UNIVERSITÀ DEGLI STUDI DI PALERMO

Dottorato di Ricerca in Ingegneria Civile, Ambientale, dei Materiali
Ingegneria Idraulica e Ambientale
Dipartimento di Ingegneria
Settore Scientifico Disciplinare ICAR/01

SPH modeling of blood flow in cerebral aneurysms

IL DOTTORE
Ing. Alessandra Monteleone

IL COORDINATORE
Ch.mo Prof. Antonina Pirrotta

IL TUTOR
Ch.mo Prof. Enrico Napoli

CICLO XXXI
ANNO CONSEGUIMENTO TITOLO 2019

Abstract

Cerebral aneurysms are pathological dilations of brain arteries. These diseases carry an inherent risk of rupture with consequent intracranial hemorrhages. Although the mechanisms of initiation, growth and rupture of cerebral aneurysms are not well understood yet, it is commonly recognized that hemodynamic factors play a very important role in these processes.

Numerical simulations can provide useful information on the hemodynamics and can be used for clinical applications. In the traditional *grid-based* numerical methods the discretization process of cerebral vessels hosting an aneurysm is very challenging. On the other hand, the Lagrangian *mesh-less smoothed particle hydrodynamics* (*SPH*) is very suitable for representing geometrically complex domains, moving boundaries and multi-phase processes.

In this research study the *SPH* method is employed to model blood flow in cerebral aneurysms using an open-source code (*PANORMUS-SPH*). New algorithms and procedures are introduced in the code to improve the accuracy, stability and computational efficiency of the numerical model, focusing on cerebral aneurysm simulations.

The truly incompressible *SPH* (*ISPH*) approach is used to solve the discretized governing equations. To this aim, the mass conservation is enforced by solving a system of *Pressure Poisson Equations* using the preconditioned *BiConjugate Gradient STABilized* method.

A novel procedure is proposed to treat *open-boundaries* in *SPH*, allowing to set *Dirichlet* pressure boundary conditions at the inlet and outlet sections or to impose the velocity profile at the inflow. Mass conservation is guaranteed during the procedure, which is a challenging task in Lagrangian modeling of domains with *open-boundaries*.

An innovative multi-resolution technique is developed in the *SPH* model. This approach is based on the domain decomposition into subdomains in each of which a proper refinement is used. The technique is crucial when dealing with geometrically irregular domains, such as cerebral vessels, for which the use of a uniform particle distribution may become unsustainable in terms of *CPU* time and memory requirements.

The computational efficiency of the *SPH* code is largely improved through its complete parallelization based on the *Message Passing Interface* paradigm. The implemented domain distribution algorithm ensures a well-balanced load even with highly irregular domains subdivided through the multi-resolution approach.

In *endovascular* treatments of cerebral aneurysms the stability of the blood clot forming inside the aneurysm sac is a key factor for the healing process. The analysis of the blood clotting process is thus extremely important to evaluate the treatment outcomes. In this study *tracer transport*, *residence time* and *mechanical platelet activation* models are implemented in the *SPH* code in order to lay the groundwork for a future *SPH*-based blood clot model.

A performance evaluation of the implemented numerical improvements is conducted through comparison of the results with available analytical and numerical solutions and with experimental measures obtained in benchmark test cases including also ideal and real aneurysm geometries.

Sommario

Gli aneurismi cerebrali sono dilatazioni patologiche di arterie cerebrali. Queste patologie hanno un intrinseco rischio di rottura con conseguenti emorragie intracraniche. Sebbene i meccanismi di formazione, crescita e rottura degli aneurismi cerebrali non sono ancora del tutto compresi, è comunemente riconosciuto che in questi processi i fattori emodinamici giocano un ruolo molto importante.

Le simulazioni numeriche possono fornire utili informazioni sull'emodinamica e possono essere usate per applicazioni cliniche. Nei tradizionali metodi numerici basati su una griglia di calcolo il processo di discretizzazione dei vasi cerebrali sui quali insiste un aneurisma è molto complesso. D'altra parte, il metodo Lagrangiano *smoothed particle hydrodynamics* (*SPH*) è particolarmente adatto per la rappresentazione di domini geometricamente molto complessi, contorni mobili e processi multi-fase.

In questo studio di ricerca il metodo *SPH* viene impiegato per modellare il flusso sanguigno all'interno di aneurismi cerebrali utilizzando un codice di calcolo *open-source* (*PANORMUS-SPH*). All'interno di questo codice sono introdotti nuovi algoritmi e procedure per migliorarne l'accuratezza, la stabilità e l'efficienza computazionale al fine di effettuare simulazioni di aneurismi cerebrali.

Per risolvere le equazioni guida discretizzate viene adottato l'approccio totalmente incomprimibile (*ISPH*). La conservazione della massa è imposta risolvendo un sistema di equazioni di *Poisson* attraverso il metodo del *gradiente biconiugato stabilizzato* con un algoritmo di precondizionamento.

Si propone una procedura innovativa per trattare confini aperti nel metodo *SPH*, consentendo di imporre condizioni di *Dirichlet* per la pressione nelle sezioni di ingresso e di uscita oppure il profilo di velocità all'ingresso. La conservazione della massa, che è un compito arduo nella modellazione Lagrangiana di domini con confini aperti, è garantita durante la procedura.

Un'originale tecnica multi-risoluzione è sviluppata nel modello *SPH*. Questo approccio è basato sulla decomposizione del dominio in subdomini in ognuno dei quali viene utilizzato un appropriato livello di discretizzazione. La tecnica è cruciale quando si studiano domini geometricamente irregolari, come i vasi cerebrali, per cui l'uso di un'uniforme dimensione delle particelle potrebbe risultare insostenibile in termini di costi computazionali e richieste di memoria.

L'efficienza computazionale del codice *SPH* è notevolmente migliorata attraverso la sua parallelizzazione basata sul paradigma *Message Passing Interface*. L'algoritmo di distribuzione del dominio permette di ottenere un carico ben bilanciato anche con domini altamente irregolari suddivisi attraverso l'approccio multi-risoluzione.

Nei trattamenti *endovascolari* di aneurismi cerebrali la stabilità del coagulo di sangue che si forma all'interno della sacca è un fattore chiave per il processo di guarigione. Pertanto, l'analisi del processo di coagulazione è estremamente importante per valutare i risultati del trattamento. Sfruttando la natura Lagrangiana del metodo *SPH*, in questo studio sono implementati i modelli del trasporto di un tracciate, del tempo di residenza e dell'attivazione meccanica delle piastrine, ciò al fine di gettare le basi per il futuro sviluppo di un modello di coagulo di sangue.

Una valutazione delle prestazioni dei miglioramenti numerici implementati è condotta confrontando i risultati di casi test di riferimento, che includono anche geometrie di aneurismi ideali e reali, con soluzioni analitiche e numeriche disponibili e con misure sperimentali.

Acknowledgements / Ringraziamenti

I would like to thank Dr. Francesca Graziano, an excellent neurosurgeon at the "Policlinico Paolo Giaccone" of Palermo. The topic of this research study was born from her passion in the field of cerebral aneurysms. Her enthusiasm involved me in the study of this pathology, with the hope of future clinical applications.

I would like to express my profound gratitude to my supervisor Prof. Enrico Napoli for his patient guidance, understanding, and friendship. He has believed in me and he constantly encourages me. I will treasure his teachings for a lifetime.

I would like to gratefully acknowledge Prof. Alejandro Frangi for the visiting period I could spend working at the University of Sheffield (UK) and for the valuable support offered. I would also like to thank all of the members of the Prof. Frangi's research group, I especially thank Ali Sarrami Foroushani and Toni Lassila for helping me and supporting me during my period abroad. The study conducted at the University of Sheffield was very precious for my thesis.

I would like to gratefully acknowledge Prof. György Paál and his group for having welcomed me at Budapest University of Technology and Economics for a research meeting. On this occasion Prof. Paál and his researchers involved me in their studies on cerebral aneurysms. Moreover, I had the privilege to see a cerebral angiography and an endovascular intervention in a patient with cerebral aneurysm carried out by the well-known neurosurgeon Prof. Dr. István Szikora. I have to thank him for this opportunity.

With reference to the laboratory application shown in this thesis, I wish to thank Prof. Salvatore Vitabile and Prof. Massimiliano Zingales for making it possible.

I wish to thank Prof. Mauro De Marchis, who encouraged me throughout my PhD, for his support on my research study.

I would like to thank all my colleagues, in particular I would like to remind here Massimiliano Monteforte for helping me and supporting me during the first year of my PhD.

For sure I have to thank my family and my love Dario; this research would not have been possible without their understanding, devotion and great love.

Desidero ringraziare la Dr. Francesca Graziano, un eccellente neurochirurgo al Policlinico "Paolo Giaccone" di Palermo. Il tema di questo studio di ricerca è frutto della sua passione nel campo degli aneurismi cerebrali. Il suo entusiasmo mi ha coinvolto nello studio di questa patologia, con la speranza di future applicazioni cliniche.

Vorrei esprimere la mia più profonda gratitudine al mio tutor Prof. Enrico Napoli per la sua paziente guida, comprensione ed amicizia. Lui ha creduto in me e mi ha costantemente incoraggiata. Farò tesoro dei suoi insegnamenti per tutta la vita.

Mi piacerebbe ringraziare il Prof. Alejandro Frangi per il periodo di studio che ho trascorso all'Università di Sheffield (UK) e per il prezioso supporto che mi ha offerto. Vorrei anche ringraziare tutti i membri del suo gruppo di ricerca, in particolare ringrazio Ali Sarrami Foroushani e Toni Lassila per avermi aiutato e supportato durante il mio periodo all'estero. Lo studio condotto all'Università di Sheffield è stato davvero prezioso per la mia tesi.

Ringrazio il Prof. György Paál e il suo gruppo di ricerca per avermi accolta all'Università di Budapest per un incontro di ricerca. Durante questa occasione il Prof. Paál e i suoi ricercatori mi hanno coivolto nei loro studi sugli aneurismi cerebrali. Inoltre, ho avuto il privilegio di assistere ad un'angiografia cerebrale e ad un intervento endovascolare su un

paziente con un aneurisma cerebrale effettuati dal noto neurochirurgo Dr. István Szikora. Devo infatti ringraziarlo per questa opportunità.

Con riferimento all'applicazione di laboratorio mostrata in questa tesi, desidero ringraziare i Proff. Salvatore Vitabile e Massimiliano Zingales per aver reso questo possibile.

Ed inoltre, desidero ringraziare il Prof. Mauro De Marchis che mi ha incoraggiato e supportato durante il mio Dottorato di Ricerca.

Desidero ringraziare i miei colleghi, in particolare vorrei qui ricordare Massimiliano Monteforte per avermi aiutato e supportato durante il primo anno del mio Dottorato di Ricerca.

Di certo devo ringraziare la mia famiglia e il mio amore Dario; questo studio non sarebbe stato possibile senza la loro comprensione, dedizione, ed immenso amore.

Methodological note

In this research study an existing *open-source* code has been further developed and specialized to some classes of fluid dynamic problems mainly related to the hemodynamics in cerebral aneurysms. The code, named *PANORMUS* (PArallel Numerical Open-souRce Model for Unsteady flow Simulations) was developed at the Department of Civil, Environmental, Aerospace, Materials Engineering (*DICAM*) of the University of Palermo (Italy). The *PANORMUS* model is distributed under the General Public Licence (*GPL*) and is available at http://www.panormus3d.org.

The numerical model contains a Finite-Volume package, named *PANORMUS-FVM* (Napoli, 2011), and a fully incompressible smoothed particle hydrodynamics (*SPH*) solver, *PANORMUS-SPH* (Napoli et al., 2015). In the *PANORMUS* software the equation solvers are written in the *FORTRAN* 95 programming language while the *Graphical User Interface* is written in C^{++} and based on the multiplatform $Qt^{®}$ library (https://www.qt.io).

New algorithms and procedures have been developed in the *SPH* solver to model cerebral aneurysms (the motivations for the choice of the *SPH* technique will be discussed in Chap. 1).

Specifically, the list of the implemented algorithms in the *PANORMUS* code is shown below.

- *Pressure Poisson equation* resolution (see Chap. 2). Implementation of the *Unpreconditioned* and *Preconditioned BiCGSTAB* method in *SPH*, and *FVM* solvers;
- Adaptive time step technique in *SPH* (see Chap. 2);
- Inflow/outflow boundary treatment for *SPH* (see Chap. 3);
- *Multi-Domain SPH* approach (see Chap. 4);
- Parallelization of the *SPH* model using the *Message Passing Interface* (*MPI*) libraries (see Chap. 5);
- *Tracer transport* and *residence time* models in *SPH* (see Chap. 7);
- *Mechanical platelet activation* model in the *SPH* framework (see Chap. 7);
- *Coupled FVM-SPH* method (see Appendix A).

Glossary

AP	*Activation potential*
BiCGSTAB	*BiConjugate Gradient STABilized*
BC	*Boundary condition*
CA	*Cerebral aneurysm*
CFD	*Computational fluid dynamics*
CFL	*Courant-Friedrichs-Levy*
CPU	*Central processing unit*
CRS	*Compressed Row Storage*
FD	*Flow diverter*
FVM	*Finite Volume Method*
HPC	*High-performance computing*
In/OutFlow-BCs	*Inflow/Outflow boundary conditions*
ISPH	*Incompressible smoothed particle hydrodynamics*
MPI	*Message Passing Interface*
MD	*Multi-Domain*
OSI	*Oscillatory shear index*
PPE	*Pressure Poisson Equation*
Pre-BiCGSTAB	*Preconditioned BiCGSTAB*
RHS	*right-hand-side*
RT	*Residence time*
SD	*Single-Domain*
SPH	*Smoothed particle hydrodynamics*
STL	*Standard Triangle Language*
TAWSS	*Time averaged wall shear stress*
TAWSSG	*Gradient of time averaged wall shear stress*
transWSS	*Transverse wall shear stress*
WCSPH	*Weakly compressible smoothed particle hydrodynamics*
WSS	*Wall shear stress*
WSSPI	*Wall shear stress pulsatility index*

Symbols

Δt	time step
" $\cdot$ "	if it is used in equations it indicates the scalar product
ceiling	if it is used in equations it indicates the operation of rounding up
W	*kernel function*
h	*smoothing length*
Δx	*starting particle distance*
i	i-th interpolating particle
m_i	mass of i
ρ_i	density of i
ν_i	kinematic viscosity of i
μ_i	dynamic viscosity of i
Ω_i	*Support domain* of i
j	j-th particle lying in Ω_i
d_{ij}	distance between the particles i and j
W_{ij}	*kernel function* at the distance d_{ij}
$\mathbf{u}_i^*$	*intermediate* velocity of i
$\mathbf{u}_i$	*corrected* velocity of i
p_i	pressure of i
ψ_i	*kinematic pressure* of i. It is equal to $\psi = p_i/\rho_i$ when the *predictor-step* equations are solved using an explicit algorithm
N_e	total number of *effective* particles in the domain
N_{mirror}	total number of *mirror* particles
$N_{species}$	number of species in the *tracer transport* model
$\mathbf{Cnc}_i$	concentration vector of the particle i

The vectors are highlighted with bold letters. Considering the vector **vect** of length n:

- $vect(l)$ indicates the value of **vect** at the position l;
- **vect**(:), that is equal to **vect**$(1:n)$, indicates all the components of **vect**;
- **vect**$(l:m)$ indicates the components of **vect** from the positions l up to m.

List of symbols used to classify the particles:

e	*effective* particles
m	*mirror* particles
IO	*in/out-flow* particles
IP	*multi-domain interface* particles
PP	*parallel* particles
PeP	*parallel effective* particles
PmP	*parallel mirror* particles
IP^*	*received interface* particles

List of symbols used with reference to the particles inside Ω_i:

s	an *effective* particles into Ω_i
g	the *effective* particle generating *mirror* and/or IO particles which lie in Ω_i
N	total number of particles lying into Ω_i
N_i	number of *effective* and *mirror* particles in Ω_i
N_i^e	number of *effective* particles in Ω_i
$N_i^{'}$	number of *effective* and *mirror* (with $g \neq i$) particles in Ω_i
N_i^M	number of *mirror* particles in Ω_i, $\forall g$
N_i^{Ms}	number of *mirror* particles in Ω_i, with $g = s$
$N_i^{M_{(g \neq s)}}$	number of *mirror* particles in Ω_i, with $g \neq s$
N_i^{IO}	number of *in/out-flow particles* in Ω_i, $\forall g$
N_i^{IOs}	number of *in/out-flow pressure particles* in Ω_i, with $g = s$
$N_i^{IO_{(g \neq s)}}$	number of *in/out-flow pressure particles* in Ω_i, with $g \neq s$
N_i^{IP}	number of *interface multi-domain particles* in Ω_i
N_i^*	number of *effective*, IO *pressure*, IP and PeP particles in Ω_i
N_i^{PP}	number of *parallel* particles in Ω_i
N_i^{PmPs}	number of *parallel mirror* particles PmP into Ω_i generated by the *parallel effective* particle s
N_i^{M+PmP}	number of *mirror* and *parallel mirror* particles in Ω_i, $\forall g$

List of symbols used in the *Multi-Domain* approach and in the *parallel computing* algorithm (see Chap. 4 and Chap. 5):

N_{procs}	number of processors
$myid$	current processor
id	index of the processor (from 0 to $N_{procs}-1$)
N^t	theoretical number of *effective* particles in each processor domain
res	number of the remaining *effective* particles in the last cell assigned to the processor id
$N_{(id)}$	number of *effective* particles in the domain of id
$N_{mirror,id}$	number of *mirror* particles in the processor id
$N_{PP,tot}$	total number of *parallel* particles received from the neighboring processors (as sum of left and right)
N_{PeP^L}	number of *effective parallel* particles received from the left
N_{PeP^R}	number of *effective parallel* particles received from the right
$N_{PeP,tot}$	total number of *effective parallel* particles received. It is equal to $N_{PeP,tot} = N_{PeP^L} + N_{PeP^R}$
N_{Blocks}	number of blocks in the *Multi-Domain* approach
B_n	generic block (or subdomain)
N_{e,B_n}	number of *effective* particles in the block B_n
$N_{e,tot}$	total number of *effective* particles in the whole domain as sum of the N_{e,B_n} with $B_n = 1,..,N_{Blocks}$
N^a_{Bn}	number of the available *effective* particles in the block B_n (not yet assigned to any processor)
lb	index of the block from which the IP has been generated
ib	index of the block where the IP is contained
$N^{ib}_{IP,id}$	number of IP particles contained in the block ib and in the domain of the processor id
$N^{tot}_{IP,id}$	total number of IP particles contained in the domain of id
$N^{tot}_{IP,tot}$	total number of IP particles generated by the current processor $myid$ as sum of the $N^{tot}_{IP,id}$ with $id = 0,..,N_{procs}-1$
$N^{*ib}_{IP,id}$	number of IP received by id that $myid$ must solve using the values of the particles in the block ib
$N^{*tot}_{IP,id}$	total number of IP received by id
$N^{*tot}_{IP,tot}$	total number of IP received as sum of the $N^{*tot}_{IP,id}$, $id = 0,..,N_{procs}-1$

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Generalities on blood flow circulation

Blood flow in the vascular system is unsteady and pulsatile due to the pumping action of the heart (Ku, 1997; Kundu et al., 2016). During a cardiac cycle, pressure and velocity fields vary periodically with time in all arteries. Although blood flow is not perfectly periodic, since it is adjusted to meet the body's blood demand, the approximation to a periodic phenomenon can be accepted when considering short periods of time, since the overall physical conditions change in a limited way.

In this compound, the *Womersley number* $Wo = D/2\sqrt{\omega/\nu}$ (where D is the reference mean diameter of the vessel, ν is the kinetic viscosity and ω is the frequency of the pulse wave) is generally used to characterize the pulsatile nature of blood flow. This dimensionless parameter denotes the ratio of unsteady inertial to viscous forces in the flow. When $Wo < 1$ viscous effects are dominant and the flow can be considered as quasi-steady since the frequency of pulsations is sufficiently low for the development of a parabolic velocity profile. In such flow the *Poiseuille's law* is thus reasonable applicable. On the other hand, with increasing Wo inertial forces become not negligible; therefore, for $Wo > 10$ the velocity profiles tend to be flat rather than parabolic since they do not have enough time to develop during the cardiac cycle. The typical *Womersley number* in cerebral vessels is of the order of unity (Shojima et al., 2005; Shimogonya et al., 2009; Passerini et al., 2012; Steinman et al., 2003); therefore, the analysis of pulsatile flows in the cerebral circulatory system is commonly accomplished using the *Womersley* solution (Womersley, 1955).

Beside the pulsatile conditions which prevent developing the parabolic velocity profile, a steady blood flow which enters from a large artery into a smaller branch could have even a flat velocity profile. In fact, a steady flow takes a certain distance, which is called the *entrance length* l, before becoming steady and fully developed with a parabolic velocity profile. Within the *entrance length*, the velocity profile changes along the flow and large velocity gradients (and thus high wall-shear stresses) exist near the wall. Inside the *entrance length* a boundary layer is formed since the fluid close to the wall is affected by the viscous forces. On the other hand, in the central region of the vessel the velocity profile is essentially flat. When the blood continues to flow along the *entrance length*, the boundary layer progressively grows in thickness since the viscous shear stress affects more fluid. At the end of the *entrance length*, where the flow is fully developed, the boundary layer occupies the whole vessel. For steady flow at low *Reynolds number* (parameter obtained through the entrance flow conditions, $Re = \bar{u}D/\nu$, where $\bar{u}$ is the mean sectional velocity),

the *entrance length* can be estimated as $l = 0.06\,Re\,D$ (Kundu et al., 2016). Considering a cerebral vessel with inlet diameter of 4 mm and $Re = 350$, the *entrance length* is approximately 21 diameters long. Cerebral vessels are not straight but are curved and have several branches, therefore, blood flow is in general not well developed. For pulsatile flow the *entrance length* depends on the *Reynolds number* as well as on the *Womersley number*, as discussed by Ku (1997).

In cerebral circulation, blood flows inside irregular vessels having a lot of curves, branches and bifurcations. Moreover, the diameter of the arteries forming the *Circle of Willis* is very variable; it is ranging between 1.5 mm and 5 mm (Kamath, 1981), while in several smaller vessels, originated from the larger ones, it is even less than 1 mm (Lang, 2001).

Laminar flow is commonly assumed in cerebral arteries; this assumption is widely justified due to the values of the *Reynolds number* being far lower than the threshold value of $2\,000 \div 2\,500$.

Since the characterization of patient-specific flow conditions is not part of the routine clinical examinations, the estimation of the blood flow rate in cerebral vessels is usually accomplished by "typical" waveforms, where the flow rates are scaled to the area of the inflow artery through empirical relationships (Cebral et al., 2008) or based on the principle of minimum work (Oka and Nakai, 1987). A typical mean flow rate over a cardiac cycle in the *Internal Carotid Artery* (*ICA*) is about 240 ml/min (Marshall et al., 2004; Hendrikse et al., 2005; Cebral et al., 2008; Damiano et al., 2015), which corresponds to an average velocity $\bar{u} = 0.32\ m/s$ considering a diameter of 4 mm. In the *ICA* a value of $Re = 350$ is assumed to be representative of the blood flow regime in the range of physiological values (Passerini et al., 2012). In the *Middle Cerebral Artery* (*MCA*) a flow rate of 120 ml/min has been measured by Stock et al. (2000) through phase-contrast MR imaging. As a consequence, average $\bar{u} = 0.41\ m/s$ and $Re = 270$ can be assumed with a mean diameter of 2.7 mm (Lang, 2001).

1.2 Cerebral aneurysms

1.2.1 Generalities

Cerebral aneurysm (*CA*) is a vascular disease characterized by local dilatation of the cerebral arterial walls. This pathology afflicts approximately $2 \div 3\,\%$ of the general population (Rinkel, 2008).

The aneurysms can be classified in different frames considering their shape and size. The most common aneurysm has saccular shape, it is thus named *saccular* (or *berry*) aneurysm. The *saccular* aneurysm is characterized by a sack sticking from the side of a blood vessel wall connected through the so-called *neck region*. Less frequently, the aneurysms have fusiform shape (Xu et al., 2018); in this case it is not identified a *neck region* and the artery walls expand in all directions involving a longer vessel segment. According to their size, the aneurysms can be classified as *tiny* (having diameter $D_a < 3\ mm$), *small* ($3 \leq D_a < 5\ mm$), *medium* ($5 \leq D_a < 13\ mm$), *large* ($13 \leq D_a < 25\ mm$) and *giant* ($D_a \geq 25\ mm$) (Badry et al., 2014).

Aneurysms predominantly occur at bifurcations apices or at the outer walls of arterial curvatures in or near the *Circle of Willis* (Chason, 1958; Stehbens, 1972; Foutrakis et al., 1999). In particular, most brain aneurysms form in the anterior circulation (Rinkel et al., 1998). The Fig. 1.1.a shows the *Circle of Willis* with the indication of *saccular* and

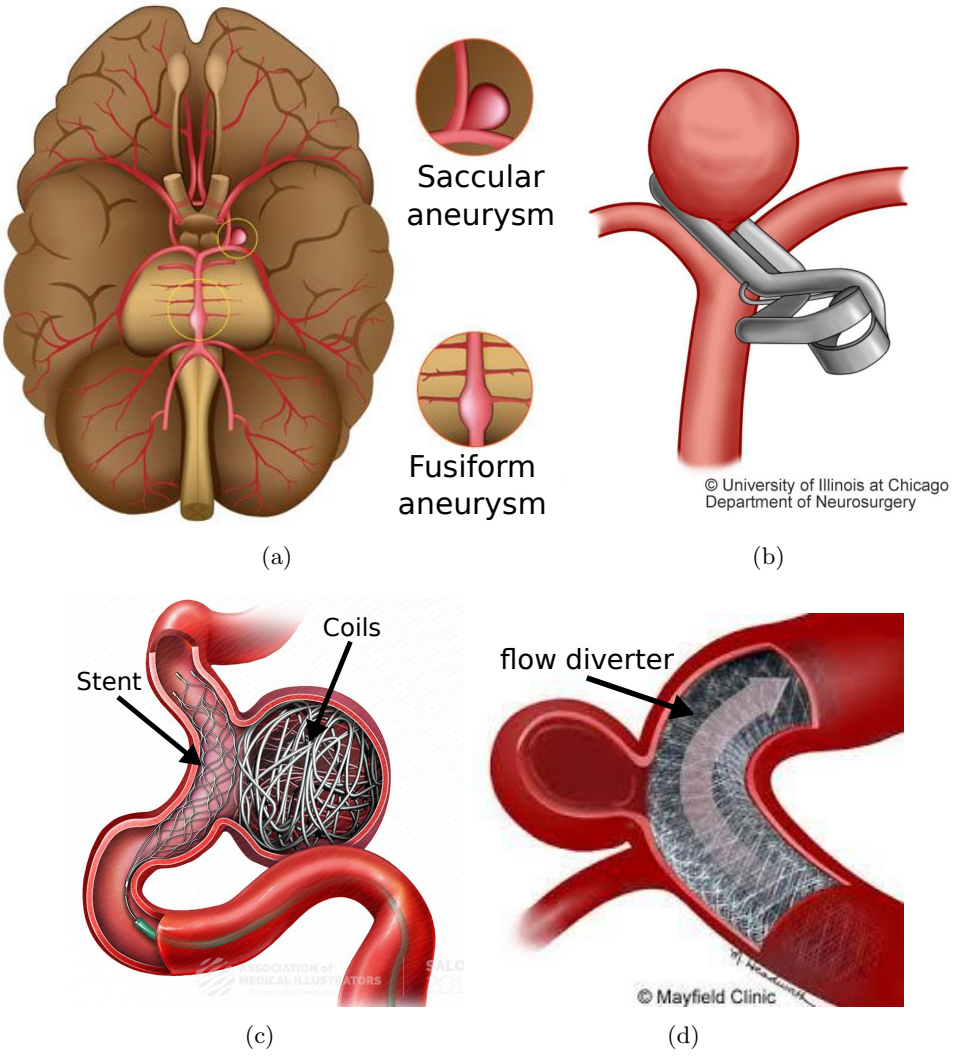


Figure 1.1: a) *Circle of Willis* with the indication of *saccular* and *fusiform* aneurysms (taken from: https://orangecountysurgeons.org); b) Surgical procedure. Aneurysm *clipping* (taken from: https://www.bafound.org); c) *Endovascular* treatment. *Stent-assisted coiling* (taken from: https://ami.org); d) *Endovascular* treatment. Aneurysm *flow diversion* (taken from: https://www.fda.gov).

fusiform aneurysms.

The main risk factors for aneurysm formation and rupture are positive family history, hypertension, older age, female gender, population, smoking, alcohol abuse, connective tissue disorder and blood vessel injury. Rupture is influenced also by aneurysm morphological characteristics including size, shape, and location (Wiebers and Marsh, 1998; Wiebers, 2003; Feigin et al., 2005; Toth and Cerejo, 2018).

The largest aneurysms can be discovered since they cause the compression of surrounding brain structure. However, most of these diseases are silent throughout the patient's life because they are asymptomatic. In the most catastrophic events they rupture and lead to intracranial hemorrhage into the *subarachnoid* space (named *subarachnoid hemorrhage*). The *subarachnoid hemorrhage*, whose main cause arises from the rupture of a cerebral aneurysm (Rinkel et al., 2001), has an associated high mortality and morbidity

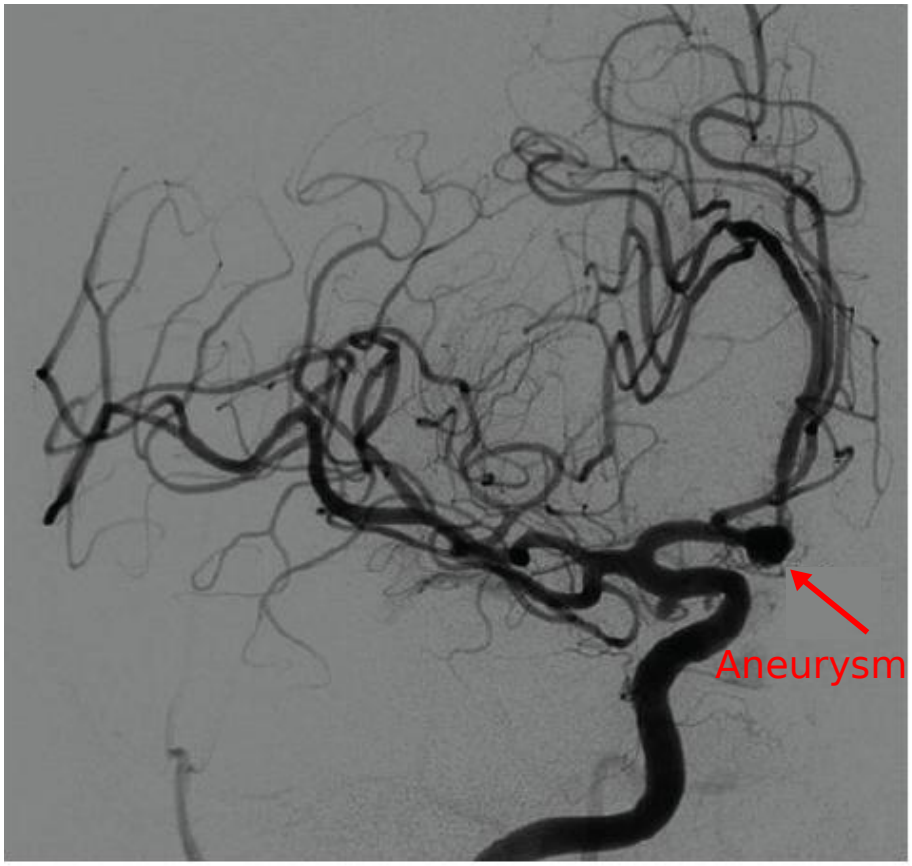


Figure 1.2: *Digital subtraction angiography* (taken from: http://www.scielo.org.za).

risk (Rinkel et al., 1998; Meng et al., 2013).

In order to detect *CAs* different techniques can be used. *Digital subtraction angiography* (see Fig. 1.2) is widely considered the current gold standard for detecting brain aneurysms (Walkoff et al., 2016; Goertz et al., 2018; Toth and Cerejo, 2018; Turan et al., 2018), although the $3D$ *rotational angiography* is able to depict smaller (diameter < 3 mm) additional aneurysms (Rooij et al., 2008).

1.2.2 The role of Wall Shear Stress

It is widely recognized in the literature that hemodynamic factors including *Wall Shear Stress* (*WSS*), vorticity, jetting, recirculation and pressure fluctuations play a very important role in the initiation, growth, and rupture of *CAs* (Jou et al., 2008; Cebral et al., 2011; Xiang et al., 2011; Munarriz et al., 2016). Vessel walls remodel their structure in order to adaptively respond to shear stress variations: if *WSS* locally increase, a punctual enlargement of the wall may take place. Moreover, *WSS* regulates endothelial functions. Therefore, it is fundamental to understand the *WSS* distribution on the vessel walls.

The mechanisms of aneurysm growth and rupture are not well understood yet. Specifically, it is not clear if either low or high shear stresses have to be considered the main triggers since the progression and rupture of *CAs* have been associated with zones of the aneurysm wall exposed to both high and low *WSS* (Meng et al., 2013; Munarriz et al., 2016). On one hand, under high shear stresses the arterial walls can weaken due to

biochemical processes and the prevalence of blood pressure forces over internal wall resistance can occur. On the other hand, blood stagnation in the aneurysm sac as result of low blood flows can lead to aggregation of red cells, accumulation and adhesion of platelets and leukocytes along the intimal surface. This process could imply the progressively wall tissue degeneration making the vessel walls no longer able to support blood pressure.

1.2.3 Therapeutic treatments

Choosing the optimal management for each patient requires the careful consideration of several factors such as patient's age and medical conditions, as well as aneurysm size and shape (*dome-neck ratio*) and site (Cowan J. et al., 2007; Rinkel, 2008; Shamloo et al., 2017; Toth and Cerejo, 2018).

In order to prevent the rupture of detected *CAs*, the therapeutic decisions are mainly *surgical clipping* or *endovascular* procedures (such as *coils* and/or *stent* and *flow diverter* devices). Unfortunately, both *endovascular* and *surgical* treatments carry the risk of associated morbidity and mortality; therefore, the treatment of complex *CAs* remains challenging. Sometimes, when the risk associated to the treatment is higher than the natural risk of rupture, the medical follow-up is chosen to check, using detection procedures, if the aneurysm is enlarging (Wiebers et al., 2003).

Surgery mainly involves placing a metallic *clip* across the aneurysm neck to separate it from the parent artery circulation (see Fig. 1.1.b). It is recommended after *CA* rupture.

Endovascular approaches are increasingly used, as they are less invasive compared to open surgery. *Interventional thromboembolization* treatments are used to stabilize the disease, promoting thrombus formation via the endovascular insertion of *coils* and *stents* or *flow-diverter* (*FD*) devices.

Coils technique involves filling the sac with platinum preshaped wires inducing the blood coagulation inside the aneurysm sac. These wires are often kept in the cavity putting meshes made up of surgical grade metal (*stent*) in the parent vessel across the neck of the aneurysm (see Fig. 1.1.c). Due to the *CA* shape and dimension, sometimes coiling has significant limitations in achieving durable occlusion. This is the case of aneurysms characterized by wide neck or fusiform shape or giant dimension (Fiorella et al., 2008; Becske et al., 2013). Furthermore, this procedure can lead to complications such as coil compaction which implies the re-growth of the aneurysm or the formation of a second pathological dilatation.

The *FD* devices aim at changing intra-aneurysmal flow deviating it away from the sac and promoting the thrombosis of the aneurysm (Vanninen et al., 2003). Moreover, *FD* structure provides neointimal growth over its struts allowing the parent vessel reconstruction. *FD* design is different from the conventional *stents* used as scaffold for *coils*. This devices are characterized by very thin wires braided like a mesh (see Fig. 1.1.d), whose level of compaction is related to the configuration assumed during the release of the *FD* from a microcatheter owing to its deformable mesh structure. Despite the flow diversion technique has shown great promise and has been adopted by much of the *neuroendovascular* community, some apparently successful cases have worsened later as result of incomplete or prolonged occlusions as well as of post-procedure *delayed hemorrhage* due to the rupture of the *CA* (Pierot, 2011; Siddiqui et al., 2012; Briganti et al., 2015; Raymond et al., 2016; Rouchaud et al., 2016; Kiselev et al., 2018). The triggers of the post-procedure rupture of the aneurysm, which has a significant morbidity risk (White et al., 2018), are poorly understood. Causal factors responsible for these clinically devastating complications have been only suggested. They are likely related to multiple factors working synergistically

such as anti-aggregation therapy (that is necessary to prevent *thromboembolism*), patient's individual anatomy and *FD* size and compaction level (Turowski et al., 2011). Concerning the latter aspect, compacting the *FD* mesh is an emerging technique to create a denser wire configurations across the aneurysm neck in order to promote a higher flow reduction inside the sac of the *CA* (Zhang et al., 2017). On the other hand, it was found that an excessive and abrupt stagnation of blood promotes the formation of non-organized and unstable *red thrombi* that are characterized by the predominance of *red blood cells*. In this respect, platelet content of the clot is thought to determine the clot stability promoting the formation of the so-called *white thrombi* that facilitate the healing process after aneurysm flow diversion. Therefore, the platelets play a very important role in the process of aneurysm occlusion. The activation of platelets can occur through both *chemical* and *mechanical stimuli*. As discussed by Xiang et al. (2014), whilst maximum *FD* compaction at the *CA* neck can improve aneurysmal bulk flow reduction, potentially accelerating the dome thrombotic occlusion, on the other hand, an excessively dense mesh of the device reduces also the blood shear stress generated near the *FD* struts which could mechanically activate platelets. Thus, attention should be paid not only to the aneurysmal bulk flow reduction but also to the shear-induced activation of platelets near the *FD* struts (which could potentially generate *white thrombus*). To the author's best knowledge, yet no study exists in the literature investigating this latter fundamental aspect.

1.3 Computational fluid dynamics

The analysis of the problems mentioned above requires a detailed description of the hydrodynamic phenomena developing in cerebral vessels with the presence of a *CA*. Since the governing equations of hemodynamics are *non-linear partial differential equations* and they can not be solved analytically, one possible strategy is to use the *Computational Fluid Dynamics* (*CFD*). The *CFD* allows to obtain a complete description of the velocity and pressure fields as well as of the shear-stress on the vessel walls. Indeed, the investigation of a vast range of engineering and science problems using *CFD* analysis is an active and very interesting research area allowing to solve the complex equations of the fluid dynamics. The *CFD* encompasses different numerical approaches which can be broadly framed into *grid-based* and *mesh-less* methods. While the conventional *grid-based* methods are characterized by large robustness and efficiency, they suffer from some difficulties mainly related to the required mesh generation to discretize the domain. Specifically, constructing a high-quality grid for geometrically complex domains as well as analyzing solid-fluid moving interfaces are very challenging for the *grid-based* methods both in terms of computational time and mathematical complexity. Several procedures have been developed to alleviate these difficulties such as the use of unstructured grids or of techniques like the Immersed Boundary Method (Mittal and Iaccarino, 2005).

On the other hand, to overcome the issues of the *grid-based* methods in the last decade a strong interest has been focused on the *mesh-less* methods, among which *Smoothed Particle Hydrodynamics* (*SPH*) is probably one of the most used. These methods, not requiring a spatial mesh, are specifically suitable to study highly complex geometries, multi-phase flows, moving boundaries and rapidly evolving air-water interfaces. Although *mesh-less* methods share some common features, the employed approximation techniques are very different as discussed in Liu and Liu (2003).

1.3.1 Numerical simulations of cerebral aneurysms

CFD analysis, thanks to progress in computer equipment and medical imaging field, can be used nowadays to study hemodynamics in realistic patient aneurysm geometries. Several studies have shown that *CFD* can be a powerful tool to investigate aneurysm hemodynamics (Jou et al., 2003; Steinman et al., 2003; Cebral et al., 2005; Shojima et al., 2005; Castro et al., 2006; Boussel et al., 2008; Szikora et al., 2008; Karmonik et al., 2009; Shimogonya et al., 2009; Karmonik et al., 2010; Marzo et al., 2011; Passerini et al., 2012; Geers et al., 2017), analyzing the complex nature of blood flow inside the aneurysm sac as well as the interaction with *endovascular* devices (Ohta et al., 2005; Meng et al., 2006; Janiga et al., 2015; Shamloo et al., 2017; Zhang et al., 2017).

Numerical investigation on *CA* is commonly undertaken using the traditional *grid-based* numerical methods. Due to the very irregular and winding geometry of the aneurysm and the surrounding vasculature, the meshing process is not trivial and specific unstructured grids must be used. Moreover, when modeling *endovascular* treatments the large difference of scale between the size of the devices (*coils* or *flow-diverter*), the parent vessel and the aneurysm sac must be addressed (Jeong and Rhee, 2012). Blood flow through the structure of *endovascular* devices is difficult to simulate; thereby, simplified representations of device geometries and deployment are frequently employed. To overcome this issue some adaptive embedding techniques have been developed (Appanaboyina et al., 2008; Cebral and Lohner, 2005). A different strategy was proposed by Augsburger et al. (2011) based on the modeling of the device as a porous medium. In this framework, an international study, named *Virtual Intracranial Stenting Challenge (VISC) 2007*, was conducted with the purpose of establishing the reproducibility of state-of-the-art hemodynamic simulation and grid generation techniques in *subject-specific* stented aneurysm models (Radaelli et al., 2008).

Differently from the traditional and widely used *grid-based* formulation, in this research study the Lagrangian *mesh-less SPH* method is employed to model blood flow in cerebral aneurysms. This methodology fits naturally for the treatment of geometrically complex and irregular domains, such as cerebral vessels, without requiring the generation of a mesh.

Although the hemodynamic simulations presented within this manuscript (see Chap. 6) could be performed, in principle, with an unstructured grid approach as well, such irregular geometries are intrinsically hard to be handled for *grid-based* methods (Chui and Heng, 2010).

In this thesis no vessel walls deformation neither multi-phase processes, whose treatment would make *SPH* a very competitive strategy than an Eulerian method, have been considered. It is necessary to highlight that this research study is a step towards a more widely and complex modeling of cerebral aneurysms, where of a whole life cycle of these diseases should be considered, which encompassing their birth, progressively growth and rupture at the end. In this context, where the vessel deformations should be taken into account, the use of a *mesh-less* method like *SPH* is extremely beneficial, as it helps to straightforwardly model moving boundaries (Liu and Liu, 2010). Therefore, this research opens the door for future applications of *SPH* aimed at modeling cerebral vessels with moving walls, where the traditional *grid-based* methods would have strong difficulties since the process of generation of unstructured grids would become very time consuming since the mesh should be updated at each time step.

Moreover, *SPH* could be effectively used for predicting the effect of interventional procedures since it is very appropriate for modeling blood clot formation occurring inside the aneurysm sac after inserting *endovascular* devices. In this regards, blood clotting

process is governed by activation of platelets through biochemical and mechanical mechanisms which are strongly related to the trajectory of each particle and are thus usually analyzed through Lagrangian particle tracking procedures (Hansen et al., 2015; Shadden and Arzani, 2015; Alemu and Bluestein, 2007). Secondly, the thrombus formation occurs as a consequence of adhesion and deposition of activated platelets. In this framework, the use of a truly Lagrangian method is very suitable since the history of each platelet can be tracked and, moreover, a integration of multiple types of media (flow-related clot aggregation-dissolution) can be studied in detail.

In this compound, another important *CFD* application is the modeling of a virtual contrast transport, the so-called "*virtual angiogram*" (Ford et al., 2005; Vali et al., 2017), which can be used to analyze *endovascular* treatment outcomes. On one hand, the analysis of injection of a virtual medical agent can be very helpful to model an anti-aggregation therapy in order to prevent *thromboembolism* downstream the aneurysm site. On the other hand, through the virtual contrast modeling it is possible to evaluate the *residence time*, which defines the so-called "*blood age*". In other words, *residence time* indicates the mean time that a blood particle spent when passes through a region of the vessel. *Residence time* is, in fact, a key factor for identification of intra-aneurysmal regions of stagnancy which are associated with an increased likelihood for thrombus formation due to the insertion of an endovascular device.

As a merit of Lagrangian nature of the *SPH* method, the model for platelet activation mechanisms, as well as *residence time* parameter and tracer agent injection can be elegantly formulated considering the history of each particle (platelet, blood and drug particle, respectively). Specifically, through *SPH* it is possible to quantify the exposure of particles to some influence as they move through the domain, such as the level of shear stress or the chemical exposure.

In this research study, a stress-exposure time model has been implemented in *SPH* in order to simulate the process of mechanical platelet activation (see Chap. 7). Moreover, the *residence time* parameter and the transport of a virtual contrast agent have been analyzed employing a tracer transport model. Future research activities will be addressed to develop a *SPH-based* blood clot model encompassing both biochemical and mechanical stimuli for the platelet activation as well as a multi-phase process for studying their adhesion and deposition.

1.4 Smoothed particle hydrodynamics

SPH is a robust Lagrangian particle method for solving partial differential equations which was initially developed for astrophysical flow problems (Lucy, 1977; Gingold and Monaghan, 1977; Benz, 1988). The method was rapidly extended and applied to compressible and incompressible flows in different fields (Monaghan, 2012).

In the classical *SPH* approach the fluid domain is represented using a finite number of particles with a uniform distribution.

In *SPH* simulations of incompressible flows, the *weakly compressible* (*WCSPH*) and the *truly incompressible* (*ISPH*) approaches can be used to solve the Navier-Stokes equations. The original method was formulated as *WCSPH* (Monaghan, 1994) where the fluid flow is treated as slightly compressible relating the pressure to the fluid density through an appropriate equation of state. As discussed by Lee et al. (2008), the *WCSPH* scheme leads to some issues related to the development of acoustic waves traveling throughout the medium whose high speed imposes to use very low time steps in order

to respect the *Courant-Friedrichs-Levy stability condition* (*CFL*). Recently, Meringolo et al. (2017) proposed a filtering technique to remove the unphysical acoustic frequencies using a wavelet-based filtering. On the other hand, the *ISPH* approach enforces the fluid incompressibility solving the pressure field with a *Pressure Poisson Equation* (*PPE*) following the projection method proposed by Chorin (1968). The *ISPH* approach allows to make use of larger time steps since no explicit dependence on the speed of sound is enforced (Lind et al., 2012). However, the solution of the *PPE*, which is required to guarantee the mass conservation, leads to computational costs much higher than *WCSPH* for each time step. In this research study the *ISPH* approach has been employed. Therefore, from now on, all the numerical considerations will focus on the *ISPH* algorithm, although most of the implemented procedures (except those closely related to the *PPE*) could be applied with limited changes to the *WCSPH* scheme as well.

1.4.1 SPH limitations and improvements

Whilst *SPH* exhibits the best behavior just in those applications where *grid-based* methods show the most serious challenges (De Padova et al., 2013), the *mesh-less* method has some disadvantages mainly related to numerical instability, tedious treatment of *boundary conditions* (*BCs*) and high computational costs.

The motion of the particles, which are initially distributed regularly, can become unstable resulting in particle clustering along the streamlines due to the well-known problem of the *tensile instability* (Monaghan, 2000). This issue has been widely studied and several techniques have been developed to overcame it (Monaghan, 2000; Xu et al., 2009; Lind et al., 2012).

Modeling of wall *BCs* is *non-trivial* and requires some special treatments such as placement of *ghost* particles outside the computational domain. Moreover, peculiar difficulties are encountered when *open-boundaries* exist since the treatment of the particles leaving or entering the domain is very challenging. On one hand, the mass conservation constraint must be satisfied, on the other hand, a regular particle distribution at the inlet must be maintained in order to avoid any occurrence of void spaces. Recently, Monteleone et al. (2017) proposed an *inflow/outflow* procedure which meets both the above requirements. The procedure will be fully described in Chap. 3.

It is widely known that the *SPH* method requires huge computational efforts (Gomez-Gesteira et al., 2010) limiting its applications to close-up analysis of relatively small regions. This issue is mainly related to the employed in the "classical" *SPH* formalism of constant space particle distribution. In this case, indeed, the particle mutual distance is imposed by the regions requiring the highest accuracy and it is applied to the whole domain. Therefore, *SPH* can be computationally more expensive than the most employed Eulerian methods where, in order to reduce the computational costs, grids are usually non uniform. Employing the *SPH* method with a uniform particle distribution to real applications involves simulations requiring a high number of particles making frequently necessary the use of *high-performance computing* (*HPC*) (Crespo et al., 2015; Guo et al., 2018). In order to overcome this drawback, several refinement strategies have been proposed allowing to use a variable space particle distribution (Nelson and Papaloizou, 1994; Owen et al., 1998; Feldman and Bonet, 2007; López et al., 2013; Barcarolo et al., 2014; Shibata et al., 2017). Recently, Monteleone et al. (2018) proposed a *multi-domain* approach, entirely different from other existing *refinement* procedures in *SPH*. This method, which will be widely described in Chap. 4, allows to partition the computational domain in different subdomains in each of which a proper refinement can be used.

Differently from the aforementioned *refinement* procedures, there are also other strategies to reduce the *SPH* computational costs based on the merging of the *SPH* approach with the *mesh-based* methods. These strategies have been developed to simulate fluid-solid interactions (Jianming et al., 2010), wave processes (Narayanaswamy et al., 2010), ice dynamics (Nolin et al., 2009). Bouscasse et al. (2013b) developed an algorithm for coupling *SPH* (in *WCSPH* formulation) with other approaches through an interface region. Napoli et al. (2016) proposed a coupled method (named *Coupled FVM-SPH*) which allows to combine the larger computational efficiency of the finite-volume method with the flexibility of *SPH*. The method consists on partitioning of the computational domain in different regions in order to use the finite-volume method in some portions, while employing the *mesh-less* Lagrangian approach in the regions with the higher geometric complexity and/or the presence of moving boundaries or rapidly evolving free-surfaces. Although the *Coupled FVM-SPH* method has not been applied in the *CA* analysis, it will be discussed in Appendix A.

1.4.2 SPH applications in blood flow analysis

The use of the *SPH* method to simulate blood flow has significantly increased in last years.

Chui and Heng (2010) developed a rheological model for medical simulations with *SPH* including a flow-related clot aggregation-dissolution technique.

Shahriari et al. (2012) and Mao et al. (2016) used *SPH* to simulate blood flow passing through heart valves, whilst Caballero et al. (2017) simulated the blood flow dynamics in two realistic left ventricular models.

Other authors (Müller et al., 2004; Chong et al., 2017; Shi et al., 2017) proposed bleeding models in the human body with *SPH* in order to show how numerical techniques can provide an essential tool for future medical training such as virtual surgery.

To the author's best knowledge, no *SPH* applications exist to study *CA* hemodynamics.

1.5 Motivations and thesis outlines

1.5.1 Motivations and Objectives

Numerical simulations can provide useful information on aneurysm hemodynamics and can be used to clinically useful applications. On one hand, the numerical tools can make more realistic the surgical training systems. On the other hand, since for unruptured aneurysms the choice between immediate treatment or simple observation is strongly controversial, numerical simulations could improve the quality and timing of the treatment planning. For example *CFD* can be exploited as a criterion for *endovascular* devices design and placement as well as to study their interaction with the blood flow and vessels. Moreover, computational tools for simulating clinical interventions in *patient-specific* geometries may help to evaluate the impact of *endovascular* devices. Therefore, it is possible to identify the hemodynamic factors that affect treatment outcomes, thereby assisting neurosurgeons in choosing a favourable treatment plan.

The *mesh-less* feature of the *SPH* method can be exploited to model complex geometries such as cerebral vessels in presence of an aneurysm and *CAs* treated with *endovascular* devices. Moreover, the truly Lagrangian nature of *SPH* makes the method suitable to model the multi-phase process of blood clot formation inside the sac of aneurysms treated

with *thromboembolization* devices.

In this research the open-source *SPH* solver *PANORMUS-SPH* has been employed (see Methodological note). The numerical code is available at www.panormus3d.org.

The aim of this research study is to perform numerical analysis of blood flow in *CAs* by developing suitable algorithms and procedures in the *SPH* model to improve the performance of the *PANORMUS-SPH* code. The numerical improvements can be used to analyze the hemodynamics of *patient-specific CA* models, to investigate the transport of a tracer passively transported with the blood flow, to determine the *"blood age"* in each point of the domain (through the *residence time* parameter) and to study the activation of platelets through *mechanical stimuli*.

1.5.2 Thesis outlines

The thesis is organized into eight chapters and one appendix that are briefed as follows:

- *Chapter* 1 introduces background knowledge on *CAs*. The role of *computational fluid dynamics* is highlighted and the features of the *grid-based* and *mesh-less* numerical techniques are briefed. The *SPH* method is introduced describing some general features, advantages, limitations, improvements as well as applications in blood flow analysis. Motivations and objectives of the thesis are outlined;

- *Chapter* 2 provides fundamental and basic concepts of the *SPH* method. Several aspects of the *PANORMUS-SPH* code are explained such as the solid boundary treatment and the procedure to solve the momentum and continuity equations in the *ISPH* approach. Peculiar attention is paid to the *PPE* resolution method. The structure of the *SPH* numerical model in the basic version is described;

- *Chapter* 3 describes the developed procedure to treat *open-boundaries* in *SPH*. Two benchmark test cases are considered to show the efficiency and accuracy of the technique. This chapter is based on the paper of Monteleone et al. (2017);

- *Chapter* 4 describes the implemented refinement technique in the *SPH* method. The $3D$ transient *Poiseuille* flow in a circular pipe and the $2D$ vortex shedding in the wake are performed in order to show the performance of the developed method. This chapter is based on the paper of Monteleone et al. (2018);

- *Chapter* 5 proposes a *CPU*-based *parallel computing* procedure in *SPH* using the *MPI* libraries. Some numerical scalability tests are performed to show the performance of the parallelized *SPH* model;

- *Chapter* 6 investigates several numerical applications of *CA* hemodynamics considering an ideal geometry as well as real geometries selected from the *Aneurisk dataset repository* (Aneurisk-Team, 2012). Some numerical results are compared with finite-volume solutions and with experimental measures;

- *Chapter* 7 presents *tracer transport*, *residence time* and *mechanical platelets activation* models. Some benchmark test cases are shown. Ideal and real geometries of *CAs* are investigated;

- *Chapter* 8 draws main conclusions and suggests possible future research directions;

- *Appendix* A describes the *Coupled FVM-SPH* method. This appendix is based on the paper of Napoli et al. (2016). The appendix provides a procedure to speed up the solution of the original method of Napoli et al. (2016). A performance evaluation is conducted through the simulations of a $3D$ confined flow test (the lid-driven cavity problem) and a $2D$ free-surface application (the solitary wave *run-up* and *overtopping* on a seawall).

Chapter 2

Smoothed particle hydrodynamics and numerical procedure

In this chapter the basic concept and formulation of the *smoothed particle hydrodynamics* (*SPH*) method are discussed. The governing equations are written in the *SPH* formulation following the *ISPH* approach. The attention is focused on the numerical procedures employed in the *PANORMUS-SPH* code. Specifically, the boundary treatment at the solid walls, the *fractional-step* method to make the numerical solution marching in time, some stabilizing techniques and the adaptive time step procedure are explained. A peculiar attention is payed to the resolution of the elliptic *Pressure Poisson Equation* with the *BiConjugate Gradient STABilized* (*BiCGSTAB*) method and to the preconditioning algorithm to speed-up the convergence of the *BiCGSTAB*. Finally, a flow chart outlines the structure of the *PANORMUS-SPH* code considering its basic and serial version.

2.1 SPH basic idea and formulation

In the *SPH* method, the fluid domain is represented using a set of particles which move according to the Navier-Stokes equations. In the classical *SPH* approach the particles are initially distributed uniformly in space with a *starting particle distance* Δx.

Each i particle has its own mass m_i, density ρ_i, volume $\Delta V_i = m_i/\rho_i$ and other physical properties. The field variables at each particle are obtained using discrete convolution integrals with filter functions of assigned shape, indicated as *kernel functions*.

In the *SPH* formulation the *kernel approximation* and *particle approximation* can be conceptually identified (Liu and Liu, 2010).

2.1.1 Kernel approximation

The *kernel approximation* consists in representing a generic function f at position vector $\mathbf{x}$ in continuous form as a convolution integral extended to the domain D

$$\langle f(\mathbf{x})\rangle = \int_D f\left(\mathbf{x}'\right)\delta(\mathbf{x}-\mathbf{x}')\,d\mathbf{x}' \tag{2.1}$$

where δ is the *Dirac delta* function equal to 1 if $\mathbf{x}=\mathbf{x}'$ and 0 elsewhere.

Replacing the *Dirac delta* function with a weight function $W(\mathbf{x}-\mathbf{x}',h)$, the *kernel*

approximation of f, indicated with the symbol $\langle f(\mathbf{x})\rangle$, can be written as

$$\langle f(\mathbf{x})\rangle = \int_D f\left(\mathbf{x}'\right) W(\mathbf{x}-\mathbf{x}', h)\, d\mathbf{x}' \tag{2.2}$$

In the *SPH* approximation the weight function is known as *kernel function* and h is its characteristic length named *smoothing length*. The *smoothing length* h, which plays a role corresponding to the cell dimension in *grid-based* methods, controls the influence domain of the *kernel function* W defining its *support domain*.

Applying the Taylor series expansion to eqn. 2.2 and using the properties of the *kernel function*, defined in Sec. 2.2.1, it can be obtain the *kernel approximation* order accuracy

$$\begin{aligned}\langle f(\mathbf{x})\rangle &= \int_D \left[f(\mathbf{x}) + f'(\mathbf{x})(\mathbf{x}'-\mathbf{x}) + O((\mathbf{x}'-\mathbf{x})^2)\right] W(\mathbf{x}-\mathbf{x}', h)\mathbf{dx}' = \\ &= f(\mathbf{x})\int_D W(\mathbf{x}-\mathbf{x}', h)\mathbf{dx}' + f'(\mathbf{x})\int_D (\mathbf{x}'-\mathbf{x}) W(\mathbf{x}-\mathbf{x}', h)\mathbf{dx}' + \\ &+ O(h^2)\end{aligned}$$

where it is $\int_D W(\mathbf{x}-\mathbf{x}', h)\mathbf{dx}' = 1$ (see *unity condition* property of the *kernel function* in Sec. 2.2.1), whilst $\int_D (\mathbf{x}'-\mathbf{x}) W(\mathbf{x}-\mathbf{x}', h)\mathbf{dx}' = 0$ since the *kernel function* is even with respect to $\mathbf{x}$ (see the *symmetric condition* in Sec. 2.2.1). Thus, it can be stated that the *kernel approximation* has second order accuracy with respect to h (Monaghan, 1992) as shown by the following equation.

$$\langle f(\mathbf{x})\rangle = f(\mathbf{x}) + O(h^2)$$

The derivatives of the generic function can be obtained replacing $f(\mathbf{x})$ with $\frac{\partial f(\mathbf{x})}{\partial x}$ in eqn. 2.2 and applying integration by parts

$$\begin{aligned}\left\langle \frac{\partial f(\mathbf{x})}{\partial x} \right\rangle &= \int_D \frac{\partial f(\mathbf{x}')}{\partial x} W(\mathbf{x}-\mathbf{x}', h)\mathbf{dx}' = \\ &= \int_D \frac{\partial \left[f(\mathbf{x}') W(\mathbf{x}-\mathbf{x}', h)\right]}{\partial x}\mathbf{dx}' - \int_D f(\mathbf{x}')\frac{\partial W(\mathbf{x}-\mathbf{x}', h)}{\partial x}\mathbf{dx}'\end{aligned} \tag{2.3}$$

In eqn. 2.3 the term $\int_D \frac{\partial[f(\mathbf{x}')W(\mathbf{x}-\mathbf{x}', h)]}{\partial x}\mathbf{dx}'$ can be transformed into a surface integral using the *divergence theorem*

$$\int_D \frac{\partial \left[f(\mathbf{x}') W(\mathbf{x}-\mathbf{x}', h)\right]}{\partial x}\mathbf{dx}' = \int_S f(\mathbf{x}') W(\mathbf{x}-\mathbf{x}', h) n_x dS$$

where nx is the x component of the unit vector normal to the surface S. Due to the *compact condition* property, defined by Liu and Liu (2003), the *kernel function* W has null values in the surface S (see Sec. 2.2.1), thus the surface integral is zero. Therefore, eqn. 2.3 can be rewritten as

$$\left\langle \frac{\partial f(\mathbf{x})}{\partial x} \right\rangle = -\int_D f(\mathbf{x}')\frac{\partial W(\mathbf{x}-\mathbf{x}', h)}{\partial x}\mathbf{dx}' \tag{2.4}$$

where the derivatives of f can be obtained through the derivatives of W.

Similarly to what observed for the *kernel approximation* of a function, it is possible to show that the *kernel approximation* of the derivative is also of second order accuracy when the *support domain* is contained inside the problem domain (Liu and Liu, 2010).

The *kernel approximation* of higher order derivatives can be obtained substituting $f(\mathbf{x})$ with the corresponding derivatives in eqn. 2.2, using integration by parts, divergence theorem and some algebra.

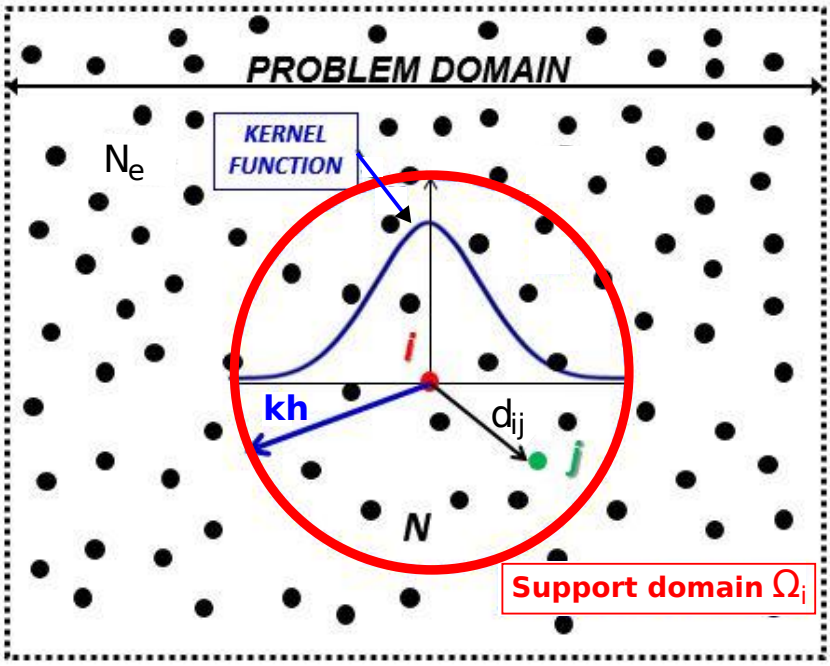


Figure 2.1: Sketch of the particle *support domain*. Continuous blue line: *kernel function*; red full cycle: particle i; empty red circle: *support domain* of i (Ω_i).

2.1.2 Particle approximation

The *particle approximation* involves representing the problem domain using N_e particles in each of which the field variables are estimated. As shown in Fig. 2.1, each particle i has a *support domain* Ω_i which includes all the surrounding particles having distance from the position of i ($\mathbf{x}_i$) lower than the product between the characteristic width h of the *kernel function* and a constant k depending on the shape of the *kernel function* (see Sec. 2.2.2).

The integral of eqn. 2.2 computed at the position $\mathbf{x}_i$ of the i particle can be replaced in a discretized form as the sum over the N particles lying into Ω_i

$$\langle f_i \rangle = \sum_{j=1}^{N} \frac{m_j}{\rho_j} f_j W_{ij} \tag{2.5}$$

where for simplicity $\langle f_i \rangle$ indicates $\langle f(\mathbf{x}_i) \rangle$ and the weighting function $W(\mathbf{x}_i - \mathbf{x}_j, h)$ is written as W_{ij} since its value depends on the distance d_{ij} between the particle i and its surrounding particle j.

As discussed by Colin et al. (2006), three different formulas can be used to compute the first derivatives of f at i particle, known as *Basic* ($BGAF$), *Difference*($DGAF$) and *Symmetric* ($SGAF$) *Gradient Approximation* (eqns. 2.6, 2.8, 2.7, respectively)

$$\left.\frac{\partial f}{\partial x}\right|_i = -\sum_{j=1}^{N} \frac{m_j}{\rho_j} f_j \left.\frac{\partial W_{ij}}{\partial x}\right|_i \tag{2.6}$$

$$\left.\frac{\partial f}{\partial x}\right|_i = -\sum_{j=1}^{N} \frac{m_j}{\rho_j} (f_i - f_j) \left.\frac{\partial W_{ij}}{\partial x}\right|_i \tag{2.7}$$

$$\left.\frac{\partial f}{\partial x}\right|_i = \rho_i \sum_{j=1}^{N} m_j \left(\frac{f_j}{\rho_j^2} + \frac{f_i}{\rho_i^2}\right) \left.\frac{\partial W_{ij}}{\partial x}\right|_i \tag{2.8}$$

where $\left.\frac{\partial f}{\partial x}\right|_i = \left\langle \frac{\partial f(\mathbf{x}_i)}{\partial x} \right\rangle$.

In this research study the *DGAF* formula has been used.

The second derivatives could be obtained applying one of the formulae above (2.6 - 2.8) to the first derivative but time-consuming double sum and additional smoothing would be required. To avoid this the formula proposed by Morris et al. (1997) is used which is based on Taylor series expansion together with the *DGAF* first derivative

$$\left.\frac{\partial^2 f}{\partial x_\alpha \partial x_\alpha}\right|_i = \sum_{j=1}^{N} 2\frac{m_j}{\rho_j} \frac{\partial W_{ij}}{\partial d_{ij}} \frac{1}{d_{ij}} (f_i - f_j) \tag{2.9}$$

Eqn. 2.9 can be expressed for the Laplacian as follows

$$\nabla^2 f_i = \sum_{j=1}^{N} 2\frac{m_j}{\rho_j} \frac{(\mathbf{x}_i - \mathbf{x}_j) \cdot \nabla W_{ij}}{d_{ij}^2 + \eta} (f_i - f_j) \tag{2.10}$$

where the symbol "$\cdot$" indicates the scalar product, ∇W_{ij} is the gradient of the *kernel function* and η is a small distance (set to $0.01\,h$ in all the applications presented in this thesis) used to avoid occurrence of singularities when the distance d_{ij} approximates to zero.

2.2 The Kernel function

2.2.1 Properties

The *kernel functions* should satisfy several conditions (Liu and Liu, 2003), the main ones are listed below:

- *Unity condition.* The integration of W over the whole domain D is equal to one;

$$\int_D W(\mathbf{x} - \mathbf{x}', h)\, d\mathbf{x}' = 1$$

- *Delta function behavior.* When h tends to zero the *kernel function* W approaches the *Dirac* function;

$$\lim_{h \to 0} W(\mathbf{x} - \mathbf{x}', h) = \delta(\mathbf{x} - \mathbf{x}')$$

- *Compact support condition.* The area of influence of W, named *support domain*, has limited width equal to kh. It is thus:

$$W(\mathbf{x} - \mathbf{x}', h) = 0, \text{ when } \left|\mathbf{x} - \mathbf{x}'\right| > kh$$

- *Positivity condition.* The function W is positive for each particle inside the *support domain* at $\mathbf{x}$:

$$W(\mathbf{x} - \mathbf{x}', h) > 0 \text{ when } \left|\mathbf{x} - \mathbf{x}'\right| \leq kh$$

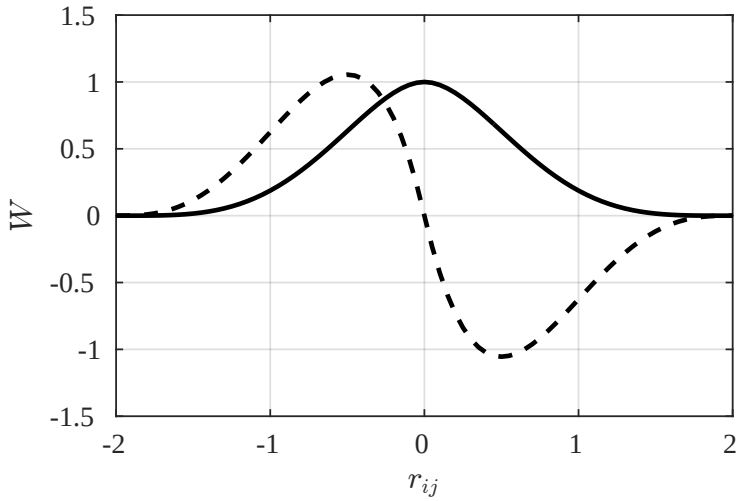


Figure 2.2: *Wendland* function (continuous black line) and its first derivative (dashed black line).

- *Monotonically decreasing behaviour.* The value of W is monotonically decreasing with the increase of the distance $|\mathbf{x} - \mathbf{x}'|$;
- *Symmetric condition.* W is an even function and particles with same distance but different positions have equal effect on a given particle:

$$\int_D (\mathbf{x} - \mathbf{x}')\, W(\mathbf{x} - \mathbf{x}', h)\, d\mathbf{x}' = 0$$

2.2.2 Wendland function

Several different functions have been proposed in the literature. In this research study, the *Wendland* function has been used (Wendland, 1995) where the proportionality constant k between the radius of the *support domain* and the *smoothing length* h is equal to 2

$$W_{ij} = \frac{21}{16\pi h^3} \begin{cases} (1 - r_{ij}/2)^4\, (2r_{ij} + 1)\,, & \text{when } 0 \leq r_{ij} < 2 \\ \\ 0, & \text{when } r_{ij} \geq 2 \end{cases} \tag{2.11}$$

where r_{ij} is the ratio $|\mathbf{x}_i - \mathbf{x}_j|/h$. The *Wendland* function and its first derivative are plotted in Fig. 2.2.

2.3 Consistency

The concept of *SPH* consistency has been in-deep described by Liu and Liu (2010). In general, if a function approximation can exactly reproduce a polynomial of up to n-th order exactly, it has C^n it. The concept of *SPH* consistency can be subdivided in *kernel* and *particle approximation* consistency. The *SPH* method has up to C^1 *kernel* consistency for the interior regions. However, due to the *kernel* truncation, occurring at the boundary regions, the *unity condition* is not satisfied; therefore, the *SPH* not even reaches C^0 *kernel* consistency and suitable procedures must be used to restore consistency.

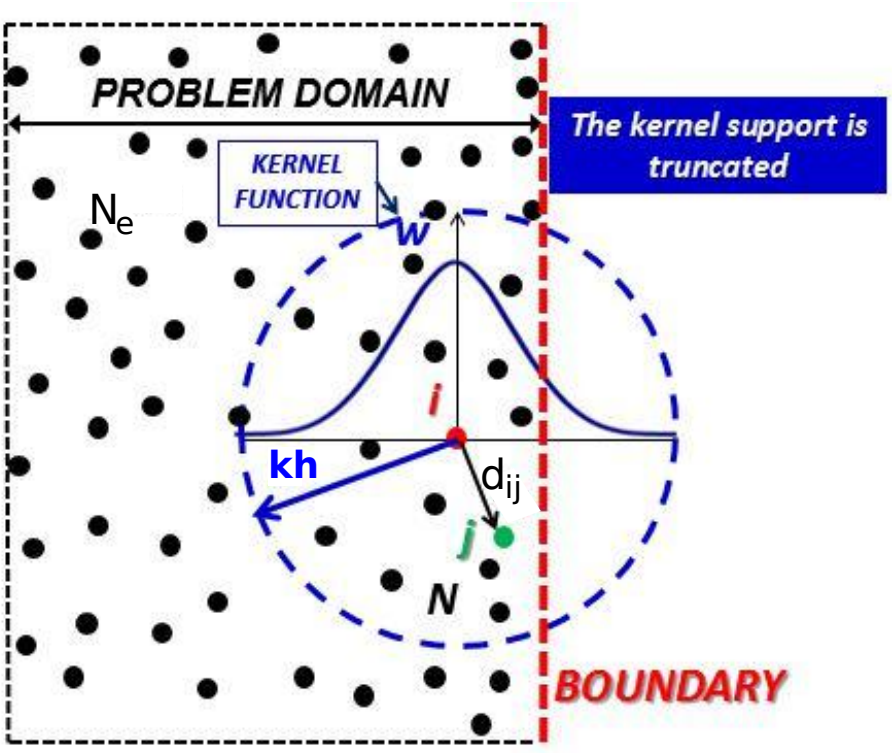


Figure 2.3: Sketch of the *kernel function* W (continuous blue line) truncated at the boundaries. Red circle: particle i; dotted blue line: *support domain* of i, (Ω_i); bold dotted red line: boundary.

The order of *kernel* consistency can be further reduced in the discretized *SPH* model due to the particle approximation process not only affecting the truncation of the *support domain* at the boundaries (as shown in Fig. 2.3) but also implying irregular particle distribution.

Moreover, the *particle approximation* process is strongly affected by two parameters: the *smoothing length* h, which characterizes the radius of interaction between particles, and the ratio of h to the *isotropic initial distance* Δx. A detailed analysis on the role of both h and $h/\Delta x$ parameters has been performed by Quinlan et al. (2006).

In *PANORMUS-SPH* the ratio $h/\Delta x$ has been set to 1 (in other words, $kh/\Delta x = 2$ considering the *Wendland* function where $k = 2$). The employed ratio ($h/\Delta x = 1$) is slightly lower than the most commonly used which ranges between 1.2 and 1.33 (Oger et al., 2007; Colagrossi and Landrini, 2003; Bouscasse et al., 2013a; Lind et al., 2012; Marrone et al., 2011; Skillen et al., 2013).

It should be noted that the ratio $h/\Delta x = 1$ leads to a quite limited number of neighbors, about 27 considering $3D$ approximations, and can involve inconsistencies when the *support domain* is truncated as discussed by Souto-Iglesias et al. (2013).

However, an higher ratio $h/\Delta x$, on one hand implies an higher smoothing level with a constant Δx and, on the other hand, requires the inclusion of more particles in each *support domain* while maintaining constant the *smoothing length*. These aspects are of particular importance at the boundaries since they affect the number of *mirror* particles to be generated in order to account for the truncation of the *support domain* at the solid walls, as it will be discussed in Sec. 2.5.1. Specifically, the number of *mirror* particles

increases with the ratio $h/\Delta x$.

Since in the *SPH* applications presented within this thesis the boundaries have a particularly considerable influence and the ratio between the domain surface and volume is very high, it is extremely important to limit as much as possible the number of *mirror* particles to be generated. Similar considerations can be done for the treatment of *open-boundaries* (see Chap. 3), *multi-domain interfaces* (see Chap. 4) and *parallel interfaces* (see Chap. 5) where an higher ratio $h/\Delta x$ would increase the number of *interface* particles and the amount of information to be shared between the processors, respectively. Anyhow, the comparison with some benchmark test cases (the available analytical solution in Secs. 3.3.1, 3.3.2, 4.3.1 and the literature data in Secs. 4.3.2, A2.1, A2.2) has shown that the employed ratio $h/\Delta x = 1$ allows to obtain good results when suitable correction procedures are employed (see Sec. 2.4)). A more in-deep analysis on the influence of this ratio on the results will be done in future work.

2.4 Correction of kernel and its gradient

As discussed in Sec. 2.3, the truncation of the *support domain* at the boundaries, the non-uniformly distribution of the particles and the limited number of neighbors considered can lead to some inconsistency.

Therefore, in the *PANORMUS-SPH* code some corrections of the *kernel function* and of its gradient are applied aimed at improving the consistency properties.

Eqn. 2.5 is thus normalized as follows

$$\langle f_i \rangle = \frac{\sum_{j=1}^{N} \frac{m_j}{\rho_j} f_j W_{ij}}{\sum_{j=1}^{N} \frac{m_j}{\rho_j} W_{ij}} \tag{2.12}$$

The derivatives of the *kernel function* (eqns. 2.6 - 2.8) must be corrected in the numerical model as well. Specifically, the procedure discussed by Oger et al. (2007) and Xu et al. (2009) is used to improve consistency of the *kernel* interpolation gradient. To this aim, the gradient of the *kernel function* ∇W_{ij} is replaced with the corrected gradient ∇W_{ij}^{C}

$$\nabla W_{ij}^{c} = \mathbf{C}^{-1} \nabla W_{ij}$$

where $\mathbf{C}$ is a (3 x 3) array whose generic element C is the derivative of the position function at the i particle. The array $\mathbf{C}$ can thus be expressed as

$$\mathbf{C} = \begin{bmatrix} \sum \frac{m_j}{\rho_j}(x_j - x_i)\frac{\partial W_{ij}}{\partial x_i} & \sum \frac{m_j}{\rho_j}(x_j - x_i)\frac{\partial W_{ij}}{\partial y_i} & \sum \frac{m_j}{\rho_j}(x_j - x_i)\frac{\partial W_{ij}}{\partial z_i} \\ \sum \frac{m_j}{\rho_j}(y_j - y_i)\frac{\partial W_{ij}}{\partial x_i} & \sum \frac{m_j}{\rho_j}(y_j - y_i)\frac{\partial W_{ij}}{\partial y_i} & \sum \frac{m_j}{\rho_j}(y_j - y_i)\frac{\partial W_{ij}}{\partial z_i} \\ \sum \frac{m_j}{\rho_j}(z_j - z_i)\frac{\partial W_{ij}}{\partial x_i} & \sum \frac{m_j}{\rho_j}(z_j - z_i)\frac{\partial W_{ij}}{\partial y_i} & \sum \frac{m_j}{\rho_j}(z_j - z_i)\frac{\partial W_{ij}}{\partial z_i} \end{bmatrix}$$

where the summation symbol $\sum_{j=1}^{N}$ is simply indicated with $\sum$.

The correction is used when calculating the diffusive term in the *predictor-step* equation (eqn. 2.20), the divergence of the velocity in eqn. 2.21 and the pressure gradient in eqn. 2.24.

2.5 Boundary condition treatment

The boundary treatment in the *SPH* method is a very challenging task. On one hand, appropriate conditions must be imposed at solid walls and at inflow and outflow bound-

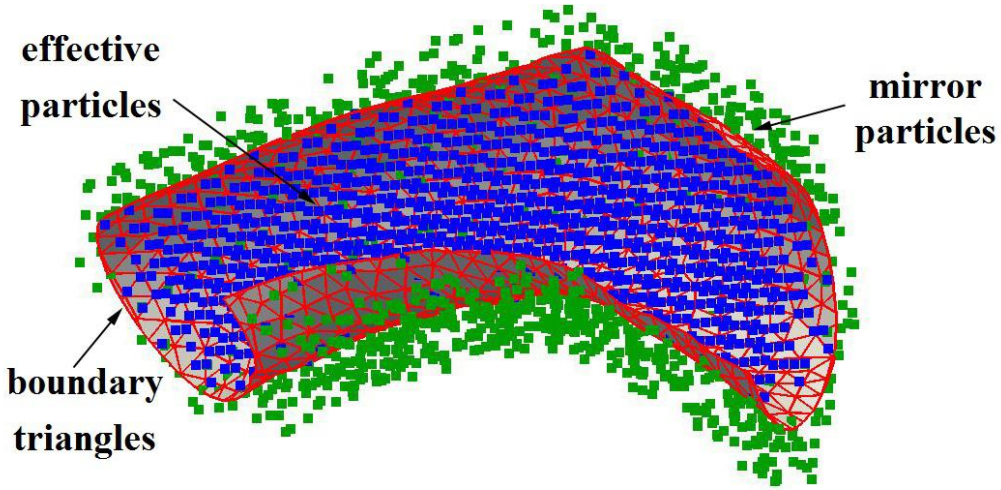


Figure 2.4: Sketch of the *mirror* particles (green squares), *effective* particle (blue squares) and boundary triangles (red lines) in a section of a cerebral blood vessel. Taken from: Napoli et al. (2016), 677, fig. 1.

aries. On the other hand, the *kernel function* truncation occurring at particles near the boundaries (see Fig. 2.3) must be overcome.

In *SPH*, several strategies have been developed to treat boundary conditions. The traditional approaches are based on the introduction of *mirror* particles (Morris et al., 1997; Takeda et al., 1994; Colagrossi and Landrini, 2003) or fixed *dummy* particles (Lee et al., 2008; Marrone et al., 2011; Bouscasse et al., 2013a; Chow et al., 2018).

In *PANORMUS-SPH* *mirror* particles are employed at solid walls, while a peculiar procedure is used at *inflow/outflow* boundary using *ghost* particles (this procedure will discussed in Chap. 3). It should be noted that in this thesis, the term "*effective* particle" is used to indicate the particles representing the computational domain (as explained in Sec. 2.1.2), whose total number is N_e; on the other hand, term "*mirror* particle" refers to the particles generated at solid walls to impose the *boundary conditions* (*BCs*), whose total number is N_{mirror}. Moreover, other types of particles, that can be classified as *ghost*, will be introduced for specific procedure in the following chapters.

2.5.1 Mirror particle technique

The *effective* particles having distance from the boundaries shorter than kh generate *mirror* particles along the directions normal to the boundary. Each *mirror* particle has the same physical properties (such as mass, density, viscosity, etc..) of the generating *effective* particle, while the velocity is chosen to ensure the satisfaction of the required boundary conditions (*adherence*, *free-slip*, *Neumann*, *wall-law*, *periodic*, etc..).

To ease up the identification of the normal direction and the distance particle-wall, in the *PANORMUS-SPH* model the boundaries of the computational domain are discretized into triangles, following a procedure employed in *mesh-based* methods with reference to the immersed boundary approach (Roman et al., 2009). The triangle discretization allows to obtain suitable descriptions of complex shapes and, moreover, it is very easy to identify the normal directions since the triangles lie in planes. The Fig. 2.4 shows the boundary

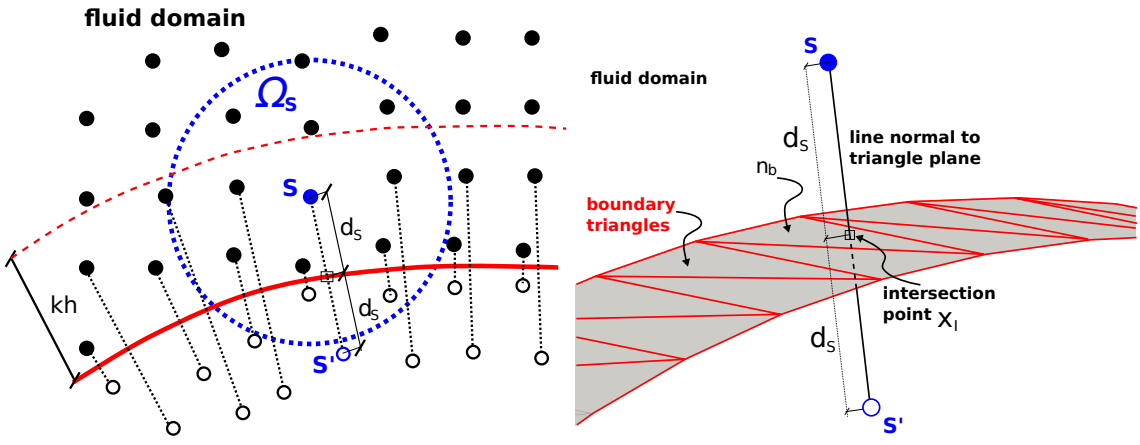


Figure 2.5: Sketch of the *mirror* particles generation. Full and empty circles: *effective* and *mirror* particles, respectively; bold red line: boundary. a) $2D$ scheme. Dotted line: *support domain* of S, Ω_S; b) $3D$ scheme. Gray area: boundary triangles surface.

triangles (red lines) of a cerebral vessel section. A boundary file must be created before carrying out the simulation. This file contains the coordinates of the nodes of the triangles followed by the triangle indices and the corresponding boundary conditions. At the beginning of the simulation the boundary file is read and the coefficients of the plane are calculated for each triangle. It should be noted that if the domain walls are fixed (that is one of the hypothesis of this research study as will by discussed in Sec. 6.1) the coefficients are calculated and stored only once, during the initialization step, since they do not change while the simulation runs.

In Fig. 2.5 it is shown how an *effective* particle S with distance from the boundary $d_S < kh$ generates the *mirror* particle S'. In the figure a $2D$ (where the boundary triangles are represented with segments) and $3D$ approximations (Figs. 2.5.a and 2.5.b, respectively) are plotted.

In the *PANORMUS-SPH* code, in order to speed up the identification of the *effective* particles to be mirrored, a virtual grid is created, made of cubic cells of side length equal to kh. The left-south-down corner of the grid ($\mathbf{x}_0$) and the number of cells in the three directions (nx, ny, nz) are calculated as follows

$$\begin{aligned} \mathbf{x}_0 &= \mathbf{x}_{min} - kh \\ nx &= \frac{x_{1,max} - x_{1,min}}{kh} + 2 \\ ny &= \frac{x_{2,max} - x_{2,min}}{kh} + 2 \\ nz &= \frac{x_{3,max} - x_{3,min}}{kh} + 2 \end{aligned} \tag{2.13}$$

where the subscripts max and min indicate the vertices of the boundary triangles having greater and lowest coordinates, respectively.

In the starting particle distribution, each cell of the virtual grid has a number of particle N_{cell} equal to

$$N_{cell} = \left(\frac{kh}{\Delta x}\right)^3, \quad \text{in 3D}$$

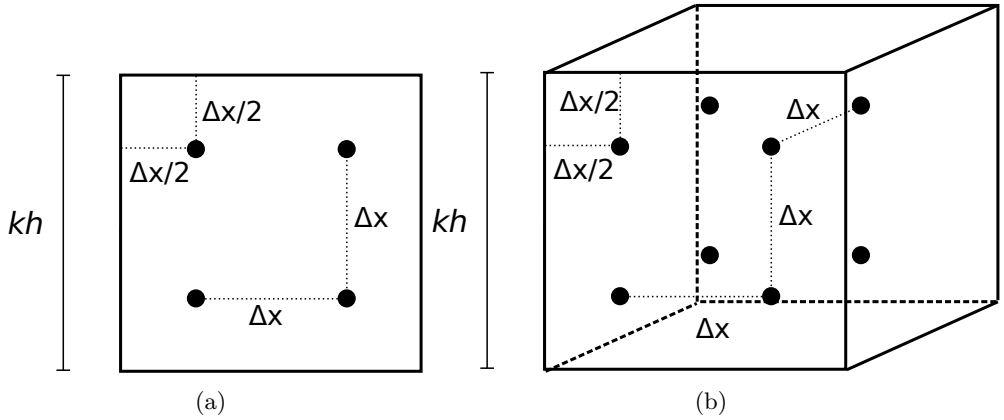


Figure 2.6: Sketch of the particle initial distribution inside a cell of the virtual grid. The ratio $h/\Delta x = 1$ is considered. a) $2D$ scheme; b) $3D$ scheme.

$$N_{cell} = \left(\frac{kh}{\Delta x}\right)^2, \quad \text{in 2D}$$

and thus, since it is $kh = 2\Delta x$, each cell has 8, and 4 particles in $3D$ and $2D$ approximations, respectively. The *effective* particles are regularly distributed inside the virtual cells (as shown in Fig. 2.6). The cell indices of each particle i are calculated, through eqn. 2.14, and stored for future use:

$$\mathbf{cell}_i = ceiling\left(\frac{\mathbf{x}_i - \mathbf{x}_0}{kh}\right) \tag{2.14}$$

where *ceiling* indicates the operation of rounding up.

The grid cells are thus classified in six different types:

- **Cell type** 1: cells contained inside the domain, with faces having distance from the boundaries $> kh$. These cells border only with cell of type 1 or 2 (to be defined below) and can only contain *effective* particles;
- **Cell type** 2: cells contained inside the domain with at least one lateral face with distance from the boundaries $< kh$. These cells can border with cell of type 1, 3 and 4 (to be defined below). They can contain only *effective* particles;
- **Cell type** 3: cells partially inside the domain and intersecting at least one of the boundaries. These cells intersect at least one boundary triangle. *Effective*, *mirror*, *IO* (defined in Chap. 3), *interface* (defined in Chap. 4), *parallel* (defined in Chap. 5) particles can be contained inside them;
- **Cell type** 4: cells entirely outside the domain. These cells can contain the same particles defined in the previous item with the exception of the *effective* ones;
- **Cell type** 5 **and** 6: These cells are used in the *parallel computing* model (see Chap. 5). They contain the particles to be shared with neighboring processors. The distinction between types 5 and 6 will be discussed in Chap 5. It should be noticed that these types are reported here only for completeness since they are not related to *mirror* technique.

