'introduzione di criteri necessari e sufficienti per la ricerca di un punto va a ridurre notevolmente le chiamate alla funzione `Locate`, passando da circa 1840000 per iterazione a 160000 per iterazione.

## 2.5 Il metodo interp

La seconda problematica incontrata nell'uso di scafi triangolati è il calcolo delle forze che agiscono su di essi. Infatti i valori della pressione $\lambda$ dovuta al vincolo di galleggiamento sono definiti sulla mesh bidimensionale del dominio $\Omega$. In particolare i valori di $\lambda$ sono costanti all'interno di ogni elemento di $\Omega_h$ e perciò si assumerà che il valore sia assunto nel baricentro dell'elemento. Dato quindi il vettore $\lambda_\Omega$ dei valore della pressione e $\mathbf{x}_\Omega$ che sono le coordinate dei punti,

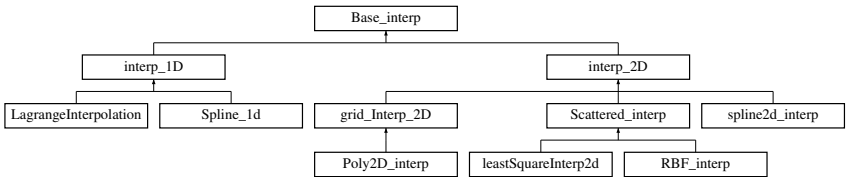



**Figura 2.3:** Legami tra le classi del metodo *interp*.

noi vogliamo conoscere il valore di $\lambda$ sui nodi che vanno a costituire lo scafo. Il problema a cui ci si riconduce è quindi un problema di *interpolazione*. Per questo motivo è stato implementato un metodo, denominato *interp*, che costruisce e valuta l'interpolante dato i valori della funzione in determinati punti.

Nella figura 2.3 si mostra come sono organizzate le principali classi all'interno del metodo *interp*. Vogliamo evidenziare come le classi `Base_interp`, `interp_1D`, `interp_2D`, `grid_Interp_2D` e `Scattered_interp` siano classi che forniscono un interfaccia comune a una particolare categoria di tecniche di interpolazione, implementandone i metodi comuni, lasciando quindi solo alle classi figlie il compito di implementare la particolare procedura di interpolazione. Inoltre una realizzazione di questo tipo, permette, nel caso di utilizzo di questo metodo integrato con un altro software (come succede con *Stratos++*) di poter usare diverse tecniche di interpolazione, con modifiche minime sul codice che utilizza le tecniche di *interp*. Come mostrato in figura 2.3, dalla classe `Base_interp` ereditano pubblicamente due classi: la classe `interp_1D` e la classe `interp_2D`. Nelle sezione successiva concentriamo la nostra attenzione sulle tecniche di interpolazione derivate dalla classe `interp_2D`.

### 2.5.1 La classe `interp_2D`

La classe `interp_2D` è una classe puramente virtuale che implementa alcune funzionalità comune a tutte le classi che interpolano in uno dominio bidimensionale. In particolare contiene le coordinate in cui si valuta l'interpolante, una *funzione2d* che contiene la funzione che ha generato i dati (se disponibile), e i metodi che calcolano l'errore e stampano i risultati tramite il software *Gnuplot*. Due sono le principali classi di interpolazione bidimensionale presentate:

1. la classe `Scattered_interp` nella quale vengono implementati i metodi di interpolazione per dati sparsi.

2. la classe `grid_Interp_2D` nella quale vengono implementati i metodi di interpolazione per dati organizzati in griglie.

Prima di introdurre la classe `grid_Interp_2D`, è necessario introdurre due ulteriori classi su cui si appoggia: la classe `Element` e la classe `Mesh`.

#### La classe *Element*

Questa classe è stata introdotta per definire un elemento di una griglia, realizzando metodi e strutture il più generali possibili, in modo da poter in seguito implementare particolari elementi (quelli introdotti sono elementi triangolari e rettangolari, anche se in linea teorica, è possibile implementare elementi con

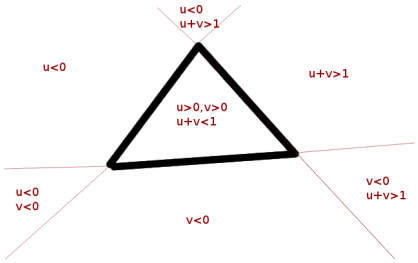



**Figura 2.4:** Coordinate baricentriche.

geometrie più complicate e volendo generalizzare anche al caso tridimensionale). Le principali strutture contenute nella classe sono le seguenti:

- `_vertice` un vettore che contiene le coordinate dei vertici dell'elemento;
- `_value` un vettore che contiene il valore della funzione nei nodi di interpolazione;
- `_nodeInterpolation` un vettore che contiene i nodi di interpolazione.

Lo scopo di questa classe è quella di fornire un'interfaccia comune per diversi elementi e quindi di far in modo da non dover reimplementare dall'inizio il metodo di interpolazione per ogni tipo di elemento. Da questa classe, come anticipato, derivano pubblicamente la classe `Triangle` e la classe `Rectangle`.
I metodi principali sono:

- `isInside` che testa se un punto è interno o esterno all'elemento.
  Nel caso di un elemento rettangolare, questo test è molto semplice, in quanto basta verificare che le coordinate del punto rispettano $V_x^1 \leq x \leq V_x^2$ e $V_y^1 \leq y \leq V_y^2$, dove $V^1$ e $V^2$ sono due vertici opposti del rettangolo.
  Se invece l'elemento è triangolare, si riscrive il punto in coordinate baricentriche, cioè $P = V^1 + u(V^3 - V^1) + v(V^2 - V^1)$ e se il punto, come mostrato in 2.4, verifica $u \geq 0\, v \geq 0\, u + v \leq 1$ è interno al triangolo.
- `setValue` che dato il vettore con i valori della funzione assegna i valori ai giusti nodi di interpolazione.
- `setNodeInterpolation` che, in base al grado del polinomio, va a calcolare i nodi di interpolazione dell'elemento. In particolare per elementi rettangolare, dato il grado del polinomio in direzione $x$ e direzione $y$, è possibile calcolare i nodi di interpolazione per ogni ordine, mentre nel caso di elementi triangolari, si possono ricavare i nodi di interpolazione fino al second'ordine.

### La classe `MeshInt`

Una volta implementati i singoli elementi, si è potuto introdurre una classe per contenere la mesh e fornire dei metodi veloci per localizzare l'elemento contenente il punto su cui si vuole interpolare. Da questa classe deriva pubblicamente la classe `Surface3d` già introdotta, le cui strutture e i metodi principali sono gli stessi ma riadattati in parte al caso di elementi bidimensionali.
In questa classe esistono due costruttori:

```
Mesh(const char * filenameCoordinate);
Mesh(const std::vector<coupleD> & nodi);
```

uno che necessita di un vettori di caratteri che corrisponde al nome del file dove si troverà la mesh e un'altra che invece necessita solo di un vettore di nodi. La differenza è che in questo caso un vettore di nodi non è sufficiente per definire una mesh, e quello che fa il costruttore è creare una triangolazione di Delaunay appoggiandosi sulla libreria *Triangle++*[3], integrata all'interno del codice *Stratos++*, che costituisce un wrapper C++ del software *Triangle.*[4]. I formati di mesh supportati sono:

- **.msh**: Il formato di file MSH ASCII, già introdotto per la classe `Surface3d`.
- **.tri**: Questo formato corrisponde ai file generati dal software *Triangle*. In particolare, questo software va a generare quattro file: il file **.node** che riporta il numero totale di nodi e le loro cordinate; il file **.ele** che riporta il numero totale di elementi, oltre al numero di vertici per elemento; il file **.poly** che contiene informazioni sui lati di bordo; il file **.neigh** che contiene la struttura di adiacenza degli elementi. Affinchè il costruttore funzioni, è necessario che il nome del file in ingresso sia del tipo *nome.tri* e che i file generati da *Triangle* abbiano la seguente denominazione: *nome.tri.poly*, *nome.tri.ele*, *nome.tri.node*, *nome.tri.poly* ;
- **.off**: sono i formati di file prodotti per esempio dal software *Geomview*. Le informazioni contenute sono analoghe a quelle nel file di formato **.msh**, cioè inizialmente un elenco delle coordinate dei nodi e successivamente l'elenco degli indici dei nodi che vanno a costruire ogni singolo elemento ;

Nel costruttore, in funzione del formato della mesh, si chiama il corrispondente metodo `buildXXXmesh` che va a inizializzare alcune strutture utili per l'interfaccia con la mesh. Le strutture che possono essere inizializzate in questo metodo dipendono ovviamente dalle diverse informazioni contenute nei vari formati. Dopo di che si inizializzano le restanti strutture nel metodo `setMeshParameter`. Questo metodo, assieme a `buildXXXmesh`, è privato in quanto deve essere chiamato solo dal costruttore e non è disponibile all'utente.
All'interno di `setMeshParameter` si chiamano tre membri della classe:

- `setElements`: che utilizzando le informazioni contenute in `_Nodes` e in `_node2elem` per costruire gli elementi della mesh;
- `setObjectForFindElement`: che va a inizializzare tutte le strutture necessarie per velocizzare la ricerca dell'elemento, analoghe a quelle già introdotte per la classe `Surface3d`;
- `setElement2element`: che va a inizializzare, nel caso non sia già stato fatto in `buildXXXmesh`, `_element2element`.

[3]Per ulteriori informazioni si veda `http://www.compgeom.com/~piyush/scripts/triangle/`

[4]Triangle genera triangolazione di Delaunay esatte, vincolate, conformi, diagrammi di Voronoi e mesh triangolari di alta qualità. Per ulteriori informazioni si visiti il sito `http://www.cs.cmu.edu/~quake/triangle.html`

Metodo fondamentale della classe `MeshInt` è `Locate`, le cui funzionalità sono le stesse del metodo della classe `Surface3d`, l'unica differenza è che in questa situazione la mesh non descrive più una superficie immersa nello spazio tridimensionale, ma è invece un dominio bidimensionale, e di conseguenza non bisogna più determinare il punto di intersezione tra una retta e un triangolo, ma testare direttamente se il punto è interno al triangolo, usando il metodo `isInside` della classe `Element`.

