POLITECNICO DI MILANO

SIMULAZIONI NUMERICHE PER L'INTERAZIONE FLUIDO-STRUTTURA IN EMODINAMICA

Matteo Pozzoli

matr. 681130

Relatori: Dr. F. Nobile, Dr. C. Vergara

MOX, Modeling and Scientific Computing,
Dipartimento di Matematica "F.Brioschi",
Politecnico di Milano, Milano

Indice

Capitolo 1

Introduzione all'Emodinamica Computazionale

Negli ultimi anni si è sviluppato un crescente interessere in ambito medico e bioingegneristico verso le simulazioni numeriche dei sistemi biologici ([28], [18], [19], [31]). In particolare, le patologie cardiovascolari hanno un impatto socio-economico rilevante nei paesi occidentali così da motivare studi in emodinamica di carattere computazionale (*Computational Fluid Dynamic* (CFD)). Queste ricerche sono in grado di fornire alcune importanti informazioni riguardo al flusso sanguigno, soprattutto nel determinare alcune grandezze fisiche difficili da misurare nei pazienti. Ad esempio, sebbene il flusso sanguigno nei vasi sanguigni è una grandezza facilmente misurabile durante analisi cliniche, il campo di velocità puntuale può essere ottenuto solo attraverso tecniche costose, come la *risonanza magnetica*. Altre grandezze come lo sforzo superficiale, sono praticamente impossibili da misurare. Queste quantità possono essere calcolate attraverso simulazioni numeriche condotte su geometrie reali tridimensionali.

Differenti sono gli scopi dell'emodinamica computazionale: un primo esempio consiste nel capire come nascono e si sviluppano le patologie. Citiamo a titolo di esempio sono le simulazioni numeriche condotte su arterie polmonari di neonati affetti da ipoplasia ventricolare sinistra, che hanno permesso di chiarire quale sia l'impatto del sangue sullo sforzo superficiale e fornito una spiegazione quantitativa delle osservazioni successive nei pazienti ([6]). Un secondo scopo è lo sviluppo di protesi mediche, ad esempio da circa 30 anni è diventata una pratica comune in cardiologia l'uso dello *stent*, per aprire una stenosi in un vaso. Uno stent è un piccolo tubo metallico che, inserito permametemente nell'arteria, aiuta a mantenerla aperta così che il sangue può fluire normalmente. In particolare ne *drug eluting stents* sono contenuti dei medicinali in grado di ridurre le possibili reazioni di infiammazione della parete vascolare. Nella loro preparazione, le informazioni ottenute grazie alle ruolo delle simulazioni numeriche permettono di progettare la pellicola di rivestimento dello stent in modo tale da garantire un rilascio ottimale del farmaco, ([35], [34]).

Un terzo esempio è fornito dalle simulazioni numeriche usate per confrontare i diversi metodi di intervento nelle malattie cardiache pediatriche dando indi-

cazioni pratiche al chirurgo ([23]).
Infine, le simulazioni numeriche possono essere utilizzate per identificare e ottimizzare alcuni parametri fisici; attraverso la risoluzione di problemi inversi è possibile ricavare una soluzione numerica che soddisfi certi criteri di ottimalità. Ad esempio le tecniche di ottimizzazione della forma sono applicate nei bypass coronarici, riducendo la probabiltà di fallimento dell'intervento chilurgico. ([30]). Per maggiori informazioni e dettagli riguardanti le applicazioni dell'emodinamica computazionale si rimanda a [19].

1.1 Breve descrizione del problema FSI

Negli ultimi anni si è sviluppato un interesse negli algoritmi per simulare le interazione fluido-struttura in emodinamica.

Il flusso sanguigno nei vasi è assai complesso da modellare in termini matematici in quanto, oltre alla necessità di descrivere il sangue con un opportuno modello fluido si deve definire un adeguato sistema struttura per modellare il vaso sanguigno. Il sangue può essere considerato un fluido omogeneo e incomprimibile, per lo meno in vasi di grosse dimensioni (raggio maggiore a $0.1cm$) e in situazioni non patologiche, che scorre in un dominio deformabile. Dunque può essere descritto come un sistema di *Navier-Stokes* per un fluido Newtoniano in un dominio mobile. La formulazione ALE ([21], [10], [18], [16], [17]) è un utile strumento per trattare dal punto di vista numerico le equazioni di Navier-Stokes definite in un dominio mobile. Su vasi più piccoli, come nei capillari, si devono usare altri modelli ([18], [31]). I vasi sanguigni hanno un comportamento molto complesso con una natura elastica e anisotropica; inoltre hanno una configurazione di pre-stress sia in direzione longitudinale che radiale. In particolare, quando un'arteria è estratta dal corpo, tende a ridurre la sua lunghezza e se si taglia un piccolo anello, esso si dilata (si veda [20]). In questo lavoro si suppone che i vasi sanguigni siano composti di un materiale elastico e isotropico descritto tramite le equazioni di *Saint-Venant Kirchhoff*.

Una possibile procedura per modellizzare il sistema sangue-vaso sanguigno è quello di trattara i due fenomeni con un unico sistema di equazioni struttura e collegate tra di loro attraverso delle condizioni di interfaccia. In particolare, tali connessioni sono *la condizioni di aderenza del fluido con la parete solida* e *il principio di azione-reazione*, ossia lo sforzo che il fluido esercita sulla parete solida deve essere uguale e contrario allo sforzo che il solido esercita sul fluido.

Il problema fluido-struttura è altamente non lineare in particolare le principali caratteristiche che denotano le non linearità del problem sono:

1. La posizione dell'interfaccia tra fluido e struttura non è nota. Ciò introduce una non linearità geometrica.

2. Il termine convettivo del problema fluido è non lineare, e nella *formulazione ALE* (si veda sezione 2 Capitolo 2), dipende anche dalla velocità del dominio fluido.

3. I sottoproblemi fluido e struttura sono accoppiati attraverso le condizioni di interfaccia, cioè la continuità della velocità e dello sforzo,

Dal punto di vista numerico, l'approccio più comune consiste nel sottoiterare fra

soluzioni dei sottoproblemi fluido e struttura. Ogni sottoproblema viene risolto separatamente con il vantaggio di poter riutilizzare codici pre-esistenti.

Gli algoritmi che risolvono i sottoproblemi fluido e struttura una volta sola ad ogni passo temporale e non soddisfano esattamente le condizioni di interfaccia sono chiamati *espliciti*.
Questa procedura non garantisce il bilanciamento energetico tra i sottoproblemi fluido e struttura, con il rischio di produrre delle instabilità negli schemi numerici. Infatti, è mostrato in [3] che l'accoppiamento esplicito non è applicabile in emodinamica a causa del forte effetto di *massa aggiunta*. Tale effetto è tanto maggiore tanto più le densità del fluido e della struttura sono simili.

Risulta quindi necessario utilizzare *schemi impliciti* che ad ogni iterazione temporale garantiscono un corretto bilancio energetico tra fluido e struttura. Tuttavia questo comporta dei costi computazionali estremamente elevati, e perciò uno dei principali obiettivi della ricerca nelle simulazioni fluido-struttura, è quello di ridurre tali costi. A questo fine, si considerino gli algoritmi basati sull'utilizzo di codici fluido e struttura pre-esistenti, usati in un contesto sotto-iterativo.

Come primo approccio si consideri il metodo di *Dirichlet-Neumann*, dove al problema fluido è prescritta la condizione di Dirichelt all'interfaccia (ovvero si impone la velocità della struttura calcolata alla precedente iterazione) e al problema struttura è invece prescritta una condizione di Neumann all'interfaccia ottenuta dallo sforzo normale calcolato dal problema fluido. Questo schema è il metodo più comune per costruire schemi partizionati (si veda [1]). Una caratteristica del metodo è la sua facile implementazione (si veda [3]). Tuttavia se il fluido e la struttura hanno densità simili, per poter ottenere la convergenza delle sotto-iterazioni è necessario introdurre un opportuno rilassamento con la conseguenza che il numero di iterazioni per arrivare a convergenza risulta molto elevato. Altri possibili schemi sono stati introdotti in [1]. Essi sono basati su condizioni di interfaccia ottenute combinando linearmente la continuità della velocità e dello sforzo normale. In questo modo si ottengono delle condizioni d'interfaccia basate su condizioni di Robin. Lo schema più generale ottenuto con questo approccio è quello di Robin-Robin, per cui sia al problema fluido che al problema struttura è prescritta una condizione di Robin all'interfaccia. Ognuna di queste condizioni dipende da un parametro, la cui scelta determina le proprietà di convergenza dello schema. In [1] è stato proposto di scegliere tali parametri a partire da modelli ridotti per il fluido e per la struttura, introdotti rispettivamente in [3] e in [26]. Si sottolinea come l'algoritmo di Dirichlet-Neumann appartenga a questa classe di schemi, per una particolare scelta dei parametri. Tra tutti i possibili algoritmi di Robin-Robin, in [1] è stato mostrato come il più performante sia lo schema di Robin-Neumann. In particolare l'analisi di convergenza su un problema modello ha mostrato che l'algoritmo di Robin-Neumann è molto meno sensibile all'effetto di massa aggiunta rispetto all'algoritmo di Dirichlet-Neumann. In [1] è stato anche mostrato numericamente che l'algoritmo di Robin-Neumann converge senza rilassamento e in un numero di iterazioni molto minore rispetto a quella necessarie per la convergenza dell'algoritmo di Dirichlet-Neumann. Tuttavia, le prove numeriche effettuate in [1] sono state ottenute solamente per il caso bidimensionale.

L'obbiettivo di questo lavoro consiste nell'effettuare simulazioni numeriche fluido-struttura basate sull'algoritmo di Robin-Neumann in domini tridimensionali. In particolare tale algoritmo è stato implementato nella libreria *LifeV* [1]. Più precisamente si è partiti dalla versione parallela della libreria, *LifeV-parallel*, e si è considerato un caso test fluido-struttura (3D-3D), risolto originariamente tramite un algoritmo di punto fisso implicito basato su una strategia con trattamento delle condizioni di interfaccia di tipo Dirichlet-Neumann. Sono state quindi apportate le modifiche necessarie in modo tale da trattare le condizioni di Robin-Neumann all'interfaccia. Il codice così ottenuto, ha quindi la pecularietà di essere *parallelo*, ossia ogni simulazione può essere svolta su più processori in modo tale da accellerare il calcolo della soluzione pur mantenendo inalterato il numero di sottoiterazioni necessarie.

In questo lavoro sono stati confrontati gli algoritmi di Dirichlet-Neumann e di Robin-Neumann. Come mostrato nei risultati riportati nel terzo capitolo, l'algoritmo di Robin-Neumann anche nel caso 3D-3D necessita di un minor numero di sottoiterazioni e non ha bisogno di essere rilassato per convergere, a differenza dell'algoritmo Dirichlet-Neumann.

I domini computazionali sono finiti e rappresentazno delle porzioni di vasi sanguigni. Per questo motivo si devono introdurre dei *bordi artificiali* di Inflow e di Outflow che delimitano il dominio computazionale e non corrispondono ad un bordo fisico realmente esistente. L'introduzione di tale bordo è generalmente fonte di inaccuratezza nel momento in cui si vanno a prescrivere le condizioni al bordo: spesso infatti i dati a disposizione non sono sufficienti per formulare un problema matematico ben posto.

È noto che, anche se il problema del fluido è descritto da equazioni paraboliche, la natura del problema di interazione fluido-struttura nell'ambito dell'emodinamica presenta caratteristiche riconducibili ad un problema iperbolico; in particolare le onde di pressione attraversano il dominio fluido. In mancanza d'informazioni sulle sezioni artificiali, l'imposizione di una pressione arbitraria da luogo a riflessioni spurie di pressione. Di conseguenza, l'imposizione di una condizione adatta sulla sezione di ingresso e uscita del fluido che non generi riflessioni è un problema molto delicato e di fondamentale importanza al fine di ottenere risultati numerici soddisfacenti.

Allo scopo di risolvere il fenomeno legato alle riflessioni spurie, si è imposta una condizione assorbente, proposte in [26], ottenuta dall'accoppiamento di un modello 3D con un modello ridotto 1D. Un secondo risultato di questo elaborato sono l'estensione delle condizioni assorbenti al problema 3D-3D.

Questo elaborato è così suddiviso: il secondo capitolo presenta la formulazione debole del problema fluido-struttura, la sua discretizzazione e la formulazione degli schemi di Dirichlet-Neumann e Robin-Neumann. Il terzo capitolo presenta gli esperimenti svolti con il confronto tra gli algoritmi di Dirichlet-Neumann e Robin-Neumann, con tre differenti geometrie: un vaso sanguig-

[1] LifeV è una libreria ad elementi finiti, sviluppata dal Politecnico di Milano (laboratorio MOX), Ecole Polytechnique Fèdèrale di Lausanne (gruppo CMCS) e INRIA (progetto REO), per la risoluzione di equazioni differenziali a derivate parziali con i metodi numerici. La versione di lifeV-parallel, rispetto alle precedenti versioni ha la caratteristica di poter lanciare un *job* su più processori riducendo i costi di calcolo del processo.

no cilindrico rettilineo, un arco arterioso e una carotide (ottenuta dalla ricostruzione di immagini mediche.)

Il quarto capitolo descrive l'implementazione del problema in LifeV-parallel, in quanto parte fondamentale del lavoro. La prima sezione del capitolo presenta il codice parallelo con particolare enfasi alla gestione dei vettori e delle matrici attraverso il pacchetto della libreria Trilinos Epetra, utilizzato nella libreria. La seconda sezione di questo capitolo descrive il problema fluido-struttura definendo le principali caratteristiche del codice `test_FSI` e le modifiche fatte per ottenere un problema con le condizioni di Robin-Neumann.

Capitolo 2

Modellazione del problema di interazione fluido-struttura

In questo capitolo sono presentati i modelli matematici e le tecniche numeriche per la modellazione del flusso sanguigno nei vasi vascolari.

Dal punto di vista fisico, il sangue è un miscuglio di diversi tipi di particelle come i globuli rossi, i globuli bianchi e le piastrine in una soluzione acquosa chiamata plasma (Figura 2.1). Il moto del sangue ha una natura *impulsiva* dovuta

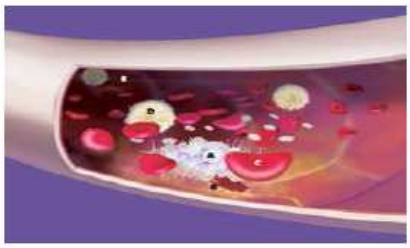

Figura 2.1: Il sangue è un insieme di particelle non omogeneo: globuli rossi, bianchi, piastrine e plasma.

all'azione del cuore. Il tempo caratteristico in questo contesto è determinato dal battito cardiaco (circa $0.8s$) in cui si può registrare una prima fase chiamata *sistole* (circa $0.3s$), quando la valvola aortica è aperta e il sangue affluisce nelle arterie, e una seconda, la *diastole*, che inizia con la chiusura dalla valvola aortica, in cui il cuore si riempie di sangue.

A causa della natura complessa del sangue si utilizzano alcune assunzioni semplificative per definire un opportuno modello matematico in grado di descriverlo. Inanzitutto il sangue può essere considerato come un fluido omogeneo incomprimibile; quest'ipotesi è vera se si considerano dei larghi vasi sanguigni (ossia di raggio maggiore a $0.1cm$) e in situazioni non patologiche. Inoltre in questi casi, la reologia del sangue è ben approssimata da una legge Newtoniana e il

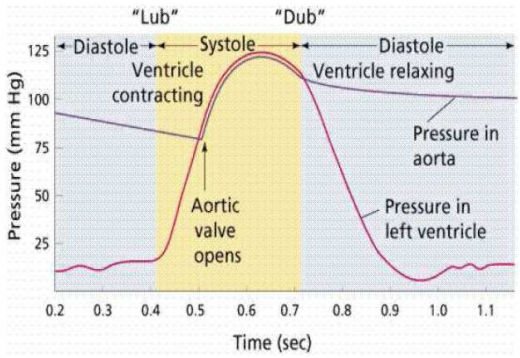

Figura 2.2: Si descrive l'instante in cui dal ventricolo sinistro il sangue effluisce nell'aorta, grazie all'apertura e chiusura della valvola aortica. La pressione sanguigna nel ventricolo sinistro cresce appena il ventricolo si constrae e continua ad aumentare fino al raggiungimento di un massimo, successivamente il ventricolo si rilassa e la pressione descresce ritornando al valore iniziale. La pressione nell'aorta ha un minimo quando la valvola aortica si apre, poi finchè la valvola cardiaca è aperta, segue l'andamento della pressione del ventricolo. quando si chiude la pressione nell'aorta si riduce con pendenza minore di quella ventricolare.

legame costitutivo tra il tensore degli sforzi interni e il tensore della velocità di deformazione è lineare. In vasi sanguigni piccoli, come i capillari, si deve considerare un diverso tipo di reologia del sangue (vedere ad esempio [18], [31]).

Supposte valide queste ipotesi si può modellare il sangue attraverso le leggi di conservazione della massa e dela quantità di moto, ovvero le *equazioni di Navier-Stokes* per un fluido Newtoniano incomprimibile.

I vasi sanguigni hanno un meccanismo complesso e variabile come ad esempio elasticità e anisotropia, tuttavia in questo lavoro si descrive la struttura attraverso un modello di elasticità lineare e isotropo definito dalle *equazioni di Saint-Venant Kirchhoff* (si veda [5]).

2.1 Notazione

Si considera un sistema eterogeneo che riveste un dominio mobile $\Omega^t \subset \mathbb{R}^3$, dove t è il tempo. L'evoluzione di Ω^t non è nota *a priori*, ma risulta dalla soluzione del problema di interazione fluido-struttura. Il problema, dal punto di vista matematico è complesso e consiste nel determinare la velocità e la pressione del fluido, lo spostamento della struttura e la posizione dei punti di Ω^t. Questo dominio è diviso in un sottodominio Ω_s^t, occupato dalla struttura elastica e dal suo complementare Ω_f^t occupato dal fluido (Figura 2.3).

L'interfaccia di connessione tra il fluido e la struttura Γ^t è il bordo in comune tra Ω_s^t e Ω_f^t, ossia $\Gamma^t = \partial\Omega_s^t \cap \partial\Omega_f^t$. Si definisce con $\mathbf{n}_f$ la normale uscente da Ω_f^t, con $\mathbf{n}_s = -\mathbf{n}_f$ la controparte sul dominio struttura e con Ω^0 la configurazione iniziale detta il *dominio di riferimento* (Figura 2.4).

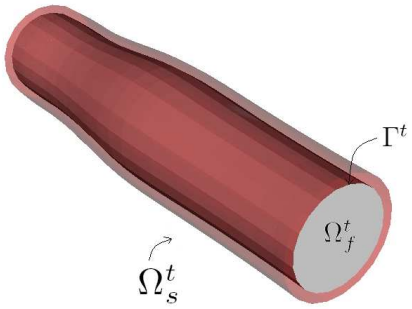

Figura 2.3: La parete arteriosa viene identificata con il dominio Ω_s^t, mentre il vaso in cui scorre il sangue con Ω_f^t; Γ^t è la parete comune dei due domini.

Per descrivere l'evoluzione del dominio Ω^t si definiscono due mappe:

$$\mathcal{L} : \Omega_s^0 \times (0,T) \to \Omega_s^t, \quad (\mathbf{x}_0, t) \to \mathbf{x} = \mathcal{L}(\mathbf{x}_0, t), \tag{2.1}$$

$$\mathcal{A} : \Omega_f^0 \times (0,T) \to \Omega_f^t, \quad (\mathbf{x}_0, t) \to \mathbf{x} = \mathcal{A}(\mathbf{x}_0, t). \tag{2.2}$$

La mappa $\mathcal{L}^t = \mathcal{L}(\cdot, t)$ determina il dominio del solido nel tempo, mentre

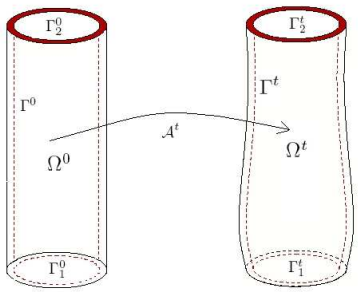

Figura 2.4: A sinistra il dominio di riferimento, mentre a destra il dominio corrente.