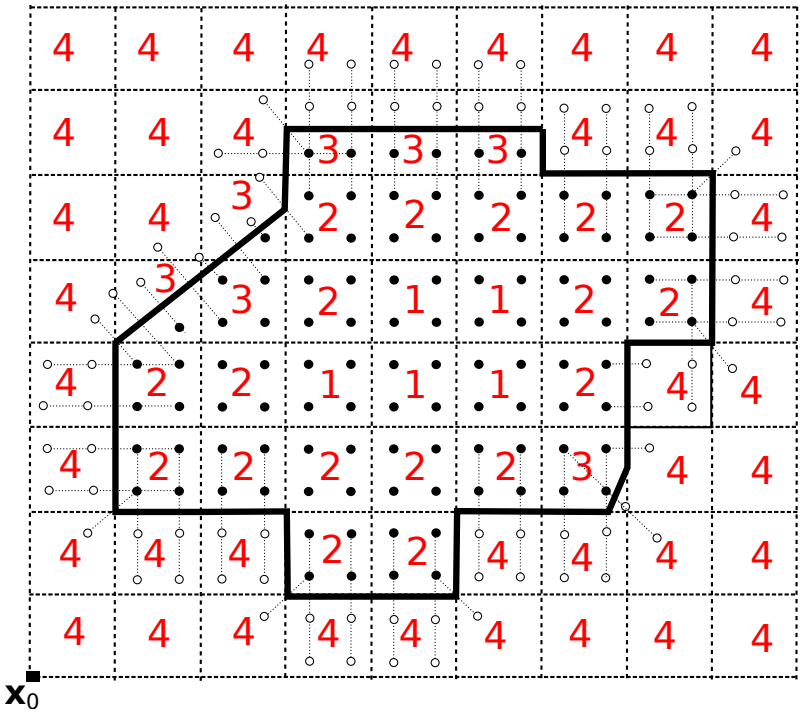


Figure 2.7: $2D$ Sketch of the virtual grid. Bold black line: boundary of the fluid domain. Full black circles: *effective* particles; empty black circles: *mirror* particles; dashed line: virtual grid (with $nx = 9$ and $ny = 8$). The cell type is indicated in red.

In order to classify the cells of the virtual grid, first the cells of type 3 are identified using the algorithm of Akenine-Möllser (2001) that checks the possible intersections between boxes and triangles in the $3D$ space, using the *separating axis* theorem. Then the internal (types 1 or 2) and external (type 4) cells are distinguished using a *ray-tracing* procedure proposed by Roman et al. (2009). Among the internal cells, the ones neighboring to cells of type 3 are first identified (type 2). For fixed boundaries, the procedure is performed only once, at the beginning of the simulation, since the cell type do not change during the simulation run. For each virtual cell, some information are recorded: the type and the number and list of triangles intersected by the cell (or close to it). The Fig. 2.7 shows a $2D$ sketch of a computational domain with the virtual grid.

The virtual grid allows to quickly select the *effective* particles near the boundary, reducing the computational time related to the mirror generation. Thereby, only the *effective* particles inside cells of type 2 and 3 are considered while generating the *mirror* particles, since they could have distance $d < kh$ from the boundaries. Moreover, after having selected one *effective* particle to be mirrored and having identified its cell (eqn. 2.14), the triangles with distance from the particle less than kh must be identified. To this aim, only the triangles intersecting the cell of the particle or one of the neighboring ones are checked, avoiding to scan all the triangles of the domain.

The *mirror* particles are numbered after the *effective* particles: the first *mirror* particle has the index $N_e + 1$ while the last one has the index $N_e + N_{mirror}$. For each *mirror* particle, the generating particle is recorded for later use.

The *mirror* generation can be summarized as follows:

1. The domain boundaries are discretized into triangle and a virtual grid is created;
2. The cells of the virtual grid are classified in types 1, 2, 3 and 4 (and 5 and 6 for *parallel computation*);
3. Only the *effective* particles belonging to cells of type 2 and 3 are considered since they can have distance from the boundary shorter than kh;
4. The distance (d_S) between each particle S (identified at step 3, which lies thus in a cell of type 2 or 3) and each triangle intersected or close the cell of S ($\mathbf{cell}_S$) is calculated. To this aim, the intersection point x_I between the current triangle plane (named nb) and its normal line starting from S is identified (see Fig. 2.5.b). The variable d_S is thus obtained as the distance between S and the intersection point x_I;
5. If $d_S < kh$ and the intersection point $\mathbf{x}_I$ is inside the triangle surface, one mirror particle S' is generated with coordinate: $\mathbf{x}_{S'} = 2\,\mathbf{x}_I - \mathbf{x}_S$.

The *mirror* technique allows to impose different boundary conditions assigning to the *mirror* particles values coherent with the required condition as explained below.

Adherence boundary condition
In order to set the velocity at the intersection point $\mathbf{u}_I$ equal to that of the current triangle nb ($\mathbf{u}_{nb}$), the velocity of the mirror S' (generated through the *effective* particle S) is set equal to

$$2\mathbf{u}_{S'} = \mathbf{u}_I - \mathbf{u}_S$$

Therefore, for fixed solid walls (where $\mathbf{u}_{nb}$ is equal to zero), it is

$$\mathbf{u}_{S'} = -\mathbf{u}_S$$

Free-slip boundary condition
In this case, the wall-normal velocity component of S' is set equal to the opposite of that of S, corresponding to null normal velocity at the wall in the linear approximation

$$u_{nS'} = -u_{nS}$$

where n is the wall normal direction. The tangential velocity of the *mirror* particle is set equal to that of the generating *effective* particle; thereby the velocity derivative in the wall-normal direction is equal to zero

$$u_{S'} = u_{\tau S}$$

where τ is the wall tangential direction. The wall-normal velocity component of the *effective* particle u_{nS} is obtained projecting the velocity of S in the direction n of the line connecting S with S', while the tangential component $u_{\tau S}$ is calculated as the magnitude of the vector $\mathbf{u}_{\tau S} - \mathbf{n}\,u_{nS}$.

Neumann boundary condition
The velocity of S' is set equal to

$$\mathbf{u}_{S'} = \mathbf{u}_S - \partial\mathbf{u}/\partial n \cdot d_S$$

where d_S is the distance between S and S' and $\partial\mathbf{u}/\partial n$ is the assigned velocity derivative in the normal direction (pointing towards the fluid region);

Periodic boundary condition
For *periodic BCs* two identical parallel boundary triangles are placed normally to the direction of the periodic condition. The physical and hydrodynamic properties of the periodic *mirror* particles are identical to that the generating *effective* particle.

2.6 Governing equations and numerical procedure

The continuity and momentum equations for incompressible flows can be written as

$$\begin{aligned} &\nabla \cdot \mathbf{u} = 0 \\ &\frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} + \frac{1}{\rho}\nabla p - \nu\nabla^2\mathbf{u} - \mathbf{f}_m = 0 \end{aligned} \tag{2.15}$$

where $\mathbf{u}$ is the instantaneous velocity, $\mathbf{f}_m$ is the mass force per unit mass, p is the pressure, t is the time, ρ is the fluid density and ν is the kinematic viscosity. These equations are solved in the framework of *ISPH* (Lind et al., 2012) using a projection method (Chorin, 1968) and requiring the solution of a *Pressure Poisson Equation* (*PPE*) to enforce incompressibility. Specifically, in *PANORMUS-SPH* the *fractional-step* procedure is employed to solve the momentum and continuity equations. The procedure can be subdivided into: *predictor-step*, *PPE* and *corrector-step*.

In the *predictor-step* the *intermediate* velocity $\mathbf{u}^*$ is calculated removing the pressure gradient term from the momentum equation

$$\frac{\mathrm{D}\mathbf{u}^*}{\mathrm{D}t} - \nu\nabla^2\mathbf{u} - \mathbf{f}_m = 0 \tag{2.16}$$

In order to correct the $\mathbf{u}^*$ velocities while imposing the continuity constraint for incompressible flows, an irrotational corrective velocity field $\mathbf{u}_c$ is calculated. The potential $\psi\Delta t$ of $\mathbf{u}_c$ is calculated solving, through an iterative procedure, the *PPE*

$$\nabla^2\psi = -\frac{\nabla \cdot \mathbf{u}^*}{\Delta t} \tag{2.17}$$

where ψ is the *pseudo-pressure* (p/ρ) having the dimension of the kinematic pressure and $\nabla \cdot \mathbf{u}^*$ is the divergence of the *intermediate* velocity. The *PPE* must be solved with boundary conditions set as follow:

- At solid walls (*adherence*, *free-slip* or *wall_law*) the *Neumann* condition is used;
- At inflow or outflow boundaries the *pressure* condition must be set, resulting in a *Dirichlet* condition for the pressure.

The following discussion refers only to the *PPE* boundary condition at solid walls; the *PPE BCs* for inflow and outflow boundaries will be discussed in Chap. 3 where a detailed description of the implemented procedure can be found. The boundary conditions for the *PPE* at solid walls are obtained assigning to the *mirror* particles the *Neumann* condition

$$\frac{\partial\psi}{\partial n} = \frac{u_n^{k+1} - u_n^*}{\Delta t} \tag{2.18}$$

where n is the wall normal direction (pointing towards the interior of the domain) and u_n^{k+1} and u_n^* are the *corrected* and *intermediate* wall velocity projections in the n direction, respectively. Differently from Cummins and Rudman (1999), here a *non-homogeneous Neumann* condition is used, obtaining the *intermediate* velocity at the boundaries through suitable extrapolations.

In the *corrector-step* the divergence-free updated velocity field $\mathbf{u}^{k+1}$ (named *corrected* velocity) is obtained as the sum of the *intermediate* velocity $\mathbf{u}^*$ and the corrective velocity $\mathbf{u}_c$

$$\mathbf{u}^{k+1} = \mathbf{u}^* + \mathbf{u}_c = \mathbf{u}^* - \nabla\psi\Delta t \tag{2.19}$$

2.6.1 *Fractional-step* in the *ISPH* formulation

Considering the i-th particle, eqn. 2.16 can be rewritten as

$$\frac{\mathbf{u}_i^* - \mathbf{u}_i^k}{\Delta t} + \frac{3}{2}\mathbf{D}_i^k - \frac{1}{2}\mathbf{D}_i^{k-1} - \mathbf{f}_{m_i} = 0 \quad (\textbf{\textit{predictor-step}}) \tag{2.20}$$

where $\mathbf{u}_i^*$ is the *intermediate* velocity of the i particle, the apex k indicates the variables at the k-th time step, Δt is the time step, $\mathbf{u}_i^k$ is the velocity of the i particle at time k and $\mathbf{f}_{m_i}$ is the mass force per unit mass acting on the i particle. The diffusive term $\mathbf{D}_i$ of eqn. 2.20 is calculated using a second-order *Adams-Bashforth* scheme

$$\mathbf{D}_i = -\sum_{j=1}^{N} m_j(\nu_i + \nu_j)\frac{(\mathbf{x}_i - \mathbf{x}_j)\cdot\nabla W_{ij}^c}{d_{ij}^2 + \eta}(\mathbf{u}_i - \mathbf{u}_j)$$

The *PPE* (eqn. 2.17) in the *SPH* approximation becomes

$$\sum_{j=1}^{N} \quad 2 \quad \frac{m_j}{\rho_j}\frac{(\mathbf{x}_i - \mathbf{x}_j)\cdot\nabla W_{ij}^c}{d_{ij}^2}(\psi_i - \psi_j) = \tag{2.21}$$

$$= \quad -\frac{1}{\Delta t}\sum_{j=1}^{N}\frac{m_j}{\rho_j}\nabla W_{ij}^c \cdot (\mathbf{u}_i^* - \mathbf{u}_j^*) \quad (\textbf{\textit{Poisson}})$$

The *PPE BCs* at solid walls are imposed assigning to the *mirror* particle m generated from the i *effective* particle the value

$$\psi_m = \psi_i - \frac{(u_n^{k+1} - u_n^*)\big|_b}{\Delta t}\, d_{im} \tag{2.22}$$

where d_{im} is the distance between the particles i and m, while the difference $(u_n^{k+1} - u_n^*)\big|_b$ is calculated at the intersection point $\mathbf{x}_b$ between the line connecting the particles m and i and the boundary triangle surface. The normal component of the *corrected* velocity $\mathbf{u}^{k+1}$ is set equal to the wall-normal velocity (null for fixed boundaries) or to the imposed value for *Dirichlet* boundary conditions. The normal component of the *intermediate* velocity at the boundary u_n^* can be obtained through an extrapolation from the interior of the problem domain as discussed by Zang et al. (1994) in the framework of *mesh-based* methods. Specifically, the *intermediate* velocity at the boundary $\mathbf{u}^*(\mathbf{x}_b)$ is obtained through a Taylor series expansion around the closest *effective* particle (indicated with P) does not have *mirror* particles in the vicinity. Considering the $\alpha - th$ component (with $\alpha = 1, 2, 3$) the velocity at $\mathbf{x}_b$ can be expressed as

$$u_\alpha^*(\mathbf{x}_b) = u_{\alpha,P}^* + \left.\frac{\partial u_\alpha^*}{\partial x}\right|_P \cdot (\mathbf{x}_b - \mathbf{x}_P)$$

The term $\left.\frac{\partial u^*_\alpha}{\partial x}\right|_P$ in the previous equation is calculated using eqn. 2.7

$$\left.\frac{\partial u^*_\alpha}{\partial x}\right|_P = \sum_{j=1}^{N} \frac{m_j}{\rho_j}(u^*_{\alpha,j} - u^*_{\alpha,P})\nabla W_{Pj}$$

where the summation is extended to the N particles lying in Ω_P. Thus, the normal component can be easily obtained projecting $\mathbf{u}^*(\mathbf{x}_b)$ in the wall-normal direction $\mathbf{n}$

$$u^*_n(\mathbf{x}_b) = \mathbf{u}^*(\mathbf{x}_b) \cdot \mathbf{n} \tag{2.23}$$

Applying the *SPH* approximation to eqn. 2.19, the *corrector-step* equation can be written as

$$\mathbf{u}_i^{k+1} = \mathbf{u}_i^* + \Delta t \sum_{j=1}^{N} \frac{m_j}{\rho_j} \nabla W^c_{ij} (\psi_i - \psi_j) \quad (\textbf{\textit{corrector-step}}) \tag{2.24}$$

After calculating the updated velocity field, the particles are moved at the end of each time step. The updated position $\mathbf{x}_i^{k+1}$ can be obtained using the mean value of the new and old velocities ($\mathbf{u}_i^{k+1}$ and $\mathbf{u}_i^k$, respectively).

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \frac{\mathbf{u}_i^{k+1} + \mathbf{u}_i^k}{2}\Delta t \tag{2.25}$$

2.7 Solution methods for the *Poisson* system

The numerical solution of the elliptic *PPE* is one the key challenges of the *ISPH* algorithm. The *PPE* is the most time consuming step in the *ISPH* procedure, typically occupying more than 80% of the total computational time. Moreover, the *PPE* matrix system is sparse and non-symmetric. Comparing to *mesh-based* methods, the added complexity for solving the *PPE* in *SPH* is that the non-null coefficients of the sparse matrix change every time step due to the movement of the particles. In the previous version of the *PANORMUS-SPH* code (Napoli et al., 2015) the *PPE* was solved by using a semi-implicit *SOR* algorithm. In this research study, in order to reduce the high computational costs of the elliptic *Poisson* equation solution and to increase the accuracy of the numerical model, the iterative *BiConjugate Gradient STABilized* (*BiCGSTAB*) method, proposed by Van der Vorst (1992), has been implemented. Therefore, the numerical solution of a *Poisson* system made of the *PPE* of all the *effective* particles of the computational domain is performed using an implicit algorithm.

2.7.1 The BiCGSTAB method

In order to implicitly solve the system of eqns. 2.21 iterating among all the particles of the computational domain, in this research study the *BiCGSTAB* method has been implemented in the numerical model. The *BiCGSTAB* method is very suitable due to the non-symmetricity and *diagonal dominant* of the coefficient matrix of the *PPE* system.

The *PPE* is iteratively solved as a linear matrix system $\mathbf{A}\,\mathbf{x} = \mathbf{b}$. In the following, before explaining the *BiCGSTAB* method, the *PPE* system is analyzed identifying the matrix coefficients and the *right-hand-side* (*RHS*) term of the system made of N_e equations.

The *PPE* system

Considering the i-th particle, eqn. 2.21 can be rewritten in the following compact form

$$\sum_{j=1}^{N} C_{ij}(\psi_i - \psi_j) = T_i \tag{2.26}$$

where the coefficients C_{ij} and the known term T_i are expressed through eqns. 2.27 and 2.28, respectively.

$$C_{ij} = 2\frac{m_j}{\rho_j}\frac{(\mathbf{x}_i - \mathbf{x}_j)\cdot\nabla W_{ij}^c}{d_{ij}^2} \tag{2.27}$$

$$T_i = -\frac{1}{\Delta t}\sum_{j=1}^{N_i^*}\frac{m_j}{\rho_j}\nabla W_{ij}^c \cdot (\mathbf{u}_i^* - \mathbf{u}_j^*) \tag{2.28}$$

In eqn. 2.28 the *mirror* particles are excluded while calculating the divergence of the *intermediate* velocity $\mathbf{u}^*$, thereby N_i^* indicates the number of *effective* particles in Ω_i.

The i-th row *diagonal* term of the coefficient matrix is

$$\sum_{j=1}^{N_i^{'}} C_{ij} \tag{2.29}$$

where the summation is extended to the *effective* and *mirror* particles in Ω_i, excluding for the latter those generated by i ($N_i^{'}$). Indeed, including the *mirror* particles generated by i and substituting eqn. 2.22 in eqn. 2.26, the difference $\psi_i - \psi_j$ reduces to $(u_n^{k+1} - u_n^*)\, d_{ij}/\Delta t$, thus contributing only to the system *right-hand-side*.

The i-th row *off-diagonal* term in the s-th column of the system coefficient matrix is equal to

$$-\left(\delta_{is}C_{is} + \sum_{j=1}^{N_i^{Ms}} C_{ij}\right) \tag{2.30}$$

where $\delta_{is} = 1$ if the *effective* particle s lies in Ω_i and $\delta_{is} = 0$ elsewhere, while the summation is extended to the N_i^{Ms} *mirror* particles generated by the particle s and lying in Ω_i (with $N_i^{Ms} = 0$ if no *mirror* particle generated by s exists in Ω_i).

The Fig. 2.8 shows a scheme of the system coefficient matrix.

The i-th equation *right-hand-side* term can be obtained adding in eqn. 2.26 the boundary conditions (eqn. 2.22) for the all *mirror* particles in Ω_i

$$RHS_i = T_i + \frac{1}{\Delta t}\sum_{j=1}^{N_i^M} C_{ij}\,(u_n^{k+1} - u_n^*)\Big|_b\, d_{gj} \tag{2.31}$$

where the index g indicates the *effective* particles generating the N_i^M *mirror* particles lying in the *support domain* of i.

$$
\begin{array}{c}
 \\
1 \\
\vdots \\
i \\
\vdots \\
\vdots \\
s(\neq i) \\
\vdots \\
N_e
\end{array}
\begin{array}{c}
\begin{array}{ccccccc} 1 & \cdots & i & \cdots & s(\neq i) & \cdots & N_e \end{array} \\
\left[\begin{array}{ccccccc}
\sum_{j=1}^{N_1'} C_{1j} & & \vdots & & \vdots & & \\
 & \ddots & \vdots & & \vdots & & \\
\cdots & \cdots & \sum_{j=1}^{N_i'} C_{ij} & \cdots & -C_{is}\delta_{is} - \sum_{j=1}^{N_i^{Ms}} C_{ij} & \cdots & \cdots \\
 & & \vdots & & \vdots & & \\
 & & \vdots & \ddots & \vdots & & \\
\cdots & \cdots & -C_{si}\delta_{si} - \sum_{j=1}^{N_s^{Mi}} C_{sj} & \cdots & \sum_{j=1}^{N_s'} C_{sj} & \cdots & \cdots \\
 & & \vdots & & \vdots & \ddots & \\
 & & \vdots & & \vdots & & \sum_{j=1}^{N_{Ne}'} C_{N_e j}
\end{array}\right]
\end{array}
$$

Figure 2.8: Scheme of the coefficient matrix of the *PPE* system.

The coefficient matrix of the system is sparse since the *off-diagonal* terms (eqn. 2.30) of the i-th row are null for the columns corresponding to *effective* particles outside Ω_i ($\delta_{is} = 0$) and not generating any *mirror* particle lying in Ω_i ($N_i^{M_s} = 0$). Moreover, the matrix is non-symmetric due to the terms related to the *mirror* particles in the *off-diagonal* terms (eqn. 2.30). Moreover, the coefficient matrix of the *PPE* is *diagonal-dominant* since the *diagonal* terms (eqn. 2.29) of the current particle i contain the sum of the C_{ij} coefficients relative to all the particles lying in Ω_i and thus are equal to the sum of the *off-diagonal* terms (eqn. 2.30) in the same row.

The *Compressed Row Storage* (*CRS*) format (Bisseling, 2004) has been adopted to reduce memory requirements, to fasten row access and matrix-vector multiplications, storing only the non-null terms of the *PPE* matrix of coefficients. An example of the *CSR* format for a matrix [8 x 8] is shown in Fig. 2.9. Three 1-dimensional vectors are created to represent the matrix of the coefficients of the system:

- ***vals***. This array, of length nv, contains all the values of the non-zero elements of the matrix. For the matrix **A** in the figure it is $nv = 22$;

- ***cols***. This array has length nc equal to that of ***vals*** ($nc = nv$). It contains the column number of the corresponding elements in the array ***vals***;

- ***limits***. Each element in this vector is a pointer to the first non-zero element of each row in vectors ***vals*** and ***cols***. The vector *limits* has length nl equal to the number of rows plus one, where the last entry is used to store nv ($nl = 9$ for the matrix **A** in the figure). It starts from 1 ($limits(1) = 1$, where $limits(1)$ indicates the value at the first position of the vector ***limits***) and ends with the number of elements with non-null values +1 ($limits(nl) = nv + 1$, where $limits(nl)$ indicates the value at the last position of the vector ***limits***).

The *Unpreconditioned BiCGSTAB* method

In the following, the algorithm of the *BiCGSTAB* method in its unpreconditioned version is shown (Van der Vorst, 1992).

$$\mathbf{A} = \begin{bmatrix} \mathbf{9} & \mathbf{5} & 0 & 0 & \mathbf{4} & 0 & 0 & 0 \\ 0 & \mathbf{7} & \mathbf{3} & \mathbf{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{9} & 0 & \mathbf{7} & \mathbf{2} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{8} & 0 & 0 & \mathbf{8} & 0 \\ \mathbf{20} & 0 & 0 & 0 & \mathbf{25} & 0 & 0 & \mathbf{5} \\ \mathbf{3} & 0 & 0 & 0 & 0 & \mathbf{3} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 & 0 & \mathbf{6} & \mathbf{5} \\ 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{8} & 0 & \mathbf{9} \end{bmatrix}$$

$$\mathbf{vals} = \begin{bmatrix} 9 & 5 & 4 & 7 & 3 & 4 & 9 & 7 & 2 & 8 & 8 & 20 & 25 & 5 & 3 & 3 & 1 & 6 & 5 & 1 & 8 & 9 \end{bmatrix}$$

$$\mathbf{cols} = \begin{bmatrix} 1 & 2 & 5 & 2 & 3 & 4 & 3 & 5 & 6 & 4 & 7 & 1 & 5 & 8 & 1 & 6 & 2 & 7 & 8 & 3 & 6 & 8 \end{bmatrix}$$

$$\mathbf{limits} = \begin{bmatrix} 1 & 4 & 7 & 10 & 12 & 15 & 17 & 20 & 23 \end{bmatrix}$$

Figure 2.9: *CRS* format example for a sparse matrix **A**.

Specifically, the non-symmetric linear system $\mathbf{A}\,\mathbf{x} = \mathbf{b}$ is considered, where $\mathbf{A}$ is a [n x n] matrix (with n the number of equations system equal to the total number of *effective* particles N_e), while $\mathbf{x}$ and $\mathbf{b}$ are the vector solution and the vector of known terms, respectively, (whose lengths are equal to n).

ALGORITHM 2.1- BiCGSTAB method

1. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\,\mathbf{x}_0$
2. Choose $\mathbf{r}_0^*$ such that $(\mathbf{r}_0^*, \mathbf{r}_0) \neq 0$. For instance $\mathbf{r}_0^* = \mathbf{r}_0$;
3. $\rho_0 = \alpha_0 = \omega_0 = 1$
   The coefficients ρ, α and ω are set to one;
4. $\mathbf{v}_0 = \mathbf{p}_0 = \mathbf{0}$
   The vectors $\mathbf{v}$ and $\mathbf{p}$ are set to zero;
5. The iterative cycle is performed until convergence ($RSQ < tol$):

   5.0 *do* $j = 1, \ldots$ *until convergence*

   5.1. $\rho_j = (\mathbf{r}_0^*, \mathbf{r}_{j-1})$;

   5.2. $\beta_j = \left(\frac{\rho_j}{\rho_{j-1}};\right)\left(\frac{\alpha_{j-1}}{\omega_{j-1}}\right)$;

   5.3. $\mathbf{p}_j = \mathbf{r}_{j-1} + \beta\,(\rho_{j-1} - \omega_{j-1}\,\mathbf{v}_{j-1})$;

   5.4. $\mathbf{v}_j = \mathbf{A}\,\mathbf{p}_j$;

   5.5. $\alpha_j = \dfrac{\rho_j}{(\mathbf{r}_0^*, \mathbf{v}_j)}$;

5.6. $\mathbf{s} = \mathbf{r}_{j-1} - \alpha_j \mathbf{v}_j$;

5.7. $\mathbf{t} = \mathbf{A}\,\mathbf{s}$;

5.8. $\omega_j = \dfrac{(\mathbf{t}, \mathbf{s})}{(\mathbf{t}, \mathbf{t})}$;

5.9. $\mathbf{x}_j = \mathbf{x}_{j-1} + \alpha_j\,\mathbf{p}_j + \omega_j\,\mathbf{s}$;

5.10. $\mathbf{res} = \mathbf{b} - \mathbf{A}\,\mathbf{x}_j$;

5.11. $RSQ = (\mathbf{res}, \mathbf{res})$

5.12. *Check if convergence is reached:*

5.12.1 *if* $(RSQ < tol)$ *then quit.*

5.12.2 *else* $\mathbf{r}_j = \mathbf{s} - \omega_j\,\mathbf{t}$ *and the cycle continues.*

---

where $\mathbf{x}_0$ at item 1 is an initial solution vector (equal to zero for the first time step of the simulation or to the ψ values of the previous time step), ρ, α, ω are scalar coefficients while $\mathbf{r}$, $\mathbf{r}_0^*$, $\mathbf{x}_0$, $\mathbf{v}$, $\mathbf{p}$, $\mathbf{s}$, $\mathbf{t}$ are vectors of length n. The iterative cycle at point 5.0 is performed until the sum of the squared errors (RSQ) becomes lower than an imposed tolerance value (tol).

The matrix-vector multiplications (items 1, 5.4, 5.7 and 5.10 in the algorithm above) in the CRS format can be made using the following algorithm that is written considering the product $\mathbf{r} = \mathbf{A}\,\mathbf{x}$.

ALGORITHM 2.2- ASUB

$$
\begin{array}{lll}
do & j & = 1, .., Ne \\
 & r(j) & = 0 \\
 & do & limits(j), limits(j+1) - 1 \\
 & & r(j) = r(j) + vals(j)\, x(cols(j))
\end{array}
$$

---

where again N_e is the number of equations in the system (equal to the number of *effective* particles) and **vals**, **cols** and **limits** are the vectors in the CRS format as explained above. The dimension of the vectors **vals** and **cols** is equal to the total number of non-null elements, while the dimension of the vector **limits** is equal to the total number of equations plus one $(N_e + 1)$.

The *Preconditioned BiCGSTAB* method

Although the algorithmic efficiency is excellent, the *BiCGSTAB* method is numerically unstable. In order to make solvers converge faster a preconditioning algorithm has been implemented (Saad, 2003).

The preconditioning modifies the spectrum of the coefficient matrix to speed-up the convergence of the iterative method.

In the *Preconditioned BiCGSTAB* (*Pre-BiCGSTAB*) algorithm the vectors $\mathbf{y}$ and $\mathbf{z}$ (with dimension $n = N_e$) are introduced. The *Pre-BiCGSTAB* algorithm is shown below.

ALGORITHM 2.3- Pre-BiCGSTAB method

1. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\,\mathbf{x}_0$
2. Choose $\mathbf{r}_0^*$ such that $(\mathbf{r}_0^*, \mathbf{r}_0) \neq 0$. For instance $\mathbf{r}_0^* = \mathbf{r}_0$;
3. $\rho_0 = \alpha_0 = \omega_0 = 1$
4. $\mathbf{v}_0 = \mathbf{p}_0 = \mathbf{0}$
5. The iterative cycle is performed until convergence ($RSQ < tol$):

5.0 *do* $j = 1, ...$ *until convergence*
5.1. $\rho_j = (\mathbf{r}_0^*, \mathbf{r}_{j-1})$;
5.2. $\beta_j = \left(\frac{\rho_j}{\rho_{j-1}};\right)\left(\frac{\alpha_{j-1}}{\omega_{j-1}}\right)$;
5.3. $\mathbf{p}_j = \mathbf{r}_{j-1} + \beta\,(\rho_{j-1} - \omega_{j-1}\,\mathbf{v}_{j-1})$;
5.4. $\mathbf{y} = \mathbf{K}^{-1}\mathbf{p}_j$ (*solving* $\mathbf{y}' = \mathbf{L}^{-1}\mathbf{p}_j$ *and* $\mathbf{y} = \mathbf{U}^{-1}\mathbf{y}'$)
5.5. $\mathbf{v}_j = \mathbf{A}\,\mathbf{y}$;
5.6. $\alpha_j = \frac{\rho_j}{(\mathbf{r}_0^*, \mathbf{v}_j)}$;
5.7. $\mathbf{s} = \mathbf{r}_{j-1} - \alpha_j \mathbf{v}_j$;
5.8. $\mathbf{z} = \mathbf{K}^{-1}\mathbf{s}$ (*solving* $\mathbf{z}' = \mathbf{L}^{-1}\mathbf{s}$ *and* $\mathbf{z} = \mathbf{U}^{-1}\mathbf{z}'$)
5.9. $\mathbf{t} = \mathbf{A}\,\mathbf{z}$;
5.10. $\omega_j = \frac{(\mathbf{t}, \mathbf{s})}{(\mathbf{t}, \mathbf{t})}$;
5.11. $\mathbf{x}_j = \mathbf{x}_{j-1} + \alpha_j\,\mathbf{y} + \omega_j\,\mathbf{z}$;
5.12. $\mathbf{res} = \mathbf{b} - \mathbf{A}\,\mathbf{x}_j$;
5.13. $RSQ = (\mathbf{res}, \mathbf{res})$;
5.14. *Check if convergence is reached:*
5.14.1 *if* $(RSQ < tol)$ *then quit.*
5.14.2 *else* $\mathbf{r}_j = \mathbf{s} - \omega_j\,\mathbf{t}$ *and the cycle continues.*

where $\mathbf{K}$ is a preconditioning matrix obtained with the lower L and upper U triangular matrix ($\mathbf{K} = \mathbf{L}\,\mathbf{U}$) at points 5.4 and 5.8. The lower and upper triangular matrix have been obtained with the *incomplete LU factorization.*

The scheme of the algorithm for the $ILU(0)$ factorization expressed in CRS format to obtain the new matrix coefficients value ($\mathbf{vals}_{ILU}$) is shown below.

ALGORITHM 2.4- *ILU(0) factorization in CRS format*

$\mathbf{vals}_{ILU} = \mathbf{vals}$; *the vector is initialized*
do $i = 2, .., Ne$
  $l_{i1} = limits(i)$ and $l_{i2} = limits(i+1) - 1$
  do $k = l_{i1}, l_{i2}$
    if $(cols(k) < i)$ then
      $l_{k1} = limits(cols(k))$ and $l_{k2} = limits(cols(k)+1) - 1$
      $vals(k) = \dfrac{val_{ILU}(k)}{val_{ILU}(lk1)}$
      do $j = l_{i1}, l_{i2}$
        if $(cols(j) > cols(k)$ *and* $cols(k) \leq n_e)$ then
          do $m = l_{k1}, l_{k2}$
            if $(cols(m) = cols(j))$ then
              $vals_{ILU}(j) = vals_{ILU}(j) - vals_{ILU}(k)\, vals_{ILU}(m)$
              quit the m cycle

where i, j, k are used as indices of the cycles, l_{i1} and l_{i2} are the limits of the column i and l_{k1} and l_{k2} are the limits of the column of the k element. As for the vector **vals**, the dimension of the vector $\mathbf{vals}_{ILU}$ is equal to the total number of non-null elements. The product for the preconditioning matrix **K** at points 5.4 and 5.8 of the *Preconditioned BiCGSTAB* algorithm is obtained considering the lower and upper matrix as shown below.

ALGORITHM 2.5- *Solve Lower and Upper system*

$\mathbf{y}' = \mathbf{L}^{-1}\mathbf{p}$ *the lower system is solved*
do $j = 1, n_e$
  $l_{j1} = limits(j)$
  $l_{j2} = limits(j+1) - 1$
  do $k = l_{j1}, l_{j2}$
    if $(cols(k) < j)$ then
      $y'(j) = y'(j) - vals_{ILU}(k)\, y'(cols(k))$

$\mathbf{y} = \mathbf{U}^{-1}\mathbf{y}'$ *the upper system is solved*
do $j = n_e, 1, -1$
  $l_{j1} = limits(j)$
  $l_{j2} = limits(j+1) - 1$
  do $k = l_{j1}, l_{j2}$
    if $(cols(k) > j$ *and* $cols(k) \leq n_e)$ *then*
      $y(j) = \dfrac{y(j)}{vals_{ILU}(l_{j1})}$

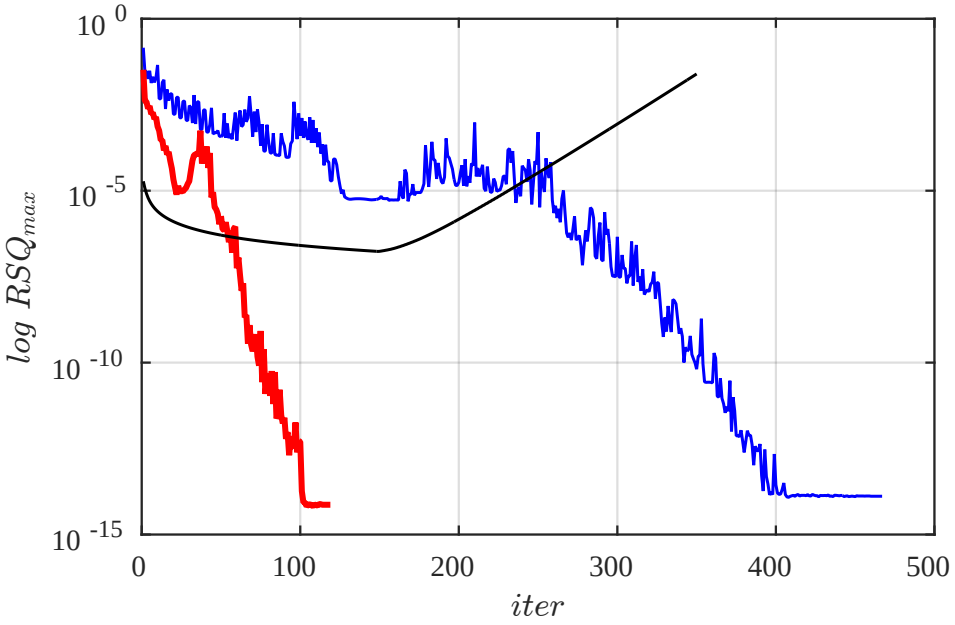


Figure 2.10: Comparison of different solution methods of the *PPE* system. Blue line: *BiCGSTAB* method; red line: *Pre-BiCGSTAB* method; black line: *SOR* algorithm. RSQ_{max}: maximum residual expressed using the semi-logarithmic scale); *iter*: iterations.

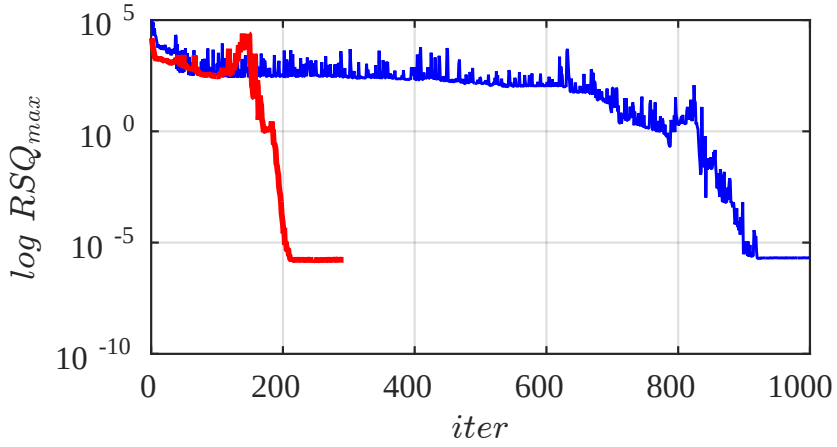
---

where the product $\mathbf{y} = \mathbf{K}^{-1}\mathbf{p}$ is solved in two steps: $\mathbf{y}' = \mathbf{L}^{-1}\mathbf{p}$ and $\mathbf{y} = \mathbf{U}^{-1}\mathbf{y}'$.

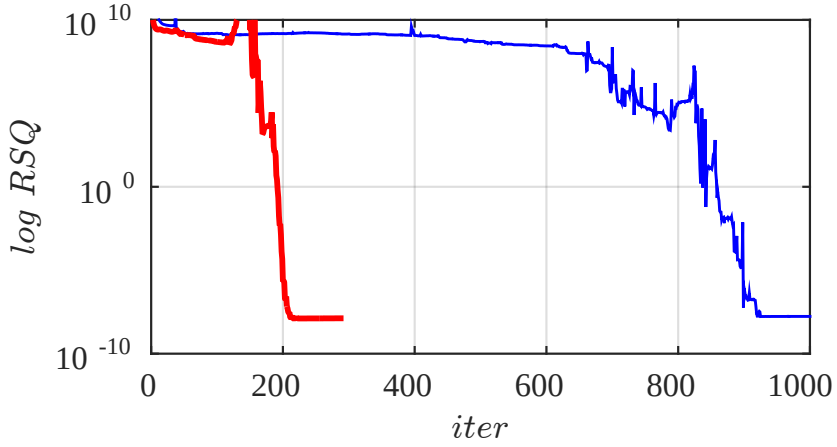
2.7.2 Results

The Fig. 2.10 shows a performance comparison among the three different algorithm explained in this section: *BiCGSTAB*, *Pre-BiCGSTAB* and the *SOR* method previously implemented in *PANORMUS-SPH*. Specifically, the 3*D lid-driven cavity* problem has been considered. The domain geometrical features and boundary conditions are identical to those discussed in Appendix A in the framework of the *Coupled FVM-SPH* method (see Sec. A2.1), while here the *SPH* method has been used for representing the whole computational domain resulting in a total number of *effective* particles $N_e = 125\,000$. The semi-logarithmic scale is used in the y-axis for the maximum residual RSQ_{max}, which expresses the maximum difference between the current solution and that of the previous iteration. As it is seen in the figure, the *SOR* algorithm reaches the minimum value of $RSQ_{max} \cong 10^{-7}$ after about 150 iteration and then diverges due to numerical instability, whilst the *BiCGSTAB* method needs 400 iteration to converge with $RSQ_{max} \cong 10^{-13}$. The *Pre-BiCGSTAB* algorithm is dramatically faster; the RSQ_{max} is quite the same of the *BiCGSTAB* but with the preconditioning algorithm the *Pre-BiCGSTAB* reaches the convergence after 100 iterations. In this application the *Pre-BiCGSTAB* algorithm is 4 times faster than the *unpreconditioned* version of the method.

The Fig. 2.11 shows a comparison of the *BiCGSTAB* and *Pre-BiCGSTAB* in a more geometrically complex domain. To this aim, the test case of the aneurysm *C*05, that will be discussed in Chap. 6, has been chosen. The maximum residual RSQ_{max} and the sum of the squared error RSQ are plotted during the iteration of the *PPE* (Figs. 2.11.a and



(a)



(b)

Figure 2.11: Comparison between *BiCGSTAB* (blue line) and *Pre-BiCGSTAB* (red line) algorithms. The test case shown in Sec. 6.4 (aneurysm *C*05). The semi-logarithmic scale is used in the y-axis. a) Maximum residual (RSQ_{max}); b) sum of the squared errors (RSQ).

2.11.b, respectively). Again the *Pre-BiCGSTAB* algorithm is about 4.5 times faster than the *unpreconditioned* version of the method.

2.8 Instability in SPH

The *tensile instability* problem, identified by Swegle et al. (1995), is a well-known issue in the *SPH* method. It happens when the motion of the particle becomes unstable resulting in particle clumping. This issue can lead to complete blow-up in the computation. The reason for the instability is that, although initially the particles are distributed regularly and under uniform initial stress, during the simulation running they can clump forming voids in the computational domain (Liu and Liu, 2010). Several methods have been pro-

posed to overcome the *tensile instability* problem such as the use of the *artificial stress* method for stabilizing the computation (Monaghan, 2000; Gray et al., 2001), the addition of stress points other than the normal particles (Dyka et al., 1997), the correction of the *kernel* to give exact linear interpolations (Dilts, 1999; Bonet and Kulasegaram, 2000) and the employing of particle *shifting* procedures (Xu et al., 2009; Lind et al., 2012; Khayyer et al., 2017).

In the *PANORMUS-SPH* solver, in order to overcome the *tensile instability*, the algorithm proposed by Xu et al. (2009) is adopted. The procedure involves slightly shifting the particles across streamlines in order to avoid the extreme stretching and bunching of particles. To this aim, the new particle position $\mathbf{x}_i^{k+1}$ (calculated through the eqn. 2.25) is modified with a further small shifting $d\mathbf{s}_i$ obtained as

$$d\mathbf{s}_i = \alpha\, \overline{u}\, \Delta t\, \mathbf{r}_i \tag{2.32}$$

where α is a constant whose values can be set in the range $0.01 \div 0.5$. Differently from the proposal of Xu et al. (2009), where the maximum particle velocity is used, here the average of the velocity particle magnitude $\overline{u}$ is employed in order to avoid too large $d\mathbf{s}_i$ when the particle velocities are quite large in specific portions of the domain. The shifting vector $\mathbf{r}_i$, which identifies the shift direction, is calculated as

$$\mathbf{r}_i = \sum_{j=1}^{N_s} \frac{\mathbf{d}_{ij}}{d_{ij}^3} \left(\frac{\sum_{j=1}^{N_s} \mathbf{d}_{ij}}{N_s} \right)^2 \tag{2.33}$$

where $\mathbf{d}_{ij}$ is the vector distance between i and the neighboring particle j and d_{ij} is its norm. It has been verified that using in the summation of eqn. 2.33 a smaller *support domain* than the classical Ω_i the shifting procedure gives the best results. Specifically, in order to obtain a more regular shifting in the particles with distances not larger than $0.7\,kh$ (rather than the standard kh value) from the position of i are considered (their number is indicated in eqn. 2.33 with the symbol N_s).

The value of the hydrodynamic variables ($\mathbf{x}_i + d\mathbf{s}_i$) are then adjusted in the new position through a Taylor series expansion

$$\Psi_i' = \Psi_i + (\nabla\Psi)_i\, d\mathbf{s}_i + O(r_i^2) \tag{2.34}$$

where Ψ is the generic variable and the superscript $'$ is used to indicate the particle properties in the new position. The shifting procedure allows to avoid clustering, maintaining an ordered particle distribution.

The shifting procedure has been employed in all the considered test cases with $\alpha = 0.1$.

2.9 Adaptive time step procedure

During the research, in the *PANORMUS-SPH* code, an adaptive time step has been introduced allowing to speed-up the simulation time when the velocity field changes suddenly (e.g. when the waveform of the cardiac cycle is imposed at the inlet as it will be discussed in Sec. 6.4 and Sec. 6.5).
To this aim, the minimum and maximum values of the *Courant-Friedrichs-Lewy* (*CFL*) stability condition are imposed and, at the end of each iteration, it is checked if

$$CFL_{min} < CFL < CFL_{max} \tag{2.35}$$

During the simulation, the CFL_{min} and CFL_{max} have been set to 0.2 and 0.5, respectively. The CFL number is calculated as

$$CFL = \frac{\overline{u}_{max}\Delta t}{\Delta x} \tag{2.36}$$

where $\overline{u}_{max}$ is the maximum magnitude particle velocity in the whole domain and Δx is the *starting particle distance*.

If the condition eqn. 2.35 is not satisfied due to $CFL < CFL_{min}$ or $CFL > CFL_{max}$, the time step is modified (increasing or decreasing, respectively, its value).

The *Adams-Bashforth* scheme to calculate the diffusive term in the *predictor-step* eqn. 2.20 is modified as

$$\frac{\mathbf{u}_i^* - \mathbf{u}_i^k}{\Delta t} + c_1\,\mathbf{D}_i^k - c_2\,\mathbf{D}_i^{k-1} - \mathbf{f}_{m_i} = 0 \tag{2.37}$$

with

$$\begin{aligned} c_1 &= \frac{2\,\Delta t_{old} + \Delta t}{2\,\Delta t_{old}} \\ c_2 &= \frac{\Delta t}{2\,\Delta t_{old}} \end{aligned}$$

where Δt and Δt_{old} are the current and the previous time step. Eqn. 2.37 becomes eqn. 2.20 when $\Delta t = \Delta t_{old}$.

2.10 Structure of the *PANORMUS-SPH* code

The Fig. 2.12 shows the flow chart of the *SPH* procedure in the *PANORMUS-SPH* code.

The code can be subdivided in three sections: *initialize*, *initial conditions* and *time marching*.

- ***INITIALIZE***

  ACTION 1: The boundary triangles are read from a file together with the boundary conditions to be set at each triangle;

  ACTION 2: The virtual grid (described in Sec. 2.5.1) is created identifying its left-south-down corner $\mathbf{x}_0$ subtracting the distance kh (in all the directions) to the boundary triangle having the vertex with the lowest coordinates. The number of virtual cubic cells in the three directions (nx, ny, nz) are calculated as well (eqns. 2.13). Then the cubic cells of the virtual grid are classified in types 1, 2, 3 and 4.

- ***INITIAL CONDITIONS***

  ACTION 3: The *SPH* numerical model needs an initial particle distribution that is generated by the code with starting isotropic particle distance equal to Δx. If the simulation is starting from developed conditions, the code reads from a file, with *vtk* format, the particle positions and their hydrodynamic variables. Specifically, velocity, *pseudo-pressure*, specie concentration (if the tracer module is activated as will be explained in Sec. 7.2) and *activation potential* (whose definition will be given in Sec. 7.4) are read. On the contrary, if the simulation starts from the rest, these variables are set to zero. The virtual cell of each particle is calculated (eqn. 2.14);

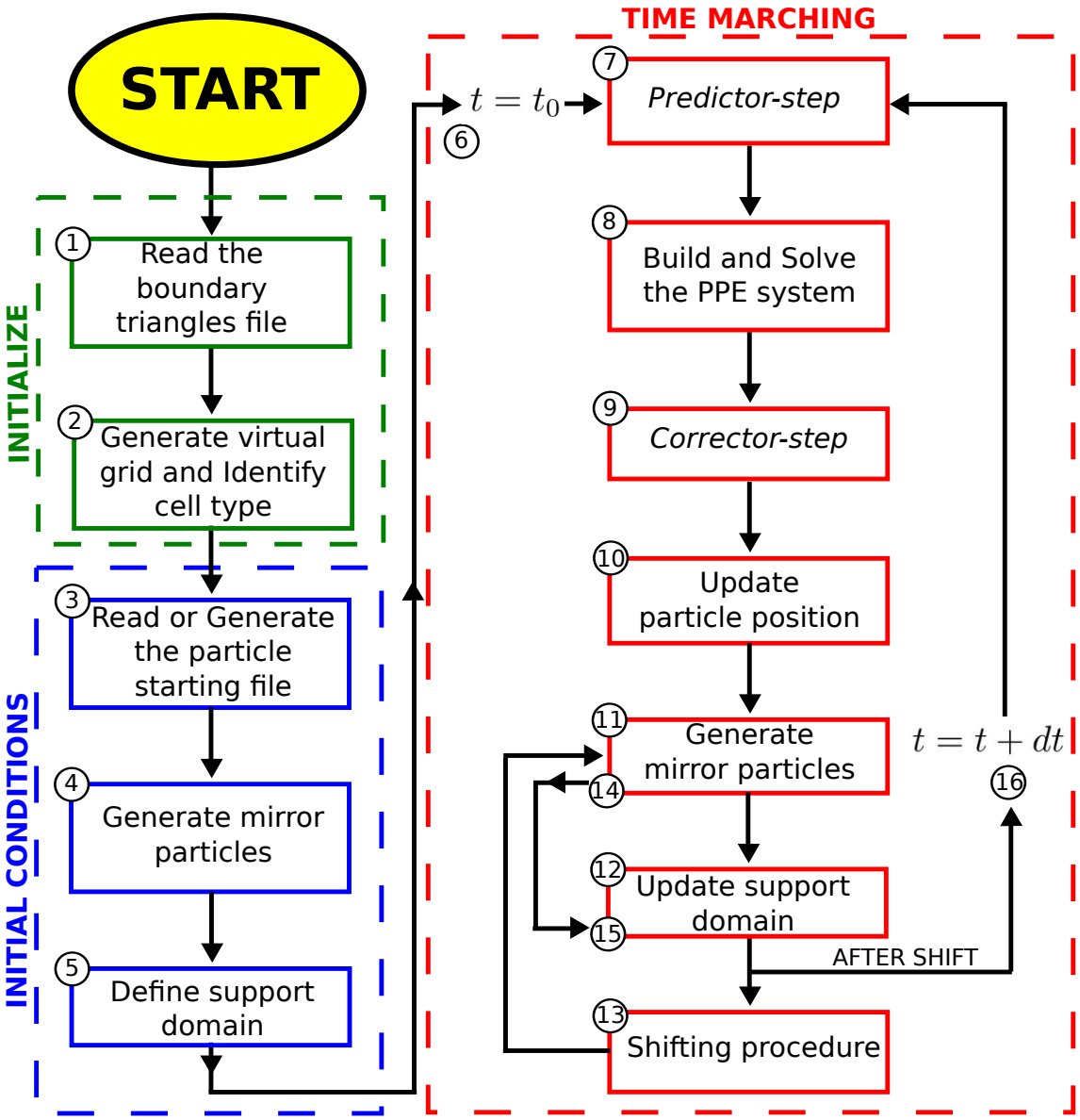


Figure 2.12: Flow chart of the *PANORMUS-SPH* code.

ACTION 4: The *mirror* particles are generated following the procedure described in Sec. 2.5;

ACTION 5: The *support domain* of the *effective* particles is thus defined. To this aim, for each *effective* particle the surrounding particles having distance from it shorter than kh are identified and are stored in a list.

- ***TIME MARCHING***

ACTION 6: The *fractional-step* procedure starts from the simulation time t_0 (that can be $t_0 = 0$ or $t_0 \neq 0$, if the simulation starts from the rest or from developed conditions, respectively);

ACTION 7: The *intermediate* velocity is calculated for the *effective* particles

through eqn. 2.20 (or eqn. 2.37 if the time step is variable as explained in Sec. 2.9). For the *mirror* particles u_n^* is calculated at the $\mathbf{x}_b$ position using eqn. 2.23;

ACTION 8: The eqn. 2.21 must be solved for all the *effective* particles as a linear system in the form of $\mathbf{A}\,\mathbf{x} = \mathbf{b}$. The *PPE* matrix is created using the *CRS* format. If the *BiCGSTAB* method is used in its preconditioned version, the $ILU(0)$ *factorization* is performed (following ALGORITHM 2.4) in order to obtain the preconditioned matrix. Therefore, the *Pre-BiCGSTAB* iterative method is applied following ALGORITHM 2.3. After calculating the ψ values of the *effective* particles, the corresponding ψ values of the *mirror* ones are obtained through eqn. 2.22;

ACTION 9: The *corrected* velocities $\mathbf{u}_i^{k+1}$ are calculated using eqn. 2.24;

ACTION 10: For each *effective* particle, the new position $\mathbf{x}_i^{k+1}$ is obtained through eqn. 2.25;

ACTION 11: After updating the particle positions, the *mirror* particles are generated and the old *mirror* ones are deleted;

ACTION 12: For each *effective* particle the *support domain* is updated saving in its list the new neighboring particles;

ACTION 13: In order to improve the particle distribution, the *effective* particles are shifted of $d\mathbf{s}_i$ (calculated with eqn. 2.32) and the hydrodynamics variables in the new position are updated through eqn. 2.34, as explained in Sec. 2.8;

ACTION 14: After the *shifting* procedure, the *mirror* particles are generated (again deleting the old *mirror* ones);

ACTION 15: This action is identical to *ACTION* 12;

ACTION 16: The simulation time is advanced by one time step ($t = t + dt$). If the adaptive time step procedure is activated it must be checked if the *Courant* limit constraint is satisfied (2.35) or if it is necessary to change the time step as discussed in Sec. 2.9.

After the sixteenth action, the procedure is restarted with the *predictor-step* (*ACTION* 7).

Chapter 3

The Inflow/Outflow procedure

In this chapter a novel procedure to treat inflow and outflow boundaries is described in the *ISPH* framework. The procedure is carefully explained in Monteleone et al. (2017), where some validation test cases are discussed too. The technique allows to set pressure values in the computational domain inlets and outlets or to assign the velocity profile at the inlets and the pressure at the outlets.

Two $3D$ numerical tests are presented to show the performance of the method considering both steady and oscillating inflow conditions.

3.1 Background and motivations

In order to reduce the computational costs, in the numerical modeling of blood flow in cerebral aneurysms it is customary to limit the size of the computational domain to the vessel tract hosting the *CA*. Therefore, artificial inlet and outlet sections must be identified (see Fig. 3.1), where proper boundary conditions must be prescribed. Due to the lack of *patient-specific* information, the correct treatment of inflow and outflow boundaries is a key problem in hemodynamics simulations.

Open-boundary treatment is relatively simple in *grid-based* method since it is sufficient to impose the values of the hydrodynamic variables at the inlet and outlet sections. On the other side, it is a well-known issues in the *SPH* numerical scheme particularly when the *ISPH* approach is employed.

Furthermore, the management of the particles leaving or entering the domain through *open-boundaries* is very challenging.

In many *SPH* simulations *periodic* boundary conditions are employed at the *open-boundaries* (Lee et al., 2008; Morris et al., 1997), using a mass force to drive the flow. This approach is limited to very simple geometries and obviously it cannot be usually employed to study *CAs* flow dynamics.

Other authors proposed procedures to assign at inflow cross-sections *Dirichlet* boundary conditions for the velocity (Hosseini and Feng, 2011; Vacondio et al., 2011; Federico et al., 2012; Khorasanizade and Sousa, 2016b). These procedures can be used when the velocity profiles are available.

Otherwise, when the velocity profile is unknown, other techniques have been developed to impose *Dirichlet* boundary conditions for the pressure at the inlet and outlet sections. Tan et al. (2015) developed a procedure not requiring the inflow velocity assignment for open channel flows. In this procedure, wall and *dummy* particles are moved considering

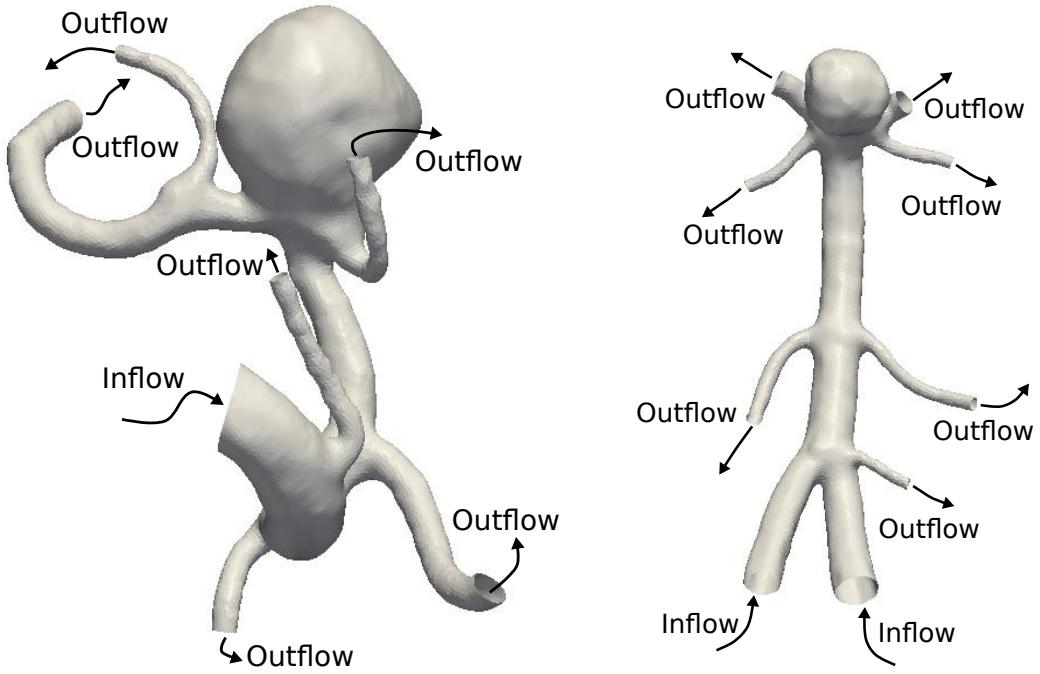


Figure 3.1: Cerebral aneurysm geometries. Taken from: Aneurisk dataset repository (Aneurisk-Team, 2012). a) case id: $C93$. Location: middle cerebral artery; b) case id $C95$. Location: basilar artery.

a uniform velocity equal to the channel depth-averaged one. However, the method cannot be efficiently employed when the effective inflow velocities are quite different from their depth-averaged values. Khorasanizade and Sousa (2016a) generalised the procedure of Khorasanizade and Sousa (2016b) to incompressible flows where inlet velocity profiles were unknown, thus extending its use to more general problems. Hirschler et al. (2016) proposed a procedure dividing the *open-boundaries* in multiple segments, each one connected to a fluid particle, which move according to the fluid velocity (the *moving-mirror-axes*, MMA, technique). A new particle is placed into the domain when the movement of a boundary segment exceeds a given distance from the initial position, while the boundary segment is correspondingly shifted behind the new particle. However, some numerical errors are introduced in the MMA method, due to the axes shifting. The MMA procedure was improved by Kunz et al. (2016) allowing the mirror axes to remain fixed (the *fixed-mirror-axes*, FMA, technique). Nevertheless, although the FMA technique works fine in a wide range of flow conditions, it has some issues related to the occurrence of void spaces and not uniform particle distribution at the inflow/outflow. In Leroy et al. (2016) the masses of the inlet/outlet particles are let to evolve over time as a function of the desired ingoing/outgoing mass flux through the corresponding cross-sections. The procedure requires specific care to ensure a smooth evolution since the mass evolution could effect the flow. Recently, Tafuni et al. (2018) developed an *open-boundary* algorithm in the *WCSPH* approach to simulate real engineering problems with free-surface. The model is based on the use of buffer layers near the open regions of the computational domain, where the particles contained are used as a means of enforcing certain boundary conditions.

A different procedure is proposed here which allows for the treatment of the inflow

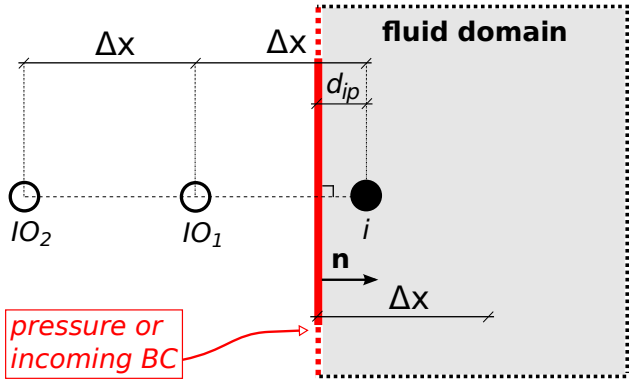


Figure 3.2: 2*D* sketch of the *IO* particles generation at inflow and outflow boundaries. Full circles: *effective* particles; empty circles: *IO* particles; bold red line: inlet and outlet boundaries.

and outflow regions avoiding any occurrence of void spaces and the reflection of numerical noises into the fluid domain. The procedure (named *In/OutFlow-BCs* in the following) has been developed in the *ISPH* framework. The *In/OutFlow-BCs* technique is specifically suitable to simulate the flow inside cerebral aneurysms and will be widely explained in this chapter. The *In/OutFlow-BCs* method can be used to impose different hydrodynamic variables at the inflow according to the available physical quantities. Specifically, it is possible to impose *Dirichlet BCs* for the velocity at the inlet sections (in the following this condition will be named *incoming BCs*), whilst *Dirichlet BCs* for the pressure and *Neumann BCs* for the velocity at the outlet sections. When the velocity profiles are not available, the technique allows to impose *Dirichlet BCs* for the pressure and null velocity derivatives (this condition will be named *pressure BCs*) at the inlet and outlet sections.

Recently, Wang et al. (2019) proposed a specific non-reflective boundary condition (NRBC) approach in *SPH* which allows outward traveling pressure and velocity messages to pass through the boundary without reflection. To this aim, a buffer layer of boundary particles placed outside the fluid domain is used. The velocity and pressure of these particles are obtained through the Lagrangian interpolation in time which is derived from the propagation of characteristic waves between particles. A similar procedure could be implemented in the *In/OutFlow-BCs* algorithm; nevertheless, for the test cases considered in this thesis the use of NRBC seems less necessary than in the problem analyzed by Wang et al. (2019) which simulated underwater explosion.

3.2 The *In/OutFlow-BCs* algorithm

The *In/OutFlow-BCs* procedure is activated when an *open-boundary* exists. The *open-boundaries* are discretized into triangles as explained in Sec. 2.5 for the domain boundaries at solid walls. As discussed above, two type of conditions can be assigned at the *open-boundaries*: *Dirichlet BCs* for the velocity (when *incoming BCs* are imposed at the *open-boundary* triangles) or *Dirichlet BCs* for the pressure (when *pressure BCs* are employed).

The new technique allows to treat in the same way inflow and outflow sections, effectively dealing with the release of new particles at inlets and the deactivation of the ones leaving the domain through the outlets.

3.2.1 The IO particles

The IO particle generation

When *incoming* or *pressure BCs* are set at one boundary triangle, the *effective* particles with distance from the triangle less than Δx are identified in order to release a particular type of *ghost* particles indicated as *in/out-flow particles* (IO). Specifically, starting from the identified particle i (having $d_{ip} < \Delta x$, where d_{ip} is the distance between i and the *pressure* or *incoming* triangle) a number "n" of IO particles are generated along the line normal to the triangle plane and passing from the i position. These IO particles are displaced at distance from the relative generating particle equal to $n\Delta x$ where $n = 1, 2, 3, ..$ is the integer part of the ratio $kh/\Delta x$. As discussed in Chap. 2, in this research study the ratio $kh/\Delta x$ has been set to 2, therefore two IO particles are generated for each selected i particle. It should be noted that, if the most common ratio $kh/\Delta x = 2.66$ had been used, three IO particles would have been generated for each i particle to avoid the *kernel* truncation.

The Fig. 3.2 shows a $2D$ sketch of the IP particle generation. In the figure, due to the $2D$ representation, the *open-boundary* triangle (*pressure* or *incoming* BC) is reduced to a segment, indicated with the bold red line.
The two IO particles (IP_1 and IP_2 in the figure) are placed at distances from i equal to Δx and $2\,\Delta x$, respectively.

Pressure boundary conditions

When *pressure BCs* are imposed to the *open-boundary* triangles, the velocity of the IO particle is set equal to that of the generating particle g (*homogeneous Neumann BCs*, as in Fig. 3.3.a). If some regularization of the inflow or outflow velocity is wanted, the *homogeneous Neumann BC* can be used for the normal velocity component only, while the tangential component is set to zero (see Fig. 3.3.b). Inflow and outflow boundaries are treated in the same way as shown in Fig. 3.3.c where an outflow boundary with null normal derivative for the normal velocity component is considered.

The potential ψ for the j-th IO particle can be obtained through a linear extrapolation based on the generating particle ψ_g value and the assigned kinematic pressure ψ_p at the *pressure* boundary (as shown in Fig. 3.4)

$$\psi_j = \psi_p \frac{d_{gj}}{d_{gp}} - \psi_g \frac{d_{gj} - d_{gp}}{d_{gp}} \tag{3.1}$$

Therefore, the PPE system must to be modified as will be explained in Sec. 3.2.5.

Incoming boundary conditions

When *incoming BCs* are set at the *open-boundary* triangles, the velocity at the inflow section must be imposed. To this aim, two different profile laws can be prescribed: the *Poiseuille* and the *Womersley* velocity profiles.

For the generic IO particle, considering a circular cross-section, the *Poiseuille* velocity profiles can be expressed as

$$\mathbf{u}_{IO} = \overline{u}_{inflow} \left[2 - 8 \left(\frac{r}{D} \right)^2 \right] \cdot \mathbf{n} \tag{3.2}$$

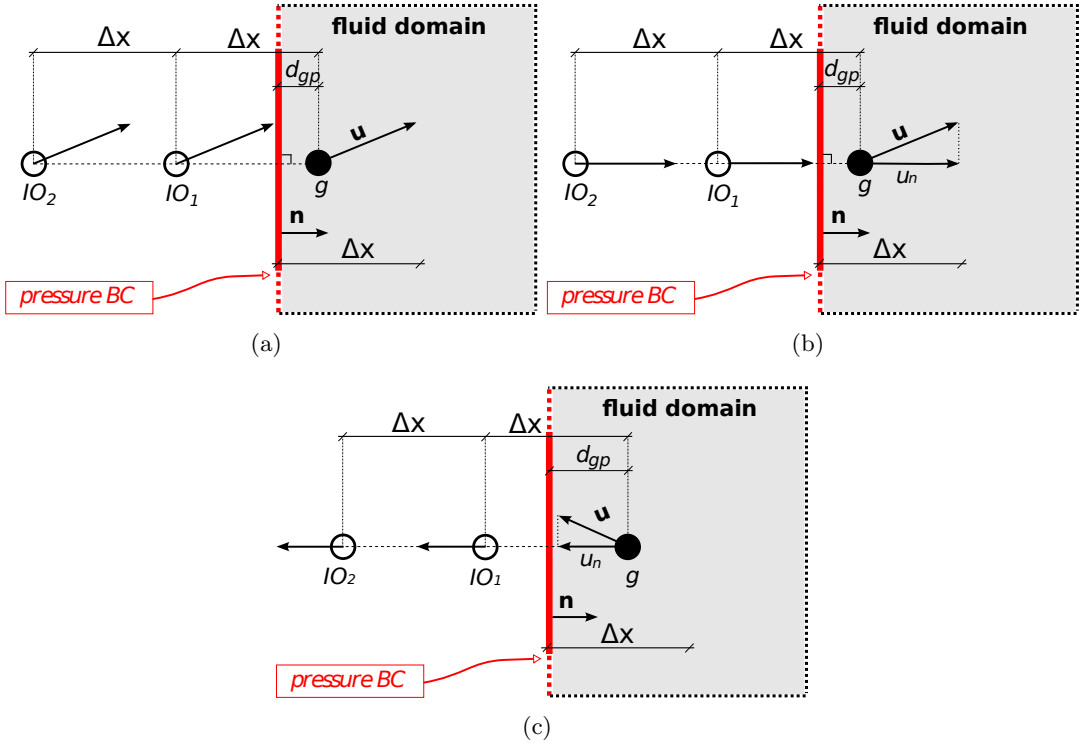


Figure 3.3: 2*D* sketch of the *IO* particles conditions for *pressure BCs*. Bold red line: inlet and outlet *pressure* boundaries. The other symbols as in Fig. 3.2. a) Inlet section with *homogeneous Neumann BCs* for the velocity; b) inlet section with *homogeneous Neumann BCs* for the normal velocity (u_n) and tangential velocity set to zero for *IO* particles; c) as in the previous point .b considering an outlet section. Taken from: Monteleone et al. (2017), 13, fig. 1.

where $\overline{u}_{inflow}$ is the mean velocity, D is the diameter of the inflow section, $\mathbf{n}$ is the triangle normal direction and r is the distance between $\mathbf{x}_c$ (the center of the inflow section) and $\mathbf{x}_{int}$ (the intersection point between the triangle plane and the line passing through the *IO* particle and normal to the plane).

For the fully developed pulsatile flow in a rigid straight pipe the *Womersley* solution (Womersley, 1955) can be prescribed rather than the *Poiseuille* profile. The *Womersley* solution can be used when transient problems such as blood flow in the cardiovascular system are considered. As discussed by Taylor et al. (1998), the velocity profiles are computed from the *Fourier* decomposition of the prescribed flow-rate curves to extract the frequency content of the volume flow waveform given the fundamental frequency ω (heart rate expressed in radians/sec). When the flow rate $Q(t)$ is known, it can be decomposed into N Fourier modes with the Fourier coefficients B_n given by

$$Q(t) \approx \sum_{n=0}^{N} B_n \, e^{in\omega t}$$

The *Womersley* velocity profile for the axial component of velocity at the position of the

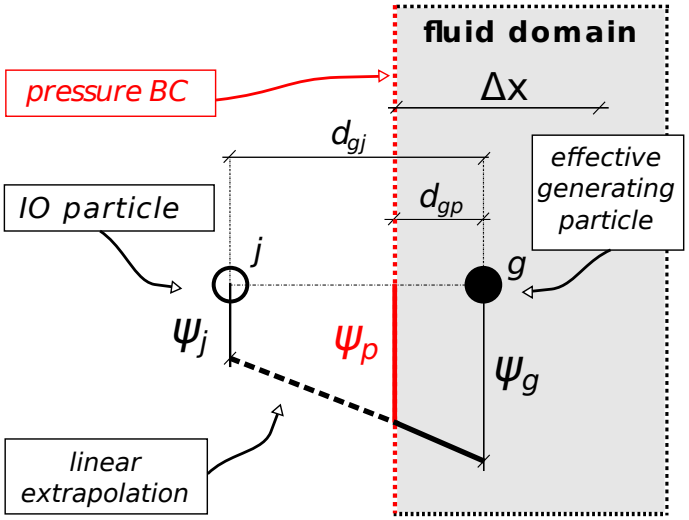


Figure 3.4: *PPE BCs* at *pressure* boundaries. Linear extrapolation between the value of the generation particle ψ_g and the value at the *pressure* boundary triangle ψ_p. Taken from: Monteleone et al. (2017), 15, fig. 5.a.

IO particle ($u_{r,IO}$) obtained as

$$u_{r,IO}(t) = \frac{2B_0}{\pi(D/2)^2}\left[1-\left(\frac{2\,r}{D}\right)^2\right] + \sum_{n=1}^{N}\left\{\frac{B_n}{\pi(D/2)^2}\frac{1-\frac{J_0(\alpha_n \frac{r}{R} i^{3/2})}{J_0(\alpha_n i^{3/2})}}{1-\frac{2J_1(\alpha_n i^{3/2})}{\alpha_n i^{3/2} J_0(\alpha_n i^{3/2})}}\right\} e^{in\omega t} \tag{3.3}$$

with

$$\alpha_n = D/2\,\sqrt{n\omega/\nu}$$

where J_0 and J_1 are the Bessel functions of the first kind of order 0 and 1, respectively, $Wo = D/2\,\sqrt{\omega/\nu}$ is the *Womersley number* and i is the imaginary number. The velocity is thus obtained as

$$\mathbf{u}_{IO}(t) = u_{r,IO}(t)\cdot\mathbf{n}$$

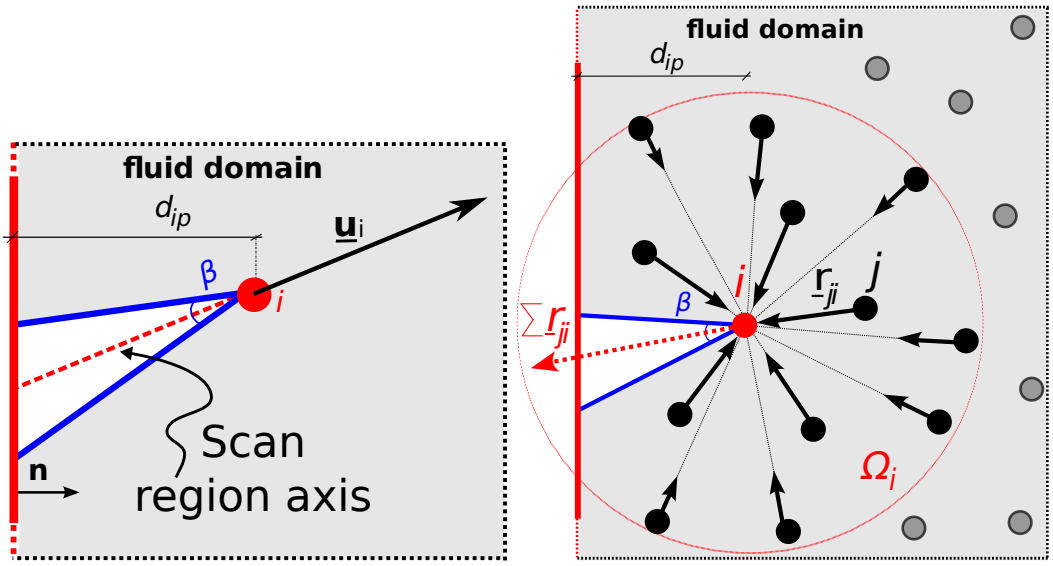
where $\mathbf{n}$ is normal direction of the triangle from which the *IO* has been generated.

Differently from the *pressure BCs*, in this case the *Poisson BCs* are the same as those explained for *mirror* particles at the solid wall (eqn. 2.22) and the *PPE* matrix system (eqns. 2.29,2.30 and 2.31) thus does not change.

3.2.2 Generation of new particles

Besides setting the correct conditions, the particles entering the domain have to be dealt with. If the distance d_{ip} of the *effective* particle i from the *open-boundary* ranges between Δx and the kh value and, moreover, the velocity component normal to the boundary triangle is positive ($\mathbf{u}_i \cdot \mathbf{n} > 0$), thus implying an inflow condition, it must be checked if the release of a new particle is required. To this aim, a conical *scan region* is considered, having the vertex in the position of the particle i and the angle β. Three strategies can be used to identify the direction of the conical region axis depending on the user's choice. The first one, suitable for regular and very simple geometries (such as the flow in a circular pipe), consists of setting the direction of the conical region axis equal to the boundary triangle normal direction $\mathbf{n}$. In the second one, which is more general, the axis

direction of the cone is assigned equal to the particle velocity ($\mathbf{u}_i$) direction, as shown in Fig. 3.5.a. Details of this procedure will be provided in Sec. 4.2.4 with reference to the *Multi-Domain* approach. In the third one, that is useful if some regularization is necessary,



(a) (b)

Figure 3.5: 2*D* sketch of the procedure to identify the axis of the *scan region*. Bold blue lines: bounds of the conical *scan region*. Bold red line: inflow boundary. a) The axis is assigned equal to the velocity direction of the particle i; b) The axis is obtained by summing up the vectors $\mathbf{r}_{ij}$ of the j particles in Ω_i having magnitude inversely proportional to the d_{ij} distance. Taken from: Monteleone et al. (2017), 13, fig. 2.a.

the direction of the conical region axis is identified by summing up the vectors $\mathbf{r}_{ij}$ pointing towards i starting from the surrounding j particles in Ω_i and having magnitude inversely proportional to the d_{ij} distance (see Fig. 3.5.b).

The surrounding *effective* particles j are then analyzed to check if lying inside or outside the *scan region*. To this aim, the angle γ between the cone axis and the line connecting the particle j with the cone vertex is calculated: if $\gamma > \beta/2$, the particle j is clearly outside the cone (see Fig. 3.6.a). If no *effective* particles are found inside the cone region, a new particle is created at a distance from i equal to Δx along the cone axis, as shown in Fig. 3.6.b. Since the new generated particle will lie at a distance from the boundary triangle shorter than Δx, two *IO* particles must be generated.

A dynamic procedure has been implemented in order to control the total number of *effective* particles in the domain. In particular, the opening angle β is dynamically changed at each time step, making it wider (by 1 degree) whenever the particle number is larger than the starting value, thus reducing the frequency of release of new particles (the opposite occurs when the particle number becomes lower than the starting value). In order to avoid too large fluctuations in the opening angle width, at each time step the average particle number in the last 10 time steps is used and the lower and upper bounds are set to 5° and 45°, respectively, with a starting value of 30°.

The new particles mass and density are set equal to those of the particle i in the vertex of the *scan region*. For *pressure BCs* the velocity of the new particle is set equal to that of the particle i (as for the *IP* particles explained in Sec. 3.2.1). For *incoming BCs* the

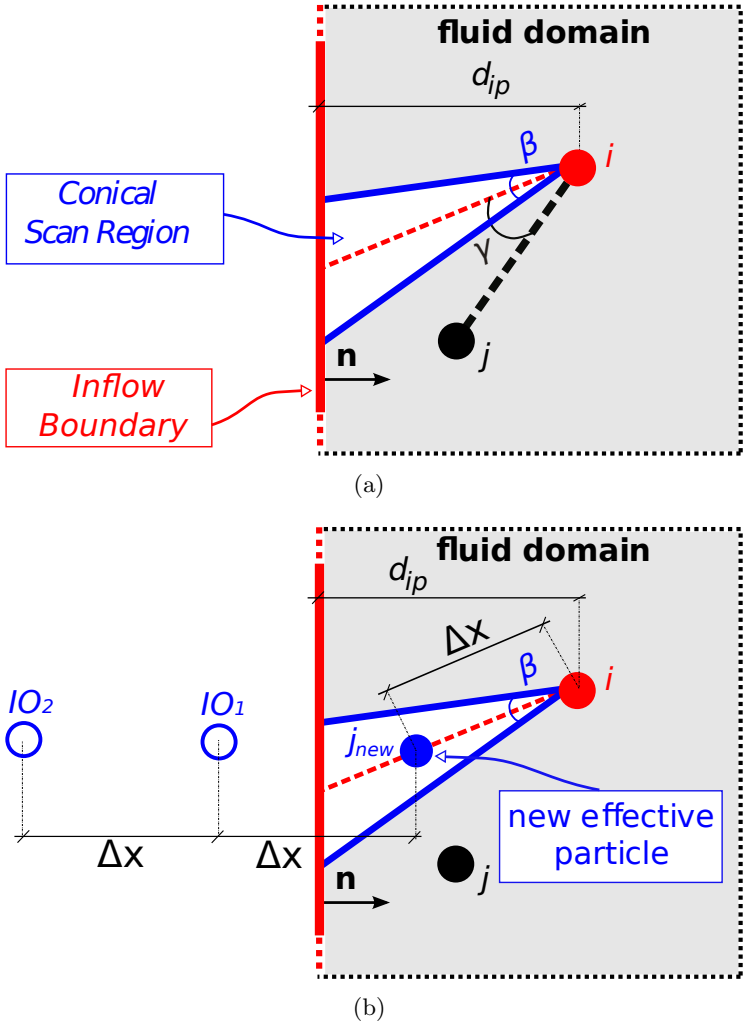


Figure 3.6: 2*D* sketch of the procedure to release new particles at inflow boundaries. Bold blue lines: bounds of the conical *scan region*; bold red line: inflow boundary. a) Checking of the position of the particle j with respect to the conical scan region; b) release of a new *effective* particle j_{new}. Taken from: Monteleone et al. (2017), 13, figs. 2.b and 2.c.

velocity of the new particle is imposed through the velocity profile law as discussed in Sec. 3.2.1.

3.2.3 Particles deactivation

If the velocity component normal to the triangle points outside the fluid domain ($\mathbf{u}_i \cdot \mathbf{n} < 0$), the boundary is an outflow section and no new *effective* particles must be generated. The *effective* particles are allowed to go out of the domain through *pressure* or *incoming* boundary triangles. Therefore, at the end of the time step the particles leaving the domain are deactivated and removed from further calculations. In order to avoid a continuous increase of the number of particles allocated in the computer memory (since new particles are continuously created at inflow boundaries), the deactivated particles are saved in a storage list from which selecting the new ones to be released. In Fig. 3.7 the *effective* particle P at the time r-th generates two IO particles, while it is deactivated at the time

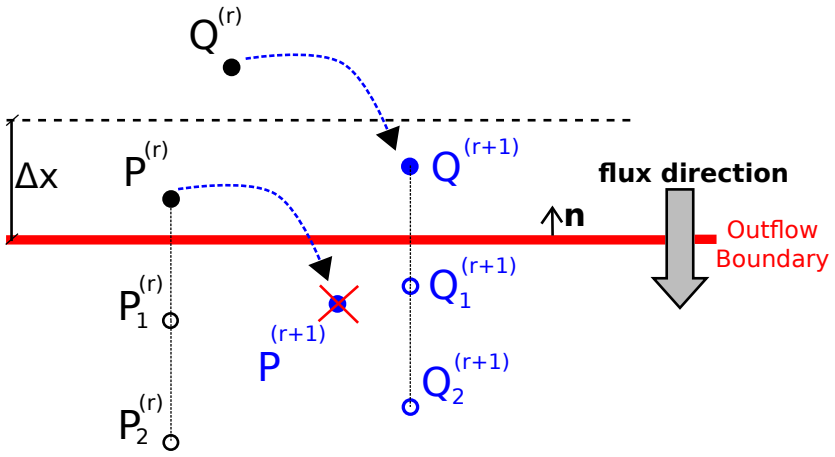


Figure 3.7: Particle deactivation at an outflow boundary (bold red line).

step $r+1$ since it has crossed the *open-boundary*. The Q particle at time $r+1$ approaches the *open-boundary* without crossing it, thus it generates two IO particles.

3.2.4 Flow chart of the *In/OutFlow-BCs* procedure

The flow chart shown in Fig. 3.8 summarizes the algorithm implemented to perform the *In/OutFlow-BCs* procedure.

1. The procedure starts for a particle i if its distance from the open boundary (with *pressure* or *incoming BCs*) is less than kh ($d_{ip} < kh$);
2. if $d_{ip} < \Delta x$ two IP particles are generated starting from i;
3. if $\Delta x < d_{ip} < kh$ and $\mathbf{u} \cdot \mathbf{n} < 0$ the boundary is an outflow and the procedure ends;
4. if $\Delta x < d_{ip} < kh$ and $\mathbf{u} \cdot \mathbf{n} > 0$ the *scan region* technique is activated for checking if the new particle must be created starting from i:
   (a) If no *effective* particles are found inside the *scan region*, a new particle is created. The new particle j generates two IP particles due to $d_{jp} < \Delta x$;
   (b) If at least one *effective* particle is found inside the *scan region* no new particle is created and the procedure ends.

3.2.5 The *PPE* system with *pressure BCs*

Pressure BCs are more challenging than the *incoming* ones since the *PPE* system (explained in Sec. 2.7.1) must be modified. The coefficient matrix and the *right-hand-side* term of the *Poisson* system (eqn. 2.21) are modified with respect to eqns 2.29, 2.30 and 2.31 when IO particles are generated through *pressure* boundaries. The potential ψ for the j-th IO particle can be obtained through eqn. 3.1 where a linear extrapolation between the value ψ_g of the generating particle g and the value ψ_p at the boundary is used (see Fig. 3.4).

The first term in eqn. 3.1 contributes to the system RHS, while the second one contributes to the coefficient matrix entries, since ψ_p and ψ_g are known and unknown values,

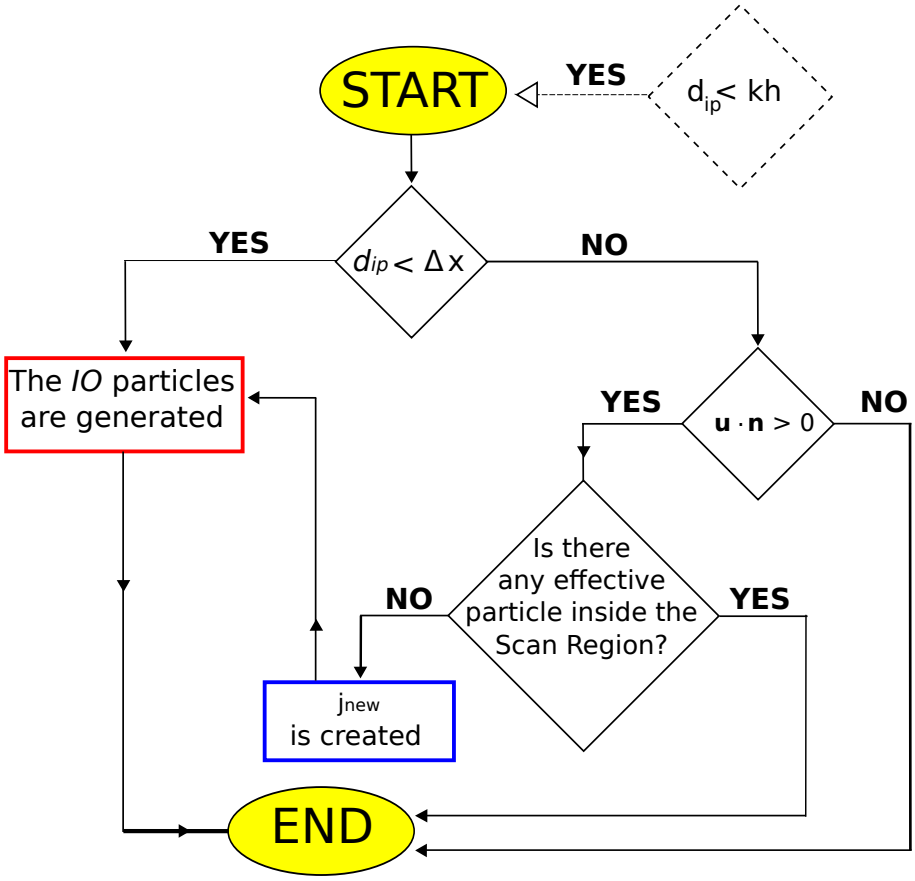


Figure 3.8: Flow chart of the *In/OutFlow-BCs* algorithm.

respectively.

Using eqn. 3.1, thus, the i-th row *diagonal* term of the coefficient matrix (eqn. 2.29) becomes

$$\sum_{j=1}^{N_i'} C_{ij} + \sum_{j=1}^{N_i^{IOi}} C_{ij} \frac{d_{ij} - d_{gp}}{d_{gp}} + \sum_{j=1}^{N_i^{IO}} C_{ij} \tag{3.4}$$

where the two additional summations (second and third term) are extended respectively to the IO particles in Ω_i generated by the current particle i only (N_i^{IOs}, with $s = i$; particles $IO_{1(i)}$ and $IO_{2(i)}$ in Fig. 3.9) and to all the IO particles in Ω_i (N_i^{IO}).

Correspondingly, the *i-s off-diagonal* entry of the system matrix, eqn. 2.30, is modified as

$$-\left(\delta_{is} C_{is} + \sum_{j=1}^{N_i^{Ms}} C_{ij}\right) + \sum_{j=1}^{N_i^{IOs}} C_{ij} \frac{d_{sj} - d_{gp}}{d_{gp}} \tag{3.5}$$

where the IO particles to be included in the last summation are only those generated by the *effective* particles s ($IO_1(s)$ and $IO_2(s)$ in Fig. 3.9) whose total number is indicated with N_i^{IOs}.

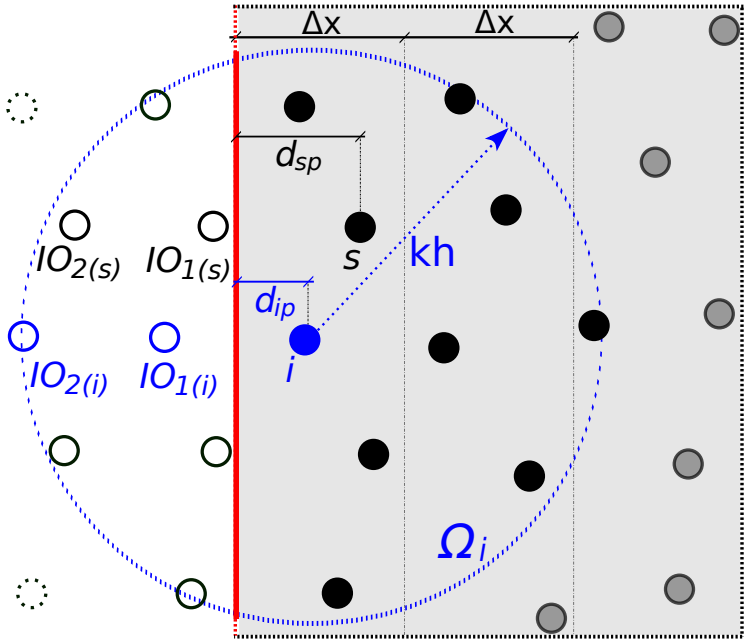


Figure 3.9: Scheme of the *IO* particles to be included in the *Poisson* system terms (eqns. 3.4, 3.5 and 3.6). Bold red line: *pressure* boundary; blue circle: current *effective* i particle; empty blue circles: *IO* particles generated by i; black circles: *effective* particle lying in Ω_i; empty black circles: *IO* in Ω_i not generated by i; grey circles: *effective* particles outside Ω_i; dotted empty circles: *IO* particles outside Ω_i. Taken from: Monteleone et al. (2017), 15, fig. 5.b.