### La classe `grid_Interp_2D`

La classe `grid_Interp_2D`, oltre a un oggetto di tipo `MeshInt`, contiene:

- `_coordinateIndex`: un vettore la cui dimensione corrisponde al numero di punti su cui si vuole interpolare, e nella posizione $i$ contiene l'indice dell'elemento che contiene lo $i$-esimo punto. Questo vettore risulta essere molto utile nel caso in cui il metodo `interpolate` venga chiamato più volte sullo stesso insieme di punti. Quando `interpolate` viene chiamato per la prima volta, il vettore è inizializzato al valore $-1$, mentre durante le iterazioni successive utilizza il valore dell'indice dell'elemento trovato nell'iterazione precedente come punto di partenza per la ricerca. Nel caso invece l'insieme dei punti su cui si interpoli cambi, è possibile resettare questo vettore attraverso il metodo `reseteCoordinateIndex`.

- `_numberElement` e `_numberNode` che mi danno il numero di elementi e di nodi della griglia.

- `_LocateElement`: un puntatore a un oggetto di tipo `element` che corrisponde all'elemento localizzato per il punto corrente.

- `_isTriangle`: una variabile booleana che mi dice se la griglia è rettangolare o triangolare.

Esistono tre costruttori di questa classe ed essi si limitano a chiamare il costruttore della classe `interp_2D` e quello della classe `Mesh`. Dopo di che vanno a impostare i valori di `_value` usando la funzione analitica, se disponibile, o altrimenti, nel primo caso leggendo i valori da un file, nel secondo leggendo i valori da un vettore.
Nel terzo caso invece sia i valori sia le coordinate vengono lette da due vettori e le coordinate vengono passate all'oggetto di tipo `Mesh`, che come già visto genera una triangolazione di Delaunay. In questo caso si suppone che i valori della funzione siano noti solo nei vertici dei triangoli, e quindi l'ordine di interpolazione è fissato a 1.
Un altro metodo importante in questa classe è il già citato metodo `interpolate`. Questo metodo riceve in ingresso un punto e l'indice dell'elemento che all'iterazione precedente conteneva quel punto (contiene $-1$ se siamo alla prima ricerca) e chiama semplicemente il metodo `locate` di `_grid`. Nel caso in cui il punto a essere testato sia il primo punto che viene dato all'interpolante, dopo aver trovato l'elemento che contiene il primo punto, testando di quanti vertici è costituito l'elemento, riusciamo a impostare in modo corretto la variabile booleana `_isTriangle`.A questo punto se si vuole implementare una tecnica di interpolazione su griglia triangolare o rettangolare è necessario implementare solo un metodo `rawInterp`.

```
class Poly2D_interp : public grid_Interp_2D {
private:
        double rawInterpRectangle(coupleD punto);
        double rawInterpTriangle(coupleD punto);
public:
        Poly2D_interp(char const * filenameCoordinate,char
            const *filenameValue,int gradoPolinomioX,int
            gradoPolinomioY, funzione2d funzione=NULL):
            grid_Interp_2D(filenameCoordinate,filenameValue,
            gradoPolinomioX,gradoPolinomioY,funzione){};
        Poly2D_interp(char const * filenameCoordinate,std::
            vector<double>& value,int gradoPolinomioX,int
            gradoPolinomioY, funzione2d funzione=NULL):
            grid_Interp_2D(filenameCoordinate,value,
            gradoPolinomioX,gradoPolinomioY,funzione){};
        Poly2D_interp(std::vector<coupleD>& coordinate,std::
            vector<double>& value, funzione2d funzione=NULL):
            grid_Interp_2D(coordinate,value,funzione){};

        double rawInterp(coupleD punto);

};
```

Questo è quello che succede per la classe `Poly_2d`, che implementa la interpolazione polinomiale su griglia. Come si può osservare

```
double Poly2D_interp::rawInterp(coupleD punto){
    if(_isTriangle)
        return rawInterpTriangle(punto);
    else
        return rawInterpRectangle(punto);

};
```

il metodo `rawInterp` non fa altro che chiamare, nel caso di griglia triangolare, il metodo `rawInterpTriangle`, altrimenti chiama il metodo `rawInterpRectangle`. Si è scelto di rendere privati questi due membri in quanto non abbiamo nessun interesse che l'utente li conosca e li possa usare direttamente. Nel caso di elemento triangolare, sono state implementate le formule di interpolazione fino al secondo ordine. Da notare che in questo caso non ha senso impostare diversi gradi di polinomio per le due direzioni $x$ e $y$ e si considera solo il grado in direzione $x$. Nel caso in cui questi due ordini fossero diversi, allora si segnala questa incongruenza con un messaggio a video. Nel caso invece di elementi rettangolari, è possibile un'interpolazione di qualunque ordine, in quanto ci si riconduce a un interpolazione monodimensionale che si risolve attraverso la classe `LagrangeInterpolation`.
La formula base per interpolazione nell'elemento triangolare è derivata come segue. Sia $\lambda^h$ l'approssimazione lineare per $\lambda$, secondo la formula:

$$\lambda^h(x, y) = a + bx + cy \tag{2.10}$$

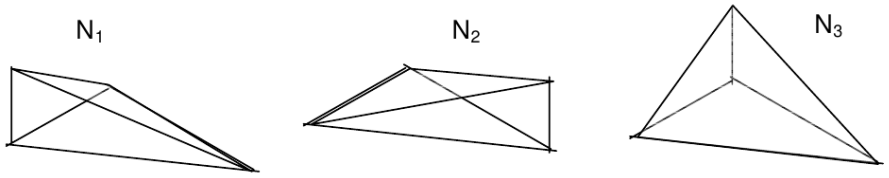



**Figura 2.5:** Funzione di base usate nell'interpolazione lineare su mesh triangolata

Imponiamo ora che

$$\begin{aligned} \lambda^h(x_1, y_1) &= u_1 = a + bx_1 + cy_1 \\ \lambda^h(x_2, y_2) &= u_2 = a + bx_2 + cy_2 \\ \lambda^h(x_3, y_3) &= u_3 = a + bx_3 + cy_3 \end{aligned} \tag{2.11}$$

Risolvendo le tre equazioni per $a$, $b$ e $c$, allora sostituendo nell'equazione originale troviamo:

$$\lambda^h = N_1(x, y)u_1 + N_2(x, y)u_2 + N_3(x, y)u_3 \tag{2.12}$$

dove

$$\begin{aligned} N_1(x, y) &= \tfrac{1}{2A}[(x_2y_3 - x_3y_2) + (y_2 - y_3)x + (x_3 - x_2)y] \\ N_2(x, y) &= \tfrac{1}{2A}[(x_3y_1 - x_1y_3) + (y_3 - y_1)x + (x_1 - x_3)y] \\ N_3(x, y) &= \tfrac{1}{2A}[(x_1y_2 - x_2y_1) + (y_1 - y_2)x + (x_2 - x_1)y] \end{aligned} \tag{2.13}$$

In 2.2 rappresentiamo i grafici di queste funzioni, dette *funzioni di forma*. Osserviamo che

- ogni funzione forma ha valore 1 allo $i$-esimo nodo del triangolo e zero agli altri nodi;
- lo schema di interpolazione corrisponderà a quello di elementi adiacenti dato che essi condividono 2 nodi in comune e la funzione varia linearmente tra loro.
- le funzioni di forma verificano $N_1(x, y) + N_2(x, y) + N_3(x, y) = 1$.

`Scattered_interp`

Questa è la classe madre di tutti quei metodi che interpolano su insieme di punti che non sono ordinati in una struttura tipo griglia. Gli oggetti contenuti sono solo un vettore per i punti e un vettore per i valori della funzione. Anche in questo caso abbiamo tre costruttori, per affrontare le situazioni nelle quali le coordinate dei punti siano in un vettore o all'interno di un file e se i valori della funzione siano in un file, in un vettore, o possano essere generati nel caso si abbia la funzione analitica. Da questa classe ereditano due classi che implementano due tecniche di interpolazione: la classe `RBF_interp` e la classe `leastSquareInterp2d`. Concentriamo la nostra attenzione sulla seconda tecnica di interpolazione.