$\mathcal{A}^t = \mathcal{A}(\cdot, t)$ il dominio fluido. Le combinazioni di queste mappe unite alla seguente condizione di continuità all'interfaccia:

$$\mathcal{L}^t = \mathcal{A}^t \qquad \text{su} \quad \Gamma^t, \qquad \forall t \in (0,T) \tag{2.3}$$

determinano un omeomorfismo in Ω^t.

Per determinare la cinematica della struttura si utilizza un *approccio Lagrangiano*, quindi la mappa solida è così determinata:

$$\mathcal{L}^t(\mathbf{x}_0) = \mathbf{x}_0 + \hat{\boldsymbol{\eta}}(\mathbf{x}_0, t)$$

dove $\hat{\boldsymbol{\eta}}$ rappresenta lo spostamento del solido rispetto alla configurazione di riferimento.

Il problema fluido è definito mediante il metodo ALE (*Arbitrary Lagrangian-Eulerian method*, si veda [21], [10], [18], [16], [17]). La mappa del dominio fluido

$\mathcal{A}^t$ è scelta in modo tale che soddisfi la condizione (2.3) attraverso un'opportuna estensione del suo valore all'interfaccia:

$$\mathcal{A}^t = \mathbf{x}_0 + \mathrm{Ext}(\hat{\boldsymbol{\eta}}(\mathbf{x}_0, t)|_{\Gamma^0}) \tag{2.4}$$

Nei problemi considerati, è stato scelto l'operatore di estensione armonica nel dominio di riferimento.

Data una funzione $\hat{g} : \Omega_s^0 \times (0,T) \to \mathbb{R}$ nel dominio di riferimento, si definisce con $g = \hat{g} \circ (\mathcal{L}^t)^{-1}$ la funzione corrispondente nel dominio corrente:

$$g : \Omega_s^t \times (0,T) \to \mathbb{R}, \qquad g(\mathbf{x}, t) = \hat{g}\,(\mathcal{L}^t)^{-1}(\mathbf{x}, t).$$

Si usa la stessa notazione nel dominio fluido: data $\hat{f} : \Omega_f^0 \times (0,T) \to \mathbb{R}$ nel dominio fluido corrente, si identifica con $\hat{f} = f \circ (\mathcal{A}^t)$ la funzione corrispondente nel dominio di riferimento:

$$\hat{f} : \Omega_f^0 \times (0,T) \to \mathbb{R}, \qquad \hat{f}(\mathbf{x}_0, t) = f(\mathcal{A}^t(\mathbf{x}_0, t)).$$

Attraverso questa notazione si definisce la derivata ALE in tempo come:

$$\partial_t f|_{\mathbf{x}_0} : \Omega_f^t \times (0,T) \to \mathbb{R}, \qquad \partial_t f|_{\mathbf{x}_0}(\mathbf{x}, t) = \partial_t \hat{f} \circ (\mathcal{A}^t)^{-1}(\mathbf{x}). \tag{2.5}$$

Infine grazie alla derivata temporale ALE si determina la velocità del dominio fluido:

$$\mathbf{w}(\mathbf{x}, t) = \partial_t \mathbf{x}|_{\mathbf{x}_0} = \partial_t \mathcal{A}^t \circ (\mathcal{A}^t)^{-1}(\mathbf{x}).$$

Dunque, considerando la (2.4), si ha che la velocità dei punti del dominio fluido è data da:

$$\hat{\mathbf{w}}(\mathbf{x}_0, t) = \mathrm{Ext}(\partial_t \hat{\boldsymbol{\eta}}(\mathbf{x}_0, t)|_{\Gamma^0})$$

2.2 Il problema Fluido-Struttura

Si suppone che il solido sia un materiale elastico, caratterizzato da un legame costitutivo che lega il tensore degli sforzi di Cauchy $\mathbf{T}_s$ al gradiente della deformazione $\mathbf{F}(\hat{\boldsymbol{\eta}}) = \mathbf{I} + \nabla\hat{\boldsymbol{\eta}}$. Il tensore di Cauchy per la struttura è dato da:

$$\mathbf{T}_s(\hat{\boldsymbol{\eta}}) = \lambda\,(\epsilon(\hat{\boldsymbol{\eta}}))\mathbf{I} + 2m\epsilon(\hat{\boldsymbol{\eta}}), \qquad \epsilon(\hat{\boldsymbol{\eta}}) = \frac{\nabla\hat{\boldsymbol{\eta}} + \nabla^T\hat{\boldsymbol{\eta}}}{2}$$

con $\lambda = \dfrac{E\nu}{(1+\nu)(1-2\nu)}$ e $m = \dfrac{E}{2(1+\nu)}$ la constanti di Lamè, E il modulo di Young e ν il coefficiente di Poisson.

Inoltre si suppone che il fluido sia omogeneo, Newtoniano e incomprimibile. Si indichi con $\mathbf{T}_f$:

$$\mathbf{T}_f(\mathbf{u}, p) = -p\mathbf{I} + 2\mu\mathbf{G}(\mathbf{u})$$

il tensore di sforzo di Cauchy, dove $p = p(t, \mathbf{x})$ è la pressione, μ la viscosità dinamica e

$$\mathbf{G}(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)$$

è il tensore di velocità di deformazione. Per descrivere il problema fluido si usano le equazioni di Navier-Stokes per un dominio mobile, utilizzando l'approccio ALE. Grazie alla (2.5) la derivata ALE della velocità è:

$$\partial_t \mathbf{u}|_{\mathbf{x}_0} = \partial_t \mathbf{u} + \mathbf{w} \cdot \nabla\mathbf{u}.$$

Quindi il problema fluido-struttura in forma forte diventa:

$$\begin{cases} \rho_f \partial_t \mathbf{u}|_{\mathbf{x}_0} + \rho_f (\mathbf{u} - \mathbf{w}) \cdot \nabla \mathbf{u} - \nabla \cdot \mathbf{T}_f = \mathbf{f}_f & \text{in} \quad (0,T) \times \Omega_f^t \\ \nabla \cdot \mathbf{u} = 0 & \text{in} \quad (0,T) \times \Omega_f^t \\ \rho_s \dfrac{\partial \hat{\boldsymbol{\eta}}^2}{\partial t^2} - \nabla \cdot \hat{\mathbf{T}}_s = \hat{\mathbf{f}}_s & \text{su} \quad (0,T) \times \Omega_s^0 \\ \mathbf{u} = \dfrac{\partial \mathbf{u}}{\partial t} & \text{su} \quad (0,T) \times \Gamma^t \\ \mathbf{T}_s \cdot \mathbf{n}_s + \mathbf{T}_f \cdot \mathbf{n}_f = 0 & \text{su} \quad (0,T) \times \Gamma^t \end{cases} \tag{2.6}$$

dove ρ_f e ρ_s sono la densità del fluido e della struttura e $\mathbf{f}_f \in L^2(\Omega_f^t)$ e $\hat{\mathbf{f}}_s \in L^2(\Omega_s^t)$ i termini forzanti. Le equazioni $(2.6)_4$ e $(2.6)_5$ sono le due condizioni di interfaccia: la prima definisce la *continuità della velocità del fluido e della struttura* dovuta alla condizione di aderenza, mentre la seconda identifica la *condizione di sforzo*, che rispecchia il principio di azione-reazione. Inoltre, bisogna aggiungere una *condizione sulla geometria* che determina dal calcolo dello spostamento del dominio fluido attraverso la mappa fluida (2.4)

$$\mathcal{A}^t(\mathbf{x}_0) = \mathbf{x}_0 + \mathrm{Ext}(\hat{\boldsymbol{\eta}}|_{\Gamma^0}), \quad \mathbf{w} = \partial_t \mathcal{A}^t \circ (\mathcal{A}^t)^{-1}, \quad \Omega_f^t = \mathcal{A}^t(\Omega_f^t) \tag{2.7}$$

Infine il sistema (2.6) - (2.7) viene chiuso con opportune condizioni di bordo su $\partial\Omega^t \setminus \partial\Gamma$. Sia Γ_N^f l'unione delle sezioni artificiali cioè i bordi che non corrispondono alla parete fisica. Nel caso trattato si considera su Γ_N^f una condizione di Neumann non omegenea ossia:

$$\mathbf{T}_f \cdot \mathbf{n} = \mathbf{h} \quad \text{su} \quad \Gamma_N^f.$$

Per definire la formulazione variazionale del problema fluido-struttura (2.6)-(2.7) si indica con $L^2(\Omega)$ lo spazio delle funzioni a quadrato sommabili nel dominio spaziale Ω, $H^1(\Omega)$ lo spazio delle funzioni in $L^2(\Omega)$ con derivata prima in $L^2(\Omega)$. Si definiscono i seguenti spazi, per un dato $t \in [0,T)$:

$$\begin{aligned} \mathbf{V}_f^t &= \{\mathbf{v} \in \mathbf{H}^1(\Omega_f^t)^d\} \\ Q^t &= L^2(\Omega^t) \\ \hat{\mathbf{V}}_s^0 &= \{\boldsymbol{\chi} \in \mathbf{H}^1(\Omega_s^0)^d\} \end{aligned}$$

si riscrivono il termine diffusivo e convettivo del fluido secondo la seguente notazione:

$$a(\mathbf{w}, \mathbf{v}) := \mu(\nabla \mathbf{w} + (\nabla \mathbf{w}^T), \nabla \mathbf{v}), \quad c(\boldsymbol{u}, \mathbf{v}, \mathbf{w}) := \int_{\Omega_t^f} (\boldsymbol{u} \cdot \nabla \mathbf{v}) \cdot \mathbf{w} d\Omega$$

dove $(\cdot,\cdot)$ è il prodotto interno in $\mathbf{L}^2(\Omega_f^t)$. Si definisca il seguente spazio funzionale:

$$\mathbf{S}^t = \{(\mathbf{v}, \boldsymbol{\chi}) \in \mathbf{V}_f^t \times \hat{\mathbf{V}}_s^0 : \mathbf{v}|_{\Gamma^t} = \boldsymbol{\chi}|_{\Gamma^0} \circ \mathcal{L}^{-1}\}$$

La formula variazionale per il problema fluido-struttura è:
dato $t \in (0,T)$ *trovare* $(\boldsymbol{u}, \boldsymbol{\eta}) \in \boldsymbol{S}^t$, $p \in Q^t$ *tale che*:

$$\begin{gathered} \rho_f(\partial_t \mathbf{u}|_{\mathbf{x}_0}, \mathbf{v})_{\Omega_f^t} + a(\mathbf{u}, \mathbf{v})_{\Omega_f^t} - (p, \nabla \cdot \boldsymbol{v})_{\Omega_f^t} + c(\boldsymbol{u} - \boldsymbol{w}, \boldsymbol{u}, \boldsymbol{v})_{\Omega_f^t} \\ +\rho_s h_s(\partial_{tt} \boldsymbol{\eta}, \boldsymbol{\chi})_{\Omega_s^0} + (\mathbf{T}_s, \nabla \boldsymbol{\chi})_{\Omega_s^0} - (\nabla \cdot \boldsymbol{u}, q)_{\Omega_f^t} = (\mathbf{f}_f, \boldsymbol{v})_{\Omega_f^t} - (\mathbf{h}, \mathbf{v})_{\Gamma_N^t} \\ = (\mathbf{f}_f, \boldsymbol{v})_{\Omega_f^t} - (\mathbf{h}, \mathbf{v})_{\Gamma_N^t} + (\hat{\mathbf{f}}_s, \boldsymbol{\chi})_{\Omega_s^0} \\ \mathbf{u}|_{\Gamma^t} = \frac{\partial \boldsymbol{\eta}}{\partial t}|_{\Gamma^0} \circ \mathcal{L}^{-1} \end{gathered} \tag{2.8}$$

per ogni $(\boldsymbol{v}, \boldsymbol{\chi}) \in \mathbf{S}^f(t)$ e $q \in Q^t$. La continuità della velocità è imposta in forma forte attraverso la $(2.8)_3$. Al contrario, la continuità dello sforzo all'interfaccia è soddisfatto in forma debole. Lo sforzo all'interfaccia fluida può essere interpretato come il residuo della forma debole dell'equazione del momento considerando una funzione test fluida che non si annulla su Γ^t:

$$\begin{aligned}(\mathbf{T}_f \cdot \mathbf{n}_f, \boldsymbol{v})_{\Gamma_t} = \rho_f(\partial_t \boldsymbol{u}|_{\mathbf{x}_0}, \boldsymbol{v}) + a(\mathbf{u}, \mathbf{v})_{\Omega_f^t} + c(\boldsymbol{u} - \boldsymbol{w}, \boldsymbol{u}, \boldsymbol{v})_{\Omega_f^t} \\ -(p, \nabla \cdot \boldsymbol{v}) - (\mathbf{f}_f, \boldsymbol{v})_{\Omega_f^t} - (\mathbf{h}, \mathbf{v})_{\Gamma_N^t} = -(\mathcal{R}_f(\boldsymbol{u}, p), \boldsymbol{v})_{\Omega_f^t}\end{aligned} \tag{2.9}$$

Infine dato il seguente spazio funzionale

$$\mathbf{M}^0 = \{\boldsymbol{\psi} \in \mathbf{H}^1(\Omega^0) : \boldsymbol{\psi}|_{\Gamma^t} = \mathbf{0}, \quad \boldsymbol{\psi} \cdot \mathbf{n}_f|_{\Gamma_N^t} = 0\}$$

si può definire la formulazione debole del problema armonico:
Trovare $\mathbf{w} \in \mathbf{M}^0$ *tale che*:

$$\begin{cases} (\nabla \hat{\mathbf{w}}, \nabla \boldsymbol{\psi})_0 = 0 \quad \forall \boldsymbol{\psi} \in \mathbf{M}^0 \\ \hat{\mathbf{w}} = \mathbf{u} \circ \mathcal{A}^t \end{cases} \tag{2.10}$$

2.3 Discretizzazione in tempo

In questa sezione si considerano gli algoritmi basati sulla soluzione di sottoproblemi fluidi e struttura (ossia *procedure di partizioni*). Ogni sottoproblema è risolto separatamente, permettendo di riutilizzare i codici e i solutori esistenti. Per imporre la continuità della velocità e dello sforzo normale all'interfaccia, si può considerare una *strategia debolmente accoppiata*, in cui si risolve il fluido e la struttura un'unica volta (o un ridotto numero di volte) per passo temporale. In questo modo il problema accoppiato non è risolto esattamente; inoltre non c'è un completo bilanciamento energetico tra i due sottoproblemi e questo può produrre delle instabilità negli schemi numerici. Ad esempio come mostrato in [3] un accoppiamento esplicito è instabile quando gli effetti della massa aggiunta sono significativi, come in emodinamica. Alternativamente, si può utilizzare un trattamento implicito (*forte*) delle condizioni ad ogni passo temporale, producendo un sistema monolitico di equazioni non lineari. Diverse strategie sono state proposte per risolvere il problema monolitico. Un esempio si possono utilizzare iterazioni di Picard o di Newton che per risolvere le non linearità del problema fluido-struttura (la determinazione della posizione dell'interfaccia e il calcolo del termine convettivo) come mostrato in [27], [12]. Una terza possibilità consiste nel trattare la posizione dell'interfaccia e il termine convettivo in modo esplicito attraverso l'estrapolazione delle informazioni dal passo temporale precedente, è trattare in implicito le condizioni d'interfaccia, sottoiterando fra i due sottoproblemi Questo algoritmo è *semi-implicito* (vedere [11], [26]). In questi casi, si genera un bilanciamento tra i due sottoproblemi e lo schema numerico è stabile. Tuttavia il numero di sottoiterazioni elevato, soprattutto quando il termine di massa aggiunta non è trascurabile. In questo lavoro si considera uno schema implicito.

Si discretizzi in tempo il sistema (2.6). Sia Δt il passo temporale e $t^n = n\Delta t$ per $n = 0, ..., N$ e si definisca con z^n l'approssimazione della funzione all'instante di tempo t^n. Si consideri uno schema di Eulero implicito per la distretizzazione in tempo del problema fluido, mentre uno schema punto medio per la struttura (schema del secondo ordine).

Per trattare la posizione del dominio fluido si utilizza un algoritmo di punto fisso sulla posizione dell'interfaccia e sul termine convettivo:
Dato $\boldsymbol{u}^n$, $\hat{\boldsymbol{\eta}}^n$, $\hat{\boldsymbol{\eta}}^{n-1}$ *e* Ω_f^n, *per* $i = 0, 1, ...$ *svolgere i seguenti passi fino a convergenza:*

1. Risolvere il problema fluido-struttura linearizzato:

$$\begin{cases} \rho_f \dfrac{\boldsymbol{u}_{i+1}^n - \boldsymbol{u}_i^n}{\Delta t} + \rho_f(\boldsymbol{u}_i^{n+1} - \boldsymbol{w}_i^{n+1})\nabla \cdot \boldsymbol{u}_{i+1}^{n+1} - \nabla \cdot \mathbf{T}_{f,i+1}^{n+1} = \mathbf{f}_f^{n+1} & \text{in} \quad \Omega_{f,i}^{n+1} \\ \nabla \cdot \boldsymbol{u}_{i+1}^{n+1} = 0 & \text{in} \quad \Omega_{f,i}^{n+1} \\ \rho_s h_s \dfrac{\hat{\boldsymbol{\eta}}_{i+1}^{n+1} - \hat{\boldsymbol{\eta}}_i^n}{\Delta t^2} - \nabla \cdot \hat{\mathbf{T}}_{s,i+1}^{n+1} = \hat{\mathbf{f}}_s^{n+1} & \text{in} \quad \Omega_s^0 \\ \dfrac{\hat{\boldsymbol{\eta}}_{i+1}^{n+1} - \hat{\boldsymbol{\eta}}_i^n}{\Delta t} = \dfrac{\dot{\hat{\boldsymbol{\eta}}}_{i+1}^{n+1} + \dot{\hat{\boldsymbol{\eta}}}_i^n}{2} & \text{in} \quad \Omega_s^0 \\ \boldsymbol{u}_{i+1}^{n+1} = \dfrac{\boldsymbol{\eta}_{i+1}^{n+1} - \boldsymbol{\eta}_i^n}{\Delta t} & \text{su} \quad \Gamma_i^{n+1} \\ \mathbf{T}_{s,i+1}^{n+1} \cdot \mathbf{n}_s + \mathbf{T}_{f,i+1}^{n+1} \cdot \mathbf{n}_f = 0 & \text{su} \quad \Gamma_i^{n+1} \end{cases} \tag{2.11}$$

2. Aggiornare il dominio fluido:

$$\begin{cases} \mathcal{A}_{i+1}^{n+1}(\mathbf{x}_0) = \mathbf{x}_0 + \mathrm{Ext}(\hat{\boldsymbol{\eta}}_{i+1}^{n+1}|_{\Gamma^0}) \\ \mathbf{w}_{i+1}^{n+1} = \dfrac{(\mathcal{A}_{i+1}^{n+1} - \mathcal{A}_i^n) \circ (\mathcal{A}_{i+1}^{n+1})^{-1}}{\Delta t}, \quad \Omega_{f,i+1}^{n+1} = \mathcal{A}_{i+1}^{n+1}(\Omega_f^0) \end{cases} \tag{2.12}$$

Il sistema così definito è ancora accoppiato attraverso le condizioni $(2.11)_5$ e $(2.11)_6$, ma è stato linearizzato sia per quanto riguarda la non linearità geometrica, che nel termine convettivo. Tuttavia la presenza delle condizioni $(2.11)_5$ e $(2.11)_6$ rende necessaria una risoluzione monolitica del problema senza la possibilità di utilizzare codici pre-esistenti. Perciò si considera una procedura di partizione per la soluzione del problema linearizzato. Questa strategia è in grado di scomporre il problema lineare fluido-struttura in due sottoproblemi separati uno struttura e uno struttura.