The RHS of the i-th equation system (eqn. 2.31) is modified as:

$$T_i + \frac{1}{\Delta t}\sum_{j=1}^{N_i^M} C_{ij}\,(u_n^{k+1} - u_n^*)\Big|_b\, d_{gj} + \sum_{j=1}^{N_i^{IO}} C_{ij}\psi_p \frac{d_{gj}}{d_{gp}} \tag{3.6}$$

where again the index g indicates the *effective* particles generating the *mirror* and/or *IO* particles j (with $g = i$ or $g \neq i$ indifferently). The known term T_i is calculated through eqn. 2.28 where the summation is extended to the *effective* and *IO* particles in Ω_i. In eqn. 3.6 the *intermediate* normal velocity component u_n^* at the *open-boundaries* is calculated according to eqn. 2.23, without using *mirror* particles while setting the values of the *IO* particles equal to that of the *effective* generating particles. The *IO* particles are also used to calculate the divergence of the *intermediate* velocity in eqn. 2.28.

Example

An example of the *Poisson* equation for an *effective* particle close to a *pressure* boundary triangle is shown below (particle 2 in Fig. 3.10).

Example- Poisson equation for the particle 2 *in Fig. 3.10*

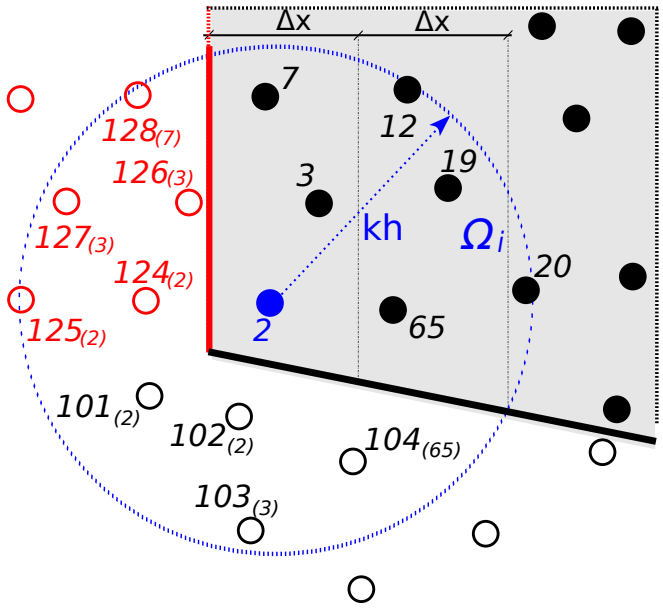


Figure 3.10: Black circles: *effective* particles; blue circle: particle 2; empty black circles: *mirror* particles; empty red circles: *IO* particles. For the *IO* and *mirror* particles the number inside the brackets is their own generating particle. Only the particles (*effective*, *mirror* or *IO*) lying in Ω_2 are numbered.

The 2-th row *diagonal* term of the coefficient matrix is

$$\begin{aligned} & C_{2,3} + C_{2,7} + C_{2,12} + C_{2,19} + C_{2,20} + C_{2,65} + C_{2,104} + \\ + \quad & C_{2,103} + C_{2,124}\,\frac{d_{2,124} - d_{2,p}}{d_{2,p}} + C_{2,125}\,\frac{d_{2,125} - d_{2,p}}{d_{2,p}} + \\ + \quad & C_{2,124} + C_{2,125} + C_{2,127} + C_{2,128} \end{aligned}$$

while the $(2-s)$ (second row, s-th column) *off-diagonal* entries of the system matrix are

- $(2-3)$ entry: $(C_{2,3} + C_{2,103}) + C_{2,126}\,\dfrac{d_{3,126} - d_{3,p}}{d_{3,p}}$
- $(2-7)$ entry: $C_{2,7} + C_{2,128}\,\dfrac{d_{7,128} - d_{7,p}}{d_{7,p}}$
- $(2-12)$ entry: $C_{2,12}$
- $(2-19)$ entry: $C_{2,19}$
- $(2-20)$ entry: $C_{2,20}$
- $(2-65)$ entry: $(C_{2,65} + C_{2,104})$

The RHS of the $2-th$ equation system is

$$T_2 + C_{2,101}\,\tilde{u}^*_{n,101}\,d_{2,101} + C_{2,102}\,\tilde{u}^*_{n,102}\,d_{2,102} + C_{2,103}\,\tilde{u}^*_{n,103}\,d_{2,103} + \\ + C_{2,104}\,\tilde{u}^*_{n,104}\,d_{2,104} + C_{2,124}\,\psi_p\,\frac{d_{2,124}}{d_{2,p}} + C_{2,125}\,\psi_p\,\frac{d_{2,125}}{d_{2,p}} + \\ + C_{2,126}\,\psi_p\,\frac{d_{2,126}}{d_{2,p}} + C_{2,127}\,\psi_p\,\frac{d_{2,127}}{d_{2,p}} + C_{2,128}\,\psi_p\,\frac{d_{2,128}}{d_{2,p}}$$

where $\tilde{u}^*_{n,m} = \frac{1}{\Delta t}\,(u^{k+1}_n + u^*_n)\big|_{bm}$, m is the index of the *mirror* and bm is the boundary triangle from which m has been generated.

3.2.6 Flow chart of the *PANORMUS-SPH* code

The flow chart described in Sec. 2.10 must be modified introducing the *In/OutFlow-BCs* algorithm in the structure of the *PANORMUS-SPH* code, as shown in Fig. 3.11. Specifically, the *IO* particles are generated simultaneously to the *mirror* ones in *ACTIONS* 4, 13 and 16. The *PPE* system is built (*ACTION* 8) including the *IO* particles (as explained in Sec. 3.2.5). After moving the *effective* particles (*ACTION* 10), the particles having gone through outflow boundaries are deactivated in *ACTION* 11 (as discussed in Sec. 3.2.3), whilst the new particles are generated during *ACTION* 12. After updating the simulation time by one time step ($t = t+dt$) in *ACTION* 18, the procedure is restarted with the *predictor-step* (*ACTION* 7).

3.3 Benchmark test cases

Two test cases are presented here in order to show that the *In/OutFlow-BCs* procedure is able to guarantee the global mass conservation and the achievement of correct velocity profiles when the velocity distribution is unknown setting the *pressure BCs* only. The non *reflective properties* of the method will be shown in Sec. 6.3.1. Several application of the *incoming BCs* will be shown in Chap. 6.

The flow in a circular pipe has been considered. Steady *pressure BCs* are employed in the first test case, whilst an oscillatory flow is considered in the second one. A simple geometry is used to compare the numerical results with analytical solutions available both for steady and oscillatory flow.

The diameter D of the circular pipe is 0.1 m, while the fluid density and kinematic viscosity are 1000 kg/m^3 and 10^{-6} m^2/s, respectively.

3.3.1 Starting transient Poiseuille flow

The transient flow in a circular pipe, starting from the rest towards the achievement of the steady-state, is considered in this test case. The Fig. 3.12 shows the boundary conditions: *pressure* boundary conditions are imposed on the triangles in the inflow and outflow cross-sections, while *adherence* conditions are employed on the triangles of the pipe lateral surface.

The kh value is set to $D/25 = 0.004$ m, while the pipe length L is equal to the diameter, resulting in the initial number of 97 500 *effective* particles. The pressure gradient is $-2 \cdot 10^{-2}$ Pa/m, which is obtained imposing the kinematic pressure values of $\psi_A =$

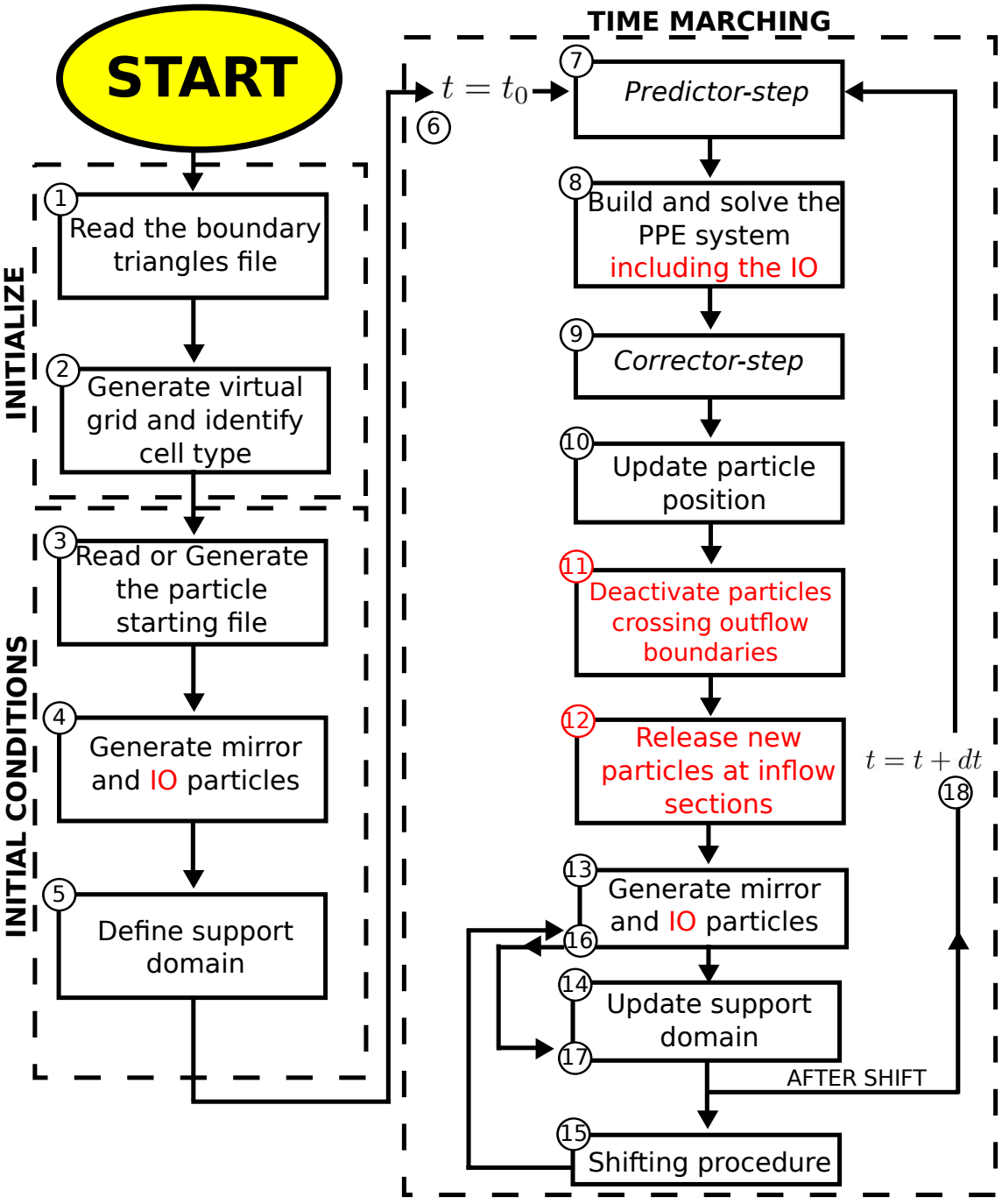
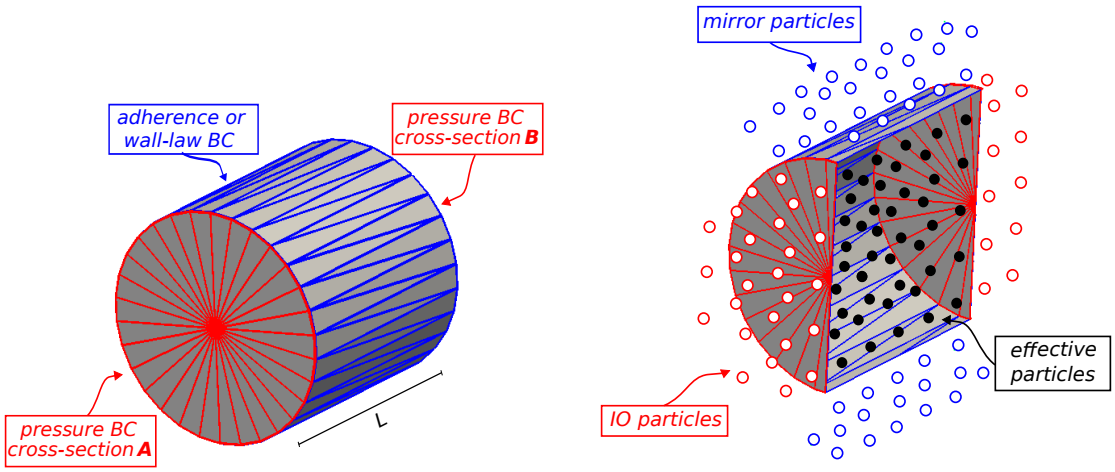


Figure 3.11: Flow chart of the *PANORMUS-SPH* code with the *In/OutFlow-BCs* procedure. The actions closely related to the *In/OutFlow-BCs* procedure are highlighted with the red color.



(a) (b)

Figure 3.12: *Benchmark test case* - Sec. 3.3. Sketch of the circular pipe. a) Boundary conditions: *adherence* conditions on the pipe lateral surface, *pressure* conditions on the triangles on the A and B cross-sections. b) Pipe longitudinal section. Full black circles: *effective* particles; red empty circles: *IO* particles; blue empty circles: *mirror* particles. Taken from: Monteleone et al. (2017), 15, fig. 6.

$2 \cdot 10^{-6}\ m^2/s^2$ and $\psi_B = 0$ in the inlet (A) and outlet (B) cross-sections, respectively. The resulting analytical cross-section averaged velocity is $V = 0.00625\ m/s$, corresponding to the *Reynolds number* $Re = 625$.

The numerical results during the transient regime before the steady-state are compared with the analytical solution of Szymanski (1932)

$$\tilde{u}\left(\tilde{r},\tilde{t}\right) = (1-\tilde{r}^2) - \sum_{n=1}^{+\infty} \frac{8}{(\alpha_n)^3 J_1(\alpha_n)} J_0(\alpha_n \tilde{r}) e^{-\alpha_n^2 \tilde{t}} \tag{3.7}$$

where $\tilde{u}$ is the streamwise velocity non-dimensionalized with the steady-state axial velocity, $\tilde{r} = 2r/D$ with r the distance from the pipe axis, $\tilde{t} = 4t\nu/D^2$ is the non-dimensional time, $J_0(x)$ and $J_1(x)$ are the Bessel functions of first kind and order 0 and 1, respectively, α_n are the roots of $J_0(\alpha_n)$ for $n = (1, 2, ...)$.

The Fig. 3.13 shows the non-dimensional axial velocity as a function of the non-dimensional time. It can be observed that the steady-state condition is achieved after about one non-dimensional time unit and a perfect agreement of the numerical axial velocity with the analytical solution (eqn. 3.7) for $\tilde{r} = 0$ is obtained during the whole simulation.

In Fig. 3.14 the numerical and analytical velocity profiles at several time steps are plotted, showing again a very good agreement of the obtained results with the analytical solution. The velocity profiles in the figure have been calculated considering one diameter only in the middle channel cross-section (no azimuthal or axial averaging is thus performed). In order to make available a more precise comparison of the numerical and analytical results, a Table with the *Coefficients of Variation* (ratio between the *Standard Deviation* $\sqrt{\sum_{i=1}^{N}(u_{NUM,i} - u_{AN,i})^2/N}$ and the mean velocity in the streamwise direction) at different time steps is included in Fig. 3.14. The Table shows that the errors reach a maximum values of 3.2%, to reduce to about 2% at the steady-state.

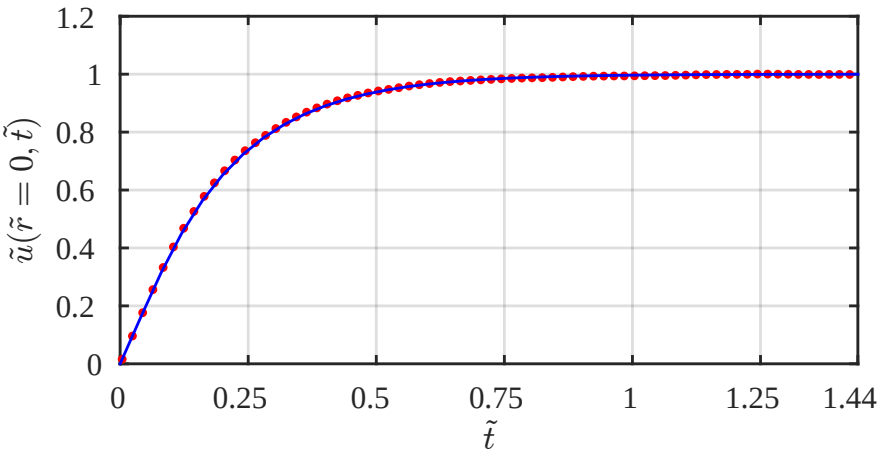
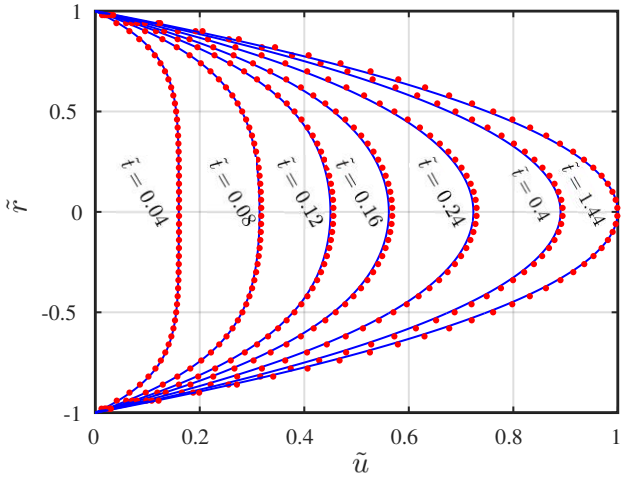


Figure 3.13: *Benchmark test case* - Sec. 3.3.1. Non-dimensional axial velocity ($r = 0$) of the starting transient *Poiseuille* flow as a function of $\tilde{t} = 4t\nu/D^2$. Blue continuous line: analytical solution (eqn. 3.7) of Szymanski (1932). Red dotted line: *SPH* numerical results. Taken from: Monteleone et al. (2017), 15, fig. 7.



$\tilde{t}$	$\widetilde{SD}$
0.04	0.009
0.08	0.014
0.12	0.019
0.16	0.023
0.24	0.028
0.40	0.032
1.00	0.028
1.44	0.020

Figure 3.14: *Benchmark test case* - Sec. 3.3.1. Velocity profiles at different time-steps across the pipe diameter. Line symbols as in Fig. 3.13. In the Table the *Coefficients of variation* are shown too, at time-steps corresponding to the velocity profiles. Taken from: Monteleone et al. (2017), 16, fig. 8.

An analysis of the velocity changes in the axial direction (see Fig. 3.15) showed that, coherently with the flow incompressibility and domain geometry, no velocity changes occurred in the axial direction.

The number of particles during the simulation remains almost constant, with changes lower than 0.05%, as it is shown in Fig. 3.16. Since the mass, volume and density of each particle in the *ISPH* algorithm are constant during the simulation, the conservation of the particle number implies also the domain mass conservation. The *In/OutFlow-BCs* procedure to account for inflow and outflow boundary cross-sections is thus able to guarantee a correct mass conservation while new particles are continuously introduced in the computational domain and other particles leave it through the outlet.

3.3.2 Pulsating flow in a circular pipe

An oscillating pressure is assigned at the A cross-section of the pipe shown in Fig. 3.12) according to the sinusoidal function

$$\psi_A = 2 \cdot 10^{-6} \sin(\omega t) \quad [\mathrm{m}^2/\mathrm{s}^2]$$

with $\omega = 2\pi/T$ and $T = 360$ s. In the section B of the pipe the constant null pressure value is imposed. Differently from the previous case, thus, the pressure gradient oscillates between positive and negative values, so that the fluid can enter or leave the domain through both the cross-sections A and B. As shown in Fig. 3.17, at some time instants in each of the cross-sections A and B an inlet and an outlet portion can be identified. The described procedure with *pressure BCs* does not need *a priori* identification of inlet and outlet boundaries that are treated in the same way, allowing also to take into account cross-sections with mixed inflow/outflow conditions.

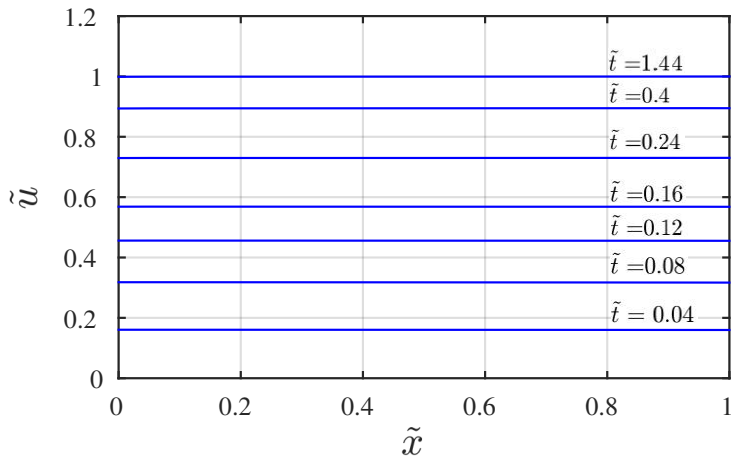


Figure 3.15: *Benchmark test case* - Sec. 3.3.1. Non-dimensional axial velocity along the non-dimensional direction ($\tilde{x} = x/L$) of the starting transient *Poiseuille* flow at different times ($\tilde{t} = 4t\nu/D^2$).

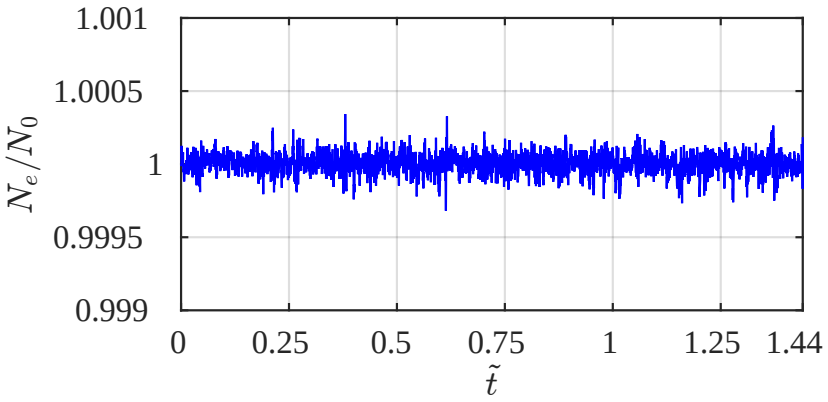


Figure 3.16: *Benchmark test case* - Sec. 3.3.1. Ratio between the number N_e of the *effective* particles in the computational domain and the initial number N_0 during the simulation. Taken from: Monteleone et al. (2017), 16, fig. 9.

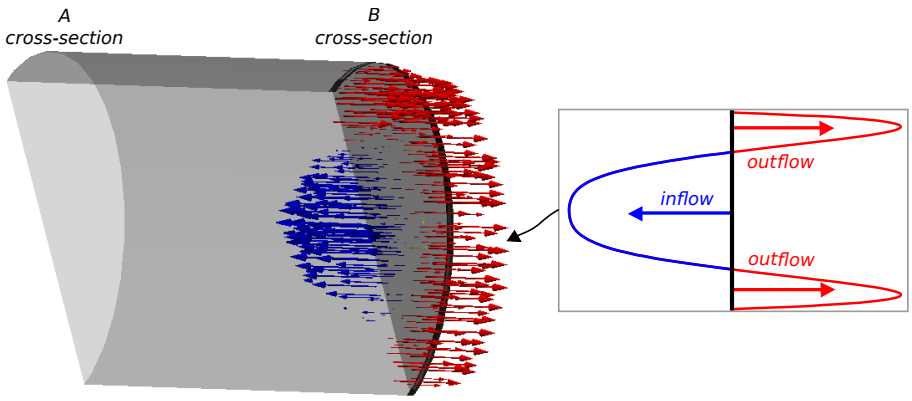


Figure 3.17: *Benchmark test case* - Sec. 3.3.2. Velocity profile with mixed positive and negative values, corresponding at the B cross-section to simultaneous inflow (blue arrows) and outflow (red arrows) conditions (with the opposite holding for the cross-section A). Taken from: Monteleone et al. (2017), 14, fig. 4.

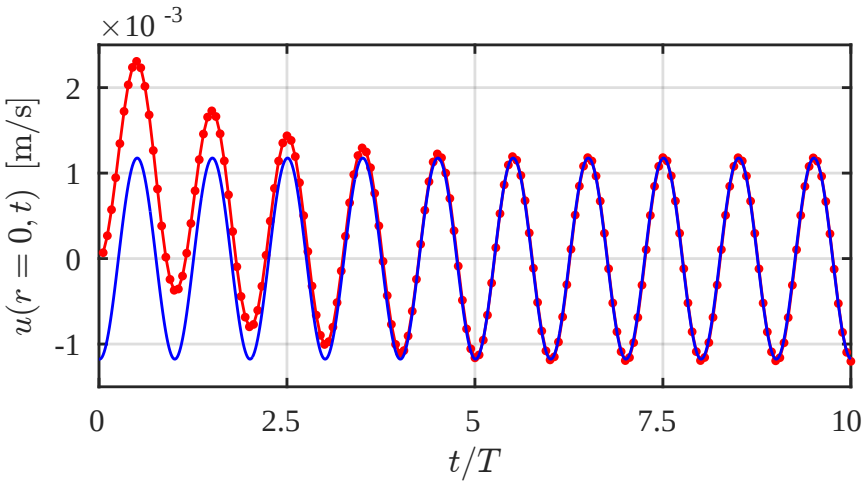


Figure 3.18: *Benchmark test case* - Sec. 3.3.2. Axial velocity ($r = 0$) as a function of the non-dimensional time t/T. Comparison between the analytical solution of Womersley (1955) (continuous blue line) and the numerical results (dotted red line). Taken from: Monteleone et al. (2017), 17, fig. 10.

The simulations are carried out for 10 periods. In order to reduce the computational time, the kh value is set to 0.006 m, 1.5 times larger than in the previous test case. The resulting total number of *effective* particles is equal to 60 000.

The numerical results can be compared with the analytical solution of Womersley (1955), where the pressure gradient is assigned through the general periodic function

$$\frac{\partial p}{\partial x} = \sum_{n=-\infty}^{\infty} C_n e^{in\omega t} \tag{3.8}$$

The resulting time-dependent velocity profile is

$$u(r,t) = \sum_{n=-\infty}^{\infty} \frac{C_n}{i\rho n\omega} \left[1 - \frac{J_0(i^{3/2}\alpha\, n^{1/2}\, 2r/D)}{J_0(i^{3/2}\, n^{1/2}\, \alpha)}\right] \cdot e^{in\omega t} \tag{3.9}$$

where α is the *Womersley number* equal to $D/2\sqrt{\omega/\nu}$ (Womersley, 1955) and i is the imaginary number.

In the considered test case, in order to obtain the sinusoidal function

$$\partial p/\partial x = -2 \cdot 10^{-2} \sin(\omega t)$$

the coefficients C_n are set to zero, with the exception of $C_{-1} = -i \cdot 10^{-2}$ and $C_1 = i \cdot 10^{-2}$.

In Fig. 3.18 the numerical velocities at $r = 0$ are plotted for 10 periods, using the analytical formula (eqn. 3.9) for comparison. As it is seen in the figure, after 5 periods there is a perfect agreement of the numerical results with the analytical one. It should be noticed that since the numerical simulation is conducted starting from the rest, while the initial axial velocity according to eqn. 3.9 is equal to $1.176 \cdot 10^{-3}$ m/s, some periods are required before achieving the regime oscillatory pattern. However, during the first 5 periods, despite the amplitude disagreement, the numerical and analytical velocities are in phase for each time step.

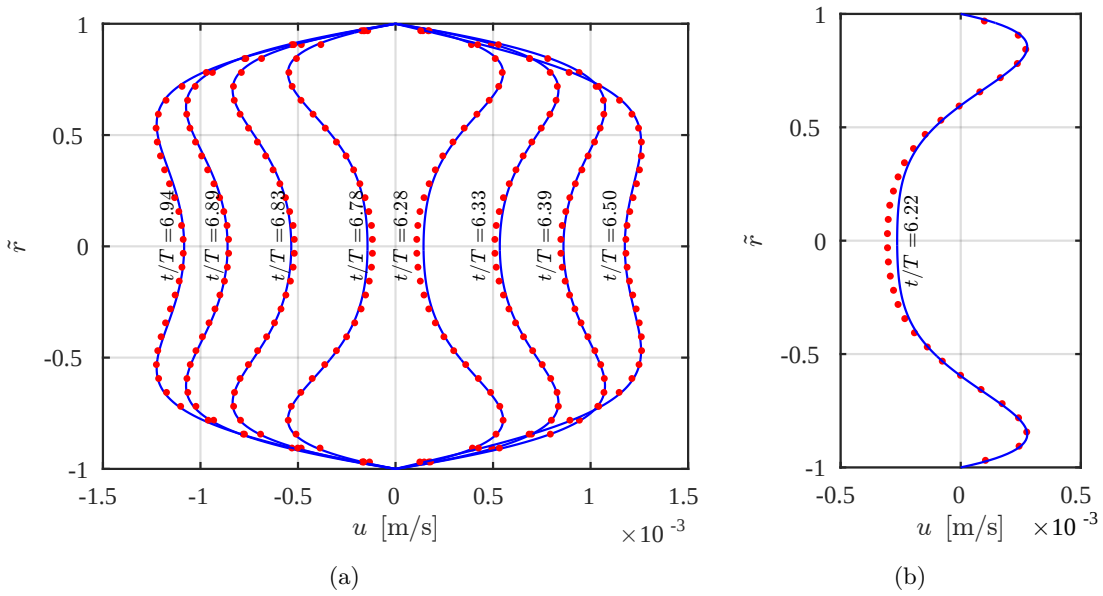


Figure 3.19: *Benchmark test case* - Sec. 3.3.2. Velocity profiles at different time steps during the 6-th oscillation period. Continuous blue line: analytical solution of Womersley (1955); dotted red line: numerical results. a) Profiles at time steps with positive or negative velocities in the whole cross-section. b) Profile with simultaneous positive and negative velocities in the cross-section. Taken from: Monteleone et al. (2017), 17, fig. 11.

In Fig. 3.19 the velocity profiles are plotted at different time steps during the 6-th oscillation period. The profiles in Fig. 3.19.a with $u > 0$ correspond to time steps in which the fluid flows comes from the cross-section A (inflow) towards B (outflow), while the opposite occurs for velocities $u < 0$, when A and B become the outlet and inlet cross-sections, respectively. The profile in Fig. 3.19.b ($t/T = 6.22$) refers on the other hand to a time instant in which inflow and outflow conditions simultaneously occur in the same cross-section.

The Fig. 3.20 shows the kinematic pressure gradient $\partial(p/\rho)/\partial x$ as a function of the dimensionless time t/T. The results have been compared with he analytical solution (Womersley, 1955) plotted in the same figure. The kinematic pressure gradients have been obtained averaging at each time step the values along the pipe axis at $x = L/4, L/2$ and $3/4L$. In the figure it is clearly seen that the numerical kinematic pressure gradients are in very good agreement with the analytical values, with a percentage error equal to 1.5% with respect to the analytical oscillation amplitude.

During the simulation the mass conservation is reasonably guaranteed, since the changes in the number of *effective* particles are limited to 3% after the simulated 10 periods.

This result is quite satisfactory considering that, due to the limited pipe length and the involved velocities, during the simulation each particle was deactivated in the average about 20 times after having left the computational domain through the outflow section, to be subsequently released as a new particle at the inflow. In order to remove the domain length effect from the analysis of the conservation properties, it is useful to compare the number of particles lost in each cycle (equal to about 90 in the average in the simulations) with the total number of the ones going through the inflow and outflow sections in the same

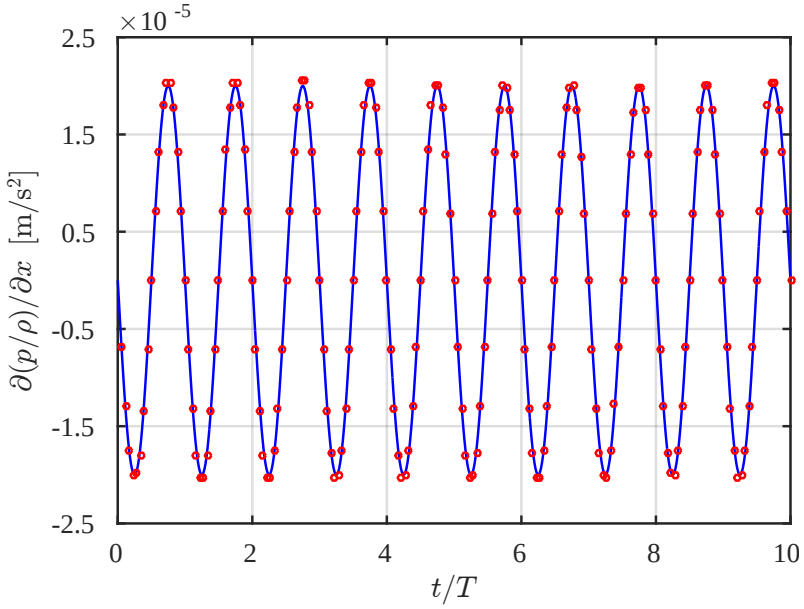


Figure 3.20: *Benchmark test case* - Sec. 3.3.2. Kinematic pressure gradient as a function of the dimensionless time t/T (numerical values obtained averaging the gradients in three points along the pipe axis at $x = L/4, L/2$ and $3/4L$). Blue continuous line: analytical solution of Womersley (1955); red dotted line: numerical results. Taken from: Monteleone et al. (2017), 18, fig. 12.

time period. This comparison shows that, although no direct relation between the number of particles leaving and entering the domain is enforced in the procedure, a negligible loss of particles has been observed in the simulations.

Chapter 4

The *Multi-Domain* approach

In this chapter a procedure is described aimed at improving the discretization refinement in *SPH* while not overloading the computation. The procedure has been published in Monteleone et al. (2018).

The method allows to partition the computational domain in subdomains (or blocks), in each of which the *smoothing length* of the *kernel function* is maintained constant while changing between blocks where a different resolution is required. The *domain decomposition* technique, the numerical procedure to match the solution between neighboring subdomains as well as the algorithms to release and to delete particles entering and leaving the blocks are explained.

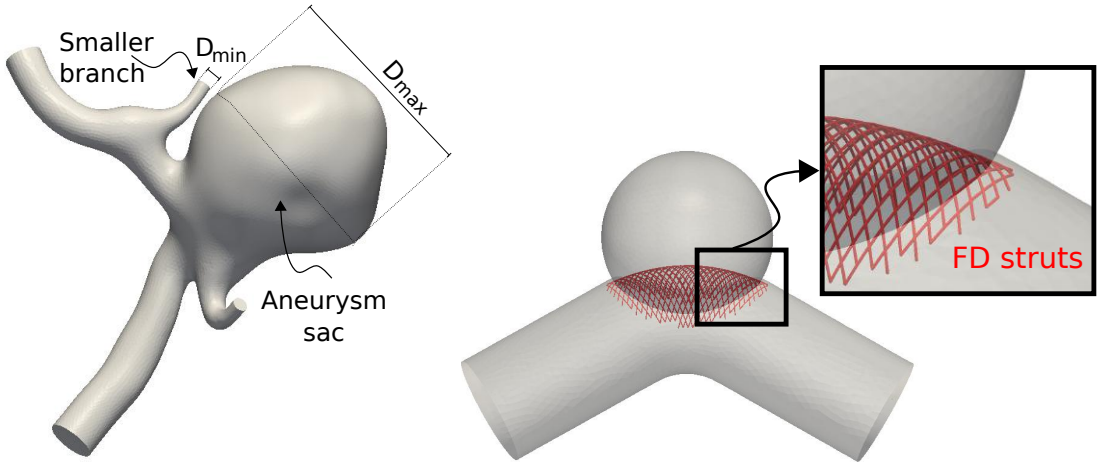
Two test cases are presented to show the efficiency and accuracy of the method and its ability to strongly reduce the computational efforts: the $3D$ unsteady channel flow in a cylindrical pipe and the $2D$ vortex shedding in the wake of a circular cylinder.

4.1 Background and motivations

In the *SPH* method, the accuracy of the computation is directly related to the *smoothing length* h. In order to obtain high quality solutions, a reasonably high number of particles must be contained in each particle *support domain*, maintaining a relatively regular space distribution during the time evolution of the simulation. The number of particles representing the computational domain (N_e) depends on their isotropic initial distance Δx, which is proportional to the *smoothing length* h. In $3D$ computations, $N_e \propto h^3$.

In *mesh-based* methods it is quite straightforward to reduce the computational efforts realizing non uniform grids, being stretched and/or clustered close to external or internal boundaries and in regions of the computational domains with high gradients of the hydrodynamic variables. On the contrary, in the "classical" *SPH* approach the *smoothing length* is uniform in space due to the difficulty of changing the width of the *kernel function* while the particles move from one region to another. The *SPH* computational efforts are thus very high, since the value of the *smoothing length* must be chosen according to the one imposed by the regions requiring the finest discretization. The same computational overload would be undergone by *grid-based* methods employing in the whole domain cubic cells with constant size (selected accordingly to the finest required width).

This is a very relevant issue when the domain geometry is characterized by regions with very different size such as the heterogeneous geometry of the *CA* with the parent vessel and the surrounding branches (see Fig. 4.1.a) and even more *CAs* treated with *endovascular* devices such as *flow diverter* (as shown in Fig. 4.1.b). Regarding the latter case, it should



(a) (b)

Figure 4.1: a) Cerebral vessel with small branches and a giant aneurysm. Taken from: Aneurisk-Team (2012). $D_{min} = 1\ mm$ is the diameter of the smaller branch whilst $D_{max} = 10\ mm$ is the ellipsoid maximum axis of the aneurysm sac; b) ideal aneurysm with *flow diverter* (*FD*) device. From: Prof. Frangi's research group (University of Sheffield).

be noted that blood flow through *FD* is difficult to simulate using the conventional *grid-based* methods as well, due to the very large difference scale between the size of the *FD* struts, the parent vessel and the aneurysm, as discussed in Jeong and Rhee (2012). To overcome this problem, some adaptive embedding techniques have been developed in *grid-based* framework (Appanaboyina et al., 2008; Cebral and Lohner, 2005) or an alternative strategy based on the modeling of the device as a porous medium (Augsburger et al., 2011).

In the *SPH* method, adopting a constant h value in the whole domain would imply the use of a huge number of particles, with a resolution exceedingly high in most of the domain. Considering the example in Fig. 4.1.a, the h value should be chosen according to the smallest vessel (whose diameter, D_{min}, is indicated in the figure) in order to obtain a sufficient number of *effective* particles. Thereby, a very high and unnecessary number of particles would be placed in the aneurysm sac, whose ellipsoid maximum axis is ten times that of the smallest branch.

In order to increase the computational efficiency of the *SPH* method, several refinement strategies have been proposed using a *smoothing length* variable in space (Feldman and Bonet, 2007; López et al., 2013). Due to the Lagrangian nature of the method, these approaches require introducing splitting and coalescing techniques for the particles, since their dimension and *support domain* must be adapted to the space dependent h (Xiong et al., 2013; Vacondio et al., 2013a; Vacondio et al., 2013b; Spreng et al., 2014; Vacondio et al., 2016; Hu et al., 2017). In this framework, Barcarolo et al. (2014) proposed a procedure in the *WCSPH* scheme based on the coupling of a particle refinement technique with an innovative particle derefinement strategy. Specifically, based on a spatial refinement criterion, several regions of the domain with a different refinement level are identified. In the refinement process, following the scheme of Feldman and Bonet (2007) and its improved version proposed by López et al. (2013), a splitting technique is used: when a bigger particle ("*mother*" particle) enters a region with a higher level of refinement it is divided into a finite number of smaller particles ("*daughter*" particles). The refinement is achieved by choosing some refinement parameters (the distance between *daughter* particles,

the radius length of the *daughter* particles with respect to the *mother* one, the mass ratio between *daughter* and *mother* particles) through minimization of a local refinement error based on the estimation of the gradient of the density function. The mass of the *mother* particles as well as the kinetic energy, the linear momentum and the angular momentum, have to match with the sum of the corresponding properties of the *daughter* particles. The *mother* particles, when entering a refinement region, are not used to compute the *SPH* operators but are kept during the simulation and passively advanced in time with the flow. When leaving the refinement zone, the *mother* particles are activated again. On the other hand, *daughter* particles are simply erased when leaving the refinement zone. Moreover, in order to avoid pressure discontinuities occurring when a *mother* particle enters (or leaves) a refinement domain, a transition region is defined between the unrefined and the refined zones where *mother* and *daughter* particles are progressively deactivated (but not erased) and activated, respectively. This procedure, that was validated for $2D$ simulations only, allows to straightforwardly switch back to the derefined distribution whenever required. Although several levels of refinement can be used, the choice of the order of the refinement levels and of the values Δx to be used has some constraints. In fact, the *mother* particles, which passively pass through the regions with a finer discretization, must enter a derefinement region with their own original value Δx in order to be again activated.

A different method based on multi-domain decomposition is proposed here which relies on the partitioning of the domain in several subdomains (or blocks) each of which has its own value of the *smoothing length*. Differently from the previously mentioned techniques, this method does not require splitting or coalescing strategies and in each block the simplicity of the classical *SPH* numerical scheme with constant h is maintained. As it will be shown in the $3D$ geometrically complex domain of Fig. 4.2, the method allows to use whatever level of refinement without the need to switch from h, $h/2$, $h/3, ...$ (which was a limit for the procedure of Barcarolo et al. (2014) as discussed before). Moreover, differently from Shibata et al. (2017) which proposed a multi-resolution technique where the whole computational domain is represented with partially overlapping subdomains (with their own spatial resolutions and particle shape) and differently from Barcarolo et al. (2014), in the procedure presented here no overlapping of the subdomains is employed, thus avoiding any artificial increase of the computational domain.

In the following the method will be named *Multi-Domain* (*MD*) approach, whilst the epithet *Single-Domain* (*SD*) will be used to indicate the classical *SPH* method with a constant *smoothing length*. The *Multi-Domain* approach is described also in the recently published paper of Monteleone et al. (2018).

4.2 The multi-domain procedure

The *Multi-Domain* technique can be subdivided in the following items:

- *Domain decomposition* technique;
- generation of the *interface* particles;
- solution matching at the *block interfaces* for the velocities and the ψ values;
- inflow/outflow procedure through the *block interfaces*.

4.2.1 Domain decomposition

As discussed above, the computational domain is partitioned in blocks without overlapping regions in order to adapt the *SPH* method to the spatial resolution required in each of the selected subregions. The Fig. 4.2 shows the subdomains obtained by the decomposition of the aneurysm in Fig. 4.1.a. The blocks are separated by plane or curved surfaces named *block interfaces* (or simply *interfaces*). These separation surfaces are discretized into triangles as explained in Sec. 2.5 and Sec. 3.2 for solid wall and open boundaries, respectively.

The *smoothing length* h and the *starting particle distance* Δx are maintained constant inside each block, while different values are assigned to the particles contained in different blocks. As a consequence, the classical *SPH* formulation (discussed in Sec. 2.1) can be used inside each block, although specific procedures have been implemented to account for the proper treatment of near-interface regions. Specifically, since in these areas the *support domain* of the particles can be truncated by the *block interfaces*, additional *interface particles* (indicated as *IP*) are added in the neighboring subdomain. The *IP* particles play an important role in order to obtain a suitable matching of the solution in neighboring blocks, as will be discussed Sec. 4.2.3.

The total number of *effective* particles of the whole computational domain $N_{e,tot}$ is the sum of the *effective* particles contained in all the blocks

$$N_{e,tot} = \sum_{n=1}^{N_{Blocks}} N_{e,B_n} \tag{4.1}$$

where N_{Blocks} is the total number of blocks and N_{e,B_n} is the number of *effective* particle in the block B_n.

A boundary triangles file as well as the initial particle distribution and the virtual grid must be created for each block. For example, considering a generic block named "A" the grid properties x_{0A} and nx_A, ny_A, nz_A can be obtained through eqns. 2.13 using the *smoothing length* h_A and the boundary vertices having lowest coordinates among the triangles of the block A.

4.2.2 IP generation

The *IP* particles are generated from the *effective* particles having distance shorter than Δx from one of the *block interfaces*.

As discussed for the generation of the *IO* particles (see Sec. 3.2.1), these *effective* particles generate "n" (where "n" is the integer part of the ratio $kh/\Delta x$) *interface* particles in the direction normal to the *block interface* in order to reach the contour of their *support domain*. Since in this research study the ratio $kh/\Delta x$ has been set to 2 (see Chap. 2), for each of these *effective* particles two IP particles are generated at distance equal to Δx and $2\Delta x$, respectively. As explained for the *IO* generation, three IP particles would have been generated if the most common ratio $kh/\Delta x = 2.66$ had been employed. As a consequence, an increase of the computational efforts of the *Multi-Domain* procedure would have occurred.

The Fig. 4.3.a shows a computational domain partitioned in two blocks: A and B. For the sake of clarity only the *effective* and the IP particles of block A are represented and a bi-dimensional sketch is considered, where the *triangle interfaces* are represented by a segment (bold red line in the figure). The particle S of block A, having distance from

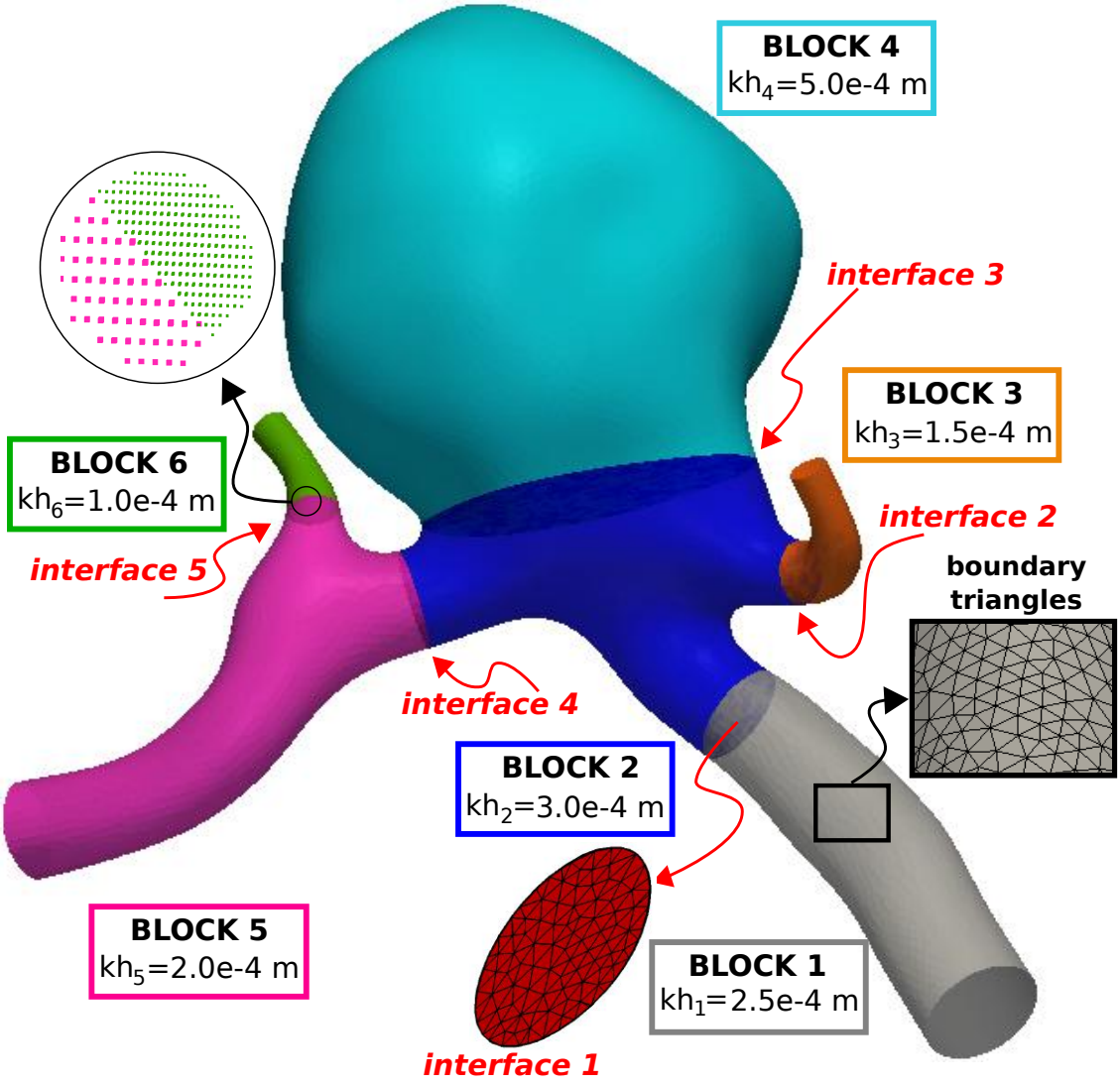
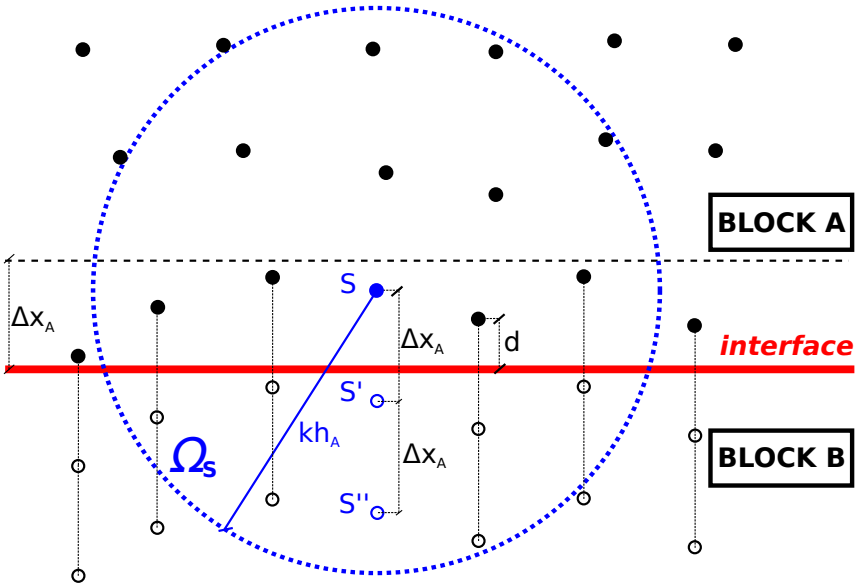
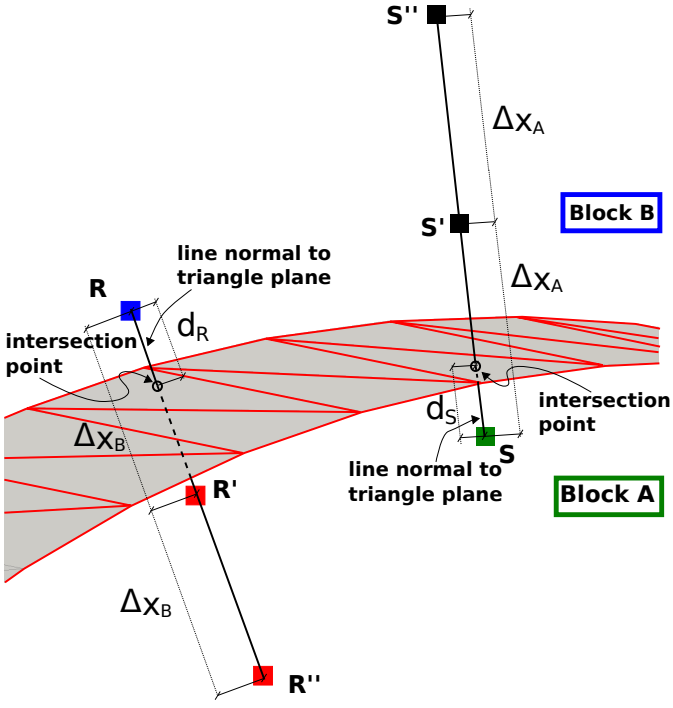


Figure 4.2: Sketch of the subdivision into 6 blocks of the aneurysm of Fig. 4.1.a. The external surfaces and the *block interfaces* are discretized into triangles (e.g., the rectangular gray area in block 1 and the elliptical red area at the *interface* between blocks 1 and 2). The change in the particle initial distance is visible in the enlargement inside the circular black line in the vicinity of the *interface* between blocks 5 and 6. Taken from: Monteleone et al. (2018), 960, fig. 2.

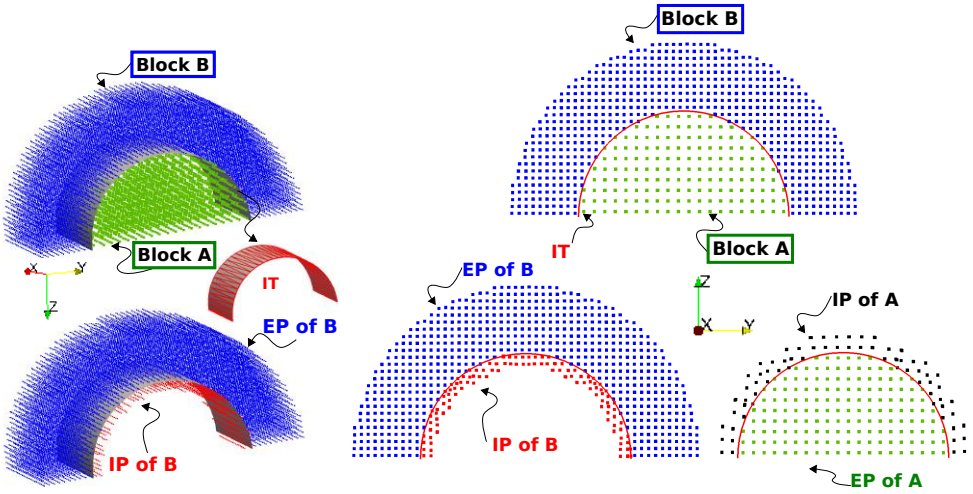


(a)



(b)

Figure 4.3: Sketch of the *IP* particle generation. a) $2D$ scheme where the *effective particles* of block A (full black circles) having distance $d < \Delta x_A$ from the *block interface* (bold red line) generate *IP* particles (empty black circles). The dotted blue circle around the *effective particle* S indicates the *support domain* S, containing the particles S' and S''; b) $3D$ scheme, where a curved *block interface* is used. *Effective* R and S particles generate two IP particles each in the neighboring subdomain in the direction normal to the *interface triangle*. Taken from: Monteleone et al. (2018), 961, fig. 3.



(a) (b)

Figure 4.4: Sketch of the domain decomposition through curve *block interface*. EP: *effective* particles; IP: *interface* particles; IT: *interface* triangles; Block A: green particles; Block B: blue particles. a) $3D$ sketch; b) $2D$ sketch. Taken from: Monteleone et al. (2018), 961, fig. 4.

the *interface* shorter than Δx_A, generates two IP particles in the direction normal to the *interface* (S' and S''). The two IP particles S' and S'' have distance from the generating particle S equal to Δx_A and $2\Delta x_A$, respectively. These particles are contained in the neighboring block B. In Fig. 4.3.b a $3D$ scheme of the IP generation from both blocks A and B is shown where the *block interface* is a curved surface. In the figure the *effective* particles S of block A and R of block B are considered. Since S and R have distance from the relative triangle planes (dR and dS) shorter than Δx_B and Δx_A, respectively, two IP particles are generated from each of them. The lines normal to the triangle planes are identified, allowing to generate the IP particles R' and R'' and the IP particles S' and S'' having distance from the corresponding *effective* particles equal to once and twice the *starting particle distances* (Δx_A, for block A and Δx_B for block B).

As discussed above, in the procedure no overlapping region is created between neighboring subdomains, which are entirely separated. Nevertheless, since the *effective* particles of a block generate, through the *block interface*, IP particles lying in the neighboring subdomain, an overlapping region is created where *effective* particles of a block coexist with IP particles generated by the neighboring block. The Fig. 4.4 shows two subdomains A (green *effective* particles) and B (blue *effective* particles) separated by a *block interface* which is represented by red triangles in Fig. 4.4.a ($3D$ view) and is plotted as a red line in Fig. 4.4.b ($2D$ view). No *effective* particles of block B (blue points in the figure) are contained in block A (which is filled with green particles) and *viceversa*, since the blocks are separated. On the contrary, the IP particles generated by the *effective* particles of block B (red points in the figure) are contained inside block A, while on the other hand the IP particles generated by block A (black points) are contained inside block B.

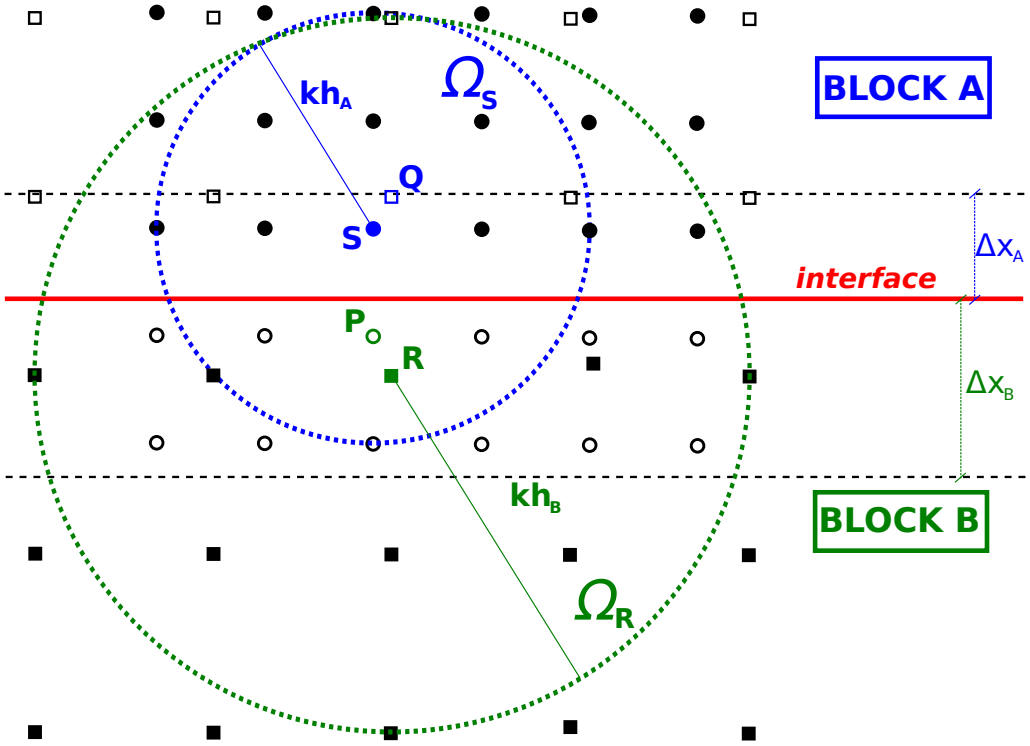


Figure 4.5: Sketch of the *interface* particle distribution. Full and empty black circles: *effective* and IP particles of block A; full and empty black squares: *effective* and IP particles of block B; bold red line: *block interface*; dotted blue and green lines: *support domains* Ω_S and Ω_R of particles S and R belonging to block A and B, respectively. Taken from: Monteleone et al. (2018), 962, fig. 5.

4.2.3 The solution matching at the block interfaces

The hydrodynamic values f (such as *intermediate* and *corrected* velocity and potential ψ) of the IP particles generated by a block are obtained through an interpolation starting from the *effective* particles of the block in which they are contained. As shown in Fig. 4.5, the IP particles of block A neighboring block B are contained inside B. Their hydrodynamic properties can be thus obtained through a Taylor series expansion around the closest *effective* particle of block B. Using the symbols P and R to indicate an IP particle of block A and its closest *effective* particle of block B, respectively, the interpolation can be written as

$$f_P^A = f_R^B + \left[\sum_{j=1}^{N_R} \frac{m_j}{\rho_j}(f_j^B - f_R^B)\ \nabla W_{Rj}\right] \cdot (\mathbf{x}_P - \mathbf{x}_R) \tag{4.2}$$

where the superscripts A and B are used to indicate particles of blocks A and B, respectively, the sum is extended to the N_R particles inside the *support domain* of R (Ω_R) with radius kh_B (dotted green line) and the expansion is truncated at first order.

In the same way, the interpolation for the IP particle Q of block B through the closest effective particle S of block B is

$$f_Q^B = f_S^A + \left[\sum_{j=1}^{N_S} \frac{m_j}{\rho_j}(f_j^A - f_S^A)\ \nabla W_{Sj}\right] \cdot (\mathbf{x_Q} - \mathbf{x_S}) \tag{4.3}$$

where the sum is extended to the N_S particles inside the *support domain* of S (Ω_S) with radius kh_A (dotted blue line in the figure).

Solution matching at block interface for the velocities

As it can be seen in the Fig. 4.5, the *support domains* Ω_R and Ω_S contain both *effective* (full squares and circles in the figure, respectively) and *IP* (empty squares and circles in the figure, respectively) particles. Eqns. 4.2 and 4.3 can be rewritten as

$$u_P^A = u_R^B + \sum_{j=1}^{N_R^e} C'_{pr}(u_j^B - u_R^B) + \sum_{j=1}^{N_R^{IP}} C'_{pr}(u_j^B - u_R^B) \tag{4.4}$$

$$u_Q^B = u_S^A + \sum_{j=1}^{N_S^e} C'_{qs}(u_j^A - u_S^A) + \sum_{j=1}^{N_S^{IP}} C'_{qs}(u_j^A - u_S^A) \tag{4.5}$$

with

$$\begin{aligned} C'_{pr} &= \frac{m_j}{\rho_j} \nabla W_{Rj} \cdot (\mathbf{x}_P - \mathbf{x}_R) \\ C'_{qs} &= \frac{m_j}{\rho_j} \nabla W_{Sj} \cdot (\mathbf{x}_Q - \mathbf{x}_S) \end{aligned} \tag{4.6}$$

where the variable f has been substituted with the m-th component of the velocity (*intermediate* or *corrected*) $u(m)$ simply indicated as u, N_R^e and N_R^{IP} are the *effective* and *IP* particles in Ω_R, respectively (obviously it is the same for N_Q^e and N_Q^{IP} refers to Ω_Q with respect to eqn. 4.5). This separation is useful since the values u_j in eqns. 4.4 and 4.5 are known if j is an *effective* particle (first summation) and are unknowns if j is an *IP* particles (second summation). In the latter case the values can be obtained using corresponding Taylor series expansions around the closest *effective* particles of block A and block B, respectively.

As a consequence, the eqns. 4.4 and 4.5 relative to the neighboring A and B blocks must be solved as a system containing one equation for each *IP* particle of the blocks

$$\begin{aligned} u_P^A - \sum_{j=1}^{N_R^{IP}} C'_{pr} u_j^B &= RHS_P \quad P = 1, \cdots N_{IP}^A \\ u_Q^B - \sum_{j=1}^{N_S^{IP}} C'_{qs} u_j^A &= RHS_Q \quad Q = 1, \cdots N_{IP}^B \end{aligned} \tag{4.7}$$

where N_{IP}^A and N_{IP}^B are the numbers of *IP* particles in the blocks A and B, respectively. The *right-hand-side* terms RHS_P and RHS_Q are

$$\begin{aligned} RHS_P &= u_R^B + \sum_{j=1}^{N_R^e} C'_{pr}(u_j^B - u_R^B) - \sum_{j=1}^{N_R^{IP}} C'_{pr} u_R^B \\ RHS_Q &= u_S^A + \sum_{j=1}^{N_S^e} C'_{qs}(u_j^A - u_S^A) - \sum_{j=1}^{N_S^{IP}} C'_{qs} u_S^A \end{aligned}$$

The equation system (4.7) is solved at each *block interface* using the *Pre-BiCGSTAB* method (explained in Sec. 2.7.1, see ALGORITHM 2.3). In order to obtain velocity vectorial values the system must be solved at each *block interface* once for each component, using the same coefficient matrix and updating the *right-hand-side* only.

The system of the interface l is made of N_{IP}^{l} equations, where N_{IP}^{l} is the total number of IP generated through l as the sum of the IP generated through the neighboring blocks A and B (N_{IP}^{lA} and N_{IP}^{lB}, respectively). Considering the component m of the velocity (*intermediate* or *corrected*) $u(m)$ (indicated simply with u), the system for the interface l can be written with the algorithm shown and explained below.

ALGORITHM 4.1- *MD velocity equations for the interface l*

---

1. *do* $n = 1, N_{IP}^{l}$
2. $diag_n = 1$ (diagonal term)
3. $n \in A$ (n is an IP of block A for example)
4. Find the closest particle R in block B
5. $RHS_n = u_R$ (the velocity of R is a known term)
6. *do* $j = 1, N_R$ (cycle on the particles in Ω_R)
7. if j is *effective* or *mirror* then
   $RHS_n = RHS_n + C'_{nr}\,(u_j - u_R)$
8. if j is IP then
   $off\_diag_{(n,j)} = C'_{nr}$
   $RHS_n = RHS_n - C'_{nr}\,u_R$

---

where the coefficient is $C'_{nr} = \frac{m_j}{\rho_j}\,\nabla W_{Rj} \cdot (\mathbf{x}_n - \mathbf{x}_R)$.

1. The cycle is repeated on all the IP particles generated by blocks A and B through the *interface* l ($N_{IP}^{l} = N_{IP}^{lA} + N_{IP}^{lB}$);
2. The *diagonal* term of each row is equal to 1;
3. The block from which the IP particle n is generated must be identified. In the example n is generated through an *effective* particle of A ($n \in A$);
4. The closest *effective* particle to n belonging to the block B is identified. This particle is named R in the example;
5. The *right-and-side* term of the n-th equation is set to the velocity of the R particle (u_R);
6. The cycle on the particles j (*effective*, *mirror* or IP) lying in the *support domain* of R (Ω_R) is performed;
7. If j is an *effective* or *mirror* particle, its velocity (u_j) goes to the *right-and-side* of the n-th equation since it is a known term. If u is the *intermediate* velocity and j is a *mirror* particle, the velocity u_j is set equal to that of its generating particle;
8. If j is an IP particle, u_j is unknown and must be added in n-j *off-diagonal* entry of the system matrix, while the u_R value is added to the *right-and-side* of the n-th row.

Solution matching at block interface for the ψ values

Differently from the *MD* velocity system, the *pseudo-pressure* values of the *effective* particles are unknown thus implying that the *MD* system for the ψ must be solved simultaneously with the *PPE* (eqn. 2.21). Therefore, a global system made of the $N_{e,tot}$ *Pressure Poisson equations* and the N_{IP} interface particle Taylor series expansions must be solved, where $N_{e,tot}$ and N_{IP} are the sums of the *effective* and *interface* particles in the whole computational domain. The global system is solved with the *Pre-BiCGSTAB* method (see ALGORITHM 2.3).

Eqns. 4.2 and 4.3 can be rewritten as

$$\begin{aligned}
\psi_P^A - \psi_R^B + \sum_{j=1}^{N_R^*} C'_{pr}\left(\psi_R^B - \psi_j^B\right) + \sum_{j=1}^{N_R^{M_{(g\neq R)}}} C'_{pr}\left(\psi_R^B - \psi_g^B\right) = RHS_P \\
\psi_Q^B - \psi_S^A + \sum_{j=1}^{N_S^*} C'_{qs}\left(\psi_S^A - \psi_j^A\right) + \sum_{j=1}^{N_S^{M_{(g\neq S)}}} C'_{qs}\left(\psi_S^A - \psi_g^A\right) = RHS_Q
\end{aligned} \tag{4.8}$$

with

$$\begin{aligned}
RHS_P = \frac{1}{\Delta t}\sum_{j=1}^{N_R^{M_R}} C'_{pr}\left(u_n^{k+1} - u_n^*\right)\Big|_b d_{Rj} - \frac{1}{\Delta t}\sum_{j=1}^{N_R^{M_{(g\neq R)}}} C'_{pr}\left(u_n^{k+1} - u_n^*\right)\Big|_b d_{gj} \\
RHS_Q = \frac{1}{\Delta t}\sum_{j=1}^{N_Q^{M_Q}} C'_{qs}\left(u_n^{k+1} - u_n^*\right)\Big|_b d_{Sj} - \frac{1}{\Delta t}\sum_{j=1}^{N_S^{M_{(g\neq S)}}} C'_{qs}\left(u_n^{k+1} - u_n^*\right)\Big|_b d_{gj}
\end{aligned}$$

where N_R^* are the *effective* and IP particles in Ω_R, $N_R^{M_{(g\neq R)}}$ are the *mirror* particles in Ω_R not generated by R and $N_R^{M_R}$ are the *mirror* particles in Ω_R generated by R. The same definitions can be given to N_S^*, $N_S^{M_{(g\neq S)}}$ and $N_S^{M_S}$ replacing Ω_R with Ω_S. The boundary conditions for the *PPE* are set through eqn. 2.22, where the index g indicates the *effective* particle generating the *mirror* particles j lying in the *support domain* of R and d_{gj} is the distance between the generating particle g and the particle j. When j is a *mirror*, the summation is extended to the $N_R^{M_{(g\neq R)}}$ particles only, since considering the *mirror* particles generated by R and substituting eqn. 2.22 the difference $\psi_R - \psi_j$ reduces to $\left(u_n^{k+1} - u_n^*\right) d_{Rj}/\Delta t$, thus contributing only to the system *right-hand-side* (as explained in Sec. 2.7.1).

When the closest *effective* particle has in its *support domain IO pressure* particles (as defined in Chap. 3), eqns. 4.8 are modified through trivial algebra

$$\begin{aligned}
&\psi_P^A - \psi_R^B + \sum_{j=1}^{N_R^*} C'_{pr}\left(\psi_R^B - \psi_j^B\right) + \sum_{j=1}^{N_R^{M_{(g\neq R)}}} C'_{pr}\left(\psi_R^B - \psi_g^B\right) + \\
&+ \sum_{j=1}^{N^{IO_R}} C'_{pr}\psi_R^B\left(1 + \frac{d_{Rj} - d_{Rp}}{d_{Rp}}\right) + \sum_{j=1}^{N^{IO_{(g\neq R)}}} C'_{pr}\left(\psi_R^B + \psi_g^B\,\frac{d_{gj} - d_{gp}}{d_{gp}}\right) = \\
&= RHS_P
\end{aligned} \tag{4.9}$$

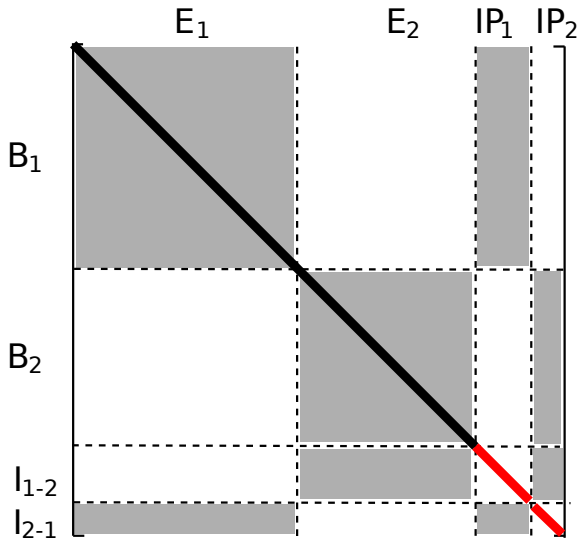


Figure 4.6: Scheme of the *MD* matrix global system for the ψ. B_1 and B_2 are the rows corresponding to equations of the *effective* particles of the blocks 1 (E_1) and 2 (E_2), respectively. I_{1-2} and I_{2-1} are the rows relative to the equations of the *interface* particles of block 1 (IP_1) and block 2 (IP_2), respectively.

with

$$RHS_P = \frac{1}{\Delta t}\sum_{j=1}^{N_R^{M_R}} C'_{pr}\left(u_n^{k+1}-u_n^*\right)\Big|_b d_{Rj} - \frac{1}{\Delta t}\sum_{j=1}^{N_R^{M_{(g\neq R)}}} C'_{pr}\left(u_n^{k+1}-u_n^*\right)\Big|_b d_{gj} + + \sum_{j=1}^{N_R^{IO}} C'_{pr}\psi_p\frac{d_{gj}}{d_{gp}}$$

where for brevity only the equation for the *interface* particle P of block A has been considered and eqn. 3.1 is used to express the ψ values of the IO *pressure* particles.

The blocks are numbered starting from 1 up to the total number of subdomains (N_{Blocks}). As discussed above, the global system is made of $N_{e,tot}$ equations of the *effective* particles (eqn. 2.21) following the order of the block number (first the equations of the block 1, then those of the block 2, etc..) and then the equations for the *interface* particles in order of block as well (first the equations of the IP generated by *effective* particles of block 2, then those generated by *effective* particles of block 2, etc..). With reference to the matrix scheme shown in Fig. 4.6, where two blocks (named 1 and 2) are considered:

- First the equations refer to the block B_1 are added (eqn. 2.21) whose number is equal to the number of *effective* particles of block 1 (E_1). In these equations the values of the *effective* and *interface* particles of block 1 are used that are thus highlighted with gray area while the bold black line indicates the *diagonal* terms. It should be noted that, for each equation, not all the particles inside the gray area are used but only these inside the *support domain* of the current particles. The coefficient matrix is thus sparse and for this reason the *CRS* format (explained in Sec. 2.7 for the only *PPE* system) is used;

- The equations refer to the second block (B_2) are added (eqn. 2.21) whose number correspond to the number of *effective* particles of block 2 (E_2). In these equations the values of the *effective* and *interface* particles of block 2 are used that are again colored in grey;

- The equations I_{1-2} of the *interface* particles generated by *effective* particles of the block 1 and thus contained in block 2 (indicated with IP_1 in the figure) are added. In these equations the values of the *effective* E_2 and *interface* particles of block 2 (IP_2) are used that are thus highlighted in grey, while the bold red line indicates the identity matrix. As explained in point 3 of the previous algorithm, the values in the *diagonal* are always 1, while the *off-diagonal* terms of this sub-matrix are null, since no IP particles of the same block are used. The IP_2 values are used since these particles can be contained in the *support domain* of the *effective* particles E_2;

- The equations I_{2-1} of the *interface* particles generated by *effective* particles of block 2 and thus contained in block 1 (indicated with IP_2 in the figure) are added. In these equations the values of the *effective* (E_1) and *interface* (IP_1) particles of block 1 are used, that are thus colored in grey, while the bold red line indicates the identity matrix. The IP_1 values are used since these particles can be inside the *support domain* of the *effective* particles E_1.

The i-th, with $i < N_{e,tot}$, row *diagonal* term of the system for the *pseudo-pressure* ψ is

$$\sum_{j=1}^{N_i'} C_{ij} + \sum_{j=1}^{N_i^{IP}} C_{ij} \tag{4.10}$$

where N_i^{IP} is the total number of the *interface* particles in Ω_i. Likewise, considering the IO *pressure* particles introduced in Sec. 3.2.5, two summations must be added to eqn. 4.10

$$\sum_{j=1}^{N_i^{IOi}} C_{ij} \frac{d_{ij} - d_{gp}}{d_{gp}} + \sum_{j=1}^{N_i^{IO}} C_{ij}$$

If $i > N_{e,tot}$ the *diagonal* term is equal to 1 (as discussed above).

The i-s *off-diagonal* entry with $i < N_{e,tot}$ of the system matrix is equal to eqn. 2.30 (or to eqn. 3.5 with IO *pressure* particles) if s is an *effective* particle, while it is equal to $-\delta_{is} C_{is}$ if s is an IP particle.

The i-s *off-diagonal* entry with $i > N_{e,tot}$ and s equal to the closest *effective* particle of i is

$$-1 + \sum_{j=1}^{N_S^*} C_{is}' + \sum_{j=1}^{N_S^{M(g \neq S)}} C_{is}' \tag{4.11}$$

While considering the IO *pressure* particles, the following two summations must be added to eqn. 4.11

$$\sum_{j=1}^{N_{IO_S}} C_{is}' \left(1 - \frac{d_{Sj} - d_{Sp}}{d_{Sp}}\right) + \sum_{j=1}^{N_{IO_{(g \neq S)}}} C_{is}'$$

where N_S^* is the total number of *effective*, IO and IP particles lying in Ω_S, $N_S^{M_{g\neq S}}$ is the number of *mirror* in Ω_S not generated by S, N_{IO_S} is the number of IO particles in Ω_S and $N_{IO_{(g\neq S)}}$ is the number of IO particles in Ω_S not generated by S.

If $i < N_{e,tot}$, the RHS of the i-th system (eqns. 2.31 and 3.6) does not change. Moreover, the IP particles are used to calculate the divergence of the *intermediate* velocity in eqn. 2.28.

If $i > N_{e,tot}$ the RHS of the i-th system is

$$\frac{1}{\Delta t}\sum_{j=1}^{N_S^{M_S}} C'_{is}\left(u_n^{k+1}-u_n^*\right)\Big|_b d_{Sj} - \frac{1}{\Delta t}\sum_{j=1}^{N_S^{M(g\neq S)}} C'_{is}\left(u_n^{k+1}-u_n^*\right)\Big|_b d_{gj} \tag{4.12}$$

where S is the closest *effective* particle to i. If IO *pressure* particles lie in Ω_S, eqn. 4.12 is modified adding the following summation

$$\sum_{j=1}^{N_S^{IO}} C'_{is}\psi_p\frac{d_{gj}}{d_{gp}}$$

The equations for the IP particles to be added to the PPE system are written using the algorithm below.

ALGORITHM 4.2- *Multi-Domain ψ equations*

---

1. *do* $l = 1, int_{tot}$
2. *do* $n = 1, N_{IP}^l$
3. $diag_n = 1$ (diagonal term)
4. $n \in A$ (n is an IP of block A for example)
5. Find the closest particle R in block B
6. $off\_diag_{(n,R)} = -1$ (the *psi* of R is an unknown term)
7. *do* $j = 1, N^R$ (cycle on the particles in Ω_R)
8. if j is *effective* or IP then
   $off\_diag_{(n,R)} = off\_diag_{(n,R)} + C'_{pr}$
   $off\_diag_{(n,j)} = off\_diag_{(n,j)} - C'_{pr}$
9. if j is *mirror* and $g = R$ then
   $RHS_n = RHS_n + C'_{pr}\left(u_n^{k+1}-u_n^*\right)\Big|_b d_{Rj}$
10. if j is *mirror* and $g \neq R$ then
    $RHS_n = RHS_n - C'_{pr}\left(u_n^{k+1}-u_n^*\right)\Big|_b d_{gj}$
    $off\_diag_{(n,R)} = off\_diag_{(n,R)} + C'_{pr}$
    $off\_diag_{(n,g)} = off\_diag_{(n,g)} - C'_{pr}$

11. if j is *IO pressure* then

$$RHS_n = RHS_n + C'_{pr}\,\psi_p\,\frac{d_{gj}}{d_{gp}}$$

12. if $g = R$ then

$$off\_diag_{(n,R)} = off\_diag_{(n,R)} + C'_{pr}\left(1 + \frac{d_{Rj} - d_{Rp}}{d_{Rp}}\right)$$

13. if $g \neq R$ then

$$off\_diag_{(n,R)} = off\_diag_{(n,R)} + C'_{pr}$$

$$off\_diag_{(n,g)} = off\_diag_{(n,g)} + C'_{pr}\frac{d_{gj} - d_{gp}}{d_{gp}}$$

---

1. The cycle on the all *interfaces* int_{tot} is performed;
2. The cycle on the IP particles generated through the *interface* l is performed;
3. The *diagonal* term of the n-th equation is set to 1;
4. It must be identified the block from which the IP particle n is generated. In the example n is generated through an *effective* particle of A ($n \in A$);
5. The *effective* particle closest to n is searched in the block B. This particle is named R in the algorithm above;
6. The value 1 must be added in the *n-j off-diagonal* entry of the system matrix due to ψ_R is unknown;
7. The cycle on the all particles lying in Ω_R is performed;
8. If j is an *effective* or IP particle, the values C'_{pr} and $-C'_{pr}$ are added in the *n-R* and *n-j off-diagonal* entries of the system matrix, respectively;
9. If j is a *mirror* particle generated by R, the *right-and-side* term corresponding to the n-th eqn. is increased by the value $C'_{pr}\,(u_n^{k+1} - u_n^*)\big|_b\,d_{Rj}$;
10. If the generating particle g of the *mirror* j is different from R, the *right-and-side* term of the n-th eqn. is decreased by $C'_{pr}\,(u_n^{k+1} - u_n^*)\big|_b\,d_{gj}$. The *n-R off-diagonal* entry of the system matrix is increased by the value C'_{pr}, while the *n-g off-diagonal* entry is decreased by the same quantity;
11. This point and the following points (12 and 13) of the algorithm are activated when the closest *effective* particle R has *IO pressure* particles in its *support domain*. Specifically, if j is an *IO pressure* particle the *right-and-side* term of the n-th eqn. is increased of the value $C'_{pr}\psi_p\,\frac{d_{gj}}{d_{gp}}$;
12. If R is also the generating particle of the *IO pressure* particle j, the *n-R off-diagonal* entry of the system matrix is increased by C'_{pr};
13. If the generating particle g is different from R, the *n-R off-diagonal* entry is increased by the quantity C'_{pr}, while the *n-g off-diagonal* entry is increased by the value $C'_{pr}\psi_p\,\frac{d_{gj}}{d_{gp}}$.

4.2.4 The inflow/outflow procedure through the block *interfaces*

The employment of a *multi-domain* approach in the framework of a Lagrangian method as *SPH* requires taking into account the movements of the particles from one block to another.

In the *Multi-Domain* approach the inflow and outflow procedures are handled separately using a technique similar to that explained in Sec. 3.2.2 and Sec. 3.2.3 with reference to the *In/OutFlow-BCs* technique. Specifically, at the end of each time step, after having calculated the particle velocities through eqn. 2.24 and having accordingly updated their position (eqn. 2.25), it is checked for each *effective* particle if it has gone through one of the *interface triangles*. In this case the *interface triangle* is considered an outflow for the block and the particle is simply removed from the list of particles of the block it comes from. *Open-boundaries* and internal *interfaces* connecting neighboring blocks are treated in the same way from this point of view, since in both cases the particle is leaving the block to which it belongs. As it is shown in Fig. 4.7.a, thus, a particle P leaving block A towards B is canceled at the next time step ($r+1$ in the figure). If at the end of the time step a particle approaches one of the *interfaces* without crossing it (particle Q in the figure), one or more IP particles are generated as it has been described in Sec. 4.2.2.

On the other hand, considering the scheme in Fig. 4.7.b, the entering of new particles in block B is handled as described in the following.

At the end of each time step, the region between the *starting particle distance* Δx and kh from the *interfaces* is considered (light green area in the figure). In order to verify if the generation of new particles is required into this region, for each *effective* particle contained in this checking area (S and T in the figure) and having positive velocity component in the direction normal to the *interface* (thus pointing towards the interior of the block), a conical *scan region* is identified, which is indicated with the yellow color in the $2D$ scheme shown in the figure. The vertex of the cone and the axis direction are assigned equal to the particle position and velocity direction, respectively, as shown in Fig. 4.7.b (particles S and T). The conical volume thus is placed on the upstream region contained between the checked particle and the *interface*. The region is used to verify if the movement of the considered particle towards the interior of the block is causing the development of an empty region, which would require the generation of a new particle to fill it. Thus, a new particle is released if no *effective* particle is found in the conical region, as occurring in the figure with reference to the cone with vertex in T. The new particle (R in the figure) is displaced at distance Δx from the cone vertex along the cone axis direction. If on the contrary some *effective* particles are found in the conical region (which occurs in the figure for the cone with vertex in S), no particle is generated inside the cone since no empty region has been identified.

In order to control the frequency of new particle release, the cone amplitude β is dynamically adjusted at each time step as discussed for the release of new particles at the inlet (see Sec. 3.2.2). Specifically, the widening of the cone increases the probability to find *effective* particles inside the conical region and thus reduces the frequency of release. Therefore, the angle β is increased by a fixed amount (1° in our test cases, starting from the initial value of 30°) when the total number of *effective* particles in the block becomes higher than the starting number and it is reduced by the same amount in the opposite case. The particles removed from the computation after having left a block through the *interface* are saved in a *storage* list, from which they are collected when new particles have to be released. The continuous increase of the number of existing particles is thus

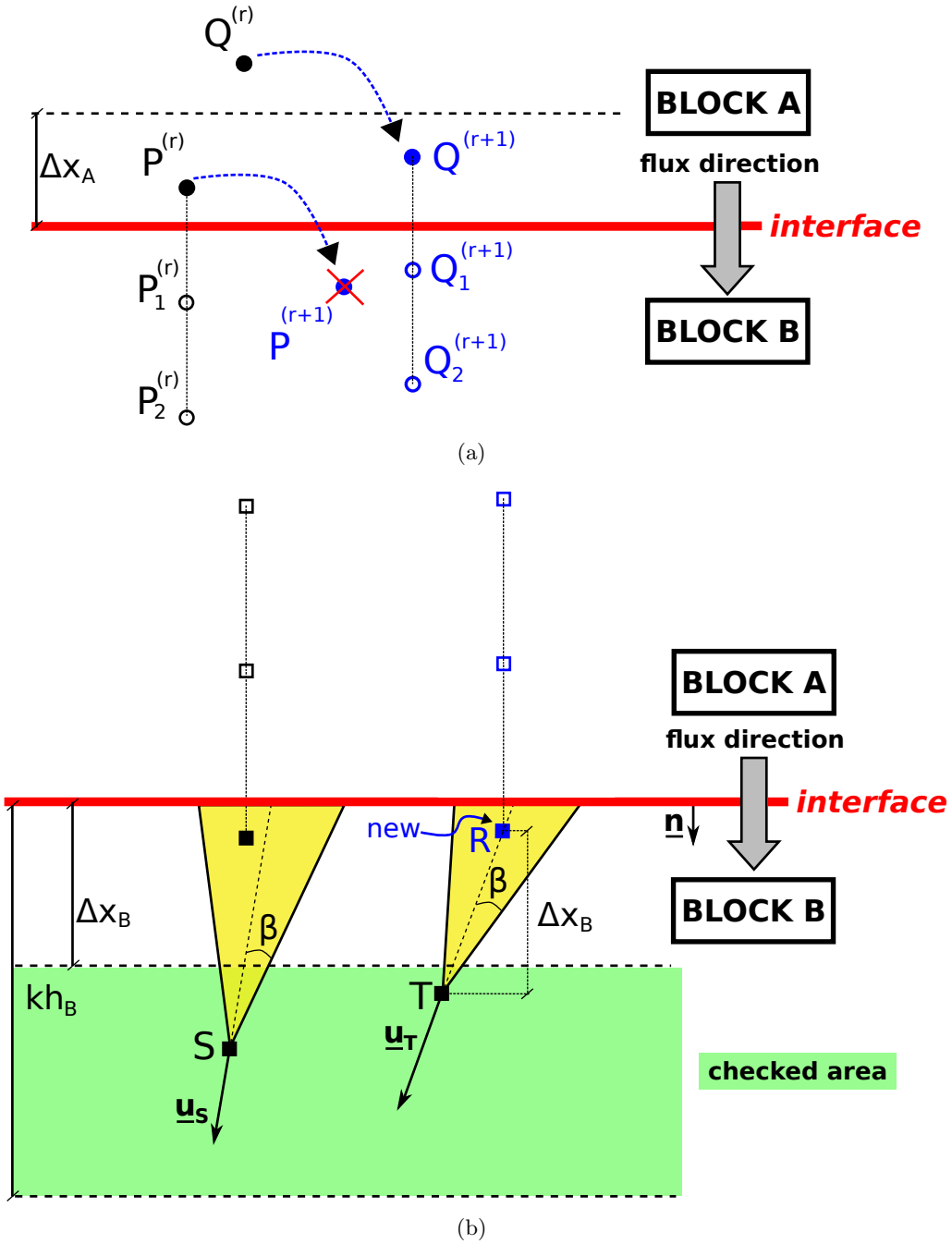


Figure 4.7: Sketch Inflow/Outflow procedure at *block interfaces*. a) Outflow. Full and empty black circles: *effective* and IP particles of block A at time r; full and empty blue circles: *effective* and IP particles of block A at time $r+1$; b) Inflow. Full and empty black squares: *effective* and IP particles of block B; yellow area: conical scan region with opening angle β; R: *effective* particle generated inside the cone with vertex in T; empty blue squares: IP particles of R; green area: checked region. Taken from: Monteleone et al. (2018), 964, fig. 6.

avoided, which would occur if the released particles would be newly generated instead of been taken from the list of the previously canceled ones.