**leastSquareInterp2d** Una seconda approssimazione possibile è attraverso il metodo di interpolazione lineare ai minimi quadrati generalizzato (*General Linear Least Square*), una tecnica che, dato un insieme di punti $(\mathbf{x}, f_i)$, approssima la funzione $\mathbf{f}(x)$ come combinazione lineare di $M$ funzioni base di $\mathbf{x}$. Nel caso implemenentato in *Interp*, si è scelto come base i polinomi, e quindi la funzione interpolante può essere così scritta:

$$\tilde{f}(\mathbf{x}) = a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + \cdots + a_{M-2} x^{M-1} + a_{M-1} y^{M-1} \quad (2.14)$$

e quindi $\tilde{f}$ è un polinomio di grado $M-1$. La forma generale per questi tipi di modello è

$$\tilde{f}(\mathbf{x}) = \sum_{k=0}^{M-1} a_k X_k(\mathbf{x}) \quad (2.15)$$

dove le quantità $X_0(x), \cdots, X_{M-1}$ sono funzioni arbitrarie e fissate di $x$, chiamate *funzioni di base*. Si osservi come le funzioni di base possono essere funzioni ampiamente non lineari di $\mathbf{x}$, infatti in questo caso il termine lineare si riferisce alla dipendenza lineare del modello rispetto ai parametri $a_k$. Per questo modello lineare definiamo la funzione di merito:

$$\chi^2 = \sum_{i=0}^{N-1} [y_i - \sum_{k=0}^{M-1} a_k X_k(\mathbf{x}_i)] \quad (2.16)$$

Cerchiamo quindi l'insieme dei parametri $\{\tilde{a}_i\}$ che minimizzano la funzione di merito. Per poter risolvere questo problema, introduciamo prima alcune definizioni utili per la trattazione. Sia $\mathbf{A}$ la matrice $N \times M$ costruita tramite le $M$ funzioni di base e valutate sugli $N$ punti $\mathbf{x}_i$, cioè

$$A_{i,j} = X_j(\mathbf{x}_i) \quad (2.17)$$

La matrice $\mathbf{A}$ è denominata *matrice di design* del problema di interpolazione. Si osserva come, solitamente, $\mathbf{A}$ abbia più righe che colonne, cioè $N \geq M$. Definiamo inoltre il vettore $\mathbf{b}$ di lunghezza $N$ come

$$b_i = y_i \quad (2.18)$$

e con $\mathbf{a}$ il vettore dei parametri di modello. Il minimo di **??** si ha quando la derivata di $\chi^2$ rispetto agli $M$ parametri $a_k$ si annulla. Si giunge quindi a dover risolvere un sistema lineare:

$$(\mathbf{A}^T \cdot \mathbf{A}) \cdot \mathbf{a} = \mathbf{A}^T \cdot \mathbf{b} \quad (2.19)$$

Questo sistema di equazioni è anche detto sistema di *equazioni normali*. Il sistema è non singolare se la matrice $\mathbf{A}$ ha *rango pieno*, e, se questa condizione è verificata, esiste ed è unica la soluzione del problema di minimizzazione. Osserviamo che la matrice $\mathbf{B} = \mathbf{A}^T\mathbf{A}$ è simmetrica e definita positiva, quindi per risolvere (2.19) si potrebbe calcolare una fattorizzazione di Cholesky di $\mathbf{B} = \mathbf{H}^T\mathbf{H}$ e poi risolvere i due sistemi $\mathbf{H}^T\mathbf{y} = \mathbf{A}^T\mathbf{b}$ e $\mathbf{Ha} = \mathbf{y}$. Ciò nonostante, a causa di errori di roundoff, il calcolo di $\mathbf{B}$ può essere influenzato da perdità di cifre significative, con conseguente perdita di definita positività o non singolarità della matrice. Se poi la matrice $\mathbf{A}$ fosse addiritura non a rango pieno, si perderebbe l'unicità della soluzione, in quanto se $\mathbf{a}$ è soluzione di (2.19), allora il

vettore $\mathbf{a}+\mathbf{z}$, con $\mathbf{z} \in \ker(\mathbf{A})$ è soluzione anch'essa. Dobbiamo quindi introdurre un ulteriore vincolo per forzare l'unicità della soluzione. Tipicamente, come implementato nel codice, si richiede che $\mathbf{a}$ abbia norma euclidea minima, cosicchè il problema di minimizzazione possa essere riformulato come segue

$$\begin{aligned} &\text{trova } \mathbf{a} \in \mathcal{R}^n \text{ con norma euclidea minima tale che} \\ &\|\mathbf{A}\mathbf{a}-\mathbf{b}\|_2^2 \leq \min_{\mathbf{x}\in\mathcal{R}^n} \|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 \end{aligned} \tag{2.20}$$

Il problema è consistente con (2.19) se $\mathbf{A}$ ha rango pieno, dato che in questo caso il problema ha un'unica soluzione che necessariamente ha norma minima. Lo strumento per risolvere (2.20) è la *decomposizione ai valori singolari* (SVD). Se con $\mathbf{U}$ e $\mathbf{V}$ indichiamo le matrici di SVD di $\mathbf{A}$, e poniamo $\mathbf{U}_{(i)}$ $i = 0, \cdots, M-1$ le colonne di $\mathbf{U}$ e poniamo $\mathbf{V}_{(i)}$ $i = 0, \cdots, M-1$ le colonne di $\mathbf{V}$, allora la soluzione di (2.20) può essere scritta come

$$\mathbf{a} = \sum_{i=0}^{M-1} \left( \frac{\mathbf{U}_{(i)} \cdot \mathbf{b}}{w_i} \right) \mathbf{V}_{(i)} \tag{2.21}$$

dove $w_i$ sono i valori singolari calcolati da SVD.

### 2.5.2 Strategia scelta

Per come è stato implementato la classe `interp_2D`, l'utente può scegliere quale dei due metodi di interpolazione utilizzare. Tuttavia la scelta fatta per le simulazioni è stato quello di usare un interpolazione lineare su griglia triangolare. Questa scelta è motivata dal fatto che usando un interpolazione di questo tipo, dato che la mesh è costituita da circa 20000 elementi, si ottiene un'approssimazione della funzione sufficientemente accurata. Un possibile svantaggio di questo tipo di approccio sta nella ricerca, ad ogni iterazione temporale, del triangolo contenente il punto su cui si vuole interpolare. Tuttavia, sfruttando il riferimento solidale all'imbarcazione, non è necessario ricreare ogni volta la griglia su cui valutare l'interpolante, che verrà invece creata solo alla prima iterazione temporale e aggiornerà solo il valore della funzione ad ogni iterazione successiva. Inoltre quando l'angolo di beccheggio e di rollio non varia eccessivamente, lo $i$-esimo punto su cui si vuole interpolare sarà spesso contenuto nello stesso elemento dell'istante precedente, o comunque molto vicino, e quindi memorizzando l'indice del triangolo che contiene il punto all'istante precedente si può velocizzare di molto la ricerca.

## 2.6 La classe `Geometry`

Nella classe `Geometry`, già implementata nella vecchia versione di *Stratos++*, si sono dovuti modificare i seguenti metodi:

- `buildActiveElem`;
- `buildLambdaOnNodes`;
- `computeForceAndMoment`.

Nel caso di geometria analitica, il metodo `buildActiveElem` è quello già implementato nella versione precedente, che individuava gli elementi attivi su $\Omega_h$ come quei elementi tali che

- i valori del moltiplicatore di Lagrange nel baricentro dell'elemento sia diverso da zero;
- ogni nodo dell'elemento cadano sulla superficie dello scafo.

Nel caso di scafi triangolati, si sono considerati invece tutti gli elementi tali che il baricentro $(\overline{x}_i, \overline{y}_i)$ verifichi

$$\begin{aligned} |\overline{x}_i - x_b| &< c_1 L \\ |\overline{y}_i - y_b| &< c_2 W \end{aligned} \tag{2.22}$$

dove con $(x_b, y_b)$ si indicano le coordinata del centro di massa dell'imbarcazione, con $L$ e $W$ rispettivamente la lunghezza e la larghezza della imbarcazione e con $c_1$ e $c_2$ delle costanti, che mi garantiscano che il dominio rettangolare $\mathcal{R}$ definito dall'insieme di punti che verificano (2.22) contengano la proiezione della barca su $\Omega$ nella configurazione rototraslata.
Una scelta di questo tipo è dettata dalla particolare tecnica di interpolazione che si è scelta per calcolare i valori di $\lambda$ sui nodi di $\mathcal{B}_h$, cioè di triangolare i baricentri degli elementi attivi individuati in `buildActiveElem` con una triangolazione di Delaunay, implementata in *Triangle++*, e interpolando in modo lineare all'interno di ogni triangolo. Infatti in generale una triangolazione di Delaunay può generare dei triangoli molto allungati nei punti più esterni rispetto all'insieme di nodi sui quali si vuole triangolare, e quindi scegliere come nodi per creare la triangolazione i soli nodi con valori di $\lambda$ non nulli portava a errori nel calcolo dell'interpolante. Per questo si è scelto di selezionare come nodi attivi tutti quelli contenuti in un dominio rettangolare che conterrà sicuramente l'imbarcazione nella configurazione rototraslata e che, grazie a una scelta oculata dei parametri $c_1$ e $c_2$, ci permetterà di evitare di valutare l'interpolante nei triangoli di bordo. Inoltre, grazie al particolare sistema di riferimento solidale con l'imbarcazione, l'insieme dei baricentri degli elementi di $\Omega_h$ che verificano (2.22) saranno sempre gli stessi per tutte le iterazioni temporali, e ciò permette di costruire la griglia di nodi $\mathcal{G}_h$ e l'interpolante associato soltanto alla prima iterazione, e di tener memoria dell'indice dell'elemento che conteneva lo $i$-esimo punto all'istante precedente.
La costruzione dell'interpolante, chiamata solo alla prima iterazione temporale, si ha nel metodo `buildLambdaOnNodes`, che si occupa anche di aggiornare il valore di $\lambda$ nelle successive iterazioni temporali.
Nella vecchia versione di `buildLambdaOnNodes`, utilizzata nel caso di scafi analitici, i nodi del dominio $\Omega_h$ corrispondono con i nodi dello scafo analitico, e quindi il valore di $\lambda$ su quest'ultimi è calcolato come media pesata dei valori di $\lambda$ sugli elementi che hanno quel nodo in comune. Presentiamo qui la parte di codice di questo metodo relativa a geometrie triangolate.

```
vector<double> lambdaOnElem(hull->getNumberTriangle(),-10);
for (int i=0;i<hull->getNumberTriangle();i++){
tripletD punto((hull->getTriangle(i)).getBaricentro());
rotateAndMovePoint(punto);
coupleD punto2D(punto.x(),punto.y());
```

```
vector<int> adiacent(hull->getElem2Elem(i));
int check(0);
for(int j=0;j<3;j++)
  check+=(abs(lambdaOnElem[adiacent[j]])<numeric_limits<double
      >::epsilon());
if(check!=3){
  lambdaOnElem[i]=_interpolante->interpolate(punto2D,
      _coordinateIndex[i]);
  for(int j=0;j<3;j++){
    if(adiacent[j]>=0)
      _coordinateIndex[adiacent[j]]=_coordinateIndex[i];
  }
  if(lambdaOnElem[i]>numeric_limits<double>::epsilon()){
    tripletI pointIndex(hull->getElem2Nodes(i));
    vector<double> tempLambda(3,0);
    for(int j=0;j<3;j++){
      tripletD point(hull->getNode(pointIndex[j]));
      rotateAndMovePoint(point);
      tempLambda[j]=_interpolante->interpolate(coupleD(point.x
          (),point.y()),_coordinateIndex[i]);
    }
    _activeElemBoat.push_back(i);
    copy(tempLambda.begin(),tempLambda.end(),back_inserter(
        _lambdaOnNodes));
  }
  }else lambdaOnElem[i]=0;
}
```