2.4 Procedura di partizione

La procedura di partizione più comunemente usata è l'algoritmo di Dirichlet-Neumann, che consiste nel risolvere il sottoproblema fluido con una condizione diinterfaccia di Dirichlet e il sottoproblema struttura con una condizione di interfaccia di neumann.
Dato $\boldsymbol{\eta}^n$, $\boldsymbol{\eta}^{n-1}$, $\boldsymbol{u}^n$ *e la corrente iterazione* $\boldsymbol{\eta}_i^{n+1}$ *di punto fisso, trovare la nuova iterazione* $\boldsymbol{\eta}_{i+1}^{k+1}$, $\boldsymbol{u}_{i+1}^{k+1}$ *e* p_{i+1}^{k+1} *tale che:*

1. risolva il problema fluido:

$$\begin{cases} \rho_f \frac{\boldsymbol{u}_{i+1}^{k+1} - \boldsymbol{u}^{n+1}}{\Delta t} + \rho_f(\boldsymbol{u}_i^{n+1} - \boldsymbol{w}_i^{k+1}) \cdot \boldsymbol{u}_{i+1}^{n+1} - \nabla \cdot \mathbf{T}_{f,i+1}^{k+1} = \mathbf{f}_f^{n+1} & \text{in} \quad \Omega_{f,i}^{n+1} \\ \nabla \cdot \boldsymbol{u}_{i+1}^{k+1} = 0 & \text{in} \quad \Omega_{f,i}^{n+1} \\ \boldsymbol{u}_{i+1}^{k+1} = \frac{\boldsymbol{\eta}_i^{n+1} - \boldsymbol{\eta}^n}{\Delta t} & \text{su} \quad \Gamma_s^{n+1} \end{cases} \tag{2.13}$$

2. risolva problema struttura con condizioni di Neumann

$$\begin{cases} \rho_s h_s \dfrac{\hat{\boldsymbol{\eta}}_{i+1}^{k+1} - \hat{\boldsymbol{\eta}}^n}{\Delta t^2} - \nabla \cdot \hat{\mathbf{T}}_{s,i+1}^{k+1} = \hat{\mathbf{f}}_s^{n+1} & \text{in} \quad \Omega_s^0 \\ \mathbf{T}_{s,i+1}^{k+1} \cdot \mathbf{n}_s + \mathbf{T}_{f,i+1}^{k+1} \cdot \mathbf{n}_f = 0 & \text{su} \quad \Gamma_i^{n+1} \end{cases} \tag{2.14}$$

Per ogni iterazione di punto fisso (2.11) è stato usato un approccio ad un *unico loop*, ovvero è stata fatta una sola iterazione di (2.13)-(2.14) per ogni iterazione di punto fisso. Una volta soddisfatto un opportuno test di convergenza, si risolve il problema (2.12) ottenendo il nuovo dominio.

Per alcuni problemi, l'algoritmo di Dirichlet-Neumann per poter convergere ha bisogno di un metodo di rilassamento e di un elevato numero di iterazioni se il fluido e la struttura hanno densità simili (effetto della massa aggiunta.) Il rilassamantento consiste nel determinare una *nuova* posizione $\hat{\boldsymbol{\eta}}_{i+1,new}^{k+1}$ attraverso una combinazione lineare della posizione al passo corrente $\hat{\boldsymbol{\eta}}_{i+1}^{k+1}$ con la posizione del passo precedente $\hat{\boldsymbol{\eta}}_i^{n+1}$:

$$\hat{\boldsymbol{\eta}}_{i+1,new}^{k+1} = \omega \hat{\boldsymbol{\eta}}_{i+1}^{k+1} + (1-\omega)\hat{\boldsymbol{\eta}}_i^{n+1}$$

con ω opportunamente scelto.

Una procedura alternativa al metodo di Dirichlet-Neumann è il partizionamento di Neumann-Dirichlet; tuttavia questo schema non possiede delle buona proprietà di convergenza (si veda [3]). In [7] viene presentato un algoritmo di Neumann-Neumann (NN), tuttavia le proprità di convergenza del NN riportate nell'articolo citato non si discostano significativamente dai risultati ottenuti con il metodo di Dirichlet-Neumann.

2.4.1 Procedure alternative

Altre possibili procedure di partizione considerano una combinazione lineare delle condizioni di continutità di velocità e sforzo all'interfaccia, così da determinare una condizione di Robin all'interfaccia. Questa condizione di interfaccia genera una famiglia di procedure di partizione con lo scopo di ottenere delle proprietà di convergenza migliori.
Le condizioni di interfaccia per un generare un algoritmo di Robin-Robin sono:

$$\begin{cases} \alpha_f \boldsymbol{u}_{i+1}^{n+1} + \mathbf{T}_{f,i+1}^{n+1} \cdot \mathbf{n}_f = \alpha_f \frac{\boldsymbol{\eta}_i^{n+1} - \boldsymbol{\eta}^n}{\Delta t} - \mathbf{T}_s^{n+1} \cdot \mathbf{n}_s & \text{su} \quad \Gamma_i^{n+1} \\ \alpha_s \boldsymbol{\eta}_{i+1}^{n+1} + \mathbf{T}_{s,i+1}^{n+1} \cdot \mathbf{n}_s = \frac{\alpha_s}{\Delta t} \boldsymbol{\eta}_{i+1}^{n+1} + \alpha_s \boldsymbol{u}_{i+1}^{n+1} \mathbf{T}_f^{n+1} \cdot \mathbf{n}_f & \text{su} \quad \Gamma_i^{n+1} \end{cases}$$

con $\alpha_s, \alpha_f > 0$. Per deteminare i valori α_f e α_s si possono utilizzare dei modelli ridotti per il fluido e per la struttura (si veda[26], [3]). Una condizione di Robin è una combinazione lineare di una condizione di Dirichlet e di Neumann. Per questo motivo un algoritmo *Robin-Robin*, include oltre all'algoritmo di Dirichelt-Neumann, tutti quei metodi formati da una condizione di Robin come *Dirichlet-Robin*, *Neumann-Robin*, *Robin-Dirichlet* e *Robin-Neumann*, si veda [1]. In [1] è stato mostrato che tra le procedure di partizione presenta dalla famiglia Robin-Robin l'algoritmo di Robin-Neumann determina le proprietà migliori di convergenza. Infatti non sembra sensibile come l'algoritmo di Dirichlet-Neumann agli effetti di massa-aggiunta e converge in un numero ridotto di iterazioni. Inoltre non necessita di una procedura di rilassamento.

2.4.2 Stima del parametro α_f

Per determinare il coefficiente della condizione di Robin si può utilizzare un modello ridotto della struttura (proposto in [26]). Questo modello descrive la struttura come una membrana *sottile* bidimensionale, che occupa la superficie Γ_0

Il modello ridotto è basato sulle seguenti ipotesi:

1. Il legame costitutivo sforzo-deformazione è lineare e il materiale è isotropo-omogeneo.

2. Le deformazioni sono piccole, e le sezioni normali alla superficie si mantengono tali dopo la deformazione;

Questo modello restringe il movimento della membrana solamente lungo la direzione normale, imponendo che lo spostamento della struttura sia solamente della forma $\boldsymbol{\eta} = (0, 0, \eta)$ rispetto a un riferimento locale. Da queste ipotesi si trova il seguente modello per la membrana

$$\begin{cases} \rho_s h_s \dfrac{\partial^2 \eta}{\partial t^2} + \beta\eta = f_s & \text{in} \quad (0,T) \times \Gamma^0 \\ \eta|_{t=0} = \eta_0 & \text{in} \quad \Gamma^0 \\ \dfrac{\partial \eta}{\partial t} = \eta_v & \text{in} \quad \Gamma^0 \end{cases}$$

dove η_0 e η_v sono le condizioni iniziali, e β è *il parametro algebrico*, definito nel modo seguente:

$$\beta(\xi_1, \xi_2) = \frac{h_s E}{1-\nu^2}(4\rho_1^2 - 2(1-\nu)\rho_2), \tag{2.15}$$

dove ρ_1 è la curvatura media, mentre ρ_2 è la curvatura Gaussiana della superficie Γ_0. Al fine di semplificare la trattazione si adotta la seguente notazione $T_f = (\mathbf{T}_f \cdot \mathbf{n}) \cdot \mathbf{n}$ e $z = \mathbf{z} \cdot \mathbf{n}$ dove $\mathbf{z}$ è una funzione data.
Le condizioni di interfacciadopo discretizzazione in tempo sono:

$$\begin{cases} u^{n+1} = \dfrac{\eta^{n+1} - \eta^n}{\Delta t} \\ T_f^{n+1} - f_s^{n+1} = 0 \end{cases} \tag{2.16}$$

Lo schema numerico che si ottiene con queste condizioni di interfaccia è stabile in quanto le condizioni di interfaccia sono trattate implicitamente, mentre il movimento del dominio in modo esplicito. Questo permette di bilanciare lo scambio di energia tra il fluido e la struttura così da garantire la stabilità.
Per semplificare la trattazione si pone $\mathbf{h} = \mathbf{0}$ e $\mathbf{f}_f = \mathbf{0}$. La formulazione debole del problema fluido-struttura per il modello ridotto diventa:

$$\begin{aligned} &\Big(\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\Delta t}, \boldsymbol{v}\Big) + a(\boldsymbol{u}^{n+1}, \boldsymbol{v}) + c(\boldsymbol{u}^n - \boldsymbol{w}^n, \boldsymbol{u}^{n+1}, \boldsymbol{v}) - (p^{n+1}, \nabla \cdot \boldsymbol{v}) \\ &= -\int_{\Gamma_n^f} \Big(\frac{\rho_s h_s}{\Delta t}(\dot\eta^{n+1} - \dot\eta^n) + \beta\frac{\eta^{n+1} + \eta^n}{2}\Big) v d\gamma \\ &= -\int_{\Gamma_n^f} \Big(\frac{2\rho_s h_s}{\Delta t}\big(\frac{\eta^{n+1} - \eta^n}{\Delta t} - \dot\eta^n\big) + \beta\frac{\eta^{n+1} + \eta^n}{2}\Big) v d\gamma \\ &= -\int_{\Gamma_n^f} \Big(\Big(\frac{2\rho_s h_s}{\Delta t^2} + \frac{\beta}{2}\Big)\eta^{n+1} - \frac{2\rho_s h_s}{\Delta t}\big(\frac{\eta^n}{\Delta t} + \dot\eta^n\big) + \frac{\beta}{2}\eta^n\Big) v d\gamma \end{aligned}$$

Avendo utilizzato la $(2.16)_1$ si è così ottenuta una condizione all'interfaccia una condizione al contorno di Robin per la componente normale della velocità del fluido:

$$\Big(T_f^{n+1} + \Big(\frac{2\rho_s h_s}{\Delta t} + \frac{\beta\Delta t}{2}\Big)u^{n+1}\Big)\Big|_{\Gamma_n^f} = -\Big(\Big(\frac{2\rho_s h_s}{\Delta t^2} + \frac{\beta}{2}\Big)\eta^n - \frac{\rho_s h_s}{\Delta t^2}\eta^{n-1} + \frac{\beta}{2}\eta^n\Big) \tag{2.17}$$

Da questa analisi, si suggerisce di prendere un coefficiente α_f per la condizione di Robin pari a

$$\alpha_f = \frac{2\rho_s h_s}{\Delta t} + \frac{\beta\Delta t}{2}$$

e sostituendo a β il valore (2.15) si ottiene:

$$\alpha_f = \frac{2\rho_s h_s}{\Delta t} + \frac{h_s E\Delta t}{1-\nu^2}(2\rho_1^2 - (1-\nu)\rho_2). \tag{2.18}$$

Si noti che è stato utilizzato uno schema del primo ordine per trattare la condizione di continuità $(2.6)_4$, mentre una condizione del secondo per la discretizzazione in tempo. Si potrebbe usare una discretizzazione del secondo ordine anche per la $(2.17)_1$ come ad esempio:

$$u^{n+1} = \frac{3\eta^{n+1} - 4\eta^n + \eta^{n-1}}{2\Delta t}.$$

In questo modo di otterrebbe il seguente α_f

$$\alpha_f = \frac{3\rho_s h_s}{\Delta_t} + \frac{\beta}{2}.$$

Allo stesso modo si può ricavare il valore di α_f nel caso si usi una discretizzazione in tempo della struttura BDF del primo ordine e la 2.17_1

$$\alpha_f = \frac{\rho_s h_s}{\Delta t} + \frac{h_s E\Delta t}{1-\nu^2}(\frac{\rho_1^2}{2} - (1-\nu)\rho_2) \tag{2.19}$$

In [1] si è proposto di utilizzare il valore di α_f ottenuro dal modello ridotto. Si noti come in [1] si sia considerata solo per il caso (2.19), esteso dagli altri due schemi in questa tesi.

2.5 Formulazione debole del problema

Si consideri la formulazione semi-distreta $(2.8)_1$ e si discretizzi in spazio secondo il metodo di Galerkin ad elementi finiti. Sia $\hat{\mathcal{T}}_h$ una triangolazione della configurazione fluida di riferimento e sia $\hat{\mathcal{A}}$ la mappa ALE per ogni $t > 0$. Si definisca $\mathcal{T}_h(t) = \mathcal{A}(\hat{\mathcal{T}}_h)$ la triangolazione del dominio corrente fluido. Si rappresenti con h la massima grandezza degli elemeni di $\mathcal{T}_h$. La triangolazione $\mathcal{T}_h$ induce una partizione $\mathcal{I}_h$ su Γ^t.

Allo stesso modo si costruisca una triangolazione sul dominio struttura di riferimento Ω_s^0 $\mathcal{T}_H$, dove H è la dimensione massima della triangolazione struttura. $\mathcal{T}_H$ determina una partiazione $\mathcal{I}_H$ su Γ^0. Le due partizioni su Γ^t sono tra di loro indipendenti, ma per semplicità di trattazione qui si considera il caso in cui le due griglie siano coincidenti $\mathcal{I}_h \equiv \mathcal{I}_H$. Siano $\hat{\mathbf{S}}_h^t$, Q_h^t, $\mathbf{M}_h^0$, $\mathbf{W}_h^0$ gli spazi

ad elementi finiti approssimazione di $\hat{\mathbf{S}}^t$, Q^t, $\mathbf{M}^0$, $\mathbf{W}^0$. Si introducono le basi di Lagrange $\{\boldsymbol{\phi}_i\}_{\mathcal{N}_f} \oplus \{\boldsymbol{\phi}_j\}_{\mathcal{N}_f}$ $\{\pi_i\}_{\mathcal{N}_p}$ $\{\boldsymbol{\psi}_j\}_{\mathcal{N}_s}$ associate rispettivamente a $\mathbf{S}_h$, , $\hat{Q}_h$, Q_h, $\mathbf{M}_h^0$, $\mathbf{W}_h$. $\mathcal{N}_\Gamma$ rappresenta l'insieme dei nodi di velocità su Γ e $\mathcal{N}_f$ i restanti nodi di velocità, mentre i nodi di pressione e di velocità della struttura sono indicati con $\mathcal{N}_p$ e $\mathcal{N}_s$. La formulazione debole alla Galerkin del problema è:
Trovare $(\boldsymbol{u}_h^{n+1}, \boldsymbol{\chi}_h^{n+1}) \in \boldsymbol{S}_h^t$ *e* $p_h^{n+1} \in Q_h$ *tali che:*

$$\begin{cases} \rho_f(\partial_t \mathbf{u}_h^{n+1}|_{\mathbf{x}_0}, \mathbf{v}_h)_{\Omega_f^{n+1}} + a(\mathbf{u}_h^{n+1}, \mathbf{v}_h)_{\Omega_f^{n+1}} + c((\mathbf{u}_h^{n+1} - \mathbf{w}_h^{n+1}), \boldsymbol{u}_h^{n+1}, \mathbf{v}_h)_{\Omega_f^{n+1}} \\ \quad +(\nabla \cdot \boldsymbol{u}_h^{n+1}, q)_{\Omega_f^{n+1}} - (p_h^{n+1}, \nabla \cdot \mathbf{v}_h)_{\Omega_f^{n+1}} + \rho_s\Big(\dfrac{\dot{\boldsymbol{\eta}}_h^{n+1} - \dot{\boldsymbol{\eta}}_h^n}{\Delta t}, \boldsymbol{\chi}_h\Big)_{\Omega_s^0} \\ +\Big(\mathbf{T}_s\Big(\frac{\boldsymbol{\eta}_h^{n+1}+\boldsymbol{\eta}_h^n}{2}\Big), \nabla\boldsymbol{\chi}_h\Big)_{\Omega_s^0} = (\mathbf{f}_s^{n+1}, \boldsymbol{\chi}_h)_{\Omega_s^0} + (\mathbf{f}_f^{n+1}, \mathbf{v}_h)_{\Omega_f^{n+1}} + \displaystyle\int_{\Gamma_N^t} \mathbf{h} \cdot \mathbf{v}_h d\gamma \\ \Big(\dfrac{\dot{\boldsymbol{\eta}}_h^{n+1} + \dot{\boldsymbol{\eta}}_h^n}{2}, \boldsymbol{\chi}_h\Big)_{\Omega_0} = \Big(\dfrac{\boldsymbol{\eta}_h^{n+1} - \boldsymbol{\eta}_h^n}{\Delta t}, \boldsymbol{\chi}_h\Big)_{\Omega_0} \end{cases} \tag{2.20}$$

Per ogni $(\mathbf{v}_h, \boldsymbol{\chi}_h) \in \mathbf{S}_h^f, q_h \in Q_h$
Il problema geometrico diventa:

$$\begin{cases} \mathcal{A}_{t^{n+1}}(\boldsymbol{x}_0) = \boldsymbol{x}_0 + \mathrm{Ext}_h(\boldsymbol{\eta}_h^{n+1}|_{\Gamma^0}) \\ \boldsymbol{w}_h^{n+1} = \dfrac{(\mathcal{A}_h^{n+1} - \mathcal{A}_h^n) \circ (\mathcal{A}_h^{n+1})^{-1}}{\Delta t} \end{cases}$$

dove $\mathrm{Ext}_h(\cdot)$ è la versione discreta dell'operatore $\mathrm{Ext}(\cdot)$.

2.6 Formulazione Algebrica

Definendo $\boldsymbol{U}_f^{n+1}$, $\boldsymbol{U}_\Gamma^{n+1}$, i vettori dei valori nodali per la velocità del fluido dei nodi interni e dei nodi all'interfaccia, e $\boldsymbol{P}^{n+1}$ il vettore dei valori nodali di pressione per il problema fluido si riformula l'approssimazione del fluido rispetto alle quantità incognite:

$$\boldsymbol{u}_h^{n+1}(\boldsymbol{x}, t^{n+1}) = \sum_{i \in \mathcal{N}_f} \boldsymbol{\phi}_i(\boldsymbol{x}, t^{n+1})(\boldsymbol{U}^{n+1})_i + \sum_{j \in \mathcal{N}_\Gamma} \boldsymbol{\phi}_j(\boldsymbol{x}, t^{n+1})(\boldsymbol{U}_\Gamma^{n+1})_j$$
$$p_h^{n+1}(\boldsymbol{x}, t^{n+1}) = \sum_{k \in \mathcal{N}_p} \pi_k(\boldsymbol{x}, t^{n+1})(\boldsymbol{P}^{n+1})_k$$

le funzioni base variano nel tempo secondo il seguente principio:

$$\boldsymbol{\phi}_i(\boldsymbol{x}, t^n) = \mathcal{A}_t^n(\hat{\boldsymbol{\phi}}_i(\boldsymbol{x}_0))$$

dove $\hat{\boldsymbol{\phi}}_i(\boldsymbol{x}_0)$ sono le funzioni base definite sulla griglia del dominio di riferimento $\hat{\mathcal{T}}_h(t)$. Allo stesso modo si riformula il problema struttura:

$$\boldsymbol{\eta}_h^{n+1}(\boldsymbol{x}_0) = \sum_{i \in \mathcal{N}_s} \phi_k(\boldsymbol{x}_0)(\boldsymbol{D}^{n+1})_i$$
$$\dot{\boldsymbol{\eta}}_h^{n+1}(\boldsymbol{x}_0) = \sum_{j \in \mathcal{N}_s} \phi_k(\boldsymbol{x}_0)(\dot{\boldsymbol{D}}^{n+1})_j$$

dove $\boldsymbol{D}^{n+1}$ e $\dot{\boldsymbol{D}}^{n+1}$ sono i vettori dei valori nodali per $\boldsymbol{\eta}_h^{n+1}$ e $\dot{\boldsymbol{\eta}}_h^{n+1}$. Dalla conformità delle mesh si ha: $\mathcal{E}_t\psi_i = \phi_i^\Gamma$ per $i \in \mathcal{N}_s$ con $\mathcal{N}_s \equiv \mathcal{N}_\Gamma$.