The inflow/outflow procedure at the *block interface* is summarized below:

- *Deleted particles.*
  If one *effective* particle at the end of the time step crosses one of the triangles defining an external boundary or an internal *interface* (connecting one block with the neighboring one) it will be removed from the computation at the next time step. The *interface triangle* is thus considered an outflow for the current block;

- *IP generation.*
  If an *effective* particle has distance d from the *interface* shorter than the particle distance Δx, two or more IP particles are generated as discussed in Sec. 4.2.2. This is valid for both inflow and outflow *interfaces*;

- *Effective* particles with $\Delta x < d < kh$ and $\mathbf{u}_i \cdot \mathbf{n} > 0$.
  If an *effective* particle has distance from the *interface* ranging between the *starting particle distance* Δx and kh, and the velocity component normal to the *interface* plane is positive ($\mathbf{u}_i \cdot \mathbf{n} > 0$, implying an inflow condition) it must be checked if the release of a new particle is required. Thus the procedure described above, based on the cone region analysis, starts;

- Particles with $\Delta x < d < kh$ and $\mathbf{u}_i \cdot \mathbf{n} < 0$.
  If an *effective* particle has distance from the *interface* ranging between Δx and kh but its velocity component normal to the *interface triangle* is negative (thus the particle points outside the interior of the particle block), no new particle is released starting from the current *effective* particle, nor any IP particle is generated since the distance from the interface is larger than Δx.

4.2.5 Flow chart of the *PANORMUS-SPH* code

In order to provide a general description of the single steps required to advance in time the solution in the proposed *multi-domain* technique, a flow chart is shown in Fig. 4.8. The actions indicated in the flow chart are briefly explained in the following:

- ***ACTION*** 1: The domain is partitioned into non-overlapping blocks having a different *smoothing length*. Each block is separated from the neighboring ones by plane or curved *interfaces*. The action is a preparatory step, which is performed only once before running the simulation. Specifically, the file containing the triangles of the whole domain is divided and a boundary triangles file for each subdomain is created. To this aim, in this research study the open-source multiple-platform application *ParaView*®(https://www.paraview.org) and the free software *Autodesk Meshmixer*®(http://www.meshmixer.com) have been used. Moreover, a particle starting file is created starting from the boundary triangle file and the h value of each block;

- ***ACTION*** 2: The starting particle distribution and the boundary triangles file are read for all the blocks. The particle hydrodynamic variables are saved for each *effective* particle;

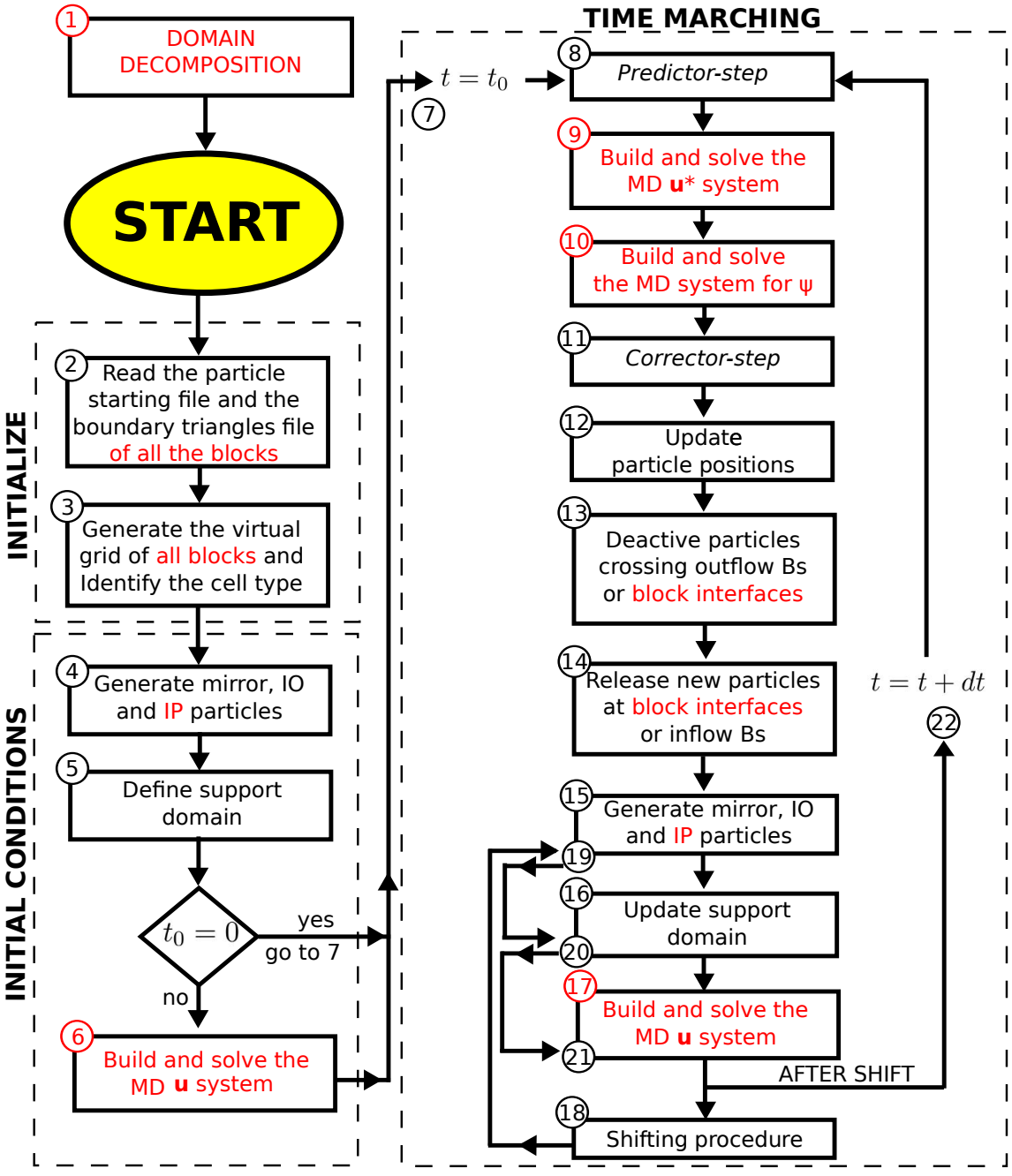


Figure 4.8: Flow chart of the *PANORMUS-SPH* code with the *Multi-Domain* approach. The actions closely related to the *Multi-Domain* approach are highlighted with the red color.

- ***ACTION*** 3: The virtual grid of each block is created. The cubic cells of each virtual grid are classified in types 1, 2, 3 or 4;

- ***ACTION*** 4: For each block the *mirror*, IO and IP particles are generated at solid walls (see Sec. 2.5.1), *open-boundaries* (see Sec. 3.2.1) and *block interfaces* (see Sec. 4.2.2), respectively;

- ***ACTION*** 5: The *support domain* of each *effective* particle is identified considering the neighboring particles lying in the same subdomain with distance shorter than the kh value of the belonging block;

- ***ACTION*** 6: If the simulation starts from developed velocity ($t_0 \neq 0$), the initial velocity of the IP particles is obtained solving the equation system 4.7;

- ***ACTION*** 7: The *fractional step* procedure starts from the initialized simulation time t_0;

- ***ACTION*** 8: In the *predictor-step*, eqn. 2.20 is solved for the *effective* particles of each block to calculate the *intermediate* velocity $\mathbf{u}^*$;

- ***ACTION*** 9: The equations system 4.7 is solved at each *interface* to obtain the *intermediate* velocity $\mathbf{u}^*$ of the IP particles generated from the neighboring blocks;

- ***ACTION*** 10: The *pseudo-pressure* ψ values of the *effective* and IP particles of all the blocks are calculated by solving one single system made up of one *Pressure Poisson equation* (eqn. 2.21) for each *effective* particle and one interpolation equation (eqn. 4.8) for each IP particle;

- ***ACTION*** 11: In the *corrector-step*, the updated velocity $\mathbf{u}$ is calculated for each *effective* particle by solving eqn. 2.24;

- ***ACTION*** 12: The positions of all the *effective* particles are updated using the *corrected* velocities $\mathbf{u}$ calculated in *ACTION* 10;

- ***ACTION*** 13: The particles crossing external outflow boundaries or internal block *interfaces* are deactivated and saved in a *storage* list (as discussed in Sec. 4.2.4);

- ***ACTION*** 14: New particles are released from inflow and *interface* triangles (Sec. 4.2.4). The new particles are generated before the solution of the *MD* system for the *corrected* velocity (*ACTION* 17) in order to improve the Taylor series expansion performance using a more regular particle distribution without voids (due to lack of particles) at the *block interfaces*. The velocity of the new particles coming from *block interface* triangles is set equal to that of the closest particle belonging to the neighboring block. For example, in Fig. 4.7 the velocity of the new particle R of the block B is set equal to that of the closest *effective* particle in block A (not represented in the figure). This velocity will be updated after solving the *MD* system for the velocity as discussed in *ACTION* 17;

- ***ACTION*** 15: Identical to *ACTION* 4;

- ***ACTION*** 16: Identical to *ACTION* 5;

- ***ACTION*** 17: The equation system 4.7 is solved at each *interface* to obtain the *corrected* velocity $\mathbf{u}^*$ of the *IP* particles generated from the neighboring blocks. After solving the system (the velocities of the *IP* particles are known now) the velocity of each new particle i (generated through the *ACTION* 14) is updated with a linear extrapolation among i and the two *interface* particles generated by i;
- ***ACTION*** 18: The shifting procedure proposed by Xu et al. (2009) is used to overcome the well-known tensile instability problem and to improve the particle distribution as described in Sec. 2.8;
- ***ACTION*** 19: The *mirror*, *IO* and *IP* particles are generated as in *ACTION* 4;
- ***ACTION*** 20: As in *ACTION* 16;
- ***ACTION*** 21: The equation system 4.7 is solved at each *interface* to obtain the *corrected* velocity $\mathbf{u}^*$ of the new *IP* particles generated after the shifting procedure;
- ***ACTION*** 22: the solution time is advanced by one time step ($t = t + dt$) and the procedure is restarted with *ACTION* 8.

The activities specifically required by the proposed *Multi-Domain* technique are indicated in red in the flow chart of Fig. 4.8.

4.3 Benchmark test cases

Two test cases have been used in order to properly demonstrate the method efficiency and accuracy through the direct comparison with reliable solutions.

Specifically, the $3D$ unsteady channel flow in a cylindrical pipe and the $2D$ vortex shedding in the wake have been considered. The flow regime in these cases is laminar. The *multi-domain* application to *CAs* will be presented in Chap. 6.

4.3.1 Transient Poiseuille flow

The flow through a cylindrical pipe with diameter $D = 0.1\ m$ and length $L = D$ has been analyzed. The Tab. 4.1 summarizes the data of the simulation. The domain has been partitioned into two coaxial cylindrical blocks, as shown in Fig. 4.9, with the diameter of the internal block $D_2 = 0.6\,D$. The *interface* (plotted in red color in the figure) is therefore a curved surface corresponding to the internal lateral wall of block 1 and the external lateral wall of block 2. The *smoothing lengths* of the external and internal blocks

D $[m]$	L $[m]$	ρ $[kg/m^3]$	ν $[m^2/s]$	∇P $[Pa/m]$	u_{max} $[m/s]$	Re $[-]$	D_2 $[m]$
0.1	0.1	1000	$1 \cdot 10^{-6}$	0.02	0.0125	625	0.06

Table 4.1: *Benchmark test case* - Sec. 4.3.1. Data. D: pipe diameter; L: pipe length; ρ: flow density; ν: kinematic viscosity; ∇P: pressure gradient; u_{max}: analytical maximum velocity; $Re = \bar{u}\,D/\nu$: *Reynolds number* where $\bar{u}$ is the cross-section averaged streamwise velocity at the steady-state; D_2: diameter of the internal block.

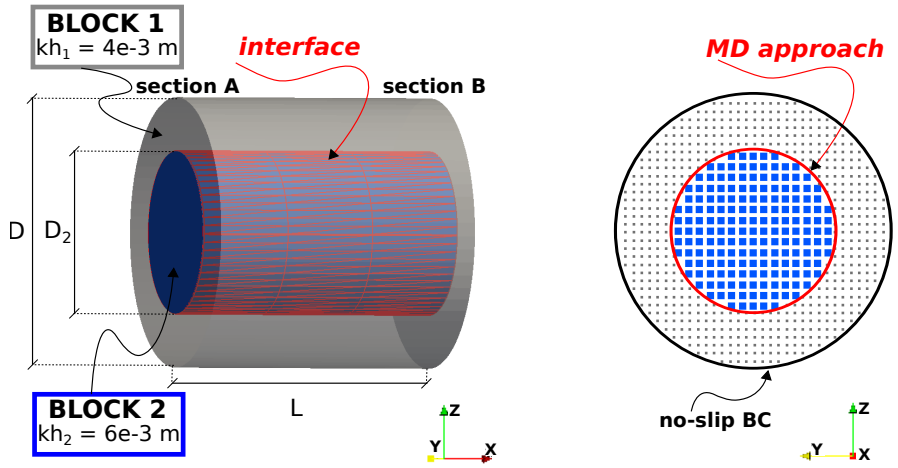


Figure 4.9: *Benchmark test case* - Sec. 4.3. Domain subdivision into blocks 1 (gray) and 2 (blue). The bold red line indicates *block interface triangles*, while A and B are the inflow and outflow sections, respectively. a) surface representation; b) cross-section with particle representation. Taken from: Monteleone et al. (2018), 967, fig. 8.

have been set to $h_1 = 2 \cdot 10^{-3}\ m$ and $h_2 = 3 \cdot 10^{-3}\ m$, respectively (correspondent to $kh_1 = 4 \cdot 10^{-3}\ m$ and $kh_2 = 6 \cdot 10^{-3}\ m$). The resulting initial number of *effective particles* is equal to $N0_1 = 62\,750$ in the first block and $N0_2 = 10\,428$ in the second one. The reduction of the particle numbers with respect to the value obtained using the smallest value of kh in the whole domain is quite moderate in this simple test case (about 25% only) since the geometry and parameters have been chosen only to show the accuracy of the method through the result comparison with the well-known analytical solution. A larger reduction could have been easily obtained employing a higher ratio of the *smoothing lengths*.

A pressure gradient has been used to drive the flow imposing *pressure BCs* at the cross-sections A and B ($p_A = 2 \cdot 10^{-3}\ Pa$ and $p_B = 0$), as discussed in Chap. 3, while *adherence BCs* have been set on the external lateral surface of block 1. The simulation has been performed starting from the rest till achieving the steady-state.

The velocity profiles across the pipe radius at two intermediate time levels (t_1 and t_2) and at the steady-state (t_3) are plotted in Fig. 4.10, showing a very good agreement with the analytical solution (Szymanski, 1932) and a quite satisfactory matching of the solutions near the curved *block interface*. In the figure the values are plotted relative to virtual points at fixed steps of 0.001 m along the pipe radius. Therefore, in the external cylinder (of length 0.02 m along the radial direction) and in the internal one (of length 0.03 m along the radial direction), 20 (empty circles in the figure) and 30 (stars in the figure) virtual points have been obtained, respectively. The number of these points is thus independent on the local refinement. Due to the Lagrangian feature of *SPH*, which allows to obtain the hydrodynamic values in the particle positions only, in each virtual point the velocity has been calculated through Taylor series expansion based on the results of the simulation.

The time evolution of the *effective* particle numbers $N_1(t)$ and $N_2(t)$ is shown in Fig. 4.11, made non-dimensional with the relative starting numbers. The changes in the total number of particles are quite limited in both the blocks, with ratios $N_1(t)/N0_1$ and $N_2(t)/N0_2$ much lower than $0,1\%$ in most of the computational time. It should be noticed

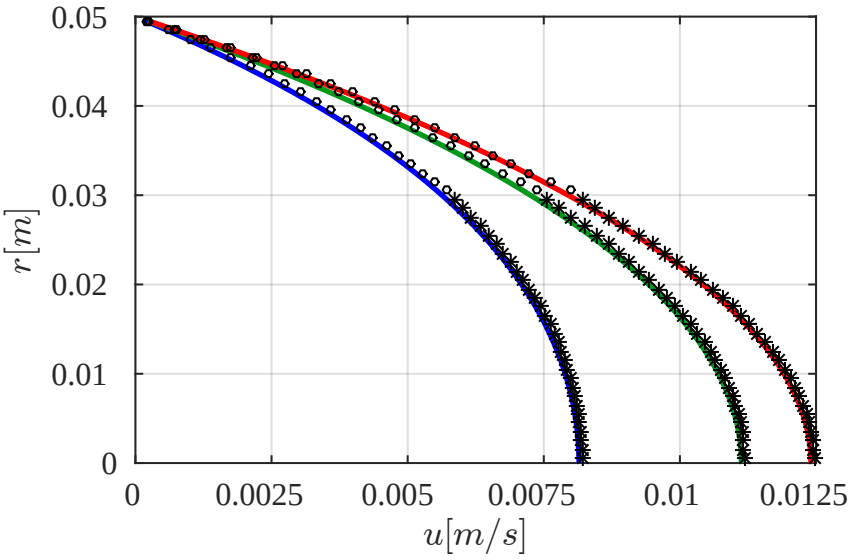
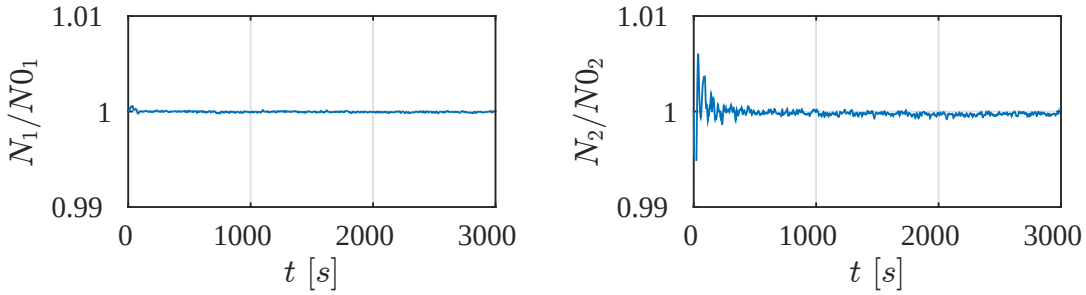


Figure 4.10: *Benchmark test case* - Sec. 4.3.1. Velocity profile as a function of the radial coordinate r. Open circles: *SPH* solution in block 1. Stars: *SPH* solution in block 2. Blue, green and red lines: analytical solutions at $t_1 = 500\ s$, $t_2 = 1\,000\ s$ and at the steady-state, respectively.



(a) (b)

Figure 4.11: *Benchmark test case* - Sec. 4.3.1. Time evolution of the numbers of particles $N_1(t)$ and $N_2(t)$ normalized with the initial numbers $N0_1 = 62\,750$ and $N0_2 = 10\,428$. Taken from: Monteleone et al. (2018), 968, fig. 10.

D $[m]$	L $[m]$	H $[m]$	L_1 $[m]$	H_1 $[m]$	ρ $[kg/m^3]$	ν $[m^2/s]$	$\bar{u}$ $[m/s]$	Re $[-]$
0.1	2.2	0.41	0.2	0.2	1	10^{-3}	1	100

Table 4.2: *Benchmark test case* - Sec. 4.3.2. Data taken from the test case *2D-2* of Schäfer et al. (1996). D: diameter of the cylinder. L: length of the channel; H: height of the channel; L_1: distance between the center of the cylinder and the inflow section; H_1: distance between the center of the cylinder and the bottom wall; ρ: fluid density; ν: fluid kinematic viscosity; $\bar{u}$: mean velocity imposed at the inlet section; $Re = \bar{u}\,D/\nu = 100$: *Reynolds number*.

that higher values (lower than 1%) can be seen only at the starting of the simulation, due to the perfectly regular initial distribution of the particles, which results in the leaving and entering of entire slices of particles through the inflow and outflow sections. This effect is particularly evident in block 2 since the particle velocity is larger, but is rapidly canceled as the particle distribution becomes less regular.

4.3.2 Von Kármán vortex shedding

The $2D$ laminar flow around a circular cylinder has been studied. The geometry, fluid properties and boundary conditions have been assigned as in the test case *2D-2* of Schäfer et al. (1996) (see Tab. 4.2), which has been used for comparison. The length L and the height H of the computational domain have been set equal to $22\,D$ and $4.1\,D$, respectively. The center of the cylinder is located at distance $L_1 = 2\,D$ from the inflow section and $H_1 = 2\,D$ from the bottom wall.

The fluid density and kinematic viscosity have been set to $\rho = 1\ kg/m^3$ and $\nu = 10^{-3}\ m^2/s$. A parabolic profile has been imposed at the inflow with mean velocity $\bar{u} = 1\ m/s$, imposing *incoming BCs* (as discussed in Chap. 3), resulting in the *Reynolds number* $Re = \bar{u}\,D/\nu = 100$. Since the simulation has been started from the rest, in order to obtain a smoother transition, the selected inflow velocity has been imposed after $0.1\ s$ from the starting of the simulation, with a linear increase from the initial null value. *Adherence BCs* have been used at the lateral walls and on the immersed body, while null velocity derivatives and null pressure have been imposed at the outflow (setting *pressure BCs* on the outflow triangles).

The refinement of the solution has been increased in the vicinity of the immersed body considering three subdomains. The Fig. 4.12 shows the domain decomposition with blocks 2 and 3 in the annular regions close to the cylinder and block 1 elsewhere. The smallest value $kh_3 = 0.02\,D$ (where $D = 0.1\ m$ is the diameter of the immersed cylinder) has been used in block 3 (between the diameters D_2 and D), the intermediate value $kh_2 = 0.04\,D$ in block 2 (between $D_2 = 1.5\,D$ and $D_1 = 2\,D$) and the largest value $kh_1 = 0.1\,D$ in block 1, covering most of the domain. The resulting initial number of *effective* particles is $N_{e,tot} = 48\,086$ (with $34\,816$ particles in block 1, $3\,456$ in block 2 and $9\,814$ in block 3), about 5% of the value that would have been obtained using a constant value of the smallest kh ($0.02\,D$) in the whole domain. During the simulation the number of particles in the blocks remained almost constant, with changes limited to 0.8% in the largest block 1 and 0.6% and 0.25% in blocks 2 and 3, respectively.

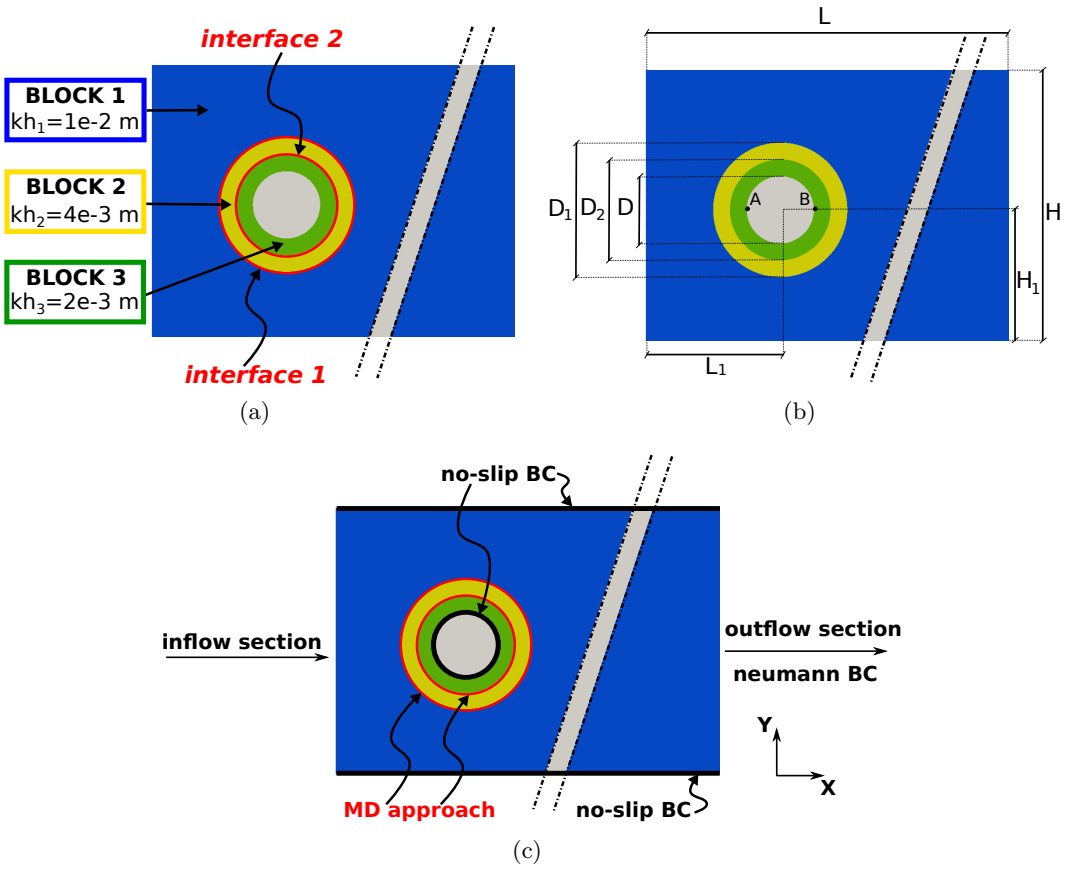


Figure 4.12: *Benchmark test case* - Sec. 4.3.2. a) Domain subdivision into blocks 1 (blue), 2 (yellow) and 3 (green). The bold red lines indicate the two *block interfaces*; b) domain dimension. $L_1 = 0.2\ m$, $L = 2.2\ m$, $D_1 = 0.2\ m$, $D_2 = 0.15\ m$, $D = 0.1\ m$, $H_1 = 0.20\ m$ and $H = 0.41\ m$. Front point A ($x_A = 0.15\ m$, $y_A = 0.20\ m$) and end point B ($x_B = 0.25\ m$, $y_B = 0.20\ m$) of the cylinder; c) boundary conditions. Taken from: Monteleone et al. (2018), 968, fig. 11.

The periodic detachment of vortices from either sides of the cylinder, characteristic of the considered flow at Re values in the range of about $50 \div 150$, is easily identified in Figs. 4.13 and 4.14. The Fig. 4.13.a shows the particle streamwise velocity at time $t = 8\ s$ after the starting of the simulation, corresponding to about 24 vortex shedding periods T. An enlargement of the near cylinder region is shown in Figs. 4.13.b and 4.13.c, where the particle streamwise velocity and velocity vectors are plotted respectively, showing a very good matching of the solution through the *block interfaces*. The increasing particle distance while moving outwards from the cylinder and the corresponding velocity vectors is clearly seen in Figs. 4.13.d and 4.13.e.

The vortices are shown in Fig. 4.14 at the time levels $t = 6.31\ s$ (one of the peaks of the lift coefficient that will be defined below), $t + T/4$, $t + T/2$ and $t + 3/4T$. In the figure the vectors are colored as the corresponding blocks in Fig. 4.12.

The smoke lines are plotted in Fig. 4.15 with reference to the inflow positions corresponding to the cylinder height (between $y = H_1 - D/2$ and $y = H_1 + D/2$), using different colors from red to light blue to indicate growing distances from the axis of the domain. A peculiar procedure has been implemented to obtain the continuity of the smoke lines while

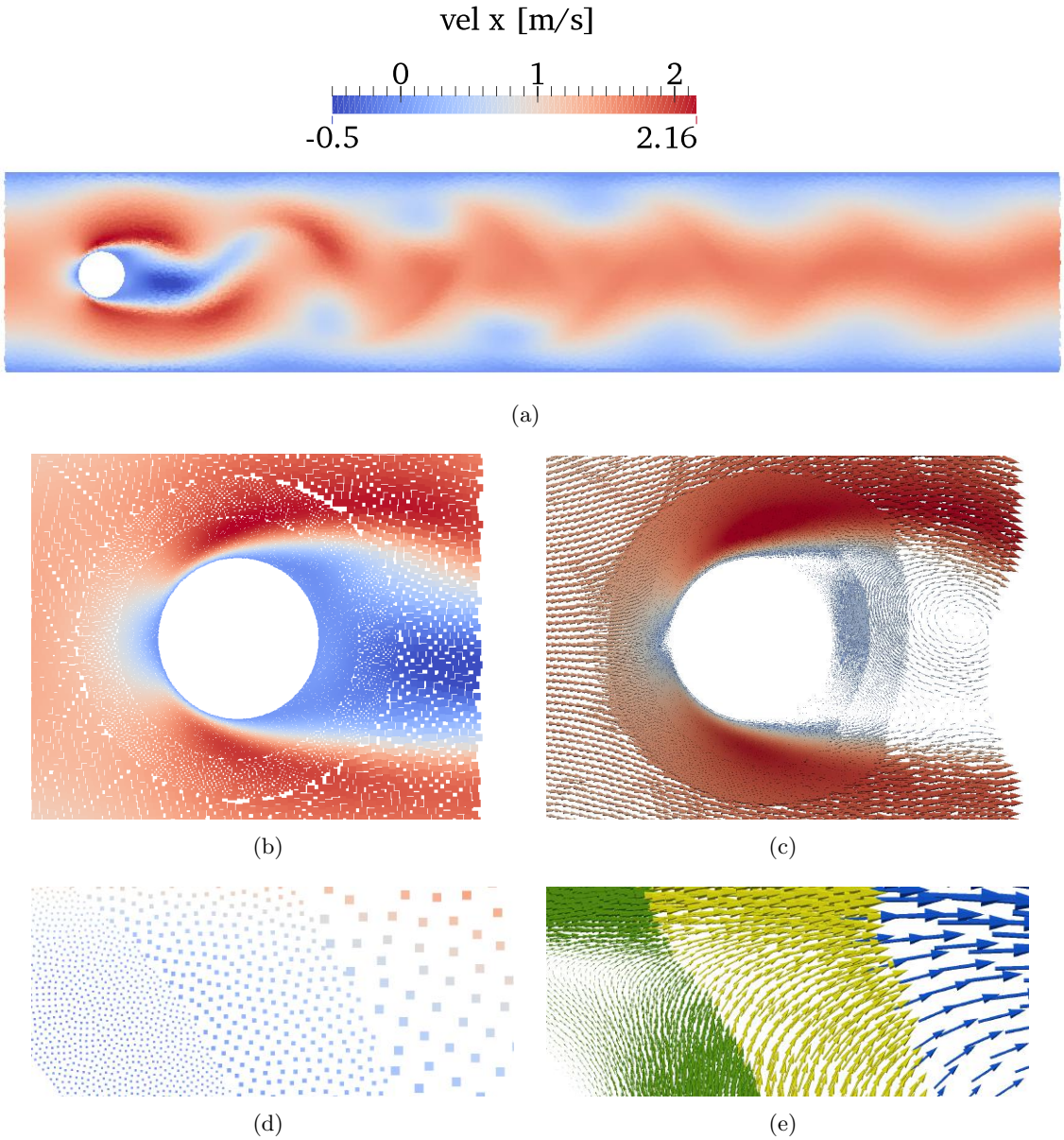
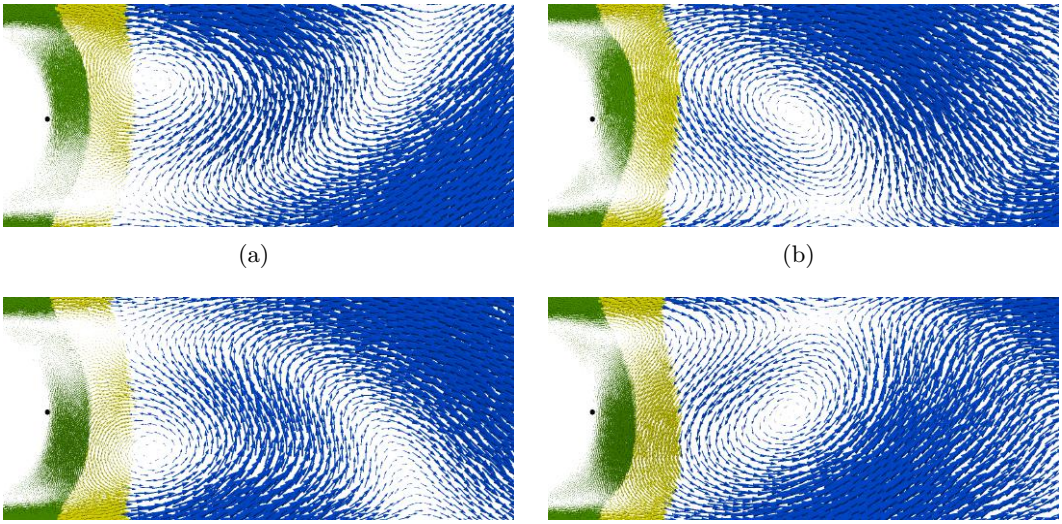


Figure 4.13: *Benchmark test case* - Sec. 4.3.2. Velocity field at time $t = 8\ s$. a) Streamwise particle velocity in the whole domain; b) streamwise particle velocity near the circular cylinder; c) velocity vectors near the circular cylinder; d) particle distribution at the subdomain transitions; e) velocity vectors at the subdomain transitions colored as the corresponding blocks in Fig. 4.12. Taken from: Monteleone et al. (2018), 969, fig. 12.

particles are deleted and generated through *block interfaces*. Specifically, the smoke value is assigned as a new property to each *effective* particle through a scalar variable named *smoke* that is initialized to zero at the beginning of the simulation. While new *effective* particles are generated at the inflow section of the block 1 they acquired the smoke value as a function of the distance from the axis of the channel. Due to the Lagrangian nature of the *SPH*, the particles carry the *smoke* property while they move. When a particle is deactivated through *block interface*, its coordinates and *smoke* value are stored for 10 iteration of the simulation in a list of the block where the particle is contained. Therefore,



(a) (b) (c) (d)

Figure 4.14: *Benchmark test case* - Sec. 4.3.2. Velocity vectors (colored as in Fig. 4.12) at different times levels. a) $t = 6.31\ s$; b) $t+T/4$; c) $t+T/2$; d) $t+3/4T$), where $T = 0.328\ s$ is the vortex shedding period.

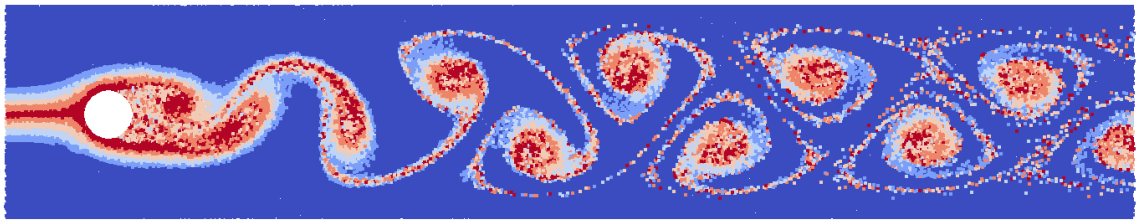


Figure 4.15: *Benchmark test case* - Sec. 4.3.2. Smoke lines at time $t = 8\ s$. Taken from: Monteleone et al. (2018), 970, fig. 14.

when a new particle is generated in a subdomain through *block interface* it is searched from the aforementioned storage list of the neighboring block the closest particle considering a limit of maximum distance equal to $\Delta x/2$. The closest particle is thus sought starting from the particle deactivated in the current time step and going back in iteration (until the limit of 10) if the distance limit is not satisfied. Assigning to the new particles the *smoke* value of the closest deactivated particles in the neighboring block the continuity of the smoke line is insured.

The Figs. 4.16.a and 4.16.b show the time evolution of non-dimensional drag coefficient C_D and lift coefficient C_L. These coefficients have been calculated as

$$\begin{aligned} C_D &= \frac{2}{\rho \bar{u}^2 D} \int_S \left(\rho\nu \frac{\partial u_t}{\partial n} n_y - p n_x \right) dS \\ C_L &= -\frac{2}{\rho \bar{u}^2 D} \int_S \left(\rho\nu \frac{\partial u_t}{\partial n} n_x + p n_y \right) dS \end{aligned} \tag{4.13}$$

where S is the cylinder surface (discretized into line segments in the $2D$ approximation), u_t is the tangential velocity, p is the pressure and n is the surface normal direction pointing

outwards, with components n_x and n_y in the horizontal and vertical directions respectively. The surface integrals in eqn. 4.13 have been calculated in discrete form considering the middle points of the line segments. The figures show that stable conditions, corresponding to the complete vortex development, are achieved after about 5 s. The maximum value of the drag coefficient after $t = 5\ s$ oscillates between 3.345 and 3.370, with an average value of 3.354, slightly larger (about 3.5%) than the optimal value of $3.22 \div 3.24$ estimated by Schäfer et al. (1996). Correspondingly, the maximum value of C_L oscillates between 0.956 and 0.989, with an average value of 0.972, only 1.7% lower than the estimated optimal value of 0.99.

The pressure difference between the front point A having coordinates $(L_1 - D/2, H_1)$ and the end point B $(L_1 + D/2, H_1)$ of the cylinder (see Fig. 4.12.b) is plotted in Fig. 4.16.c. The obtained mean value in the time period $t = 5 \div 10\ s$ is $\Delta p = 2.484\ Pa$, in perfect agreement with the values suggested by Schäfer et al. (1996), where the range $2.46 \div 2.50\ Pa$ is reported. The mean frequency of separation f has been estimated from the period of oscillation T of the lift coefficient C_L, resulting in the value $f = 3.043\ Hz$. Correspondingly, the value of the *Strouhal* number $St = Df/\bar{u} = 0.304$ has been obtained, which is again in perfect agreement with the reference values of $0.295 \div 0.305$.

The Fig. 4.17 shows a comparison of the results at $t = 2\ s$ with those obtained using a constant value of the *smoothing length* with $kh = 0.04D$ (single domain SD in the figure). The velocity, the smoke lines and the pressure field (shown in a enlargement in the vicinity of the immersed body) are plotted in Figs. 4.17.a, 4.17.b and 4.17.c, respectively, showing

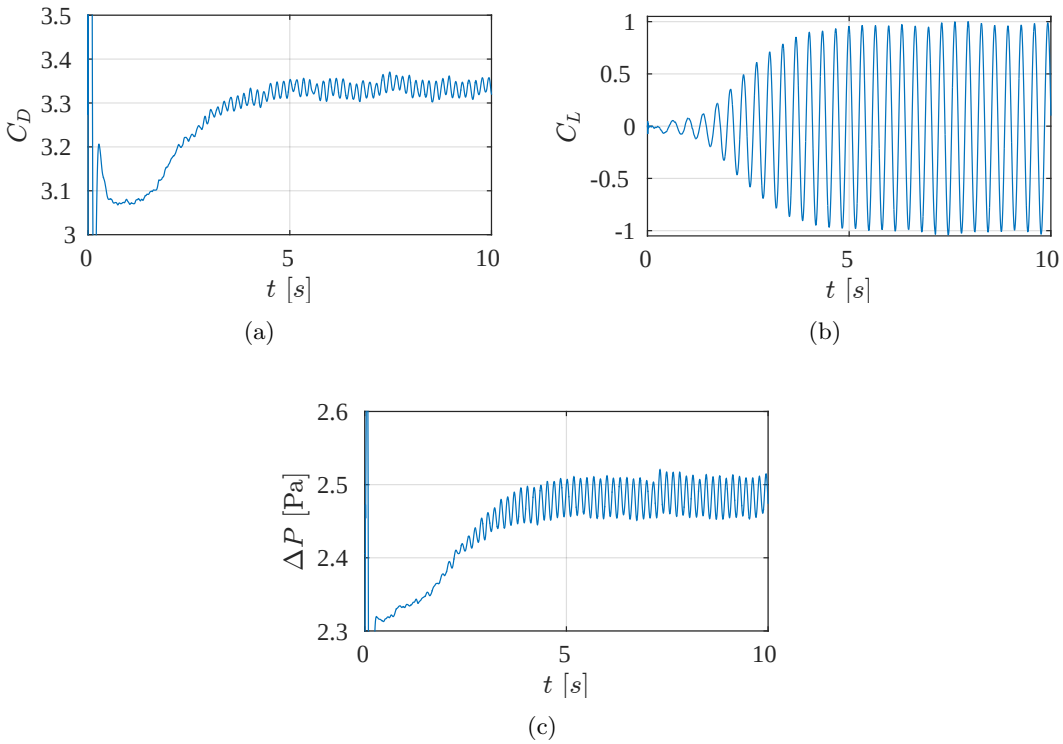
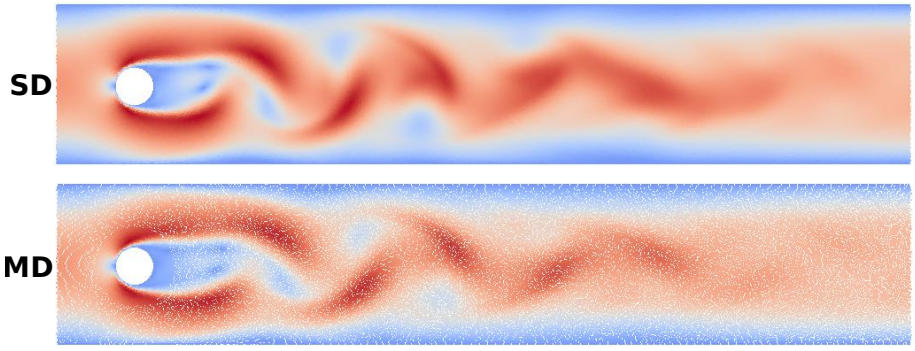
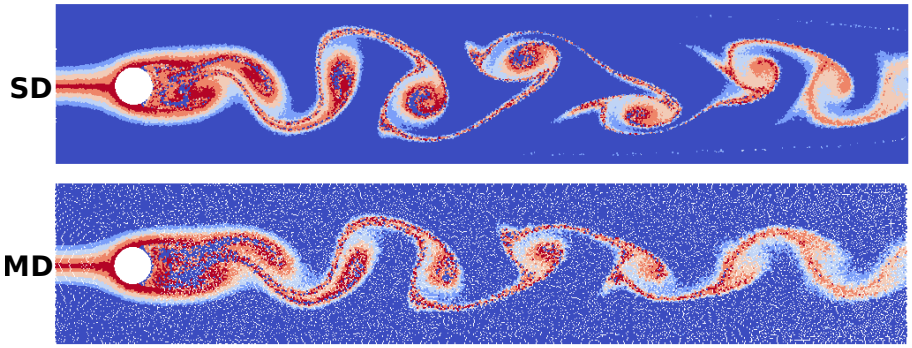


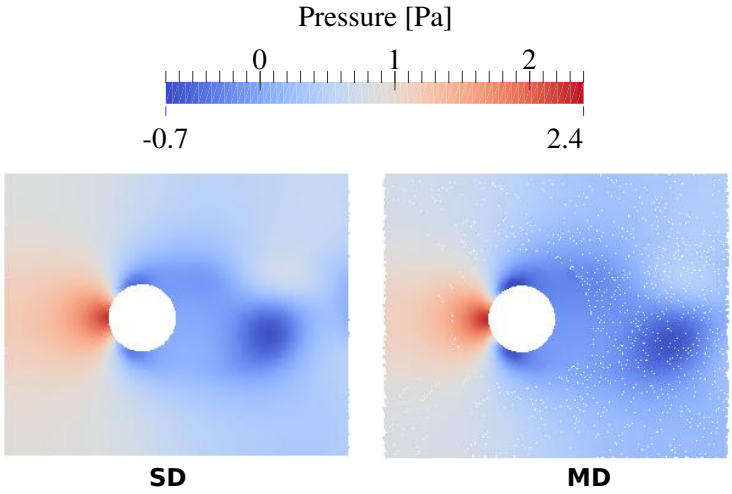
Figure 4.16: *Benchmark test case* - Sec. 4.3.2. a) Drag coefficient C_D; b) lift coefficient C_L; c) pressure difference Δp $(p_A - p_B)$. Taken from: Monteleone et al. (2018), 971, fig. 15.



(a)



(b)



(c)

Figure 4.17: *Benchmark test case* - Sec. 4.3.2. Comparison between single domain (*SD*) and the proposed multi-domain (*MD*) approaches at $t = 2\ s$. a) Streamwise particle velocity. The scale is the same of Fig. 4.13; b) smoke lines; c) pressure field in the vicinity of the immersed body. Taken from: Monteleone et al. (2018), 972, fig. 16.

a very good agreement between the *MD* and *SD* results.

Chapter 5

SPH Parallel Computing for *Single* and *Multi-Domain* approaches

This chapter presents the implemented *parallel computing* scheme for *Single* and *Multi-Domain SPH* approaches. Several implementation details are provided as well as some scalability tests to analyze the performance of the parallel *SPH* code.

5.1 Background and motivations

Applying the *SPH* technique to cerebral vessels with *CAs* involves simulations requiring a very large number of particles and consequently leads to unsustainable computational efforts. Multi-resolution techniques, as the *Multi-Domain* method (described in Chap. 4), can dramatically reduce the computational costs alleviating this issue, however these procedures alone are not enough to perform *real-time* simulations. In this framework, *high-performance computing* (*HPC*) is necessary, allowing to meet the computational complexity of hemodynamics in *CAs* with efficient scalable programs.

In Lagrangian particle methods, as *SPH*, the implementation of *parallel computing* is not trivial since it must be taken into account that the particles move during the computation. Therefore, parallelization algorithms must dynamically handle particles leaving or entering the domain of each processor.

One of the most important step in the code parallelization is the *domain distribution* where the computational domain is divided among several processors. This step is common for *WCSPH* and *ISPH* solvers. The efficiency of the *parallel computing* is closely linked to the *domain distribution*: if the computation division is not fair, the overloaded processor becomes the process bottleneck. In the *SPH* method an efficient *load balancing*, and thus a highly scalable parallel performance, is achieved assigning a fair number of *effective* particles to each processor as well as grouping physically close particles within a single processor in order to reduce inter-processor communication. These operations are not trivial in computational domains with complex geometries due to the irregular distribution of the particles (such as the geometry of the *CA* and the parent vessel with the surrounding branches). Moreover, a peculiar attention must be paid to maintain during the simulation an efficient *load balancing* while the particles move from one region of the

computational domain to another.

When the *ISPH* approach is adopted, the *parallel computing* implementation is more challenging since the *PPE* linear system must be solved. It should be noted that the *Poisson* solver is responsible for over 80% of the computation time, thus great attention must be focused on the *Poisson* system parallelization. Moreover, as discussed in Sec. 2.7, since the particle connectivity is constantly changing with the evolution of the flow, the *ISPH* algorithm requires the solution of a new *PPE* system, whose sparse coefficient matrix changes at each time step.

HPC strategies are based on the employment of parallel multiple *Central* and/or *Graphical Processor Units* (*CPU* and *GPU*, respectively) architectures. Although the use of multiple *GPUs* is a very promising and powerful alternative to *HPC* clusters, currently in *CFD* simulations multiple *CPUs* paradigm is the most widespread and used standard where efficient and stable libraries can be found and robust algorithms can be thus developed.

Some *CPU*-based parallel computing procedure have been proposed in the *ISPH* framework. Yeylaghi et al. (2016) proposed an *OpenMP*-based parallel scheme, although in this procedure the *Poisson* equation is solved explicitly without the use of a matrix, which affects accuracy and limits the time step size. Recently, Guo et al. (2018) developed a massively parallel scheme to solve incompressible *SPH* for free-surface flows for simulations involving even more than 100 million particles using the *Message Passing Interface* (*MPI*) paradigm.

It should be noted that in the last years the attention towards the *GPU* architecture is strongly developing especially in *SPH*. The *open-source DualSPHysics* code is ought to be mentioned in this context. This code, which is developed in the *WCSPH* approach, is designed to launch simulations either on multiple *CPUs* using *OpenMP* or on a *GPU* (Crespo et al., 2015). Domínguez et al. (2013) added in the *DualSPHysics* code an *MPI* implementation for *Multi-GPU* execution. Recently, Chow et al. (2018) proposed a novel implementation of a parallel *ISPH* algorithm on a single *GPU*. To the author' best knowledge, the solution of the *PPE* system on multiple *GPUs* is still a goal to reach.

In this research study the *PANORMUS-SPH* code has been parallelized on multiple *CPUs* using the *MPI* paradigm for communication between partitions. Future work will be aimed at parallelizing the code within the emerging *GPU* architecture. The implemented *high-performance computing* in *SPH* (*SPH-HPC* in the following) is general for the classical *SPH* technique with constant resolution (*Single-Domain*) and for the *Multi-Domain* approach. In this chapter the *SPH-HPC* is first explained with reference to the *Single-Domain* approach. Then the *HPC* implementation is extended to the *Multi-Domain* approach where the *HPC* procedure more complex due to the use of variable kh values in the subdomains.

It should be highlighted that in the following the term *"decomposition"* refers to the creation of different blocks starting from the whole computational domain in the *MD* approach, whilst the term *"distribution"* is used to indicate the division of the domain among processors in the *parallel computing* scheme.

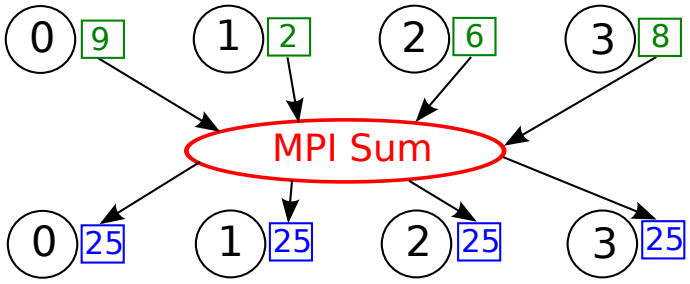


Figure 5.1: Scheme of the *MPI_ALLREDUCE* function. Black circles: processor *id*; green rectangles: input values; blue rectangles: output values.

5.1.1 Message Passing Interface

The *MPI* paradigm is a specification for a standard library for *message passing* proposed by a broadly based committee of vendors, implementors, and users (Gropp et al., 1996). The *message-passing* paradigm has been widely used for implementing *high performance computing* since the available optimized *MPI* libraries and their wide portability. It can be used in fact in communication for *distributed-memory* and *shared-memory* multiprocessors, networks of workstations, as well as combinations of these elements. *MPI* provides a set of libraries for writing, debugging, and performance-testing distributed programs with language bindings for C, C^{++}, and $FORTRAN$.

The *MPI* functions mentioned in this chapter are listed below.

- $MPI\_ALLREDUCE$: this function applies a reduction operation to the vector *send-buffer* of each processor and distributes the results to all processes. In Fig. 5.1 each processor sends the value to be shared (green square in the figure). These values are summed and then the result is distributed to all processors (blue squares in the figure).
- $MPI\_SENDRECV$: it is a blocking send and receive operation. In the implemented *SPH-HPC* algorithm, this function is used to share particles with neighboring processors (explained in Sec. 5.2.3 and Sec. 5.2.4);
- $MPI\_ALLTOALL$: it allows each process to send and receive distinct data (with the same amount of information) from other processors, including itself. In this research study, it is used to share the number of *interface* particles (see Sec. 5.3.3);
- $MPI\_ALLTOALLV$: it allows to send data from all to all processes. Differently from the function $MPI\_ALLTOALL$, each processor may send a different amount of data may provide displacements for the input and output data. This function is used to share the coordinates of the *interface parallel* particles (explained in Sec. 5.3.3).

5.2 SPH-HPC for the *Single-Domain* approach

5.2.1 Domain distribution

The *domain distribution* procedure implemented in the *PANORMUS-SPH* code allows to subdivide the computational domain among the selected number of processors N_{procs}

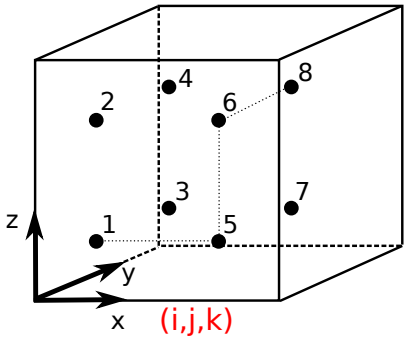


Figure 5.2: Sketch of the particle numbering for the cell (i, j, k) where i, j, k are the cell index in the x, y, z-directions, respectively.

in order to obtain a well-balanced loads distribution. In the following the processors are indicated with the index *id* numbered starting from zero up to $N_{procs} - 1$; on the other hand, when explaining a specific action, the concerned processor will be named *myid*.

In order to partition the computational domain, the virtual grid of cubic cells discussed in Sec. 2.5.1 is used. Before explaining the subdivision procedure, a brief explanation of the particle numbering inside the virtual grid must be provided. When the particles are initially distributed inside the domain, in each cell (i, j, k) the *effective* particles are arranged first in the z-direction, then in the y-direction and finally in the x-direction as shown in Fig. 5.2. Likewise, in the virtual grid the particles are numbered spanning the cell columns in the z-direction starting from the bottom to the top and increasing first the j index (from 1 to ny) and then the i index (from 1 to nx). Considering the scheme in Fig. 5.3.a, first the particle in cell $(1, 1, 1)$ are thus numbered, than those in cells $(1, 1, 2)$, $(1, 1, 3)$ and so on till $(1, 1, nz)$. Later the second index is increased starting again from the bottom and thus spanning the cells from $(1, 2, 1)$ till $(1, 2, nz)$ (Fig. 5.3.b). The same process is repeated with the index j increasing till ny (Fig. 5.3.c). Finally, the index i is increased from 1 till nx (Figs. 5.3.d, 5.3.e and 5.3.f).
The Fig. 5.4 shows a simpler $2D$ scheme of the particle numbering employing the same procedure.

At the beginning the cells of the virtual grid containing *effective* particles are distributed among the selected number of processors in order to allot to each processor a number of particles close to the theoretical one N^t (equal to the least integer greater than or equal to N_e/N_{procs})

$$N^t = ceiling(N_e/N_{procs})$$

Each processor scans the particles from 1 to N_e following the numbering explained above. The cells containing the first N^t particles are assigned to the first processor ($id = 0$). The second and, in general, the following processors (id from 1 to $N_{procs}-1$), while scanning the particles, check if the corresponding cell (identified through eqn. 2.14) has been already assigned to another processor and start to count the theoretical number of particles when the first cell yet not assigned is found. If one cell has been allotted to the processor id, all the particles inside the cell will belong to id, even if the required number N^t had been reached. Therefore, the particles belonging to the processor id at the beginning, indicated as $N_{(id)}$, are

$$N_{(id)} = N^t + res \tag{5.1}$$

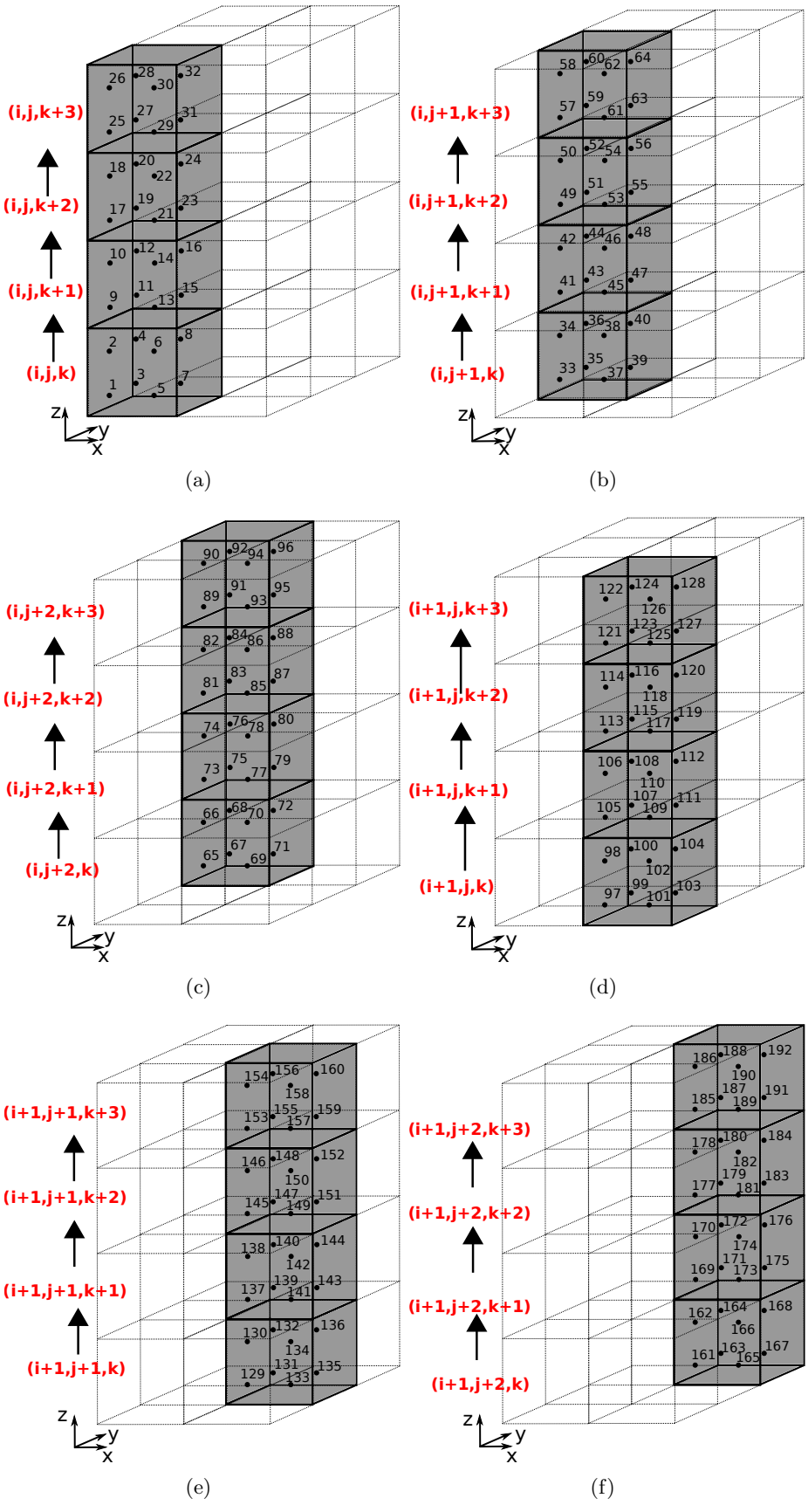


Figure 5.3: 3*D* Sketch of the particle order in the virtual grid.

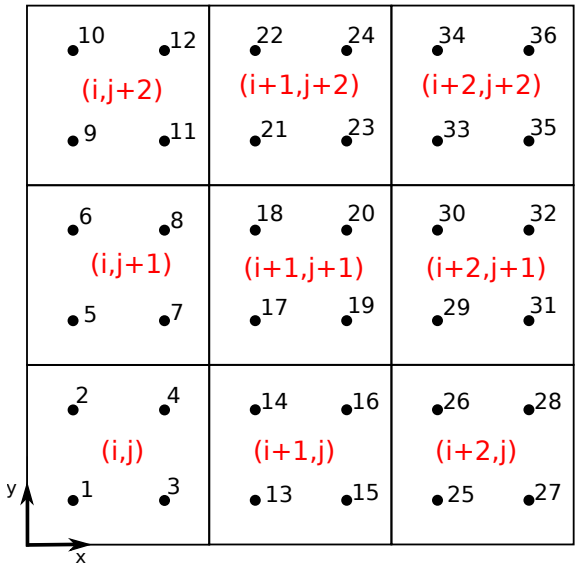


Figure 5.4: $2D$ Sketch of the particle numbering in the virtual grid.

where res is the number of remaining *effective* particles inside the last cell assigned to the processor id. In $3D$ approximation the maximum and minimum values of res are 7 (if only one particle was taken in the last cell) and 0 (if all the particles were taken in the last cell), respectively, since each cell contains 8 particles. On the other hand, in $2D$ approximation the maximum value of res is 3, since 4 particles are contained in each cell.

If the domain walls are fixed, that is one of the hypothesis of this research study (as discussed in Sec. 6.1), each processor maintains the same cells during the simulation while changing $N_{(id)}$ since the particles move and can switch to cells belonging to other processors. A detailed description of the procedure of the particle leaving and entering the processor domain is explained in Sec. 5.2.4.

The Fig. 5.5 shows a $3D$ scheme of the cell distribution, considering a domain with $N_e = 288$ and two processors ($N_{procs} = 2$). In the figure the domains of the processors are represented with different colors (cyan and yellow for the first and second processors, respectively) and the first and last particle counted by each processor are highlighted by red full squares and diamonds, respectively. In this simple example, the first processor counts the particles from 1 to 144 (that is the theoretical number of particles N^t), thus the cells of these particles are assigned to id 0 (cyan cells in the figure). The second processor counts the particles starting from the first particles lying in a cell not yet assigned (particles 145) until the required number of particles is reached (particles 288). Therefore, the cells of these particles are assigned to id 1 (yellow cells in the figure). In this example $res = 0$ for both processors.

The Figs. 5.6 and 5.7 show a $2D$ example with $N_e = 348$ considering different number of processors. In Fig. 5.6.a, the computational domain is distributed between two processors; the theoretical number of particles for each processor id is $N^t = ceiling(348/2) = 174$. For the first processor the particles from 1 to 174 are counted and the cells of these particles are assigned to id 0 (cyan cells in the figure). Since the last cell has other 2 particles (175 and 176) also these particles are counted and thus $N_{(0)} = 176$. For the second processor the first cell not yet assigned is the one containing the particle 177; thus the particles

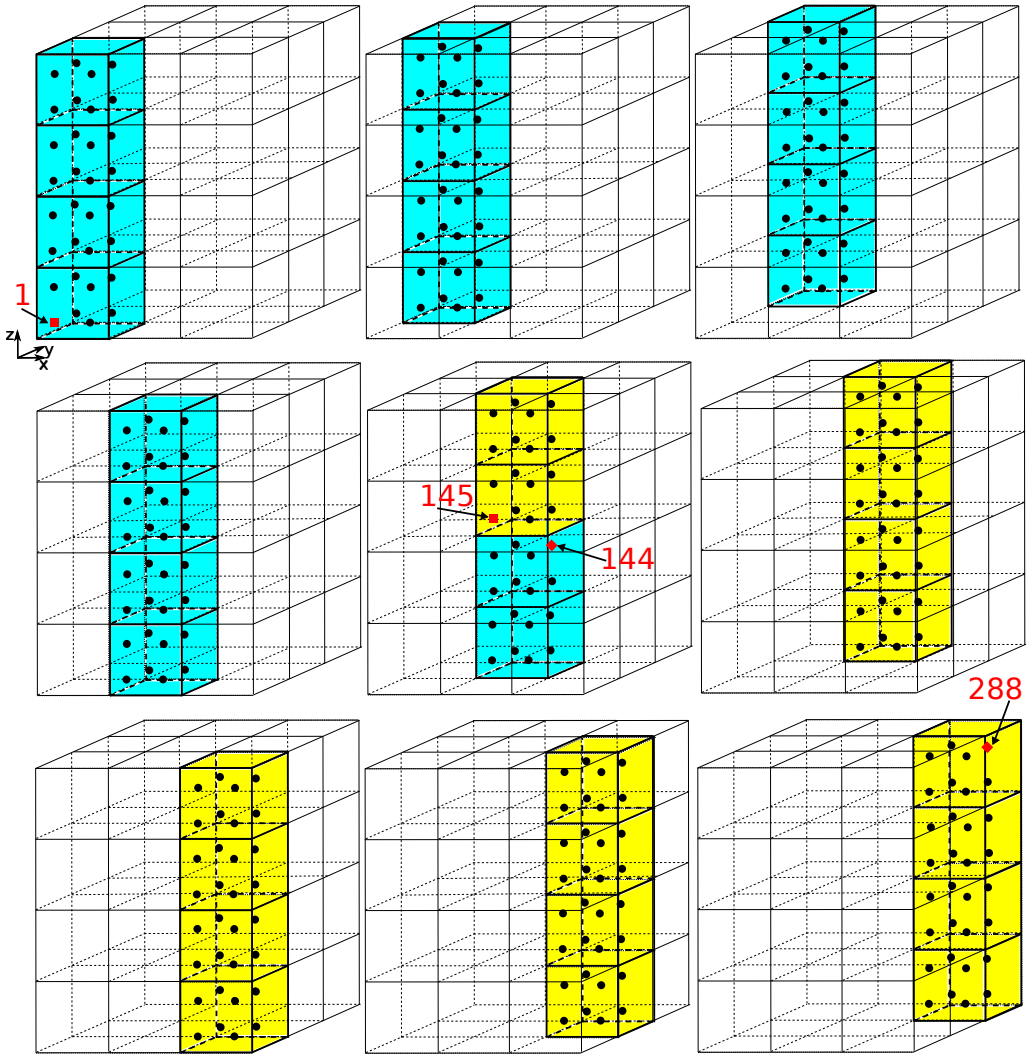
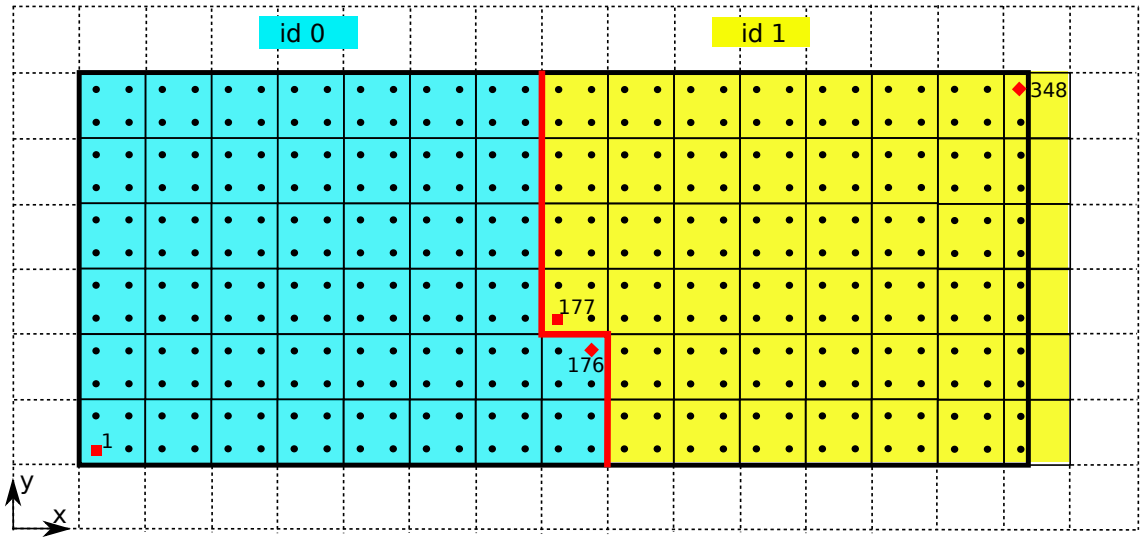


Figure 5.5: $3D$ Sketch of the distribution of the cells containing *effective* particles. $N_e = 288$, $N_{procs} = 2$. Cyan region: domain of the processor *id* 0; yellow region: domain of the processor *id* 1; red full squares and diamond: first and last particles of each processor.

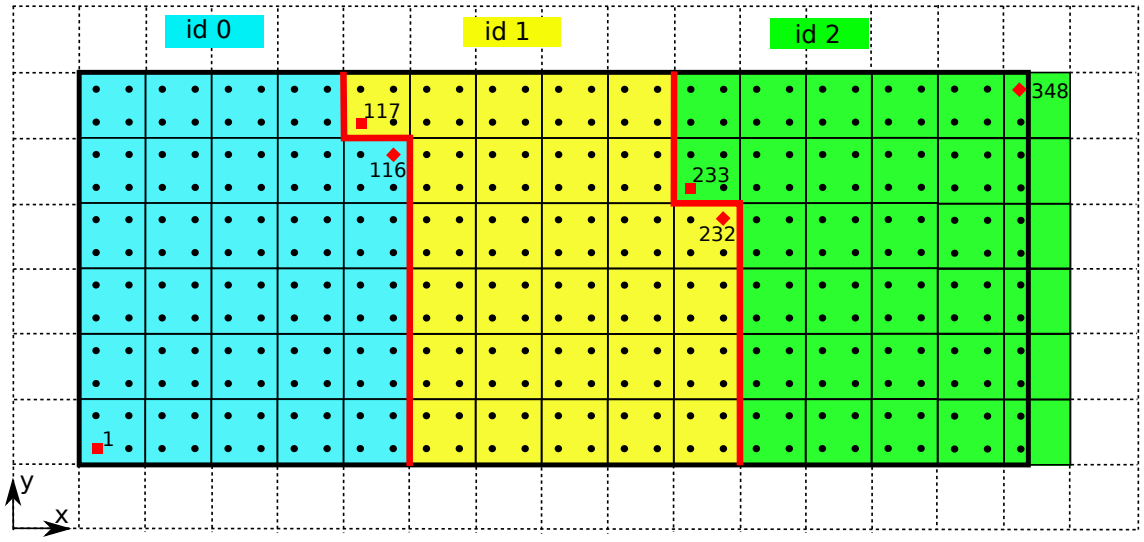
are counted starting from 177 to 348, resulting in $N_{(1)} = 172$. Therefore, the cells of these particles are assigned to *id* 1 (yellow cells in the figure). The surfaces separating neighboring processors are named *parallel interfaces*. Due to the $2D$ representation these surfaces are represented with segments (bold red lines) in the figures.

The Fig. 5.6.b shows the cell distribution with 3 processors ($N^t = ceiling(348/3) = 116$). The first processor takes the cells from the particle 1 to 116 with $N_{(0)} = 116$ (cyan cells in the figure), the second takes the cells from 117 to 232 with $N_{(1)} = 116$ (yellow cells) and the third those from 233 to 348 with $N_{(2)} = 116$ (green cells).

In Fig. 5.6.c the cell distribution with 4 processors in shown with $N^t = ceiling(348/4) = 87$. The first processor takes the cells from the particle 1 to 88 with $N_{(0)} = 88$ (cyan cells), the second those from 89 to 176 with $N_{(1)} = 88$ (yellow cells), the third those from 177 to 264 with $N_{(2)} = 88$ (green cells) and the fourth those from 265 to 348 with $N_{(3)} = 84$ (pink cells). The Fig. 5.7 shows other two schemes of the cell distribution with 4 and 8 processors (Fig. 5.7.a and Fig. 5.7.b, respectively). The number of particles for each



(a) $N_{procs} = 2$

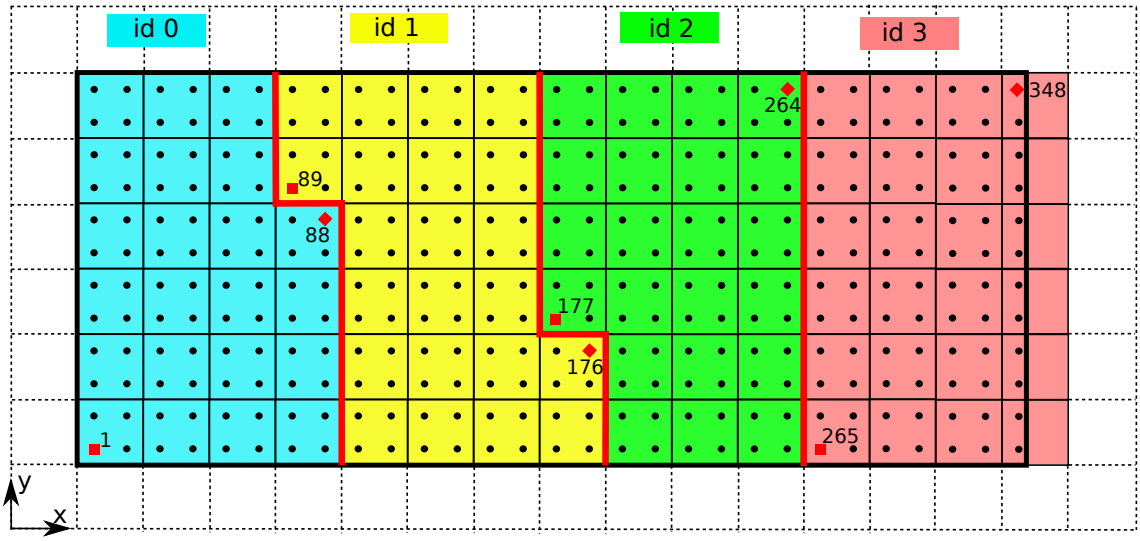


(b) $N_{procs} = 3$

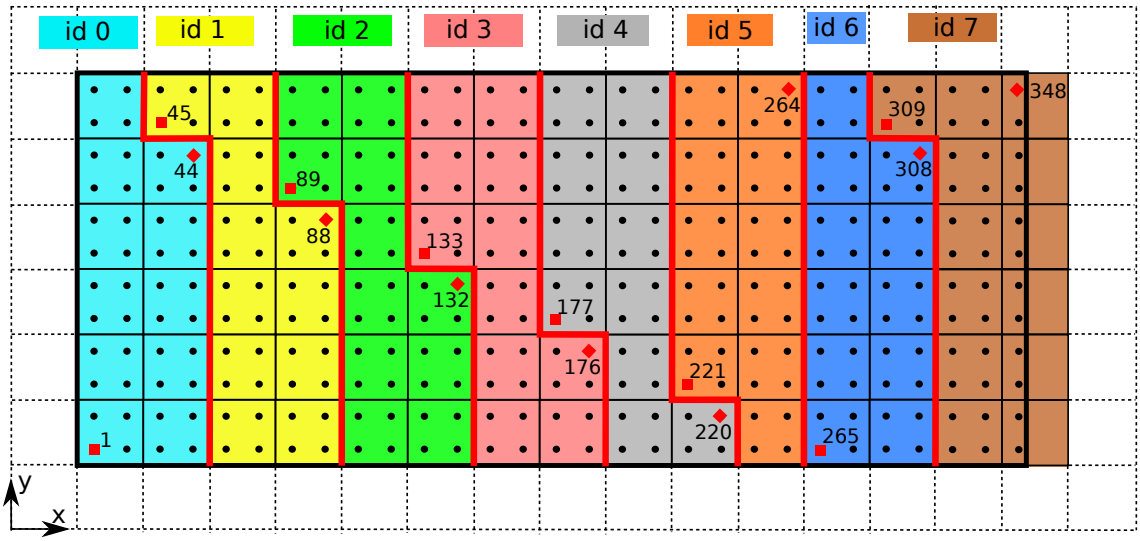
Figure 5.6: $2D$ Sketch of the distribution of the cell containing *effective* particles. $N_e = 348$. Black circles: *effective* particles; bold black line: domain boundary; black line: cells containing *effective* particles (cell types 1, 2 and 3); dashed black line: external cells (type 4); bold red lines: *parallel interfaces*; red full squares and diamond: first and last particles of each processor.

processor of Figs. 5.6 and 5.7 are summarized in Tab. 5.1.

In the following the scheme with 3 processors (shown in Fig. 5.6.b) will be used to explain the other steps of the implemented *parallel computing* procedure.



(a) $N_{procs} = 4$



(b) $N_{procs} = 8$

Figure 5.7: $2D$ Sketch of the distribution of the cell containing *effective* particles. $N_e = 348$. Symbols as in Fig. 5.6.

N_{procs}	N_t	N_0	N_1	N_2	N_3	N_4	N_5	N_6	N_7
2	174	176	172	–	–	–	–	–	–
3	116	116	116	116	–	–	–	–	–
4	87	88	88	88	84	–	–	–	–
8	44	44	44	44	44	44	44	44	40

Table 5.1: Number of particles for each processor. Examples shown in Figs. 5.6 and 5.7. $N_e = 348$.

5.2.2 Identification of the particles to be shared

After partitioning the cells containing *effective* particles, the current processor *myid* classifies as type 4 (external cells) the cells not belonging to its domain. The Fig. 5.8 shows the domains of the three processors considering the scheme in Fig. 5.6.b. Moreover, *myid* records, for each cell of the whole computational domain, the processor *id* whose the cell belongs.

The classical *SPH* formulation can be used inside the domain assigned to each processor, although, since the *support domain* of the particles can be truncated near the *parallel interfaces*, the processors must receive, and must send in turn, the particles near these regions from/to the neighboring processors.

To ease up the identification of the particles to be shared with the neighboring processors, other two types of cells are introduced (as discussed in Sec. 2.5.1): type 5 (cells whose particles must be sent to the right, dark gray cells in Fig. 5.8) and type 6 (cells whose particles must be sent to the left, light gray cells in Fig. 5.8). In order to identify these cells, the current processor *myid* checks if each cell of its domain borders:

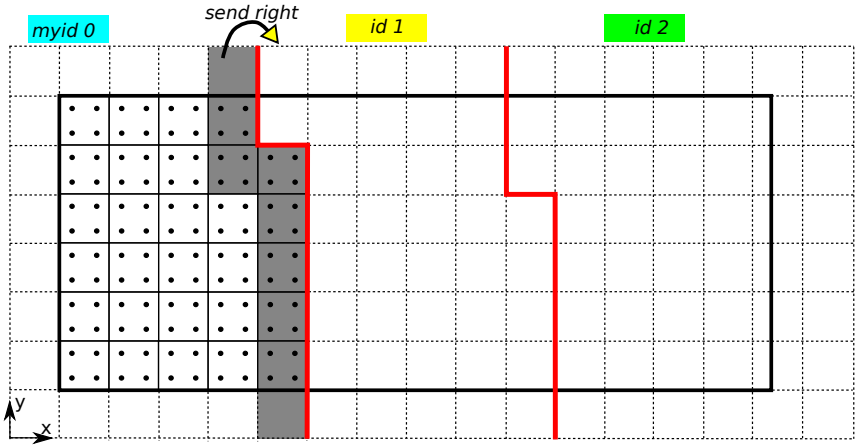
- only with its own cells, thus the cell must not be shared;
- with at least one cell of the processor $id = myid + 1$: the particles inside the cell must be shared on the right; the cell is set to type 5;
- with at least one cell of the processor $id = myid - 1$: the particles inside the cell must be shared on the left; the cell is set to type 6.

As discussed in Chap. 2, employing for $kh/\Delta x$ a value higher than 2 (which is the value chosen in this research study), a greater number of particles should be shared with the neighboring processors making the proposed *HPC* scheme less efficient.

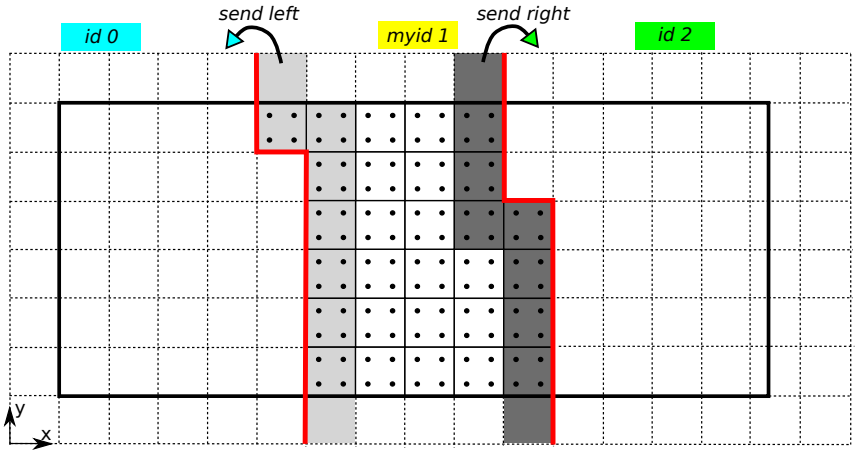
5.2.3 Sharing values procedure

The values of all the particles (*effective*, *mirror*, *IO*) lying in the cells to be shared must be sent to the left neighboring processor and/or to the right one. Simultaneously, each processor receives the corresponding values of the neighboring processors from the right and/or from the left. The received particles are indicated with PP (*parallel particles*) and the receiving processor is able to identify if each PP was an *effective* (PeP) or a *mirror* (PmP) particle in the domain of the processor from where the PP comes from.
Considering the scheme with three processors in Fig. 5.9, where the cells to be shared (types 5 and 6) are represented with cyan, yellow and green colors (for *id* $0, 1, 2$, respectively), the procedure can be summarized as follows:

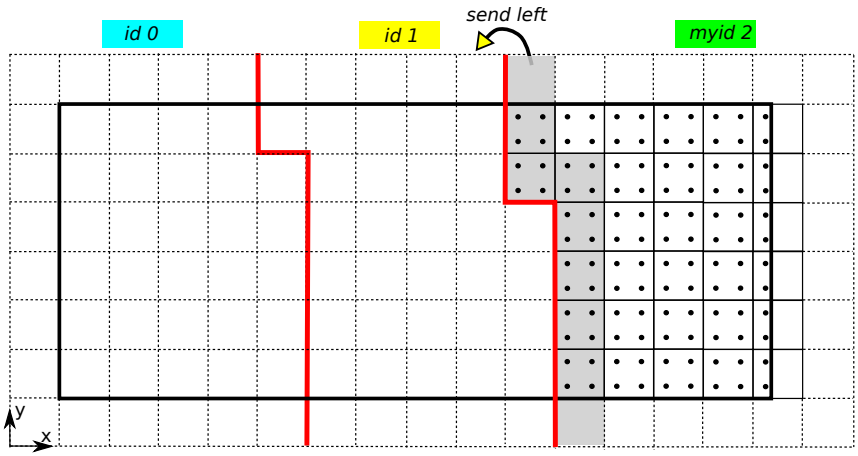
- *myid* 0 sends to *id* 1 (processor on the right) the values into its cells of type 5 (cyan cells in the figure). Simultaneously, it receives the values from *id* 1 from the right (neighboring yellow cells). The first processor (*id* 0) sends and receives to/from the right only;
- *myid* 1 sends to *id* 0 (processor on the left) the values inside its cells of type 6 neighboring to *id* 0 (yellow cells border to the first *parallel interface*) and receives the values of *id* 0 from the right (cyan cells). It sends on the right to *id* 2 the values inside the cell of type 5 neighboring *id* 2 (yellow cells border to the second *parallel interface*) and receives from the left the values of *id* 2 (green cells). The intermediate



(a)



(b)



(c)

Figure 5.8: 2*D* Sketch of the identification of the cell of types 5 (dark gray area) and 6 (light gray area). Black circles: *effective* particles belonging to the current processor; bold black line: domain boundary; Black line: cell grid contained *effective* particles (cell types 1, 2 and 3); dashed black line: external cell grid (cell type 4); bold red line: processor boundary. a) Domain of the first processor (*myid* 0); b) domain of the second processor (*myid* 1); c) domain of the third processor (*myid* 2).

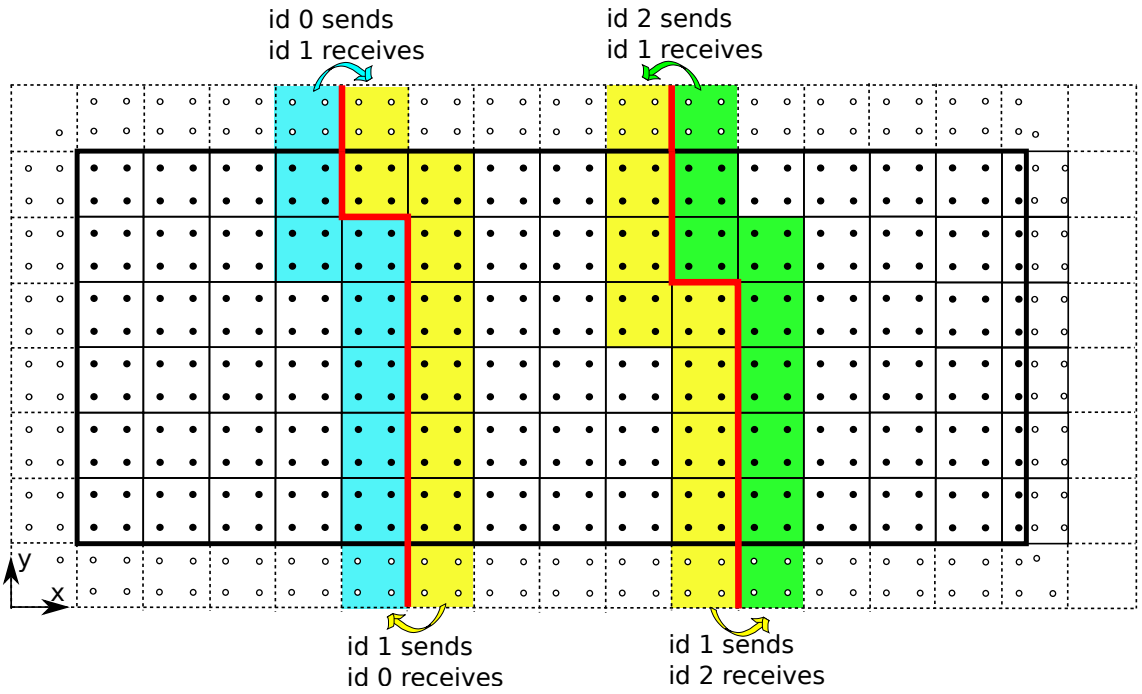


Figure 5.9: $2D$ Sketch of the sending/receiving procedure. The cells of type 5 are highlighted with the color of the processor subdomain in which they lie (cyan, yellow and green for *id* $0, 1$ and 2, respectively, as in Fig. 5.6.b); full and empty black circles: *effective* and *mirror* particles, respectively, of the whole computational domain.

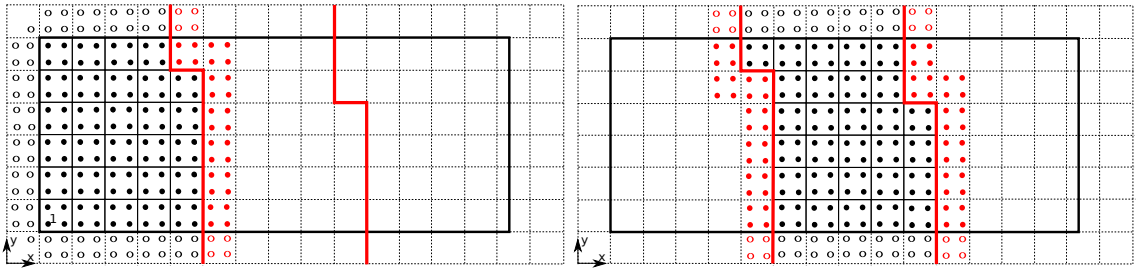
processors (such as *myid* 1 in this example) exchange values with the neighboring processors in both directions (left and right);

- *myid* 2 sends to *id* 1 on the left the values inside the green cells and receives the values of the type 6 cells of *id* 1 (yellow cells neighboring to the second *parallel interface*) from the left. The last processor (*myid* 2 in the example in the figure) sends and receives to/from the left only.

As it is seen in the figure, also the *mirror* particles are shared, therefore the sending/receiving procedure must be performed after the *mirror* particles generation (as it will be explained in Sec. 5.2.6).

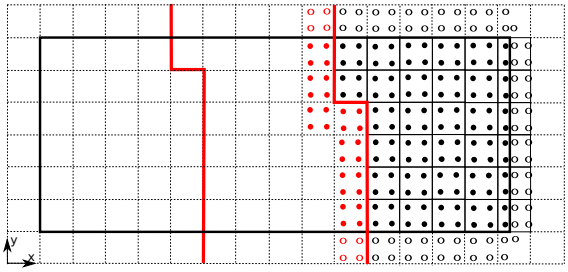
At the beginning of each time step, the positions of the particles inside the cells of types 5 and 6 are shared and each processor adds the PP particles inside its own cells (as shown in Fig. 5.10). The PP particles are numbered after the *effective* and *mirror* particles. Therefore, the first PP particle has the index $N_{(id)} + N_{mirror,id} + 1$ (where $N_{(id)}$ and $N_{mirror,id}$ are the number of *effective* and *mirror* particles, respectively, in the domain of the processor), while the last one has the index $N_{(id)} + N_{mirror,id} + N_{PP,tot}$ (where $N_{PP,tot}$ is the total number of *parallel* particles as the sum of those received from the left and from the right).

Each processor builds the vectors of the values to be shared on the right (values of the particles inside the cells of type 5) and on the left (values of the particles inside the cells of type 6). The values to be shared are: positions, *intermediate* and *corrected* velocities,



(a) Subdomain of *myid* 0

(b) Subdomain of *myid* 1



(c) Subdomain of *myid* 2

Figure 5.10: $2D$ Sketch of the processor domain after sharing the particle positions. Full and empty black circles: *effective* and *mirror* particles, respectively, of the processor *myid*; full and empty red circles: *effective* and *mirror* particles, respectively, received from the neighboring processors (PP particles).