Si può osservare come in questo non ci si limiti a chiamare l'interpolante per ogni nodo di $\mathcal{B}_h$, ma invece si utilizza una strategia diversa, per ridurre il numero di ricerche. Infatti, come si vede nel codice presentato, andiamo a scorrere tutti gli elementi di $\mathcal{B}_h$ (e non i nodi) e per prima cosa verifichiamo se sull'elemento corrente si possa considerare attivo il moltiplicatore $\lambda$. Per fare ciò andiamo a controllare il valore di $\lambda$ sui baricentri degli elementi adiacenti e, se si è già valutato il valore del moltiplicatore sul baricentro di quei triangoli e risulta che almeno un elemento abbia un valore diverso da zero, in quel caso si procede a valutare l'interpolante nel baricentro dell'elemento corrente. Questo controllo permette di diminuire il numero di valutazione dell'interpolante. Se l'elemento verifica questa condizione, andiamo a valutare l'interpolante nel baricentro, utilizzando come elemento di $\mathcal{G}_h$ dal quale partire nella ricerca uno degli elementi che conteneva il baricentro di uno dei triangoli adiacenti, se sono già stati valutati, altrimenti usando la procedura di ricerca già introdotta per il metodo `Locate` della classe `surface3d` e memorizzando l'indice del triangolo, appartente a $\mathcal{G}_h$, che contiene tale punto. Se il valore dell'interpolante in tale punto risulta diverso da zero, lo andremo a valutare nei vertici di questo triangolo usando come elemento di partenza da cui ricercare in $\mathcal{G}_h$ quello che conteneva il baricentro del triangolo corrente, permettendo quindi di accelerare il calcolo dell'interpolante nei vertici. In questo modo si riesce a migliorare di molto le prestazioni limitando il numero di valutazione dell'interpolante.
Si osservi inoltre, come gli indici degli elementi attivi su $\mathcal{B}_h$ sia memorizzato in un vettore `_activeElemBoat`, che verrà poi usato nella versione modificata di `computeForceAndMoment`, che userà questo indice per determinare i vertici

dell'elemento attivo e ricavare quindi il valore della pressione dovuto al vincolo di galleggiamento sul singolo elemento, che porterà al calcolo della forza e del momento sulla struttura.

## 2.7 Le classi `interfaceWithKime` e `coupledManagement`

In queste due classi vengono implementate tutte i metodi per l'interfaccia con il software Kimè e per l'accoppiamento tra il modello fluidodinamico risolto da *Stratos++* e il modello della struttura risolto dal software *Kimè*. La classe `interfaceWithKime`, oltre ai metodi `Get` e `Set` per impostare i valori da passare a *Kimè* e per ottenerne i valori calcolati implementa tre metodi che sono i principali di questa classe:

- `saveValue`: metodo che va a scrivere sul file di input usato da Kimè.
- `readOutput`: metodo che legge i valori calcolati da Kimè dal file di output.
- `launchKime`: metodo che attraverso una chiamata alla routine *system*, lancia da stratos++ il software Kimè.

Un oggetto di tipo `interfaceWithKime` è contenuto nella classe `coupledManagement`, che si occupa di gestire l'accoppiamento tra i due modelli e in particolare implementa il metodo CSS con predittore strutturale del primo ordine. Questa classe ha quindi il compito di calcolare e resistuire al risolutore *Stratos++* la posizione della struttura predetta rispetto alla velocità e alla posizione all'istante precedente. Inoltre, dato che il passo temporale associato al risolutore fluidodinamico sarà minore rispetto a quello di *Kimè*, va a calcolare le forze input per il risolutore alla struttura come una media integrale in tempo rispetto ai sottocicli di risoluzione del problema fluidodinamico.
Come predittore della posizione della struttura, si è scelto di usare un predittore del primo ordine, dato dalla formula

$$\tilde{\mathbf{u}}^{n+1} = \mathbf{u}^n + \Delta t_s \dot{\mathbf{u}} \tag{2.23}$$

Infine all'interno della classe `Problem` è stato necessario implementare un nuovo metodo, detto `solveWithKime`, che permettesse di gestire i diversi passi temporali usati per la risoluzione dei due problemi accoppiati e che si appoggierà sul metodo di `Geometry` detto `computeDynamicsKime`, che a sua volta chiamerà il metodo `fluid2structure` di `coupledManagement` per assegnare a *Kimè* la corretta forza fluidodinamica.

## 2.8 La funzione main e i metodi per il post-processing

La funzione main si occupa di leggere tutti i parametri forniti a *Stratos++* attraverso un file di input, istanziare un oggetto di classe `Problem` e chiamare il metodo che permette di risolvere numericamente le equazioni del modello matematico prescelto. Il file di configurazione è importato attraverso il parser *GetPot*, una libreria header-only molto leggera e decisamente funzionale. *GetPot* permette di costruire un file di input strutturato, nel quale è anche possibile inserire dei commenti, in modo da rendere il suo utilizzo il più intuitivo possibile. Inoltre, a ciascuno dei parametri è stato anche assegnato un valore di default. Qui di seguito si riporta un esempio di file di configurazione:

```
# ------------------------------------------------------------
# DATAFILE for STRATOS++
#
# Created by Simone Palamara on 22/11/11.
# Copyright 2011 Politecnico di Milano. All rights reserved.
# ------------------------------------------------------------


# Characteristics of the computational grid
meshFile  =./meshDir/etaImposed.tri
referenceHeight  =2.0
numberOfLayers  =24
layersDescription  =0.0 0.25 0.5 0.75 1.0 1.5 1.75 1.8 1.85 1.9 1.925
 1.95 1.975 2.01 2.025 2.05 2.075 2.1 2.15 2.2 2.25
 2.5 3.0 3.5 4.0

# Characteristics of the rowing scull
shellType  =Triangulate # WigleyHull, Elliptic or Triangulate
meshBoat =MeshStampo25.neu # File contains boat's mesh
mass  =435.6 # Canoe mass in kilograms
dimensions  =8.1 0.296 0.3 # Length, width and height
inertials  =515.25 515.25 515.25 # Inertial moments
initialPosition  =6.3302 0.0 2.25 # Initial position of the center of mass
initialVelocity  =5.3 # Initial horizontal velocity
initialAcceleration  =0.0 # Initial horizontal acceleration
initialPitching  =0.0  # Initial pitching angle
initialRolling  =0.0 # Initial rolling angle

# Characteristics of the problem
referenceFrame  =NonInertial # Inertial or NonInertial
problemType  =HydroDynamicWithKime # HydroStatic, HydroDynamic or HydroDynamicWithKime
chorinTemam  =NonIncremental # Incremental or NonIncremental
timeStep  =0.05
totalSteps  =100
lagrangian  =3
uzawaParameter  =9.80665
chezyNumber  =20.0

# Additional informations
windVelocity  =0.0 0.0
outputDir  =./outputDir/
startTime  = 0.0

# Information for Kime
fileInputKime  =./kimeDir/testcompleto.kim
deltaTkime  = 0.05
```

Nella prima parte, quella dedicata alla struttura della griglia di calcolo, deve essere indicato il file checontiene la geometria della mesh, l'altezza della superficie libera rispetto al fondo del canale e la descrizione degli strati posti lungo la direzione verticale. Nella seconda parte, invece, vanno indicate tutte le informazioni relative all'imbarcazione. Si parte dalla forma prescelta, per poi indicare la sua massa, compresa dei vogatori, e le sue dimensioni caratteristiche. Inoltre, devono essere segnalati i momenti di inerzia, la posizione iniziale del suo baricentro e le condizioni iniziali per la dinamica. Infine, si procede alla descrizione del problema fluidodinamico e dei parametri che il metodo numerico deve ricevere in ingresso. Tra questi, è necessario stabilire se si intende risolvere il problema nel riferimento inerziale oppure non inerziale, nella sua versione idrostatica, non idrostatica o non idrostatica accoppiata con *Kimè* e avvalendosi dello schema di Chorin-Temam incrementale oppure no. Inoltre, si indica il passo di discretizzazione temporale per *Stratos++* e *Kimè* e il numero totale di passi, insieme al numero di quelli che verranno utilizzati per risalire la caratteristica. Per ultimi sono definiti il parametro di accelerazione per il metodo di Uzawa e il coefficiente di Chezy, utilizzato per la simulazione dello strato limite sul fondo del canale. Le ultime informazioni riguardano la velocità del vento in superficie, che nelle simulazioni è sempre stata trascurata, oltre alla posizione in cui verranno memorizzate i risultati delle simulazioni. Infatti all'interno delle classi `Elevation` e `Geometry` si dispone del metodo `printVtk`, che consente di generare gli output nel formato vtk che verrà poi processato attraverso il software *Paraview*. In particolare si ha la possibilià di processare il campo di elevazione, la posizione dell'imbarcazione e il campo di pressione associato al vincolo di galleggiamento. Al momento del post-processing, poi, tutti questi file possono anche essere utilizzati contemporaneamente, in modo da ottenere una rappresentazione globale del sistema composto dal fluido e dall'imbarcazione.