Dalla condizione di continuità $(2.6)_4$ si ha:

$$\boldsymbol{U}_\Gamma^{n+1} = \frac{\boldsymbol{D}_\Gamma^{n+1} - \boldsymbol{D}_\Gamma^n}{\Delta t}$$

Per scrivere la discretizzazione completa del problema accoppiato ad un dato instante di tempo, si devono definire un insieme di matrici. Si introducono gli indici α e β necessari a identificare il fluido f, della struttura s e dell'interfaccia Γ. Si usa le lettere minuscole per identificare gli elementi delle matrici $k_{\alpha\beta} = K_{\alpha\beta}^{i,j}$ con $i \in \mathcal{N}_\alpha$ e $j \in \mathcal{N}_\beta$. Si definiscono quindi le matrici $K_{\alpha\beta}$, $M_{\alpha\beta}$, $C_{\alpha\beta}$, $G_{\alpha\beta}$ e $S_{\alpha\beta}$ di elementi

$$\begin{aligned}
k_{\alpha\beta} &:= (\nabla\boldsymbol{\phi}_i, \nabla\boldsymbol{\phi}_j)_{\Omega^f_{t^{n+1}}} + c(\boldsymbol{u}_h^{n+1} - \boldsymbol{w}_h^{n+1}, \boldsymbol{\phi}_i, \boldsymbol{\phi}_j)_{\Omega^f_{t^{n+1}}} \\
m_{\alpha\beta} &:= (\boldsymbol{\phi}_i, \boldsymbol{\phi}_j)_{\Omega^f_{t^{n+1}}}, \qquad c_{\alpha\beta} := \frac{1}{\Delta t} m_{\alpha\beta} + k_{\alpha\beta} \\
g_{\alpha\beta} &:= -(\nabla\phi_i, \pi_j)_{\Omega^f_{t^{n+1}}},
\end{aligned}$$

con $G_{\alpha\beta} = D_{\alpha\beta}^T$ e si identifica con M_Γ la matrice di massa sull'interfaccia. All'instante di tempo t^{n+1} si può riscrivere il sistema (2.20) in forma matriciale:

$$A\mathbf{X}^{n+1} = \mathbf{b}^{n+1} \tag{2.21}$$

dove:

$$A = \begin{bmatrix} C_{ff} & G_f & C_{f\Gamma} & 0 & 0 \\ D_f & 0 & D_\Gamma & 0 & 0 \\ 0 & 0 & M_\Gamma & -M_\Gamma/\Delta t & 0 \\ C_{\Gamma f} & G_\Gamma & C_{\Gamma\Gamma} & S_{\Gamma\Gamma} & S_{\Gamma s} \\ 0 & 0 & 0 & S_{s\Gamma} & S_{ss} \end{bmatrix},$$

$$\mathbf{X}^{n+1} = \begin{bmatrix} \boldsymbol{U}_f^{n+1} \\ \boldsymbol{P}^{n+1} \\ \boldsymbol{U}_\Gamma^{n+1} \\ \boldsymbol{D}_\Gamma^{n+1} \\ \boldsymbol{D}_s^{n+1} \end{bmatrix}, \qquad \mathbf{b}^{n+1} = \begin{bmatrix} \mathbf{b}_f^{n+1} \\ \mathbf{0} \\ -M_\Gamma \boldsymbol{D}_\Gamma^n/\Delta t \\ \mathbf{b}_\Gamma^{n+1} \\ \mathbf{b}_s^{n+1} \end{bmatrix}.$$

Si identifica con $\mathbf{b}^{n+1}$ il termine forzante della struttura che comprende i termini relativi alla discretizzazione in tempo. Le prime due righe sono la completa discretizzazione in tempo della conservazione delle quantità di moto e della massa del fluido, la terza identifica la continuità della velocità dell'interfaccia, la quarta riga determina la continuità dello sforzo nella formulazione debole, infine la quinta identifica il problema struttura nei nodi interni.

Se le mesh non sono conformi, si dovrebbero moltiplicare la terza e la quarta riga per una matrice di proiezione (o interpolazione) tra lo spostamento della struttura all'interfaccia e la velocità del fluido (si veda [27]). Ogni procedura di partizionamento può essere riscritta secondo uno schema iterativo a blocchi Gaiuss-Seidel

$$PA\mathbf{X}^{n+1} = P\mathbf{b}^{n+1}$$

dove P è un opportuno precondizionatore. Si definisce la seguente partizione di $\mathbf{X}^{n+1}$

$$\mathbf{X}_f^{n+1} = \begin{bmatrix} \boldsymbol{U}_f^{n+1} \\ \boldsymbol{P}^{n+1} \\ \boldsymbol{U}_\Gamma^{n+1} \end{bmatrix}, \qquad \mathbf{X}_s^{n+1} = \begin{bmatrix} \boldsymbol{D}_\Gamma^{n+1} \\ \boldsymbol{D}_s^{n+1} \end{bmatrix}.$$

Questa scelta suddivide il sitema fluido-struttura in due blocchi *fluido* e *struttura*. Si identificano i seguenti blocchi di PA e $P\mathbf{b}^{n+1}$ come

$$PA := \begin{bmatrix} B_{ff} & B_{fs} \\ B_{sf} & B_{ss} \end{bmatrix}, \qquad P\mathbf{b}^{n+1} := \begin{bmatrix} (P\mathbf{b}^{n+1})_f \\ (P\mathbf{b}^{n+1})_s \end{bmatrix}.$$

Quindi, dato $\mathbf{X}^k$, una procedura a blocchi Gauss-Seidel per la soluzione del sistema (2.21) al passo temporale $n+1$ è ottenuta mediante i seguenti passi di uno schema iterativo: risolvere:

$$B_{ff}\mathbf{X}_f^{k+1} = (P\mathbf{b})_f - B_{fs}\mathbf{X}_f^k \qquad B_{ss}\mathbf{X}_s^{k+1} = (P\mathbf{b})_s - B_{sf}\mathbf{X}_f^k$$

fino a quando non viene soddisfatta un opportuna condizione d'arresto. Fissata una tolleranza ϵ un possibile criterio è il seguente:

$$\frac{||\mathbf{r}^{k+1}||}{||\mathbf{r}^0||} = \frac{||\mathbf{b}^{n+1} - A\mathbf{X}^{n+1}||}{||\mathbf{b}^{n+1} - A\mathbf{X}^0||} < \epsilon$$

Questo criterio richiede il calcolo del residuo per il sistema monolitico (2.21). Si determina ora la procedura di partizione per l'algoritmo di Dirchlet-Neumann e Robin-Neumann.
Nel caso di Dirichlet-Neumann la matrice di precondizionamento è $P_{ND} = I$ la matrice identità e il residuo si riduce a:

$$\mathbf{r}_{DN}^{k+1} = -M_\Gamma \boldsymbol{U}_\Gamma^{k+1} + M_\Gamma \Delta t \boldsymbol{D}_\Gamma^{k+1}$$

Nel caso si usi uno schema di Robin-Neumann la matrice di precondizionamento diventa:

$$P_{RN} = \begin{bmatrix} I & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 \\ 0 & 0 & \alpha_f I & I & 0 \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \end{bmatrix}$$

dove I è la matrice identità. Il residuo di questo metodo è:

$$\mathbf{r}_{RN} = -\alpha_f M_\Gamma \boldsymbol{U}_\Gamma^{k+1} - C_{\Gamma\Gamma} U_\Gamma^{k+1} + M_\Gamma \delta t \boldsymbol{D}_\Gamma^{k+1}$$

2.7 Condizioni assorbenti

In numerosi problemi di fluidodinamica, il bordo, o parte di esso, che individua il dominio computazionale non corrisponde ad un bordo fisico, realmente esistente, ma viene introdotto per individuare la regione di interesse (sezione artificiale). L'introduzione di tali bordi artificiali è in genere fonte di inaccuratezza nel momento in cui si vanno a prescrivere le condizioni al bordo: spesso infatti i

dati a disposizione su tali bordi non sono sufficienti per formulare un problema matematicamente ben posto.

È noto che, anche se il moto del fluido è descritto da equazioni paraboliche, la natura del problema di interazione fluido-struttura nell'ambito dell'emodinamica presenta caratteristiche riconducibili ad un problema iperbolico; in particolare, le onde di pressione attraversano il dominio del fluido. In mancanza d'informazioni sulle sezioni artificiali, l'imposizione di una pressione arbitraria da luogo a riflessione spurie di pressione. Di conseguenza, l'imposizione di una condizione adatta sulle sezioni di ingresso e di uscita del fluido che non generi riflessioni è un problema molto delicato e di fondamentale importanza al fine di ottenere risultati numerici soddisfacenti.

Allo scopo di risolvere il fenomeno legato alle riflessioni spurie, è possibile imporre al bordo una condizione assorbente ottenibile dall'accoppiamento del modello 3D con un modello ridotto 1D, come proposto in [26]. In particolare, riferendoci ad un cilindro elastico come mostrato in Figura 2.5, dove L è la

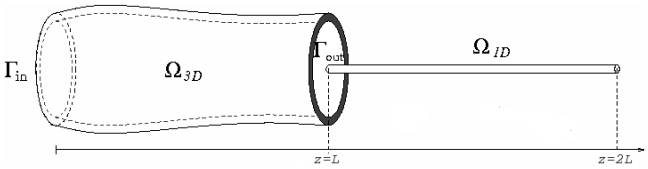

Figura 2.5: Dominio computazionale per il modello accoppiato 3D/1D.

lunghezza, un modello semplificato 1D può essere ottenuto integrando per ogni tempo t le equazioni di Navier-Stokes lungo ogni sezione S normale all'asse z del cilindro. Il modello 1D, per ogni $t > 0$ e $0 < z < L$, assume la forma seguente (si veda [15], [13], [14]):

$$\begin{cases} \dfrac{\partial A}{\partial t} + \dfrac{\partial F}{\partial z} = 0 \\ \dfrac{\partial F}{\partial t} + \dfrac{\partial}{z}\alpha\dfrac{F^2}{A} + A\dfrac{\partial P}{\partial z} + K_R\dfrac{F}{A} = 0 \end{cases} \tag{2.22}$$

dove F è il flusso che attraversa la sezione S, A è l'area di S,P è la pressione media su S,K_R è un parametro che tiene conto della viscosità del fluido, mentre α dipende dalla forma del profilo di velocità del fluido sulla sezione S. Per esempio, la scelta di $\alpha = 1$ corrisponderebbe ad un profilo piatto di velocità. Il sistema (2.22) è costituito da due equazioni in tre incognite (P, F,A). La chiusura del sistema è effettuabile solamente introducendo una terza equazione che descriva la pressione media P in funzione dell'area A al tempo t e di quella al tempo zero A_0 . Si considera il modello algebrico relativo alla struttura, con $\rho_s = 0$, si ottiene

$$P = \frac{\beta}{\pi}(\sqrt{A} - \sqrt{A_0})$$

dove β è data da (2.15) e lo sforzo normale viene approssimato con la pressione in uscita. Il sistema (2.22) risulta essere un sistema iperbolico chiuso in grado di mettere in luce il fenomeno di propagazione delle onde di pressione lungo la direzione dell'asse del cilindro. Il sistema ha due autovalori ed i corrispettivi autovettori sono le variabili caratteristiche del problema, date da (si veda [15],

[27])

$$W_{1,2} = \frac{F}{A} \pm \frac{2\sqrt{2}}{\sqrt{\rho_f}}\Big(\sqrt{P + \beta\sqrt{A_0}} - \sqrt{\beta\sqrt{A_0}}\Big)$$

Nel seguito si propone la derivazione di una condizione al bordo assorbente per le sezioni di ingresso e di uscita del fluido che non richiede l'accoppiamento del modello 3D con il modello 1D, come proposto in [26]: questa può essere ottenuta semplicemente imponendo che la caratteristica entrante nel dominio 3D sia nulla, il che equivale ad imporre che nessuna informazione entri nel dominio. Si osservi che se non venisse considerato l'accoppiamento del modello 3D con quello 1D, non si avrebbe nessun fattore di assorbimento e si creerebbero riflessioni all'interno del dominio fluido. In particolare, per la sezione di uscita, imponendo:

$$W_2|_{\Gamma^1_{out}} = \frac{F}{A} - \frac{2\sqrt{2}}{\sqrt{\rho_f}}\Big(\sqrt{P + \beta\sqrt{A_0}} - \sqrt{\beta\sqrt{A_0}}\Big)\Big|_{\Gamma^t_{f,out}} = 0$$

si ottiene:

$$P|_{\Gamma^t_{f,out}} = \frac{F}{A} - \frac{2\sqrt{2}}{\sqrt{\rho_f}}\Big(\Big(\sqrt{P + \beta\sqrt{A_0}}\Big)^2 - \beta\sqrt{A_0}\Big)\Big|_{\Gamma^t_{f,out}} \tag{2.23}$$

La relazione (2.23) lega la pressione media P al flusso F in corrispondenza della sezione di uscita Γ. Trattando in maniera esplicita il flusso F e interpretando la pressione media espressa nella relazione (2.23) come uno sforzo normale costante nello spazio, come suggerito in [22], si perviene alla seguente *condizione di Neumann absorbente* (si veda [26]):

$$(\mathbf{T}_f^{n+1} \cdot \mathbf{n}_f)|_{\Gamma^n_N} = \Big(\Big(\Big(\frac{\sqrt{\rho_f}}{2\sqrt{2}}\frac{F^n}{A^n} + \sqrt{\beta\sqrt{A_0}}\Big)^2 - \beta\sqrt{A_0}\Big) \cdot \mathbf{n}_f\Big)\Big|_{\Gamma^t_N} \tag{2.24}$$

Si osserva che per una geometria simile a quella rappresentata nella Figura (2.5) l'applicazione alla sezione di uscita della condizione al bordo (2.24) equivale a modellizzare un cilindro infinitamente lungo. Una condizione al bordo assorbente analoga può essere applicata anche per la sezione di ingresso del fluido $\Gamma_{t,in}$ seguendo il medesimo procedimento.

Capitolo 3

Risultati Numerici

In questo capitolo sono presentati i principali risultati numerici riguardanti gli schemi presentati nel secondo capitolo, nel caso particolare si considerino geometrie 3D-3D. La prima sezione considera l'algoritmo di Dirichlet-Neumann su una geometria cilindrica rettilinea, con lo scopo di determinare quale condizione al contorno, tra una di Neumann omogenea e una assorbente (si veda sezione 7 Capitolo 2), sia più adatta da applicare sulla sezione di uscita del fluido. In questo test si osserva che nel caso di una condizione di Neumann omogenea, quando l'onda di pressione che si propaga nel vaso giunge sulla parete di uscita si genera una riflessione di pressione che altera la configurazione della struttura e del fluido. Imponendo una condizione assorbente si nota come tali riflessioni siano praticamente eliminate Dati i migliori risultati ottenuti con le condizioni assorbenti si è deciso di usare queste condizioni nelle successive simulazioni se non diversamente indicato.

Nella seconda sezione si confrontano gli algoritmi Dirichlet-Neumann e Robin-Neumann su una geometria cilindrica, al fine di analizzare le prestazioni dei due schemi, in termini di numero medio di iterazioni dell'algoritmo di punto fisso. I principali test svolti considerano i possibili effetti numerici che si possono verificare:

- Massa aggiunta: molti algoritmi per la risoluzione di problemi di fluido-struttura risentono negativamente dell'effetto di massa aggiunta, ovvero quando le densità dei due problemi sono simili e ciò comporta un aumento del costo computazionale e del numero di sottoiterazioni necessarie a convergere. Questo effetto si manifesta nell'algoritmo di Dirichlet-Neumann, mentre non sembra significativo nell'algoritmo di Robin-Neumann.

- Variazione della tolleranza del metodo di punto fisso: riducendo il valore di ϵ necessario a garantire la convergenza del metodo di punto fisso, si osserva un aumento del numero di sottoiterazioni necessarie a soddisfare la tolleranza richiesta.

- Variazione del passo temporale: riducendo il passo temporale si osserva un aumento del numero di sottoiterazioni necessarie a garantire la convergenza.

Si sottolinea che nell'algoritmo di Robin-Neumann non è stato utilizzato nessun metodo di rilassamento; al contrario per garantire la convergenza dell'algoritmo

di Dirichlet-Neumann, il rilassamento è necessario (si utilizza il metodo di Aitken per determinare il valore di ottimale del parametro di rilassamento). I risultati qui ottenuti mostrano come l'algoritmo di Robin-Neumann, in problemi 3D-3D, sia più robusto e veloce del Dirichlet-Neumann per problemi di fluido-struttura in emodinamica.

La terza sezione applica l'algoritmo di Robin-Neumann su geomtria più complesse, al fine di testarne la bontà e la robustezza. Le geometrie considerate sono un arco aortico di forma toroidale e la geometria di una carotide umana, costruita tramite dati clinici. Tuttavia bisogna segnalare che la velocità di convergenza dell'algoritmo di Robin-Neumann dipende dal parametro di curvatura che nelle geometrie complesse può risultare complicato da determinare in modo analitico. Si possono utilizzare delle approssimazioni di β per applicare l'algoritmo di Robin-Neumann, come nel caso test della carotide dove si è utilizzato un valore di raggio medio della geometria.

3.0.1 Alcuni parametri standard delle simulazioni

Si definiscono i parametri standard usati durante le simulazioni[1]. Il problema fluido viene modellato tramite l'equazione di Navier-Stokes per un fluido incomprimibile, mentre il problema struttura è definito dal modello di lineare e isotropo di Saint-Venant Kirchhoff. I parametri del problema fluido-struttura sono riportati nella Tabella (3.1).

Fluido	
ρ_f	1.0 g/cm^3
ν	0.02 cm^3/s

Struttura	
ρ_s	1.2 g/cm^3
E	3 $\times 10^6$
μ	0.30 $dyne/cm^2$

Tabella 3.1: Parametri del problema fluido: ρ_f defininisce la densità, mentre ν la viscosità cinematica. Parametri del problema struttura: ρ_s identifica la densità, E è il modulo di Young e μ il coefficiente di Poisson.

Il problema fluido è stato discretizzato attraverso lo schema di Eulero implicito, mentre il problema struttura con uno schema di punto medio. Come passo temporale si è scelto $0.001s$ e come istante finale $T = 0.3s$.

Sono state scelte le seguenti condizioni al bordo:

Per il problema armonico: in ingresso e uscita si è applicato uno sforzo tangenziale nullo e uno spostamento normale nullo, mentre all'interfaccia si è imposto lo spostamento della struttura.

Per il problema fluido: in ingresso si è imposto un impulso di pressione in direzione normale di intensità pari a:

$$\Delta P = \begin{cases} 1.324 \cdot 10^4 dyne/cm^2 & \text{per} \quad t < 0.003s \\ 0 & \text{per} \quad t > 0.003s \end{cases} \tag{3.1}$$

che equivale a circa $10mmHg$. In uscita a seconda del test svolto viene applicata o una condizione di sforzo nullo o una condizione assorbente.

[1] Scelte differenti dei parametri verranno segnalate

A seconda dello schema trattato all'interfaccia si è utlizzata la condizione corrispondente.

Nel Problema struttura: si è scelto in ingresso e uscita uno sforzo nullo in direzione tangenziale, mentre una condizione di spostamento nullo lungo la normale. All'interfaccia si è imposto lo sforzo fluido e uno sforzo nullo sulla parete esterna.