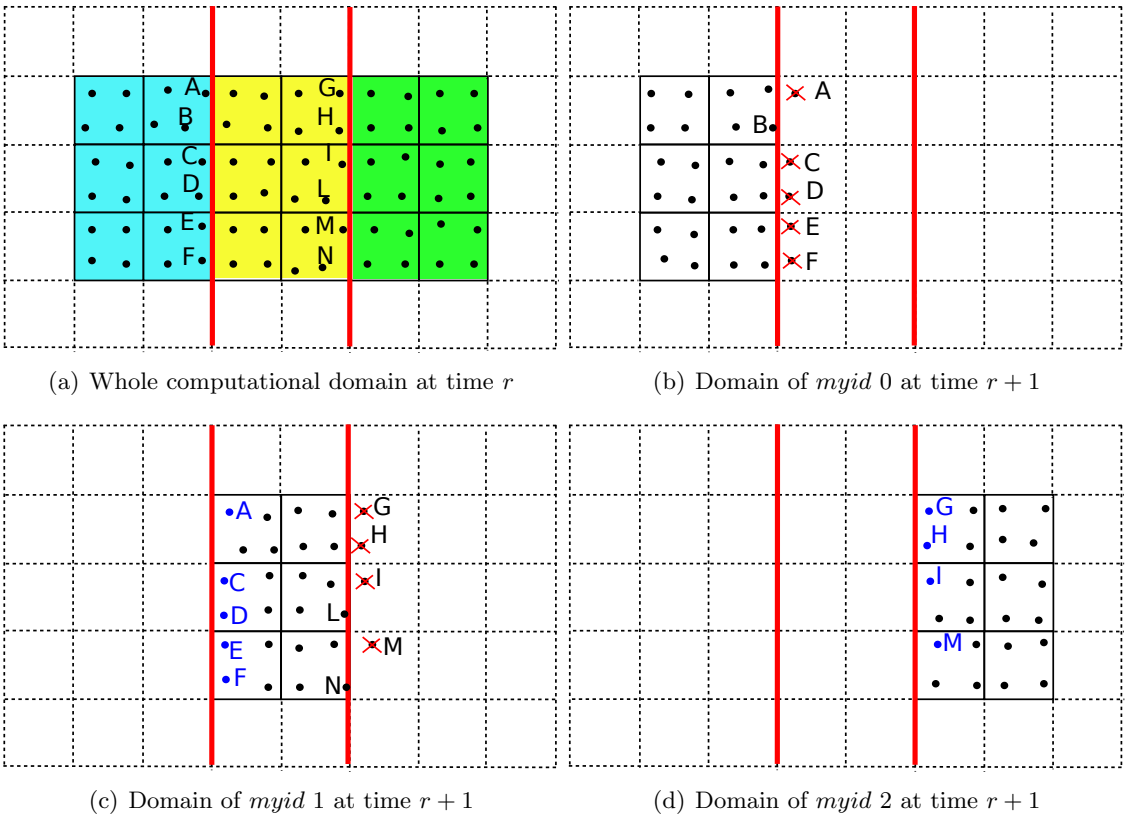
pseudo-pressure, value $(u_n^{k+1} - u_n^*)\big|_b d_{gj}$ (to be used in the *Poisson right-and-side* of eqn. 5.4, see Sec. 5.2.5) and the tracer concentrations (see Sec. 7.2.2).

It should be noted that for sending array, for example the positions, the first particles to be share fills the first three elements of the sending vector, the second particles starts from the fourth element to the sixth one, and so on for the others.

5.2.4 Management of the particle leaving/entering the processor domain

As already discussed, due to the Lagrangian nature of the *SPH* method, the particles leaving and entering the domain assigned to each processor have to be dealt with. To this aim, at the end of each time step the processors check if some of their own *effective* particles have left their domain crossing the *parallel interfaces*. The particles leaving the domain of the processor are deactivated and are added in a local storage list (each processor has a list of the deactivated particles that it is the same of that explained in Chap. 3 and Chap. 4 for the particles deactivated through *outflow BCs* and *block interfaces*, respectively). The particles deactivated by the current processor *myid* can belong now to cells of the neighboring processors on the right ($id = myid + 1$) or on the left ($id = myid - 1$). Therefore, the current processor builds two vectors (one for sending the information to the processor on the right and one to the processor on the left), adding the required values of these particles. Specifically, the information to be sent are: positions, velocities, accelerations at the current time step (in order to use the *Adams-Bashforth* scheme in the *predictor-step* at the next time step), *pseudo-pressure*, concentration of the analyzed species (see Sec. 7.2.3), *platelet activation potential* (see Sec. 7.4). The receiving processor records each new particle inside its own cell to which the particle belongs and gives to the new particle

a local index picking up it from its list of deactivated particles or increasing the number of its *effective* particles if the aforementioned list is empty.



(a) Whole computational domain at time r

(b) Domain of *myid* 0 at time $r+1$

(c) Domain of *myid* 1 at time $r+1$

(d) Domain of *myid* 2 at time $r+1$

Figure 5.11: $2D$ Sketch of the particle leaving/entering the processor domain. Color of the processor subdomain as in Fig. 5.6.b. Black circles: *effective* particles; blue circles: new particles; black circles with red cross: deactivated particles; bold red lines: *parallel interfaces*; black continuous lines: cells containing *effective* particles; black dashed lines: external cells.

The Fig. 5.11 shows a simple computational domain partitioned into three processors. In Fig. 5.11.a the whole domain is represented at time r. Some *effective* particles close to the *parallel interfaces* are highlighted: the particles A, B, C, D, E, F of the first processor (cyan area in the figure) and the particles G, H, I, L, M, N of the second processor (yellow area in the figure). The Figs. 5.11.b,c,d show the domain of the processors *id* $0, 1$ and 2, respectively, after one time step (time $r+1$). For the first processor (Fig. 5.11.b), the particles A, C, D, E and F have crossed the first *parallel interface* and thus they are deactivated (in the figure the particles deactivated are marked with a red cross). While the particle B not having crossed the *parallel interface* stays, for the current time instant, in the domain of the first processor. The current processor *myid* 0 identifies the *id* of the processor to which these particles belong in the new position. To this aim, the cell of each deactivated particle is calculated through eqn. 2.14 and the processor owning this cell is identified. For example, the particle A at time $r+1$ belongs to a cell of $id = myid\ +1$. In this case all the particles deactivated belongs now to cells of the processor *id* 1 (as explained in Sec. 5.2.3 the first processor can sends/receives particles only to/from the

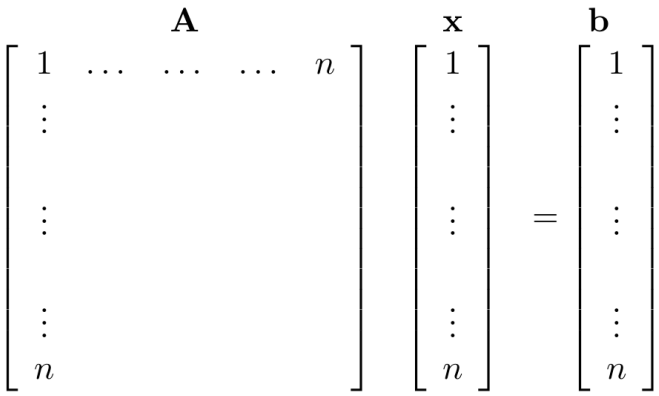


Figure 5.12: Scheme of the *PPE* linear system $\mathbf{A}\,\mathbf{x} = \mathbf{b}$ in *serial mode*. $\mathbf{A}$ is a matrix [n x n] with $n = N_e$ while $\mathbf{x}$ and $\mathbf{b}$ are the vector solution and the vector of known terms, respectively, of length n.

right). Therefore, the current processor *myid* 0 builds a vector to send these particles on the right, adding the three components of the position, the three components of the velocity, the acceleration and the *pseudo-pressure* of the first particle to be sent (the particle A in the figure) and, in the vector queue, the same values of the other particles (particle C, D, E and F in figure). These particles are received by the processor on the right, as shown in Fig. 5.11.c where the new particles are represented by the blue color. Simultaneously, the second processor deactivates the particles crossing the second *parallel interface* (particles G, H, I, M) and identifies the processor to which they must be sent: the processor on the right (*id* 2) in this example. As explained for the first processor, *myid* 1 creates a vector with the values of these deactivated particles. It should be noted that, since *myid* 1 is an intermediate processor (preceded by *id* 0 and followed by *id* 2), it could send *effective* particles on the right and left simultaneously. Specifically, if any particle (not existing in the scheme in the figure) passes through the first *parallel interface*, the current processor must send it to *id* 0, building a second vector containing the values of the particles to be sent on the left. The processor *myid* 2 receives the values of the particles G, H, I, and M as shown in Fig. 5.11.d. The last processor can send/receive particles to/from the left only.

5.2.5 The equation *Poisson* system in *parallel computing*

As explained in Sec. 5.1, the *parallel computing* implementation in the *ISPH* approach is challenging since it is necessary to solve the *PPE* system made of N_e equations: a eqn. 2.21 for each *effective* particle of the computational domain. In the implemented *parallel computing* algorithm, the whole *Poisson* linear system, explained in Sec. 2.7 and shown in Fig. 5.12, is partitioned among the processors.

Specifically, each processor must solve a new linear system $\mathbf{A}\,\mathbf{x} = \mathbf{b}$ represented in Fig. 5.13, where

- the coefficient matrix $\mathbf{A}$ has dimensions [n x m] where n is the number of *effective* particles in the current processor $N_{(myid)}$ and m is the sum of $N_{(myid)}$ and the *effective* particles received from the neighboring processors $N_{PeP,tot}$ ($m = N_{(myid)} + N_{PeP,tot}$). The *parallel effective* particles are in turn the sum of those received from

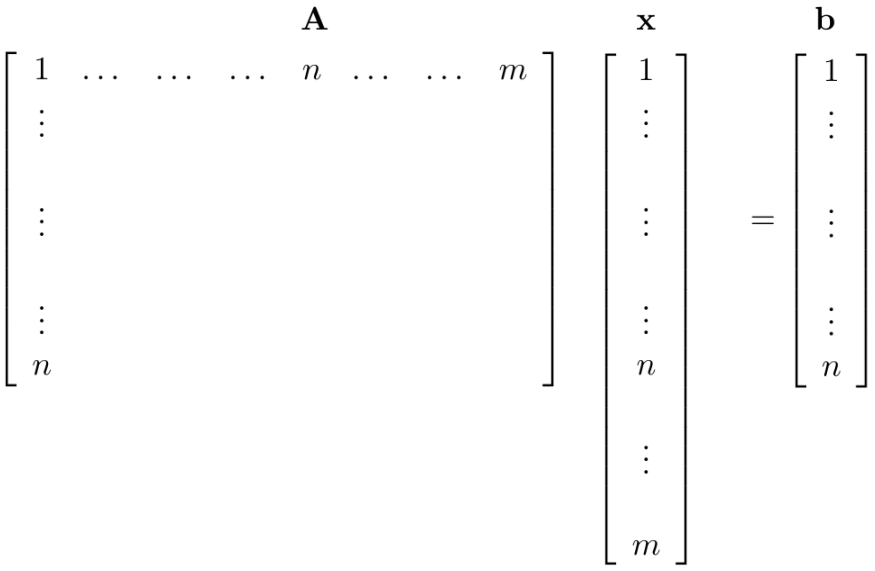


Figure 5.13: Scheme of the *PPE* linear system $\mathbf{A}\,\mathbf{x} = \mathbf{b}$ of each processor in *parallel computing*. $\mathbf{A}$ is a matrix [n x m] with $n = N_{(myid)}$ and $m = N_{(myid)} + N_{PeP,tot}$; the vector $\mathbf{x}$ has length m while the vector $\mathbf{b}$ has length n.

the left N_{PeP^L} and those received from the right N_{PeP^R} ($N_{PeP,tot} = N_{PeP^L} + N_{PeP^R}$). For each matrix row:

- from the column 1 to the column $N_{(myid)}$ the values of its own *effective* particles are stored

$$1 \leq col \leq N_{(myid)}, \text{ for } \textit{effective} \text{ particles}$$

- from the column $N_{(myid)} + 1$ to the column $N_{(myid)} + N_{PeP^L}$ the values of the *parallel effective* particles received from the left are stored

$$N_{(myid)} + 1 \leq col \leq N_{(myid)} + N^L_{PeP}, \text{ for } PeP^L$$

- from the column $N_{(myid)} + N_{PeP^L} + 1$ to the column $N_{(myid)} + N_{PeP^L} + N_{PeP^R}$ the values of the *parallel effective* particles received from the right are stored

$$N_{(myid)} + N^L_{PeP} < col \leq N_{(myid)} + N^L_{PeP} + N^R_{PeP}, \text{ for } PeP^R$$

The i-th row *diagonal term* of the coefficient matrix eqn. 2.29 becomes

$$\sum_{j=1}^{N_i'} C_{ij} + \sum_{j=1}^{N_i^{PP}} C_{ij} \tag{5.2}$$

where N_i^{PP} is the number of *parallel* particles (sum of the *parallel effective* and *parallel mirror* particles, PeP and PmP, respectively) in Ω_i, while the other symbols are known.

The i-th row *off-diagonal* term in the s-th column with $s \leq N_{(myid)}$ of the system coefficient matrix is equal to eqn. 2.30 since s is an *effective* particle. Otherwise, if $s \geq N_{(myid)}$, s is a *parallel effective* particle; therefore, eqn. 2.30 must be modified

substituting the term N_i^{Ms} with the N_i^{PmPs} (which is the number of *parallel mirror* particles PmP into Ω_i generated by the *parallel effective* particle s).

$$-\left(\delta_{is}C_{is}+\sum_{j=1}^{N_i^{PmPs}}C_{ij}\right) \quad \text{with } s \geq N_{(myid)} \tag{5.3}$$

- the vector solution $\mathbf{x}$ is extended to the *effective* particles received from the neighboring processors $N_{PeP,tot}$. Thus it has length $m = N_{(myid)} + N_{PeP,tot}$ and the values from the positions $n+1$ to m are received from the neighboring processors. Differently from the procedure explained in Sec. 5.2.3, here only the values of the *effective* particles inside the cells of type 5 and 6 must be sent that are stored in the vectors $\mathbf{GC}^R$ and $\mathbf{GC}^L$, respectively. The solution value of each of these particles is searched in the $\mathbf{x}$ vector at the position of the particle column $x(col)$ (a value that can span between 1 and n since it is an *effective* particle) and it is stored in the arrays $\mathbf{GC}_x^R$ or $\mathbf{GC}_x^L$ (depending on if the particle comes from the vector $\mathbf{GC}^R$ or from the vector $\mathbf{GC}^L$). The receiving processor fills its $\mathbf{x}$ vector starting from the position $n+1$, placing the values received from the left (PeP^L) followed by those received from the right (PeP^R). Obviously, the first processor has only the values received from the right (PeP^R), while the last processor has only those received from the left (PeP^L).

- the vector of known terms $\mathbf{b}$ has length equal to $n = N_{(myid)}$.

  The i-th equation *right-hand-side* term eqn. 2.31 can be rewritten adding the sum of the PmP lying in Ω_i (N_i^{PmP})

$$RHS_i = T_i + \frac{1}{\Delta t}\sum_{j=1}^{N_i^{(M+PmP)}} C_{ij}\,(u_n^{k+1} - u_n^*)\Big|_b\, d_{gj} \tag{5.4}$$

  where the *mirror* and PmP particles in Ω_i are written in the compact form $N_i^{(M+PmP)}$. Moreover, the PeP particles are used in eqn. 2.28 to calculate T_i.

The *BiCGSTAB* method (used for iteratively solving the *PPE* system as explained in Sec. 2.7.1) has been entirely parallelized. The ALGORITHM 2.1 (or ALGORITHM 2.3 considering the preconditioned version) has been modified in the *parallel computing* scheme considering the non-symmetric linear system of Fig. 5.13. For *parallel computation*, the vectors $\mathbf{x}$, $\mathbf{x}_0$, $\mathbf{s}$, $\mathbf{p}$, $\mathbf{y}$ and $\mathbf{z}$ of the *BiCGSTAB* algorithm are extended to the total number of *effective* particles received from the neighboring processor (PeP, *parallel effective* particles). The dimension of these vectors is thus equal to the total number of *effective* particles of the current processor *myid*, $N_{(myid)}$, plus the number of *effective* particles received from the neighboring processors as sum of left and right ($N_{PeP,tot} = N_{PeP}^L + N_{PeP}^R$). As for the *serial mode*, the dimension of the vectors $\mathbf{r}_0$, $\mathbf{r}$, $\mathbf{b}$, $\mathbf{v}$, $\mathbf{t}$ is equal to the number of equations $N_{(myid)}$ (since the equations are written for each *effective* particle of the processor domain). The new *Pre-BiCGSTAB* algorithm is shown below.

ALGORITHM 5.1- Parallel Pre-BiCGSTAB method

---

1. *Send* $\mathbf{GC}_{x_0}^R$ and $\mathbf{GC}_{x_0}^L$ / *Receive* $\mathbf{x}_{0\,(n+1:m)}$

2. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\,\mathbf{x}_0$
3. Choose $\mathbf{r}_0^*$ such that $(\mathbf{r}_0^*, \mathbf{r}_0) \neq 0$. For instance $\mathbf{r}_0^* = \mathbf{r}_0$;
4. $\rho_0 = \alpha_0 = \omega_0 = 1$;
5. $\mathbf{v}_0 = \mathbf{p}_0 = \mathbf{0}$;
6. The iterative cycle is performed until convergence ($RSQ < tol$):

6.0 *do* $j = 1, \ldots$ *until convergence*

6.1 $\rho_{j,myid} = (\mathbf{r}^*_{0\,(1:n)}, \mathbf{r}_{j-1\,(1:n)})$;
call $MPI\_ALLREDUCE(\rho_{j,myid}, \rho_j)$;

6.2 $\beta_j = \left(\dfrac{\rho_j}{\rho_{j-1}}\right)\left(\dfrac{\alpha_{j-1}}{\omega_{j-1}}\right)$;

6.3 $\mathbf{p}_{j\,(1:n)} = \mathbf{r}_{j-1\,(1:n)} + \beta\left[(\rho_{j-1} - \omega_{j-1}\,\mathbf{v}_{j-1\,(1:n)}\right]$;

6.4 $\mathbf{y}_{(1:n)} = \mathbf{K}^{-1}\mathbf{p}_{j\,(1:n)}$ ($solving\ \mathbf{y}' = \mathbf{L}^{-1}\mathbf{p}_j\ and\ \mathbf{y} = \mathbf{U}^{-1}\mathbf{y}'$)
Send $\mathbf{GC}_y^R$ and $\mathbf{GC}_y^L$ / *Receive* $\mathbf{y}_{(n+1:m)}$

6.5 $\mathbf{v}_j = \mathbf{A}\,\mathbf{y}$;

6.6 $\alpha^*_{j,myid} = (\mathbf{r}_0^*, \mathbf{v}_j)$
call $MPI\_ALLREDUCE(\alpha^*_{j,myid}, \alpha^*_j)$
$\alpha_j = \dfrac{\rho_j}{(\alpha^*_j)}$;

6.7 $\mathbf{s}_{(1:n)} = \mathbf{r}_{j-1\,(1:n)} - \alpha_j \mathbf{v}_{j\,(1:n)}$;

6.8 $\mathbf{z}_{(1:n)} = \mathbf{K}^{-1}\mathbf{s}$ ($solving\ \mathbf{z}' = \mathbf{L}^{-1}\mathbf{s}\ and\ \mathbf{z} = \mathbf{U}^{-1}\mathbf{z}'$)
Send $\mathbf{GC}_z^R$ and $\mathbf{GC}_z^L$ / *Receive* $\mathbf{z}_{(n+1:m)}$

6.9 $\mathbf{t} = \mathbf{A}\,\mathbf{z}$;

6.10 $\omega^*_{j,myid} = (\mathbf{t}_{(1:n)}, \mathbf{s}_{(1:n)})$;
call $MPI\_ALLREDUCE(\omega^*_{j,myid}, \omega^*_j)$;
$\omega^{**}_{j,myid} = (\mathbf{t}_{(1:n)}, \mathbf{t}_{(1:n)})$;
call $MPI\_ALLREDUCE(\omega^{**}_{j,myid}, \omega^{**}_j)$;
$\omega_j = \dfrac{\omega^*_j}{\omega^{**}_j}$;

6.11 $\mathbf{x}_{j\,(1:m)} = \mathbf{x}_{j-1\,(1:m)} + \alpha_j\,\mathbf{y}_{(1:m)} + \omega_j\,\mathbf{z}_{(1:m)}$;

6.12 $\mathbf{res}_{(1:n)} = \mathbf{b}_{(1:n)} - \mathbf{A}\,\mathbf{x}_j$;

6.13 $RSQ_{myid} = \left[\mathbf{res}_{(1:n)}, \mathbf{res}_{(1:n)}\right]$;
call $MPI\_ALLREDUCE(RSQ_{myid}, RSQ)$;

6.14 *Send* $\mathbf{GC}_x^R$ and $\mathbf{GC}_x^L$ / *Receive* $\mathbf{x}_{(n+1:m)}$

6.15 *Check if convergence is reached:*
if $(RSQ < tol)$ *then quit.*
else $\mathbf{r}_{j\,(1:n)} = \mathbf{s}_{(1:n)} - \omega_j\,\mathbf{t}_{(1:n)}$ *and continue.*

---

where the *sending/receiving* actions (points 1, 6.4, 6.8 and 6.14) are highlighted with

blue color. A local FORTRAN subroutine ("*SPH_Share_GC*") has been implemented using the *MPI* function *MPI_SENDRECV* for sending and receiving vectors during the iterative procedure. Specifically, the subroutine needs as input the vectors $\mathbf{GC}_x^R$ and $\mathbf{GC}_x^L$ containing, depending on the vector to be shared, the value of x_0 ($\mathbf{GC}_{x_0}^R$ and $\mathbf{GC}_{x_0}^L$), y ($\mathbf{GC}_y^R$ and $\mathbf{GC}_y^L$), z ($\mathbf{GC}_z^R$ and $\mathbf{GC}_z^L$) or x ($\mathbf{GC}_x^R$ and $\mathbf{GC}_x^L$), as appropriate, of the *effective* particles to be shared. These vectors are created as explained above for the vector solution and are sent to the neighboring processors. The receiving processor puts these values in the corresponding vector ($\mathbf{x}_0$, $\mathbf{y}$, $\mathbf{z}$ or $\mathbf{x}$) from the position $n+1$ up to m starting from the values received from the left. The matrix-vector multiplications ($\mathbf{A}$ $\mathbf{x}$) in points 2, 6.5, 6.9, 6.12 of the algorithm above, are possible since the number of columns of the matrix $\mathbf{A}$ are equal to the number of elements in the vector $\mathbf{x}$ (length $m = N_{(myid)} + N_{PeP,tot}$). Each processor performs the $\mathbf{A}$ $\mathbf{x}$ product after having received the values from the neighboring processors. In the point 6.1 the ρ value is calculated separately from each processor (obtaining $\rho_{(myid)}$) using their own $\mathbf{r}$ vector.
Through the *MPI* function *MPI_ALLREDUCE* the value of each processor is added to that of the others in order to obtain the total value ($\rho_{(myid)}$). The same procedure is used at points 6.6, 6.10 and 6.13 to calculate α, ω and RSQ, respectively.

5.2.6 Flow chart of the *PANORMUS-SPH* code

The Fig. 5.14 shows the flow chart of the *SPH* code in *parallel computing* and for the *Single-Domain* approach. The steps of the flow chart, explained below, are performed by each processor with the exception of the *ACTION* 2 that is only performed by the first processor at the beginning of the simulation.

- ***ACTION*** 1: The current processor reads the particle starting file and the boundary triangles file as described in Sec. 2.10;

- ***ACTION*** 2: The processor *id* 0 creates the virtual grid and prints in a file (which is named *sph_initialize.inp*) the cell type classification (it is performed at the beginning of the simulation only). The other processors read that file;

- ***ACTION*** 3: The particles are distributed among the selected number of processor N_{procs} reading their coordinates through the starting file (*ACTION* 1) until the required theoretical number (N^t) is reached (as described in Sec. 5.2.1). The cells in which these particles lie are assigned to the current processor. All the remaining particles inside the last cell assigned (res) belong to the current processor, thus the real number of particles of each *myid* can be different from N^t ($N_{(myid)} = N^t + res$). Therefore, in this step each processor identifies its own cells. Moreover, the current processor knows the processors *id* to which each cell of the whole computational domain belongs. The cells belonging to other processors are set to type 4. As explained in Sec. 5.2.2, the processor identifies the cells of type 5 whose values must be shared to the right (with the processor $myid+1$) and those of type 6 to be shared to the left (with the processor $myid-1$). The cells of type 4 neighboring to the ones of type 5 are set to type 5 (the same is for the cells neighboring the ones of type 6) since the *mirror* values must be shared too;

- ***ACTION*** 4: The processor generates the *mirror* particles starting from its own *effective* particles close to the domain boundaries, as described in Sec. 2.5.1 for the *serial mode*;

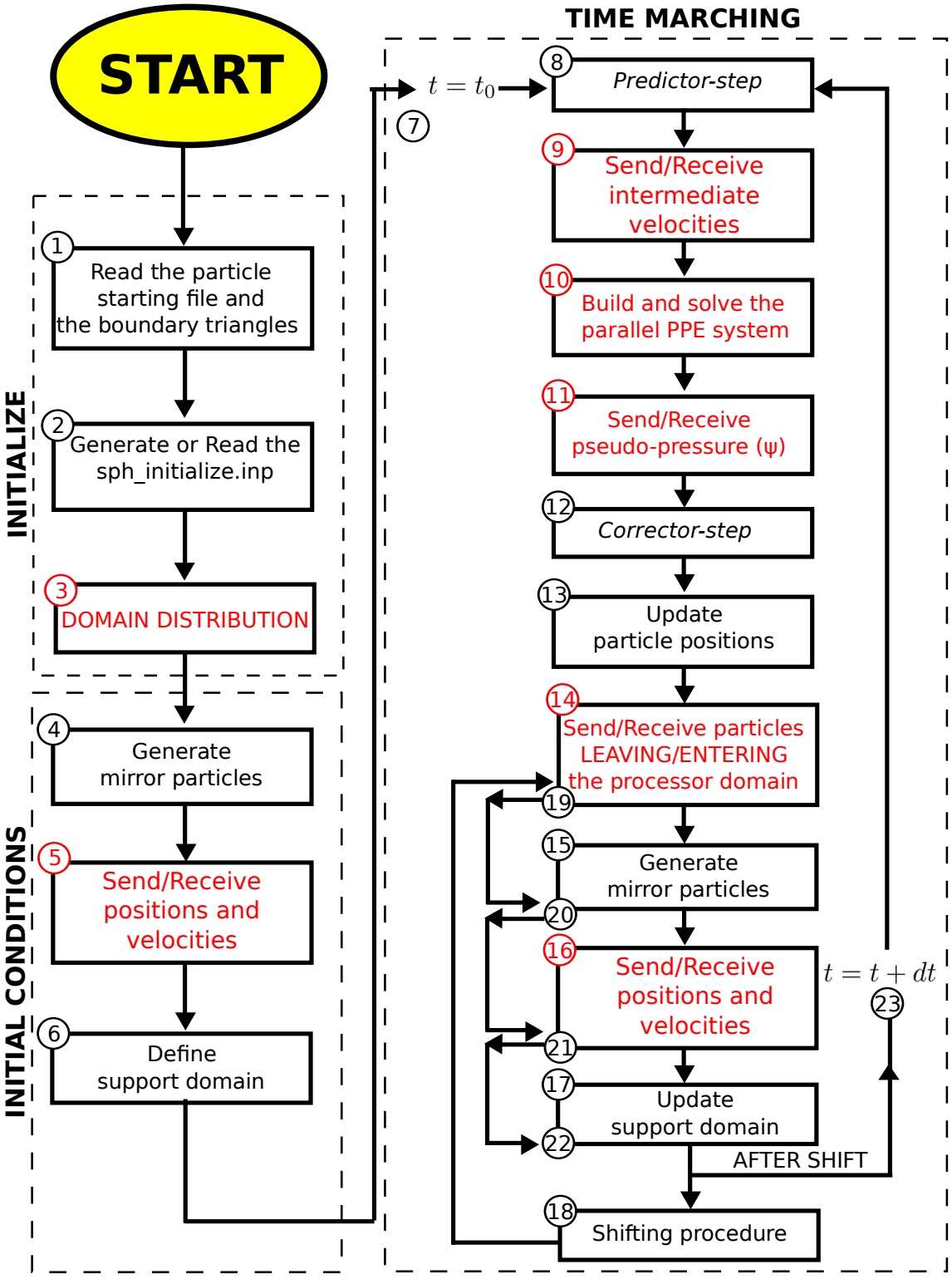


Figure 5.14: Flow chart of the *PANORMUS-SPH* code with the *parallel SD* procedure. The *parallel computing* actions are highlighted with red color.

- ***ACTION*** 5: The processor sends positions and velocities of all the particles (*effective* and *mirror*) inside the cells of types 5 and 6. It receives, simultaneously, the

values inside the cells of types 5 and 6 of the neighboring processors. The received particles are named *parallel* particles (PP) and the processor is able to identify if the PP is an *effective* (PeP) or a *mirror* (PmP) particle in the domain of the sending processor;

- ***ACTION*** 6: For each *effective* particle i belonging to the current processor, the *support domain* is identified storing in the list of i the particles (*effective*, *mirror*, IO, PP) having distance from i shorter than kh;
- ***ACTION*** 7: The time marching procedure starts from the initial time $t = t_0$;
- ***ACTION*** 8: In the *predictor-step* the *intermediate* velocities $\mathbf{u}_i^*$ are calculated for the *effective* particles of the current processor through eqn. 2.20 (or eqn. 2.37 if the time step is variable as explained in Sec. 2.9). For the *mirror* particles the u_n^* is calculated at the $\mathbf{x}_b$ position using eqn. 2.23;
- ***ACTION*** 9: The processor sends the values of the *intermediate* velocities of the particles inside the cells of type 5 and 6. Simultaneously, it receives the values inside the cells of type 5 and 6 of the neighboring processors;
- ***ACTION*** 10: The value $(u_n^{k+1} - u_n^*)\big|_b d_{gj}$ (to be used to build the *Poisson right-and-side*, eqn. 5.4) is shared. The coefficient matrix is built in the CRS format creating the vectors **vals**, **cols** and **limits**. The *Pre-BiCGSTAB* method is performed until convergence following the ALGORITHM 5.1;
- ***ACTION*** 11: The ψ values are shared with neighboring processors;
- ***ACTION*** 12: In the *corrector-step* the *corrected* velocities $\mathbf{u}_i^{k+1}$ of the *effective* particles of the current processor are calculated using eqn. 2.24;
- ***ACTION*** 13: The processor updates the positions of its own *effective* particles through eqn. 2.25;
- ***ACTION*** 14: At the end of the time step the current processor *myid* checks if any of its *effective* particles have left its domain through *parallel interfaces*. The particles crossing the *parallel interfaces* are deactivated for the processor and are added in its storage list. On the other hand, the processor receives the *effective* particles (positions, velocities, *pseudo-pressure* and accelerations) which have left the domain of the neighboring processors and now lie in cells belonging to *myid* as explained in Sec. 5.2.4;
- ***ACTION*** 15: As in action 4;
- ***ACTION*** 16: After moving the *effective* particles and generating the *mirror* ones, the processor sends the values of the positions and velocities of the particles inside the cells of type 5 and 6 and, simultaneously, it receives the values inside the cells of type 5 and 6 of the neighboring processors (as in *ACTION* 5);
- ***ACTION*** 17: As in *ACTION* 6;
- ***ACTION*** 18: The shifting procedure is performed as explained in Sec. 2.8;
- ***ACTION*** 19: As in *ACTION* 14;

- ***ACTION*** 20: As in *ACTION* 4;
- ***ACTION*** 21: As in *ACTION* 5;
- ***ACTION*** 22: As in *ACTION* 6;
- ***ACTION*** 23: The simulation time is advanced by one time step ($t = t + dt$). If the adaptive time step procedure is activated it must be checked if the *Courant* limit checking is satisfied (eqn. 2.35) or if it is necessary to change the time step as discussed in Sec. 2.9. In eqn. 2.36 the $\overline{u}_{max}$ is the highest velocity value among all processors, and it is obtained through the *MPI* function $MPI\_ALLGATHER$.

After the *ACTION* 23, the procedure is restarted with the *predictor-step* (*ACTION* 8).

5.2.7 Scalability test

A cylindrical pipe of length $L = 0.012\ m$ and diameter $D = 0.001\ m$ has been considered in order to perform a scalability test. A parabolic velocity profile has been imposed at the inlet, zero pressure has been set at the outlet section (the *incoming* and *pressure BCs*, respectively, described in Chap. 3) and *adherence BCs* have been adopted at the lateral walls.

The *smoothing length* h has been set to $2.5\,10^{-5}\ m$, corresponding to an initial number of *effective* particles $N_e = 606\,720$.

The time required to execute one time step in the *time marching* procedure has been calculated using the *serial mode* ($N_{procs} = 1$), and the *parallel computing* with 2, 4, 8, 16 and 32 processors having 303 360, 151 680, 75 840, 37 920 and 18 960 initial number of *effective* particles, respectively. The results (red stars) and the *trend-line* (dashed black line) whose slope is equal to -0.796 are plotted in Fig. 5.15 using a double logarithmic scale.

As it is seen in the figure a very good scalability has been obtained, although in order to have a perfect *linear scalability* the slope of *trend-line* should be -1 [1].

5.3 SPH-HPC for the *Multi-Domain* approach

The *parallel computing* algorithm has been further extended to the *Multi-Domain* approach.

[1] This can be obtained through trivial algebra. The theoretical time of each processor T_{id} can be expressed as

$$T_{id} = \frac{T_{serial}}{N_{procs}}$$

where T_{serial} is the time required in *serial mode* when $N_{procs} = 1$. Thus T_{serial} is a constant

$$T_{id}\, N_{procs} = T_{serial} = cost$$

Applying the logarithm operator to both the side of the previous equation

$$log(T_{id}) + log(N_{procs}) = log(T_{serial})$$

Moving the term $log(N_{procs})$ to the right-and-side

$$log(T_{id}) = -log(N_{procs}) + log(T_{serial})$$

where $f(x) = log(T_{id})$, $x = log(N_{procs})$, $b = log(T_{serial})$ is the intercept and the slope of the line is -1.

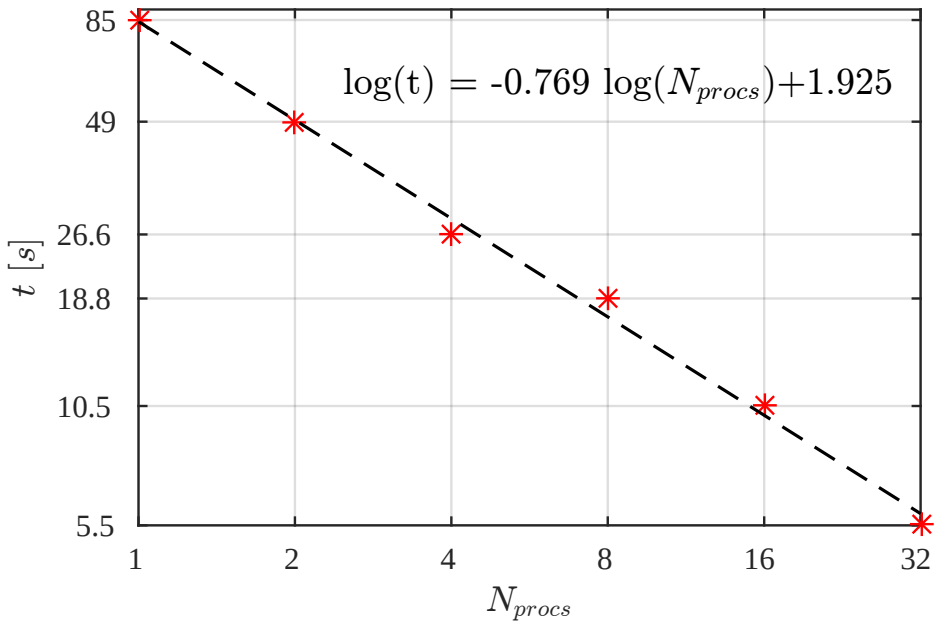


Figure 5.15: *Scalability test*: *Parallel SD*. $N_e = 606\,720$; t is the *CPU* time and N_{procs} is the number of processors. Red stars: time required using different N_{procs}; dashed black line: *trend-line*.

As explained in Chap. 4, in the *Multi-Domain* approach the computational domain is partitioned into blocks (numbered from 1 to N_{Blocks}) separated through *block interfaces* (see Sec. 4.2.1). The matching of the solution in neighboring subdomains is obtained generating at the *block interfaces* a new type of ghost particles named *IP* particles (see Sec. 4.2.3).

In one such approach, the *HPC* implementation becomes extremely complex due to the difficulty on balancing among processors the particles belonging to different blocks separated by *block interfaces* which, furthermore, can be in general shared among different partitions. Therefore, the *parallel computing* scheme must take into account both the handling of different blocks with their own kh value as well as the interaction of several processors.

5.3.1 Domain distribution

In the *Multi-Domain* approach the total number of *effective* particles in the whole computational domain $N_{e,tot}$ is obtained summing the *effective* particles N_{e,B_n} of each block B_n (eqn. 4.1). Each block B_n has a own virtual grid of side kh_{B_n} (*smoothing length* of the block B_n) where the *effective* particles are numbered following the scheme of Fig. 5.3 separately for each block (the numbering starts again from the particle number 1).

In the *domain distribution* step, the cells of the whole computational domain, as sum of the cells of the virtual grid of each block containing *effective* particles, are distributed among the selected number of processors. These cells are distributed in order to allot to each processor a number of particles close to the theoretical one, which is equal to $N^t = N_{e,tot}/N_{procs}$.

All the processors know the virtual grid of each block, reading it from a file at the beginning of the simulation.

The current processor *myid* identifies the cells to be assigned to each processor. For each cell of the computational domain a variable, named *cell_processor*, containing the *id* of the processor of the selected cell is calculated. The procedure starts from the first processor ($id = 0$) and the first block ($B_n = 1$).

Each processor performs the algorithm described below.

ALGORITHM 5.2-Domain distribution in MD

---

1. $cell\_processor(:,:,:) = -10$
2. *do* $id = 0, N_{procs} - 1$
3. *do* $n = 1, N_{Blocks}$
4. SEARCH the first particles (P_S)
5. COUNT the particles from P_S to P_E
6. SET $cell\_processor = id$ for the cells of these particles
7. if $id = myid$
   Add these particles to the particle list of the block n
8. Different CASES may occur:

8.1 if CASE 1: $N^t = N^a_{Bn}$

8.2 if CASE 2: $N^t < N^a_{Bn}$, $N^R = N^a_{Bn} - N^t$
   if CASE 2.1: $N^R < tol$
   if CASE 2.2: $N^R > tol$

8.3 if CASE 3: $N^t > N^a_{Bn}$, $N^R = N^t - N^a_{Bn}$
   if CASE 3.1: $N^R < tol$
   if CASE 3.2: $N^R > tol$

---

1. At the beginning no cell has been assigned and the variable *cell_processor* is initialized to the value -10 for all the cells of the computational domain;
2. The cycle on the processor is performed;
3. The current processor *myid* scans, for each processor *id*, all the blocks starting from $n = 1$;
4. *Myid* starts to count the required number of particles from the first particle P_S lying in the first cell not yet assigned whose variable *cell_processor* is still equal to -10;
5. The particles are counted until the theoretical number N^t is reached for the processor *id*. The last particles counted is named P_E;
6. The *cell_processor* variable is set to *id* for the cells to which the counted particles (starting from P_S) belong;
7. When $myid = id$ the counted particles are recorded in the particle list of the processor of the corresponding block;

8. Three main cases may occur. In cases 2 and 3 a tolerance value is used in order to prevent that a processor manages a block with an extremely low number of particles.

   - CASE 1: If the theoretical number of particles is equal to the available particles in the block N^a_{Bn} (corresponding to the particles whose cells are not yet assigned to any processor) the cycle on the processor id is ended. The cycle starts again at point 3 with the processor $id+1$;
   - CASE 2: The processor needs less particles than the available number in the block n. The residual number of particles is $N^R = N^a_{Bn} - N^t$. The cases 2.1 and 2.2 may occur.
     - CASE 2.1: if $N^R < tol$, where tol is a given tolerance (for example $tol = 0.1\,N^t$), all the cells of the remaining particles in the block n are assigned to id;
     - CASE 2.2: if $N^R > tol$, the cycle on the processor id is ended. The cycle starts again at point 3 with the processor $id+1$;
   - CASE 3: The processor id needs more particles than these available in the current block $N^a_{B_n}$. The residual number of particles is $N^R = N^t - N^a_{Bn}$. The cases 3.1 and 3.2 may occur.
     - CASE 3.1: if $N^R < tol$, the cycle on the processor id is ended. The cycle starts again at point 3 with the processor $id+1$;
     - CASE 3.1: if $N^R > tol$, the cycle at point 3 is increased with the same id in order for taking the required further particles in the next block. The required number of particles for the processor id are $N^t = N^R$, these particles are counted in the next block $n+1$ starting from the particle 1.

The Fig. 5.16 shows the flow chart of the *domain distribution* procedure in the *Multi-Domain* approach considering the generic processor id. Each processor performs the algorithm represented in the figure N_{procs} times: starting from $id = 0$ up to $id = N_{procs} - 1$ (point 2 of the ALGORITHM 5.2). In the following a description of the figure is provided considering only the first cycle in the algorithm above (when $id = 0$). The processor $myid$ starts counting from the particle 1 ($P_S = 1$) of the first block (since no cell of the block $n = 1$ has yet been assigned) until N^t is reached. As mentioned before, several cases can happen:

- CASE 1: $N_{e,B_1} = N^t$. The processor id 0 takes all the cells of the block 1. The number of particles in the processor is thus $N_{(0)} = N_{e,B_1}$. The next processor ($id = 1$) starts to count from the first particle of the second block;
- CASE 2: $N_{e,B_1} > N^t$. The processor needs less particles than the remaining ones in the block 1.
  - CASE 2.1: $(N_{e,B_1} - N^t) < tol$. As in CASE 1. The difference between the particles of the block 1 and those required is lower than a given tolerance (for example $tol = 0.1\,N^t$). The residual particles in the block 1 are thus taken $N_{(0)} = N_{e,B1}$. Therefore, all the cells of block 1 are assigned to $myid$ 0. The next processor $id = 1$ starts counting from the first particle of the second block;
  - CASE 2.2: $(N_{e,B1} - N^t) > tol$. The difference between the particles of the block $n = 1$ and those requested is greater than the tolerance. The processor

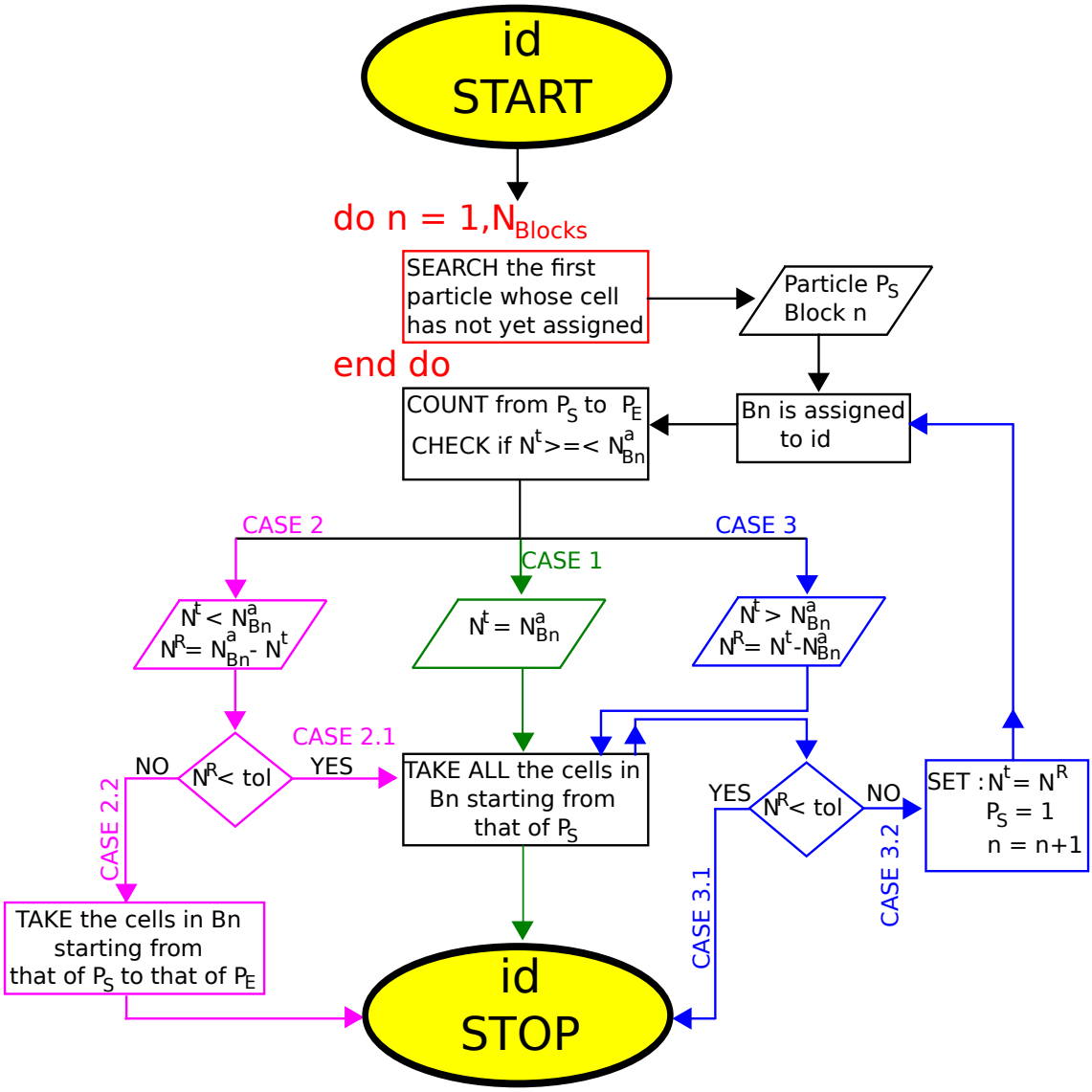


Figure 5.16: Flow chart of the *domain distribution* algorithm for the *Multi-Domain* approach. Each processor performs the algorithm N_{procs} times: starting from $id = 0$ up to $id = N_{procs} - 1$.

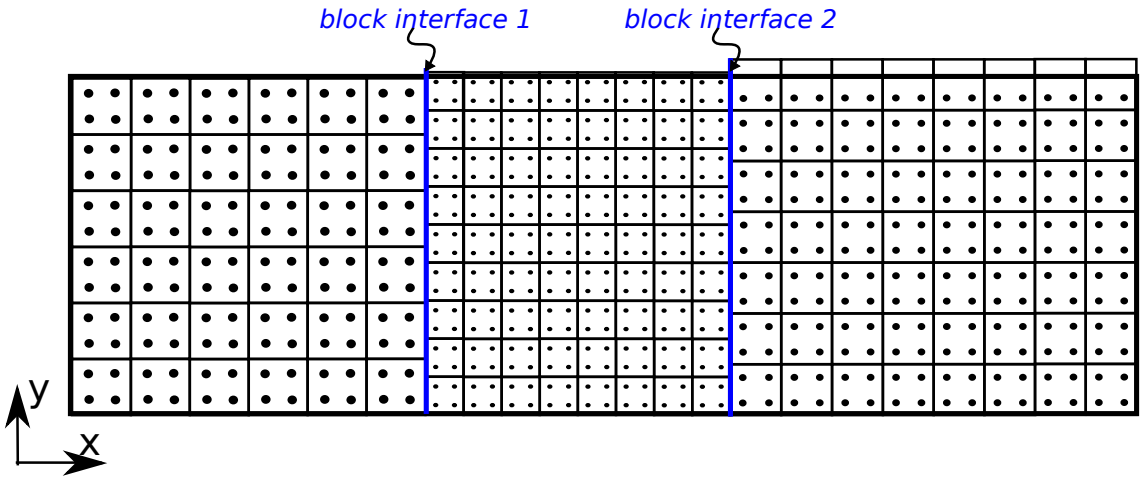
takes the cells starting from that of the particle P_S up to that of P_E. All the particles in the last assigned cell belong to the current processor as well ($N_{(0)} = N^t + res$ as explained for the *Single-Domain* approach, see Sec. 5.2). The other particles in the block 1 ($N_{e,B_1} - N_{(0)}$) will be taken by the next processor ($id = 1$) which starts counting from the first particle lying in the cell of the block 1 not yet assigned.

- CASE 3: $N_{e,B1} < N^t$. The processor needs more particles than the remaining ones in the block 1.
  - CASE 3.1: $(N^t - N_{e,B1}) < tol$. As in CASE 1. The difference between the particles of the block 1 and those required is lower than the tolerance. The processor does not take other particles ($N_{(0)} = N_{e,B1}$). All the cells of the

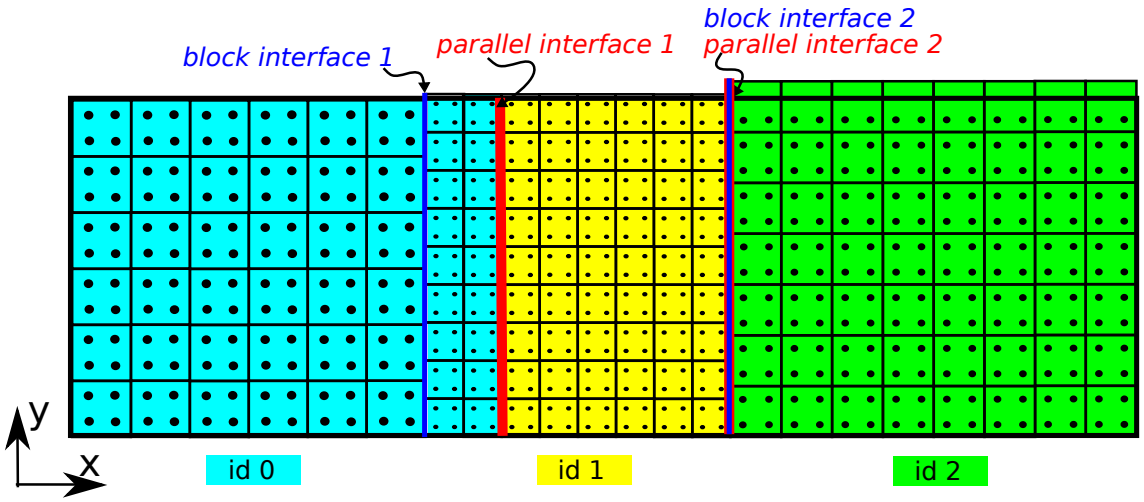
block are assigned to the processor *id* 0;

– CASE 3.2: $(N^t - N_{e,B1}) > tol$. The difference between the particles of the block and those required is greater than the tolerance. The processor *id* 0 takes the remaining required particles ($N^R = N^t - N_{e,B1}$) from the next block (block 2). Obviously, the previous cases (1, 2 and 3) can be repeated in the new block if the $N_{e,B_2} = N^R$, $N_{e,B_2} > N^R$ and $N_{e,B_2} < N^R$, respectively. All the cells of the block 1 and the cells of the particles counted in the block 2 (or in the next blocks if necessary) are assigned to the current processor.

It should be noted that the processors could have a fraction of a block (the block is then assigned to more than one processor), a whole block, a whole block and a fraction of another, more then one block (and so on..). Each processor knows only the blocks assigned to it.



(a)



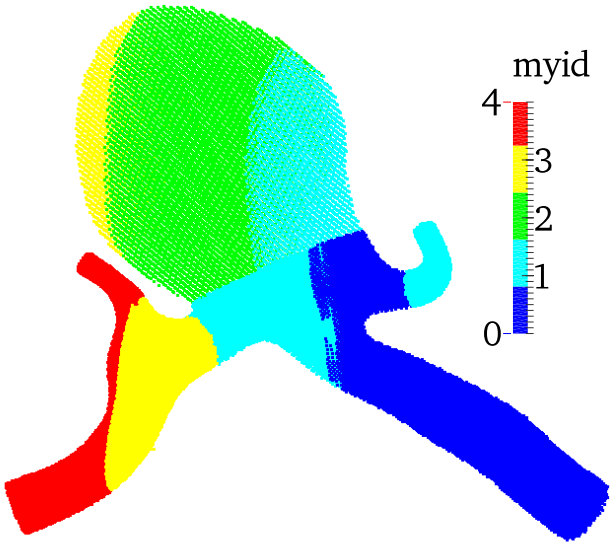
(b)

Figure 5.17: 2*D* Sketch of the *domain distribution* in the *Multi-Domain* approach. a) Scheme of the domain subdivision in three blocks. Blue lines: *block interfaces*; b) scheme of the parallel distribution into three processors. Red lines: *parallel interfaces*.

The Fig. 5.17.a shows a $2D$ simple computational domain partitioned into 3 blocks with plane *block interfaces* (represented as blue lines). In the figure the external cells of each block are not shown for the sake of clarity. The blocks have 144, 255, 208 particles (N_{e,B_1}, N_{e,B_2}, and N_{e,B_3}, respectively); the total number of particles is thus $N_{e,tot} = 640$. The Fig. 5.17.b shows the *domain distribution* into 3 processors. The theoretical number of particles is $N^t = 214$ for each processor. The first processor takes all the cells of the first block (corresponding to 144 particles); since it needs other 70 particles ($N^R = N^t - N^a_{B_1} = 214 - 144$ with $N^R > tol$ and $tol = 0.1N^t = 21$), it continues to count the particles in the second block starting from the particle 1 up to the particle 70 and takes the cells of the counted particles. Therefore, the particles of *id* 0 are $N_{(0)} = 216$: 144 in the block 1, 70 in the block 2 and other 2 particles lying in the last assigned cell (of the same block 2). The domain of *id* 0 is shown in the figure with the cyan color: the processor has the whole block 1 and a fraction of the block 2. The second processor *id* 1 starts to count the particles from the one lying in first cell not yet assigned (particle 73 of the block 2) up to the particle 286 of the same block 2. The second processor needs less particles than those available in the block ($N^a_{B_2} = 216$); since the difference $N^a_{B_2} - N^t$ is lower than the tolerance *tol* ($216 - 214 < tol$) it still takes all the cells of the block 2 (yellow region in the figure) starting from that of the particle 73. The number of particles of *id* 1 is $N_{(1)} = 216$. The last processor *id* 3 starts to count from the first particle of the block 3 (since it lies in the first cell not yet assigned) and counts until the last particle in block 3 (particle 208). Therefore, it takes all the cells in block 3 (green cells in the figure) and can not continue since there are not other subdomains (the number of particles of the last processor is equal to that of the third block $N_{(2)} = 208$).

The Fig. 5.18.a shows the *domain distribution* between five processors considering the aneurysm of Fig. 4.2 that has been partitioned into six blocks. The number of particles of each processor is exactly the same (equal to 22 560) with exception of the last processor (having 22 557 particle) although the difference is negligible (as shown in Fig. 5.18.b). The domain assigned to each processor is shown in Figs. 5.18.c,d,e,f,g where only the particles of the considered processor are represented. As it is seen in the figures, the processors can have particles belonging to different blocks: for example the first processor (whose particles are highlighted with blue color) has all the particles of block 1 ($N_{e,B_1} = 18\,146$) and a fraction of block 2 (equal to 4 414 particles); the second processor (whose particles are represented with cyan color) has the remaining particles in block 2 (whose number is $13\,136 = N_{e,B_2} - 4\,414$), all the particles in block 3 ($N_{e,B_3} = 5\,766$) and a fraction of block 4 (equal to 3 658 particles); and so on for the other processors as shown in Fig. 5.18.b. The Fig. 5.18 highlights the excellent performance of the *domain distribution* technique considering a domain with a very complex geometry.

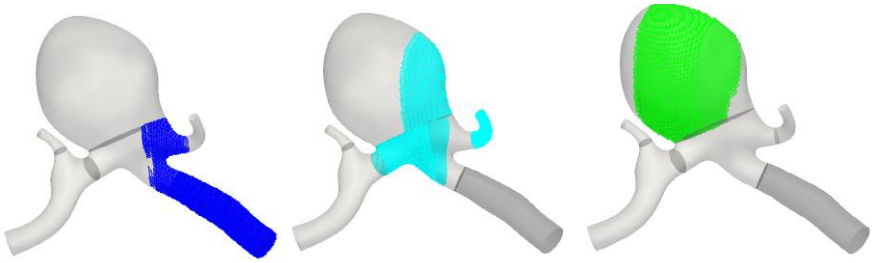
When the *parallel interface* does not coincide with the *block interface*, each processor must identify the cells containing the particles to be shared with the neighboring processors on the right and on the left (cells of type 5 and 6, respectively) using the same procedure explained in Sec. 5.2.1 for the *Single-Domain* approach. The Fig. 5.19 shows the domain of each processor with reference to the example of Fig. 5.17. The first processor, whose domain in shown in Fig. 5.19.a, has the whole block 1 (in the figure the external cells of block 1 are not represented for the sake of clarity), thus no particle of this block must be shared. Moreover, since *id* 0 has a fraction of the second block it set to type 5 the cells neighboring those of the second processor (dark gray cells in the figure); therefore, the particles within these cells must be shared on the right. In order to send the *mirror*



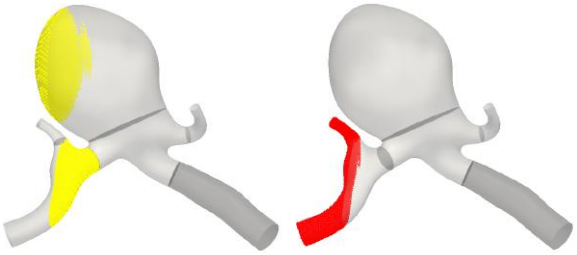
(a) Scheme of the whole domain

Block/id	*id* 0	*id* 1	*id* 2	*id* 3	*id* 4
$N_{e,B_1} = 18\,146$	18 146	–	–	–	–
$N_{e,B_2} = 17\,550$	4 414	13 136	–	–	–
$N_{e,B_3} = 5\,766$	–	5 766	–	–	–
$N_{e,B_4} = 30\,246$	–	3 658	22 560	4 028	–
$N_{e,B_5} = 32\,040$	–	–	–	18 532	13 508
$N_{e,B_6} = 9\,049$	–	–	–	–	9 049
$N_{e,Tot} = 112\,797$	22 560	22 560	22 560	22 560	22 557

(b) Number of particles

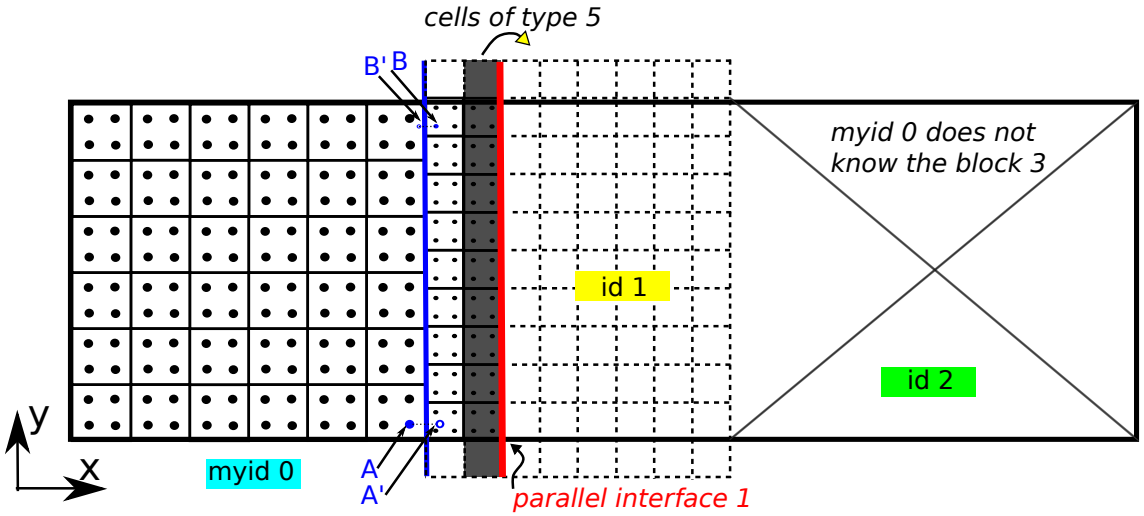


(c) *myid* 0 (d) *myid* 1 (e) *myid* 2

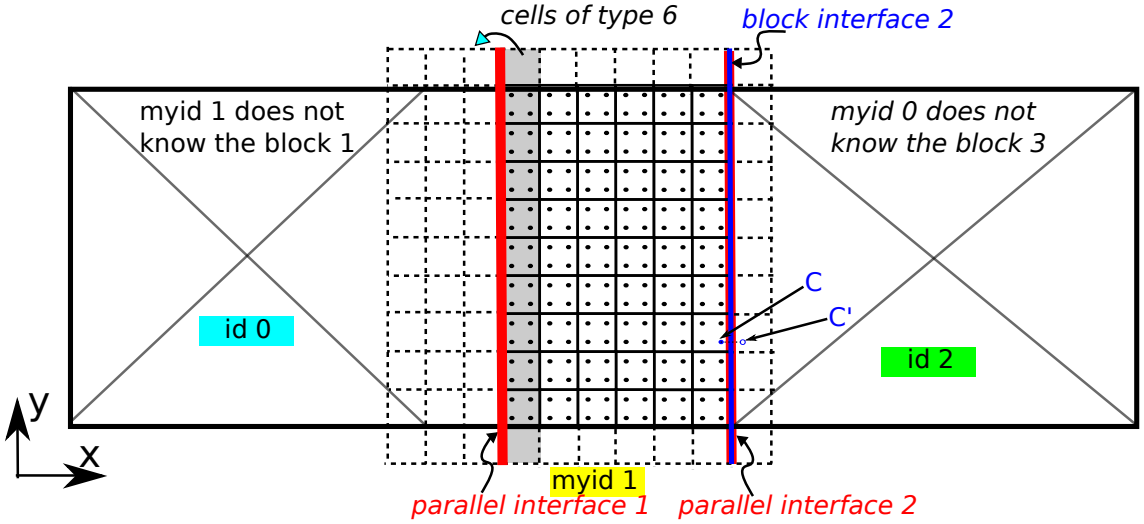


(f) *myid* 3 (g) *myid* 4

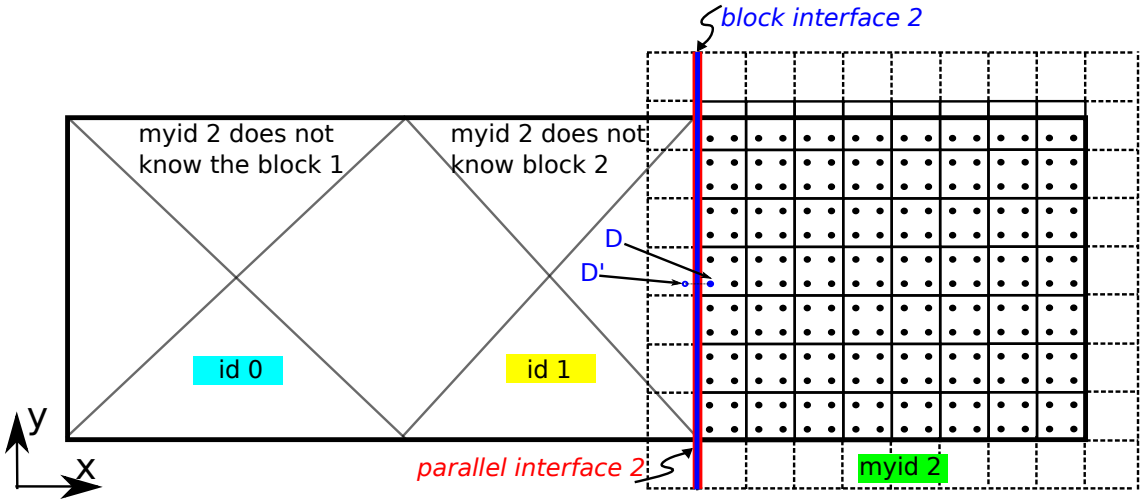
Figure 5.18: *Domain distribution* of the aneurysm shown in Fig. 4.2. $N_{procs} = 5$ and $N_{Blocks} = 6$.



(a) Domain of *id* 0



(b) Domain of *id* 1



(c) Domain of *id* 2

Figure 5.19: 2*D* Sketch of the domain assigned to each processor with reference to the example shown in Fig. 5.17. Dark and light gray area: cells of type 5 and 6, respectively.

particles, the external cells (of type 4) neighboring cells to be shared on the right are set of type 5 as well. Further, the processor *id* 0 set the remaining cells of the block 2 (that are assigned to the second processor) to type 4 (dashed black lines in the figure). The processor *id* 0 does not have cells of the third block therefore it does not know any particles in the block 3. The Fig. 5.19.b shows the computational domain of the second processor. The processor *id* 1 does not know the first and third blocks, it has only a fraction of the second block. It set to type 6 the cells bordering those of the first processor. No cells of type 5 are identified since the second *parallel interface* coincides with the *block interface* 2. In this example, the last processor (*myid* 2) does not share *effective* particles since the *parallel interface* 2 coincides with the *block interface* 2 (see Fig. 5.19.c). This situation happens whenever the processor has only one/or more whole blocks.

5.3.2 The solution matching at the block interfaces in *parallel computing*

As explained in Sec. 4.2.2, in order to match the solution at the *block interfaces*, the *IP* particles are generated from the *effective* ones having distance shorter than Δx (the *starting particle distance* of the belonging block) from one of the *block interfaces*. The hydrodynamic values of a *IP* particle generated by an *effective* particle in the block *lb* are calculated through an interpolation starting from the closest *effective* particle belonging to the block in which the IP is contained (that will be indicated as block *ib* as discussed in Sec. 4.2.3).

In *parallel computation*, the *effective* particles of a processor *id* can generate IP particles lying in the same domain of *id* or in the domain of the neighboring processor. Therefore, the processor generating the IP from the *effective* particle in its own block *lb* may not know the block *ib* where the IP is contained.

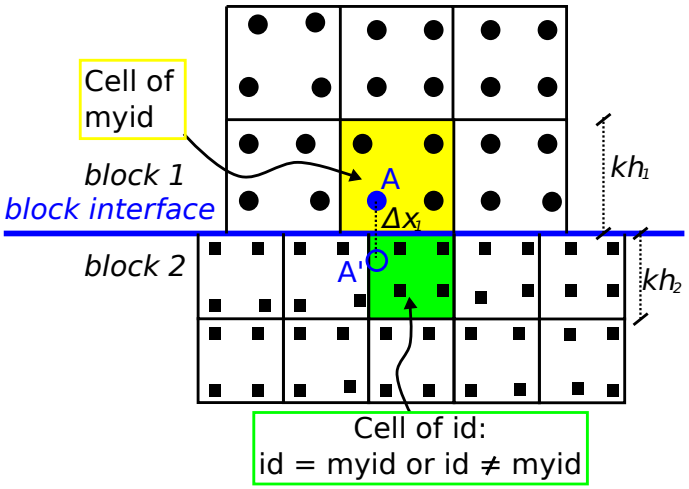


Figure 5.20: 2D Sketch of the IP generation in *parallel computing*. Black full circles and squares: *effective* particles of block 1 and 2, respectively; bold blue line: *block interface*; full blue circle: *effective* particle A of block 1 ($lb = 1$) lying in the yellow cell belongs to the processor *myid*; empty blue circle: IP particle A' generated by A and lying in the green cell (of block $ib = 2$) belongs to the processor *id* (with $id = myid$ or $id \neq myid$).

The Fig. 5.20 shows two blocks (block 1 and block 2) separated by the *block interface* (represented by the bold blue line). In the figure the *effective* particle A lying in a cell

of the current processor *myid* (the yellow cell in the figure) in the first block ($lb = 1$) is considered. It generates the IP particle A' contained in the subdomain of the neighboring block 2 ($ib = 2$). The cell in which the particle A' is contained (green cell in the figure) can belong to the same processor *myid* (if the variable *cell_processor* of the green cell is equal to *myid*) or to the neighboring processor $id = myid - 1$ or $id = myid + 1$ (if the variable *cell_processor* $\neq$ *myid* for the green cell in the figure). In the former case the *parallel interface* is internal to the block, whilst in the latter the *parallel interface* overlaps the *block interface*. If $id \neq myid$, the current processor *myid* can not do the interpolation to obtain the value (*intermediate* and *corrected* velocities, *pseudo-pressure*, etc..) of the particle A' since it does not know the *effective* particle surrounding the particle A' in the block 2 (black square circles). It happen when the *parallel interface* overlaps the *block interface*.

With reference to the example shown in Fig. 5.19.a, the *effective* particles of the first processor will generate IP particles in cells assigned to the same processor. For example, the generic *effective* particle A in the block 1 will generate the IP particle A' in the block 2 inside a cell assigned to *myid* 0; thus, the processor knows the *effective* particles in the block 2 neighboring to A'. Likewise, the generic *effective* particle B in the block 2 will generate the IP particle B' in the block 1 inside of a cell assigned to *myid* 0; thus, the processor knows the *effective* particles in the block 1 neighboring to B'. Considering the domain of the second processor (shown in Fig. 5.19.b), the generic *effective* particle C of the block 2 will generate its IP particle C' in a cell assigned to the processor *id* 2, since the *parallel interface* 2 coincides with the *block interface* 2. Therefore, *myid* 1 does not know the *effective* particles in the block 3 neighboring to C' and can not do the interpolation. Likewise, in Fig. 5.19.c the processor *myid* 2 does not know the *effective* particles in the block 2 neighboring to its IP particle D' (generated by the *effective* particle D of the third block).

A general procedure has been implemented in order to obtain the values of the IP particles. Specifically, the hydrodynamic values of a IP particle are determined by the processor owning the cell in which the IP lies, that may coincide or not with the same processor from which the IP has been generated. Therefore, considering the Fig. 5.20, the values of the *interface* particle A' are determined by the processor *id* that can coincide with *myid* if the yellow and green cells belong to the same processor. Considering the example of Fig. 5.19.a, the values of the particles A' and B' (generated by the particles A and B, respectively, of the first processor) are both obtained by the first processor; on the other hand, those of the particles C' shown in Fig. 5.19.b (generated by the particle C of the second processor) and D' in Fig. 5.19.c (generated by the particle D of the third processor) are determined by the third and second processors, respectively.

5.3.3 Sending/Receiving procedure of the *interface* particles

The processor *myid*, while generating an *interface* particle starting from its *effective* particle of the block *lb*, identifies the block *ib* where the IP is contained and the processor *id* to which the cell of the IP particle (in the *ib* block) belongs (this information is read from the *cell_processor* variable). In the example in Fig. 5.20, the particle A of the block 1 ($lb = 1$) generates the particle A' contained in block 2 ($ib = 2$). The processor *myid* must identify the *id* of the processor having the cell of A' (yellow cell in the figure).

For each IP particle generated in a cell of the processor *id*, the current processor *myid*

records in the matrix $\mathbf{slist}_{id}$:

$$\mathbf{slist}_{id} = \begin{matrix} \\ 1 \\ \vdots \\ N^{tot}_{IP,id} \end{matrix} \begin{matrix} \begin{matrix} npm & lb & ib \end{matrix} \\ \begin{bmatrix} 92903 & 5 & 6 \\ \vdots & \vdots & \vdots \\ & & \end{bmatrix} \end{matrix}$$

- the index npm of the IP particle (for example the particle 92 903);
- the block lb of origin, that is equal to the block of the generating *effective* particle (for example $lb = 5$);
- the block ib where the IP particle is contained, whose particles will be used by the processor id to obtain the hydrodynamic values of the IP particle (for example $ib = 6$).

The number of rows of the matrix is equal to the total number of *interface* particles generated in the domain of id ($N^{tot}_{IP,id}$).

After generating the IP particles for all its blocks lb, $myid$ identifies and records the number of the IP particles generated in each ib block of each processor id ($N^{ib}_{IP,id}$ with $ib = 1, .., N_{Blocks}$ and $id = 0, .., N_{procs} - 1$) including itself. As will be explained later, the processor id, after solving the equations for these IP particles using the values of the particles in the ib block, must send these information to the processor $myid$. Therefore, $N^{ib}_{IP,id}$ represents also the number of particles that the processor $myid$ must receive from the processor id after that id will have obtained the values of these IP particles.

The processor $myid$ creates a matrix, named $\mathbf{S}^{num}$, whose generic element $N^{ib}_{IP,id}$ represents the number of IP particles contained in the block ib and in a cell of the processor id. Moreover, $myid$ records the total number of particles generated in a cell of each processor id in the vector $\mathbf{S}^{num,tot}$. Considering a domain divided into six blocks and distributed in four processors, the matrix $\mathbf{S}^{num}$ and the vector $\mathbf{S}^{num,tot}$ can be expressed as

$$\mathbf{S}^{num}_{myid} = \begin{matrix} \\ B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \\ B_6 \\ Tot \end{matrix} \begin{matrix} \begin{matrix} id_0 & id_1 & id_2 & id_3 \end{matrix} \\ \begin{bmatrix} N^1_{IP,0} & N^1_{IP,1} & N^1_{IP,2} & N^1_{IP,3} \\ N^2_{IP,0} & N^2_{IP,1} & N^2_{IP,2} & N^2_{IP,3} \\ N^3_{IP,0} & N^3_{IP,1} & N^3_{IP,2} & N^3_{IP,3} \\ N^4_{IP,0} & N^4_{IP,1} & N^4_{IP,2} & N^4_{IP,3} \\ N^5_{IP,0} & N^5_{IP,1} & N^5_{IP,2} & N^5_{IP,3} \\ N^6_{IP,0} & N^6_{IP,1} & N^6_{IP,2} & N^6_{IP,3} \\ N^{tot}_{IP,0} & N^{tot}_{IP,1} & N^{tot}_{IP,2} & N^{tot}_{IP,3} \end{bmatrix} \end{matrix}$$

$$\mathbf{S}^{num,tot}_{myid} = \begin{bmatrix} N^{tot}_{IP,0} \\ N^{tot}_{IP,1} \\ N^{tot}_{IP,2} \\ N^{tot}_{IP,3} \end{bmatrix}$$

Obviously, the total number of IP particles generated by the current processor $myid$ can be expressed as

$$N^{tot}_{IP,tot} = \sum_{id=0}^{N_{procs}-1} N^{tot}_{IP,id} \tag{5.5}$$

In the example of Fig. 5.17, the matrices of the three processors are

$$\mathbf{S}_0^{num} = \begin{array}{c} \\ B_1 \\ B_2 \\ B_3 \\ Tot \end{array} \begin{array}{c} \begin{array}{ccc} id_0 & id_1 & id_2 \end{array} \\ \begin{bmatrix} N_{IP,0}^1 & 0 & 0 \\ N_{IP,0}^2 & 0 & 0 \\ 0 & 0 & 0 \\ N_{IP,0}^{tot} & 0 & 0 \end{bmatrix} \end{array}$$

$$\mathbf{S}_1^{num} = \begin{array}{c} \\ B_1 \\ B_2 \\ B_3 \\ Tot \end{array} \begin{array}{c} \begin{array}{ccc} id_0 & id_1 & id_2 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & N_{IP,2}^3 \\ 0 & 0 & N_{IP,2}^{tot} \end{bmatrix} \end{array}$$

$$\mathbf{S}_2^{num} = \begin{array}{c} \\ B_1 \\ B_2 \\ B_3 \\ Tot \end{array} \begin{array}{c} \begin{array}{ccc} id_0 & id_1 & id_2 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & N_{IP,1}^2 & 0 \\ 0 & 0 & 0 \\ 0 & N_{IP,1}^{tot} & 0 \end{bmatrix} \end{array}$$

where in the matrix $\mathbf{S}_0^{num}$ the only elements $\neq 0$ are $N_{IP,0}^1$ and $N_{IP,0}^2$ since the processor generates *interface* particles contained in blocks 1 and 2 inside its own domain (the particles A' and B' for example); thus $N_{IP,2}^{tot} = N_{IP,0}^1 + N_{IP,0}^2$. In the matrix $\mathbf{S}_1^{num}$ only the element $N_{IP,2}^3$ is non-null, since the generated IP particles are contained inside cells of the block 3 of the third processor ($N_{IP,2}^{tot} = N_{IP,2}^3$). Likewise, the only non-null term in the matrix $\mathbf{S}_2^{num}$ is $N_{IP,1}^2$, since the IP particles generated by id 2 are contained inside the domain assigned to the second processor id 1 ($N_{IP,1}^{tot} = N_{IP,1}^2$).

The current processor *myid* must send each column of the matrix $\mathbf{S}^{num}$ to the corresponding processor id; simultaneously, it must receive the column of its own id from all the processors including itself.

To this aim, the *MPI* function *MPI_ALLTOALL* is used allowing to each process to send/receive distinct data (with the same amount of information) to/from all the processors, including itself. Each processor must create a whole vector (named *sendbuf*) containing the sending data ordered for processor id and for block number (first all the blocks of the first processor, then those of the second processor, etc..), while the received data are placed in the *recvbuf* array. The *sendbuf* vector is shown in Fig. 5.21 for the current processor *myid* considering the previous example with 6 blocks and 4 processors.

The Fig. 5.22 shows the structure of the **sendbuf** and **recvbuf** vectors of all the four processors. The oval, empty rectangle, full rectangle and dashed rectangle elements, outlined in the figure for each **sendbuf** vector with different colors, contain the values from 1 up to N_{Blocks} to be sent to the processors: id 0 from the position 0 up to 5 (red color in the figure), id 1 from the position 6 up to 11 (gray color), id 2 from the position 12 up to 17 (blue color) and id 3 from the position 18 up to 23 (green color). Thus, for example, the red oval in the vector $\mathbf{sendbuf}_0$ contains the number of particles of each block (from 0 up to 5 since the block are six) that the processor 0 must send to itself. Likewise, the blue empty rectangle in the vector $\mathbf{sendbuf}_1$ contains the number of particles contained into each block (from 12 up to 17 since the block are six) that the second processor must send to the third processor (id 2). On the other hand, the information inside these elements are received in the **recvbuf** vector of each processor,

where the received values from the processor 0 are placed in the first positions (oval element of different color for each processor), from 1 up to N_{Blocks}, until those received by the last processor (dashed rectangle in the figure). Each element (oval, empty rectangle, full rectangle, dashed rectangle) in the **recvbuf** arrays contains the number of the IP particles that the processor will have to solve (using the values of the particles in the

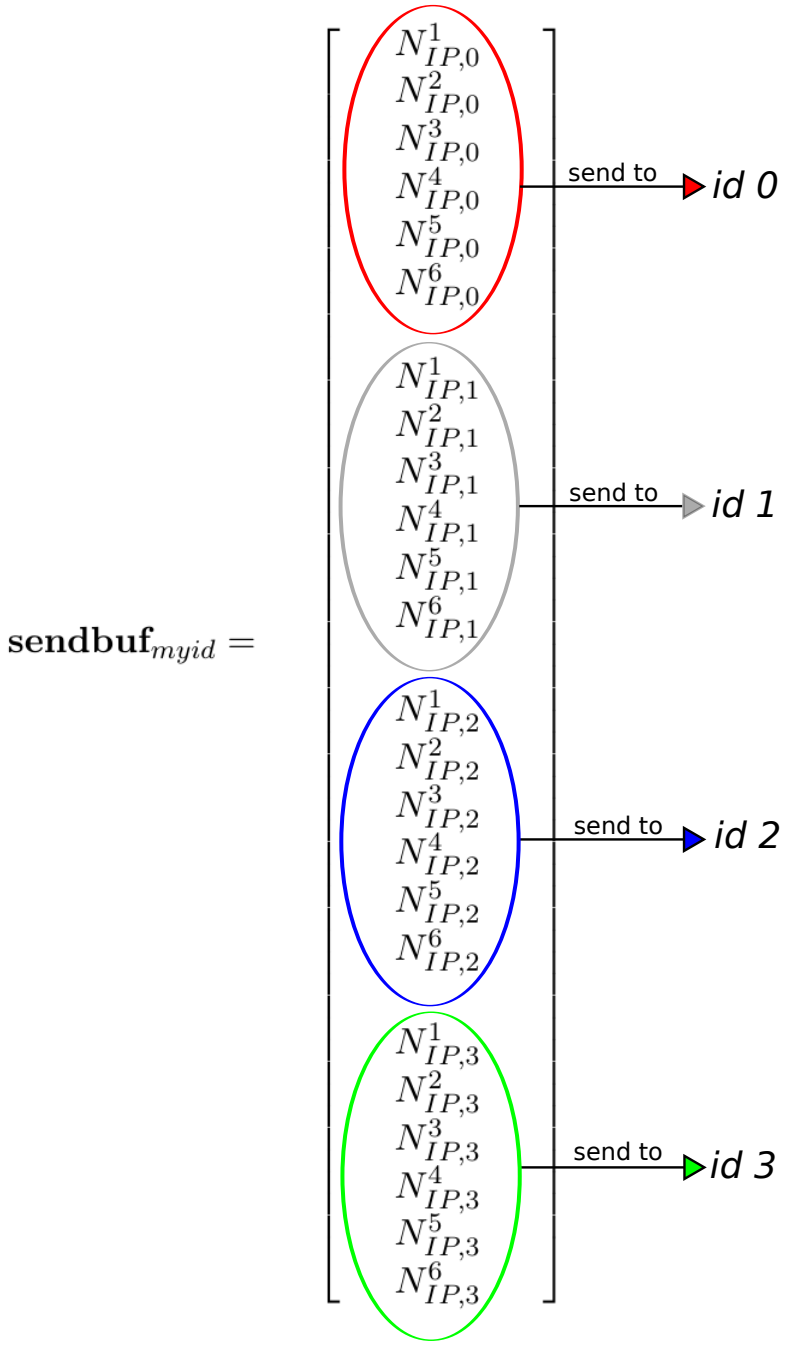


Figure 5.21: Vector **sendbuf** of the current processor $myid$ to send the number of IP particles. $N^{ib}_{IP,id}$ is the number of IP particles that the current processor $myid$ must receive from the processor id after that id will have obtained the values of these particles using the particles in the block ib.

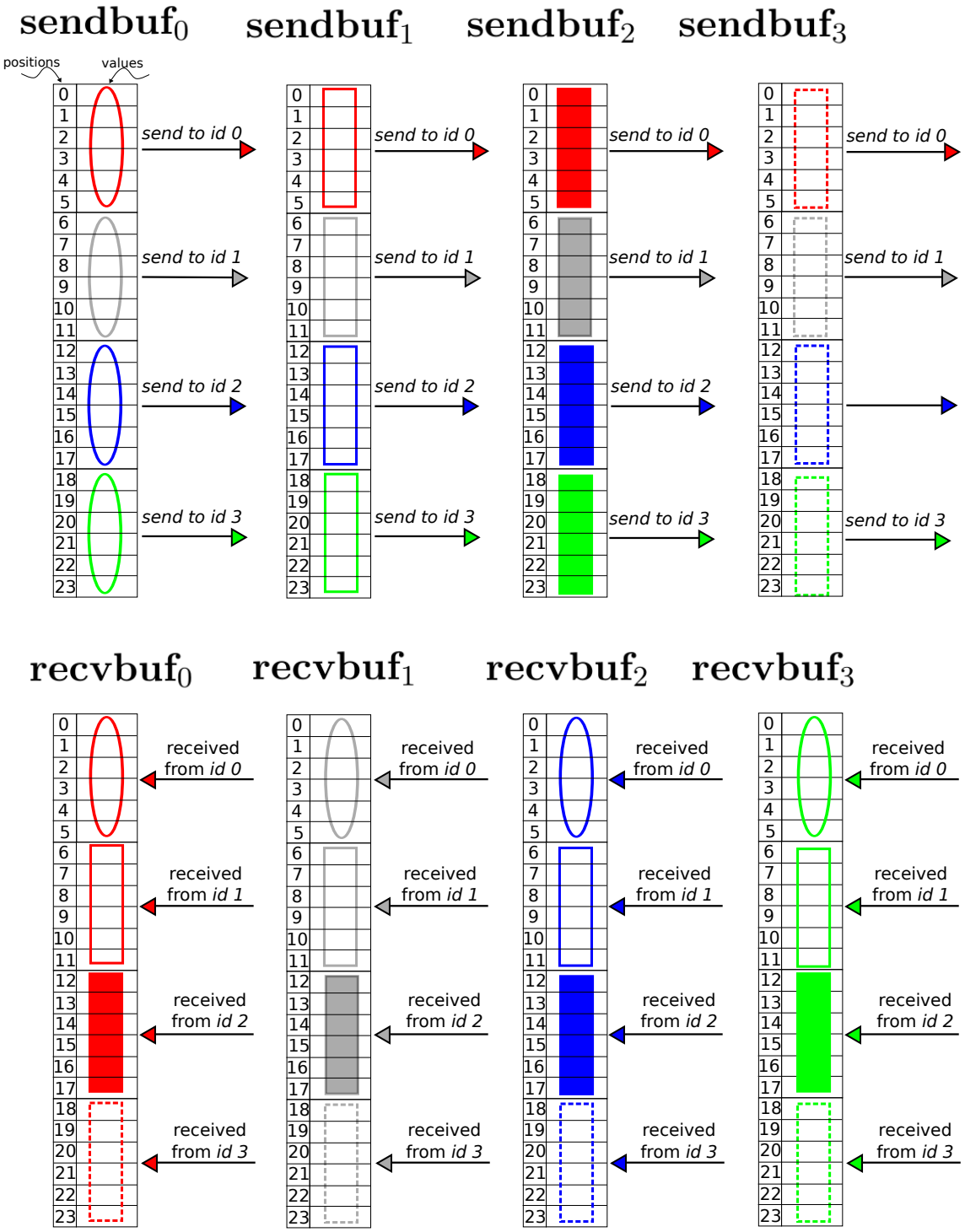


Figure 5.22: Structure of the **sendbuf** (to send the number of IP particles whose values must to be received afterwards) and **recvbuf** vectors (to receive the number of IP particles whose values must to be determined and subsequently sent). $N_{procs} = 4$, $N_{Blocks} = 6$. Only the positions are indicated.

corresponding block) and will have to send to the corresponding processor from which the *interface* particles have been generated. Thus, considering the third processor (*id* 2), it

will have to solve the N_2^{ib} particles contained in the blue oval (considering the blocks from $ib = 1$ up to $ib = N_{Blocks}$ corresponding to the positions from 0 up to 5) and then it will have to send these results to the first processor (*id* 0). In the same way, the values of the IP particles contained in the rectangle, full rectangle and dashed rectangle of blue color, will be obtained by *id* 2 (using the particles of the corresponding block) and will be sent to the processor *id* 1, *id* 2, *id* 3, respectively. Therefore, the full green rectangle in the $\mathbf{recvbuf}_3$ vector contains the number of IP for each *ib* block (from 12 up to 17) whose values will be obtained by the fourth processor using the particles of the corresponding block and then will be sent to the third processor.

The Fig. 5.23 shows the vectors *sendbf* and *recvbf* for sending the number of IP particles, considering the example of Fig. 5.17 and assuming that:

- the first processor, starting from the *effective* particles of block 1 close to the first *block interface*, generates 24 particles contained in block 2 ($N^1_{IP,0} = 24$) in the domain assigned to itself. Moreover, it generates 36 particles starting from the *effective* particles of block 2 that are thus contained in the first block ($N^1_{IP,0} = 36$) in the domain assigned to itself. The processor *id* 0 will solve and will send to itself the values of these particles;

- the second processor generates 40 particles starting from the *effective* particles of block 2 close to the second *block interface*, that are thus contained in block 3 ($N^3_{IP,2} = 40$) in the domain assigned to the third processor. The processor *id* 2 will solve and will send to the second processor (*id* 1) the values of these IP particles;

- the third processor generates 26 particles starting from the *effective* particles of block 3 close to the *block interface* 2 that are thus contained in block 2 ($N^2_{IP,1} = 26$) in the domain assigned to the second processor. The *id* 1 will solve and will send to *id* 2 the values of these particles.

Differently from Fig. 5.22, where only the positions of the elements in the vectors *sendbf* and *recvbf* are represented (considering six blocks and four processors), in Fig. 5.23 the values to be sent/received are also shown.