# Capitolo 3

# Casi test

In questo capitolo andremo a presentare tre diverse simulazioni con le quali si è voluto testare il codice *Stratos++V.2*. Nel paragrafo 3.1 si validerà il risolutore fluidodinamico implementato nel codice confrontando i risultati delle simulazioni con alcuni dati sperimentali; in 3.2 si confronterà i risultati delle simulazioni usando uno scafo descritto da una funzione o triangolato; infine in 3.3 si analizzeranno i risultati delle simulazioni su un'imbarcazione di canottaggio a quattro posti, simulando la dinamica dello scafo attraverso il codice *Kimè*.

In tutti i casi test qui presentati lo scafo è posizionato all'interno di un canale lungo 35 $m$, largo 24 m e profondo 2 $m$. Sui lati di *inflow* e *outflow* si è imposta un'elevazione fissata pari allàltezza di rifermento, mentre sui bordi laterali, che corrispondono alle sponde del canale, si è scelto di adottare delle condizioni non riflettenti. L'estensione del dominio è piuttosto grande, in modo da poter evitare gli effetti di bordo. La triangolazione costruita sul dominio bidimensionale è composta da circa 175000 elementi, ed è stata raffinata in modo da avere molti più triangoli nella zona in cui agisce l'impronta di pressione dello scafo, come mostrato in 3.1.

## 3.1 Validazione del risolutore fluidodinamico

Per poter validare il risolutore fluidodinamico implementato in *Stratos++* ci si è confrontati con alcuni dati sperimentali ottenuti dall'Università di Tokyo in collaborazione con l'Instituto di Ricerca Navale e riportati in [J.H85]. I dati sperimentali riguardano i profili d'onda sulla superficie di un Wigley-Hull con moto a velocità costante e assetto fisso. La funzione analitica che definisce lo scafo è la seguente:

$$z = H\sqrt{1 - \frac{\frac{2y}{W}}{1 - (\frac{2x}{L})^2}} \tag{3.1}$$

dove $H = 0.15625$, $L = 2.5$ e $W = 0.25$. I dati sperimentali sono stati raccolti per differenti numeri di Froude (si veda 3.2). Per raggiungere la velocità di riferimento $\mathbf{V}_\infty$ la canoa parte da fermo e raggiunge $\mathbf{V}_\infty$ all'istante $t = 2s$. I valori del profilo d'onda sono stati presi all'istante $t = 7s$ in modo che sia esaurita la dinamica relativa al moto di accelerazione. Nelle figure 3.2, 3.3 e 3.4 confrontiamo l'andamento del profilo d'onda delle simulazione con quello

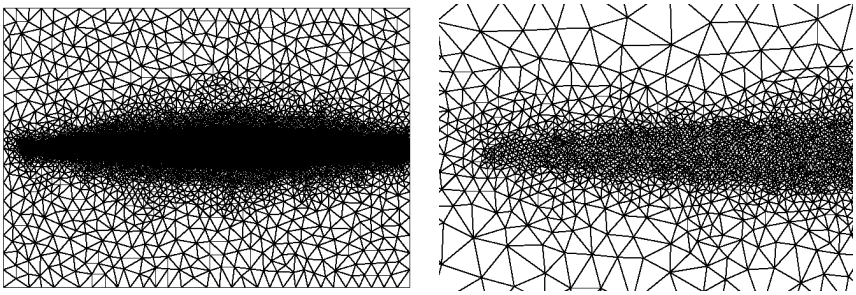



**Figura 3.1:** Mesh usate nelle simulazione che discrettizza il dominio bidimensionale $\Omega$.

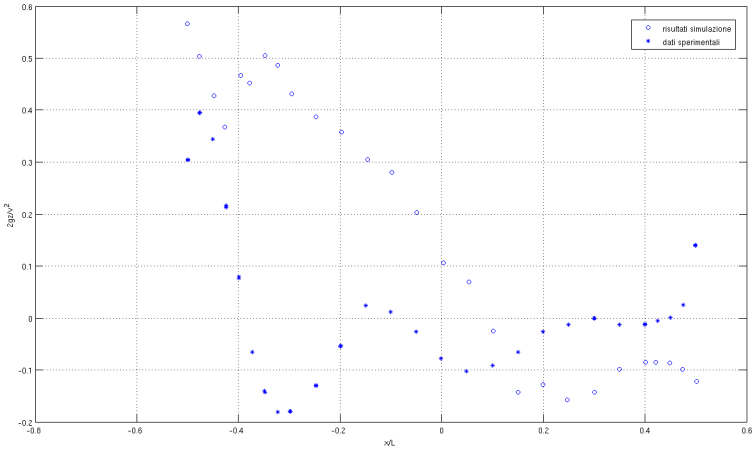



**Figura 3.2:** Profilo d'onda con $Fr = 0.25$.

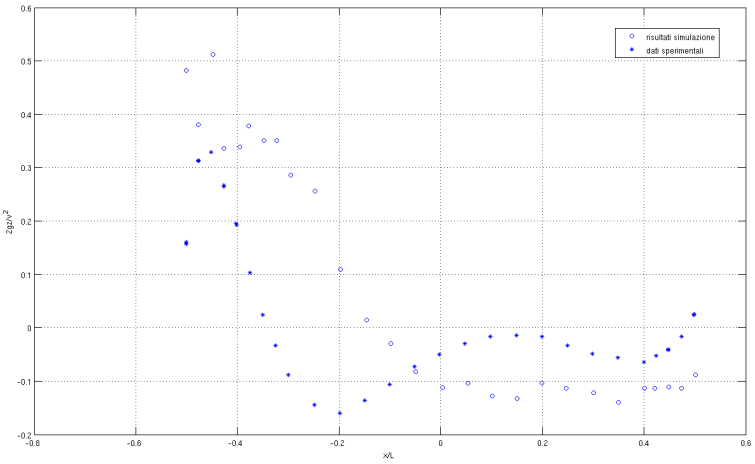



**Figura 3.3:** Profilo d'onda con $Fr = 0.316$.

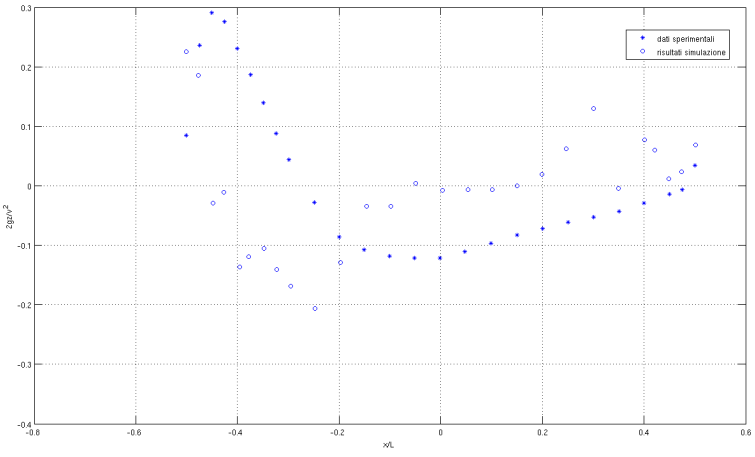



**Figura 3.4:** Profilo d'onda con $Fr = 0.408$.

| | | | |
|---|---|---|---|
| $V_\infty$ | 1.2381 | 1.5649 | 2.0205 |
| $Fr = \frac{V_\infty}{\sqrt{gl}}$ | 0.250 | 0.316 | 0.408 |

**Tabella 3.1:** Numero di Froude con cui vengono calcolati i profili d'onda presentati in [J.H85]

| | |
|---|---|
| Lunghezza | 8.1m |
| Larghezza | 0.296m |
| Altezza | 0.3m |
| Massa (con Vogatore) | 110 Kg |
| Momento di inerzia asse X | $80Kg/m^2$ |
| Momento di inerzia asse Y | $280Kg/m^2$ |

**Tabella 3.2:** Caratteristiche fisiche e geometriche utilizzate nelle simulazioni con il Wigley-Hull.

prodotto dai dati sperimentali. Quello che riusciamo a evincere è che il modello nei primi due casi sovrastima il picco di elevazione, e ciò si traduce in un profilo d'onda con un comportamento qualitativo analogo, ma quantitativamente diverso. Nell'ultimo caso invece, quello più critico dato l'elevato valore della velocità, il picco di elevazione iniziale viene sottostimato e, benchè anche in questo caso gli ordini di grandezza sono simili a quelli dei dati sperimentali, anche qualitativamente il comportamento risulta differente. Una delle cause principali di questo differente comportamento può ricercasi nella particolare triangolazione scelta, che, alla luce dei risultati ottenuti, risulta probabilmente troppo lasca per cogliere bene la fluidodinamica intorno allo scafo.

## 3.2 Test su scafo triangolato

Nel secondo caso test si è voluto validare la nuova funzionalità introdotta in *Stratos++v.2*, cioè la possibilità di interfacciarsi con scafi triangolati. Lo scafo scelto per la simulazione è il Wigley-Hull presentato in 1.1.9,le cui dimensioni e specifiche sono presentate in tabella 3.2 e approssimano quelle di un imbarcazione di canottaggio usato per una gara da singolo. Per produrre una triangolazione di questo scafo, ci si è avvalsi del software *Gmsh*, un generatore di griglia 3D che, dato un file **.geo** contenente la descrizione geometrica della superficie, è in grado di restituire una triangolazione di Delaunay isotropa o anisotropa della superficie. Lo scafo triangolato (rappresentato in figura 3.5) è dato da una triangolazione di Delaunay isotropa costituita da circa 9000 nodi e 20000 elementi. Lo scafo è soggetto a un moto uniformemente accelerato con partenza da fermo e accelerazione $a = 2m/s^2$ fino al raggiungimento della velocità di $5m/s^2$, dopo la quale si muoverà di moto rettilineo uniforme.