Il modello 3D è stato discretizzato in spazio usando elementi finiti *P1-P1 bolla* per il fluido e *P1* per la struttura. Tutte le geometrie fluido-struttura considerate sono state costruite mediante delle mesh conformi all'interfaccia.

3.1 Condizioni Assorbenti in un vaso sanguigno rettilineo

Come test preliminare si considera l'assegnamento di un'opportuna condizione di bordo per la sezione artificiale di uscita del problema fluido. Le condizioni per il problema fluido trattate sono: uno sforzo nullo omogeneo lungo la direzione normale e la condizione assorbente presentata nel secondo capitolo. Lo schema numerico usato in questi test è l'algoritmo di Dirichlet-Neumann e la geometria è quella di un cilindro rettilineo. La condizione assorbente si determina impondo la condizione di Neumann assorbente (2.24) sulla sezione di uscita. Per questo motivo si detemina il parametro geometrico β, mediante la formula (2.15). Segue:

$$\beta = \frac{h_s E}{1-\nu^2}\frac{1}{R^2} \tag{3.2}$$

in quanto nel cilindro ρ_1 e ρ_2 valgono rispettivamente $1/R^2$ e 0 (per il calcolo esplito di β per il cilindro si veda [26]). Si consideri un vaso sanguigno rettilineo di lunghezza $5cm$ e di raggio $0.5cm$. La mesh del modello 3D contiene 4680 tetraedri e 1050 nodi per la mesh fluida, mentre 4800 tetraedri per la mesh solida, rappresentate in Figura 3.1.

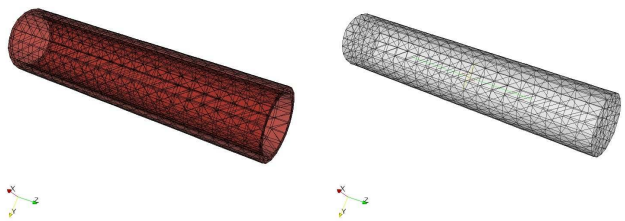

Figura 3.1: A sinistra è raffigurata la mesh solida, mentre a destra la mesh struttura.

Durante le simulazioni numeriche, si osserva che, nel caso del vaso sanguigno rettilineo, il massimo valore dell'impulso giunge sulla parete di uscita all'instante temporale $0.012s$, dopo di che l'impulso viene riflesso dalla superficie di uscita.

Per ridurre la riflessione sulla parete di uscita si è applicata una condizione assorbente. Nelle Figure 3.2-3.4 sono riportati la pressione e il campo di moto per alcuni instanti temporali.

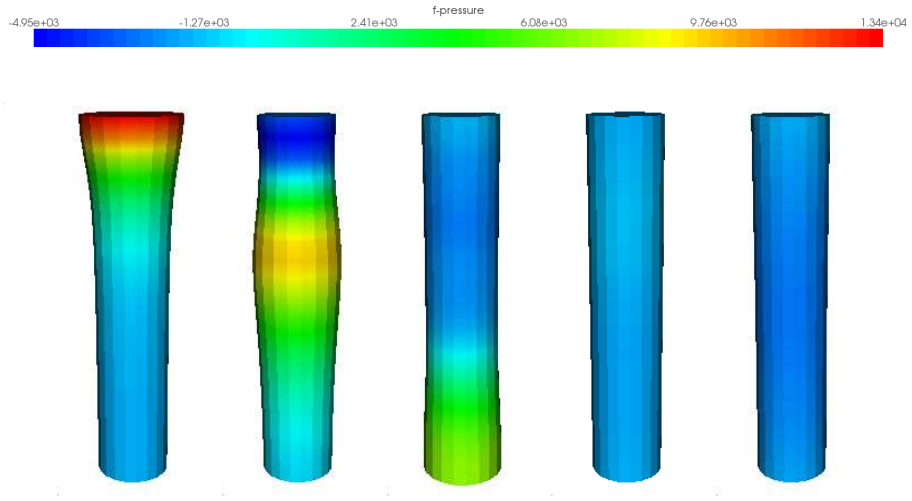

Figura 3.2: Andamento della pressione, avendo imposto la condizione assorbente in uscita a diversi instanti temporali $t = 0.001, 0.006, 0.012, 0.018, 0.024s$.

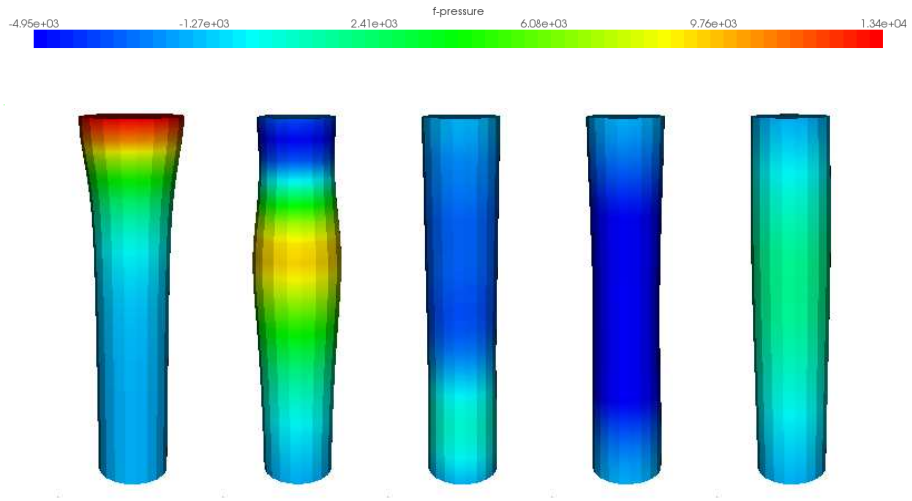

Figura 3.3: Andamento della pressione, avendo imposto lo sforzo nullo in uscita a diversi instanti temporali $t = 0.001, 0.006, 0.012$, $0.018, 0.024s$.

Tali Figure descrivono come si comporta la sezione di uscita con le due diverse condizioni: si noti che nei primi instanti di tempo non ci sono differenze. A poco a poco che il flusso si propaga nella sezione di uscita con condizioni assorbenti, si sviluppa una maggiore pressione e una minore velocità rispetto

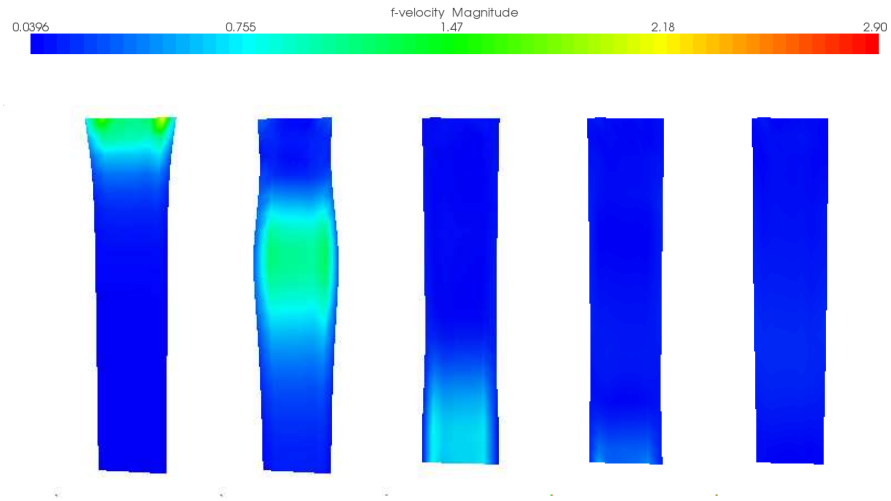

Figura 3.4: Campo di velocità all'interno del cilindro, avendo imposto la condizione assorbente in uscita a diversi instanti temporali $t = 0.001$, 0.006, 0.012, 0.014, 0.018 s.

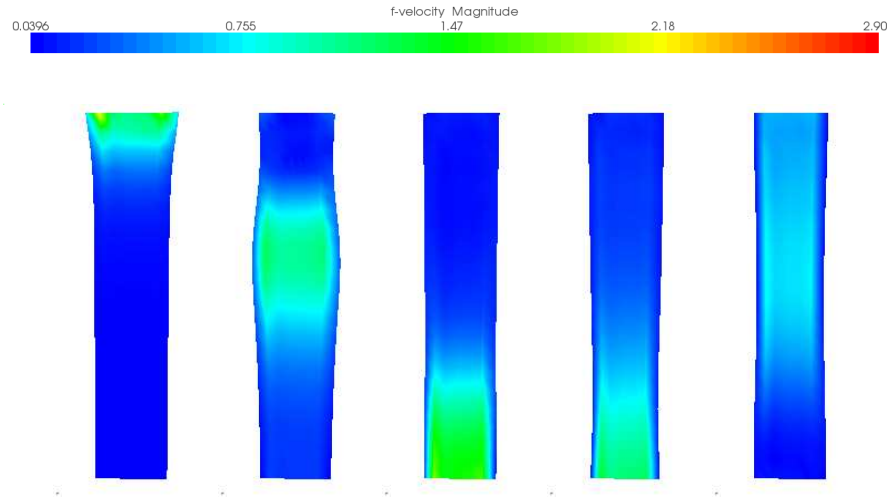

Figura 3.5: Campo di velocità all'interno del cilindro con sforzo nullo sulla parete di uscita, a diversi istanti temporali $t = 0.001$. 0.006, 0.012, 0.014, 0.018 s.

alla condizione di sforzo nullo. La parete di uscita con condizioni assorbenti riduce la quantità di flusso riflesso, infatti dopo che il flusso giunge sulla parete si osserva un ridotto valore della velocità, della pressione e della deformazione. Riassumendo l'uso di una condizione di sforzo nullo produce una riflessione spuria. Questo non accade con le condizioni di Neumann assorbenti. In entrambi i casi il numero medio di sottoiterazioni fluido-struttura necessarie a convergere è 18.

3.2 Robin-Neumann e Dirichlet-Neumann

In questa sezione si presentano i principali risultati ottenuti dal confronto tra l'algoritmo di Robin-Neumann e Dirichlet-Neumann sul caso test di un vaso sanguigno cilindrico rettilineo. Per poter applicare l'algoritmo di Robin-Neumann si deve stimare il valore di α_f ottimale della condizione di interfaccia fluida. Sostituendo il valore di β in (2.4.2) si ottiene α_f per il cilindro

$$\alpha_f = 2\frac{\rho_s h_s}{\Delta t} + \frac{h_s E}{\Delta t}\frac{1}{2R^2}$$

Come prime simulazioni per la validazione dell'algoritmo di Robin-Neumann sono state riprodotte le simulazioni della sezione precedente riguardanti le condizioni di sforzo nullo e condizione di Neumann assorbente. Da un'analisi grafica non si osservano differenze tra le soluzioni ottenute con i due schemi; infatti la convergenza dell'algoritmo di punto fisso garantisce una soluzione uguale a meno di errori minori della tolleranza dello schema numerico. Si riporta in Figura 3.6 l'andamendo della pressione del fluido e della deformazione della struttura ottenuti con lo schema di Robin-Neumann e avendo imposto le condizioni assorbenti.

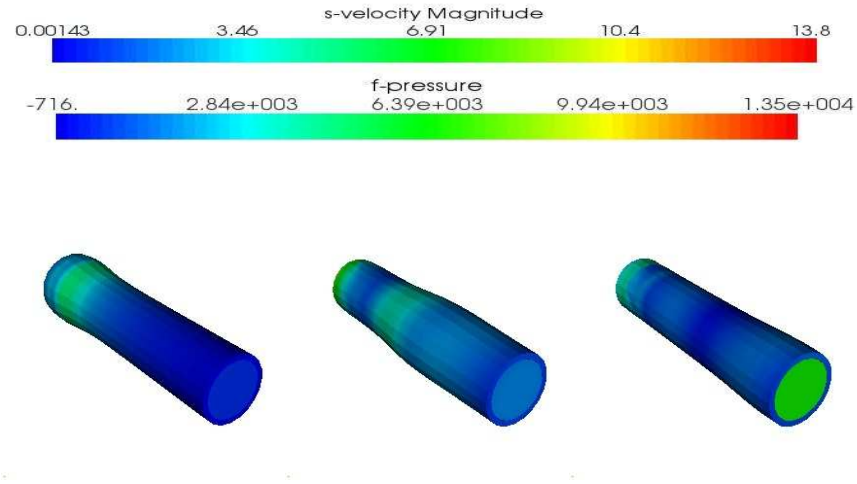

Figura 3.6: Propagazione dell'impulso di pressione nel vaso sanguigno cilindrico: sono raffigurati la pressione del fluido sulla sezione di uscita e il campo di moto della struttura agli instanti di tempo $t = 0.001s$, $0.007s$, $0.012s$.

L'unica differenza che si osserva, tra i due algoritmi, è nel numero medio di iterazioni che determinano la convergenza. Più precisamente nel metodo di Robin-Neumann sono sufficienti in media 5 iterazioni ad ogni passo temporale in confronto alle 18 iterazioni dello schema di Dirichle-Neumann.

Bisogna notare che l'algoritmo di Robin-Neumann utilizzato non applica nessuna procedura di rilassamento, mentre viene usata nell'algoritmo di Dirichlet-Neumann per garantire la convergenza.

Si confrontano ora le prestazioni in termini di numero medio di iterazioni, per l'algoritmo di Robin-Neumann rispetto all'algoritmo di Dirichlet-Neumann.

3.2.1 Massa aggiunta

Quando si risolve un problema fluido-struttura numericamente si può riscontrare l'effetto della massa aggiunta, che è presente in alcuni algoritmi se la densità del fluido e della struttura sono simili. Allo scopo di verificare se l'algoritmo di Robin-Neumann sia soggetto o meno a tale effetto sono state svolte alcune simulazioni con un diverso valore della densità della struttura ρ_s, mentre si mantiene costante la densità del problema fluido. Si riportano i risultati delle simulazione nella Tabella 3.2. Dai risultati ottenuti si osserva come l'algoritmo

Dirichlet-Neumann	
$\rho_s\,(g/cm^3)$	#
1.2	18
5	15
50	10
100	8

Robin-Neumann	
$\rho_s\,(g/cm^3)$	#
1.2	5
5	4
50	4
100	4

Tabella 3.2: Numero medio di sottoiterazioni necessarie ai due algoritmi a giungere a convergenza, al variare della densità del problema struttura.

di Robin-Neumann non sembra essere soggetto a effetti di massa aggiunta in quanto il numero di sottoiterate necessarie a convergere varia solo tra i primi due valori di ρ_s mantenendosi costante per i successivi valori. Ripetendo questa simulazione con l'algoritmo di Dirichlet-Neumann si osserva che all'aumentare della densità della struttura si riduce il numero delle sottoiterazioni necessarie a giungere a convergenza. Questo fatto conferma come questo algoritmo sia soggetto all'effetto di massa aggiunta.

3.2.2 Variazione della tolleranza del metodo di punto fisso

In queste simulazioni si determina come la variazione della tolleranza dello schema di punto fisso influenzi il numero medio di sottoiterazioni. Una minore tolleranza del punto fisso rende la soluzione più accurata. In Tabella 3.3 sono riportati i principali risultati. Riducendo il valore di ϵ si osserva in entrambi gli algoritmi un aumento del numero di sottoiterazioni necessarie a consentire la convergenza.

Dirichlet-Neumann	
ϵ	#
10^{-4}	18
10^{-5}	23
10^{-6}	27

Robin-Neumann	
ϵ	#
10^{-4}	5
10^{-5}	7
10^{-6}	9

Tabella 3.3: Numero medio di sottoiterazioni necessarie ai due algoritmi a giungere a convergenza, al variare del passo temporale.

3.2.3 Variazione del passo temporale

In questa prova si determina se la convergenza degli algoritmi è influenzata da variazioni del passo temporale. A differenza degli altri test si è utilizzata una tolleranza di 10^{-6}.

Dalla Tabella (3.4) si osserva che mentre nell'algoritmo di Dirichlet-Neumann una riduzione del passo temporale determina un maggior numero di iterazioni, nell'algoritmo di Robin-Neumann non si verifica questo effetto, infatti il numero di sottoiterazioni dell'algoritmo di Robin-Neumann non varia.

Dirichlet-Neumann	
$\Delta t\,(s)$	#
0.0002	50
0.0005	37
0.001	26

Robin-Neumann	
$\Delta t\,(s)$	#
0.0002	9
0.0005	8
0.001	9

Tabella 3.4: Numero medio di sottoiterazioni necessarie ai due algoritmi a giungere a convergenza, al variare della tolleranza dell'algoritmo di punto fisso.

3.3 Robin-Neumann su altre geometrie:

In questa sezione si vuole semplicemente provare l'algoritmo di Robin-Neumann in geometrie diverse dal cilindro rettilineo al fine di testare la generalità e robustezza del metodo.

3.3.1 Toro

Come prima geometria si considera un vaso sanguigno a forma toriodale come mostrato in Figutra 3.7. Questa geometria schematizza un arco aortico. Il dominio struttura è formato da 1008 nodi e da 3840 tetraedri e il dominio fluido da 1785 nodi e 9120 tetraedri, come mostrato in Figura (3.7). Per determinare il valore di β necesssario a stimare la pressione in uscita per le condizioni assorbenti e il valore di α_f, si considerano le seguenti costanti definite dalla Figura 3.8, a (raggio del toro) e c distanza delle basi dal centro dal piano. Il calcolo del β per il toro, è riportato in [26], e vale:

$$\beta(u,v) = \frac{h_s E}{1-\nu^2}\Big((1-\nu)\Big(\frac{\cos^2 v}{(c+a\cos v)^2}+\frac{1}{a^2}\Big)+\nu\Big(\frac{\cos\nu}{(c+a\cos v)^2}+\frac{1}{a}\Big)^2\Big)$$

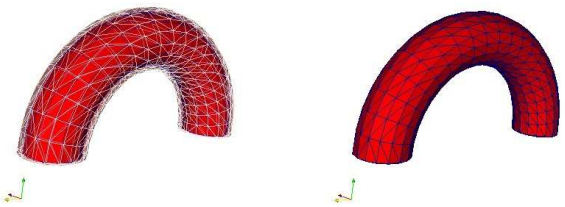

Figura 3.7: A sinistra mesh del problema solido e a destra mesh del problema fluido.

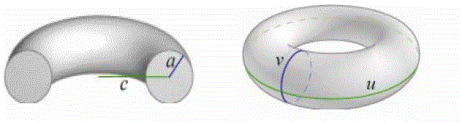

Figura 3.8: Raggi e coordinate del toro.

Risolvendo il problema fluido-struttura con lo schema di Dirichlet-Neumann e Robin-Neumann con i parametri standard (definiti nella prima sezione) si osserva che in media sono necessarie 7 iterazioni per l'algoritmo di Robin-Neumann, mentre per l'algoritmo di Dirichlet-Neumann ne servono 23.

Sono state poi ritestate le condizioni assorbenti e le condizioni di sforzo nullo. Dalla Figura 3.9, si nota come anche per questo caso nelle prime iterazioni il comportamento della pressione nelle due prove, è simile. Tuttavia nelle condizioni assorbenti più il flusso si propaga in direzione di outflow maggiore è la pressione nella sezione di uscita con una minore riflessione. All'instante $t = 0.012s$ il massimo valore dell'impulso giunge sulla parete di uscita. In Figura 3.10 è riportato l'andamento della velocità del fluido: nei primi instanti le condizioni assorbenti si comportano in modo analogo allo sforzo nullo, ma man mano che il flusso si propaga si ha una maggiore riduzione della velocità per le condizioni assorbenti rispetto alle condizioni non assorbenti. Inoltre dopo che l'impulso è giunto sulla parete di uscita si osserva che la velocità si annulla se si utilizza la condizione assorbente, diversamente alla condizione di sforzo nullo che conservano una velocità residua.