The received values are registered in the $\mathbf{R}^{num}$ matrix whose generic element $N^{*ib}_{IP,id}$ is the number of IP particles received by the processor *id* in each block *ib*. The total number of IP particles received by each processor N^{*tot}_{id} are recorded in the vector $\mathbf{R}^{num,tot}$. Considering an example with six blocks and four processors, the matrix $\mathbf{R}^{num}$ and the vector $\mathbf{R}^{num,tot}$ can be written as follows for the current processor *myid*

$$\mathbf{R}^{num}_{myid} = \begin{array}{c} \\ B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \\ B_6 \\ Tot \end{array} \begin{array}{c} \begin{array}{cccc} id_0 & id_1 & id_2 & id_3 \end{array} \\ \left[\begin{array}{cccc} N^{*1}_{IP,0} & N^{*1}_{IP,1} & N^{*1}_{IP,2} & N^{*1}_{IP,3} \\ N^{*2}_{IP,0} & N^{*2}_{IP,1} & N^{*2}_{IP,2} & N^{*2}_{IP,3} \\ N^{*3}_{IP,0} & N^{*3}_{IP,1} & N^{*3}_{IP,2} & N^{*3}_{IP,3} \\ N^{*4}_{IP,0} & N^{*4}_{IP,1} & N^{*4}_{IP,2} & N^{*4}_{IP,3} \\ N^{*5}_{IP,0} & N^{*5}_{IP,1} & N^{*5}_{IP,2} & N^{*5}_{IP,3} \\ N^{*6}_{IP,0} & N^{*6}_{IP,1} & N^{*6}_{IP,2} & N^{*6}_{IP,3} \\ N^{*tot}_{IP,0} & N^{*tot}_{IP,1} & N^{*tot}_{IP,2} & N^{*tot}_{IP,3} \end{array} \right] \end{array}$$

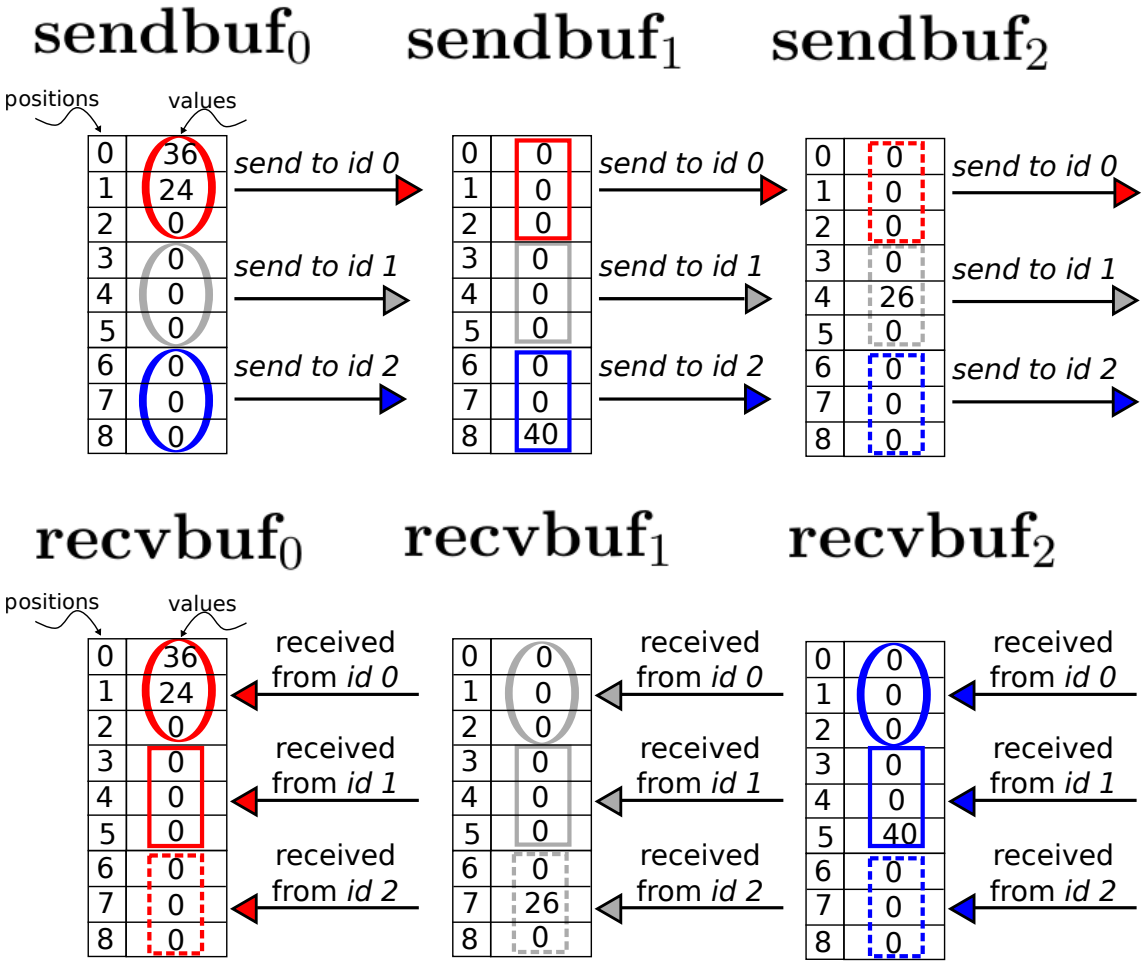


Figure 5.23: **sendbuf** and **recvbuf** vectors considering the example of Fig. 5.17 and $N^1_{IP,0} = 36$, $N^2_{IP,0} = 24$, $N^3_{IP,2} = 40$, $N^2_{IP,1} = 26$.

$$\mathbf{R}^{num,tot}_{myid} = \begin{bmatrix} N^{*tot}_{IP,0} \\ N^{*tot}_{IP,1} \\ N^{*tot}_{IP,2} \\ N^{*tot}_{IP,3} \end{bmatrix}$$

Obviously, the total number of IP particles received by $myid$ can be expressed as

$$N^{*tot}_{IP,tot} = \sum_{id=0}^{N_{procs}-1} N^{*tot}_{IP,id} \tag{5.6}$$

In order to obtain the values of the received IP particles, each processor must receive the coordinates of these particles from the processors from which they have been generated. To this aim, the MPI function $MPI\_ALLTOALLV$ is used, allowing to send data from each processor to every other processor. Differently from the $MPI\_ALLTOALL$ function, each processor may send a different amount of data and may provide displacements for input and output data. The **sendbuf** vector must be built as explained for the sending of

the number of IP particles. This vector contains the three coordinates of the IP particles (whose order is the same described previously, for processor id and for block number) as shown in Fig. 5.24.

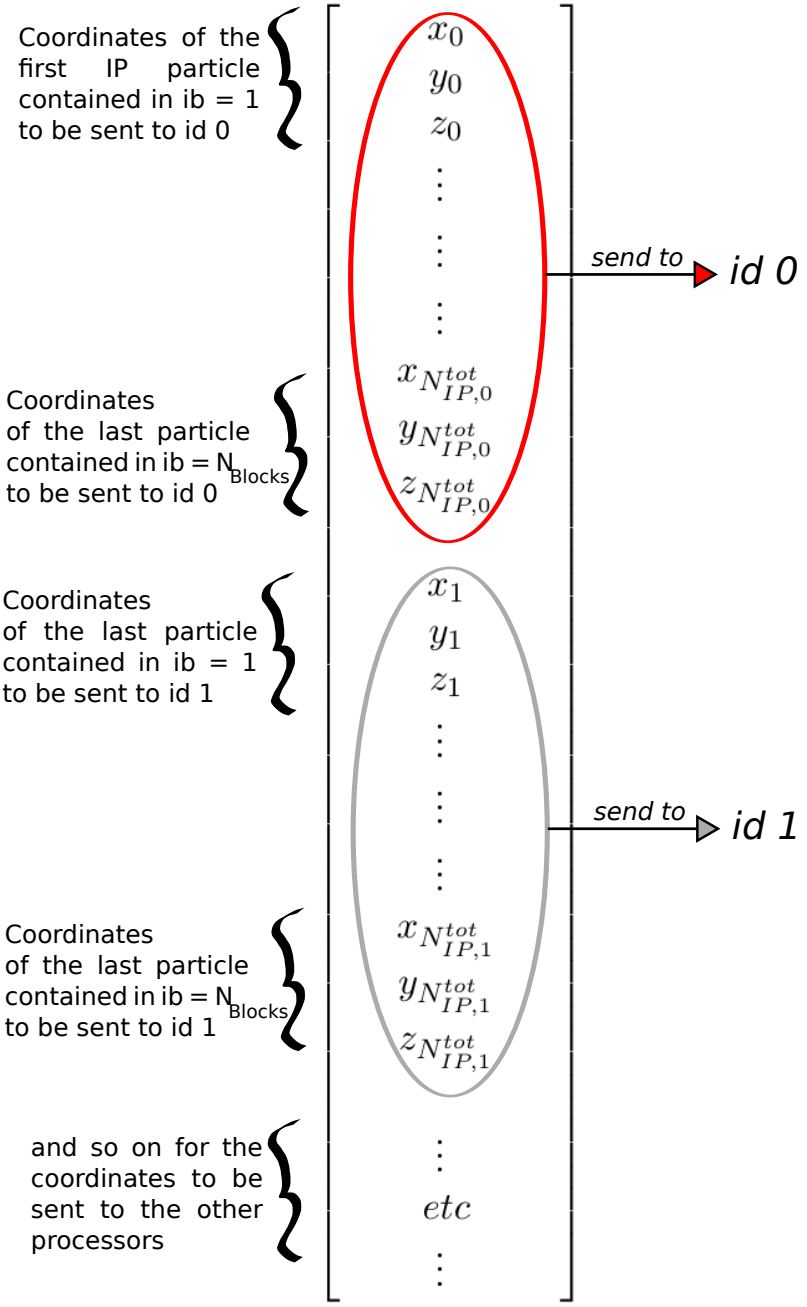


Figure 5.24: Structure of the **sendbuf** vector to send the coordinates of the IP particles.

The function $MPI\_ALLTOALLV$ needs other four vectors:

- **scount**. Integer array equal to the group size specifying the number of elements to be sent to each processor;
- **sdispl**. Integer array (of length group size) where the entry j specifies the displacement relative to **sendbuf** from which to take the outgoing data destined to processor

j;

- **rcount**. Integer array equal to the group size specifying the maximum number of elements that can be received from each processor;
- **rdispl**. Integer array (of length group size) where the entry j specifies the displacement of **recvbuf** at which to place the incoming data from processor j.

The Figs. 5.25 and 5.26 show the **sendbf** and **recvbf** vectors, respectively, considering the example of Fig. 5.17. In Fig. 5.25 the processor *id* 0 must send the coordinates of 60 particles (24 + 36), thus 180 values in total (3 coordinates x 60 values), to itself (the first element in the **scount** vector is 180) starting from the position zero in the **sendbuf** array (**sdispl** value at the first position). The processor *id* 1 must send the coordinates of 40 particles thus 120 values in total (3 coordinates x 40 values) to the processor *id* 2 (**scount** at the third position is 120) starting from the position zero in the **sendbuf** array (corresponding to the **sdispl** value at the third position). The processor *id* 2 must send the coordinates of 26 particles, thus 78 values in total (3 x 78), to the processor *id* 1 (**scount** at the second position is 78) starting from the position zero in the **sendbuf** array (corresponding to the **sdispl** value at the second position).

On the other hand, the processor *id* 0 must receive 180 values from itself (**rcount** at the first position is 180) as shown in Fig. 5.26. It places these data starting from the position zero in the **recvbuf** array (**rdispl** at the first position). The processor *id* 1 must receive 78 values from *id* 2 (**rcount** at the third position is 78). The processor fills the **recvbuf** array starting from zero (value of the **rdispl** array at the third position). The processor *id* 2 must receive 120 values from *id* 1 (**rcount** at the second position is 120) and it places these values starting from the position zero in the **recvbuf** array (**rdispl** at the second position).

The coordinates of the received IP particles are recorded in the matrix $\mathbf{R}^x$

$$\mathbf{R}^x_{myid} = \begin{array}{c} \\ 1 \\ \vdots \\ \\ \\ 3\,N^{*tot}_{IP,id} \end{array} \begin{array}{c} \begin{array}{cccc} id_0 & id_1 & id_2 & id_3 \end{array} \\ \begin{bmatrix} x_1 & x_1 & x_1 & x_1 \\ y_1 & y_1 & y_1 & y_1 \\ z_1 & z_1 & z_1 & z_1 \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \end{array}$$

After receiving the number and the coordinates of the IP particles, the processor can obtain the hydrodynamic values (*intermediate* and the *corrected* velocities and the *pseudo-pressure*) of the received *interface* particles, as will be explained in the following sections.

5.3.4 The equation velocity system in *parallel MD* approach

Each processor must build and solve the *MD* velocity system whose number of equations n_{eq} is equal to the total number of the received IP particles $N^{*tot}_{IP,tot}$ (eqn. 5.6).

The velocity system, shown in Fig. 5.27, is made of a coefficient matrix $\mathbf{A}$ [n_1 x n_2] (where $n_1 = n_{eq}$, $n_2 = n_{eq} + N^{tot}_{IP,tot}$ and $N^{tot}_{IP,tot}$ is expressed through eqn. 5.5), the vector

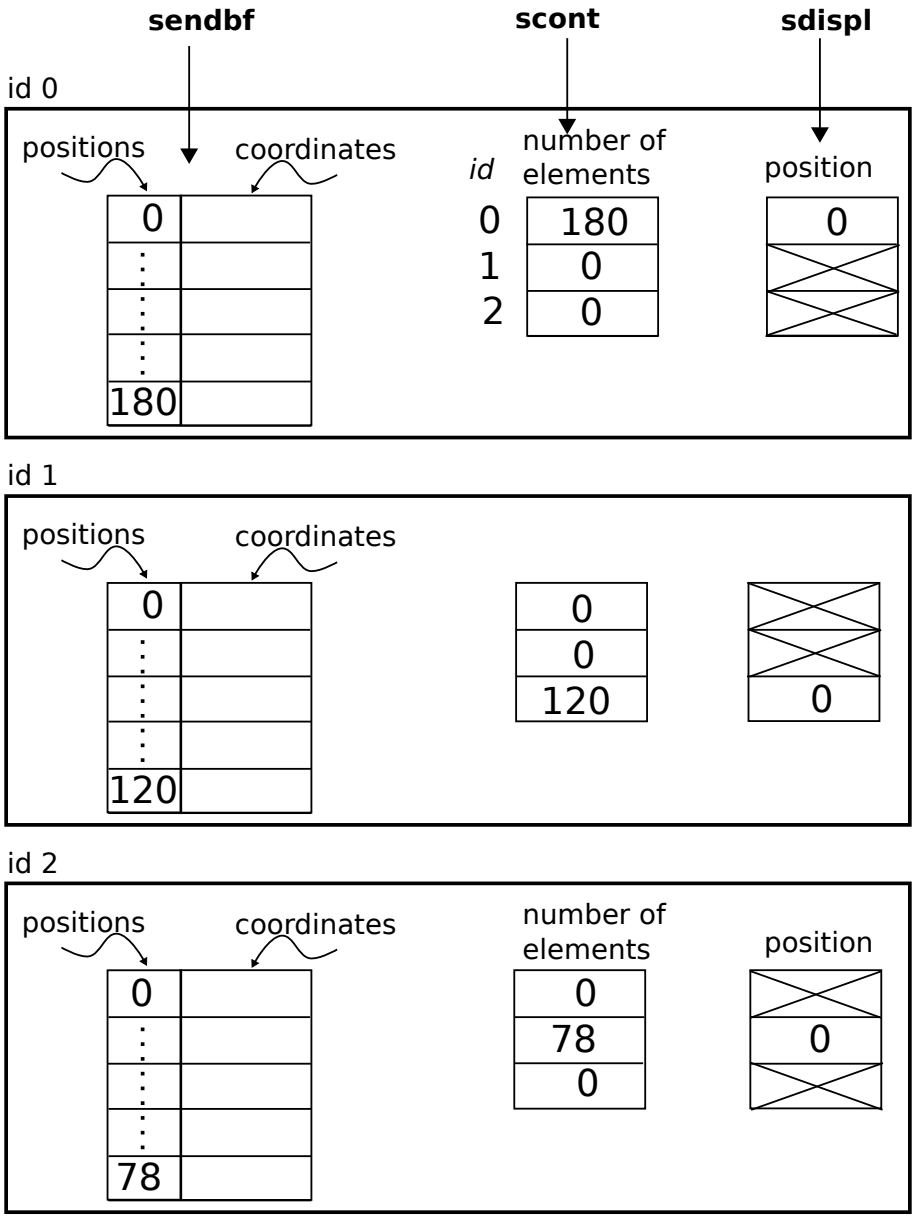


Figure 5.25: **sendbuf** vectors with reference to the example of Fig. 5.17.

b of length n_1 and the vector **x** (of original length n_{eq}) whose length has been extended to the IP particles generated by the current processor (it has thus length equal to n_2). The velocities of the generated IP particles are received from the processors having the cells in which the IP particles are contained. These values are recorded in the vector **x** from the position $n_1 + 1$ up to n_2, following the order in which the numbers and coordinates of the IP have been previously sent to each processor. For each matrix row of the current processor *myid*:

- from the column 1 to the column n_1 (with n_1 the total number of equations) there are the values of the IP particles contained in cells of *myid* whose number and coordinates had been received from all processors and had been recorded in the matrices $\mathbf{R}^{num}$ and $\mathbf{R}^{x}$, respectively (as discussed in Sec. 5.3.3). These particles are

indicated with the symbol IP^*;

- from the columns $n_1 + 1$ up to n_2 the values of the IP particles generated by the current processor $myid$ (indicated simply with IP) are registered. Specifically, in the velocity system the columns of IP particles correspond to the order in which the particles have been sent to each processor. This value must be added to n_{eq} (number of IP^* particles for which the equations must be written). Thus in the **cols** vector of the CSR format (see Sec. 2.7.1), the first particle contained in the first block ($ib = 1$) and sent to the first processor ($id = 0$) has column number equal to $1 + n_{eq}$. On the other hand, the last particle contained in the last blocks ($ib = N_{Blocks}$) and sent to the last processor ($id = N_{procs} - 1$) has column number $N^{tot}_{IP,tot} + n_{eq}$ (where

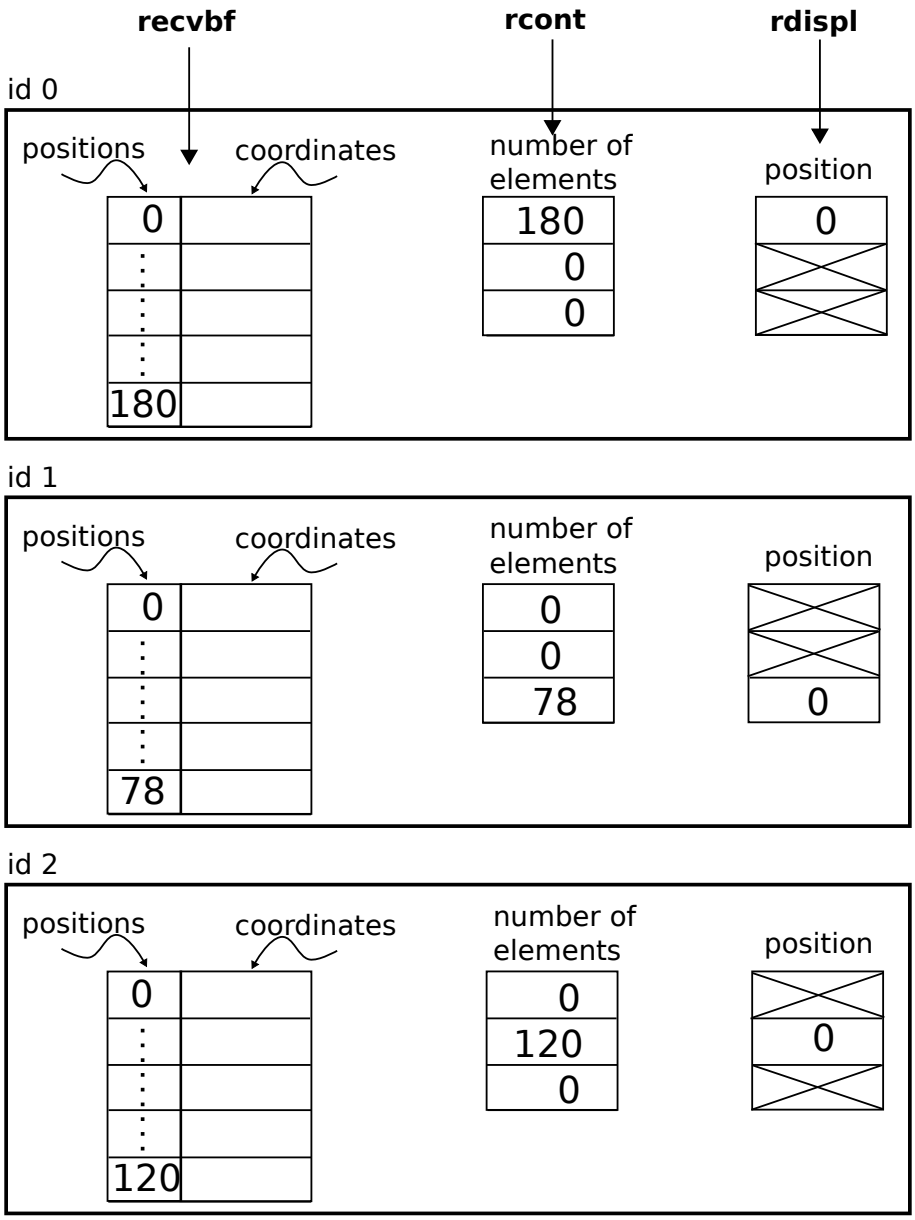


Figure 5.26: **recvbuf** vectors to receive the coordinates of the IP particles with reference to the example shown in Fig. 5.17.

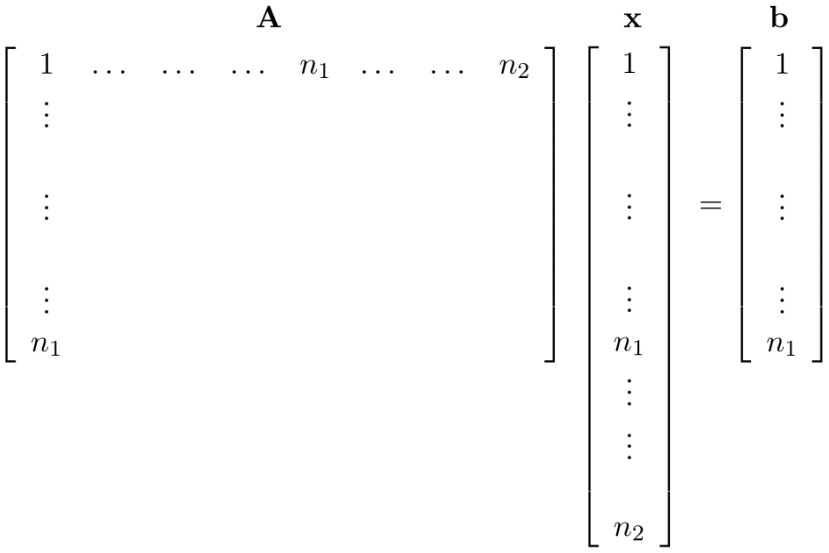


Figure 5.27: Scheme of the equation velocity system in the *Parallel MD* approach. $n_1 = n_{eq}$, $n_2 = n_{eq} + N^{tot}_{IP,tot}$. **A** is a matrix [$n_1$ x n_2], the vectors **b** and **x** have length n_1 and n_2, respectively.

$N^{tot}_{IP,tot}$ is the total number of IP particles generated by the current processor *myid*, eqn. 5.5).

A scheme of the **cols** vector is shown below:

$$\begin{array}{cc} & \mathbf{cols}_{IP} \\ \text{first particle: } ib = 1, id = 0 & \begin{bmatrix} 1 + n_{eq} \\ \vdots \\ \vdots \\ \vdots \\ N^{tot}_{IP,tot} + n_{eq} \end{bmatrix} \\ \text{last particle: } ib = N_{Blocks}, id = N_{procs} - 1 & \end{array}$$

Each processor must write the equations described in Sec. 4.2.3 for the received *interface* particles (IP^*). Specifically, for each IP particle the closest *effective* particle (named R) in the block ib is sought and the Taylor series expansion around R is made. For the generic received IP^* particle (indicated as P) contained in the block ib of the current processor *myid*, the equation for the component m of the velocity $u(m)$ (which is indicated with u in the following equations) can be expressed as

$$u_P - \sum_{j=1}^{N_R^{IP}} C'_{pr} u_j = RHS_P, \quad P = 1, .., N^{*tot}_{IP,tot} \tag{5.7}$$

where, as mentioned before, R is the closest *effective* particle in the block ib lying in a cell of the current processor *myid*, N_R^{IP} is the number of the *interface* particles generated by the current processor lying in Ω_R (whose values must be received).

The *right-hand-side* term RHS_P is

$$RHS_P \quad = \quad u_R + \sum_{j=1}^{N_R^e} C'_{pr}(u_j - u_R) - \sum_{j=1}^{N_R^{IP}} C'_{pr} u_R \tag{5.8}$$

where N_R^e and N_R^{IP} are the *effective* and IP particles, respectively, in Ω_R.

The implemented algorithm for building the velocity system in *parallel computing* is shown below.

ALGORITHM 5.3- *Equation velocity system in parallel MD approach*

---

1. $n_e = 0$: number of the current equation
2. *do* $id = 0, N_{procs} - 1$
3. $n_i = 0$: number of the current IP^* received by id (named P)
4. *do* $ib = 1, N_{Blocks}$
5. *do* $n = 1, N_{IP,id}^{*ib}$
6. $n_e = n_e + 1$
7. $diag_{n_e} = 1$ (diagonal term)
8. $n_i = n + N_{IP,id}^{*(ib-1)}$
9. Take the coordinates of P from the matrix $\mathbf{R}^x$:
   column id and rows from $[3(n_i - 1) + 1]$ to $[3(n_i - 1) + 3)]$
10. Find the closest particle R in block ib
11. $\mathbf{RHS}_{n_e} = \mathbf{u}_R$ (the velocity of R is a known term)
12. *do* $j = 1, N_R$ (cycle on the particles in Ω_R)

12.1 if j is *effective* or *mirror* then
$\mathbf{RHS}_{n_e} = \mathbf{RHS}_{n_e} + C'_{pr}(\mathbf{u}_j - \mathbf{u}_R)$

12.2 if j is IP then
$off\_diag_{(n_e,j)} = C'_{pr}$
$\mathbf{RHS}_{n_e} = \mathbf{RHS}_{n_e} - C'_{pr}\,\mathbf{u}_R$

---

where P is the received *interface* particle whose velocity must be obtained and the coefficient $C'_{pr} = \frac{m_j}{\rho_j} \nabla W_{Rj} \cdot (\mathbf{x}_P - \mathbf{x}_R)$

1. The counter of the number of the current equation n_e is initialized to zero and it will be increased for each received *interface* particle (IP^*) at point 6 until reaching, at the end, the sum of the total number of particles received by all the processors (that is equal to the total number of equations $n_e = n_{eq}$);

2. the cycle on the processor id is performed to search all the particles received by all the processors;

3. the counter of the number of received IP particles by the processor id is initialized to zero;

4. the cycle on the block ib is made;

5. the cycle on the IP particles contained in the block ib and received from the processor id is performed;

6. the counter of the equations is increased by one unit;

7. the diagonal term of the n_e-th row is equal to 1 since the velocity of the received IP particle is an unknown term;

8. the counter of the number of the current IP particle received by the processor id is calculated as the sum of the counter n and the total number of particles received by the same processor id in the previous block $ib-1$;

9. the coordinates of the current received IP particle are taken from the matrix $\mathbf{R}^x$;

10. the closest *effective* particle R is sought in the cells (belonging to the block ib) of the current processor $myid$;

11. the **RHS** (that in the *MD* system for the velocity is a matrix with three column) at the position of the current equation is equal to the velocity of R that is a known term;

12. the cycle on all the j particle lying in the *support domain* of R is made:

    12.1) if the particle j is *effective* or *mirror* the *right-and-side* of the current equation n_e is increased by the quantity $C'_{pr}(\mathbf{u}_j - \mathbf{u}_R)$;

    12.2) if j is an *interface* particle the *off-diagonal* term of the matrix system (row n_e and column of j) is increased by the quantity C'_{pr}, whilst the *right-and-side* is decreased by the quantity $-C'_{pr}\mathbf{u}_R$.

The equation system is solved using the *Pre-BiCGSTAB* method once for each velocity component m (with $m = 1, 2, 3$) using the same coefficient matrix and updating the *right-and-side* only. To this aim, the vector solution $\mathbf{x}$ (of length n_2 as shown in Fig. 5.27) becomes a matrix [n_2 x 3] as well as the initial solution $\mathbf{x}_0$ [n_2 x 3] and the known terms vector $\mathbf{b}$ [n_1 x 3]. The row of the matrices $\mathbf{x}$ and $\mathbf{x}_0$ and the length of the vectors $\mathbf{x}$, $\mathbf{x}_0$, $\mathbf{s}$, $\mathbf{p}$, $\mathbf{y}$ and $\mathbf{z}$ are extended to the total number of the *interface* particles generated by the current processor. As for the *serial mode*, the row of the matrix $\mathbf{b}$ and the dimension of the vectors $\mathbf{r}_0$, $\mathbf{r}$, $\mathbf{b}$, $\mathbf{v}$, $\mathbf{t}$ and $\mathbf{res}$ are equal to the number of equations $n_{eq} = n_1$.

The ALGORITHM 2.3 has been modified to solve the velocity system for the *MD parallel computing* scheme as shown below.

ALGORITHM 5.4- *Parallel Pre-BiCGSTAB for the MD velocity system*

---

1. *do* $m = 1, 3$
2. *Send* $\mathbf{x}_{0\,(1:n_1,m)}$/*Receive* $\mathbf{x}_{0\,(n_1:n_2,m)}$
3. $\mathbf{r}_0 = \mathbf{b}_{(:,m)} - \mathbf{A}\,\mathbf{x}_{0(:,m)}$
4. Choose $\mathbf{r}_0^*$ such that $(\mathbf{r}_0^*, \mathbf{r}_0) \neq 0$. For instance $\mathbf{r}_0^* = \mathbf{r}_0$
5. $\rho_0 = \alpha_0 = \omega_0 = 1$
6. $\mathbf{v}_0 = \mathbf{p}_0 = \mathbf{0}$

6.0 *do* $j = 1, \ldots$ *until convergence*

6.1 $\rho_{j,myid} = (\mathbf{r}^*_{0\,(1:n_1)}, \mathbf{r}_{j-1\,(1:n_1)})$;
call $MPI\_ALLREDUCE(\rho_{j,myid}, \rho_j)$;

6.2 $\beta_j = \left(\frac{\rho_j}{\rho_{j-1}}\right)\left(\frac{\alpha_{j-1}}{\omega_{j-1}}\right)$;

6.3 $\mathbf{p}_{j\,(1:n_1)} = \mathbf{r}_{j-1\,(1:n_1)} + \beta\left[(\rho_{j-1} - \omega_{j-1}\,\mathbf{v}_{j-1\,(1:n_1)}\right]$;

6.4 $\mathbf{y}_{(1:n_1)} = \mathbf{K}^{-1}\mathbf{p}_{j\,(1:n_1)}$ (*solving* $\mathbf{y}' = \mathbf{L}^{-1}\mathbf{p}_j$ *and* $\mathbf{y} = \mathbf{U}^{-1}\mathbf{y}'$)
Send $\mathbf{y}_{(1:n_1)}$ / *Receive* $\mathbf{y}_{(n_1+1:n_2)}$

6.5 $\mathbf{v}_j = \mathbf{A}\,\mathbf{y}$;

6.6 $\alpha^*_{j,myid} = (\mathbf{r}^*_0, \mathbf{v}_j)$
call $MPI\_ALLREDUCE(\alpha^*_{j,myid}, \alpha^*_j)$
$\alpha_j = \frac{\rho_j}{(\alpha^*_j)}$;

6.7 $\mathbf{s}_{(1:n_1)} = \mathbf{r}_{j-1\,(1:n_1)} - \alpha_j \mathbf{v}_{j\,(1:n_1)}$;

6.8 $\mathbf{z}_{(1:n_1)} = \mathbf{K}^{-1}\mathbf{s}$ (*solving* $\mathbf{z}' = \mathbf{L}^{-1}\mathbf{s}$ *and* $\mathbf{z} = \mathbf{U}^{-1}\mathbf{z}'$)
Send $\mathbf{z}_{(1:n_1)}$ / *Receive* $\mathbf{z}_{(n_1+1:n_2)}$

6.9 $\mathbf{t} = \mathbf{A}\,\mathbf{z}$;

6.10 $\omega^*_{j,myid} = (\mathbf{t}_{(1:n_1)}, \mathbf{s}_{(1:n_1)})$;
call $MPI\_ALLREDUCE(\omega^*_{j,myid}, \omega^*_j)$;
$\omega^{**}_{j,myid} = (\mathbf{t}(1:n_1), \mathbf{t}(1:n_1)$;
call $MPI\_ALLREDUCE(\omega^{**}_{j,myid}, \omega^{**}_j)$;
$\omega_j = \frac{\omega^*_j}{\omega^{**}_j}$;

6.11 $\mathbf{x}_{j\,(:,m)} = \mathbf{x}_{j-1\,(:,m)} + \alpha_j\,\mathbf{y}_{(:)} + \omega_j\,\mathbf{z}_{(:)}$;

6.12 $\mathbf{res}_{(1:n_1)} = \mathbf{b}_{(1:n_1,m)} - \mathbf{A}\,\mathbf{x}_j$;

6.13 $RSQ_{myid} = \left[\mathbf{res}_{(1:n_1)}, \mathbf{res}_{(1:n_1)}\right]$;
call $MPI\_ALLREDUCE(RSQ_{myid}, RSQ)$;

6.14 *Send* $\mathbf{x}_{(1:n_1,m)}$ / *Receive* $\mathbf{x}_{(n_1+1:n_2,m)}$

6.15 *Check if convergence is reached:*
if $(RSQ < tol)$ *then quit.*
else $\mathbf{r}_{j\,(1:n_1)} = \mathbf{s}_{(1:n_1)} - \omega_j\,\mathbf{t}_{(1:n_1)}$ *and continue.*

---

where the *Sending* / *Receiving* actions are highlighted in red, $\mathbf{x}_0$ at point 2 is an initial solution whose value is set equal to the velocity of the closest *effective* particle R for the particles from the position 1 to n_1, whilst it is received for the values from the position $n_1 + 1$ up to n_2.
The *MPI* function $MPI\_ALLTOALLV$ is used at points 1, 6.4, 6.8, 6.14 to *send/receive* the values related to the *interface* particles. As explained for sending the coordinates of the IP (see Sec. 5.3.3), the **sendbf** vector must be created. It contains the elements of the vector $\mathbf{x}_0$ from the positions 1 up to n_1 for point 1 (and $\mathbf{y}$, $\mathbf{z}$ and $\mathbf{x}$ for points 6.4, 6.8 and 6.14, respectively) corresponding to the x_0 values of the IP^* particles. The **scount** vector is the vector $\mathbf{R}^{num,tot}$. An example of the **sdisplcs** vector is shown below considering four

processors

$$\mathbf{sdisplcs} = \begin{bmatrix} 0 \\ N^{*tot}_{IP,0} + 1 \\ N^{*tot}_{IP,0} + 1 + N^{*tot}_{IP,1} \\ N^{*tot}_{IP,0} + 1 + N^{*tot}_{IP,1} + N^{*tot}_{IP,2} \end{bmatrix}$$

The **rercvbf** is equal to the vector $\mathbf{x}_0$ for point 1 (and **y**, **z** and **x** for points 6.4, 6.8 and 6.14, respectively) from the positions $n_1 + 1$ up to n_2, **rcount** is equal to $\mathbf{S}^{num,tot}$ (containing the total number of IP particles that the current processor must receive by each processor). An example of the **rdisplcs** vector is shown below considering four processors:

$$\mathbf{rdisplcs} = \begin{bmatrix} 0 \\ N^{tot}_{IP,0} + 1 \\ N^{tot}_{IP,0} + 1 + N^{tot}_{IP,1} \\ N^{tot}_{IP,0} + 1 + N^{tot}_{IP,1} + N^{tot}_{IP,2} \end{bmatrix}$$

After solving the system, the velocity is assigned to the IP particles following the algorithm above.

ALGORITHM 5.5- *Assignment of values to the IP particles*

---

1. *do* $id = 0, N_{procs} - 1$
2. *do* $n = 1, N^{tot}_{IP,id}$
3. Take the index npm of the IP from the matrix **slist** of id:
   $npm = slist_{id}(n, 1)$
4. Take the block lb from which npm has been generated from $\mathbf{slist}_{id}$:
   $lb = slist_{id}(n, 2)$
5. The value of npm (*intermediate* or *corrected* velocity, *pseudo-pressure*) is the element of the **x** vector in the position of the column of npm
   $u_{npm} = x(col_{npm})$

5.3.5 The equation *Poisson* system in *parallel MD* approach

Each processor must build and solve the ψ system made of n_{eq} equations: $N_{(myid)}$ *Pressure Poisson Equations* for its *effective* particles (as described in Sec. 2.7) plus $N^{*tot}_{IP,tot}$ Taylor series expansions for the received *interface* particles (as explained in Sec. 5.3.4). Therefore, for the current processor $myid$, n_{eq} is

$$n_{eq} = N_{(myid)} + \sum_{id=1}^{N_{procs}} N^{*tot}_{IP,id} = N_{(myid)} + N^{*tot}_{IP,tot}$$

The linear system for the ψ (shown in Fig. 5.28, where $n_1 = N_{(myid)}$, $n_2 = n_1 + N^{*tot}_{IP}$, $n_3 = n_2 + N^{tot}_{IP}$, and $n_4 = N_{PeP,tot}$) is made of a coefficient matrix **A** [n_2 x n_4] and

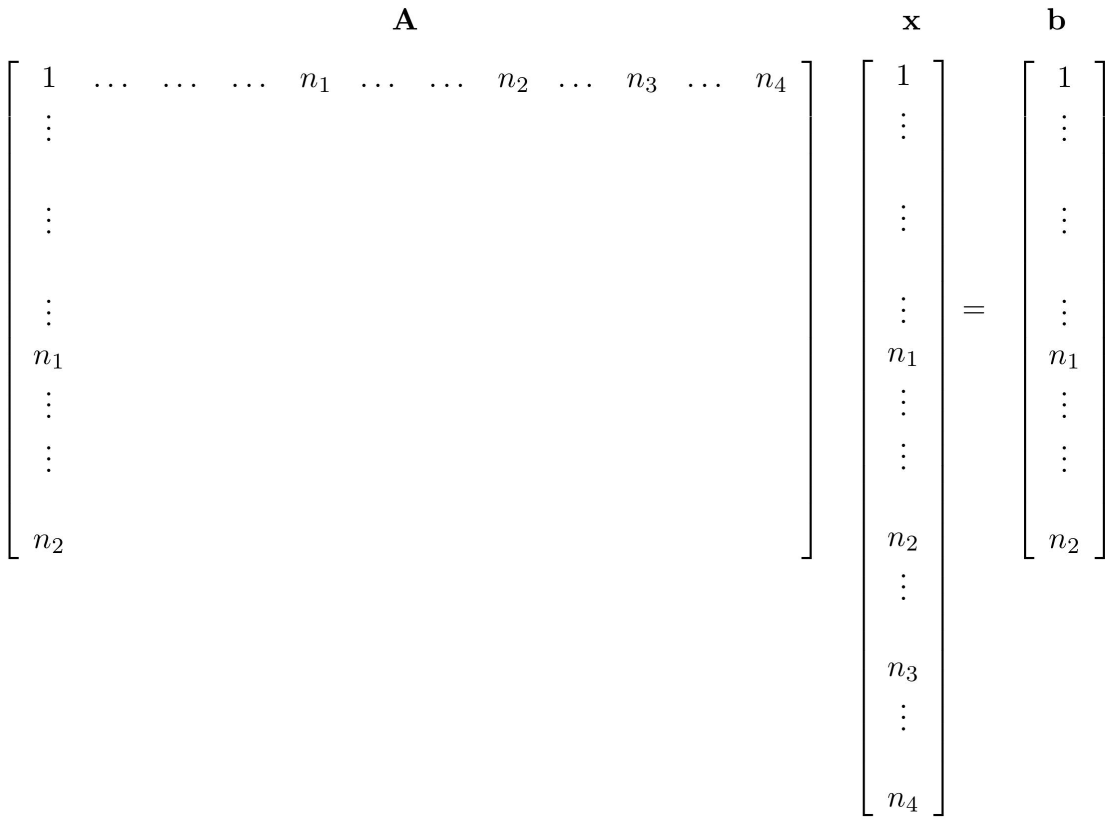


Figure 5.28: Scheme of the equation *Poisson* system in *Parallel MD* approach. $n_1 = N_{(myid)}$, $n_2 = n_1 + N^{*tot}_{IP,tot}$, $n_3 = n_2 + N^{tot}_{IP,tot}$, and $n_4 = N_{PeP,tot}$. **A** is a matrix [n_2 x n_4] and the vectors **x** and **b** have length n_4 and n_2, respectively.

the vectors **x** and **b** having lengths n_4 and n_2, respectively. Specifically, the vector of unknown terms **x** (of original length equal to the total number of equations $n_{eq} = n_1 + n_2$) is extended to the sum of the *interface* particles generated by the current processor $N^{tot}_{IP,tot}$ (whose values are received with the $MPI\_ALLTOALLV$ function and recorded in the **x** array starting from the positions $n_2 + 1$ up to n_3) and the *effective* particles received from the neighboring processors $N_{PeP,tot}$ (whose values are recorded from the positions $n_3 + 1$ up to n_4 of the **x** array).

The coefficient matrix **A** of the processor *myid* is described below:

- from the columns 1 up to n_1: the *effective* particles of the processor (considering all the blocks of *myid*) are registered. The column of the *effective* particles follows the order of the block number (first the possible particles of block 1, then those of block 2, etc..);

- from the columns $n_1 + 1$ up to n_2: the IP^* whose ψ values must be obtained by *myid* are registered;

- from the columns $n_2 + 1$ up to n_3: the *interface* particles generated by the current processor *myid* are registered. As for the velocity, in the ψ system the column of theses particles follows the order in which they have been sent to each processor; to this number n_{eq} must be added;

- from the columns n_3+1 up to n_4: the values of the *effective* particles received by the neighboring processors are registered. The columns of the PeP follow the order in which the particles have been received (for each block: first the particles received from the left and then from the right) to which the value $n_{eq}+N^{tot}_{IP,tot}$ (that is equal to $n_1+n_2+n_3$) is added.

The ψ system is solved by each processor using the *Pre-BiCGSTAB* method whose original version (see ALGORITHM 2.3) has been modified for the *parallel MD* scheme as shown below.

ALGORITHM 5.6- *Parallel Pre-BiCGSTAB method for MD ψ system*

---

1. *Send* $\mathbf{x}_{0\,(n_1+1:n_2)}$ / *Receive* $\mathbf{x}_{0\,(n_2+1:n_3)}$ for IP
2. *Send* $\mathbf{GC}^R_{x_0}$ and $\mathbf{GC}^L_{x_0}$ / *Receive* $\mathbf{x}_{0\,(n_3+1:n_4)}$ for PeP
3. $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\,\mathbf{x}_0$
4. Choose $\mathbf{r}^*_0$ such that $(\mathbf{r}^*_0, \mathbf{r}_0) \neq 0$. For instance $\mathbf{r}^*_0 = \mathbf{r}_0$;
5. $\rho_0 = \alpha_0 = \omega_0 = 1$
6. $\mathbf{v}_0 = \mathbf{p}_0 = \mathbf{0}$
7. The iterative cycle is performed until convergence ($RSQ < tol$):

7.0 *do* $j = 1, \ldots$ *until convergence*

7.1 $\rho_{j,myid} = (\mathbf{r}^*_{0\,(1:n_2)}, \mathbf{r}_{j-1\,(1:n_2)})$;
call $MPI\_ALLREDUCE(\rho_{j,myid}, \rho_j)$;

7.2 $\beta_j = \left(\dfrac{\rho_j}{\rho_{j-1}}\right)\left(\dfrac{\alpha_{j-1}}{\omega_{j-1}}\right)$;

7.3 $\mathbf{p}_{j\,(1:n_2)} = \mathbf{r}_{j-1\,(1:n_2)} + \beta\left[(\rho_{j-1} - \omega_{j-1}\,\mathbf{v}_{j-1\,(1:n_2)}\right]$;

7.4 $\mathbf{y}_{(1:n_2)} = \mathbf{K}^{-1}\mathbf{p}_{j\,(1:n_2)}$ ($solving\ \mathbf{y}' = \mathbf{L}^{-1}\mathbf{p}_j\ and\ \mathbf{y} = \mathbf{U}^{-1}\mathbf{y}'$)
Send $\mathbf{y}_{(n_1+1:n_2)}$ / *Receive* $\mathbf{y}_{(n_2+1:n_3)}$ for IP
Send $\mathbf{GC}^R_y$ and $\mathbf{GC}^L_y$ / *Receive* $\mathbf{y}_{(n_3+1:n_4)}$ for PeP

7.5 $\mathbf{v}_j = \mathbf{A}\,\mathbf{y}$;

7.6 $\alpha^*_{j,myid} = (\mathbf{r}^*_0, \mathbf{v}_j)$
call $MPI\_ALLREDUCE(\alpha^*_{j,myid}, \alpha^*_j)$
$\alpha_j = \dfrac{\rho_j}{(\alpha^*_j)}$;

7.7 $\mathbf{s}_{(1:n_2)} = \mathbf{r}_{j-1\,(1:n_2)} - \alpha_j \mathbf{v}_{j\,(1:n_2)}$;

7.8 $\mathbf{z}_{(1:n_2)} = \mathbf{K}^{-1}\mathbf{s}$ ($solving\ \mathbf{z}' = \mathbf{L}^{-1}\mathbf{s}\ and\ \mathbf{z} = \mathbf{U}^{-1}\mathbf{z}'$)
Send $\mathbf{z}_{(n_1+1:n_2)}$ / *Receive* $\mathbf{z}_{(n_2+1:n_3)}$ for IP
Send $\mathbf{GC}^R_z$ and $\mathbf{GC}^L_z$ / Receive $\mathbf{z}_{(n_3+1:n_4)}$ for PeP

7.9 $\mathbf{t} = \mathbf{A}\,\mathbf{z}$;

7.10 $\omega^{*}_{j,myid} = (\mathbf{t}_{(1:n_2)}, \mathbf{s}_{(1:n_2)})$;
call $MPI\_ALLREDUCE(\omega^{*}_{j,myid}, \omega^{*}_{j})$;
$\omega^{**}_{j,myid} = (\mathbf{t}_{(1:n_2)}, \mathbf{t}_{(1:n_2)})$;
call $MPI\_ALLREDUCE(\omega^{**}_{j,myid}, \omega^{**}_{j})$;
$\omega_j = \dfrac{\omega^{*}_{j}}{\omega^{**}_{j}}$;

7.11 $\mathbf{x}_{j\,(1:n_3)} = \mathbf{x}_{j-1\,(1:n_3)} + \alpha_j\,\mathbf{y}_{(1:n_3)} + \omega_j\,\mathbf{z}_{(1:n_3)}$;

7.12 $\mathbf{res}_{(1:n_2)} = \mathbf{b}_{(1:n_2)} - \mathbf{A}\,\mathbf{x}_j$;

7.13 $RSQ_{myid} = \left[\mathbf{res}_{(1:n_2)}, \mathbf{res}_{(1:n_2)}\right]$;
call $MPI\_ALLREDUCE(RSQ_{myid}, RSQ)$;

7.14 *Send* $\mathbf{x}_{(n_1+1:n_2)}$ / *Receive* $\mathbf{x}_{(n_2+1:n_3)}$ for IP
Send $\mathbf{GC}^{R}_{x}$ and $\mathbf{GC}^{L}_{x}$ / Receive $\mathbf{x}_{(n_3+1:n_4)}$ for PeP

7.15 *Check if convergence is reached:*
if $(RSQ < tol)$ *then quit.*
else $\mathbf{r}_{j\,(1:n_2)} = \mathbf{s}_{(1:n_2)} - \omega_j\,\mathbf{t}_{(1:n_2)}$ *and continue.*

The vectors $\mathbf{x}$, $\mathbf{x}_0$, $\mathbf{s}$, $\mathbf{p}$, $\mathbf{y}$ and $\mathbf{z}$ (of original length n_2) are extended to the sum of the IP particles generated by the current processor plus the total number of *effective* particles received from the neighboring processors. The dimension of these vectors is thus equal to n_4. As for the *serial mode*, the dimension of the vectors $\mathbf{r}_0$, $\mathbf{r}$, $\mathbf{b}$, $\mathbf{v}$, $\mathbf{t}$ is equal to the number of equations n_2 (since the current processor *myid* writes and solves an equation for each of its *effective* particles and for each received IP^* particle). The *sending/receiving* actions are highlighted with red and blue colors for IP and PeP, respectively, in the algorithm above.
The vector x_0 (at points 1, 2 and 3) is an initial solution whose value can be set to:

- zero for the first time step of the simulation (or the *psi* of the previous time step) from the positions 1 up to n_1 (for the *effective* particles);
- the *psi* of the closest *effective* particles from the position n_1 up to n_2 (for the IP^* particles);
- the received values from the positions $n_2 + 1$ up to n_3 (for the IP particles), and likewise for the positions from $n_3 + 1$ up to n_4 (for the PeP particles).

In the *sending/receiving* procedure for IP, the function $MPI\_ALLTOALLV$ is used to send the values of the IP^* particles and to receive the values of the IP particles to/from all the processors (as explained for the velocity system in Sec. 5.3.4). Specifically, the vectors $\mathbf{x}_0$, $\mathbf{y}$, $\mathbf{z}$, $\mathbf{x}$ (at points 1, 7.4, 7.8 and 7.14, respectively, of the algorithm above) are sent from the position $n_1 + 1$ up to n_2. The received values are placed from the positions $n_2 + 1$ up to n_3.

In the *sending/receiving* procedure for the PeP particles (points 2, 7.4, 7.8 and 7.14 of the algorithm above) the function $MPI\_SENDRECV$ is used (as explained in Sec. 5.2.5 for the ALGORITHM 5.1). The received values are placed from the positions $n_3 + 1$ up to n_4.

After solving the ψ system, the ψ values of the IP particles are assigned using the same algorithm explained for the velocity assignment (ALGORITHM 5.5), in which, to

calculate the position in the $\mathbf{x}$ vector, the column of the IP particles in the ψ system is used.

5.3.6 Flow chart of the *PANORMUS-SPH* code

The Fig. 5.29 shows the flow chart of the *PANORMUS-SPH* code with the implemented *parallel MD* approach. Each action of the flow chart is performed by each processor (with reference to its own *effective* particles) with the exception of the generation of the file containing the cell classification (*sph_initialize_nn.inp*) which is made only by the first processor in *ACTION* 2 at the beginning of the simulation.

- ***ACTION*** 1: The computational domain is partitioned into non-overlapping blocks (whose total number is N_{Blocks}) as explained in Sec. 4.2.1;
- ***ACTION*** 2: The first processor ($id = 0$) creates a file containing the cell classification for each block (*sph_initialize_nn.inp*, $nn = 1, .., N_{Blocks}$). The other processors read that file. All the processors read the particle starting file and the boundary triangles file for each block;
- ***ACTION*** 3: The computational domain is distributed among the N_{procs} processors following the procedure described in Sec. 5.3.1. The cells containing the *effective* particles to be shared (cells of type 5 and 6) are identified;
- ***ACTION*** 4: For each block of the current processor *myid*, *mirror*, IO and IP particles are generated starting from its own *effective* particles (as explained in Sec. 2.5, Sec. 3.2.1 and Sec. 4.2.2). While generating the IP particles the vector **slist** is created;
- ***ACTION*** 5: The current processor *myid* creates the matrix $\mathbf{S}^{num}$ containing the number of IP particles generated in the domain of each processor *id* for each block *ib*. The values of these particles (whose number is $N^{ib}_{IP,id}$) will be obtained by *id* using its own particles in the block *ib*. To send these information the *MPI* function $MPI\_ALLTOALL$ is used. To this aim, the **sendbf** array must be created. The processor *myid* receives from all the processors the corresponding numbers of IP^* particles to be solved for each block *ib* (these data are contained in the **recvbf** array) and it records these information in the matrix $\mathbf{R}^{num}$. The processor must send to each *id* the coordinates of the $N^{tot}_{IP,id}$ particles. The function $MPI\_ALLTOALLV$ is used and the arrays **sendbf**, **scount**, **sdispl** must be created, as explained in the example of Fig. 5.25. Likewise, the array **recvbf**, **rcount**, **rdispl** must be created to receive the corresponding values, as explained in the example of Fig. 5.26. The received information are then saved in the matrix $\mathbf{R}^x$. The number of IP particles received and their coordinates will be used to build the velocity and ψ system;
- ***ACTION*** 6: The processor sends the positions and the velocities of the particles inside the identified cells of type 5 and 6 and, simultaneously, it receives from the neighboring processors the particles inside their cells of type 5 and 6 (as in *ACTION* 5 of Fig. 5.14);
- ***ACTION*** 7: The processor identifies the *support domain* of its own *effective* particles;

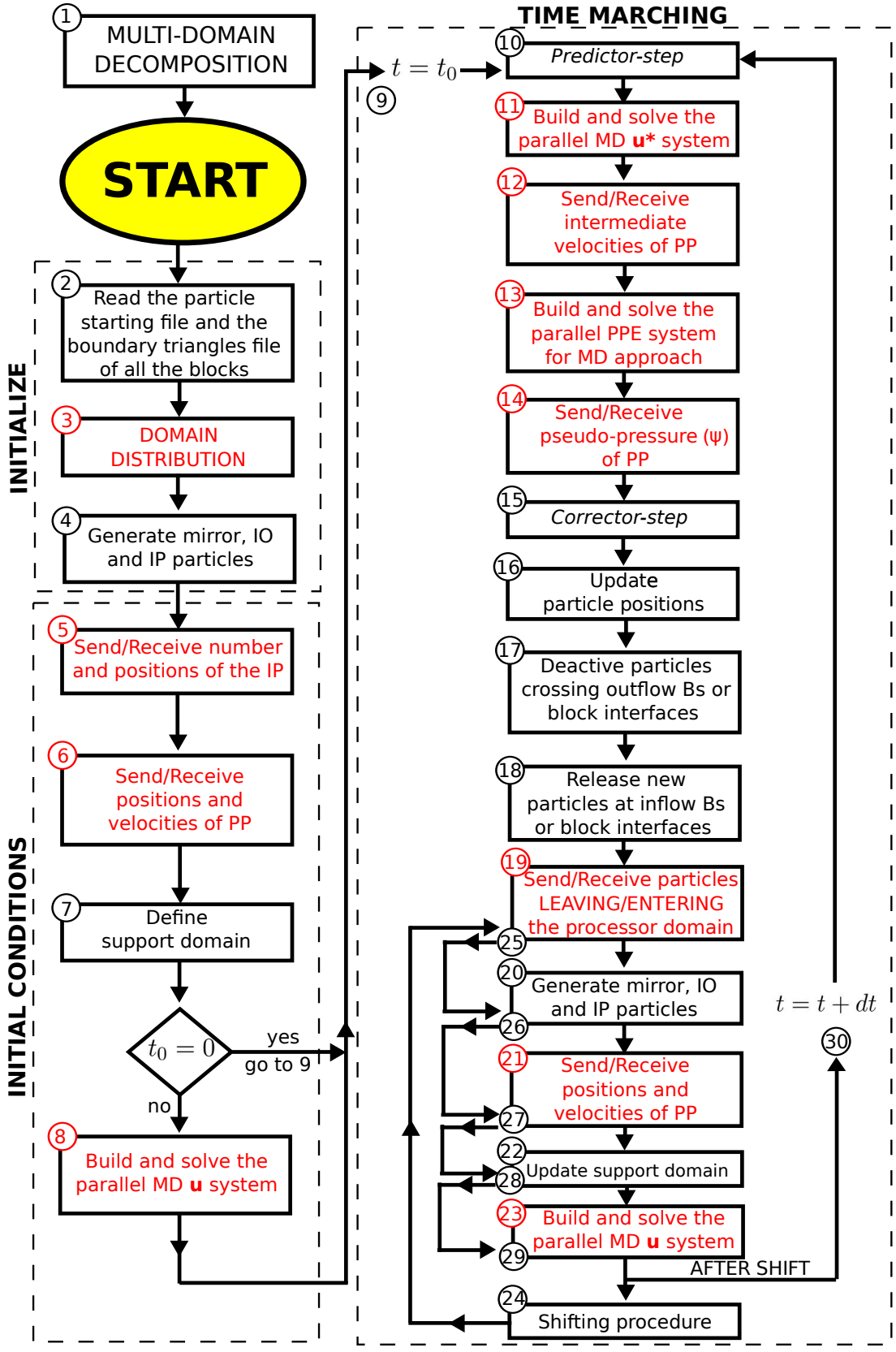


Figure 5.29: Flow chart of the *PANORMUS-SPH* code with the *parallel MD* procedure. The actions closely related to the parallelization are highlighted with red color.

- ***ACTION*** 8: If the simulation starts from developed velocities ($t_0 \neq 0$), the IP velocity values must be obtained before the *predictor-step*. The velocity system is built (see ALGORITHM 5.3) and solved (see ALGORITHM 5.4) in order to obtain the $\mathbf{u}$ of each IP^*. Correspondingly, the values of its own IP are received through the function $MPI\_ALLTOALL$ and are assigned using the ALGORITHM 5.5;

- ***ACTION*** 9: The time marching procedure starts from the initial time $t = t_0$;

- ***ACTION*** 10: The *intermediate* velocities $\mathbf{u}_i^*$ of the *effective* particles are calculated (eqn. 2.20 or eqn. 2.37 if the time step is variable). For the *mirror* particles the u_n^* value is calculated using eqn. 2.23;

- ***ACTION*** 11: The *intermediate* velocity system is built (see ALGORITHM 5.3) and solved (see ALGORITHM 5.4) in order to obtain the $\mathbf{u}^*$ of each IP^*. Correspondingly, the values of the IP are received through the function $MPI\_ALLTOALL$ and are assigned using the ALGORITHM 5.5;

- ***ACTION*** 12: The *intermediate* velocities of the *parallel* particles (PP) are shared (see *ACTION* 9 of Fig. 5.14);

- ***ACTION*** 13: The *parallel PPE system* for *multi-domain* approach is built and solved (see ALGORITHM 5.6) in order to obtain the ψ of the *effective* and IP^* particles. At the end of the iterative procedure, the ψ values of the IP particles are received through the function $MPI\_ALLTOALL$ and are assigned through the ALGORITHM 5.5;

- ***ACTION*** 14: The *pseudo-pressure* values of the *parallel* particles are shared (see *ACTION* 11 of Fig. 5.14);

- ***ACTION*** 15: The *corrected* velocities $\mathbf{u}^{k+1}$ of the *effective* particles are calculated (eqn. 2.24);

- ***ACTION*** 16: The positions of the *effective* particles are updated (eqn. 2.25);

- ***ACTION*** 17: The *effective* particles crossing external outflow boundaries or internal *block interfaces* are deactivated and saved in a storage list of the processor (as discussed in Sec. 4.2.4);

- ***ACTION*** 18: New *effective* particles are released from *inflow* boundaries or *block interfaces* (see Sec. 3.2.2 and Sec. 4.2.4, respectively) using the *scan region* procedure;

- ***ACTION*** 19: As in *ACTION* 14 of Fig. 5.14. At the end of the time step the processor checks if some *effective* particles have left the domain crossing *parallel interfaces* non-overlapping *block interfaces* (if the *parallel interfaces* coincides with the *block interfaces* the particles are already deactivated in *ACTION* 17). When a particle is deactivated it is added in the storage list of the processor and it is sent to the processor having its cell (by using the variable *cell_processor* of the cell at the new position of the particle). Moreover, simultaneously the processor receives, in the same block from which they came from, the values of the *effective* particles (positions, velocities, *pseudo-pressure*, acceleration, etc..) which have left the domain of the neighboring processors and now lie in cells of the processor domain (as explained in Sec. 5.2.4);

- ***ACTION*** 20: As in *ACTION* 4;
- ***ACTION*** 21: As in *ACTION* 6;
- ***ACTION*** 22: The processor identifies the *support domain* of the *effective* particles (as in *ACTION* 7);
- ***ACTION*** 23: The *corrected* velocity system is built (see ALGORITHM 5.3) and solved (see ALGORITHM 5.4) in order to obtain the $\mathbf{u}^{k+1}$ of each IP^*. Correspondingly, the values of the IP are received through the function $MPI\_ALLTOALL$ and are assigned using the ALGORITHM 5.5;
- ***ACTION*** 24: The *shifting* procedure of Xu et al. (2009) is performed (as explained in Sec. 2.8);
- ***ACTION*** 25: After performing the *shifting* procedure, it must be checked if some *effective* particles have left the processor domain (as in *ACTION* 19);
- ***ACTION*** 26: The *mirror*, IO and IP particles must be generated again after the *shifting* procedure, as in *ACTION* 4;
- ***ACTION*** 27: As in *ACTION* 6;
- ***ACTION*** 28: As in *ACTION* 7;
- ***ACTION*** 29: As in *ACTION* 23;
- ***ACTION*** 30: The simulation time is advanced by one time step ($t = t + dt$).

After the ACTION 30 the procedure is restarted with the *predictor-step* (*ACTION* 10).

5.3.7 Scalability test

The *Parallel Multi-Domain* approach has been validated considering the flow in the pipe shown in Fig. 5.30. The domain has been subdivided into 2 blocks where the same *smoothing length* has been used ($h_1 = h_2 = 2.5 \cdot 10^{-5}\ m$) in order to make the scalability test independent on the refinement level. Two simulations have been performed using two different lengths for the block 2: $L_2 = L1$ (in *test case* 1) and $L_2 = 2\,L1$ (in *test case* 2) corresponding to 303 360 and 606 720 particles, respectively.

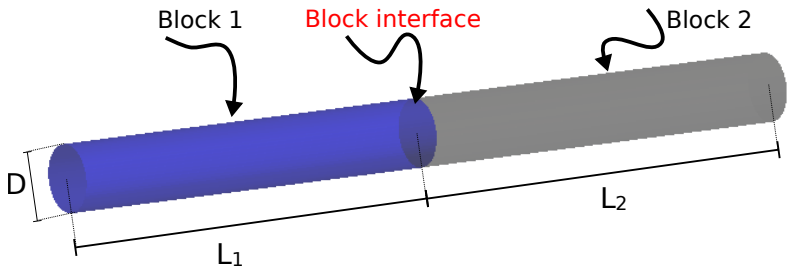
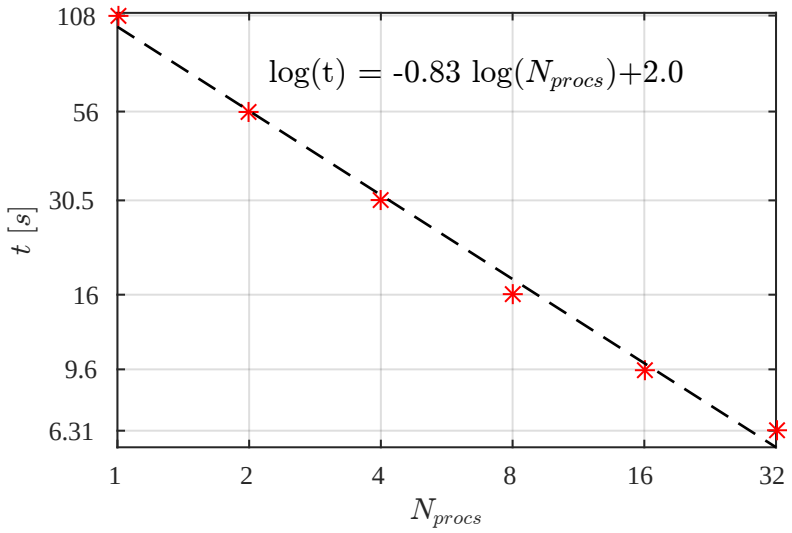
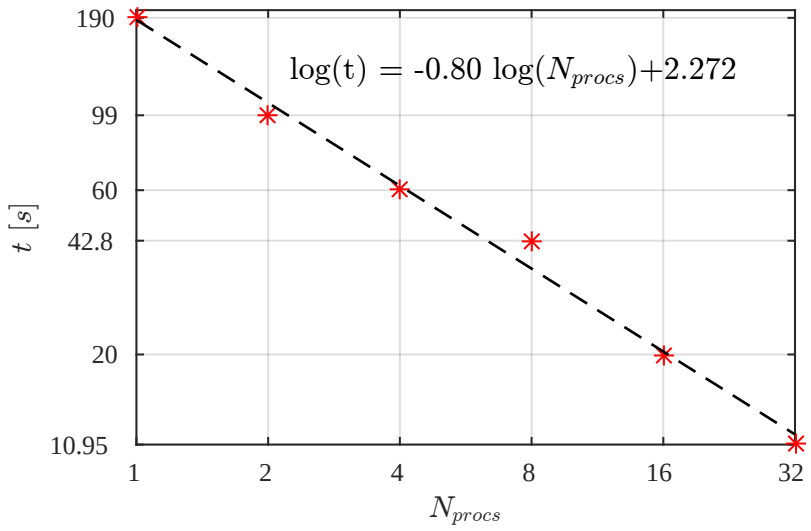


Figure 5.30: *Scalability test. Parallel MD* algorithm. Computational domain. $L_1 = 6 \cdot 10^{-3}$, $D = 1 \cdot 10^{-3}\ m$; $\Delta x_1 = \Delta x_2 = 2.5 \cdot 10^{-5}\ m$; $N_1 = 303\,360$.

In both the simulations the block 1 (of length L_1) has 303 360 particles. Thus, the total number of *effective* particles in tests 1 and 2 are $N_{e,tot} = 606\,720$ and $N_{e,tot} = 910\,080$,



(a)



(b)

Figure 5.31: *Scalability test. Parallel MD* algorithm. Results. a) Test 1. $L_2 = L_1$; $N_2 = 303\,360$; b) test 2. $L_2 = 2\,L_1$; $N_2 = 606\,720$. Red stars: time required with different N_{procs}; dashed black line: *trend-line*.

respectively. The time taken to perform one time step of the *time marching* procedure has been calculated using the *serial mode* ($N_{procs} = 1$), and the *parallel computing* with 2, 4, 8, 16 and 32 processors. The results with different N_{procs} have been plotted in the graphs shown in Fig. 5.31.a and 5.31.b for *test case* 1 and 2, respectively, using a double logarithmic scale. The *trend-line* (dashed black line) is plotted together with the results showing a slope quite similar in the two simulations (0.83 in the first test case and 0.8 in the second one). As it is seen in the figures, comparing the slope of the *trend-line* with the theoretical slope of -1 (that is the reference value of the perfect linear scalability as discussed in Sec. 5.2.7) the parallelized code shows very good scalability up to at least thirty two processors allowing to solve problems which exceed the capability of sequential calculations.

Chapter 6

Analysis of cerebral aneurysm hemodynamics

This chapter focuses on numerical simulations of blood flow in cerebral aneurysms. An ideal aneurysm as well as several real aneurysm geometries are analyzed using the *SPH* numerical model with the implemented techniques (discussed in the previous chapters). Real geometries are taken from the *Aneurisk research project* (Aneurisk-Team, 2012) which proposed an integrated analysis of the morphological and fluid dynamics features of pathologic vessels reconstructing the vessel geometries from *DICOM* images of *3D rotational angiographies*. The *Aneurisk database* provides the surface triangulation in the *STL* format.

A portion of a cerebral vessel without aneurysm is also analyzed adopting *Dirichlet pressure BCs* (described in Chap. 3) at the inlet and the outlet sections. The *non-reflective* properties of the *In/OutFlow-BCs* technique are shown.

Some numerical results are validated with a commercial finite-volume solver and with a laboratory experimental application.

6.1 Modeling assumptions

In all the presented test cases, blood has been modeled as an incompressible *Newtonian* fluid in a laminar flow regime. The fluid density ρ and dynamic viscosity μ have been set to 1060 kg/m^3 and 0.004 $Pa\,s$, respectively, corresponding to a kinematic viscosity $\nu = 3.77 \cdot 10^{-6}\ m^2/s$. Although blood is a *Non-Newtonian* fluid with a *shear-thinning* behavior (blood viscosity decreases by increasing shear strain rate), assuming constant viscosity is appropriate for the considered test cases since the sensitivity of hemodynamic predictions to different rheology models is negligible in the context of *CA* hemodynamics (Morales et al., 2013; Cebral and Lohner, 2005).

Vessel walls have been assumed rigid with *adherence* boundary condition. This is a common assumption in the context of *CA* due to the limited availability of physical information of arterial wall properties such as elasticity and thickness (Marzo et al., 2011; Cebral et al., 2005; Cebral and Lohner, 2005). Moreover, Dempere-Marco et al. (2006) showed that wall motion has only a limited effect on the hemodynamics in *CA*.

The parent vessel hosting the *CA* have been cut in order to analyze a limited region of the domain focused on the *CA*. In order to reduce the irregularity of the identified inflow and outflow sections, cylindrical flow extensions have been added. The length of these extensions is adaptively selected between $0.5 \div 4.5$ times the clipped section *diameter*. It

should be noted that, due to the vessel irregularity, the term *diameter* indicates here the dimension of the equivalent circular section. In this way, the flow profile is allowed to develop in the cylindrical extensions, prior to entering the vessel domain.

At the inlet and outlet sections the *In/OutFlow-BCs* technique (described in Chap. 3) has been employed. Specifically, in the first, third and fourth test cases (Sec. 6.2, Sec. 6.4 and Sec. 6.5, respectively) an inflow velocity profile has been set at the inlet, whilst zero pressure has been imposed at the outlet sections (the *incoming* and *pressure BCs*, respectively). *Dirichlet pressure BCs* have been used at the inlet and outlet sections of the second test case (Sec. 6.3) imposing *pressure BCs* on the triangles at the *open-boundaries*.

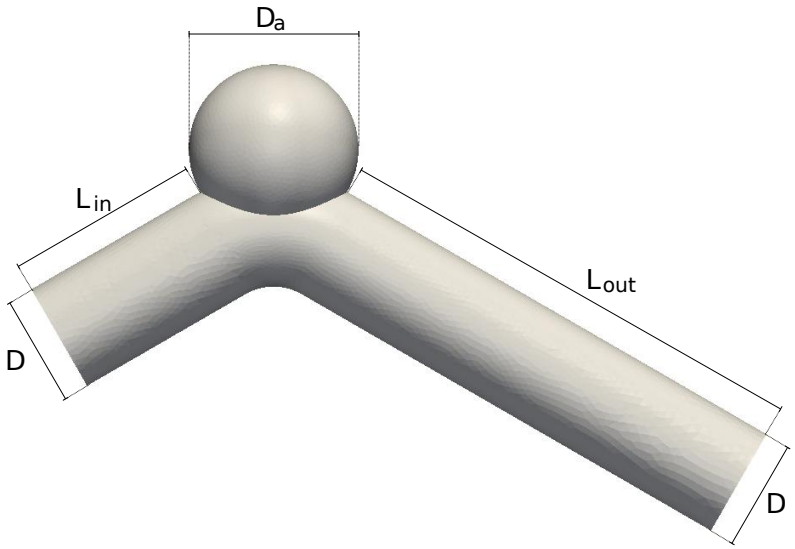


Figure 6.1: *Test case* - Sec. 6.2. Ideal aneurysm geometry (Gester et al., 2015). $D = 4.2 \cdot 10^{-3}\ m$; $D_a = 6.4 \cdot 10^{-3}\ m$; $L_{in} = 1.8\,D$; $L_{out} = 4.3\,D$.

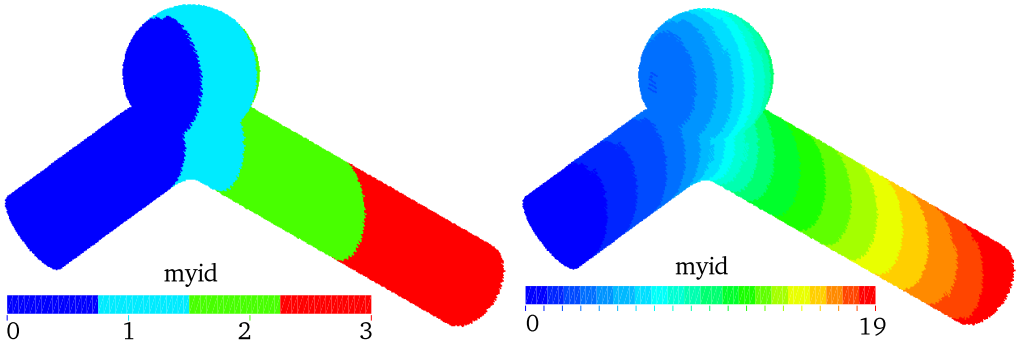
6.2 Ideal aneurysm

The ideal aneurysm geometry shown in Fig. 6.1 with cylindrical parent vessel of constant diameter D, spherical sac of diameter D_a and oval neck has been considered. The vessel has an angle of 120° at the aneurysm neck.

The simplified geometry, that is identical to that of Gester et al. (2015), has been provided by the research group of *Prof. Alejandro Frangi* (*University of Sheffield, UK*).

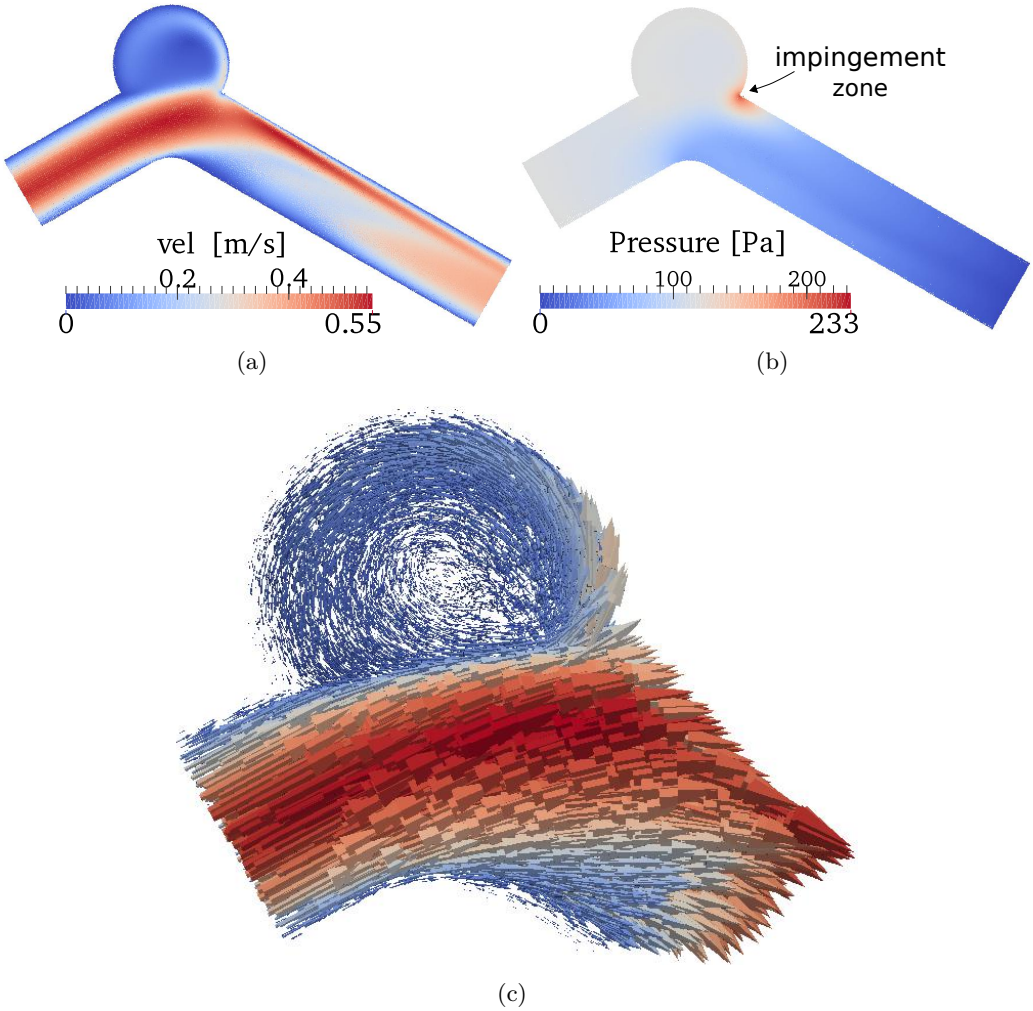
A mean flow rate of 220 ml/min has been imposed at the inlet section with a *Poiseuille* velocity profile (as described in Sec. 3.2.1, eqn. 3.2). The resulting *Reynolds number* $Re = \bar{u}\,D/\nu = 295$ (calculated considering the mean velocity at the inner section $\bar{u} = 0.2647\ m/s$) is well within the laminar regime. The mean shear stress on the vessel walls τ_0, which is an important variable for blood flow-related problems, is equal to $\tau_0 = \lambda\,\rho\,\bar{u}^2/8 = 2.01\ N/m^2$, where λ is the pipe friction factor that in the laminar regime can be expressed as $\lambda = 64/Re$.

The parent vessel has an inlet length of $L_{in} = 1.8\,D$ which has been verified to be sufficient for developing the imposed velocity profile before entering in the aneurysm sac. The length of the outlet section has been set larger than that at the inlet, $L_{out} = 4.3\,D$, in order to allow the flux to become again linear after the aneurysm vortex flow and, thus, to correctly impose the constant zero-pressure value at the outlet section.



(a) $N_{procs} = 4$ (b) $N_{procs} = 20$

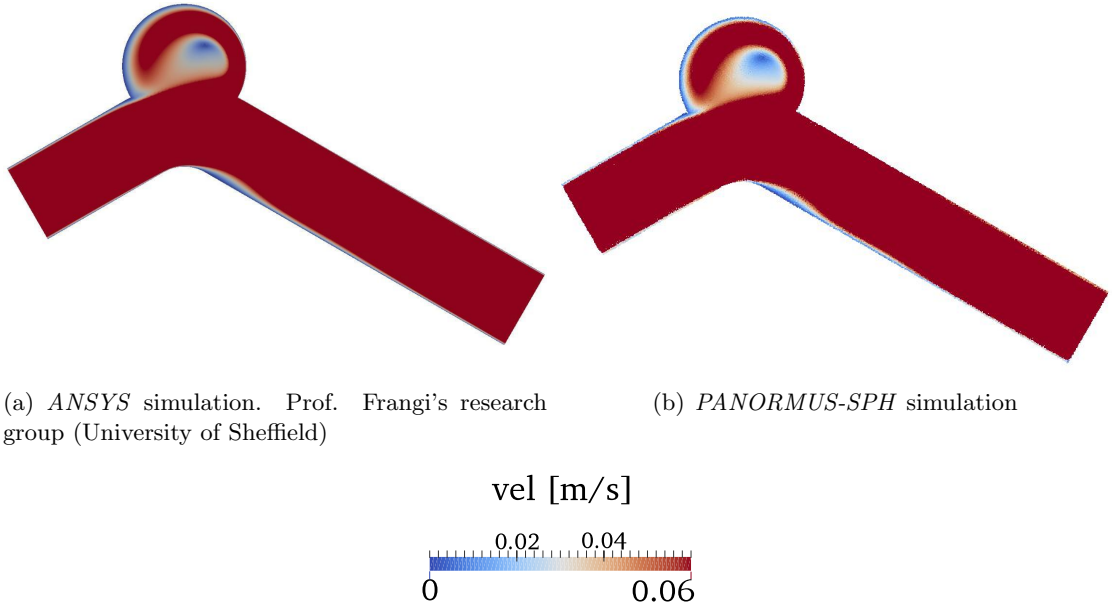
Figure 6.2: *Test case* - Sec. 6.2. *Domain distribution* among several processors.



(a) (b)

(c)

Figure 6.3: *Test case* - Sec. 6.2. a) Velocity x component in $[m/s]$ at the steady-state; b) Pressure field in $[Pa]$; c) Enlargement in the vicinity of the aneurysm neck and sac. Velocity vectors colored with the velocity magnitude using the same scale of point .a.



(a) *ANSYS* simulation. Prof. Frangi's research group (University of Sheffield)

(b) *PANORMUS-SPH* simulation

Figure 6.4: *Test case* - Sec. 6.2. Velocity magnitude field in $[m/s]$ at the steady-state. A commercial finite volume solver (*ANSYS* software) *Vs* the *PANORMUS-SPH* code.

A *smoothing length* $h = 1.0 \cdot 10^{-4}\ m$ has been set corresponding to 42 particles along the vessel inner section with a total number of particles at the beginning $N_e = 517\,453$. The classical *Single-Domain* approach has been employed in this test case. The *parallel computing* scheme (see Sec. 5.2) has been employed with $N_{procs} = 20$; each processor has 25 872 particles except for the last one having 7 particles less because of the residual in the allocation of the other processors (the *res* value in eqn. 5.1). The Fig. 6.2.a shows an example of *domain distribution* considering 4 processors, whilst the *domain distribution* with $N_{procs} = 20$ is shown in 6.2.b.

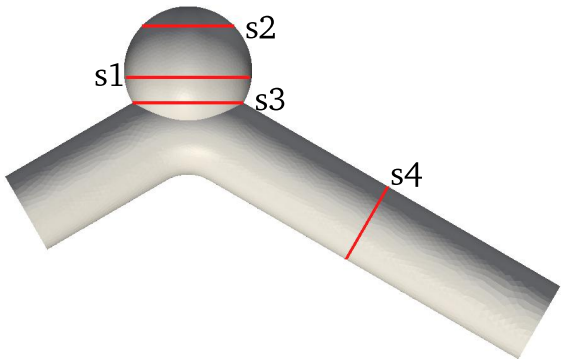
The simulation starts from the rest until the steady-state is achieved. The Fig. 6.3 shows the velocity magnitude (Fig. 6.3.a) and pressure fields with indication of the over-pressure region (Fig. 6.3.b) at the steady-state. In order to highlight the vortex occurring inside the aneurysm, the velocity vectors are represented in Fig. 6.3.c using one particle every 10 for the sake of the graphical representation.

The velocity results have been compared with those obtained with a commercial *vertex-centered finite volume* solver ($ANSYS$ software). The simulation with *ANSYS* has been performed by the research group of the Prof. A. Frangi. The elements of the FVM grid have size $4 \cdot 10^{-4}\ m$ inside the vessel and the aneurysm, whilst near the wall the mesh elements have maximum edge size of $1 \cdot 10^{-4}\ m$ with three prismatic layers (about $500 \cdot 10^3$ nodes and $1.3 \cdot 10^6$ elements).

The comparison of the two simulations, shown in Fig. 6.4, proves the very good performance of the *SPH* numerical model since a similar pattern has been obtained.

A convergence analysis has been carried out aimed at studying the dependence of the results on the h value (that in this research study is equal to the *starting particle distance* Δx, as discussed in Chap. 2). To this aim, six values of the *smoothing length* have been used and the velocity profiles along the lines s_1, s_2, s_3 and s_4 (see the scheme in Fig. 6.5.a) are plotted in Figs. 6.5.b,.c,.d,.e, respectively. The value $h = 1.0 \cdot 10^{-4}\ m$ (red continuous line in the figure) allows to obtain a satisfactorily accurate solution with a

relatively limited number of particles. It should be noted that in *SPH* sensitivity analyses involve the ratio $h/\Delta x$ as well as the *smoothing length* h. Such issue will be addressed in future work.



(a) scheme

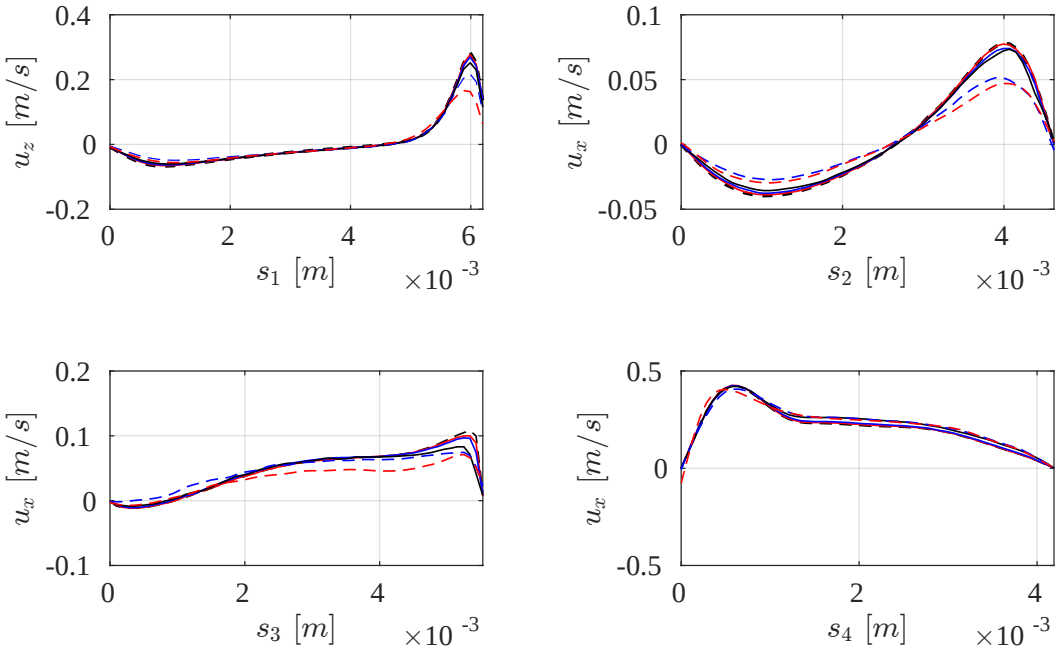


Figure 6.5: *Test case* - Sec. 6.2. Convergence analysis. Velocity profiles along the lines s_1, s_2, s_3 and s_4. The values of the *smoothing length* h are expressed here with reference to the *smoothing length* $h_0 = 1.0 \cdot 10^{-4}$ m used in the present section. Black dashed line: $h = \frac{2}{3}h_0$ ($N_e = 1\,7191\,066$); red continuous line: $h = h_0$ ($N_e = 517\,452$); blue continuous line: $h = \frac{4}{3}h_0$ ($N_e = 210\,345$); black continuous line: $h = 2h_0$ ($N_e = 64\,134$); blue dashed line: $h = 3h_0$ ($N_e = 19\,111$); red dashed line: $h = 4h_0$ ($N_e = 8\,045$).

6.3 Human cerebral blood vessels

The blood flow through a human cerebral vessel located at the *middle cerebral artery* (MCA) has been analyzed.

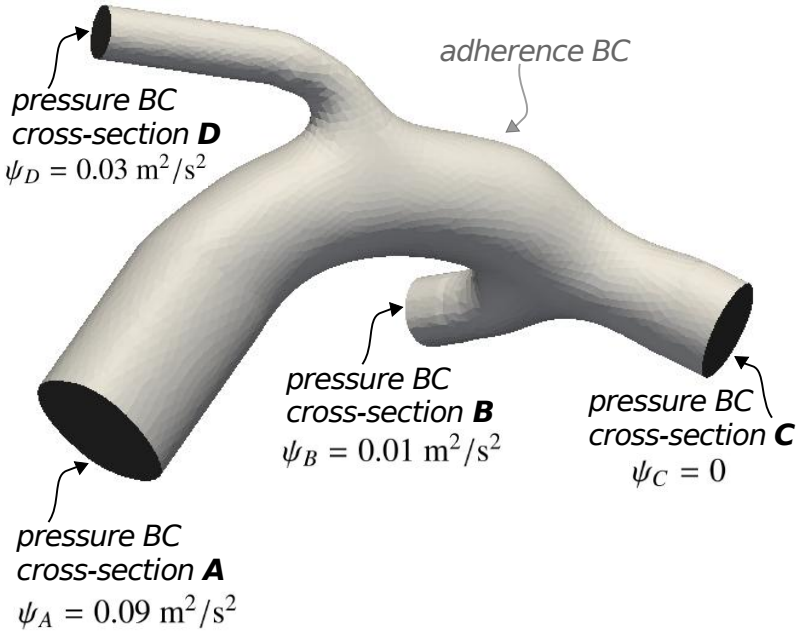


Figure 6.6: *Test case* - Sec. 6.3. Vessel geometry with *pressure BCs* at in/outflow cross-sections. Taken from: Monteleone et al. (2017), 18, fig. 15.

The geometry of the vessel (shown in Fig. 6.6) is a portion of the $C93$ test case included in the dataset repository of the *Aneurisk* project.

The selected vessel has one inlet and three outlet sections in which *pressure BCs* (see Chap. 3) have been imposed. Each branch has been extended in the axial direction adding a cylindrical tube with constant cross-section. The additional tubes have lengths ranging between 0.5 and 3 diameters (the latter in the smallest branch). The resulting axial length of the domain is about 0.02 m.

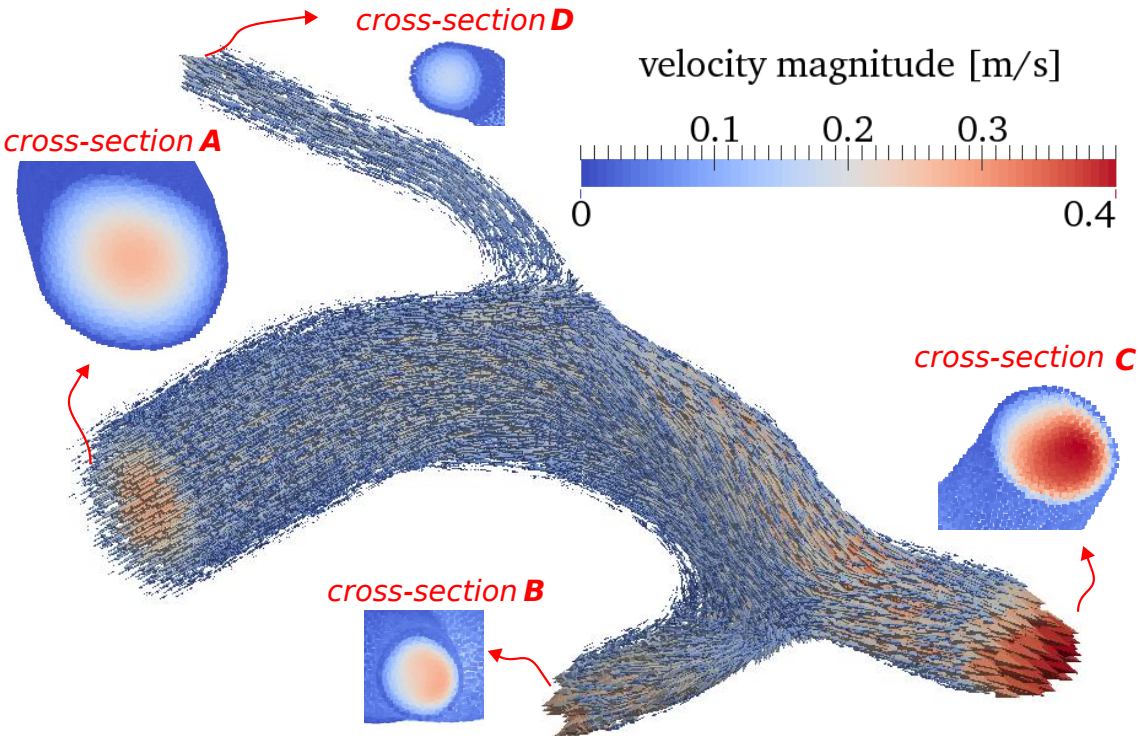
The vessel diameters range between about $0.15 \cdot 10^{-2}$ m in one of the outflow branches and $0.33 \cdot 10^{-2}$ m in the vicinity of the inflow section.