Durante il moto di avanzamento di un'imbarcazione di canottaggio, si assiste a un particolare fenomeno fisico, cioè la generazione di strutture d'onda particolari, chiamati *coni di Froude*, causate dal superamento da parte dell'imbarcazione della velocità di propagazione delle onde. Dato che siamo nelle ipotesi di acque basse, il calcolo delle velocità di propragazione delle onde può essere effettuato utilizzando la teoria lineare. In questo caso, considerando come

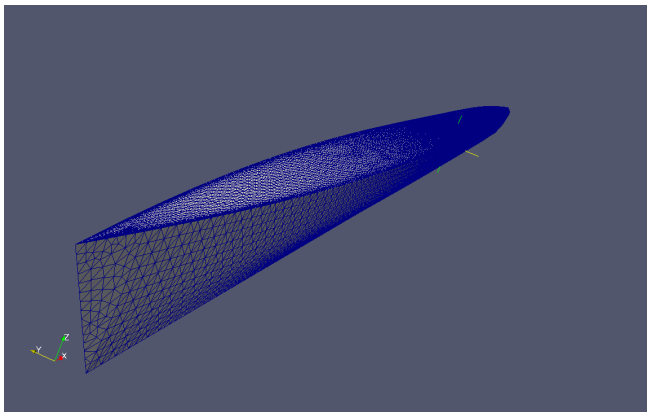
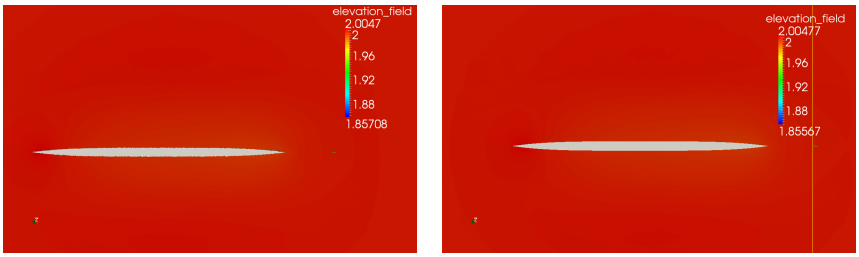


**Figura 3.5:** Scafo Wilgley-Hull triangolato.

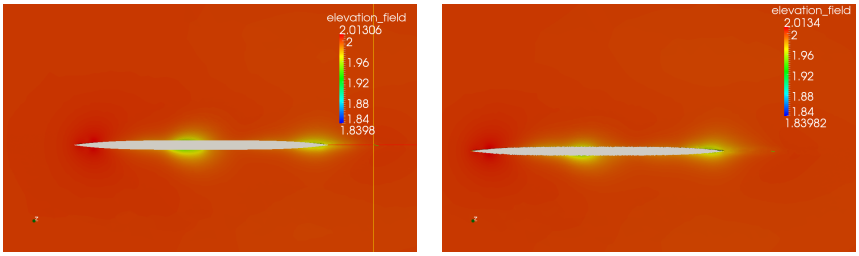
altezza la profondità del canale, si può approssimare la velocità di propragazine delle onde come

$$c \simeq \sqrt{gh} = 4.43 m/s$$

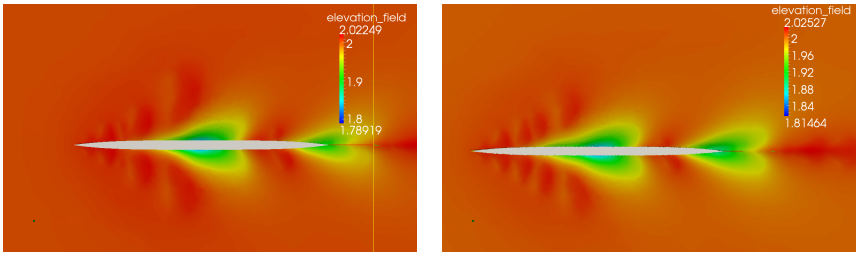
Quindi quando l'imbarcazione supera questa velocità (e nel caso in esame ciò accade all'istante $t = 2.2s$) si assiste alla comparsa dei coni di Froude. In figura 3.6 mostriamo il campo di elevazione per imbarcazione descritta dalla funzione analitica a confronto con il campo di elevezione nel caso di geometria triangolata. Possiamo osservare che le simulazioni in entrambi i casi riescono a cogliere e riprodurre il fenomeno dei coni di Froude. Tuttavia si ha la formazione, in corrispondenza della prua e della poppa dell'imbarcazione, di piccole oscillazioni del campo di elevazione che poi si propagano durante la sviluppo dei coni di Froude. Questo fenomeno, che viene riprodotto da entrambi i casi, è legato all'espressione della funzione che descrive il Wigley-Hull, in quanto essa assegna implicitamente spessore nullo agli estremi dell'imbarcazione. Questo va a riflettersi in una difficoltà da parte della mesh bidimensionale $\Omega_h$ di rappresentare la geometria dello scafo nell'intorno di questo punto, come mostrato in figura 3.9. Questo errata approssimazione provoca le oscillazioni numeriche che si osservano nella simulazione, che si potrebbero evitare con un infittimento della mesh $\Omega_h$ nell'intorno di tale punto. Tuttavia non si è voluto indagare ulteriormente in questa direzione in quanto nelle imbarcazioni reali la prua avrà uno spessore e quindi non ci si troverai ad affronterare problematiche di questo tipo. Nei grafici 3.7 e 3.8 viene presentata l'evoluzione temporale della posizione del baricentro dello scafo in direzione $z$ e l'andamento della forza di lift. L'andamento qualitativo è lo stesso in entrambi i casi, tuttavia si può osservare come la forza di lift al primo istante temporale risulta maggiore nel caso di geometria analitica rispetto allo scafo triangolato, e ciò condiziona la posizione del baricentro dello scafo negli istanti successivi, portando il baricentro dell'imbarcazione più in alto rispetto al secondo caso. Queste differenze sono conseguenza sia della differente rappresentazione dello scafo sia sopratutto dalla diversa procedura con cui si calcolano le forze agenti su di esso: infatti nel primo caso i nodi della mesh $\Omega_h$ sono gli stessi con cui si discretizza lo scafo, e dato il valore dei moltiplicatori di Lagrange $\lambda$ costante su ogni elemento di $\Omega_h$, il valore di $\lambda$ sui nodi viene ricostruito come una media pesata con l'area degli elementi che hanno quel nodo in



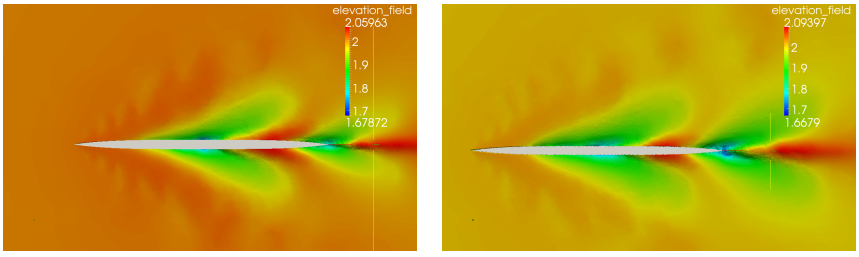

**(a)**




**(b)**




**(c)**




**(d)**

**Figura 3.6:** Confronto tra i risultati del campo di elevazione calcolato con scafo analitico (a sinistra) e scafo triangolato (a destra) su $4s$ di simulazione.

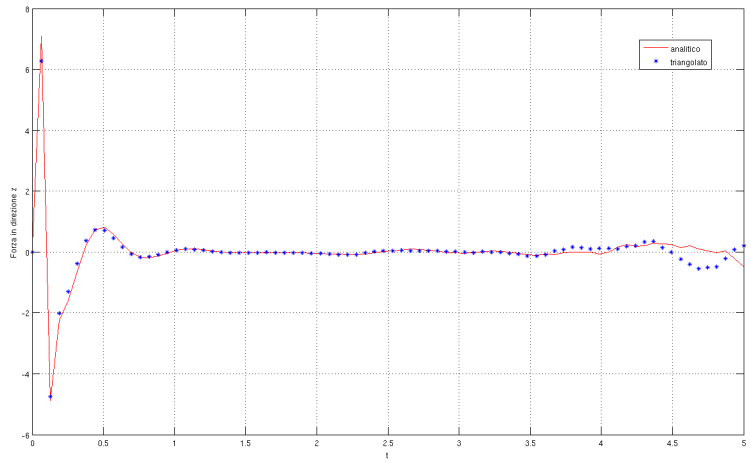



**Figura 3.7:** Andamento delle forza di lift che agisce sullo scafo.

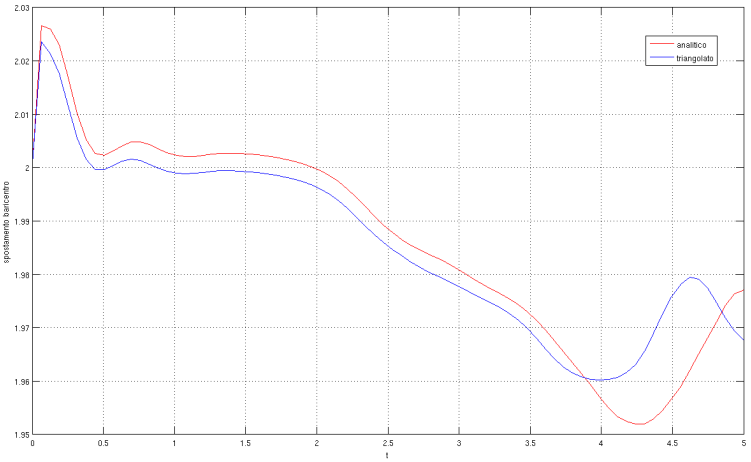



**Figura 3.8:** Andamento dello spostamento del baricentro dell'imbarcazione nella direzione $z$.

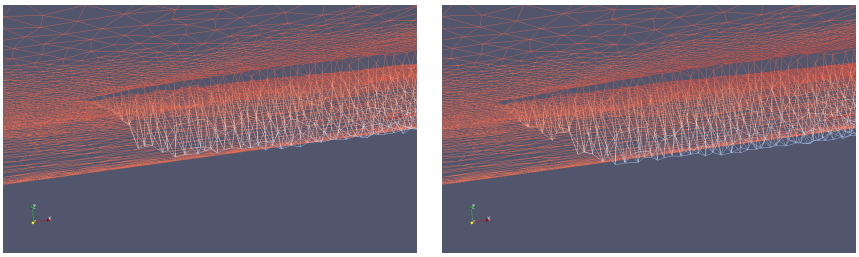
| | |
|---|---|
| Lunghezza | 12.7 $m$ |
| Larghezza | 0,43 $m$ |
| Altezza | 0.38 $m$ |
| Massa Scafo | 49.6 $Kg$ |
| Posizione centro di massa | $(6.3302, 0, -0.134887)$ |
| Momento di inerzia asse X | 515.25 $Kg/m^2$ |
| Momento di inerzia asse Y | 515.25 $Kg/m^2$ |
| Momento di inerzia asse Z | 515.25 $Kg/m^2$ |

**Tabella 3.3:** Alcune caratteristiche fisiche e geometriche dell'imbarcazione.

comune; nel secondo caso invece i nodi che descrivono $\Omega_h$ sono sfalsati rispetto ai nodi di $\mathcal{S}_h$ ed è quindi stato necessario introdurre un operatore interpolante per il calcolo di $\lambda$ sui nodi di $\mathcal{S}_h$.