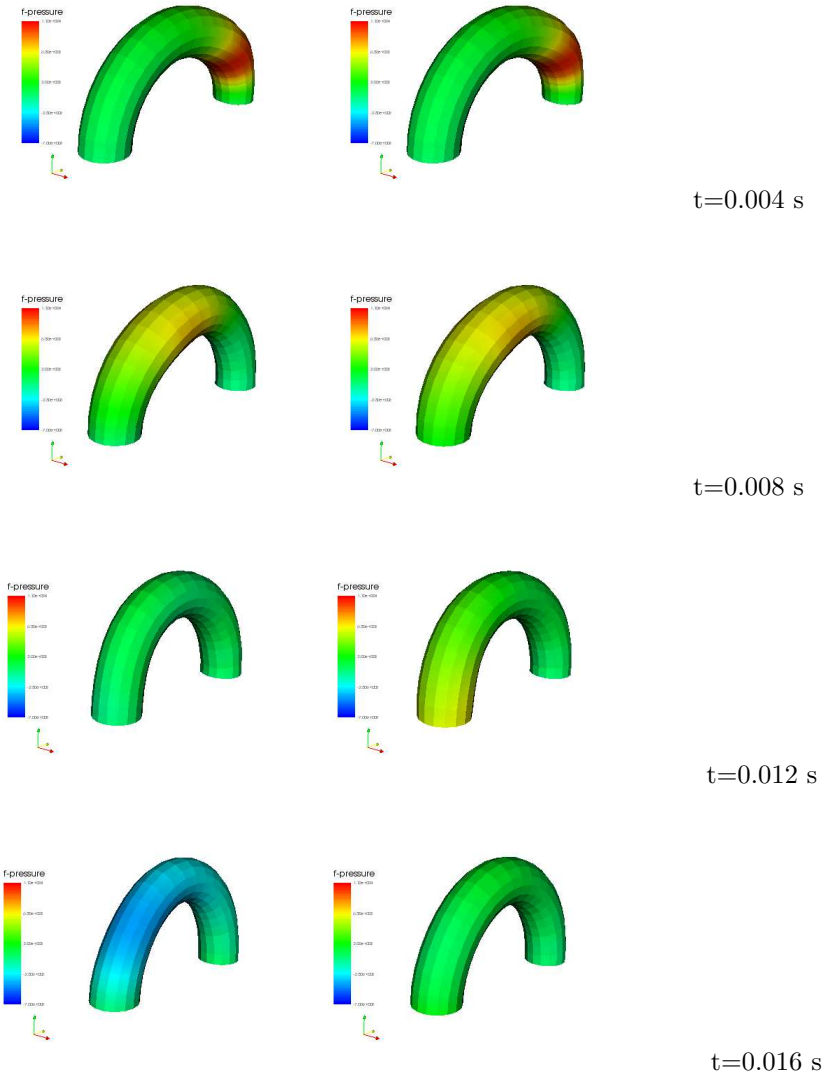

Figura 3.9: Propagazione di un impulso di pressione senza (sinistra) e con (destra) delle condizioni assorbenti.

.

3.3.2 Carotide

Si simula un impulso di pressione in una biforcazione carotidea usando i parametri standard definiti in precedenza, a parte lo spessore del vaso sanguigno che in questo caso vale $0.5cm$. La mesh della carotide utilizzata è stata costruita attraverso dei dati clinici. La mesh solida è formata da 24100 vertici, mentre la mesh fluida da 20072. La sezione di ingresso è circolare di raggio $0.5cm$, mentre una faccia di uscita ha raggio $0.375cm$ e l'altra $0.334cm$. Si risolve il prob-

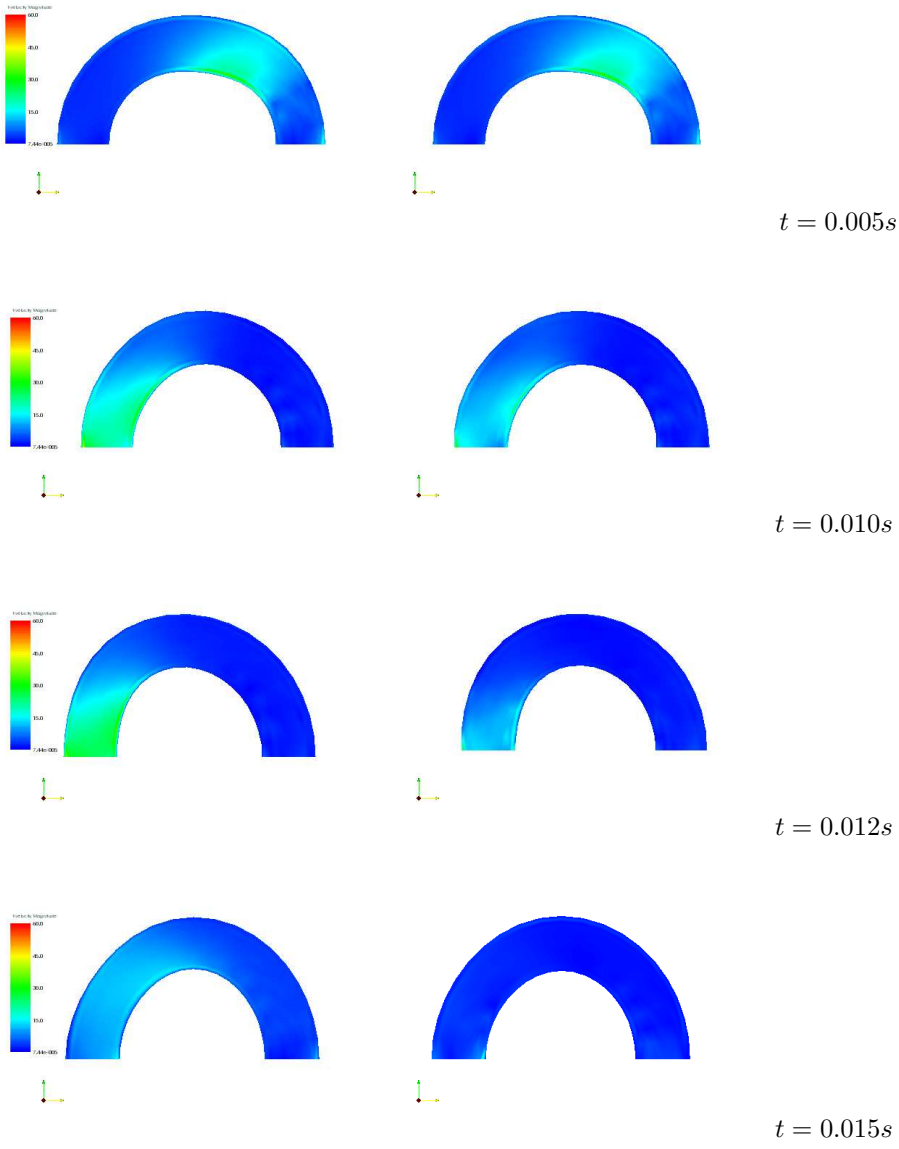

Figura 3.10: Evoluzione della velocità senza (a sinistra) e con condizioni assorbenti (a destra)

lema fluido-struttura attraverso l'algoritmo di Robin-Neumann con condizioni assorbenti in uscita. Per determinare α_f e il valore della pressione assorbente si approssima il parametro di curvatura attraverso un raggio medio $r = 0.4$ e poi si applica la formula del β del cilindro. In Figura (3.11) sono riportati l'andamento della pressione a diversi instanti temporali. Si osserva che la biforcazione genera una riflessione in pressione fisiologica, mentre non si verifica riflessioni in

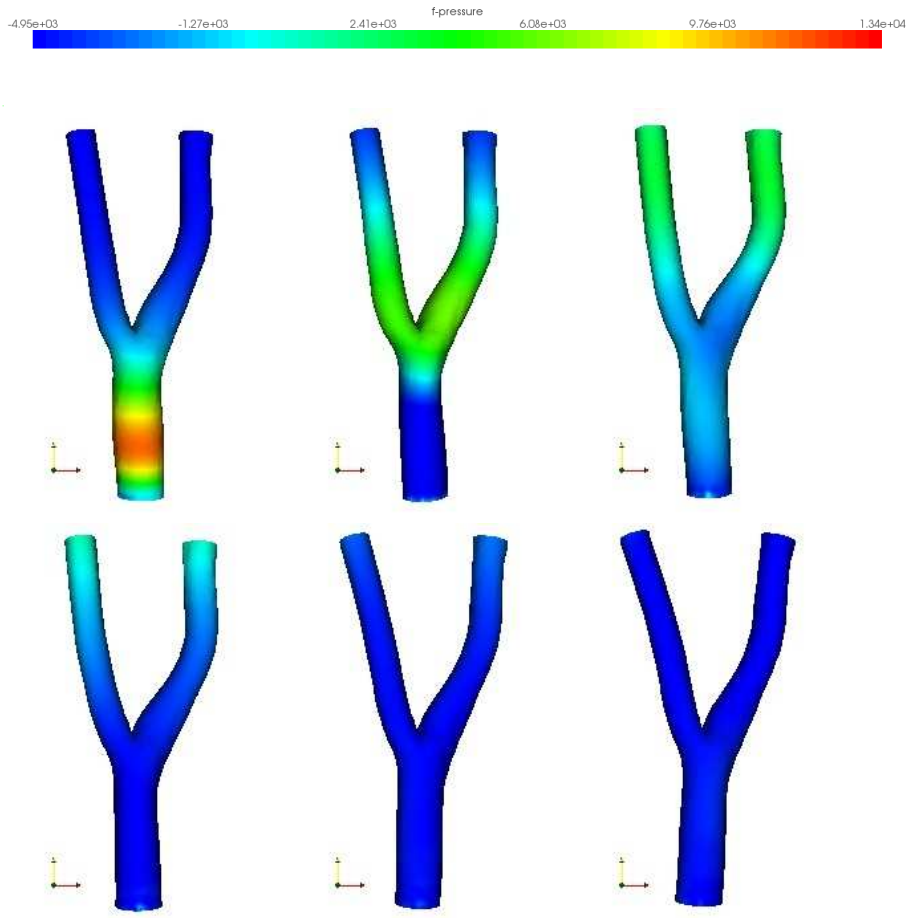

Figura 3.11: Andamendo della pressione a diversi istanti temporali: (in alto) $t = 0.003$, 0.007, $0.011s$, (in basso) $t = 0.015$, $t = 0.020$, $t = 0.024s$.

uscita grazie all'utilizzo delle condizioni assorbenti.

Capitolo 4

Introduzione a LifeV parallelo

LifeV è un progetto sviluppato congiuntamente da Politecnico di Milano (laboratorio MOX), Ecole Polytechnique Fèdèrale di Lausanne (gruppo CMCS) e INRIA (progetto REO) che ha avuto inizio nel 1999/2000.
LifeV è una libreria a elementi finiti in C++ che per la risoluzione di equazioni differenziali a derivate parziali orientata in particolare a applicazioni biomediche. LifeV è stata utilizzata con successo e nell'industria per simulare problemi di interazione fluido struttura, di trasporto di massa, simulazioni multiscala. In particolare contiene una serie di casi test per le applicazioni emodinamiche che sono state messe a punto durante il progetto europeo HaeModel.
Grazie a un continuo lavoro degli autori e dei collaboratori, dal 2000 LifeV ha continuato a evolversi in nuove versioni e aggiornamenti che hanno arricchito e migliorato il codice; infine nel 2006 gli autori hanno creato una nuova versione parallela di LifeV, ossia in grado di eseguire una simulazione su più processori. Per ottimizzare gli effetti del parallelismo sono stati utilizzati alcuni pacchetti gratuiti disponibili in rete come mpi (implementato tramite openmpi o mpich), ParMETIS, e la libreria Trilinos di cui sono stati usati i pacchetti Epetra, AztecOO e IFPACK. Lo scopo di questa sezione è quello di mostrare brevemente LifeV con particolare attenzione alla libreria Epetra. Illustreremo le principali caratteristiche del problema fluido-struttura.

4.1 La Partizione della Mesh

ParMETIS (`http://glaros.dtc.umn.edu/gkhome/views/metis`) è una libreria MPI che implementa una varietà di algoritmi per la partizione non strutturata dei grafi, delle mesh e per la partizione delle matrici sparse. ParMETIS estende le funzionalità di METIS e include le caratteristiche per il calcolo parallelo AMR e le simulazioni a larga scala.
In LifeV ParMETIS è usato per partizionare la mesh distribuendola su tutti i processori disponibile della simulazione: la mesh originale viene letta e copiata da ogni processore creando delle mesh locali, conclusa questa fase la mesh originale viene cancella.
Nella fase del caricamento della mesh il principale limite è la dimensione della

mesh che se eccessivamente grande può rendere poco efficace il parallelismo. Rispetto alle versioni precedenti di LifeV la libreia ParMETIS ha introdotto una nuova classe per la mesh:

```
partitionMesh< RegionMesh<LinearTetra> >
```

Il costruttore riceve una mesh di tipo `RegionMesh<LinearTetra>` e chiama `ParMETIS_V3_ParKway` per partizionarla e distribuirla su più processori.

4.2 La gestione dei Vettori e Matrici: il pacchetto Epetra

Nella versione parallela di LifeV è stata modificata la gestione delle matrici e dei vettori in modo tale da poter essere distribuiti su più processori: mediante l'uso della libreria Epetra (`http://trilinos.sandia.gov`) le matrici sono diventate Epetra_FECrsMatrix, mentre i vettori Epetra_FEVector.
La distribuzione deigradi di libertà è determinata dalla mesh partizionata: mediante un loop su ogni elemento della mesh locale si crea la lista dei gradi di libertà gestista dal processore assegnato. Questa procedura definisce una mappa *ripetuta*, un'instanza di Epetra_Map con entrate ripetute tra i processori (che appartengono all'interfaccia tra i processori). Successivamente viene creata una mappa *unica* attraverso la rimozione delle entrate ripetuti . La mappa unica è usata per la soluzione di vettori e matrici, mentre la mappa ripetuta per accedere alle informazioni condivise da più processori (ad esempio il termine convettivo deve essere noto da entrambi i lati dell'interfaccia per questo motivo è creato attraverso la mappa ripetuta). Epetra implementa alcuni membri di import ed export che gestiscono la distribuzione di informazioni tra vettori con differenti tipi mappe.
L'assemblaggio del sistema lineare (Stiffness, massa,...) è fatto attraverso un loop sugli elementi locali di una mesh locale. Le classi Epetra_FEVector e Epetra_FECrsMatrix gestiscono i commutatori così da poter utilizzare anche i gradi di libertà assegnati su altri processori, questa procedura viene svolta attraverso la chiamata del metodo GlobalAssemble(), dopo di ciò non può essere modificata la costruzione della matrice.
Per accedere da LifeV alle classi di Epetra sono state create alcune interfaccie grafice come EpetraMap.hpp EpetraVector.hpp e EpetraMatrix.hpp che permettono di utilizzare le potenzialità di Trilinos senza dovervi accedere direttamente ai metodi.

Ora vengono presentate le principali caratteristiche di Epetra utilizzate in LifeV. Per poter usare gli Epetra si deve definire un Epetra communicator, Epetra_Comm, una classe virtuale che registra le informazioni generali necessarie a far funzionare il codice sia in versione seriale che in parallela. Una volta definito un Epetra_Comm si può costruire un Epetra_Map necessaria per definire gli Epetra_Vector e Epetra_Matrix.

4.2.1 EpetraComm

Gli Epetra_Comm sono usati per costruire tutti oggetti Epetra_Map che servono a definire le classi Epetra. Gli Epetra_Comm hanno due possibili implemntazioni:

- Epetra_SerialComm per l'esecuzioni in modalità seriale;

- Epetra_MpiComm per l'esecuzioni in modalità MPI;

In LifeV si costruisce l'Epetra_Comm nel seguente frammento di codice:

```
#include "Epetra_config.h"
#ifdef HAVE_MPI
#include "Epetra_MpiComm.h"
#else
#include "Epetra_SerialComm.h"
#endif

int main(int argc, char** argv)
{
#ifdef HAVE_MPI
    MPI_Init(&argc, &argv);
    cout << "% using MPI" << endl;
#else
    cout << "% using serial Version" << endl;
#endif

#ifdef HAVE_MPI
        MPI_Finalize();
#endif
```

Questa classe possiede alcuni metodi simili alle funzioni MPI; tra questi metodi si ricordano: `MyPID()`, `NumProc()`, `Barrier()`, `Broadcast()`, `SumAll()`, `GatherAll()`, `MaxAll()`, `ScanSum()`.
Dopo aver definito un oggetto Epetra_Comm si può costruire un elemento della classe Epetra_Map necessaria per definire le matrici e i vettori distribuiti e per raccogliere informazioni.

4.2.2 EpetraMap

Una mappa può essere definita come una distribuzione di un insieme di elementi su più processori e serve per la registrazione delle informazioni necessarie a costruire vettori e matrici gestibili in parallelo. Una mappa è un oggetto della classe Epetra_Map (più precisamente di Epetra_BlockMap poichè Epetra_Map è una sua classe derivata).

In questa parte vengono mostrate le principali caratteristiche degli Epetra_Map, un possibile costruttore di questa classe è il seguente

```
Epetra_Map (int NumGlobalElements, int NumMyElements,
            const int *MyGlobalElements,
            int IndexBase, const Epetra_Comm& Comm);
```

le variabili rappresentano:

`NumGlobalElements` il numero totale di elementi su tutti i processori.

`NumMyElements` il numero di elementi posseduti dal processore chiamante;

`MyGlobalElements` una lista di lunghezza `NumMyElements` che contiene l'ID globale degli elementi posseduti dal processore chiamante.

`IndexBase` il minimo valore per identificare la referenza dell'array. Assume un valore intero e per codici scritti in C/C++ è identificato da 0 e per Fortran 1, ma può assumere un qualunque altro valore.

`Comm` l'Epetra_Comm commutatore.

A seconda dei valori assunti da `NumGlobalElements` e `NumMyElements` si verifica una delle seguenti situazioni:

- se `NumGlobalElements` è uguale al `NumMyElements` (diverso da zero) la mappa è definita come una mappa ripetuta. Gli oggetti costruiti con questa mappa sono identicamente ripetuti su tutti i processori disponibili.
- se `NumGlobalElements` è uguale a -1 e `NumMyElements` è passato segue che `NumGlobalElements` è calcolato come la somma di `NumMyElements` su tutti i processori.
- altrimenti il `NumGlobalElements` viene definito come la somma di `NumMyElements` su tutti i processori. Un possibile errore può verificarsi se la dimensione non è la stessa.

I principali membri di questa classe restituiscono le informazioni riguardanti il numero di elementi su tutti i processori e sul processore corrente (`NumGlobalElements()` e `NumMyElements()`), quale sia il minimo e il massimo indice globale su tutti i processori (`MinAllGID()` e `MaxAllGID()`) e sul processore chiamente (`MinMyGID()` e `MaxMyGID()`) e il minimo e massimo indice locale sul processore chiamente (`MinLID()` e `MaxLID()`). Sono inoltre definiti `LinearMap()` e `DistribuitedGlobal()`. Il primo restituisce `true` se gli elementi sono distribuiti linearmente su tutti i processori, ossia sul processore 0 i primi n/p elementi, sul processore 1 i successivi n/p elementi e così via, mentre il secondo restituisce `true` seil vettore è distribuito su più di un processore (questo è sempre vero tranne nel caso di vettori *seriali* e oggetti creati attraverso una mappa di tipo Epetra_LocalMap). Nell'interfaccia grafica costruita per LifeV sono stati aggiunti i seguenti membri per facilitare la costruzione dei vari tipi di mappe:

`operator =`, `operator +` e `operator + =` tra due mappa: copia una mappa a partire dall'altra, somma due mappe, somma la mappa a sinistra dell'uguale a quella di destra.

`getEpetra_Map()` richiama la mappa corrente;

`getRepeatedEpetra_Map()` richiama la mappa ripetuta;

`getUniqueEpetra_Map()` richiama la mappa unica;

`createMap()` rinizializza una mappa esistente;

`uniqueMap()` ridefinisce la mappa come unica;

`bubbleSort()` che riceve in ingresso un vettore di tipo Epetra_IntSerialDenseVector e ordina dal più piccolo al più grande gli elementi del vettore;

`setUp()` definisce gli elementi della mappa.