The kinematic pressure at the C outflow section has been set to zero, while the values 0.09, 0.01 and 0.03 m^2/s^2 have been imposed at the A, B and D sections, respectively, as shown in Fig. 6.6.

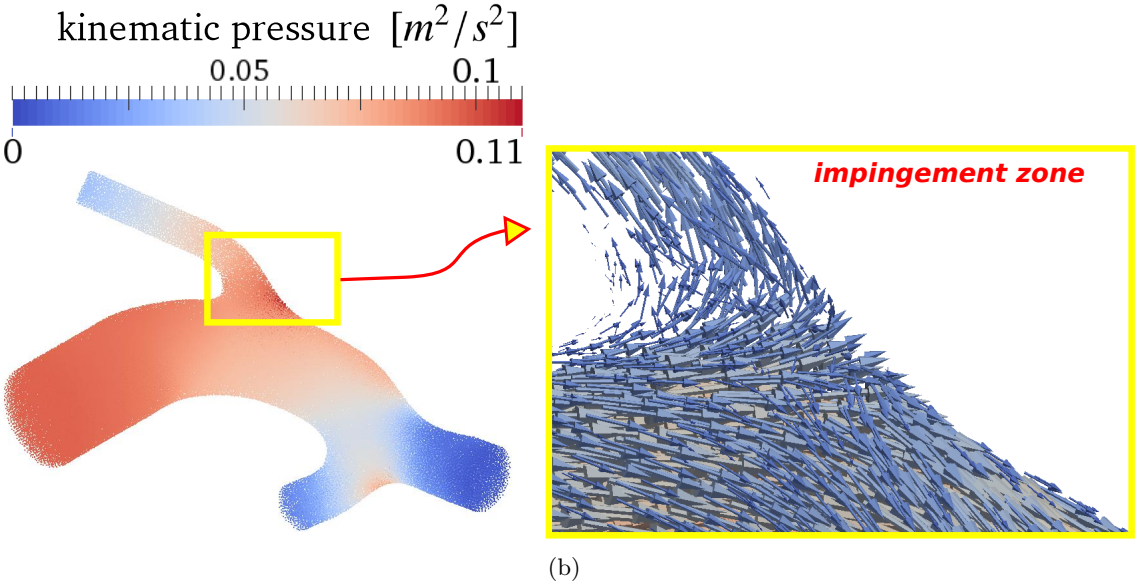
The classical *SPH* method with constant kh value (*Single-Domain* approach) has been employed in this test case. To this aim, in order to obtain a sufficient number of particles in the smaller branches, the *smoothing length* has been set to $h = 1 \cdot 10^{-4}$ m. The resulting total number of *effective* particles at the beginning is 177 320. The simulation is performed starting form the rest till the steady-state is achieved.

In the Fig. 6.7 the velocity magnitude (Fig. 6.7.a) and pressure fields (Fig. 6.7.b) are shown at the steady-state. For the sake of clarity, the vectors are shown considering only 1 particle every 10. As it is seen in Fig. 6.7.a, the velocity field in the sections far enough from the curves (A and D cross-sections) is quite symmetrical, while a significant asymmetry is obtained in the vicinity of the highest curvature branches (B and C cross-sections). The velocities in the less irregular branches are coherent with the values obtained from the *Poiseuille's law* in constant diameter pipes with similar geometry and pressure drops. The pressure distribution is consistent with the imposed values at the in/outflow boundaries, with a quite smooth pattern. Due to the winding geometry, overpressure zones

are identified close to the impingement points at the starting of the three outflow branches. In Fig. 6.7.b one of the overpressure region is highlighted using a yellow rectangle, with an enlargement to show the correspondent flow patterns inside the area.



(a)



(b)

Figure 6.7: *Test case* - Sec. 6.3. a) Velocity vectors in the computational domain and velocity contour-lines in the in/outflow cross-sections; b) Pressure field with an enlargement of the velocity vectors in one of the impingement zones (yellow area). Taken from: Monteleone et al. (2017), 19, fig. 16.

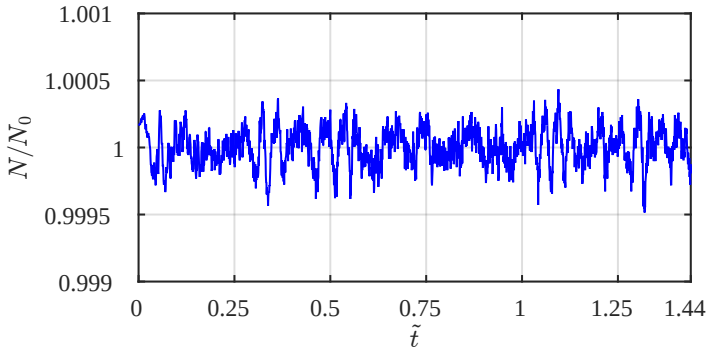
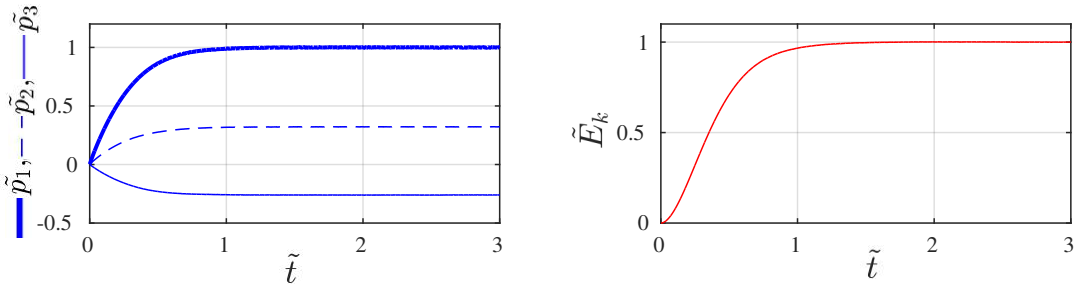


Figure 6.8: *Test case* - Sec. 6.3. Ratio between the number N of the *effective* particles in the computational domain and the initial number $N_0 = 177\,320$ during the simulation. Taken from: Monteleone et al. (2017), 20, fig. 17.

The Fig. 6.8 shows the excellent mass conservation, since the maximum variation $N(t)/N_0$ of the number of particles at the end of each time step (indicated as N) with respect to the initial value (N_0) is limited well below 0.05 % during the whole simulation. Specifically, the number of particles in the figure is plotted against the non-dimensional time $\tilde{t} = tU_i/L_v$, where U_i and L_v are the regime bulk inflow velocity and vessel axial length, respectively. During the simulation the opening angle β (see Sec. 3.2.2) ranged between the assigned lower and upper bounds (5° and 45°), stabilizing its value at about 28° after the steady-state was achieved.

The evolution in time of the momentum components (indicating with $\tilde{p}_r$ the projection in the r-th direction) and kinetic energy $\tilde{E}_k$ in the whole domain are plotted in Fig. 6.9. The plots are obtained summing up the values relative to the whole set of domain particles and are made non-dimensional with the regime values (the x_1 component is used to make non-dimensional the three momentum projections). The figures show that the regime conditions are achieved after less than 1.5 L_v/U_v cycles.



(a) (b)

Figure 6.9: *Test case* - Sec. 6.3. a) Dimensionless momentum components ($\tilde{p}_1$, $\tilde{p}_2$, $\tilde{p}_3$); b) Dimensionless kinetic energy ($\tilde{E}_k$). Taken from: Monteleone et al. (2017), 20, fig. 18.

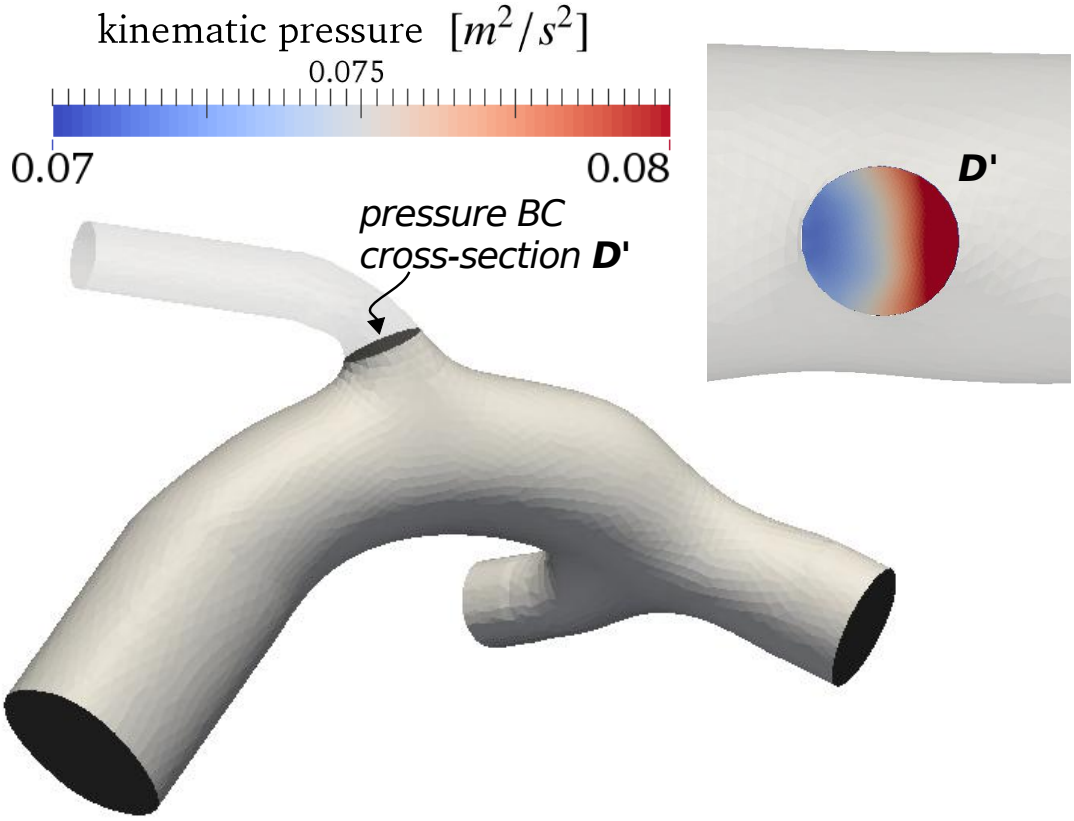


Figure 6.10: *Test case* - Sec. 6.3.1. Domain with cut branch: *pressure BCs* at the new outlet section D'. Taken from: Monteleone et al. (2017), 20, fig. 19.

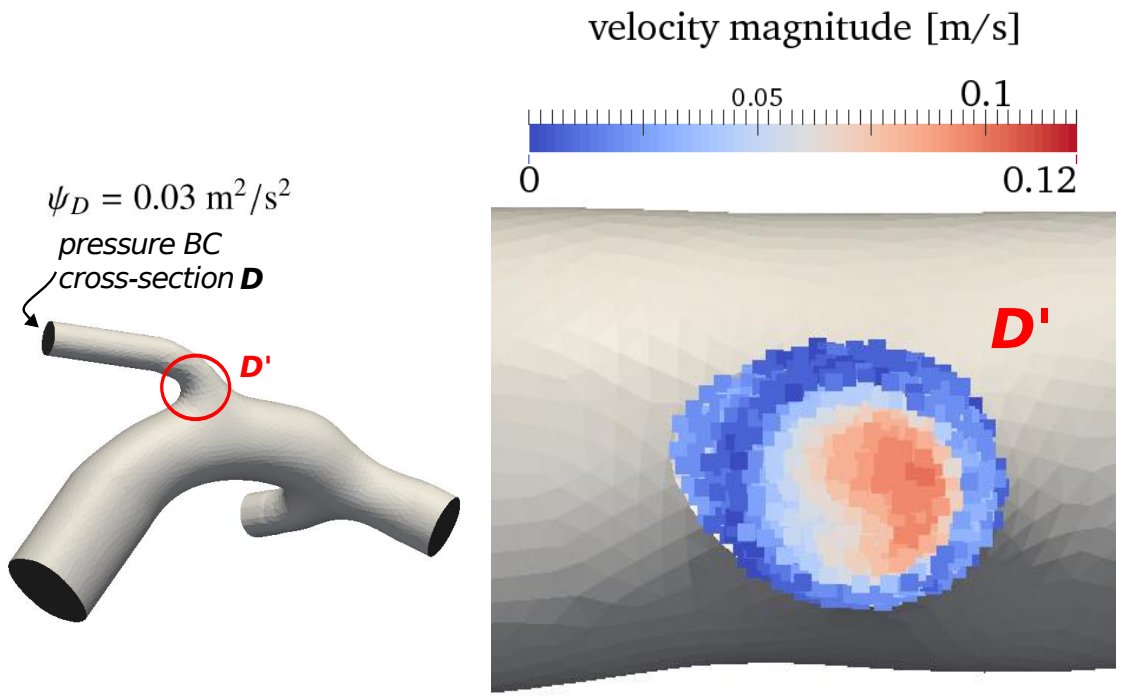
6.3.1 Verifying *non-reflective* properties

In order to assess the *non-reflective* properties of the *In/OutFlow-BCs* procedure at the *open-boundaries*, a second simulation has been performed after having cut the smallest branch at the D' cross-section, as shown in Fig. 6.10. The kinematic pressure in the D' section in the new simulation is imposed using the corresponding values obtained in the first simulation at the steady-state, ranging between 0.07 and 0.08 m^2/s^2. Differently from the previous case (simulation of Sec. 6.3), in fact, the kinematic pressure is variable along the outflow section due to the significant flow curvature in the region, as seen in Fig. 6.7.b.

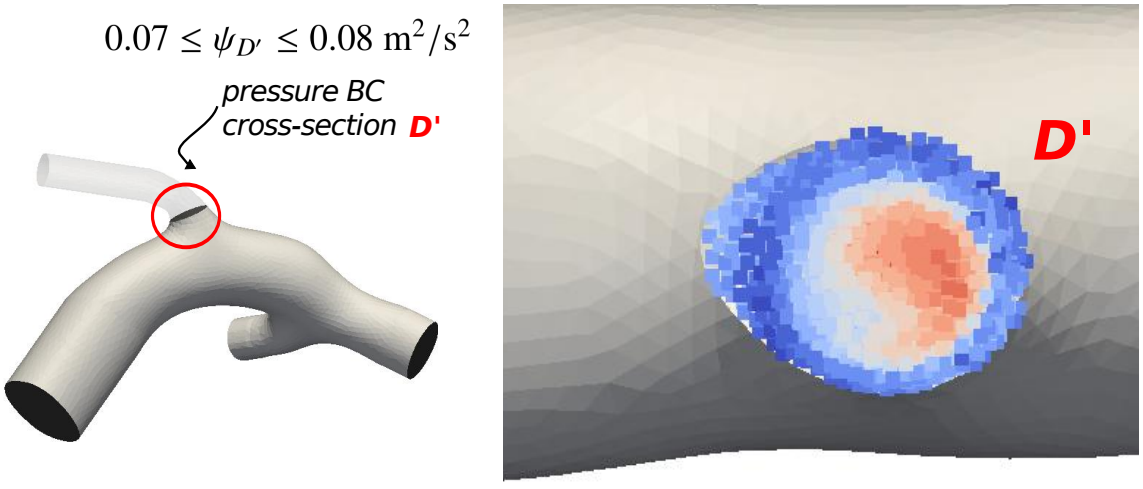
The second simulation has been started from the rest and carried on till the steady-state as well, with the aim to compare the pressure and velocity patterns in the two simulations, in order to identify the effect of the domain cutting.

The velocity magnitude contours at the D' cross-section are shown in the Fig. 6.11 as obtained in the two test cases: when imposing the constant $\psi_D = 0.03\ m^2/s^2$ value at the D section (Fig. 6.11.a) and the variable $\psi_{D'}$ distribution at the D' section (Fig. 6.11.b). The figure shows that a quite similar pattern has been obtained. The resulting differences in both the mean and maximum velocity in the D' cross-section are lower then 1.5% (specifically 1.1% and 1.5 %, respectively). The pressure distribution (not shown) is virtually indistinguishable in the common part of the two simulations.

The results show that the *In/OutFlow-BCs* method allows to obtain quite regular



(a)



(b)

Figure 6.11: *Test case* - Sec. 6.3.1. Velocity magnitude contours at the D' cross-section; a) First simulation (whole domain): *pressure BCs* at the D outlet section (Sec. 6.3); b) second simulation (cut branch): *pressure BCs* at the D' outlet section. Taken from: Monteleone et al. (2017), 21, fig. 20.

particle outflow and correct velocity patterns when imposing the pressure distribution, without regard to the outlet section position.

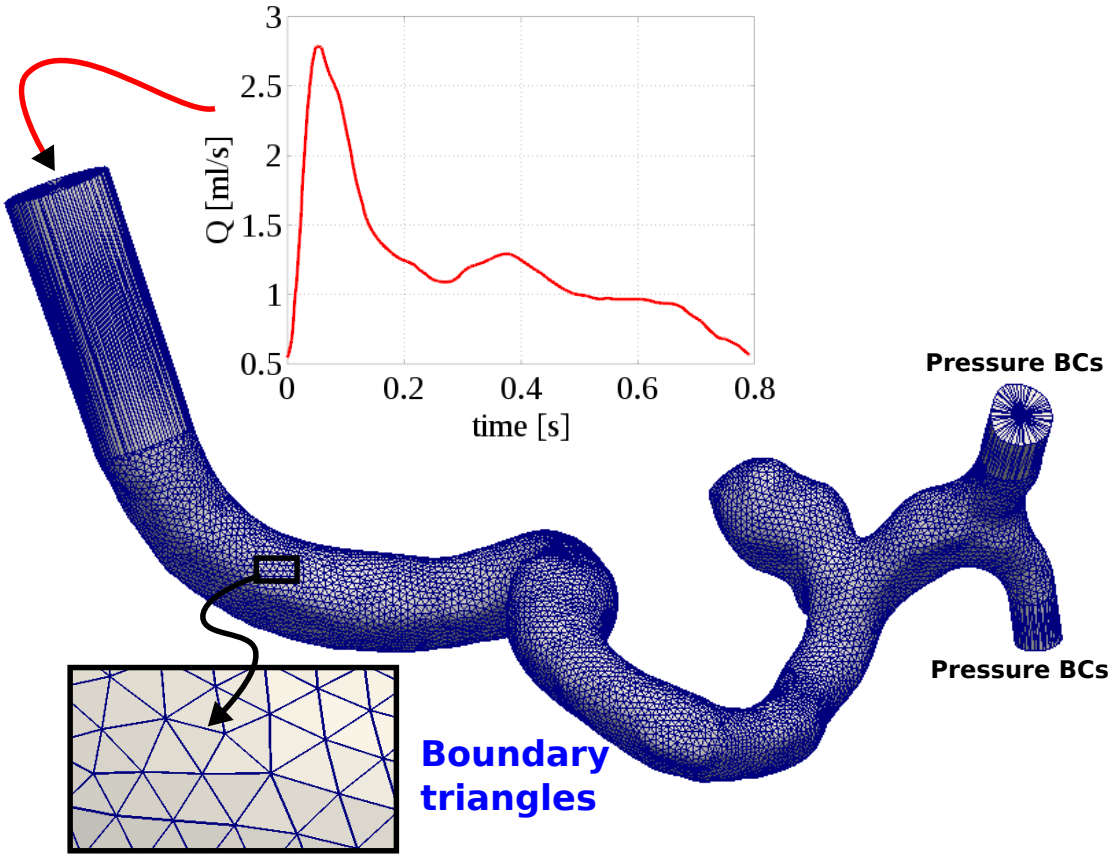


Figure 6.12: *Test case* - Sec. 6.4. Domain geometry and boundary conditions.

6.4 Aneurysm *C05*

The test case $C05$ of the *Aneurisk* dataset repository has been analyzed. The Fig. 6.12 shows the computational domain with the boundary triangles which discretize the surface (an enlargement is shown in the figure inside the black rectangle). Differently from the previous case, at the inlet section, having diameter $D = 0.00366\ m$, a time-varying flux has been imposed using a typical waveform of healthy individual (Radaelli et al., 2008; Marzo et al., 2011) with period $T = 0.792\ s$ (plotted in red in the figure) and *Womersley* velocity profile (eqn. 3.3). The resulting *Womersley number* is $Wo = 2.65$, while the mean value of the *Reynolds number* over a cardiac cycle is 112 (which is within the laminar regime). Zero pressure has been assigned at the outlets. The adaptive time step procedure has been employed as discussed in Sec. 2.9.

In order to reduce the effect of initial transients, the sixth of six simulated cardiac cycles has been analyzed.

The Fig. 6.13 shows the instantaneous particle velocity magnitude at the peak systole. The velocity contours are shown in Fig. 6.14 where the particle velocity (see 6.14.a) and the velocity vectors (see 6.14.b) are represented in a region near the aneurysm considering the cross section of Fig. 6.14.c. The velocity vectors clearly show the vortex formed inside the aneurysm sac.

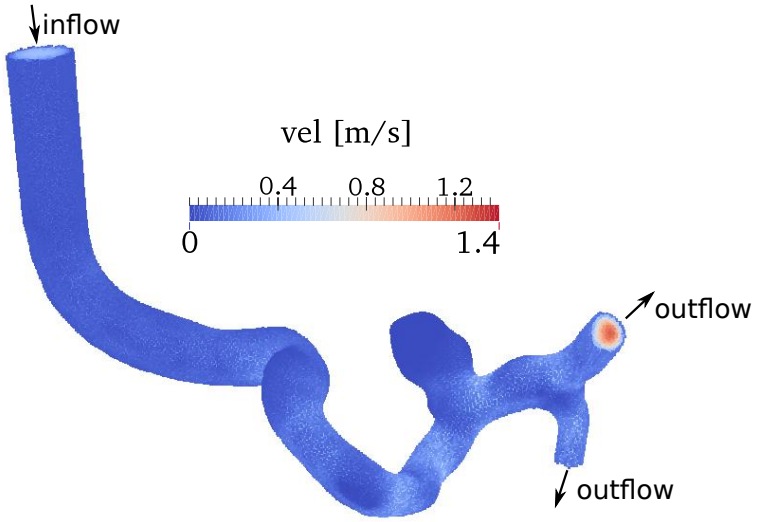
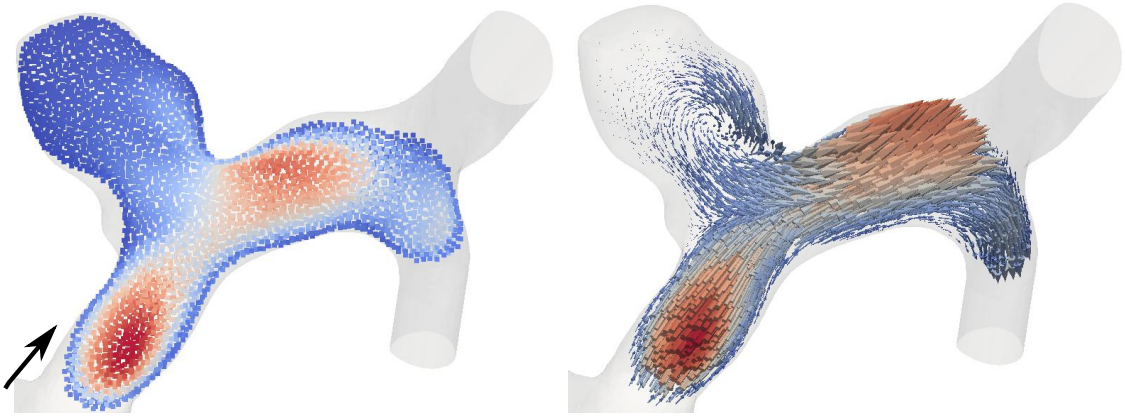


Figure 6.13: *Test case* - Sec. 6.4. Particle velocity magnitude in $[m/s]$ at peak systole.



(a) Particle velocity

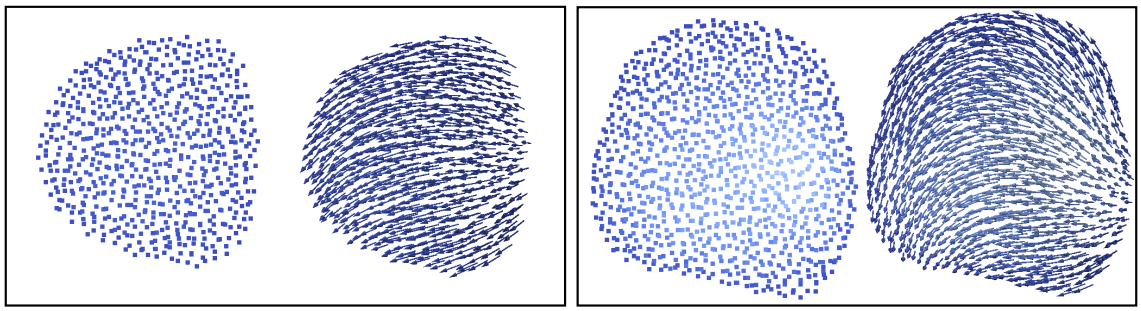
(b) Velocity vectors

(c) Cutting plane

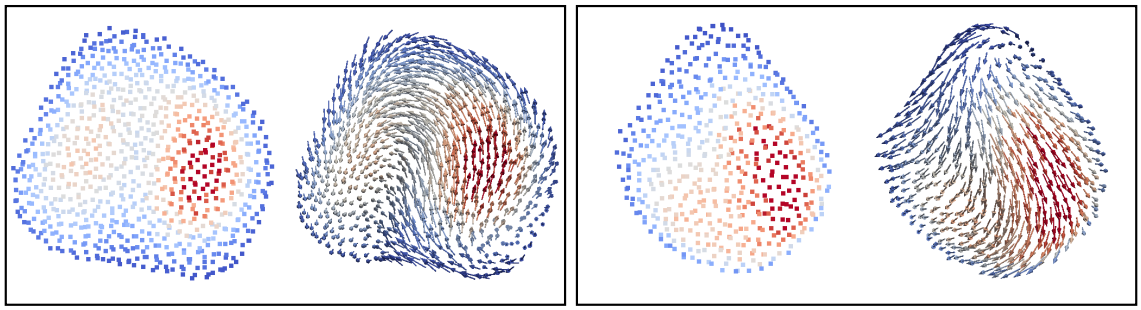
Figure 6.14: *Test case* - Sec. 6.4. Velocity contours: particle and vector representations (subfigure .a and .b, respectively) at the section indicated in the subfigure .c. Same scale of Fig. 6.13

A more detailed view of the complex aneurysm hemodynamics are provided at the peak systole considering different planes. Specifically, four slices have been considered along the aneurysm dome (see the scheme in Fig. 6.15.e), where the particle velocity (on the left) and vectors (on the right) are represented with a top view. The vectors are colored with the scale of the velocity magnitude, while a constant dimension has been chosen for the arrays in order to clearly show the flow pattern that is very different moving from the aneurysm neck to the top of the aneurysm sac (see Figs. 6.15.a.b.c.d).

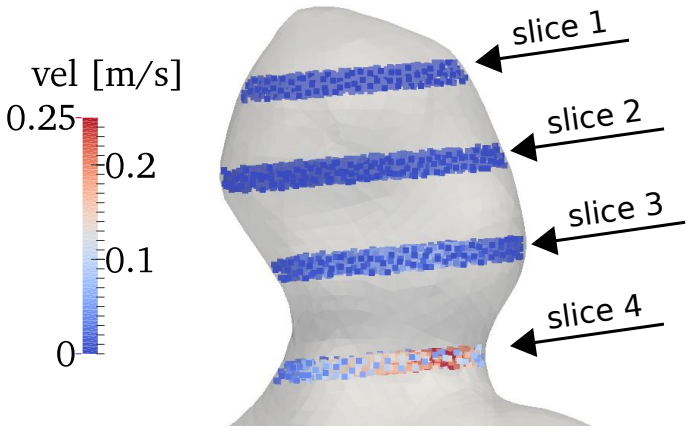
The Fig. 6.16 shows the pressure evolution against the non-dimensional time (t/T) at selected cross-sections whose intersection points with the *centerline* are indicated with



(a) Slice 1 (b) Slice 2



(c) Slice 3 (d) Slice 4



(e) Slice scheme

Figure 6.15: *Test case* - Sec. 6.4. Particle velocity on the left and vectors on the right at the peak systole considering four different slices.

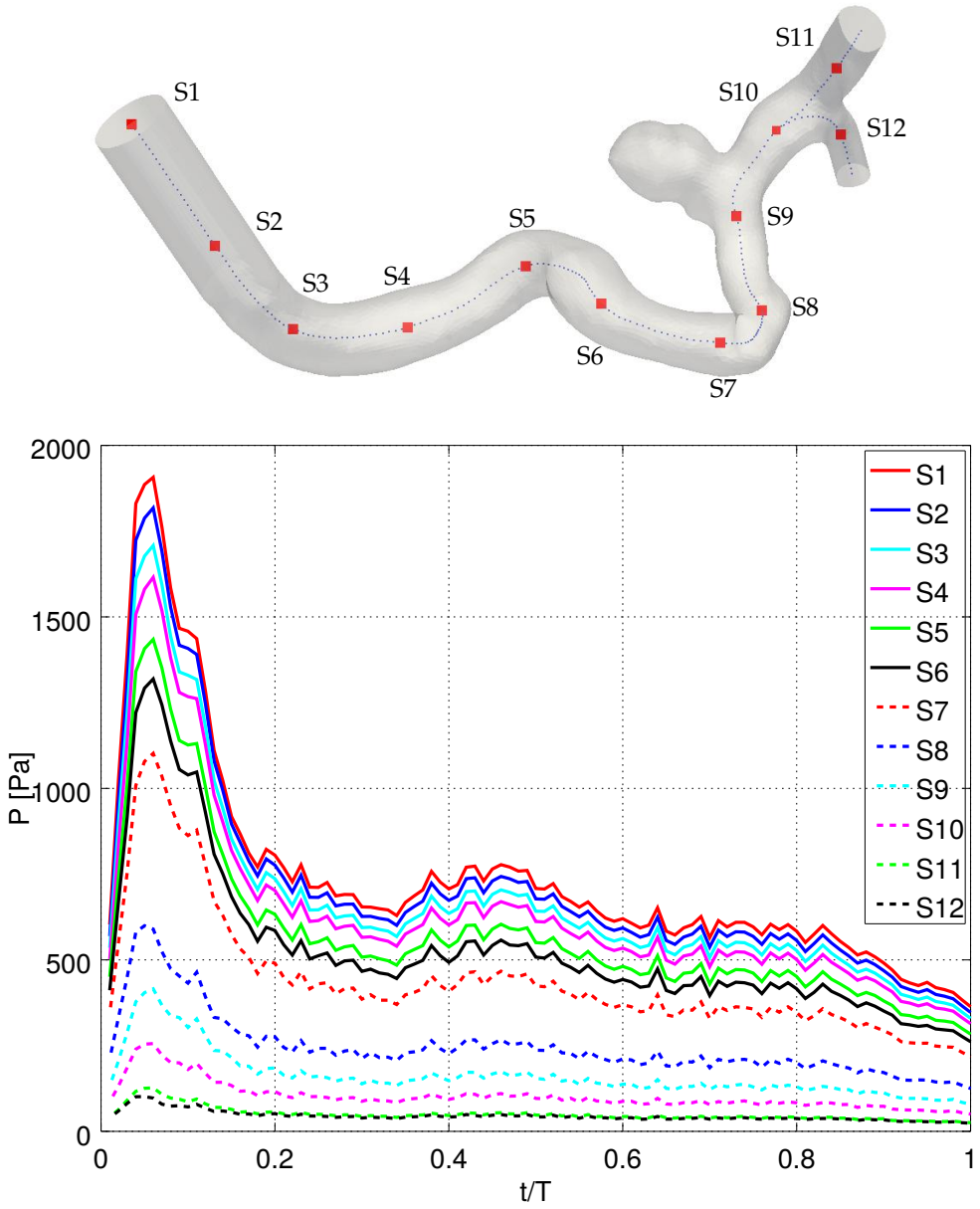


Figure 6.16: *Test case* - Sec. 6.4. Time-dependent pressure in $[Pa]$ at different cross-sections.

the letter S from 1 to 12. The figure clearly shows that the pressure follows the pattern of the flux. The technique used for calculating the pressure values at selected cross-sections is explained below. The *centerline* has been obtained using the *Vascular Modelling Toolkit*, a collection of libraries and tools for 3D reconstruction, geometric analysis, mesh generation and surface data analysis for image-based modeling of blood vessels (available at http://www.vmtk.org). A FORTRAN code has been created in order to modify the *centerline* for obtaining points with the required relative distance equal to $kh/100$. The step size of $kh/100$ has been used to avoid abrupt slope variations between two consecutive *centerline* segments. The coefficients of the planes passing through points with step of

kh (one point every 100) of the *centerline* and normal to the line connecting the points immediately on the left ($\mathbf{x} - kh/100$, with $\mathbf{x}$ the coordinates of the selected point) and on the right ($\mathbf{x} + kh/100$) has been obtained. A test file containing the *centerline* points with step of kh and the relative coefficients of the plane has been created. This file is read at the beginning of the simulation by the *PANORMUS-SPH* code. In each of these *centerline* points the pressure has been obtained as the average of the Taylor series expansions carried out around all the *effective* particles with distance from the *centerline plane* ranging between $\pm\Delta x$. For the sake of clarity, the representation in the figure is limited to twelve points of the *centerline*.

6.4.1 Calculation of Wall Shear Stress variables

Hemodynamics is widely believed to play an important role in the initiation, evolution and rupture processes of *CAs*. The characterization of state stress of the vessel walls is an active research area trying to correlate indices *WSS-related* with the risk of *CA* rupture. In this research study some *WSS* indices have been calculated.

The concept of *WSS* refers to the tangential, frictional stress exerted by the action of blood flow on the vessel walls. On the other hand, the *SPH* method allows to obtain hydrodynamic values at the position of the particles. Therefore, an extrapolation procedure has been used to obtain wall values starting from the particles ones. Specifically, in order to obtain the values from the interior of the fluid domain to the surface walls, the *WSS* has been calculated in each boundary triangle centroid $\mathbf{x}_c$ using the *Cauchy theorem*. To calculate the stress tensor, the velocity derivatives have been obtained adopting the *Basic Gradient Approximation formula* (eqn. 2.6) which allows using only the velocity values of the particles neighboring the point $\mathbf{x}_c$ (since the velocity at the $\mathbf{x}_c$ point is unknown). Therefore, the derivative of the α-th velocity component in the β-direction at the point $\mathbf{x}_c$ has been calculated as

$$\frac{\partial u_\alpha}{\partial x_\beta} = -\sum_{j=1}^{N} \frac{m_j}{\rho_j} \frac{\partial W_{c,j}}{\partial x_\beta}$$

where the sum is extended to the total number of particles N having distances shorter than kh from the centroid of the triangle (whose *support domain* Ω_c is represented in Fig. 6.17.a) and $W_{c,j}$ is the *kernel function* considering the distance between the point $\mathbf{x}_c$ and the neighboring particle j. The stress $\mathbf{\Phi}_n$ on the triangle surface of normal direction $\mathbf{n}$ and its magnitude $|\mathbf{\Phi}_n|$ can be obtained as follows

$$\begin{aligned} \Phi_{n,\beta} &= 2\nu\rho \sum_{\alpha=1}^{3} n_\alpha \frac{\partial u_\alpha}{\partial x_\beta} \text{ , with } \quad \beta = 1,2,3 \\ |\mathbf{\Phi}_n| &= \sum_{\beta=1}^{3} n_\beta \, \Phi_{n,\beta} \end{aligned}$$

where n_α is the α-th component of the vector $\mathbf{n}$.

Thus, the *WSS* vector $\boldsymbol{\tau}_w$ (equal to the tangential component of $\mathbf{\Phi}_n$) is calculated by subtracting from $\mathbf{\Phi}_n$ its normal component ($|\mathbf{\Phi}_n| \cdot \mathbf{n}$)

$$\boldsymbol{\tau}_w = \mathbf{\Phi}_n - |\mathbf{\Phi}_n| \cdot \mathbf{n} \tag{6.1}$$

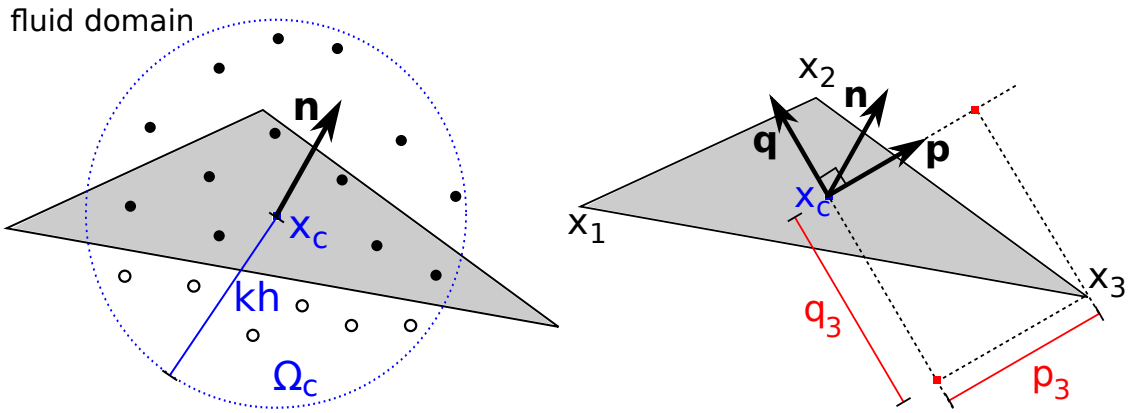


Figure 6.17: Triangle t (gray area) with normal vector $\mathbf{n}$ and centroid c. a) *Support domain* Ω_c of the triangle centroid; full and empty black circles: *effective* and *mirror* particles, respectively, in Ω_c; b) p- and q-directions.

The magnitude of the instantaneous *WSS* vector (defined at the surface point $\mathbf{x}_c$ of each boundary triangle and at time t) can be easily obtained as

$$|\boldsymbol{\tau}_w| = \sqrt{\sum_{\beta=1}^{3} \tau_{w,\beta}^2} \tag{6.2}$$

Using the *WSS* values, five fundamental hemodynamic parameters have been calculated whose definitions have been taken from Geers et al. (2017). A FORTRAN code has been created to calculate these variables in *post-processing*. These parameters have been determined for each boundary triangle t at the position of its centroid as described for the *WSS*. In the following the definitions of the *WSS* indices and the calculation techniques in the *SPH* method are provided.

For pulsatile flow, the *time averaged WSS magnitude* (*TAWSS*) has been calculated by integrating the *WSS* magnitude $|\boldsymbol{\tau}_w|$ of the triangle t over the cardiac cycle T

$$TAWSS_t = \frac{1}{T}\int_0^T |\boldsymbol{\tau}_w|\, dt \tag{6.3}$$

The *gradient* of the *time averaged WSS magnitude* (*TAWSSG*), parameter indicating the state of disrupted flow, has been calculated by taking the spatial derivative of *WSS* as discussed in Lei et al. (2001) and in Miura et al. (2013)

$$TAWSSG_t = \sqrt{\left(\frac{\partial \tau_{w,p}}{\partial p}\right)^2 + \left(\frac{\partial \tau_{w,q}}{\partial q}\right)^2} \tag{6.4}$$

where the p-direction corresponds to the *time-averaged* direction of the *WSS* vector and the q-direction is perpendicular to the p-direction. The unit vectors $\hat{\mathbf{p}}$ and $\hat{\mathbf{q}}$ (in the direction of and perpendicular to, respectively, the *TAWSS* vector) can be obtained as

$$\hat{\mathbf{p}} = \frac{\int_0^T \boldsymbol{\tau}_w\, dt}{\left|\int_0^T \boldsymbol{\tau}_w\, dt\right|}, \qquad \hat{\mathbf{q}} = \hat{\mathbf{p}} \times \hat{\mathbf{n}}$$

where $\hat{\mathbf{n}}$ is unit vector of $\mathbf{n}$.

The derivatives of $\boldsymbol{\tau}_w$ in the p- and q-directions have been obtained assuming a planar distribution of the *WSS* on the triangle surface. Specifically, the projections of $\boldsymbol{\tau}_w$ in the p- and q- directions have been calculated at the three nodes of the triangle and the coefficients of the planes passing through these points have been determined solving the following linear systems

- *FIRST SYSTEM:*

$$\begin{bmatrix} p_1 & q_1 & 1 \\ p_2 & q_2 & 1 \\ p_3 & q_3 & 1 \end{bmatrix} \begin{bmatrix} a_p \\ b_p \\ c_p \end{bmatrix} = \begin{bmatrix} \tau_{w,p_1} \\ \tau_{w,p_2} \\ \tau_{w,p_3} \end{bmatrix}$$

- *SECOND SYSTEM:*

$$\begin{bmatrix} p_1 & q_1 & 1 \\ p_2 & q_2 & 1 \\ p_3 & q_3 & 1 \end{bmatrix} \begin{bmatrix} a_q \\ b_q \\ c_q \end{bmatrix} = \begin{bmatrix} \tau_{w,q_1} \\ \tau_{w,q_2} \\ \tau_{w,q_3} \end{bmatrix}$$

where p_1, p_2 and p_3 are the coordinates of the nodes 1, 2 and 3, respectively, in the p-direction. Likewise, q_1, q_2 and q_3 are the coordinates of the nodes 1, 2 and 3, respectively, in the q-direction. The Fig. 6.17.b shows the projections of the coordinates of the node 3 in the p- and q-direction (p_3 and q_3, respectively). $\tau_{w,p1}$ and $\tau_{w,q1}$ are the components of the *WSS* vector in the p- and q-directions, respectively, calculated at the node 1 (as well as $\tau_{w,p2}$ and $\tau_{w,q2}$ for the node 2 and $\tau_{w,p3}$ and $\tau_{w,q3}$ for the node 3). a_p, b_p and c_p are the coefficients of the plane passing through the points $\tau_{w,p1}$, $\tau_{w,p2}$ and $\tau_{w,p3}$. Likewise, a_q, b_q and c_q are the coefficients of the plane passing through the points $\tau_{w,q1}$, $\tau_{w,q2}$ and $\tau_{w,q3}$. Therefore, the sought derivatives are equal to

$$\frac{\partial \tau_{w,p}}{\partial p} = a_p \ , \quad \frac{\partial \tau_{w,q}}{\partial q} = a_q$$

The *transverse WSS* (*transWSS*) at the triangle t has been calculated as

$$transWSS_t = \frac{1}{T} \int_0^T |\boldsymbol{\tau}_w \cdot \hat{\mathbf{q}}| \tag{6.5}$$

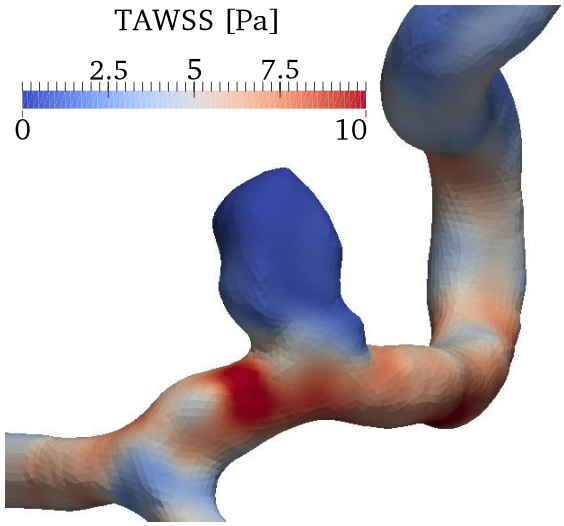
The temporal variation in the *WSS* magnitude during the cardiac cycle can be analyzed through the *WSS pulsatility index* (*WSSPI*)

$$WSSPI_t = \frac{max(|\boldsymbol{\tau}_w|)_t - min(|\boldsymbol{\tau}_w|)_t}{TAWSS_t} \tag{6.6}$$

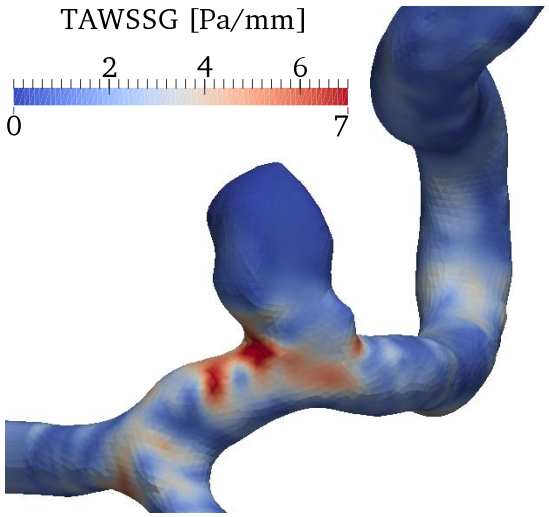
where $max(|\boldsymbol{\tau}_w|)_t$ and $min(|\boldsymbol{\tau}_w|)_t$ are the maximum and minimum τ_w at the triangle t inside the cardiac cycle T.

The *oscillatory shear index* (*OSI*) is a measure of the directional change of *WSS* during the cardiac cycle. This index identifies regions of high cyclic departure of the *WSS* vector from its predominant axial alignment over the cardiac cycle. *OSI* is defined as

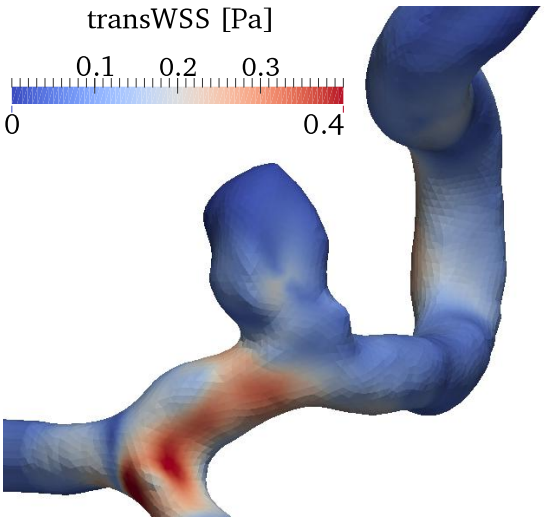
$$OSI = \frac{1}{2}\left(1 - \frac{\left|\int_0^T \boldsymbol{\tau}_w \, dt\right|}{\int_0^T |\boldsymbol{\tau}_w| \, dt}\right) , \ OSI \in \left[0, \frac{1}{2}\right] \tag{6.7}$$



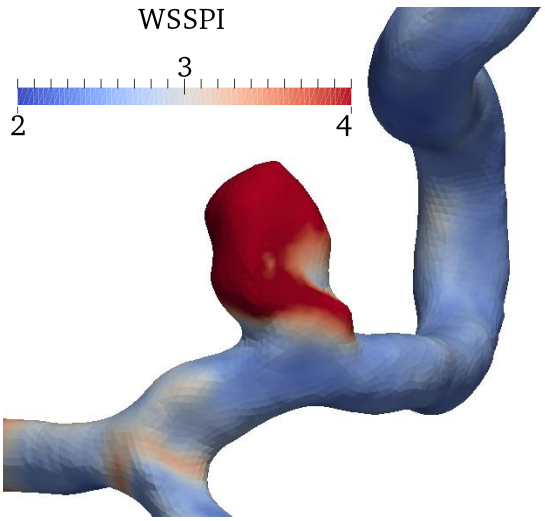
(a) eqn. 6.3



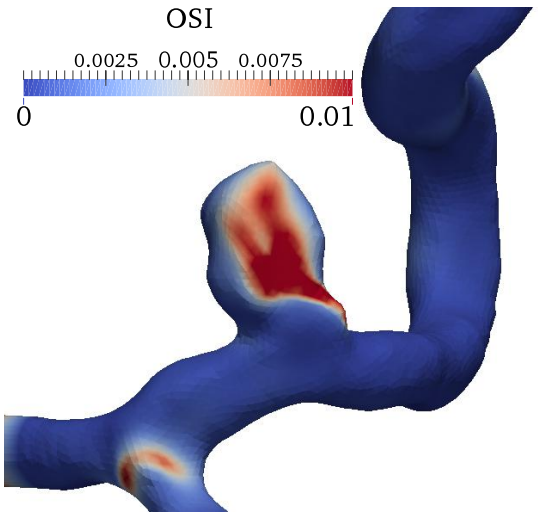
(b) eqn. 6.4



(c) eqn. 6.5



(d) eqn. 6.6



(e) eqn. 6.7

Figure 6.18: *Test case* - Sec. 6.4. Hemodynamic indices distribution.

The Fig. 6.18 shows the five hemodynamic variables calculated for the $C05$ aneurysm. As it is seen in Fig. 6.18.a, a large variations in space-averaged $TAWSS$ is observed with values ranging from 0.1 to 10 $[Pa]$. The Fig. 6.18.b shows the distribution of the $TAWSSG$ where the larger spatial variation in $TAWSS$ is reflected by higher gradients. The distribution of the $transWSS$ is shown in Fig. 6.18.c, WSS vectors change direction more strongly when the flow coming out from the sac goes towards the bifurcation at the outlet. Regions of high $WSSPI$ values (see Fig. 6.18.c) are located at the aneurysm sac, while the upstream values are relatively low. The distribution of the OSI index is shown in Fig. 6.18.e. This index allows to identify regions of highly disturbed flow. High values of OSI are particularly evident at the aneurysm dome due to flow entering into the aneurysm sac forming vortex (see Fig. 6.15) whose shape can change during the cardiac cycle. Moreover, elevated values of OSI can be observed at the outlet bifurcation.

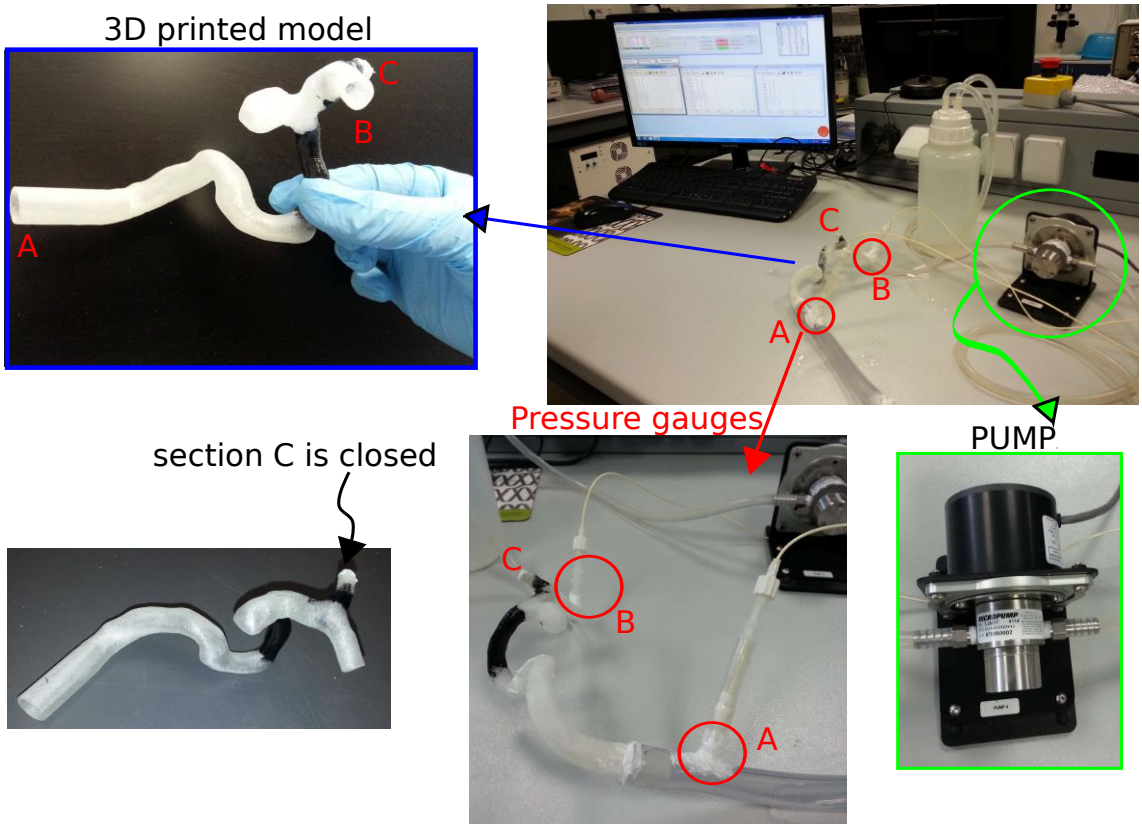


Figure 6.19: *Test case* - Sec. 6.4.2. Experimental setup.

6.4.2 Comparison with experimental measure

A performance evaluation has been conducted comparing numerical results with laboratory application. The experimental setup, shown in Fig. 6.19, is made of a $3D$ printed model of the aneurysm $C05$ using a scale factor of 4, a centrifugal pump (operanting in the range $0 \div 1760\, ml/min$, MICROPUMP), two pressure gauges (MPR 500, Millar Instruments Inc.) and a software processing the pressure measures (Wintest 7, BOSE). The $3D$ printed model has been realized by Prof. Salvatore Vitabile (University of Palermo, BIND department),

while the laboratory experiment has been carried out with the collaboration of the research group of Prof. Massimiliano Zingales (University of Palermo, Bio/Nanomechanics for Medical Science Lab, Aten Center). Differently from the geometry described in Fig. 6.12, in the printed model one of the outlet sections has been closed (section C in the figure). The pressure drop between A and B sections has been measured by the pressure gauges while imposing through the pump a stationary flux of 500 ml/min (the flow rate value has been chosen in order to maintain the laminar flow regime) and using water as working fluid. The measured pressure drop was of about 0.8 $mmHg$.

The simulation of Sec. 6.4 has been repeated using the same scale of the experimental model and imposing *adherence BCs* at the cross-section C (in order to simulate rigid wall), zero pressure at the outlet cross-section B and the same flux of the laboratory experiment at the inlet cross-section A. Moreover, a *Newtonian* fluid of $\rho = 1000\ kg/m^3$ and $\nu = 1 \cdot 10^{-6}\ m^2/s$ has been used.

The obtained numerical pressure drop at the steady-state has been of 0.86 $mmHg$.

The comparison between measured and numerical results confirms the excellent performance of the *SPH* method. Future research activities should aim to encompass laboratory measures of velocity and pressure fields using pulsatile flow.

6.5 Aneurysm *C93*

The simulation of blood flow inside a human cerebral vessel with small branches and a *giant* aneurysm (Morley, 1969) has been performed. Specifically, the geometry *C93* of the *AneuRisk* project database has been selected.

The geometry is very suitable for applying the *Multi-Domain* approach due to the different scale between the branch mean diameters and the aneurysm sac. The computational domain of the *C93* aneurysm has been used in Sec. 4.2.1 (see Fig. 4.2) for describing the partitioning into non-overlapping blocks. A *multi-resolution* approach is necessary since, in order to obtain a sufficiently accurate description of the velocity profile, the *smoothing length* h in each branch should be at least 20 times smaller than the mean diameter. As it is seen in Fig. 4.2, the computational domain has been thus subdivided in six blocks, having different values of kh in the range $1 \div 5 \cdot 10^{-4}\ m$. Specifically, a quite small value of kh was required in block 6 ($kh_6 = 1 \cdot 10^{-4}\ m$), which would have been excessively small for the branches of blocks 1 and 5 and even more for the blocks 2 and 4 corresponding to the aneurysm neck and sac, respectively. Thus, adopting a constant kh in the whole

B_n	$kh\ [m]$	$l_{ref}\ [m]$	l_{ref}/kh
B_1	$2.5 \cdot 10^{-4}$	$2.30 \cdot 10^{-3}$	9.2
B_2	$3.0 \cdot 10^{-4}$	$4.82 \cdot 10^{-3}$	16.1
B_3	$1.5 \cdot 10^{-4}$	$1.15 \cdot 10^{-3}$	7.7
B_4	$5.0 \cdot 10^{-4}$	$9.66 \cdot 10^{-3}$	19.3
B_5	$2.0 \cdot 10^{-4}$	$1.86 \cdot 10^{-3}$	9.3
B_6	$1.0 \cdot 10^{-4}$	$0.86 \cdot 10^{-3}$	8.6

Table 6.1: Non-dimensional refinement value l_{ref}/kh.

domain would have implied a huge number of particles, with a resolution exceedingly high

in most of the domain. Moreover, the pathological dilatation is characterized by relatively low velocities, making even less necessary the fine discretization.

In Tab. 6.1 a reference length l_{ref} has been used for each block to make non-dimensional the refinement value kh. Specifically, the mean diameter of the vessels has been chosen as reference length for the blocks 1, 3, 5 and 6. Due to the extremely irregular shape of blocks 2 and 4 (the aneurysm neck and sac, respectively), the diameter of the sphere having equal volume has been used as the reference distance for these subdomains.

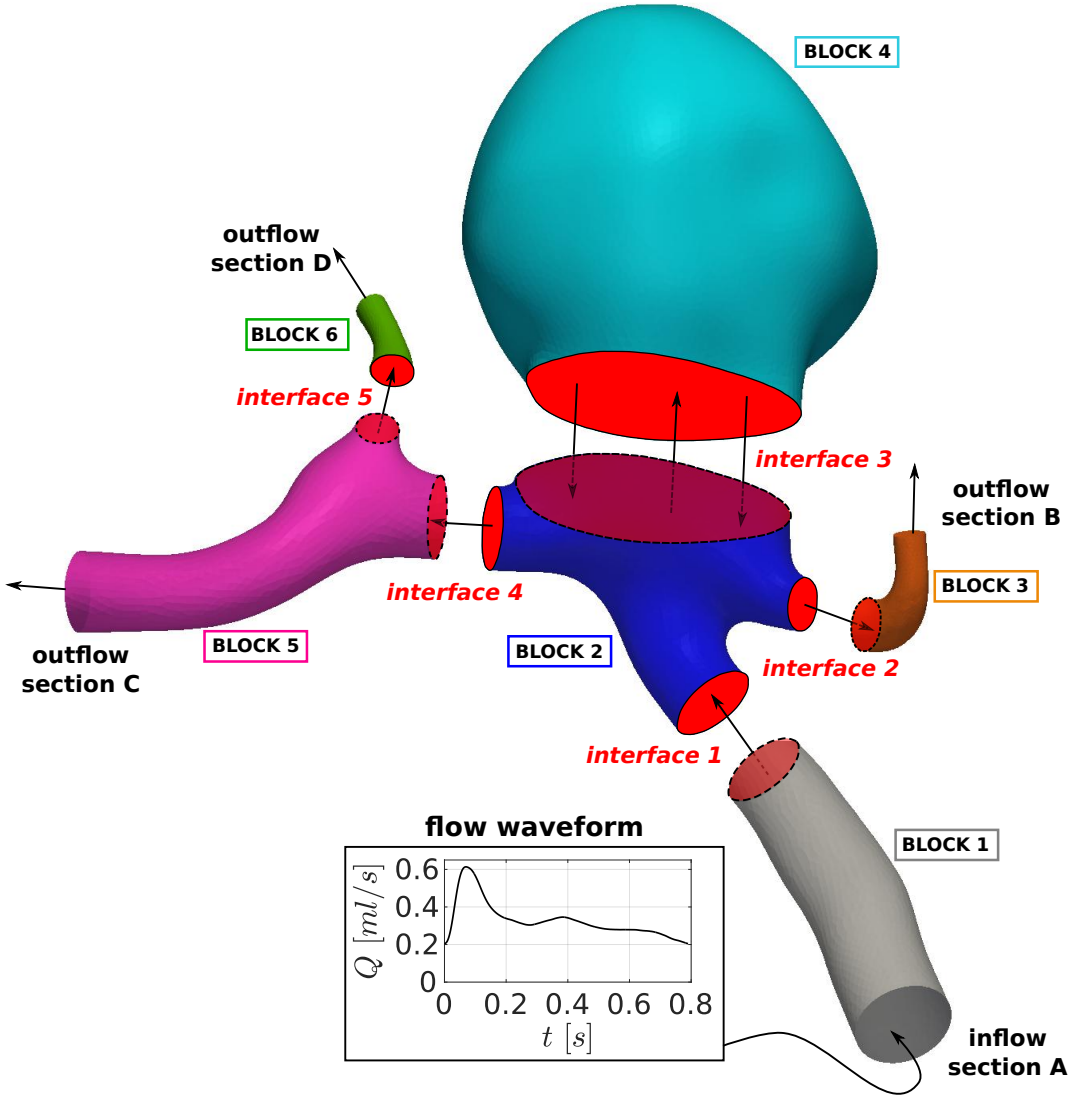


Figure 6.20: *Test case* - Sec. 6.5. Boundary conditions for each block. Taken from: Monteleone et al. (2018), 973, fig. 17.

The resulting total number of particles $N_{e,tot} = 112\,618$ is about 2% of the number that would have been obtained using the smallest kh value ($1 \cdot 10^{-4}\ m$) in the whole domain.

The boundary conditions for each block are shown in Fig. 6.20. At the inlet section A, having diameter $D = 0.0023\ m$, pulsatile flow condition has been prescribed by imposing *incoming BCs* (as described in Sec. 3.2.1) at the inlet triangles. To this aim, the same waveform used for the aneurysm $C05$ (see Sec. 6.4) has been employed with period $T =$

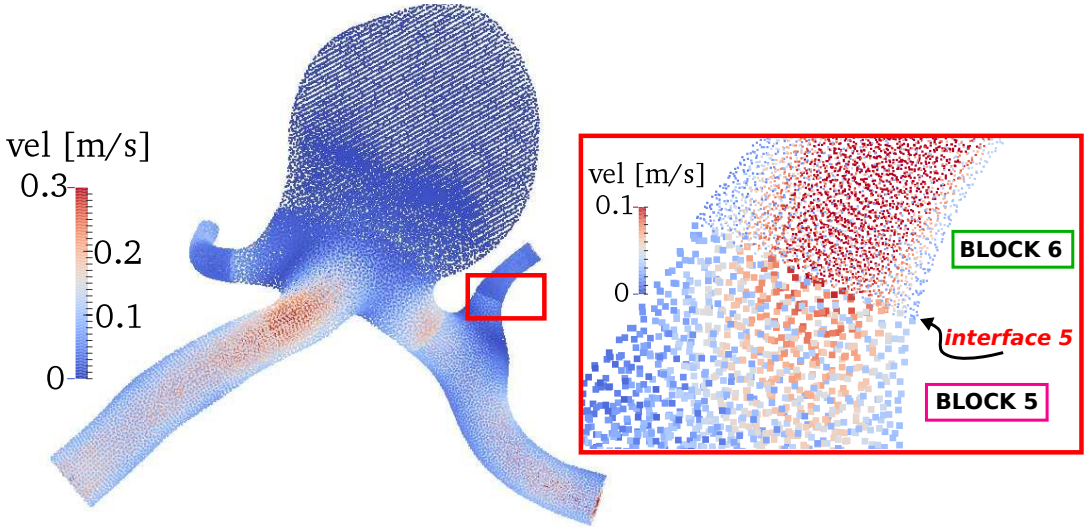


Figure 6.21: *Test case* - Sec. 6.5. Particle velocity magnitude in $[m/s]$. Enlargement at *interface* 5 (red rectangle) separating block 5 from block 6. Taken from: Monteleone et al. (2018), 973, fig. 18.

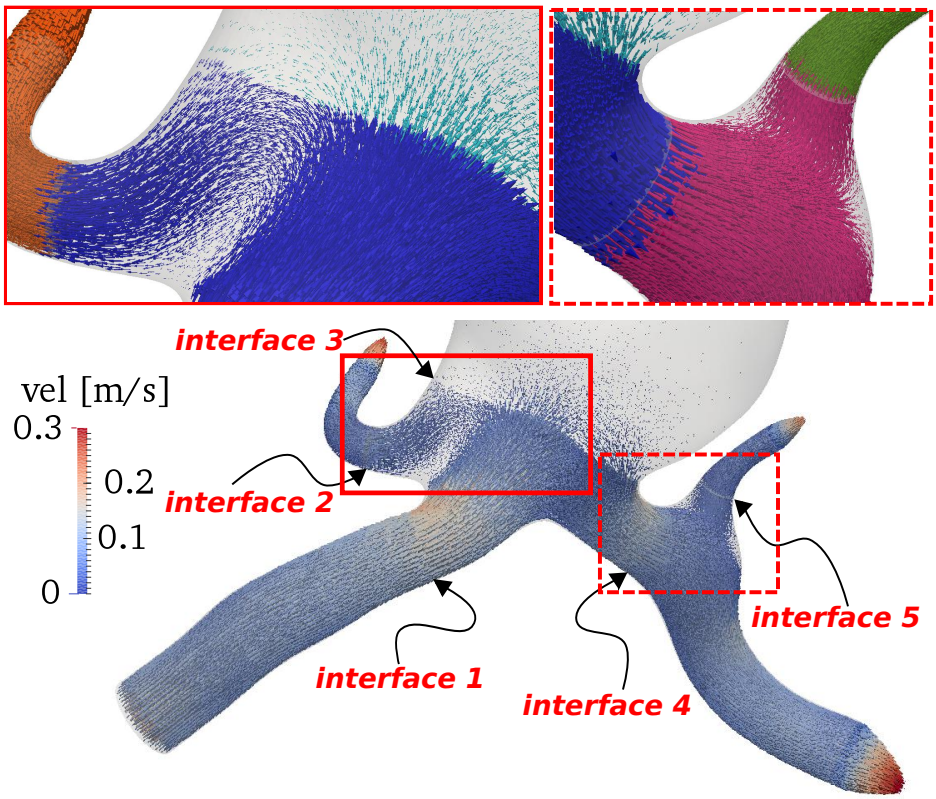


Figure 6.22: *Test case* - Sec. 6.5. Velocity vectors. Enlargements at *interfaces* 2 and 3 (bold red line) and *interfaces* 4 and 5 (dashed red line) using different colors to indicate velocity vectors relative to particles belonging to different blocks (see scheme in Fig. 4.2 for the colors). Taken from: Monteleone et al. (2018), 974, fig. 19.

0.792 s and time-averaged flow rate of about 0.3 ml/s obtained following the power law relationship between the flow rate and the cross-sectional area proposed by Cebral et al. (2008).

The resulting mean value of the *Reynolds number* over a cardiac cycle is 690. The velocity profile at the inlet has been thus obtained applying the *Womersley* solution through eqn. 3.3 (*Womersley number* $Wo = 1.7$). A constant pressure has been imposed at the outlet sections (B, C and D in the figure), while at the five *block interfaces* (red areas in the figure) the *Multi-Domain* procedure described in Chap. 4 has been applied.

The simulation has been performed over six cardiac cycles and the results of the last cycle have been considered. The adaptive time-step procedure, discussed in Sec. 2.9, has been employed.

The Fig. 6.21 shows the particle velocity magnitude in the whole domain and an enlargement in the vicinity of *interface* 5 which highlights the very good matching of the solution in the neighboring blocks (5 and 6). Further enlargements are shown in Fig. 6.22 where the velocity vectors are plotted.

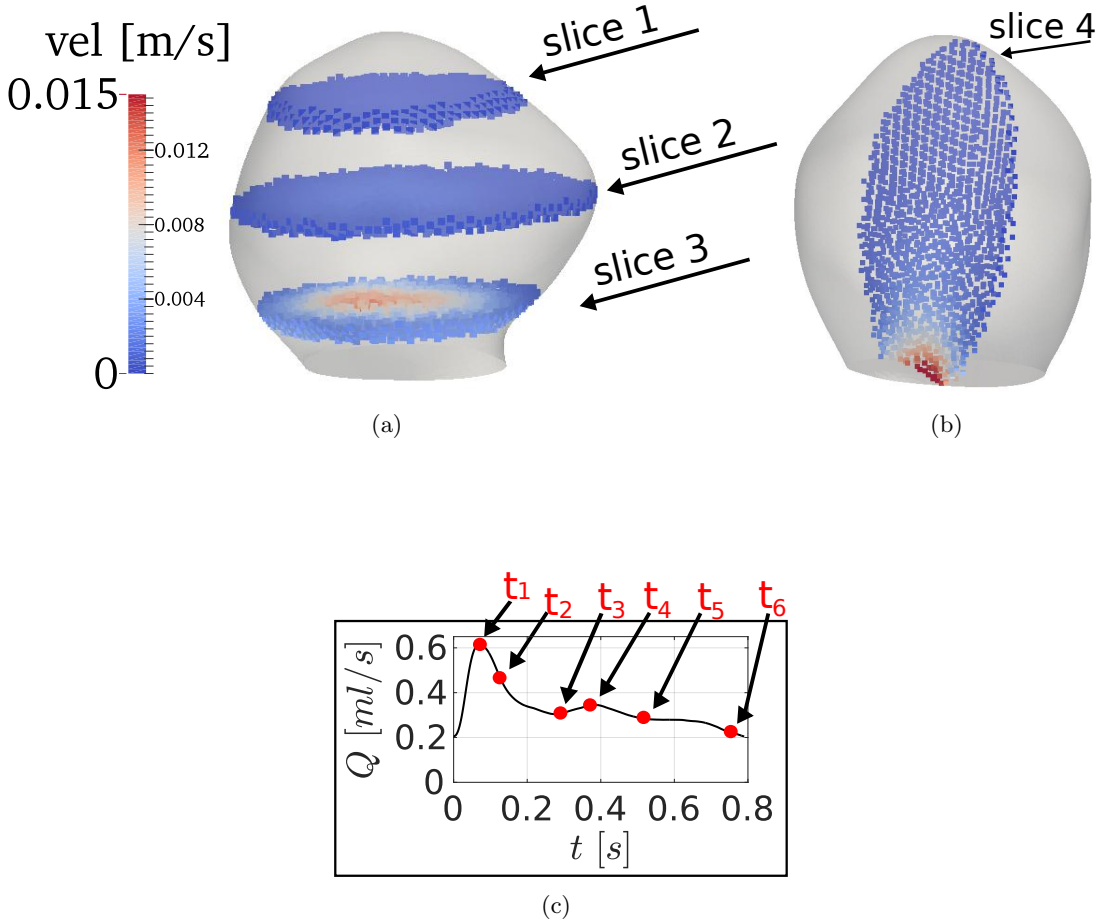
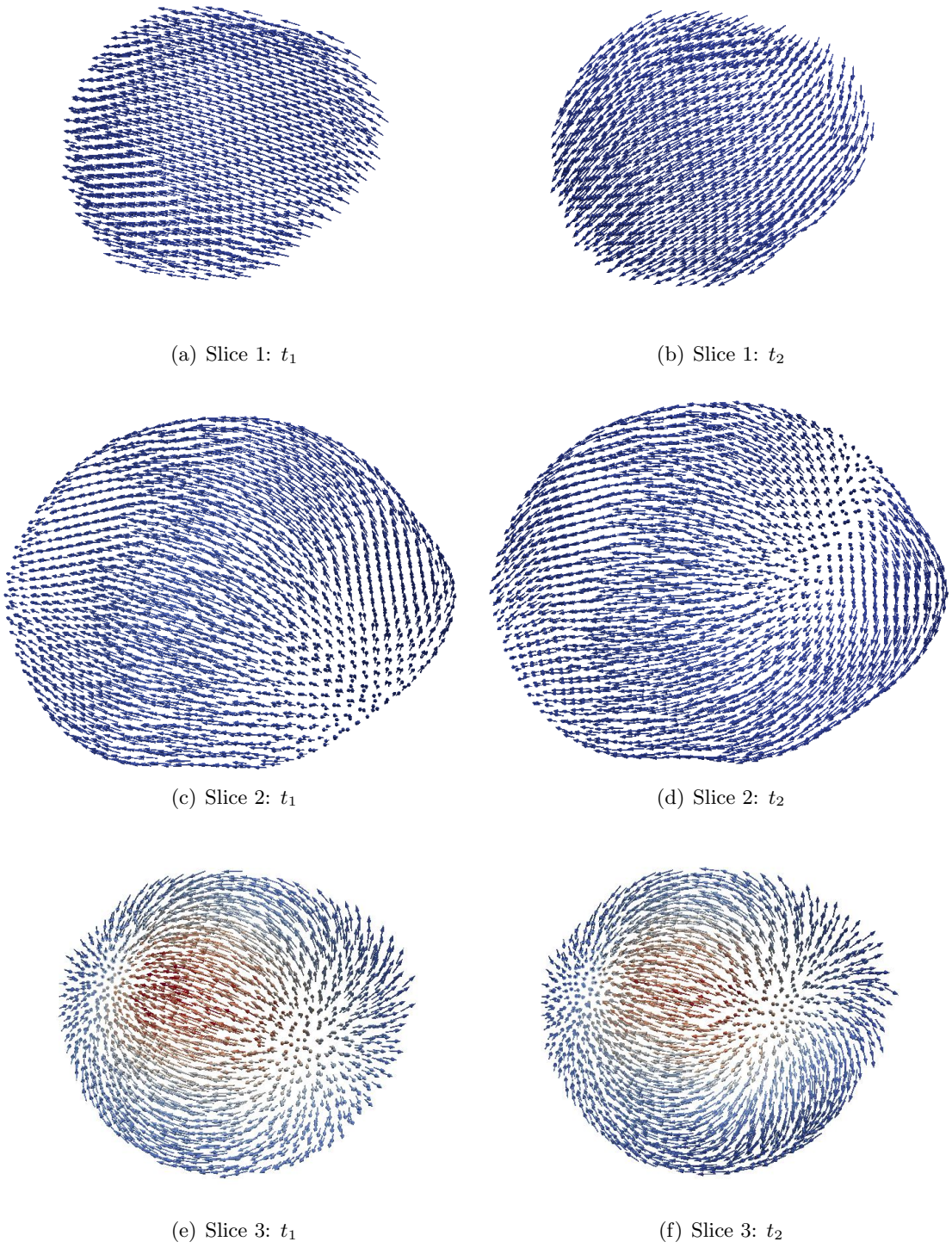


Figure 6.23: *Test case* - Sec. 6.5. Particle velocity at different slices and at the peak systole (time t_1 shown in the same figure at point $.c$).

The Figs. 6.23.a and 6.23.b show the particle velocity considering four different slices along the aneurysm sac (hence only the fourth block is considered in the figures) at the peak systole (time t_1 in the Fig. 6.23.c). The velocity vectors at time t_1 and t_2 (indicated



(a) Slice 1: t_1

(b) Slice 1: t_2

(c) Slice 2: t_1

(d) Slice 2: t_2

(e) Slice 3: t_1

(f) Slice 3: t_2

Figure 6.24: *Test case* - Sec. 6.5. Velocity vectors at different slices (see the scheme in Fig. 6.23.a) and instants of time (see Fig. 6.23.c).

in the waveform of Fig. 6.23) and taking into account the three slices of Fig. 6.23.a are shown in Fig. 6.24 which provides a more detailed description of the flow patterns. The

vectors are colored with the velocity magnitude (using the same scale of Fig. 6.23), while the length of the array is constant in order to make it independent from the velocity which dramatically decreases going from the neck to the top of the aneurysm dome. The figure helps to highlight the complexity of the flux inside the aneurysm, with patterns varying markedly throughout the cardiac cycle.

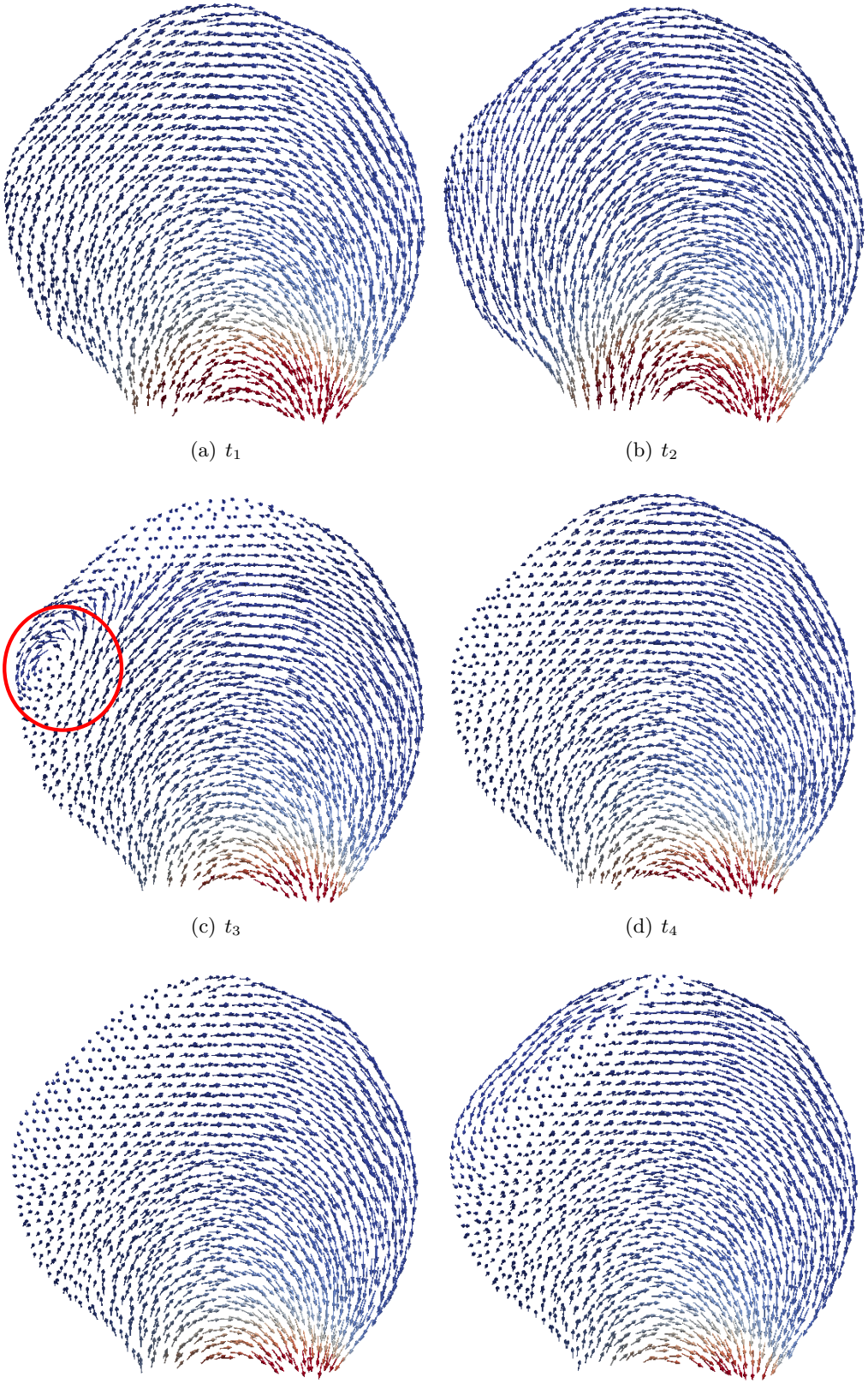
More details are provided in Fig. 6.25 showing the velocity vectors at different times of the waveform (see Fig. 6.23.c) along the longitudinal slice represented in Fig. 6.23.b. In the figure it is clearly seen the variability of the flow pattern inside the dome of the aneurysm where vortices form and rapidly dissolve. A vortex is indicated inside the red circle in Fig. 6.25.c that disappears in the following Fig. 6.25.d.

The pressure evolution in eight points distributed in the blocks along the vessel *centerline* is shown in Fig. 6.26 during one cardiac cycle time period using the procedure described in Sec. 6.4. The plotted patterns are coherent with the imposed velocity flux at the domain inflow (where the point $P1$ is placed) and show a correct time evolution of the pressure values from one block to the others, with correct pressure drops while moving in the downstream direction along the *centerline*.

In order to show the satisfaction of the mass conservation, the volume discharges in different branch cross-sections have been calculated based on the number of particles going through the corresponding sections during a fixed amount of time ($\Delta t = 0.002\ s$). The obtained discharge Q_1 in the cross-section S_1 of the inflow vessel, shown in Fig. 6.27.a, has been compared with the sum of the discharges Q_3 and Q_5 in two cross-sections of the downstream branches 3 and 5 (S_3 and S_5 in the figure), showing that the continuity constraint is correctly obeyed ($Q_1 = Q_3 + Q_5$).

Since a further bifurcation occurs in block 5 downstream of the S_5 cross-section, the discharge Q_5 is compared in Fig. 6.27.b with the sum of the discharges in the downstream cross-sections $S_{5'}$ (in the same block 5) and S_6 (in block 6), showing again that the mass conservation is fulfilled. It is worthwhile highlighting that the satisfaction of the continuity equation in cross-sections belonging to different subdomains in a Lagrangian method, although being a necessary requirement for a reliable solution, is not automatically guaranteed, since it is not explicitly enforced in the solved equations.

A further confirmation of the mass conservation satisfaction is obtained in Fig. 6.28 showing the time evolutions of the particle number N_n ($n = 1, .., 6$) in each of the six blocks normalized with the corresponding initial number of particles $N0_n$ ($N0_1 = 18\,241$, $N0_2 = 17\,336$, $N0_3 = 5\,735$, $N0_4 = 30\,160$, $N0_5 = 32\,207$ and $N0_6 = 9\,033$). The maximum amplitude of the oscillations in each block is always lower then 0.1% with the exception of the initial stages of the first cardiac cycle where values of about 0.2% have been achieved. This result confirms the effectiveness of the proposed dynamic adjustment of the cone angle amplitude β at the inlet *interface* of each block discussed in Sec. 4.2.4.

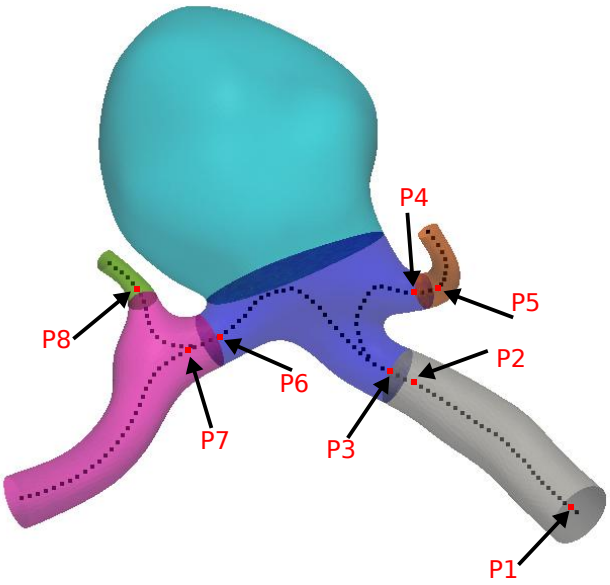


(a) t_1 (b) t_2

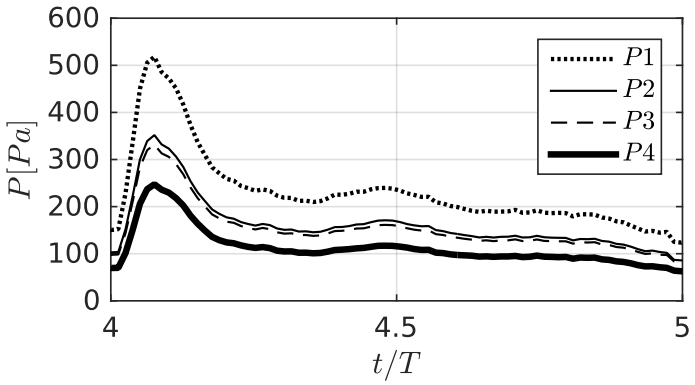
(c) t_3 (d) t_4

(e) t_5 (f) t_6

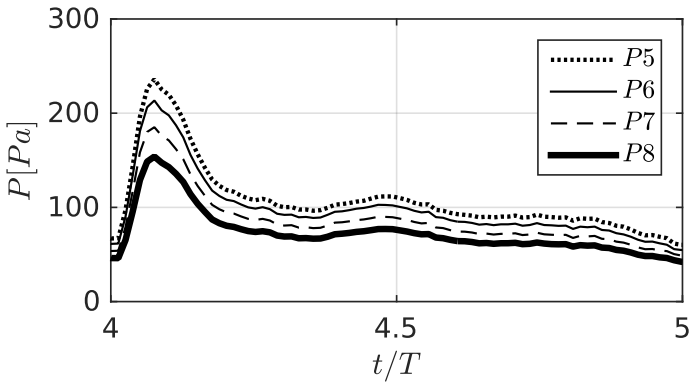
Figure 6.25: *Test case* - Sec. 6.5. Velocity vectors at the slice 5 shown in Fig. 6.23.b and at different time instants (see Fig. 6.23.c).



(a) Scheme

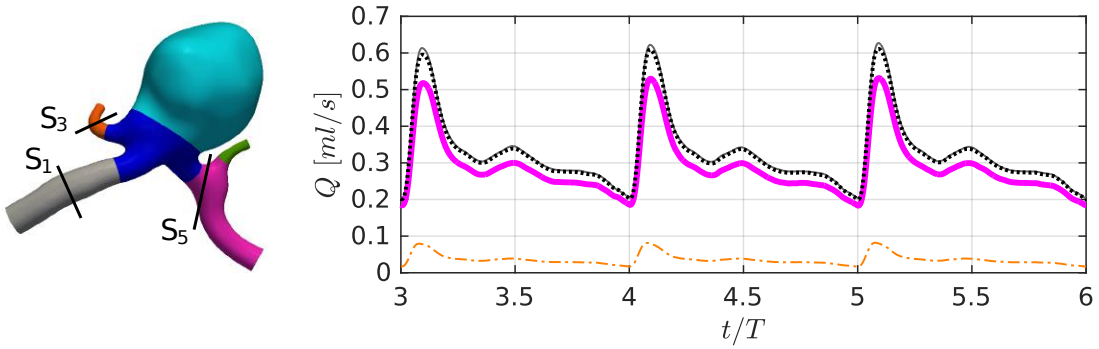


(b)

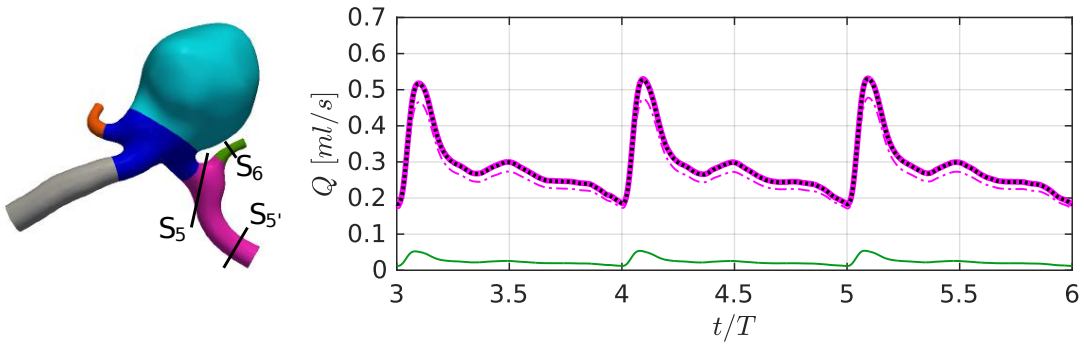


(c)

Figure 6.26: *Test case* - Sec. 6.5. Pressure in $[Pa]$ over one cardiac cycle in different points of the centerline ($P1$, $P2$, $P3$, $P4$, $P5$, $P6$, $P7$, $P8$). Taken from: Monteleone et al. (2018), 974, fig. 20.



(a)



(b)

Figure 6.27: *Test case* - Sec. 6.5. Flow rates $Q(t)$ in $[ml/s]$ over the last three cycles through the sections S_1, S_3, S_5, $S_{5'}$ and S_6 (bold black lines in the domain sketch). a) Thin grey line: Q_1; dash-dot orange line: Q_3; magenta bold line: Q_5; dotted black line: $Q_3 + Q_5$; b) dash-dot magenta line: $Q_{5'}$; thin green line: Q_6; dotted black line: $Q_6 + Q_{5'}$. Taken from: Monteleone et al. (2018), 975, fig. 21.