## 3.3 Simulazione in condizione di regata

In questo ultimo caso test si è voluto testare la seconda nuova funzionalità introdotta in *Stratos++v.2*, cioè la possibilità di accoppiarsi con il software *Kimè* per ottenere simulazioni di un imbarcazione da canottaggio in condizione di regata. In particolare, l'imbarcazione di canottaggio scelta (rappresentata in figura 3.10) è quella con 4 vogatori, le cue caratteristiche fisiche sono presentate in 3.3. La tecnica di accoppiamento usata è quella CSS con predittore strutturale del primo ordine e un differente passo temporale per il risolutore *Stratos++* e per *Kimè*, rispettivamente $\Delta t_f = 0.01$ e $\Delta t_s = 0.05$ e vengono simulati $5s$ con velocità iniziale pari a $\mathbf{v} = (5.3, 0, 0)$ e accelerazione nulla. Nella simulazione si è inotre imposto una velocità nulla nella direzione trasversale, questo per evitare il fenomeno della deriva dell'imbarcazione, dato che all'interno di *Stratos++* questo fenomeno non viene modellato.
Per avere un termine di paragone si è deciso di confrontare i dati della simulazione ottenute risolvendo la fluidodinamica attraverso il risolutore di *Stratos++* con i risultati delle simulazioni usando il modello a potenziale implementato all'interno di *Kimè*.
In figura 3.11 mostriamo l'evoluzione temporale della posizione del baricentro rispetto alla coordinata $z$. Quello che si può osservare è che utilizzando *Stratos++* come risolutore lo spostamento del baricentro in direzione $z$ risulta più irregolare rispetto al risolutore di *Kimè*, anche se in entrambe le situazioni ci si assesta intorno alla posizione di equilibrio. Risultati analoghi si ottengono per l'evoluzione temporale dell'angolo di rollio, come mostrato in figura 3.14, dove con entrambi i risolutori otteniamo un andamento periodico di periodo circa $T = 1.5s$ ma con un diversa evoluzione all'interno del periodo.
L'evoluzione del angolo di beccheggio, mostrato in figura 3.13, presentando un andamento qualitativo molto simile, in quanto risultano entrambi periodici sempre di periodo $T$ e mostrano una tendenza ad amplificarsi al crescere dell'istante temporale. Tuttavia quantitativamente c'è una differenza di circa un ordine di grandezza tra i due. In figura 3.12 e 3.15 presentiamo rispettivamente l'evoluzione temporale della componente della velocità in direzione $x$ e direzione $z$. Mentre per quanto riguarda la velocità nella direzione $z$ osserviamo un com-



**(a)**

**Figura 3.9:** Campo di elevazione in corrispondenza della prua dello scafo nel caso di geometria ricavata dalla funzione analitica (a sinistra) e di scafo triangolato (a destra).