Si ricorda che attraverso il pacchetto Epetra è possibile definire vettori *sequenzali* per le piattaforme seriali e parallele. Un vettore sequenziale è un vettore che non necessita di essere partizionato su più processori. In quest'ottica ogni processore definisce i propri vettori sequenzali e le modifiche di un elemento di questo vettore non influenza direttamente i vettori costruiti su altri processori. Si considerano ora i Vettori distribuiti.

4.2.3 EpetraVector

Epetra_Vector è una classe per la gestione di vettori densi distribuiti su uno o più processori. Gli oggetti Epetra (derivanti dalla clase Epetra_DistObject) sono creati attraverso un Epetra_Map. Nel pacchetto di Trilinos sono stati forniti quattro diversi tipi di costruttori per questa classe:

```
1 Epetra_Vector(const Epetra_BlockMap& Map, bool zeroOut = true);

2 Epetra_Vector(const Epetra_Vector& Source);

3 Epetra_Vector(Epetra_DataAccess CV,
  const Epetra_BlockMap& Map, double *V);

4 Epetra_Vector(Epetra_DataAccess CV,
   const Epetra_MultiVector& Source, int Index);
```

Il primo costruttore è di default alloca lo spazio di memoria e inizializza tutti gli elementi del vettore a zero; il secondo è un costruttore di copia, mentre il terzo e il quarto servono quando si devono usare dei file di dati. La variabile Epetra_DataAccess determina in che modo si deve accedere al file di dati. Questa variabile assume due possibili valori.

Copy alloca la memoria e copia i dati che riceve in ingresso; con questa modalità una volta copiato il file di dati, il vettore è indipendente dai dati;

View crea un view dei dati che riceve in ingresso, in questo modo i dati di ingresso se alterati modificano il vettore; per questo motivo è consigliabile usarli solo per ottimizzare il codice dopo averlo costruito in modalità Copy ed essersi assicurati del buon funzionamento.

I principali metodi definiti per la classe Epetra_Vector sono:

`ReplaceMyValues()` o `SumIntoMyValues()` che replica o somma alla componente i-esima *locale* del vettore un double;

`ReplaceGlobalValues()` o `SumIntoGlobalValues()` che inizializza o somma alla componente *i*-esima *globale* del vettore un double.

Va notato che nessun processore può inizzializzare una componente *locale* di un altro processore, ossia queste funzioni sia locali che globali si riferiscono alle componenti del vettore assegnate sul processore chiamante.
Nell'interfaccia costruita per LifeV sono stati definiti alcuni membri che richiamano le funzioni del pacchetto Epetra senza dovervi accedere direttamente:

`getEpetraVector()` richiama l'attributo della classe EpetraVector, viene usato per richiamare i membri di Trilinos. Nell'implementazione dei mebri seguenti si utilizza getEpetraMap(), inoltre viene usata per richiamare i membri di Trilinos per cui non è stata definita un'opportuna interfaccia. Ad esmpio per inizzializzare un EpetraVEctor con un valore non nullo (uguale a n) si può usare la seguente costruzione:

```
EpetraMap map;
EpetraVector<double> vector(map);
vector.getEpetraVector().PutScalar(n);
```

`Map()` accede alla mappa del vettore;

`GlobalAssemble()`

`operator []` e `operator ( )` accedono all'elemento della riga i-esima.

`Norm2()` e `NormInf()` calcola la norma in 2 e norma infinito del vettore;

`Import` ed `Export` questi membri ricevono in ingresso un EpetraVector e un Epetra_CombineMode:

```
EpetraVector& Import (const Epetra_FEVector& _vector,
                      Epetra_CombineMode combineMode);

const EpetraVector& Export ( EpetraVector& _vector,
                     Epetra_CombineMode combineMode) const;
```

Il membro `Import` copia il vettore di destra nel vettore di sinistra, mentre il membro `Export` copia il vettore sinistra nel vettore di destra. Se le due mappa non sono uguali questi membri importano/esportano il vettore di sinistra sulla mappa di destra e successivamente viene copiato il vettore. I nodi in comune delle due mappa vengono trattati attraverso la modalità definita dal CombineMode:

`Add` le componenti del processore A vengono aggiunte insieme

`Insert` le componente del vettore B sono inserite nel vettore A (o sovrascrivono quelle esistenti).

`InsertAdd` le componenti del vettore B vengono inserite nel vettore A e aggiunte ai valori già esistenti.

`Average` calcola la media;

`AbsMax` viene inserito il valore massimo;

`Zero` vengono posti uguale a zero;

`operator =` copia un EpetraVector di destra nel vettore di sinistra, se le mappe dei due EpetraVector non sono uguali,viene chiamata la funzione `Import(B,Add)`.

`operator +`, `operator +=`, `operator -=`: definiscono le operazioni tra due EpetraVector;

`operator *=`, `operator *`, che moltiplica ogni componente del vettore per uno scalare.

I metodi `Import` ed `Export` utilizzano le classi Epetra_Export e Epetra_Import che permettono alle classi Epetra di comunicare. Il loro costruttore riceve in ingresso due EpetraMap: la prima mappa specifica quali sono gli IDs globali degli elementi posseduti dal processore chiamante, mentre la seconda specifica gli IDs degli elementi che si vuole importare. Quando si usa un Epetra_Import significa che il processore chiamante conosce che cosa si vuole importare, mentre un Epetra_Export che cosa si vuole inviare. Un Epetra_Import può essere usato per fare un Export come operazione inversa ed equivalentemente un Epetra_Export può essere usato per fare un Import.
In LifeV questi metodi servono a scambiare informazioni tra una mappa *unica* a una mappa *ripetuta*. Si prenda ad esempio il termine `rhsFull` del problema fluido:

```
  vector_type rhsFull(*M_localMap.getRepeatedEpetra_Map());
  rhsFull.Import(M_rhsNoBC, Zero);
  bcManage( matrix, rhsFull, *M_uFESpace.mesh(), M_uFESpace.dof(),
            BCh, M_uFESpace.feBd(), 1.0, M_data.time() );
 rhsFull.Export(rhs,Add);
```

Esso viene costruito come un EpetraVector sulla mappa del problema fluido *ripetuta* e inizzializzato a zero inizialmente; successivamente si importa nel vettore i valori di M_rhsNoBC definito sulla mappa *unica*, si applicano le condizioni al contorno sul termine rhsFull tramite il metodo `bcManage` e infine si esporta `rhsFull` nel vettore `rhs`.

4.2.4 EpetraMatrix

Epetra contiene vari tipi di matrici, ma si distinguono in due differenti famiglie *seriali* e *distribuite*. Le prime hanno generalmente una dimensione piccola o vengono usate solo localmente nel codice per cui non conviene distribuirle su più processori, mentre le restanti matrici sono generalmente distribuite.
In questa sezione si fa un accenno alle matrici seriali in quanto hanno le stesse caratteristiche delle matrici comuni del C++; mentre tra le varie classi di matrici distribuite viene presentata solo Epetra_FECrsMatrix poichè è la principale classe usata in LifeV.
Un possibile metodo per costruire una matrice seriale (`matrix`) di dimensione n per m è il seguente:

```
#include"Epetra_SerialDenseMatrix.h"
Epetra_SerialDenseMatrix matrix(n,m)
```

Questa classe di matrici ha i principali metodi per la gestione delle matrici, ad esempio per accedere alle componenti si utilizzano gli operatori di parentesi

(`matrix( i,j )` o `matrix[j][i]`[1])e sono definiti i metodi classici delle matrici. La classe Epetra_FECrsMatrix serve a definire una matrice distribuita su più processori, questa classe generalizza la classe Epetra_CrsMatrix infatti ha in più la capacità di poter assegnare anche gli elementi della matrice che non sono locali (ossia che non appartengono al processore chiamante) attraverso il metodo `GlobalAssemble()`. Questo metodo permette di assemblare tutte le componenti non locali della matrice che sono stati modificati da ogni processore così da creare un unico pattern della matrice. Più precisamente nell'implementazione di GlobalAssemble() viene richiamato il membro `FillComplete()` che riorganizza i file dei dati così che ogni processore conosca quali elementi sono interni, di bordo ed esterni.
I costruttori di default degli Epetra_FECrsMatrix accettano come variabili:

`Epetra_DataAccess CV` determina il modo (*Copy* o *Views* con cui la matrice è definita[2])

il secondo termine può essere o:

un Epetra_Map definisce la distribuzione delle righe della matrice;

due Epetra_Map dove il primo definisce la distribuzione delle righe della matrice, mentre il secondo determina l'insieme degli indici della matrice appartenente ad ogni processore.

un Epetra_CrsGraph che definisce la struttura non nulla della matrice;

un booleriano che determina se la matrice deve ignorare o no le entrate locali della matrice; di default è inizializzato a false;

Se si utilizza il costruttore con Epetra_Map si può introdurre un intero che stima il numero di righe (colonne) della matrice per migliorare le prestazioni.
Tra gli altri costruttori forniti da Trilinos si ricorda il costruttore di copia che riceve in ingresso un Epetra_FECrsMatrix.
Come per gli EpetraVector esistono tutti quei metodi che permettono di accedere alle componenti globali della matrice (`NumGlobalNonzeros()`, `NumGlobalRows()`, `NumGlobalCols()`, `NumGlobalDiagonals()`), alle componenti del processore chiamante (`NumMyNonzeros()`, `NumMyRows()`, `NumMyCols()`, `NumMyDiagonals()`) e quei metodi che permettono di inizzializzare o sommare una componente (`SumIntoGlobalValues()`, `InsertGlobalValues()`, `ReplaceGlobalValues()`, `InputGlobalValues`, `InputNonlocalGlobalValues`, `InputNonlocalValue`).
Nell'interfaccia di LifeV sono stati definiti i seguenti metodi:

`getEpetraMatrix()` che restituisce la Epetra_FECrsMatrix;

`RowMap()` (o `ColMap()`) restituisce la mappa delle righe (o delle colonne);

`operator +=`, `operator +`, `operator - =` operazioni tra matrici;

`operator *=`,`operator *` moltiplica la matrice per uno scalare;

[1] nell'operatore di parentesi quadra sono scambiati gli indici, ossia si definisce prima la colonna e poi la riga.
[2] si rimanda alla parte degli EpetraVector per la loro definizione.

`diagonalize( )` assegna agli elementi della diagonale un valore che riceve in ingresso come variabile o come vettore;

`set_mat_inc()` e `set_mat()` ricevono in ingresso l'indice della riga i e della colonna j e un valore e assegna alla componente i, j della matrice il valore in ingresso, ma nel primo caso applica il metodo `ReplaceGlobalValues()`, mentre nel secondo `InsertGlobalValues` (la differenza tra i 2 metodi consiste nel fatto che nel primo vengono modificati solo i valori già esistenti nella matrice quindi non può essere applicato per inizializzare un elemento, mentre il secondo inizializza l'elemento).

Per maggiori dettagli si rimanda al manuale di Trilinos e alla libreria Epetra.

4.3 Solver parallelo: IFPACK e AztecOO

Per la soluzione dei sistemi algebrici è stato utilizzato il pacchetto di Trilinos IFPACK (`http:/trilinos.sandia.gov`) che fornisce una libreria di precondizionatori algebrici per la risoluzione iterativa di matrici. La configurazione dei precondizionatori è flessibile, il programmatore può decidere il numero di sovrapposizioni e il tipo di solutori locali (solitamente LU o ILU).
Una volta definito il precondizionatore e la matrice si utilizza AztecOO (`http:/trilinos.sandia.gov`) per risolvere il problema lineare attravesro un metodo iterativo (quelli usati sono GMRES, BiCGStab, CG), il precondizionatore una volta costruito nel codice viene conservato poichè spesso viene riutilizzato.

4.4 Il problema Fluido-Struttura

Il problema fluido-struttura in LifeV ha la pecularità di combinare i codici scritti per la risoluzione del problema di Oseen, del problema di Venant-Kirchhoff e dell'estensione armonica. Per questo motivo sono state create dagli autori di LifeV alcune classi in grado di gestire l'accoppiamento dei vari problemi. In questa sezione verranno presentati gli algoritmi più significativi per trattare le condizioni di interfaccia fluido-struttura.

Nella versione parallela bisogna notare che per lanciare il programma si devono usare almeno due processori, in quanto il problema struttura viene trattato su un processore e i rimanenti processori gestiscono il problema fluido.

4.4.1 FSIOperator

Questa classe fornisce i principali opertori necessari per la gestione delle connessioni di interfaccia del problema fluido-struttura. Le connessioni sono definite a due a due tra i tre problemi e gli operatori qui definiti permettono di aggiornare ad ogni iterazione le condizioni di interfaccia; per questo motivo, in questa classe, sono state inseriti gli attribuiti e i membri necessari a convertire il problema da Dirichlet-Neumann a Robin-Neumann. Alcuni membri della classe `FSIOpertor` sono stati definiti virtuali e sono stati definiti in seguito in una sua classe figlia `fixedPointBase`. Si presenta ora la gestione delle condizioni di bordo. I principali attributi di questa classe sono i

`BCVectorInterface`, e gli `DofInterface3Dto3D` e gli `EpetraVector` necessari a gestire le condizioni di interfaccia tra fluido e struttura (per maggiori spiegazioni si veda il paragrafo successivo). I principali membri di questa classe sono `meshMotion`, `transferMeshMotionOnFluid`, `transferFluidOnInterface`, `transferSolidOnInterface`, `interpolateVelocity`,... questi membri hanno in ingresso due `EpetraVector` e trasferiscono le informazioni dal primo vettore al secondo. Più precisamenta questi metodi convertono le informazioni contenute nel primo vettore (definito su una mesh), nel secondo vettore definito sulla seconda mesh. Questi metodi sono usati nella classe fixedPointBase per costruire l'algoritmo di punto fisso (si veda la sezione FSISolver). Ora si definisce brevemente la procedura necessaria per applicare le condizioni di bordo in lifeV.

Condizioni al contorno

Per definire una condizioni al contorno per uno dei problemi considerati si inizzializza un contenitore `BCHandler` in cui a seconda dell'identificatore del bordo (*flag*) si definisce un'opportuna condizone. Le condizioni al contorno possono essere definite secondo due diverse procedure:

> *BCFunctionBase* impongono la condizione di bordo in modo analitico, ossia attraverso una *funzione* di 5 variabili `x`, `y`, `z`, `t`, `i`. Dove `i` identifica l'i-esima componente del vettore associato alla BCFunction.
>
> *BCVectorBase* impongono la condizione di bordo attraverso un *EpetraVector*.

Entrambe le procedure definiscono le principali condizioni al bordo: *Dirichlet* in formulazione forte, *Neumann* formulazione debole e la condizione di *Robin*. In più i BCVector sono definiti in modo tale che le condizioni di *Neumann* sono fornite o già integrate contro le funzioni di base (`tipo=0`) o da integrare solo in direzione normale (`tipo=1`) o da integrare in tutte le direzioni (`tipo=2`)

Le condizioni di interfaccia sono ottenute durante la risoluzione dei diversi problemi, esse sono già in formato `EpetraVector` quindi vengono gestite attraverso `BCVectorInterface`. Un suo possibbile costruttore è:

```
BCVectorInterface( const EpetraVector<double>& vec, UInt nbTotalDof,
          UInt nbTotalDof, dof_interface_type dofIn, UInt type=0 );
```

Come nei `BCVector` esso riceve in ingresso un EpetraVector, la dimensione del vettore e il tipo, ma in più possiede `dof_interface_type` che definisce la connessione di interfaccia tra le due pareti. Più precisamente nel problema trattato `dof_interface_type` è un elemento di `DofInterface3Dto3D` il cui costruttore è così definito:

```
DofInterface3Dto3D(const RefFE& refFE, const Dof& dof1, const Dof& dof2);
```

dove i primi due elementi definiscono in quale mappa si vuole convertire l'EpetraVector, mentre gli ultimi due quale sia la mappa di provenienza. Ad esempio, in questo lavoro, per implementare la condizione di Robin è stato definito il seguente membro:

```
void FSIOperator::
setStructureLoadToFluid(const vector_type &res,UInt type){
   M_bcvStructureLoadToFluid->setup(res,
                      M_solid->dFESpace().dof().numTotalDof(),
              M_dofStructureToFluid,type);}
```

che converte il residuo della struttura in una condizione per il problema fluido.

4.4.2 FSISolver

Questa classe permette di risolvere il problema fluido-struttura ad ogni passo temporale attraverso gli operatori definiti in `FSIOperator`e in `fixedPointBase`. La funzione principale di questa classe è `iterate` che risolve il problema fluido-struttura accoppiato mediante un algoritmo di punto fisso. Sia $n+1$ il passo temporale corrente e sia η^n la posizione dell'interfaccia al passo temporale n; prima di aplicare l'algoritmo di punto fisso si determinano un'estrapolazione della posizione dell'interfaccia $\hat{\boldsymbol{\eta}}$ e della velocità al passo temporale $n+1$:

$$\begin{cases} \hat{\boldsymbol{\eta}}^{n+1} = \boldsymbol{\eta}_s^n + \dot{\boldsymbol{\eta}}_s^n \Delta t & \text{n} = 1, \\ \hat{\boldsymbol{\eta}}^{n+1} = \boldsymbol{\eta}_s^n + \frac{1}{2}\Big(3\dot{\boldsymbol{\eta}}_s^n - \dot{\boldsymbol{\eta}}^n\Big)\Delta t & \text{n} > 1 \end{cases}$$

$$\dot{\hat{\boldsymbol{\eta}}}^{n+1} = \dot{\boldsymbol{\eta}}_s^n$$

dove $\boldsymbol{\eta}_s$ è la posizione dell'interfaccia della struttura.

Si applica l'algoritmo di punto fisso attraverso la chiamata del template `nonLinRichardson`. Posto con $k+1$ l'indice dell'iterata corrente, k l'indice dell'iterata precedente si definisce l'errore ad ogni iterazione e l'errore iniziale come

$$||err^{k+1}||_\infty = ||\boldsymbol{\eta}^{k+1} - \boldsymbol{\eta}^k||_\infty, \qquad ||err^{n+1}||_\infty = ||\boldsymbol{\eta}^{n+1} - \boldsymbol{\eta}^n||_\infty.$$

Il criterio d'arresto dell'algoritmo di punto fisso è:

$$||err^{k+1}||_\infty << \epsilon = toll_{abs} + toll_{rel}||err^{n+1}||_\infty$$

dove $toll_{abs}$ e $toll_{rel}$ sono la tolleranza assoluta e quella relativa definite nel file di dati, mentre $||err^{n+1}||_\infty = ||\boldsymbol{\eta}^{n+1} - \boldsymbol{\eta}^n||_\infty$. Posto $\boldsymbol{\eta}^k = \hat{\boldsymbol{\eta}}^{n+1}$ si calcola $\boldsymbol{\eta}^{n+1}$ nel seguente modo:

1. Si determina la velocità dell'interfaccia fluida all'iterata k-esima:

$$\boldsymbol{\eta}_f^k = \boldsymbol{\eta}^k$$

2. Si risolve il problema armonico imponendo lo spostamento dell'interfaccia fluida $\boldsymbol{\eta}_f^k$ come condizione di interfaccia, mentre si applica uno sforzo nullo sulle restanti faccie. In questo modo si determina lo spostamento dell'interfaccia fluida $\boldsymbol{\eta}_a^{k+1}$ all'iterata $k+1$.

3. Si trasferisce $\boldsymbol{\eta}_a^{k+1}$ al problema fluido, tramite l'operatore `transferMeshMotionOnFluid` ottenendo $\boldsymbol{\eta}_f^{k+1}$.

4. Si calcola la velocità del fluido all'iterata $k+1$:

$$\boldsymbol{u}^{k+1} = \frac{\boldsymbol{\eta}_f^{k+1} - \boldsymbol{\eta}_f^k}{\Delta t}$$

che per l'algoritmo di Dirichlet-Neumann diventa la condizione di interfaccia per il problema fluido. Si applica `moveMesh` (funzione per il partizionamento della mesh fluida su più processori) così da considerare anche i nodi della mappa ripetuta dell'EpetaVector associato perchè altrimenti c'è la possibilità che si perdino le informazioni dei nodi di connessione delle mesh partizionate.