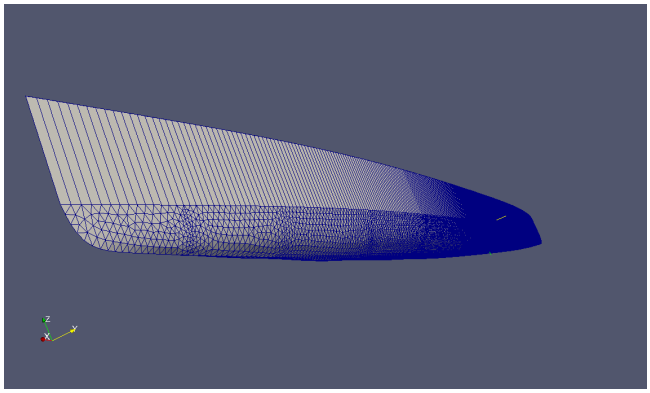


**Figura 3.10:** Scafo di un imbarcazione di canottaggio a 4 posti.

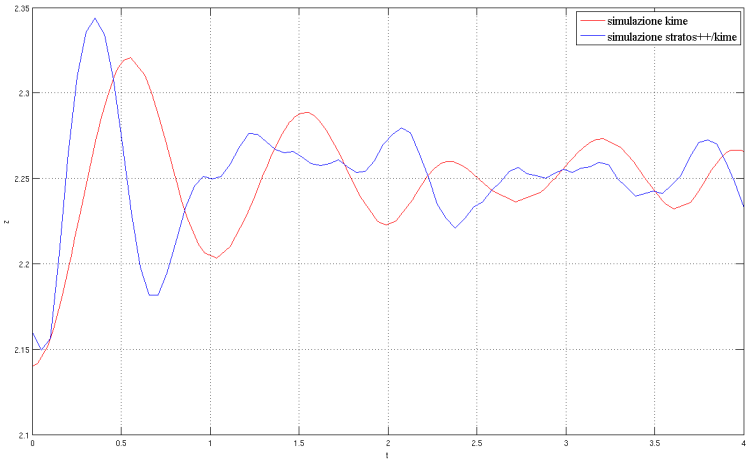



**Figura 3.11:** Evoluzione temporale della posizione del baricentro in direzione $z$.

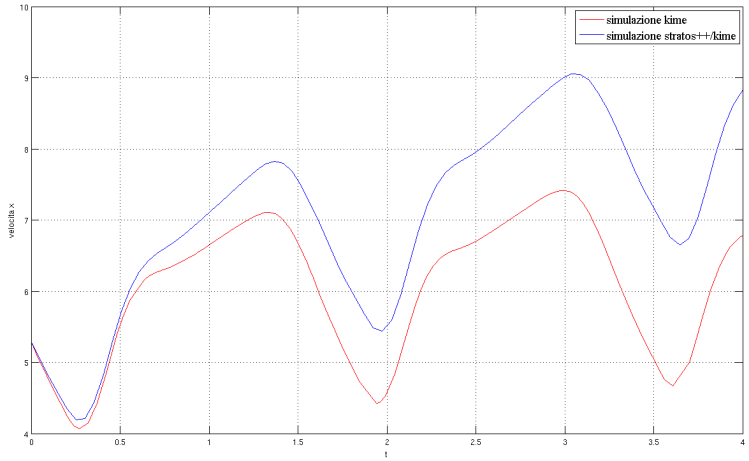



**Figura 3.12:** Evoluzione temporale della velocità dell'imbarcazione in direzione $x$.

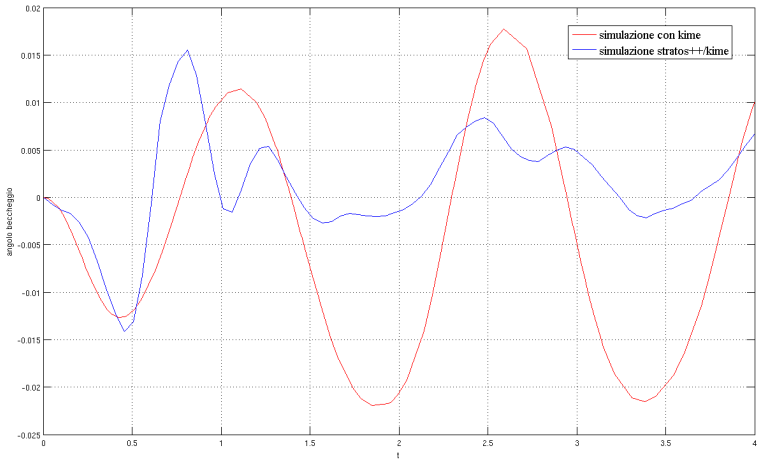



**Figura 3.13:** Evoluzione temporale dell'angolo di beccheggio.

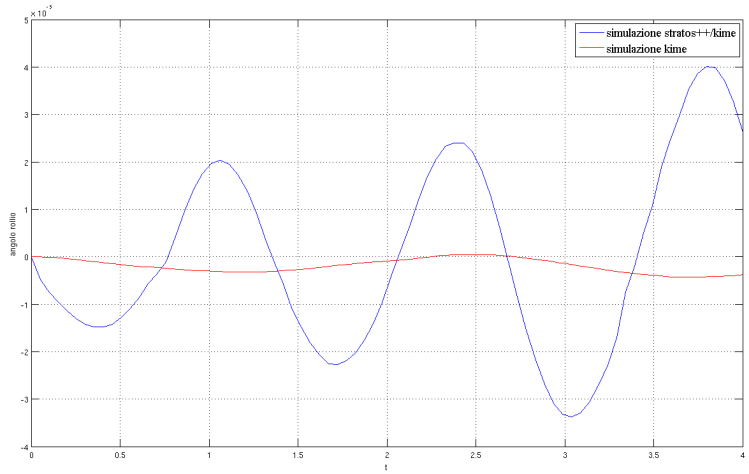



**Figura 3.14:** Evoluzione temporale dell'angolo di rollio.

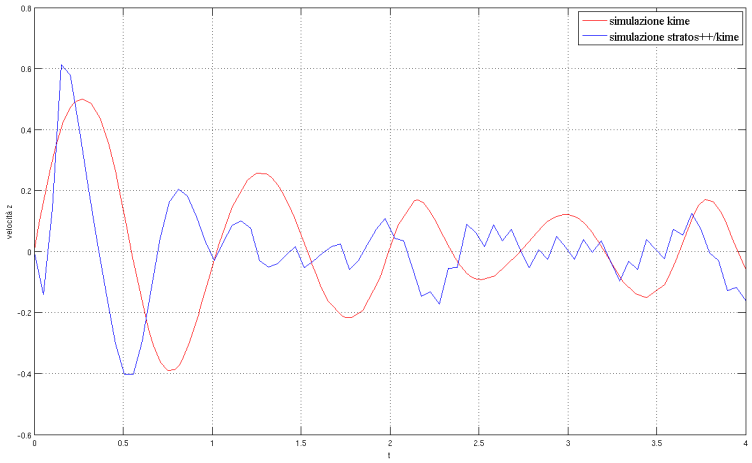



**Figura 3.15:** Evoluzione temporale della velocità dell'imbarcazione in direzione $z$.

portamento più irregolare introdotto dal risolutore *Stratos++*, comunque con ordini di grandezza comparabili a quelli ottenuti con *Kimè*, per la velocità in direzione $x$ il comportamento risulta qualitativamente identico, ma con una velocità che cresce maggiormente nelle simulazioni con *Stratos++*. Quest'ultimo comportamento riflette la mancanza nel modello di termini che modellino gli effetti viscosi sull'imbarcazione, effetti che invece vengono considerati all'interno del modello implementato in *Kimè*.
Molte possono essere le possibili cause che portano a una diversa evoluzione della dinamica dell'imbarcazione. Innanzitutto nel software *Kimè* l'integrazione in tempo è fatta attraverso un metodo Runge-Kutta che ha un passo di tempo adattivo, che permette di seguire la dinamica dei vogatori, mentre nel nostro caso scegliamo un passo temporale costante e pari a $\Delta t_s = 0.05s$, assumendo la velocità e la posizione della struttura costante in questo $\Delta t_s$, e ciò risulta probabilmente un approssimazione troppo forte che non permette di cogliere bene l'evoluzione della dinamica dell'imbarcazione. Ovviamente una seconda differenza è legata al modello usato per risolvere la fluidodinamica intorno allo scafo, che nel caso di *Kimè* è un modello a potenziale $3D$ in cui la matrice di massa e di damping e calcolata per singola frequenza, e ciò può motivare la differenza di regolarità nelle due soluzioni. Un'ultima motivazione è legata al numero di strati usati per la simulazione, che potrebbero essere non sufficienti a cogliere bene la dinamica dello scafo.

# Conclusione

L'obiettivo principale del lavoro è stato quello di definire una strategia finalizzata alla soluzione di un problema di interazione fluido-struttura. Per modellare la parte fluidodinamica abbiamo presentato il modello 3D multistrato per le equazioni *Shallow Waters*, già implementato nella precedente versione di *Stratos++*, alla quale sono state applicate le modifiche necessarie per renderlo utilizzabili con superfici triangolate. Questo obbiettivo è stato raggiunto introducendo una procedura ottimizzata per la determinazione della quota dello scafo, cercando di ridurre le ricerche attraverso l'introduzione di condizioni necessarie e sufficienti e sfruttando al massimo le informazioni delle precedenti chiamate. La seconda importante modifica apportata a *Stratos++* è il calcolo delle forze sull'imbarcazione, in quanto nel caso di scafi rappresentati da superfici triangolate, i nodi della mesh sui quali conosciamo il valore del moltiplicatore di Lagrange $\lambda$ non corrispondono ai nodi che discretizzano l'imbarcazione. Abbiamo per questo introdotto un interpolante del primo ordine che permettesse di ottenere i valori di $\lambda$ sullo scafo e quindi poter calcolare le forze e i momenti agenti su di esso. Per modellare la dinamica dell'imbarcazione dovuta alla presenza dei vogatori ci si è invece appoggiati al modello utilizzato nel software *Kimè*, un modello calibrato attraverso l'utilizzo di dati sperimentali. La terza fondamentale modifica apportata al codice è la possibilità di accoppiare il modello fluidodinamico con quello alla struttura attraverso una procedura CSS con predittore strutturale del primo ordine. Queste modifiche ci hanno permesso di ottenere uno strumento capace di produrre simulazioni realistiche e in grado di fornire utili suggerimenti per il miglioramento delle prestazioni delle canoe. Per quanto riguarda gli sviluppi di questo lavoro, si potrebbe pensare di introdurre nel modello di *Stratos++* dei termini che tengano conto degli effetti viscosi sull'imbarcazione e che contrastino il fenomeno della deriva. Inoltre le proprietà di stabilità e convergenza della soluzione sono fortemente legate a proprietà geometriche della mesh relative alla posizione dei circocentri, e quindi un possibile sviluppo potrebbe essere quello di rendere più stabile la soluzione a raffinamenti della mesh introducendo per esempio mesh ibride che usano elementi quadrilateri. Infine si potrebbe proporre una riscrittura del codice in versione parallela. Esistono infatti diverse parti, come l'inversione della matrice di pressione, la determinazione della quota dello scafo o il calcolo delle forze che agiscono su di esso, che si prestano a essere parallelizzate ottenendo un codice più performante, che ci permetta di utilizzare delle mesh più raffinate sia per il dominio bidimensionale $\Omega$, sia per la descrizione dello scafo, soprattutto in corrispondenza degli estremi dello scafo. Inoltre apportando questa modifica si potrebbe ridurre il passo temporale usato nell'accoppiamento fluido-struttura, ottenendo una descrizione più precisa della dinamica dell'imbarcazione da canottaggio in condizione di regata.

# Bibliografia


[Alt09] M. S. Altieri, *Modellazione numerica della dinamica di imbarcazioni tramite vincolo unilatero sulla superficie libera*, Master's thesis, Politecnico di Milano, 2009.

[Are88] J. Arenberg, *Ray triangle intersection with barycentric coordinates*, Ray Tracing News **1** (November 4 1988), no. 11.

[EM02] P. C. E. Miglio, F.Saleri, *Algebraic factorizations for 3d non-hydrostatic free surface flows*, Computing and Visualization in Science **5** (2002), 85–94.

[FdR77] P. K. C. Felippa, C. A. e J. A. de Runtz, *Stabilization of staggered solution procedures for fluid-structure interaction analysis*, Computational methods for fluid-structure interaction problems (1977), 95–124.

[FL00] C. Farhat e M. Lesoinne, *Two efficient staggered algorithms for the serial and parallel solution of three-dimensional nonlinear transient aeroelastic problems*, Computer Methods in Applied Mechanics and Engineering **182** (2000), no. 3-4, 499 – 515.

[FMMM09] L. Formaggia, E. Miglio, A. Mola e A. Montano, *A model for the dynamics of rowing boats*, International Journal for Numerical Methods in Fluids **61** (2009), no. 2, 119–143.

[Fra79] R. Franke, *A critical comparison of some methods for interpolation of scattered data*, Nonlinear Analysis (1979).

[J.H85] M. J.H., *Collected experimental resistance component and flow data for three surface ship model hulls*, Tech. report, David Taylor Research Center, 1985.

[Kas] C. Kassiotis, *Nonlinear fluid-structure interaction: a partitioned approach and its application through component technology*, Ph.D. thesis, university paris-est.

[LF08] A. M. A. S. Luca Formaggia, Edie Miglio, *Numerical simulation of the dynamics of boat by a variational inequality approach*, Tech. report, Politecnico di Milano, 2008.

[LT11] E. M. A. S. Lorenzo Tamellini, Luca Formaggia, *An uzawa iterative scheme for the simulation of floating bodies*, Submitted to Applied Numerical Mathematics (18 october 2011).

[MT05] T. Möller e B. Trumbore, *Fast, minimum storage ray/triangle intersection*, ACM SIGGRAPH 2005 Courses (New York, NY, USA), SIGGRAPH '05, ACM, 2005.

[PF97] S. Piperno e C. Farhat, *Design and Evaluation of Staggered Partitioned Procedures for Fluid-Structure Interaction Simulations*, Tech. Report RR-3241, INRIA, Sep 1997.

[PF01] S. Piperno e C. Farhat, *Partitioned procedures for the transient solution of coupled aeroelastic problems – part ii: energy transfer analysis and three-dimensional applications*, Computer Methods in Applied Mechanics and Engineering **190** (2001), no. 24-25, 3147 – 3170, Advances in Computational Methods for Fluid-Structure Interaction.

[PFL95] S. Piperno, C. Farhat e B. Larrouturou, *Partitioned procedures for the transient solution of coupled aroelastic problems part i: Model problem, theory and two-dimensional application*, Computer Methods in Applied Mechanics and Engineering **124** (1995), no. 1-2, 79 – 112.

[Phi07] G. M. Phillips, *Interpolation and approximation by polynomials*, Springer, 2007.

[Pip97] S. Piperno, *Explicit/implicit fluid/structure staggered procedures with a structural predictor and fluid subcycling for 2d inviscid aeroelastic simulations*, International Journal for Numerical Methods in Fluids **25** (1997), no. 10, 1207–1226.

[Pre07] W. H. Press, *Numerical recipes, the art of scientific computing*, Cambridge, 2007.

[Qua08] A. Quarteroni, *Matematica Numerica (terza ed.)*, Springer-Italia, Milan, IT, 2008.

[Sal08] S. Salsa, *Partial differential equations in action - from modelling to theory*, Springer, 2008.