5. Si determinano alcuni parametri per la risoluzione del problema fluido, `alpha` coefficiente della matrice di massa $(1/\Delta t)$, `M_beta` il termine convettivo, e `M_rhsNew`. Si determinano `alpha` e `M_rhsNew` attraverso il seguente frammento di codice:

```
double alpha = this->M_bdf->coeff_der( 0 ) / M_dataFluid->timestep();

*M_rhsNew   = *this->M_rhs;
*M_rhsNew  *= alpha;
```

mentre `M_beta` si determina dai seguenti passaggi:

```
*M_beta *= 0.;
vector_type const meshDispDiff( M_meshMotion->dispDiff(),
              this->M_meshMotion->getRepeatedEpetraMap() );
this->interpolateVelocity(meshDispDiff, *M_beta);
*M_beta *= -1./M_dataFluid->timestep();
*M_beta += this->M_bdf->extrap();
```

ossia una volta determinato il vettore differenza dello spostamento della mesh fluida tra due iterate successive, esso viene trasferito in `M_beta`, si moltiplica `M_beta` per $-1/\Delta t$ e infine si somma l'estrapolazione della velocitá del fluido.

Nel caso dello schema di Robin si calcola il prodotto tra veloctà del fluido per il valore di α_f. Questo termine può essere uno scalare o un vettore come nel caso trattato del arco aortico per questo motivo è stato implementato un metodo degli EpetraVector che permette di calcolare il prodotto componente per componete tra EpetraVector.

Infine si richiama la funzione `updateSystem` che trasferisce i parametri appena calcolati al problema fluido.

6. Si risolve il problema fluido attraverso la funzione `iterate` che determina il residuo del problema fluido res_f^{k+1}.

7. Si trasferisce il residuo fluido all'interfaccia con il comando `transferFluidOnInterface` ottenendo res_f^{k+1}. Attraverso res_f^{k+1} si determina il $r\hat{e}s_s^{k+1}$ che diventa la condizione di interfaccia del problema solido.

8. Si risolve il problema struttura con le opportune condizioni di bordo e si transferiscono le informazioni del problema struttura (spostamento, velocità e residuo) all'interfaccia attraverso il comando `transferSolidOnInterface` ottenendo $\boldsymbol{\eta}_s^{k+1}$, $\dot{\boldsymbol{\eta}}_s^{k+1}$ e res_s^{k+1}.

9. Si determina $\boldsymbol{\eta}^{n+1}$ e $\dot{\boldsymbol{\eta}}^{n+1}$ uguagliando i valori appena calcolati del problema struttura.

10. Se $\boldsymbol{\eta}^{k+1}$ soddisfa il criteri d'arresto si ha $\boldsymbol{\eta}^{n+1} = \boldsymbol{\eta}^{k+1}$ altrimenti si riparte da 1. Nel caso dell'algoritmo di Dirichlet-Neumann all'inizio di ogni nuova iterazioni ($k > 1$) si applica la procedura di rilassamento attraverso il metodo `solveJac`. Per determinare il parametro di rilassamnto ottimo si utilizza il metodo di *Aitken* (si veda [8]).

Una volta determinato $\boldsymbol{\eta}^{n+1}$ il ciclo di punto fisso termina e si passa al passo temporale successivo. Nel caso del trattamento delle condizioni assorbenti si determina il valore della pressione in uscita.

Oseen, Venant-Kirchhof, HarmonicExtension

In LifeV i programmi per l'implementazione e la risolzuione di problemi fisici sono contenuti in *lifesolver*. Queste classi hanno in comune i principali metodi che sono:

`initialize()` in cui vengono inizializzate le condizioni iniziali dei problemi.

`buildSystem()` e `updateSystem()`, questi metodi assemblano le matrici del sistema algebrico associate al problema, il primo metodo viene chiamato solo all'inizio, mentre il secondo le aggiorna ad ogni iterazione e definisce `MatrNoBC`, la matrice del sistema senza le condizioni di bordo.

`iterate()` ad ogni instante temporale risolve il sistema nel seguente modo: si determina `rhsNoBC` e si applicano le condizioni al bordo mediante la funzione `applyBoundaryCondition()` alla matrice e al vettore creando il `matrFull` e `rhsFull`. Si risolve il sistema attraverso il metodo `solveSystem()` e infine si calcola il residuo:

```
M_residual=M_rhsNoBC;  M_residual-=*M_matrNoBC*M_sol;
```

4.4.3 Condizioni di interfaccia

Sia nell'algoritmo di Dirichlet-Neumann che di Robin-Neumann le condizioni di interfaccia del problema solido e armonico sono le stesse e sono quelle originarie del codice:

All'interfaccia del problema armonico si è definito una condizione di Dirichlet, attraverso un `BCVectorInterface`, al quale è associato un EpetraVector. Questo vettore è determina lo spostamento della mesh solida dell'iterata precedente.

All'interfaccia del problema struttura si impone lo sforzo fluido dell'iterata corrente, attraverso un `BCVectorInterface` Natural di tipo `1 ossia che deve essere integrato rispetto la direzione normale.

Algoritmo di Dirichlet-Neumann

Nell'algoritmo di Dirichlet-Neumann per determinare le condizioni di Dirichlet al fluido basta imporre la seguente uguaglianza:

$$\boldsymbol{u}^{n+1} = \frac{\boldsymbol{\eta}^{n+1} - \boldsymbol{\eta}^{n}}{\Delta t} \qquad \text{su} \quad \Gamma$$

Inoltre va notato che durante il metodo del punto fisso per l'algoritmo di Dirichlet-Neumann si applica una procedura di rilassamento, così da permettere allo schema numerico di convergere. Più precisamente mediante il metodo di Aikten ([4], [7], [25]) è calcolato il valore ottimale del parametro di rilassamento w che viene usato per determinare il valore dello spostamento dell'interfaccia all'iterata corrente mediante la seguente formula:

$$\boldsymbol{\eta}^{k+1} = \boldsymbol{\eta}^{k} + w^{k}(T_s^{-1}(-T_f(\boldsymbol{\eta}^{K})))$$

dove $T_s(\boldsymbol{\eta}) = (\mathbf{T}_s(\boldsymbol{\eta}) \cdot \mathbf{n}_s)$ su Γ^0 e $T_f(\boldsymbol{\eta}) = (\hat{\mathbf{T}}_f(\boldsymbol{u}, p) \cdot \mathbf{n}_f)$.

Algoritmo di Robin-Neumann

Nell'algoritmo di Robin-Neumann sull'intefaccia fluida si deve imporre la seguente condizione di Robin:

$$\int_{\Gamma} \Big(\alpha_f \boldsymbol{u}^{n+1} + \mathbf{T}_f^{n+1} \cdot \mathbf{n}_f \Big) d\gamma = (\mathbf{f}_s^{n+1}, \mathbf{v})_{\Gamma}$$

Il termine forzante è:

$$(\mathbf{f}_s, \boldsymbol{v})_{\Gamma} = \int_{\Gamma} \alpha_f \frac{\boldsymbol{\eta}^{n+1} - \boldsymbol{\eta}^{n}}{\Delta t} \boldsymbol{v} d\gamma - \mathcal{R}_s(\boldsymbol{\eta}^{n+1})$$

dove $\mathcal{R}_s(\boldsymbol{\eta}^{n+1})$ è il residuo struttura, già integrato. Per determinare numericamente la condizione di Robin, sono state sfruttate le seguenti osservazioni:

quando si richiama la funzione `iterate` del problema struttura viene calcolato automaticamente il residuo $\mathcal{R}_s(\boldsymbol{\eta}^{n+1})$.

su uno stesso bordo si può applicare più di una condizione, generando una nuova condizione data dalla loro combinazione.

Per questi motivi, solo per applicare numericamente la condizione di Robin al problema fluido, si è deciso di scomporla in due condizioni una di Robin e una di Neumann.

La prima condizione è:

$$\alpha_f \boldsymbol{u} + (\mathbf{T}_s \cdot \mathbf{n}_s) \cdot \mathbf{n}_s = \alpha_f \frac{\boldsymbol{\eta}^{n+1} - \boldsymbol{\eta}^{n}}{\Delta t}$$

e viene trattata attraverso un `BCVectorInterface Mixte` così definito:

```
void FSIOperator::setHarmonicExtensionVelToFluidAlpha
                                (const vector_type &velalphaf,
                                 vector_type &alphaf,UInt type)
```

```
{
 M_bcvHarmonicExtensionVelToFluidAlpha->setup(
       velalphaf, M_fluid->velFESpace().dof().numTotalDof(),
       M_dofHarmonicExtensionToFluid, type);

 M_bcvHarmonicExtensionVelToFluidAlpha->setMixteVec(alphaf);
}
```

dove la funzione `setup` permette di inserire nel `BCVector` il valore della forzante associata alla condizione di Robin, che nel caso particolare è $\alpha_f \dot{\boldsymbol{\eta}}$, mentre in `setMixteVec` l'EpetraVector associato al coefficiente del termine di Robin, che in questo caso è α_f. Per determinare $\alpha_f \dot{\boldsymbol{\eta}}$ è stato definito overloadding dell'operatore `operator *=` degli EpetraVector così da poterlo determinare sia per un α_f scalare che vettoriale. Per definire il valore di α_f si è utilizzata la formula (2.4.2) con opportuni valori di β che per il tubo e la carotide sono stati scelti costanti e quindi definiti come `Real` e successivamente inseriti in un `BCVector` attraverso la funzione `PutScalar`. Nella geometria del toro β non è una varianile costante ma ipende dalla geometria secondo la formula analitica Allo scopo di costruire il *beta* per il toro si è definita una `BCFunction` e succesivamente è stata convertita in `EpetraVector` tramite il metodo `interpolate` della classe `FESpace` tramite il seguente metodo di Oseen:

```
setCurvatura(const Function& beta) {
  M_Curvatura.reset (new vector_type
              (*M_localMap.getRepeatedEpetra_Map()));
  M_uFESpace.interpolate(beta,M_betaVec, 0.0); }
```

dove `betaVec` è un `EpetraVector`. La funzione `interpolate` interpola la `Function` sull'`EpetraVector` corrispondente con una tolleranza data dal terzo elemto, ossia `0.0`.

`Function`, costruita tramite una funzione analitica

La seconda condizione è:

$$\int_\Gamma \mathbf{T}_f \cdot \mathbf{n}_f d\gamma = \mathcal{R}$$

che viene trattato come un `BCVectorInterface Natural` di tipo `0`, ossia che non deve essere integrato.

```
void FSIOperator::setStructureLoadToFluid(
             const vector_type &res,  UInt type){
M_bcvStructureLoadToFluid->setup( res,
        M_solid->dFESpace().dof().numTotalDof(),
        M_dofStructureToFluid,   type);}
```

Quando si determina la soluzione del punto fisso nel caso di Robin si è disattivato il rilassamneto in quanto non necessario a convergenza e inoltre se si volesse applicare il rilassamento andrebbe modificato il metodo di Aikten in modo tale che tenga conto non solo dello spostamento del ma anche della velocità di della

mesh.
il criterio di arresto dell'algoritmo di punto fisso è lo stesso dell'algoritmo di Dirchlet-Neumann sebbene in teoria andrebbe fatto su $\dot{\boldsymbol{\eta}}$ e non su $\boldsymbol{\eta}$, tuttavia vale la seguente disuguaglianza:

$$||\dot{\boldsymbol{\eta}}^{n+1} - \dot{\boldsymbol{\eta}}^{n}|| \leq ||\boldsymbol{\eta}^{n+1} - \boldsymbol{\eta}^{n}||$$

che garantisce la convergaenza dell'algoritmo imponendo il criterio d'arresto sulla posizione.

4.4.4 Condizini Assorbenti

Per il trattamento delle condizioni assorbenti si utilizza la formula (2.24), il valore dell'area e del flusso corrente sono ottenuti attrraverso i metodi: `area` e `flux`. Questi membri ricevono in ingresso il valore della faccia di bordo per cui si vuole ricevare il valore numerico. Per il determinare l'area si calcola l'integrale di bordo come somma della superficie dei triangoli della faccia. Usando la stessa procedutra per determinare il l'area si determina anche il flusso attraverso della velocità. Il metodo usato nelle simulazioni è esplicito ossia il valore della pressione viene calcolato all'inizio del passo temporale attraverso la solzuione (area e flusso) del passo temporale precedente. È stata testata anche una procedura completamente implicita, ossia ad ogni iterazione si impone il valore di della pressione, tuttavia il costo computazionale aumente e non sembra significativamente generare una soluzione migliore. Infine si è provato ad applicare il rilassamento del valore della presione ossia ad ogni instante temporale si determina il valore della pressione sulla faccia di uscita attraverso un parametro di rilassamento $\omega \in [0, 1]$:

$$P_{out}^{n+1} = P_{out}^{n}\omega + P_{out}^{n-1}(1 - \omega)$$

tuttavia per i valori testati non si è determianto nessun valore ottimale.

Bibliografia

[1] S. Badia, F. Nobile, C. Vergara *Fluid-structure partitioned procedures based on Robin transmission conditions*

[2] S.Badia, A. Quaini, A. Quarteroni *Splitting methods based on algebraic factorization for fluid-structure interaction* MOX-Report No. 03/2007

[3] P.Causin, J.F. Gerbeau, F. Nobile. *Add-mass effect in the design of partitioned algoritms for fluid-structure problems* Comp. Mech. Appl. Mech. Eng 194 (2005), 4506-4527.

[4] M. Cervera, R.Codina, M.Galoindo, *On the computational efficiency and implementation of block-iterative algorithms for non linear coupled problems,* Engrg, Comput. 13 (96) (1996) 4-30.

[5] P.G. Ciarlet, *Mathemataical elasticty. Studies in Mathematics and its Applications* vol II, 27, North-Holland Publishing Co., Amsterdam 1997

[6] A. Corno , M. Prosi, P. Fridez, P. Zunino, A. Quarteroni , L. von Segesser, *The non-circular shape of FlaWatch-Pab prevents the need for pulmonary artery reconstruction after banding. Computational fuid-dynamics and clinical correlaions,* Europ. J. Cardio-thoraic Surg., 2005, in press.

[7] S. Deparis, M. Discacciati, A. Quarteroni, *A domain decomposition framework for fluid-structure interaction problems.* in: Proceedings of the Third International Conference on Computational Fluid Dynamics (ICCFD3), Toronto, July 2004, 2004, submitted

[8] S. Deparis, M. Discacciati, G. Fourestey, A. Quarteroni, *Fluid-structure algorithms bases on Steklov-Poincaré operators* Comput. Methods Appl. Mech. Engrg., in press, 2006.

[9] S. Deparis, G. Fourestey, C. Winkelmann, *A parallel framework for the LifeV library* March 12,2008

[10] J. Donea , *An arbitrary lagrangian-eulerian finite element method for transient dynamic fluid-structure interactions*, Comp. Meth. Appl. Mech. Eng., 33 (1982), 689-723.

[11] M.A. Fernánder, J.F. Gerbeau, C. Grandmont *A projection semiimplicit scheme for the coupling of an elastic structure with an incompressible fluid.* International Journal for Numeraical Methods in Engineering, 69(4); 794-821, 2007.

[12] M.A. Fernández, M. Moubachir *A Newton method using exact Jacobians for solving fluid-structure coupling.* Computers and Structures, 83(2-3): 127-142, 2005.

[13] L. Formaggia, A. Veneziani, *Reduced and multiscale models for the human cardiovascular system*, Lecture Notes, VKI Lecture Series, 2003

[14] L. Formaggia, F. Nobile, A. Quarteroni, A. Veneziani, *Multiscale modelling of the circulatory system: a preliminary analysis*, Comp. Vis. Sc., 2 (1999), 75-83.

[15] L. Formaggia, J.F. Gerbeau, F. Nobile, A. Quarteroni, *On the coupling of 3D and 1D Navier-Stokes equations for flow prolems in compliant vessels,* Comp. Meth. Appl. Mech. Eng., 191 (2001), 561-582.

[16] L. Formaggia, F. Nobile, *A stability analysis for the arbitrary Lagrangian Eulerian formulation with finite elements,* East-West J. Numer. Math., 7(2) (1999), 105-131.

[17] L. Formaggia, F. Nobile, *Stability analysis of second-order time accurate schemes for ALE-FEM,* Comput. Methods Appl. Mech. Eng., 193(39-41) (2004), 4097-4116.

[18] L. Formaggia, A. Quarteroni, *Mathematical Modelling and Numerical Simulation of the Cardiovascular System, In Modelling of Living Systems,* Handbook of Numerical Analysis, Ayache N, Ciarlet PG, Lions JL (eds), Elsevier Science, Amsterdam, 2004.

[19] L. Formaggia, A. Quarteroni, A. Veneziani, *The circulatory system: from case studies to mathematical modeling*, In Integration of Complex Systems in Biomedicine, Formaggia L., Quarteroni A., Veneziani A. eds., Springer-Italia, Milano, 2006 (in press).

[20] Y.C. Fung, *Biomechanics: mechanical proprieties of living tissues.* Springer-Verlag, New York,1993

[21] T.J.R Hughes, W.K. Liu, T.K. Zimmermann, *Lagrangian-Eulerian finite element formulation for incompressible viscous flows*, Comp. Meth. Appl. Mech. Eng., 29(3) (1981), 329-349.

[22] J.G. Haywood , R. Rannacher, S. Turek, *Artificial Boundary and Flux and Pressure Conditions for the Incompressible Navier-Stokes Equations*, Int. Journ. Num. Meth. Fluids, 22 (1996), 325-352.

[23] K. Laganá, R. Balossino, F. Migliavacca, G. Pennati , E.L. Bove, M.R. de Leval, G. Dubini,*Multiscale modeling of the cardiovascular system: application to the study of pulmonary and coronary perfusions in the univentricular circulation*, J. of Biomech.; 38: 1129-1141, 2005.

[24] S. Mittal, T.E. Tezduyar, *Parallel finite element simulation of 3D incompressible flows: fluid-structure interactions*, Int. J. Numer. Methods Fluids, 21 (1995), 933-953.

[25] D. P. Mok, W. A. Wall, *Partitioned analysis schemes for the transient interaction of incompressible flows and nonlinear flexible structures,* in: K. Schweizerhof, W. Wall (Eds.), Trends in Computational Structural Mechanics, K.U. Bletzinger, CIMNE, Barcelona, 2001.

[26] F. Nobile, C. Vergara *An effective fluid-structure interaction formulation for vascular dynamics by generalized Robin conditions*. MOX report n. 89. Also submitted.

[27] F. Nobile, *Numerical Approximation of Fluid-Structure Interaction Problems with Application to Haemodynamics*, Ph.D. Thesis, EPFL, Lausanne, 2001.

[28] C.A. Taylor, M.T. Draney, J.U. Ku, D. Parker, B.N. Steele, K. Wang, C.K. Zarins *Predictive medicine: Computational techniques in therapeutic decisionmaking, Computer Aided Surgery*; 5(5): 231-247, 1999.

[29] P. Le Tallec , J. Mouro, *Fluid structure interaction with large structural displacements*, Comp. Meth. Appl. Mech. Eng., 190 (2001), 3039-3067.

[30] A. Quarteroni, G. Rozza, *Optimal control and shape optimization of aortocoronaric by-pass anastomosis*, M3AS; 13(12): 1801-1823, 2003.

[31] A. Quarteroni, M. Tuveri, A. Veneziani, *Computational Vascular Fluid Dynamics: Problems, Models and Methods*, Computing and Visualisation in Science; 2: 163-197,2000.

[32] A.Quarteroni, A.Valli *Numerical Approximation of Partial Differential Equations* Springer-Verlag, 1994

[33] A. Quarteroni, A. Veneziani, P. Zunino, *Mathematical and numerical modeling of solutes dynamics in blood flow and arterial wall*, SIAM J. Num. An; 39(5): 1488-1511, 2002.

[34] C. Vergara, P. Zunino *Multiscale modeling and simulation of drug release from cardiovascular stents* SIAM Multiscale Modeling and Simulation.

[35] P. Zunino, *Multidimensional pharmacokinetic models applied to the design of drug eluting stents*, Cardiov. Eng.; 4(2), 2004.