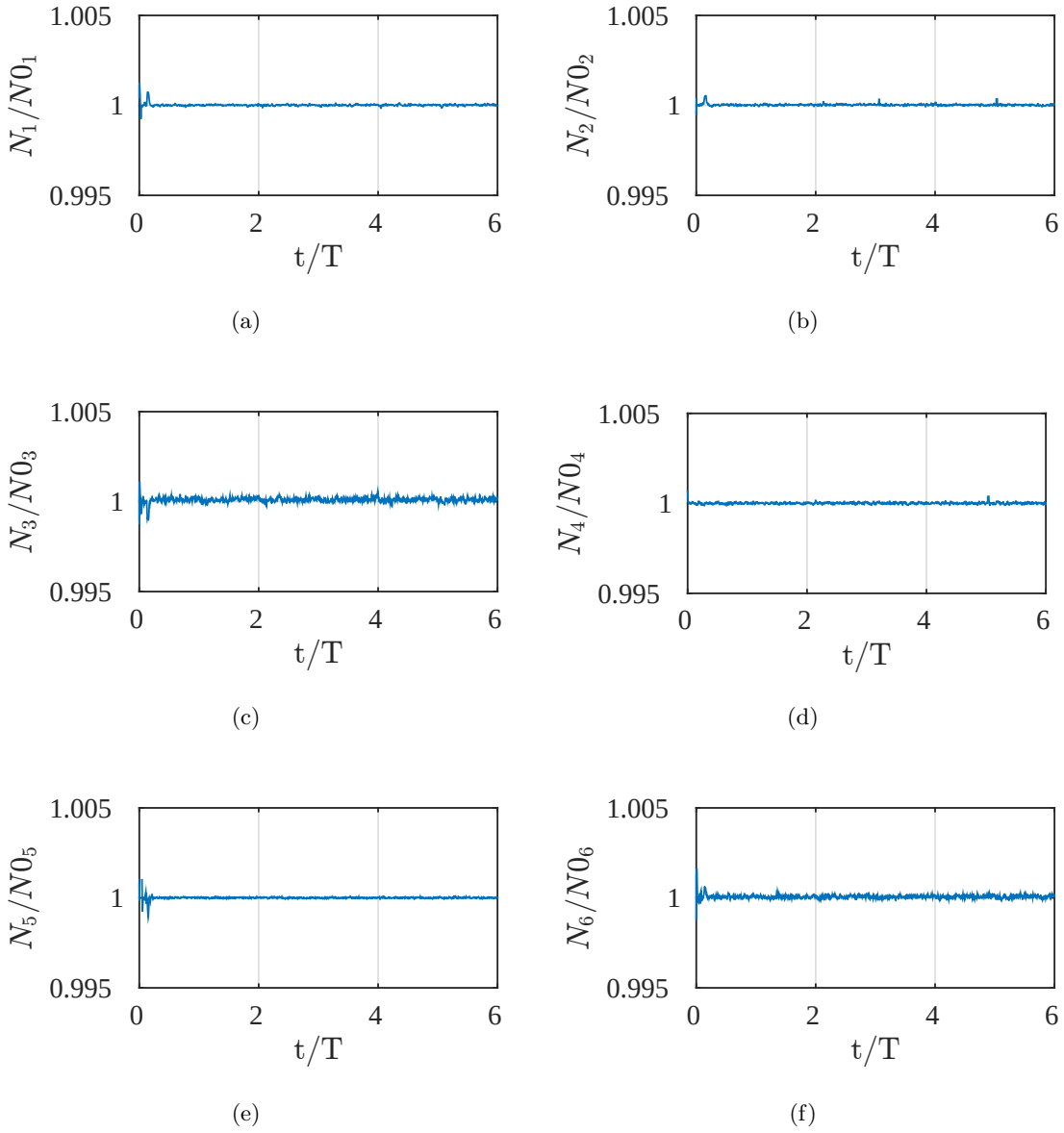


Figure 6.28: *Test case* - Sec. 6.5. Time evolution of the number of particles over six cardiac cycles. The time t is normalized with the cardiac cycle period T while the number of particles in each block with the starting number of particles $N0_n$. a) block 1; b) block 2; c) block 3; d) block 4; e) block 5; f) block 6. Taken from: Monteleone et al. (2018), 975, fig. 22.

Chapter 7

Tracer transport, residence time and mechanical platelet activation models

In this chapter *tracer transport*, *residence time* and *platelets mechanical activation* models are presented for *Single* and *Multi-Domain* approaches both in *serial* and *parallel computing* algorithms. These models are applied to ideal and real aneurysm geometries.

7.1 Background and motivations

Numerical modeling of a virtual contrast agent which passively follows the flow streamlines (the so-called *virtual angiogram*) can be a powerful tool for flow visualization and prediction of intra-aneurysmal regions prone to *thrombus* deposition.

Several authors proposed tracer transport models for different purposes. Calamante et al. (2003) proposed a *patient-specific* model constructed from anatomical and physiologic *magnetic resonance* data in which the arterial blood flow pattern and the transport of a bolus of contrast agent were calculated using finite element analysis. The method was used to characterize the changes in bolus shape due to a stenosis. Kim et al. (2004) developed a numerical dye method for the visualization of the unsteady flow in the ascending aorta by coupling the *convection-diffusion* equation to the Navier-Stokes equations using a finite volume projection-like algorithm. Ford et al. (2005) described a *virtual angiographic* technique for indirectly validating *patient-specific* CFD models against the clinical *in vivo* data. They simulated, through finite element analysis, the time-varying injection of contrast agent into a precomputed *patient-specific CFD* model. Subsequently, they constructed time-series of images by simulating the attenuation of *X-rays* through the computed $3D$ contrast-agent flow dynamics. Vali et al. (2017) analyzed two *basilar* aneurysms considered for surgical treatment using a finite-volume solver. Specifically, they modeled the transport of contrast agent using two approaches: a passive tracer (for predicting post-surgical flow regions prone to thrombus deposition) and the transport of a mixture of blood with an iodine-based contrast agent (for comparing and verifying the numerical results with *in vivo X-ray angiography* data).

In this research study a *tracer transport* model has been implemented in the Lagrangian *SPH* method. The model is able to mimic the process of contrast agent injection in the

radiological procedure. Moreover, the analysis of the tracer transport using a Lagrangian method contains useful information related to the flow and *residence time* (*RT*) patterns in *CAs*. The *RT* parameter gives information on recirculation and stagnation zone and it can be employed thus to predict platelet activation and aggregation as well as thrombus deposition. In the *endovascular* treatment of *CAs*, such as the employing of *flow diverter* (*FD*) devices, it is important to study the *thrombosis* process for verifying if the forming clot can lead to a complete occlusion of the aneurysm sac avoiding, on the other hand, the formation of *thrombi* in the parent vessel. Platelets play a key factor in the process of clot formation. Platelet content of the blood clot promotes the formation of the organized and stable *white thrombi* which facilitate the healing process after aneurysm flow diversion. As discussed in Chap. 1, platelets can be activated by *chemical* or *mechanical stimuli*. Adding the constituents involved in platelet activation, the tracer model could be used to simulate the biochemical activation of platelets in order to analyze the *thrombus* formation inside the aneurysm sac after using *endovascular* devices. Moreover, the model could be used for simulating anti-aggregation species that are used in *endovascular* treatments of *CAs* as post-intervention therapy to prevent *thromboembolism*.

In this research study a different and very important aspect of blood clotting process has been considered: the *mechanical platelet activation*. Experiments have shown that *mechanical platelet activation* is a function of both the magnitude and duration of applied stresses (Brown C.H. et al., 1975; Ramstack et al., 1979; Wurzinger et al., 1985). Further, platelet activation also requires a certain critical level of shear rate to occur. Hellums (1994) plotted an activation locus that showed a power law relationship between threshold stress and exposure time. On the other hand, it has been hypothesised that *shear-induced* platelet activation near the *FD* struts and their subsequent attachment to the forming blood clot inside the *CA* sac, remarkably affects the clot platelet content (Xiang et al., 2014). Several studies involved *shear-induced* platelet activation under pathological conditions such as arterial stenosis (Bluestein et al., 1997; Holme et al., 1997; Tambasco and Steinman, 2003; Shadden and Hendabadi, 2013) and abdominal aortic aneurysms (Hansen et al., 2015). However, to the author's best knowledge, no study was focused on *shear-induced* activation of platelets near the *FD* struts in *CAs* treated with these *endovascular* devices.

Differently from the Lagrangian particle tracking traditionally used to simulate platelet transport (Hansen et al., 2015; Shadden and Arzani, 2015), here a *mechanical platelet activation* model has been developed in a truly Lagrangian framework. Platelets have been modeled as property of the fluid particles and the total level of blood shear-stress has been calculated in each fluid particle following a *stress-exposure time* model (Bluestein et al., 1997; Shadden and Hendabadi, 2013; Hansen et al., 2015). *Tracer transport*, *residence time* and particle level of shear stress have been quantified and examined both in ideal and real aneurysm geometries *FD*-free. Moreover, in order to qualitatively validate the *mechanical platelet activation* model a benchmark test case has been considered (Taylor et al., 2016). The implemented *mechanical platelet activation* model could be applied in *CA* geometries with *FD* in order to evaluate the level of activated platelets as the device porosity changes.

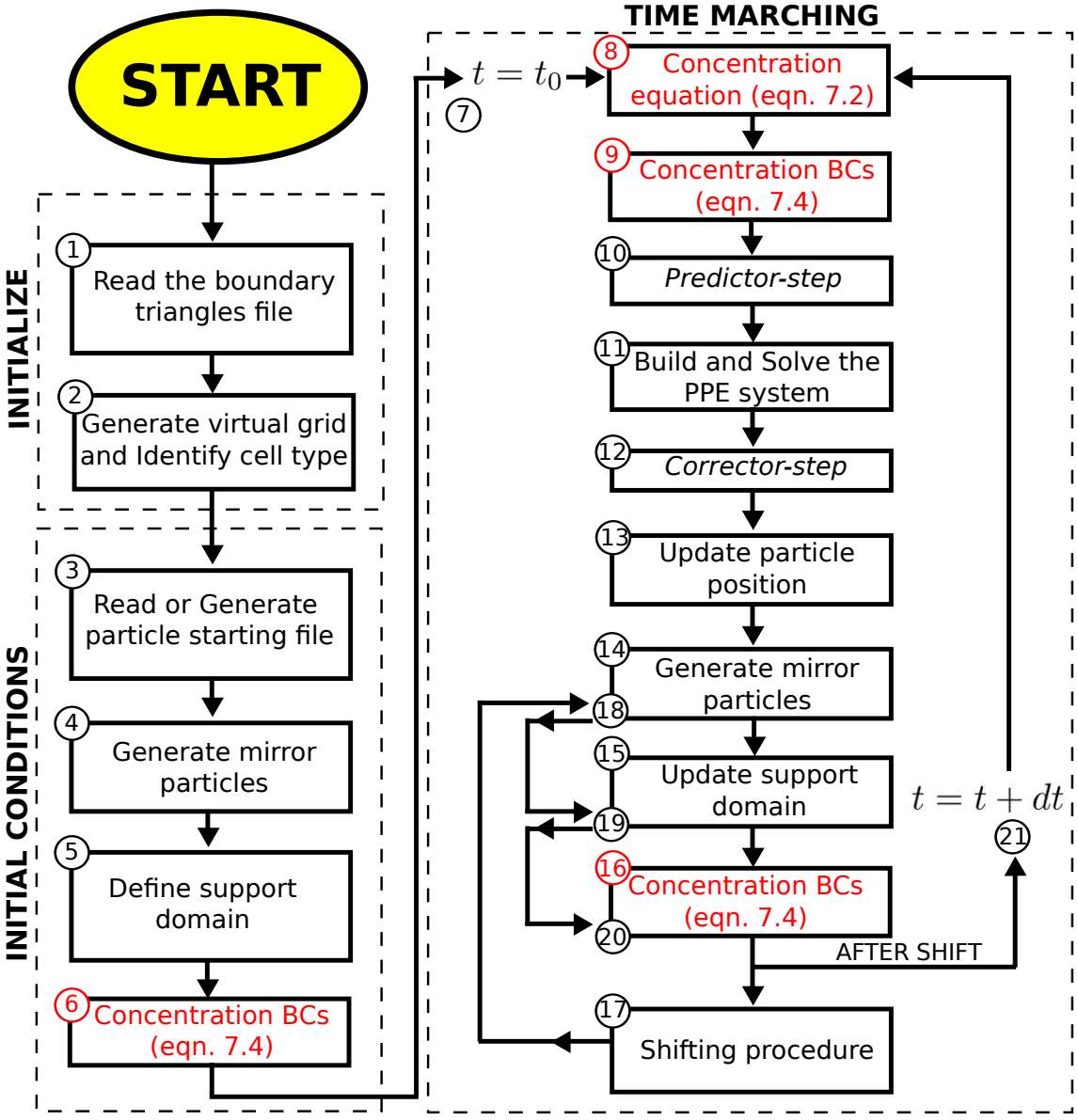


Figure 7.1: Flow chart of the *PANORMUS-SPH* code with the *tracer transport* model. The actions closely related to the tracer transport module are highlighted with the red color.

7.2 Tracer transport model

7.2.1 The model

The tracer (*i.e.,* a biochemical specie or a virtual contrast agent) is numerically injected at the inlets. Different species, whose total number is indicated as $N_{species}$, can be simulated simultaneously.

The transport of each specie through the flow domain has been modeled by solving the *convection-diffusion equation* that can be written as

$$\frac{\Delta C}{\Delta t} - \alpha \nabla^2 C - S = 0 \tag{7.1}$$

where C is the time dependent concentration of the specie, α and S are the diffusivity and the source term, respectively and $\frac{\Delta C}{\Delta t}$ is the total derivative operator which includes, in the Lagrangian approach, the convective terms. Eqn. 7.1 is solved for each specie substituting its own α coefficient and source term.

At each particle i a variable indicating the specie concentration at point $\mathbf{x}_i(t)$ has been assigned. Specifically, this new property is indicated as **Cnc** that is a vector which contains for each position, from 1 to $N_{species}$, the concentration of the corresponding specie. Therefore, for the particle i and the specie s, eqn. 7.1 can be written as

$$Cnc_{i,s}^{k+1} = Cnc_{i,s}^{k} + \alpha_s \frac{3D_i^k - D_i^{k-1}}{2} \Delta t + S_s \, \Delta t \tag{7.2}$$

where $Cnc_{i,s}$ is the component s (corresponding to the specie s) of the vector **Cnc** assigned to the particle i and α_s and S_s are the diffusive coefficient and the source term, respectively, of the specie. A second-order discretization scheme is adopted in eqn. 7.2 for the diffusive term D_i that can be expressed, using eqn. 2.10, as

$$D_i = \sum_{j=1}^{N_i} 2 \frac{m_j}{\rho_j} \frac{(\mathbf{x}_i^k - \mathbf{x}_j^k) \cdot \nabla W_{ij}}{d_{ij}^2} (Cnc_{i,s} - Cnc_{j,s}) \tag{7.3}$$

At solid walls null normal derivatives can usually be imposed

$$\frac{\partial Cnc_{i,s}}{\partial n} = 0 \tag{7.4}$$

The condition of eqn. 7.4 has been assigned by imposing to the *mirror* particle the concentration value of its generating particle ($\mathbf{Cnc}_m = \mathbf{Cnc}_g$ where m and g are the *mirror* and its generating particle, respectively). This condition has been applied also at the outlets where its effect is to prescribe vanishing diffusive fluxes.

The Fig. 7.1 shows the flow chart for the *SPH* code with the tracer module. The boundary conditions for the tracer concentration (eqn. 7.4) are set after the *mirror* generation (*ACTIONS* 6, 9, 16 and 20), whilst the concentration equation (eqn. 7.2) is solved before the *predictor-step*. After the *ACTION* 21 the procedure is restarted from *ACTION* 8.

7.2.2 Tracer transport model in *parallel computing*

The tracer transport model has been included in the *parallel computing* algorithm. Specifically, the concentration of the particles inside the cells of type 5 and 6 (see the definition

in Sec. 5.2.2) must be shared to the neighboring processors (as explained in Sec. 5.2.3 for the sharing of the hydrodynamic variables).

Moreover, the concentration values of the particles going outside the processor domain through *parallel interfaces* must be sent to neighboring processors. On the other hand, the concentration values of the new particles coming from the neighboring processors must be received (as discussed in Sec. 5.2.4).

7.2.3 Tracer transport model in the *MD* approach

Concentration value of the *IP* particles

In the *Multi-Domain* approach the concentration values of the IP particles are obtained by solving a system like 4.7 (discussed for the velocities in Sec. 4.2.3) for each specie:

$$\begin{aligned} Cnc_P^A - \sum_{j=1}^{N_R^{IP}} C'_{pr}\, Cnc_j^B &= RHS_P \quad P = 1, \cdots N_{IP}^A \\ Cnc_Q^B - \sum_{j=1}^{N_S^{IP}} C'_{qs}\, Cnc_j^A &= RHS_Q \quad Q = 1, \cdots N_{IP}^B \end{aligned} \tag{7.5}$$

where Cnc is the specie concentration ($Cnc = Cnc_s$), P and Q are IP particles generated starting from the blocks A and B, respectively, R and S are their closest *effective* particles in blocks B and A, respectively (where the IP are contained), the coefficients C'_{pr} and C'_{qs} are expressed through eqns. 4.6 and N_{IP}^A and N_{IP}^B are the numbers of IP particles contained in the blocks A and B, respectively.

The *right-hand-side* terms RHS_P and RHS_Q are

$$\begin{aligned} RHS_P &= Cnc_R^B + \sum_{j=1}^{N_R^e} C'_{pr}(Cnc_j^B - Cnc_R^B) - \sum_{j=1}^{N_R^{IP}} C'_{pr}\, Cnc_R^B \\ RHS_Q &= Cnc_S^A + \sum_{j=1}^{N_S^e} C'_{qs}(Cnc_j^A - Cnc_S^A) - \sum_{j=1}^{N_S^{IP}} C'_{qs}\, Cnc_S^A \end{aligned}$$

Concentration value of the new particles generated through *block interfaces*

As explained for the velocity assignment, the new particles coming from *block interface* triangles take the concentration value of the closest *effective* particle belonging to the neighboring block. After solving the system (the velocities and concentrations of the IP particles are known now) the concentration of each new particle i is updated using a linear extrapolation among i and the two *interface* particles generated by i.

The ALGORITHM 5.3 of Sec. 5.3.4 can be rewritten as follows:

ALGORITHM 7.1- *Equation velocity and concentration system in parallel MD approach*

1. $n_e = 0$: number of current equation
2. *do* $id = 0, N_{procs} - 1$
3. $n_i = 0$: number of the current IP^* received by id (named P)
4. *do* $ib = 1, N_{Blocks}$
5. *do* $n = 1, N^{*ib}_{IP,id}$
6. $n_e = n_e + 1$
7. $diag_{n_e} = 1$ (diagonal term)
8. $n_i = n + N^{*(ib-1)}_{IP,id}$
9. Take the coordinates of P from the matrix $\mathbf{R}^x$:
   column id and rows from $[3\,(n_i - 1) + 1]$ to $[3\,(n_i - 1) + 3)]$
10. Find the closest particle R in block ib
10.1 $\mathbf{f}_{R(1:3)} = \mathbf{u}_R$
10.2 $\mathbf{f}_{R(4:N_{species}+3)} = \mathbf{Cnc}_R$
11. $\mathbf{RHS}_{n_e} = \mathbf{f}_R$ (velocity and concentration of R are known terms)
12. *do* $j = 1, N_R$ (cycle on the particles in Ω_R)
12.1 if j is *effective* or *mirror* then
12.1.1 $\mathbf{f}_{j(1:3)} = \mathbf{u}_j$
$\mathbf{f}_{R(4:N_{species}+3)} = \mathbf{Cnc}_R$
12.1.2 $\mathbf{RHS}_{n_e} = \mathbf{RHS}_{n_e} + C'_{pr}\,(\mathbf{f}_j - \mathbf{f}_R)$
12.2 if j is IP then
$off\_diag_{(n_e,j)} = C'_{pr}$
$\mathbf{RHS}_{n_e} = \mathbf{RHS}_{n_e} - C'_{pr}\,\mathbf{f}_R$

where $\mathbf{f}$ is a vector containing in the first three positions the velocity vector of the particle R or of its neighboring particle j (points 10.1 and 12.1.1, respectively). From the forth to the $N_{species} + 3$ positions the concentration values of each specie of the particle R or of its neighboring particle j (points 10.2 and 12.1.2, respectively) are registered. For the other definitions see Chap. 5.

The global system made of eqns. 4.7 and 7.5 is solved using the *Pre-BiCGSTAB* method once for each velocity component m and for each concentration specie s (with $m = 1, 2, 3$ and $s = 1, .., N_{species}$), using the same coefficient matrix and updating the *right-hand-side* only. To this aim, the vector solution $\mathbf{x}$ (of length n_3) becomes a matrix $[n_3$ x $3 + N_{species}]$ as well as the known terms vector $\mathbf{b}$ $[n_2$ x $3 + N_{species}]$.

In order to solve the system, the ALGORITHM 5.4 of Sec. 5.3.4 is modified at point 1 including in the first cycle the number of species $N_{species}$.

7.2.4 Flow chart of the *PANORMUS-SPH* code

The Fig. 7.2 shows the flow chart of the implemented *tracer transport* model in the *SPH* code with the *parallel MD* algorithm. In the figure the diagram begins after the *ACTION*

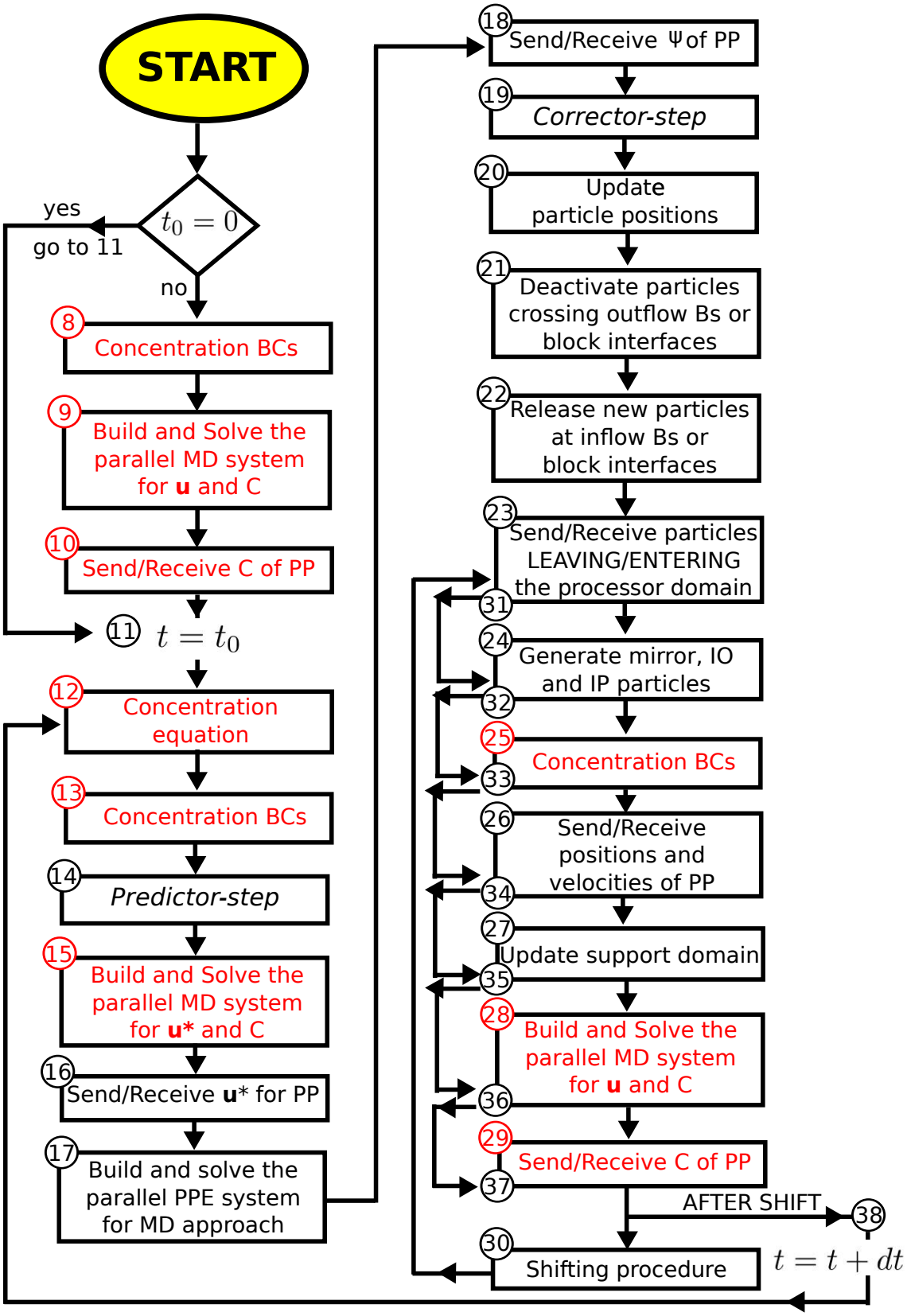


Figure 7.2: Flow chart of the *PANORMUS-SPH* code with the *parallel MD* procedure and the *tracer transport* model. The actions closely related to the *tracer transport* model are highlighted with the red color.

7 of the Fig. 5.29.

If the simulation starts from developed velocities and concentrations ($t_0 \neq 0$), the *ACTIONS* 8, 9 and 10 must be performed in order to obtain the values (velocity and concentration) of the IP particles. The concentration and velocities values of the *effective* particles are read from the starting particle file.

In the *ACTIONS* 8, 13 and 25 the boundary conditions for the concentration (eqn. 7.4) are set after the *mirror* generation by imposing the concentration of each *mirror* particle equal to that of the corresponding generating particle.

In the *ACTIONS* 9, 15 and 28, in order to obtain the velocities and the concentration of the IP particles, the MD system is built (see ALGORITHM 7.1) and is solved through the *Pre-BiCGSTAB* method (see ALGORITHM 5.4).

The current processor sends the concentration values (for all the analyzed species) of the particles (*effective* and *mirror*) inside the cells of type 5 and 6; it simultaneously receives the corresponding values from the neighboring processors (*ACTIONS* 10, 29 and 37).

In *ACTION* 12 the eqn. 7.2 is solved for each *effective* particle of the processor and for each specie.

For the *ACTION* 14 see the descriptions of the *ACTION* 10 of Fig. 5.29 (described in Sec. 5.29), as well as for the *ACTIONS* from 16 to 30 see the corresponding descriptions in Fig. 5.29 (*ACTIONS* from 12 to 24).

In *ACTION* 23 the processor receives the *effective* particles (positions, velocities, *pseudo-pressure*, acceleration and concentration of the analyzed species) which have left the domain of the neighboring processors and now lie in cells belonging to *myid* as explained in Sec. 5.2.4.

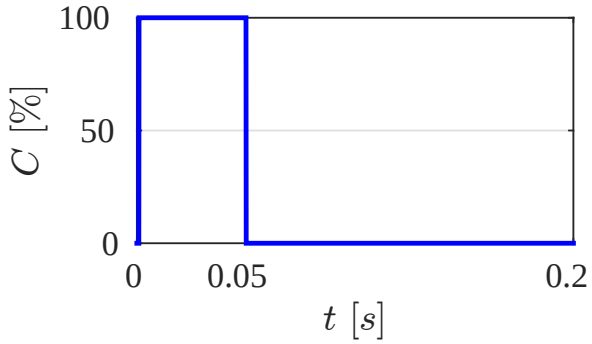


Figure 7.3: *Benchmark test case* - Sec. 7.2.5. Inflow concentration law. C is expressed as percentage compared to the maximum value of the new particles generated at the inlet section.

7.2.5 Benchmark test case: tracer transport in a circular pipe

A cylindrical pipe having diameter $D = 1{\cdot}10^{-3}\ m$ and length $L = 10\,D$ has been considered to analyze the transport of a passive tracer. The fluid properties are discussed in Sec. 6.1.

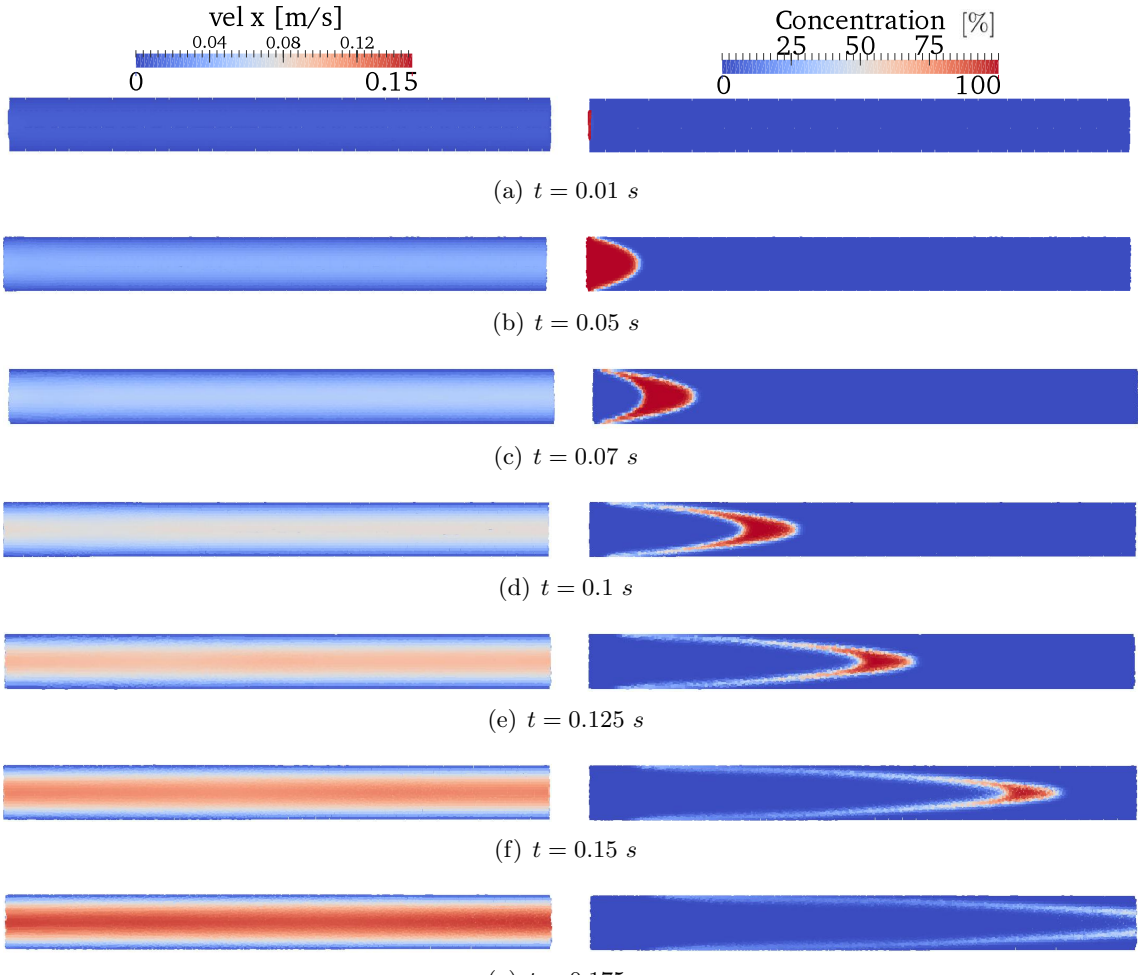
A parabolic velocity profile has been imposed at the inlet (*incoming BCs* described in Chap. 3) with a mean velocity of 0.2 m/s. *Adherence BCs* have been set at the lateral solid walls, whilst zero pressure (*pressure BCs* described in Chap. 3) has been imposed at

the outlet. The initial particle distance has been set to $\Delta x = 5 \cdot 10^{-5}\ m$ in order to have 20 particles along the pipe diameter; the resulting total number of particles is $N_e = 63\,200$. A passive tracer with zero source term and diffusive coefficient $\alpha = 4 \cdot 10^{-8}\ m^2/s$ has been numerically injected at the inlet with a square wave represented in Fig. 7.3.

In order to simulate the filling patterns, the whole domain has been initialized with zero tracer concentration.

The simulation has been performed in *parallel computing* using 10 processors.

The Fig. 7.4 shows the velocity (on the left) and the tracer concentration (on the right) along a longitudinal section of the pipe at different time instants.



(a) $t = 0.01\ s$

(b) $t = 0.05\ s$

(c) $t = 0.07\ s$

(d) $t = 0.1\ s$

(e) $t = 0.125\ s$

(f) $t = 0.15\ s$

(g) $t = 0.175\ s$

Figure 7.4: *Benchmark test case* - Sec. 7.2.5. Axial velocity (left) and tracer concentration (right) at different times.

7.2.6 Benchmark test case: tracer transport in a circular pipe with the *MD* approach

The transport of two tracers along a cylindrical pipe has been studied using the MD approach. The domain has been subdivided into 2 blocks (Fig. 7.5) having $\Delta x_1 = 4.5 \cdot 10^{-5}\ m$ and $\Delta x_2 = 6 \cdot 10^{-5}\ m$. A parabolic velocity profile with mean velocity of

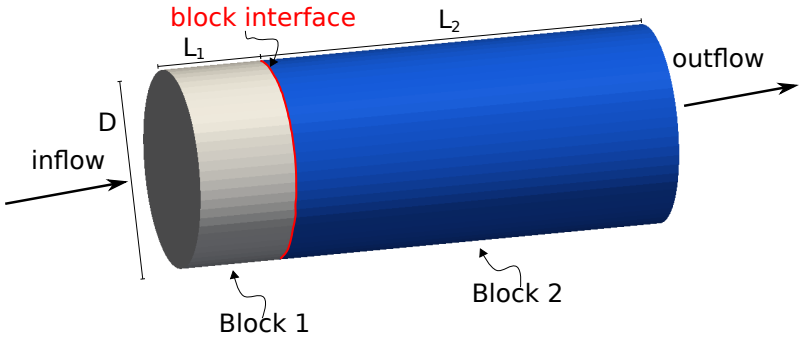


Figure 7.5: *Benchmark test case* - Sec. 7.2.6. Domain decomposition into 2 blocks. $D = 1 \cdot 10^{-3}\ m$; $L_1 = D/2$, $L_2 = 2\,D$.

0.2 m/s has been set at the inlet section of block 1, whilst zero pressure has been imposed at the outlet section of block 2. At the *block interface* (represented in red in the figure) which corresponds to the outflow and inflow section of block 1 and 2, respectively, the *Multi-Domain* procedure has been applied in order to obtain velocity, *pseudo-pressure* and tracer concentrations of the $interface$ particles.

The tracers have different diffusion coefficient: $\alpha_1 = 4 \cdot 10^{-8}\ m^2/s$ (*tracer* 1) and $\alpha_2 = 4\ 10^{-7}\ m^2/s$ (*tracer* 2). They are numerically injected at the inlet section of block 1 following, for both the tracers, the same concentration law represented in Fig. 7.3.

The concentration of the two tracers at different times is shown in Fig. 7.6 along a longitudinal section of the pipe. As it is seen in the figure the tracer concentration follows the paraboloid shape of the velocity. The different diffusion of the two tracers is more visible in Fig.7.6.e, where the maximum concentration is about 60 and 100 for the tracer 2 (represented on the left) and tracer 1, respectively. Moreover, in the figure it is clearly seen the perfect matching of the concentration solution obtained at the *block interface*.

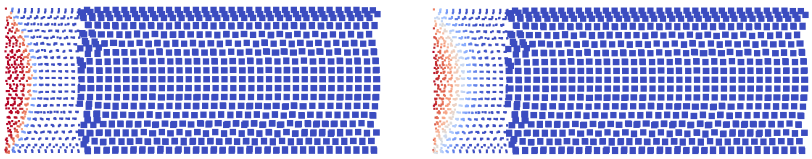
7.3 Residence time

The *residence time* (RT in the following) is an important parameter to identify recirculation and stagnation zones and thus to predict platelet activation and thrombus deposition.

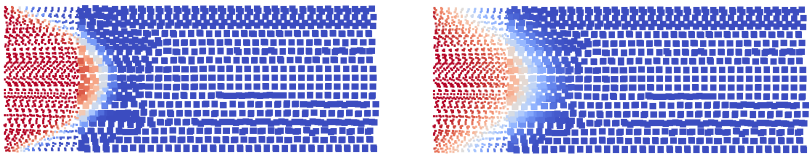
As discussed by Rayz et al. (2010) and Menichini and Xu (2016), *residence time* can be modeled as a tracer passively transported with the flow solving a *convection-diffusion* equation. In this study RT has been treated as an additional specie taking place in the vector **Cnc**. For the RT estimation, in eqn. 7.2 α_s represents the *self-diffusivity* of blood (set to $10^{-15}\ m^2/s$) and S_s is the source term equal to 1 considering an unit increase in RT for each unit increase in time.

As for the tracer transport, *Neumann BCs* have been assigned at the solid walls as well as at the outlet sections, setting the RT of the mirror particles equal to that of the generating particles. Differently, at the inlet section the RT of the new particles (and of their *mirror*) is set to zero.

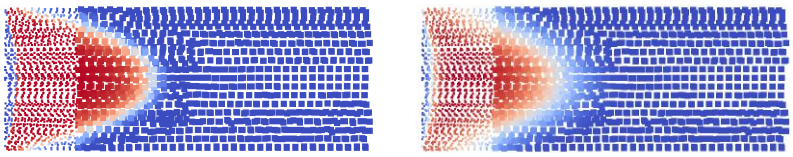
The RT estimation procedure in *Multi-Domain* approach and *parallel computing* is identical to that explained in Sec. 7.2.3 for the *tracer transport* model.



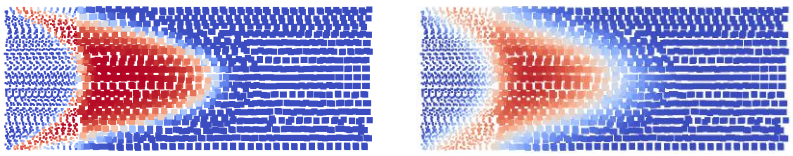
(a) $t = 0.002\ s$



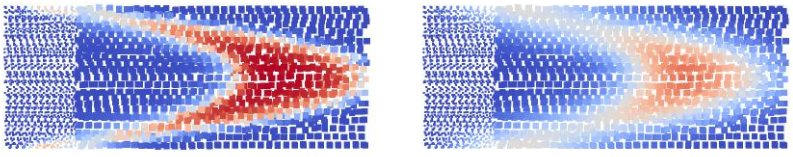
(b) $t = 0.004\ s$



(c) $t = 0.005\ s$



(d) $t = 0.006\ s$



(e) $t = 0.008\ s$

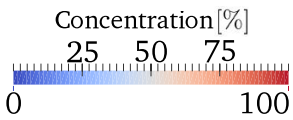


Figure 7.6: *Benchmark test case* - Sec. 7.2.6. Concentration of the two tracers at different times along the pipe longitudinal section: *Tracer* 1 with $\alpha_1 = 4 \cdot 10^{-8}\ m^2/s$ (left) and *Tracer* 2 with $\alpha_2 = 4 \cdot 10^{-7}\ m^2/s$ (right). Small and big squares: *effective* particles of block 1 and 2, respectively.

7.4 Mechanical platelet activation

7.4.1 The activation potential model

The platelet *activation potential* (AP) is the parameter governing *mechanical platelet activation* under the action of stress higher than a threshold value that Hellums (1994) suggests to be equal to $3.5\,Ns/m^2$. In this research study platelets have been considered passive particles moving accordingly to the blood velocity field. Exploiting the Lagrangian nature of *SPH* and assuming the same discretization for the fluid and platelet particles, the *activation potential* AP has been treated as a new property of each fluid particle. Therefore, no any new set of particles has been added in the computation to model the transport of platelets.

AP has been calculated following the *stress-exposure time* model (Bluestein et al., 1997; Shadden and Hendabadi, 2013; Hansen et al., 2015). Specifically, for each fluid particle i, AP can be calculated considering the magnitude of the total rate of deformation acting on the i particle through its trajectory

$$AP_i = \sum_{n=1}^{N_t} \tau_i(\mathbf{x}_i, t)\Delta t \tag{7.6}$$

where N_t is the total number of time steps the particle i spends inside the domain and τ_i is the scalar stress at the position $\mathbf{x}_i$ of the particle i and at time t.

The scalar stress τ_i can be calculated as follows

$$\tau_i\left(\mathbf{x}_i, t\right) = \sqrt{2}\mu\,||\mathbf{D}|| \tag{7.7}$$

where the *Newtonian approximation* ($\mathbf{s}(\mathbf{x}, t) = 2\mu\,\mathbf{D}(\mathbf{x}, t)$ with $\mathbf{s}$ the viscous stress tensor) has been assumed, μ is the dynamic viscosity, $\mathbf{D}$ is the rate of deformation tensor (eqn. 7.8) and $||\mathbf{D}||$ is its Frobenius norm (eqn. 7.9) that allows to obtain a scalar stress measure direction-independent. The scalar shear stress gives a measure of the total shear stress acting on a fluid particle.

$$\mathbf{D}\left(\mathbf{x}_i, t\right) = \frac{1}{2}\left(\nabla\mathbf{u}_i(\mathbf{x}_i, t) + \nabla\mathbf{u}_i(\mathbf{x}_i, t)^T\right) \tag{7.8}$$

$$||\mathbf{D}|| = \sqrt{tr(\mathbf{D}\,\mathbf{D}^T)} \tag{7.9}$$

7.4.2 New particles from *block interfaces*

Since the AP is a time-dependent parameter related to the particle trajectory, a specific procedure has been implemented to obtain the AP value of the new particles generated through *block interfaces* in the *Multi-Domain* approach.

When a new particle is created in the block *lb*, its AP value can be set equal to that of the closest *effective* particle leaving the neighboring block *ib*. Therefore, at each time step the position and AP value of the particles leaving the block *ib* through *block interface* and "virtually" entering the domain of the block *lb* are recorded for 10 time step. When a new particle is created in the block *lb* through the *block interface* separating the block *ib*, the particles leaving *ib* are scanned starting from the current time instant and progressing toward the previously ones (up to a maximum of 10 step) until the closest particle (whose distance from the new particle must be less than $\Delta x/2$) is found.

7.4.3 New particles from *parallel interfaces*

In *parallel computing* the processor generating the new particle may not know the *effective* particle of block *ib*. To this aim, the processor *myid*, while identifying the *effective* particles leaving the block *ib* and "virtually" entering in the block *ib*, checks the processor *id* owning the cell of these particles. The particles leaving the processor domain through *block interfaces* are recorded for processor *id* in which the particles are "virtually" contained and for block. The current processor *myid* sends coordinates and AP value of these particles to the corresponding processor *id*, following a procedure similar to that explained in Chap. 5 to send the coordinates of the IP particles. The received values are sorted for blocks regardless of the sending processor. When a new particle is created in the block *lb*, the processor searches the closest particle using the receiving coordinates related to the block *ib*.

In *parallel computing* the AP values of the *effective* particles going outside the processor domain through *parallel interfaces* must be sent to the neighboring processors. On the other hand, the AP values of the new particles coming from the neighboring processors must be received (as explained in Sec. 5.2.4).

7.4.4 Flow chart of the *PANORMUS-SPH* code

The Fig. 7.7 shows the flow chart of the *SPH* code in the *parallel MD* approach with the implemented *mechanical platelet activation* model whose actions are highlighted in red. The *ACTIONS* from 1 to 16 are the same explained in Sec. 5.3.6 (see Fig. 5.29). The deactivated particles which have crossed a *block interface* are recorded in *ACTION* 17 and sent in *ACTION* 19, whilst the AP value of the new particles generated through *block interfaces* (during *ACTION* 18) is set in *ACTION* 20 following the procedure explained in Sec. 7.4.2. The AP value of each new particle generated at inflow boundaries (*ACTION* 18) is set to zero since this parameter is calculated starting from the time instant in which the particle enters into the domain. In *ACTION* 21 the processor receives the *effective* particles (positions, velocities, *pseudo-pressure*, acceleration and *activation potential*) which have left the domain of the neighboring processors and now lie in cells belongs to *myid*, as explained in Sec. 5.2.4. In *ACTION* 32 the AP value at time t of each *effective* particle i belonging to the current processor is obtained by summing up the value $\tau_i(\mathbf{x}_i, t)\ \Delta t$ to the AP value calculated starting from the previous time step

$$AP_i(t) = \tau_i(\mathbf{x}_i, t)\Delta t + \sum_{n=1}^{nt-1} \tau_i(\mathbf{x}_i, n)\Delta t = \tau_i(\mathbf{x}_i, t)\Delta t + AP(t-1)$$

where nt is the number of iteration the particle i spends in the domain until the current time t.

7.4.5 Benchmark test case: asymmetric sudden expansion

The study of Taylor et al. (2016) has been considered to qualitatively validate the *activation potential* model. In the selected study, a computational model for macroscopic predictions of cardiovascular device-induced thrombosis was developed. The authors compared their numerical results with the experimental results of Taylor et al. (2014) (a data set of time-dependent thrombus size collected in vitro with magnetic resonance imaging).

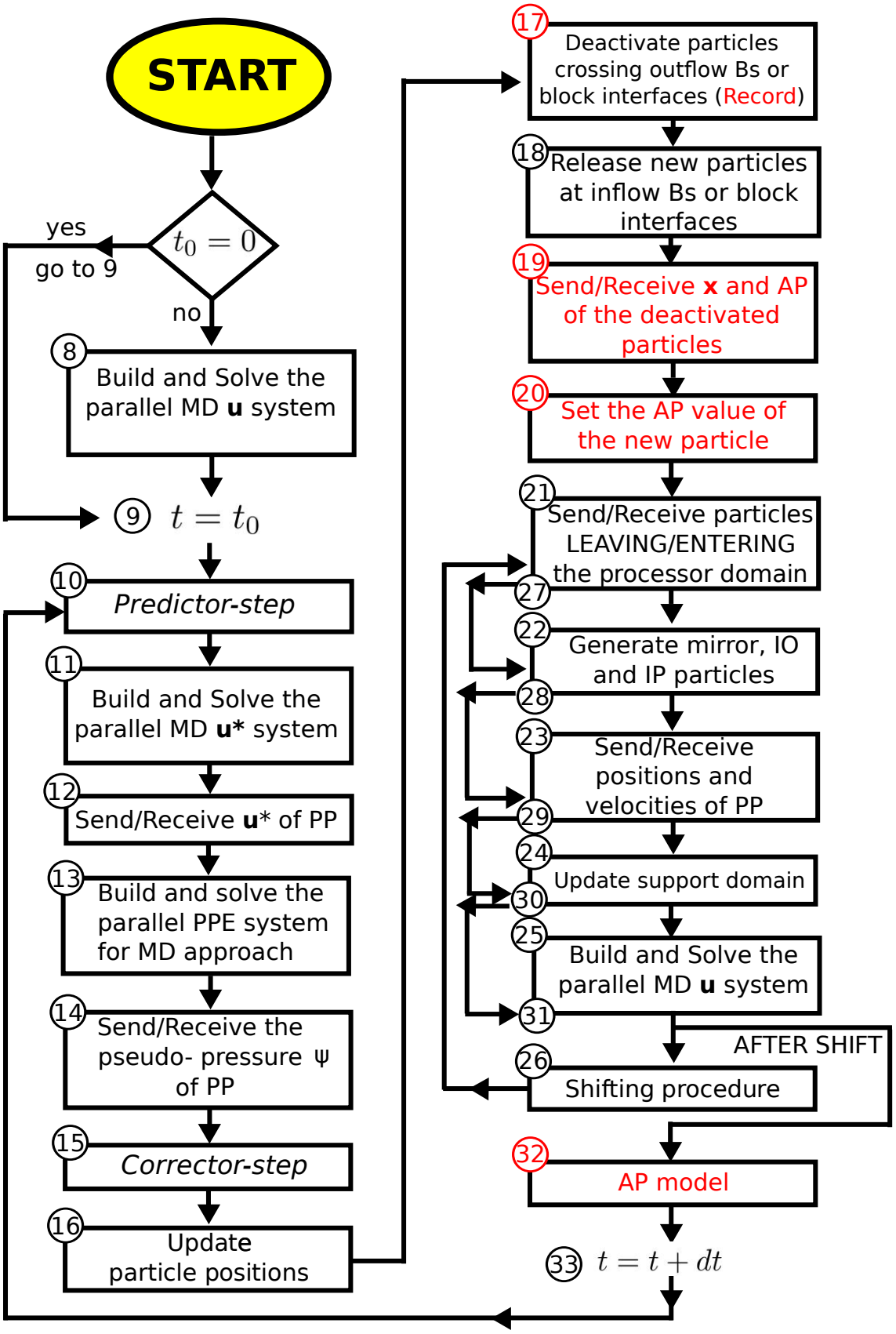
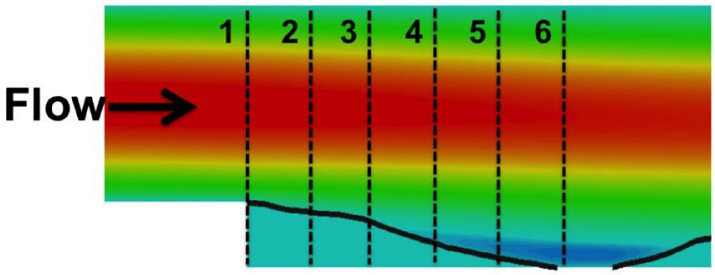


Figure 7.7: Flow chart of the *PANORMUS-SPH* code with the *parallel MD* procedure and the *mechanical platelet activation* model. The actions closely related to the *mechanical platelet activation* model are highlighted with the red color.

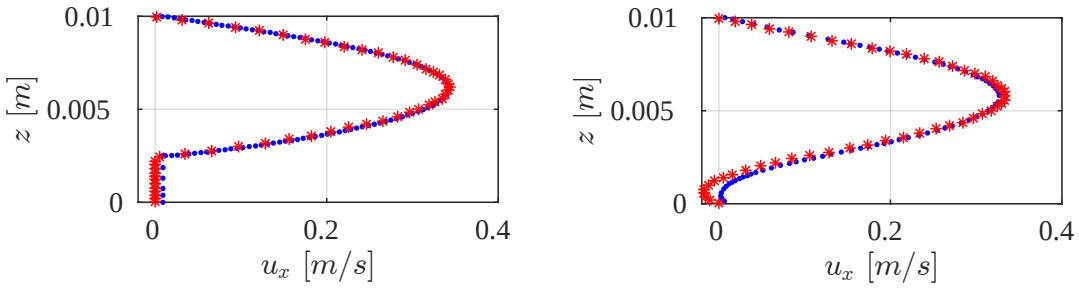
Figure 7.8: *Benchmark test case* - Sec. 7.4.5. Channel geometry. $H = 10\ mm$, $H_s = 7.5\ mm$, $L_1 = 10\ mm$, $L_2 = 50\ mm$.

It should be highlighted that there are some differences between the model implemented by Taylor et al. (2016) and the platelet model developed in this research study. Specifically, the research group of Taylor et al. (2016) modeled the *platelets activation* considering both mechanical and chemical stimuli. They quantified the *mechanical* activation with a simplified form of a Lagrangian power law model of Soares et al. (2013), and the platelet activation using a function of adenosine diphosphate concentration. Moreover, the authors considered the thrombus deposition and growth in regions of low wall shear stress assuming fluid mechanics as the dominant predictor of thrombus formation. In the *PANORMUS-SPH* code the thrombus formation and deposition is not modeled and only the *mechanical platelet activation* has been included.

Although these differences some qualitative comparisons can be made related to the



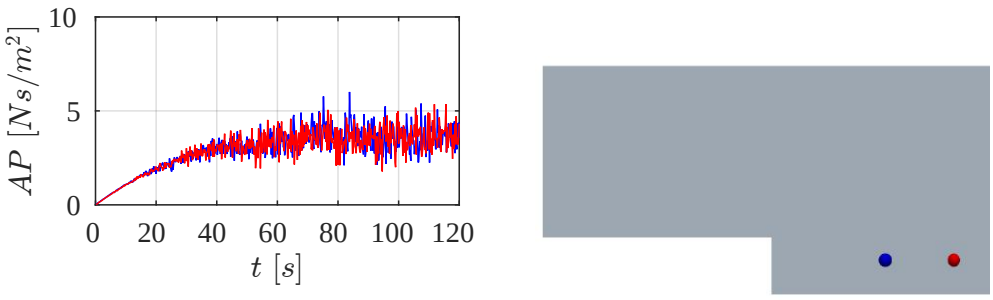
(a) Profile scheme



(b) cross-section 1

(c) cross-section 6

Figure 7.9: *Benchmark test case* - Sec. 7.4.5. Velocity profiles. Blue diamonds: Taylor et al. (2016) results; red stars: *PANORMUS-SPH* results.



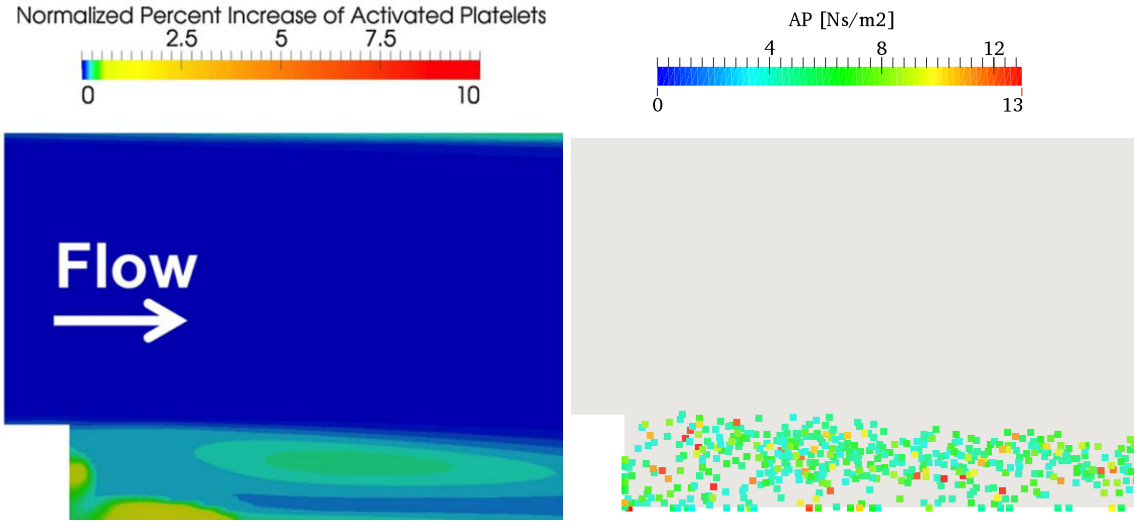
(a) (b) Points scheme

Figure 7.10: *Benchmark test case* - Sec. 7.4.5. Time evolution of the AP at points $A(0.015, 0.0015)$ and $B(0.018, 0.0015)$ (blue and red lines, respectively).

velocity field and the location and level of the platelet *activation potential*.

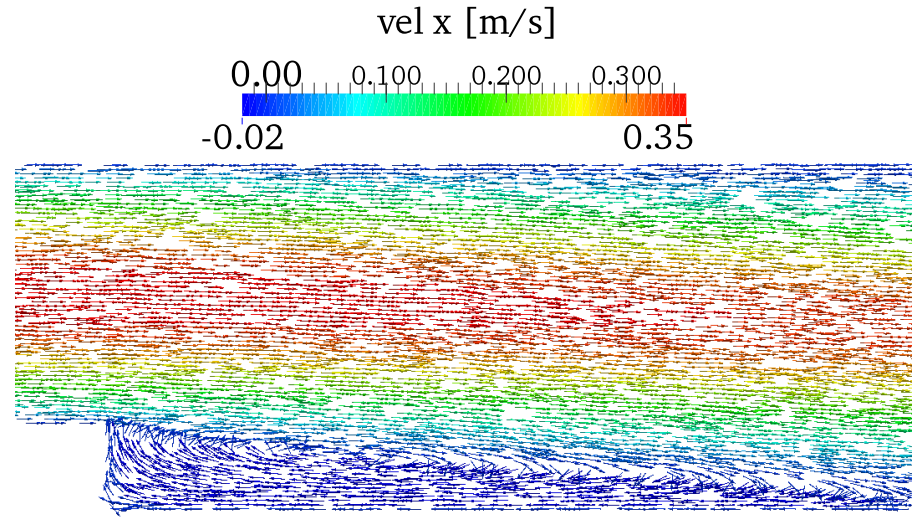
The considered computational domain is a $2D$ asymmetric sudden expansion (see Fig. 7.8). This geometry, mimicking the flow separation, is widely used in the literature to study the thrombosis formation. In this configuration the thrombus growth downstream of the expansion is enhanced by the capture of activated platelets by the recirculating flow.

At the inlet section a parabolic velocity profile has been imposed with a mean velocity of 0.229 m/s, whilst *Dirichlet BCs* for the pressure have been set at the outlet section. *Periodic BCs* have been prescribed at the walls parallel to the flux direction while *adherence BCs* have been used at the other domain walls. The $2D$ results have been obtained

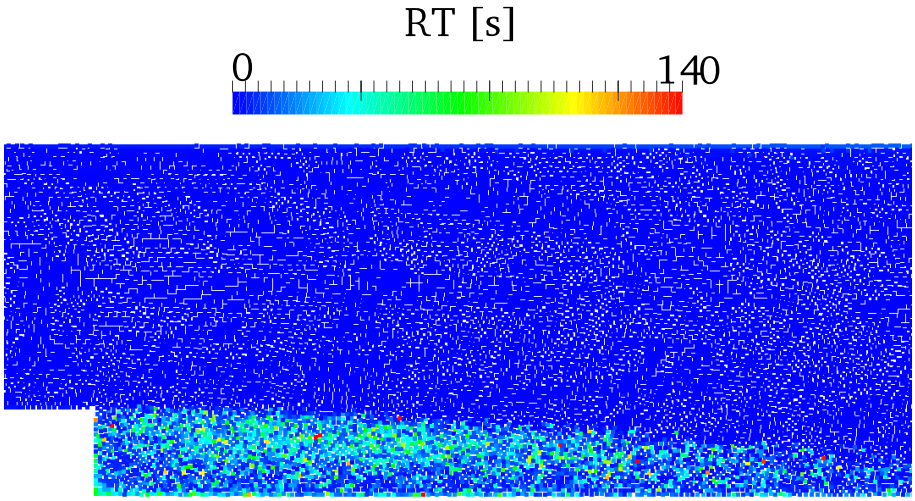


(a) (b)

Figure 7.11: *Benchmark test case* - Sec. 7.4.5. Qualitative comparison of the platelets activation. Enlargement near the expansion. a) Taylor et al. (2016): normalized percent increase in the concentration of activated platelets above the background level ($t = 10\ min$); b) *PANORMUS-SPH*: AP of the particles whose value is higher than 3.5 Ns/m^2 ($t = 2\ min$).



(a) Velocity vectors



(b) Residence time

Figure 7.12: *Benchmark test case* - Sec. 7.4.5. Enlargement near the expansion. Velocity vectors and *RT*.

considering plane flows with one single layer of particles in the direction normal to the plane (as explained in Sec. 2.5.1). The *smoothing length* has been set to $h = 2.5 \cdot 10^{-4}\ m$ with 38 000 total number of *effective* particles.

The Fig. 7.9 shows the comparison of the velocity profiles at two different cross-sections (sections 1 and 6 indicated in Fig. 7.9.a). The differences are located at the step where the Taylor et al. (2016) profiles do not have negative streamwise velocities (which are present in the *SPH* results) due to the thrombus deposition.

The Fig. 7.10 shows the *activation potential* evolution in time at two different points (A and B in the figure). As it is seen in the figure, the AP reaches the steady-state after about 80 seconds with a mean value of about 3.7 Ns/m^2.

The Fig. 7.11 shows a qualitative comparison of the platelet activation. The results of Taylor et al. (2016) shown in Fig. 7.11.a represent the normalized percent increase in the concentration of the activated platelets above the background level after 10 minutes

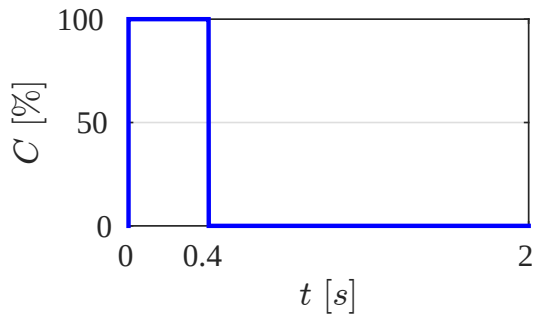


Figure 7.13: *Test case* - Sec. 7.5.1. Inflow concentration law. The concentration C is expressed as percentage compared to the maximum value of the new particles generated at the inlet.

of simulation (since it is necessary more time to simulate the thrombus deposition). The *SPH* results (Fig. 7.11.b) show the particles whose AP level is higher than the threshold value suggested by Hellums (1994) at $t = 120\ s$ (when the steady-state of the *activation potential* has been reached). The location of the *SPH* particle having $AP > 3.5\ Ns/m^2$ is in agreement with the location of the activated platelets obtained by Taylor et al. (2016).

The recirculation region at the step is shown in Fig. 7.12.a where the RT parameter achieves the highest values (Fig. 7.12.b).

7.5 Test cases

7.5.1 Ideal aneurysm

The ideal aneurysm described in Sec. 6.2 has been considered to show the performance of the *tracer transport*, the *residence time* and the *activation potential* models.

A passive tracer with diffusion coefficient $\alpha = 1 \cdot 10^{-8}\ m^2/s$ and null source term has been injected at the inlet following the tracer law represented in Fig. 7.13.

The Fig. 7.14 shows the concentration of the tracer at different time instants: during the tracer injection (Figs. 7.14.a, 7.14.b, 7.14.c, 7.14.d and 7.14.e) and after the injection when $t > 0.4\ s$ (Figs. 7.14.f, 7.14.g, 7.14.h and 7.14.i). In the figure, a threshold value of 10 % has been selected to define the interface of the *tracer-laden* and *tracer-free* flow. Therefore, only the particles having concentration higher than the selected threshold value have been represented for each time instant.

At the beginning the tracer takes the paraboloid shape of the velocity (Fig. 7.14.a), then it curves (Fig. 7.14.b) due to the presence of the sphere (the aneurysm sac) and impacts in the right intersection between vessel and aneurysm (which is the impingement area highlighted in Fig. 6.3.b) as shown in Fig. 7.14.c, thus it enters inside the aneurysm (Figs. 7.14.d and 7.14.e). For $t > 0.4\ s$ the new particles are released at the inlet with zero concentration following the tracer law of Fig. 7.13. These particles moving in the flow direction acquire a non-null concentration related to diffusive term tanks to the high gradient of concentration. After stopping the injection, the tracer takes again the paraboloid shape represented in Fig. 7.14.f (having opposite gradient to that shown in Fig. 7.14.a) since new particles which have been generated with zero concentration at the inlet in previous instants. In Fig. 7.14.g and, even more, in Fig. 7.14.h all the particles inside the

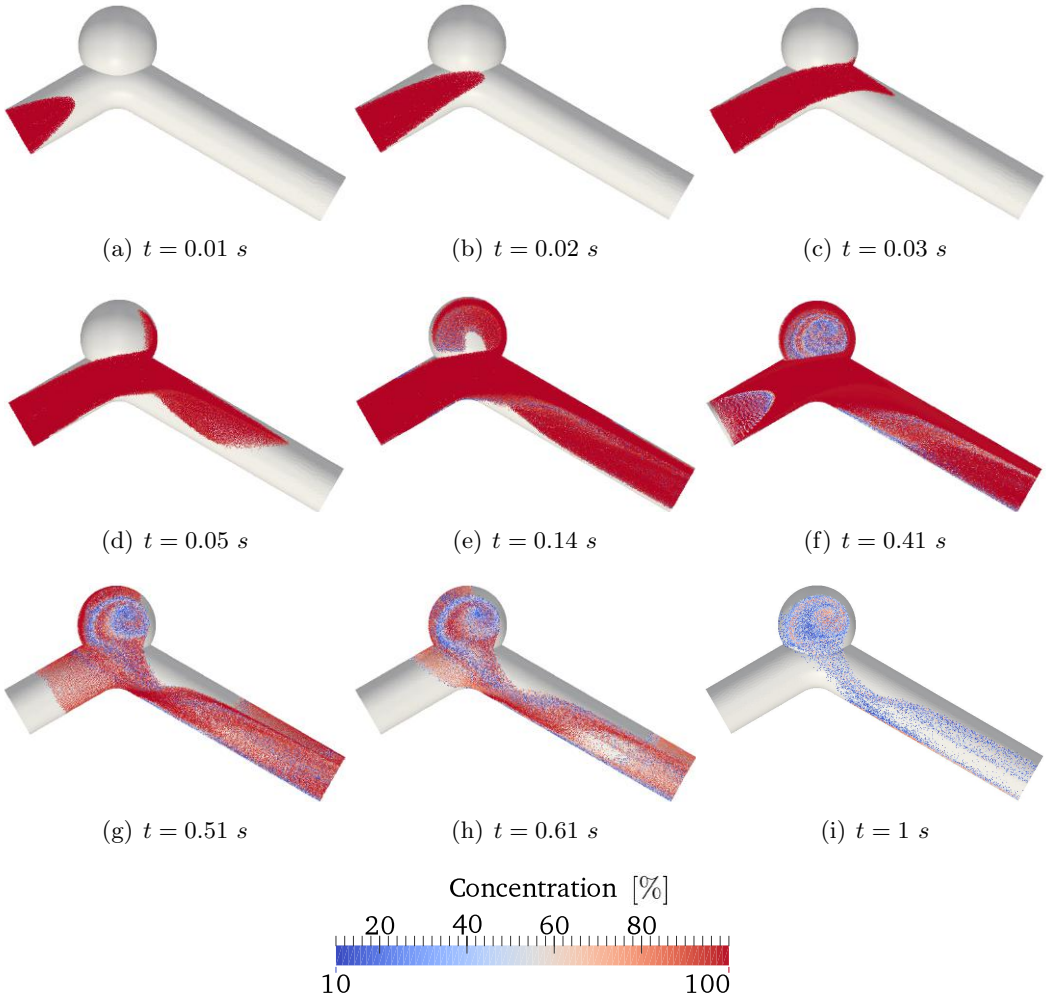
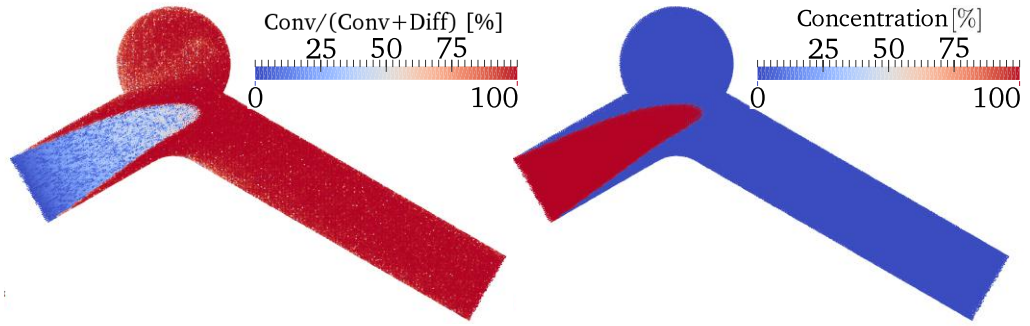


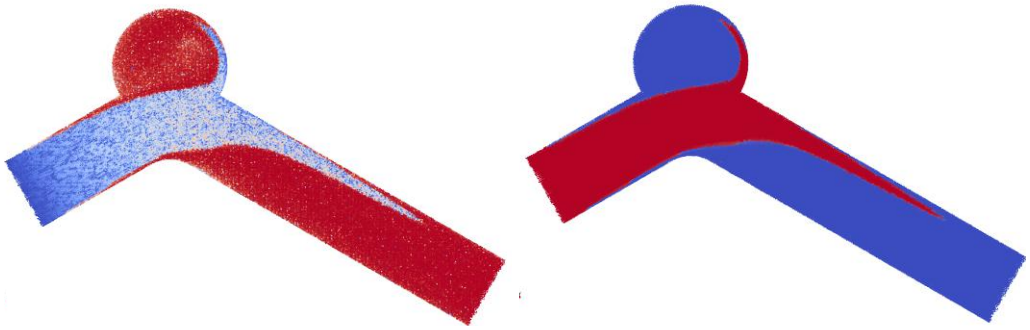
Figure 7.14: *Test case* - Sec. 7.5.1. Tracer concentration at different time instants. Gray area: domain contour.

input and output vessels have null concentration value except for those close to the walls that maintain the highest values due to the low velocities. Non-null concentration values are inside the aneurysm where the tracer takes the shape of the vortex occurring inside the aneurysm sac (as it has been shown in Fig. 6.3.c). Here the tracer highest values are close to left side wall of the vessel since the flux impacting on the right side moves away from the right wall. The vortex traps the particles to the sphere center allowing to have non-zero concentration values inside the aneurysm sac for a long time (see Fig. 6.3.i). After $2\,s$, the concentration values become zero throughout the domain.

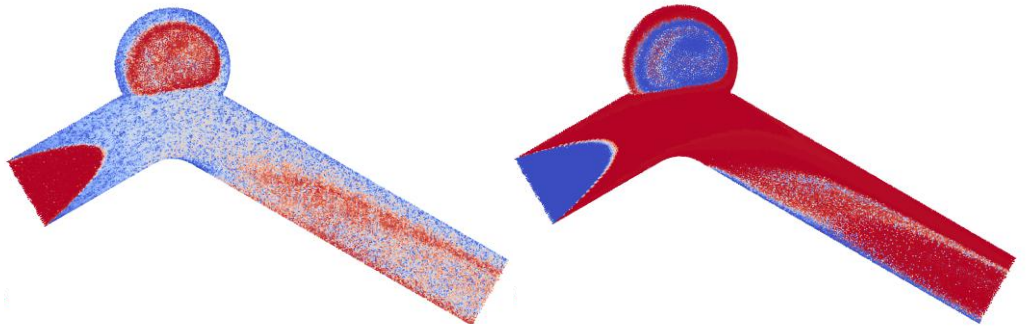
An analysis of the relative weights of convective and diffusion terms has been carried out. To this aim, Fig. 7.15 shows the percentage ratio $Conv/(Conv+Diff)$ (where $Conv$ is the sum of the convective terms and $Diff$ is that of the diffusive ones) on the left and the tracer concentration on the right at different time instants along the longitudinal section of the ideal vessel. It should be noted that in a Lagrangian framework the convective terms are not explicitly calculated, therefore, in order to provide this comparison these terms have been explicitly obtained using the velocity and concentration fields. As it



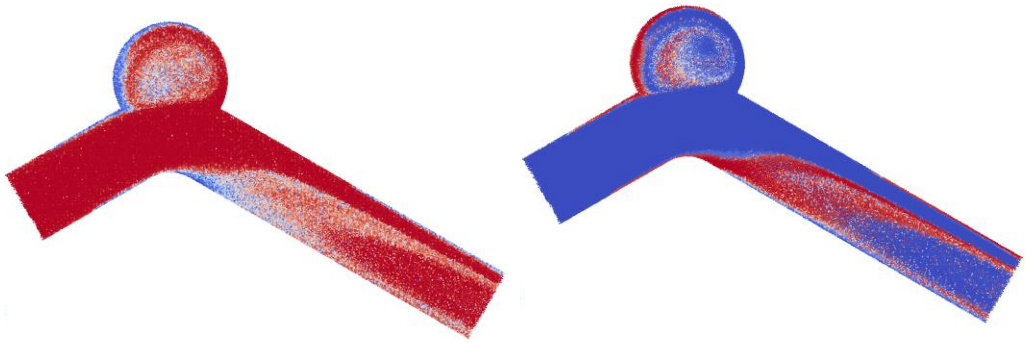
(a) $t = 0.01\ s$



(b) $t = 0.05\ s$

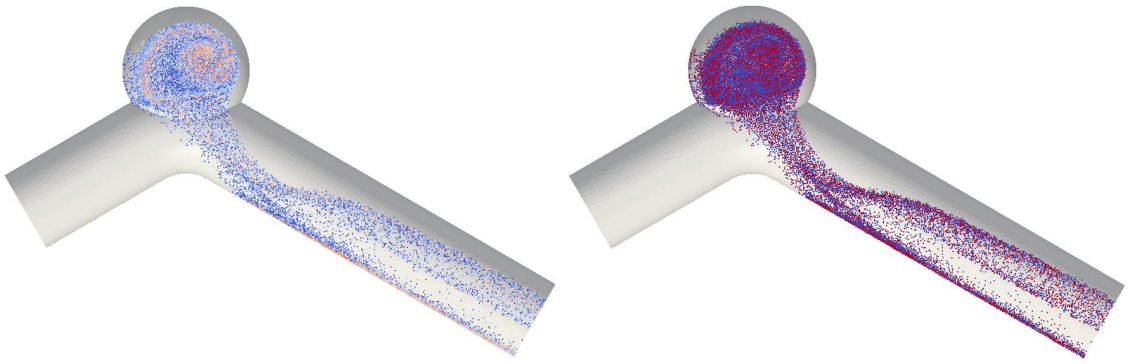


(c) $t = 0.41\ s$



(d) $t = 0.51\ s$

Figure 7.15: *Test case* - Sec. 7.5.1. Left: percentage ratio between the convective term and the sum of the convective and diffusive terms ($\frac{Conv}{Conv+Diff}$); right: tracer concentration in %. Longitudinal section of the ideal vessel.



(a) $\alpha_1 = 1 \cdot 10^{-8} m^2/s$ (b) $\alpha_2 = 1 \cdot 10^{-12} m^2/s$

Figure 7.16: *Test case* - Sec. 7.5.1. Concentration of two tracers having different diffusion coefficient. $t = 1\ s$. Same scale of Fig. 7.14.

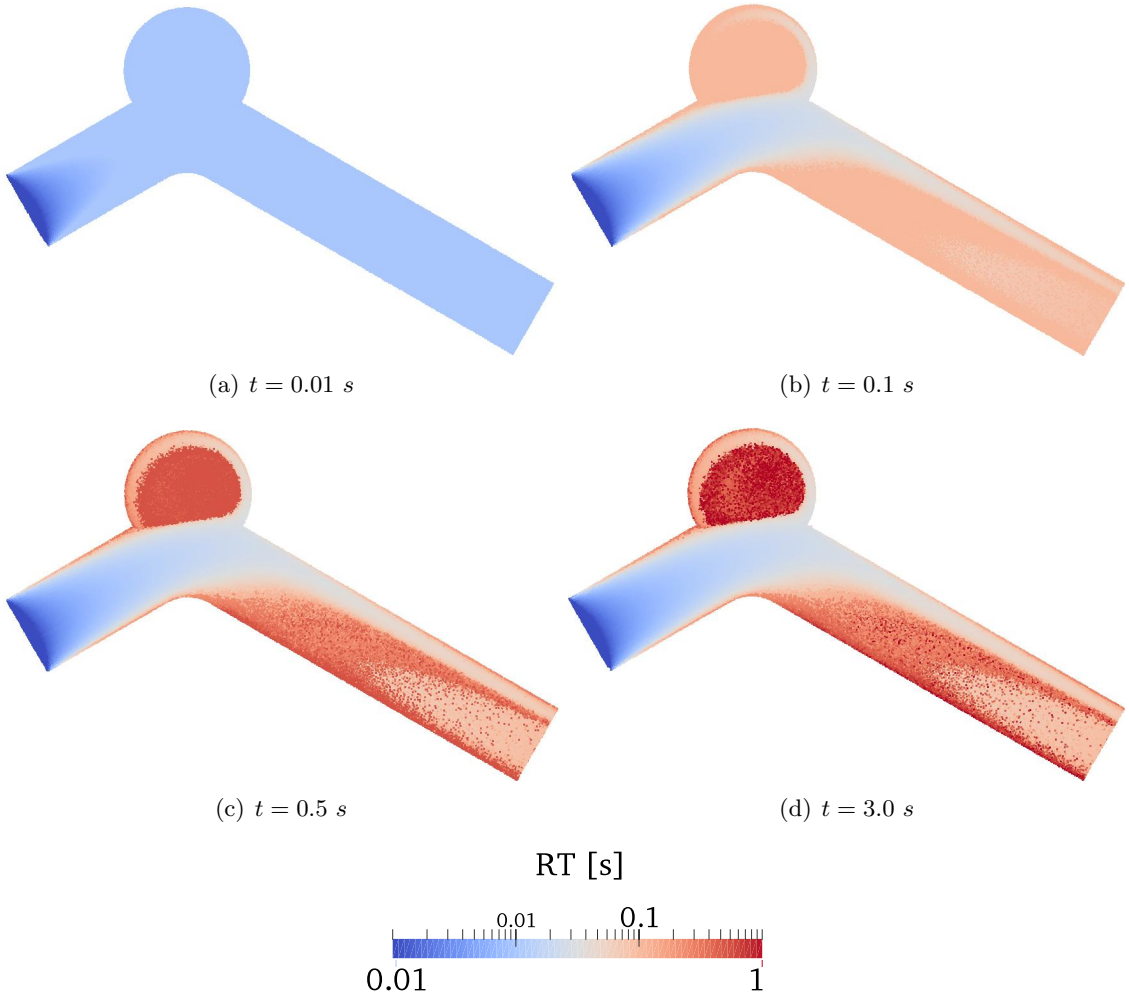
can be seen, in the regions where the tracer concentration is not negligible the ratio $Conv/(Conv + Diff)$ ranges between 30% and 70%. Similar results are observed at each time instant. Thus, no prevailing mechanism exist and both the contributions are relevant.

The Fig. 7.16 shows a comparison at time $t = 1\ s$ between the tracer discussed above and another tracer with $\alpha_2 = 1 \cdot 10^{-12} m^2/s$. The assumed shape is similar for the two tracers, but the second tracer has higher concentration values due to the lower diffusion coefficient (Fig. 7.16.b).

The *residence time* parameter is shown in Fig. 7.17 at different time instants. At the beginning, when the flow is not yet developed, the particles inside the aneurysm sac have the same RT (Fig. 7.17.a), then the particles are dragged by the vortex (Fig. 7.17.b) and are trapped inside the aneurysm (Fig. 7.17.c). The process continues until the steady conditions for the RT are reached (Fig. 7.17.d) when the RT becomes a constant property of the space representing the so-called *blood age*. The RT evolution in time has been calculated at fixed points (points P_1, P_2, P_3 and P_4 in Fig. 7.18.a) as average of the RT of the particles inside a sphere with center in each of these points and radius equal to $0.2 \cdot 10^{-3}\ m$ (that is two times the initial particle distance). The values are plotted in Fig. 7.18.b obtaining the mean values of $0.9\,s$, $0.07\,s$, $0.78\,s$ and $0.56\ s$ in the center, on the right wall, on the left wall and on the top of the aneurysm sac, respectively.

The *residence time* results at the steady-state have been compared with those obtained by the research group of Prof. A. Frangi (University of Sheffield, UK) using the *ANSYS* software (as shown in Fig. 7.19). As it is seen in the figure, the RT value obtained with *ANSYS* are higher close to the right wall of the aneurysm and to the outlet vessel wall than those obtained by the *SPH* model. This difference is related to the discretization of the finite-volume simulation where the mesh near the wall is composed by elements with maximum edge size of $0.1 \cdot 10^{-3}\ m$ (see Fig. 8.2 where the mesh is represented for the ideal aneurysm with the flow diverter device) with three prismatic layers. However, since there is a strong boundary layer near the wall, where flow is highly unpredictable, the comparisons can be considered acceptable.

The Fig. 7.20 shows the *activation potential* at different times. The AP value of the particles increases when the particles are trapped by the vortex on the right wall of the aneurysm that is clearly seen in the Figs. 7.20.a and 7.20.b. Then AP grows in the sac, while the particles are dragged inside the aneurysm, due to the highest RT values (Figs.



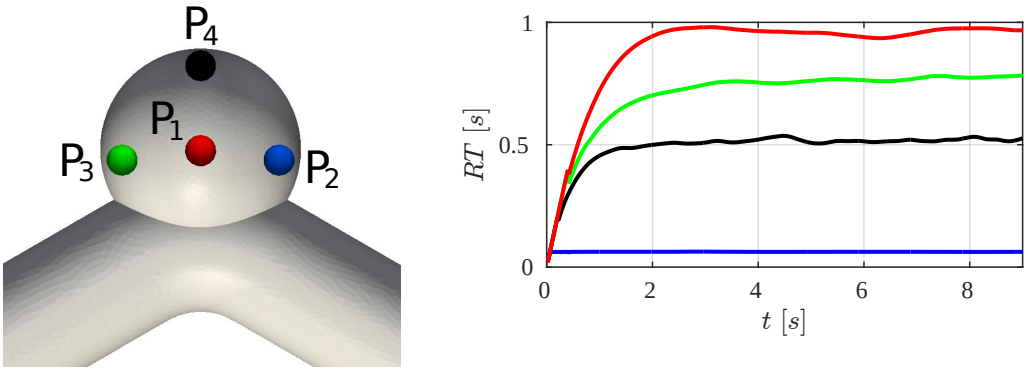
(a) $t = 0.01\ s$ (b) $t = 0.1\ s$

(c) $t = 0.5\ s$ (d) $t = 3.0\ s$

Figure 7.17: *Test case* - Sec. 7.5.1. *RT* at different time instants. Logarithmic scale.

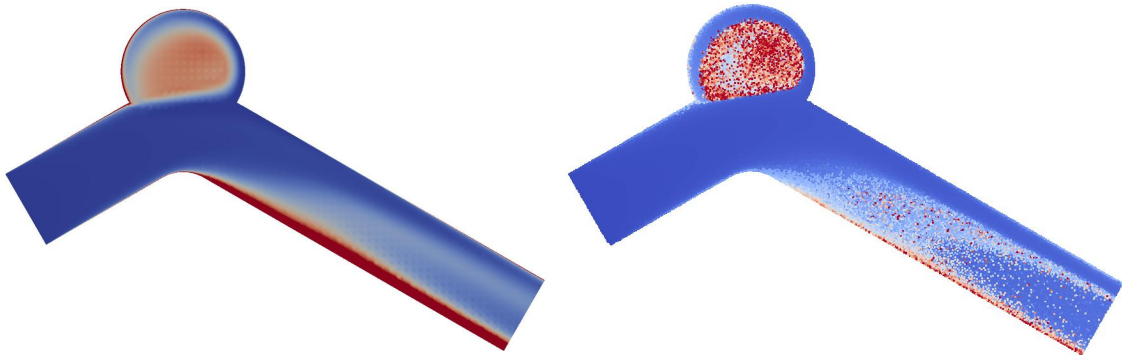
7.20.c and 7.20.d).

This is confirmed in Fig. 7.21.a where a threshold value of 1.1 Ns/m^2 is used and only the particles whose $AP > 1.1\ Ns/m^2$ have been represented. However, adopting the threshold value of Hellums (1994), no platelets have been activated since the shear stresses are too low for activating platelets mechanically.



(a) (b)

Figure 7.18: *Test case* - Sec. 7.5.1. a) Scheme of the points; b) bold red line: P_1; blue line: P_2; green line: P_3; black line: P_4.



(a) *ANSYS* simulation. Prof. Frangi's research group (University of Sheffield)

(b) *PANORMUS* simulation

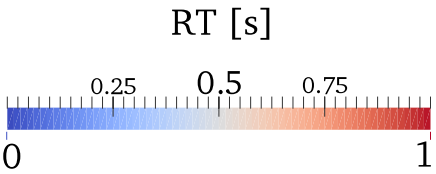
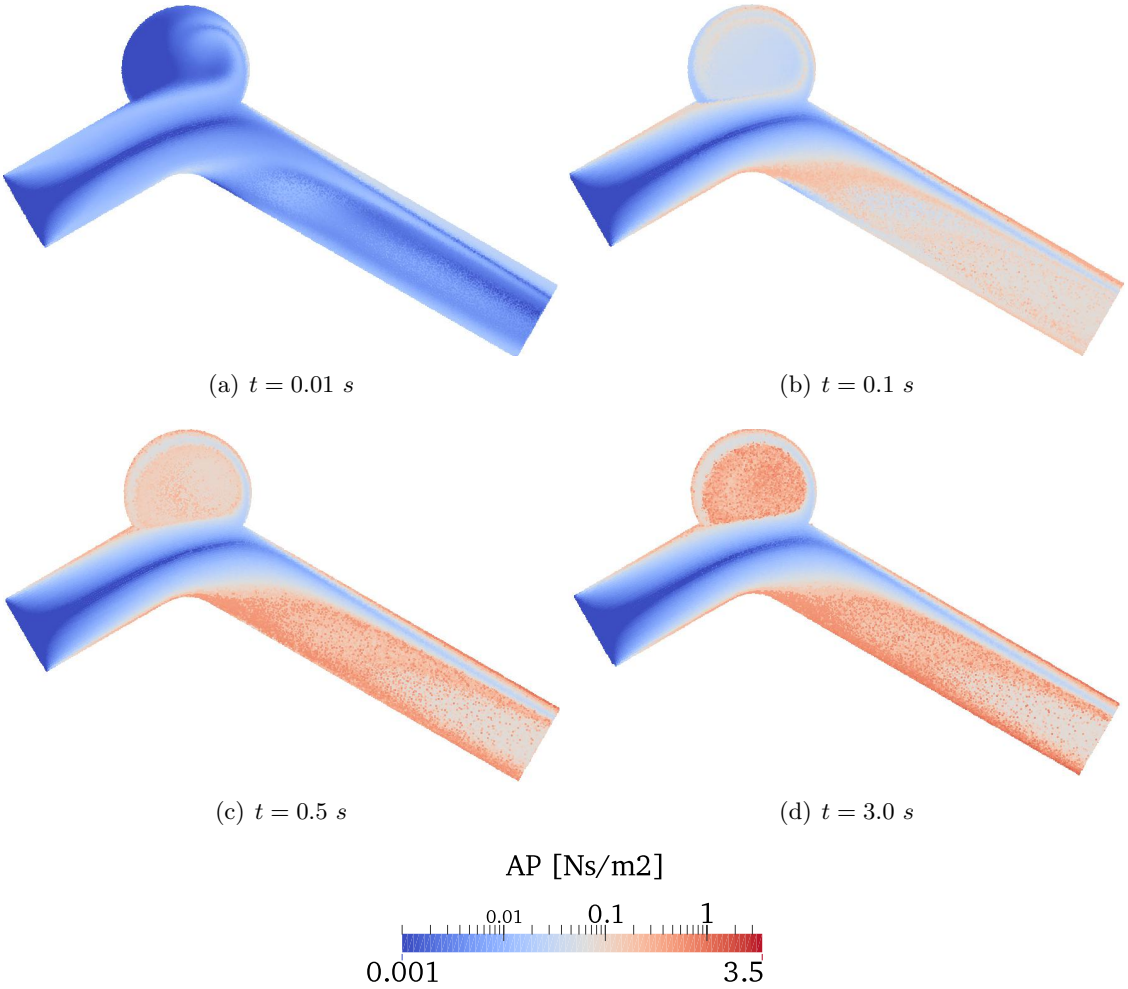


Figure 7.19: *Test case* - Sec. 7.5.1. *Residence time* at the steady-state. A commercial finite-volume solver (*ANSYS* software) *Vs* the *SPH* code (*PANORMUS-SPH*).



(a) $t = 0.01\ s$ (b) $t = 0.1\ s$

(c) $t = 0.5\ s$ (d) $t = 3.0\ s$

Figure 7.20: *Test case* - Sec. 7.5.1. *AP* at different time instants. Logarithmic scale.

(a) (b)

Figure 7.21: *Test case* - Sec. 7.5.1. a) AP threshold: 1 Ns/m^2; b) AP threshold: 3.5 Ns/m^2 (Hellums, 1994). Same scale of Fig. 7.20.

7.5.2 Aneurysm *C05*

The test case discussed in Sec. 6.4 has been used to analyze the transport of a passive tracer, the *residence time* and the *activation potential* parameters. A passive tracer with $\alpha = 4\ 10^{-8}\ m^2/s$ has been numerically injected at the inlet with 100 % of concentration starting from $t = 0$ to $t = 0.475\ s$ (about $3/5\,T$, where T is the pulsatile flow period equal to 0.792 s). The Fig. 7.22 shows the concentration at different times: during the tracer injection (Figs. 7.22.a,b,c,d,e and f) and after that (Figs. 7.22.g,h,i,j,k and l). As discussed in Sec. 7.5.1, a threshold value of 10% has been adopted to highlight the interface *tracer-laden* and *tracer-free* flow.

The RT development in time has been calculated in two points (A and B represented in Fig. 7.23.a) as the average of the values of the particles inside two spheres centered in the points A and B and having radius equal to $2\,\Delta x$. Differently from the trend obtained in the ideal aneurysm (shown in Fig. 7.18) where a steady flow had been imposed, the RT evolution plotted in Fig. 7.23.b follows the cardiac cycle since a pulsatile flow has been considered (as discussed in Sec. 6.4). The highest values are inside the aneurysm sac (point A) with a maximum value of 0.5 s and minimum of 0.25 s.

The RT has been calculated also in a third point corresponding to the center of the inflow section (point C in 7.23.a whose result has not been plotted in Fig. 7.23.b), obtaining the same shape of the pulsatile flow (as for the points A and B) but very low values due to the high velocity in this region. Specifically, the maximum RT obtained at point C has been $1 \cdot 10^{-3}\ s$ that is 0.2 % of the maximum value at point A (which is equal to 0.5 s as mentioned above).

The *activation potential* time evolution at point A is plotted in Fig. 7.23.c. As for the RT evolution, the AP has a pulsatile trend although it presents more fluctuations.

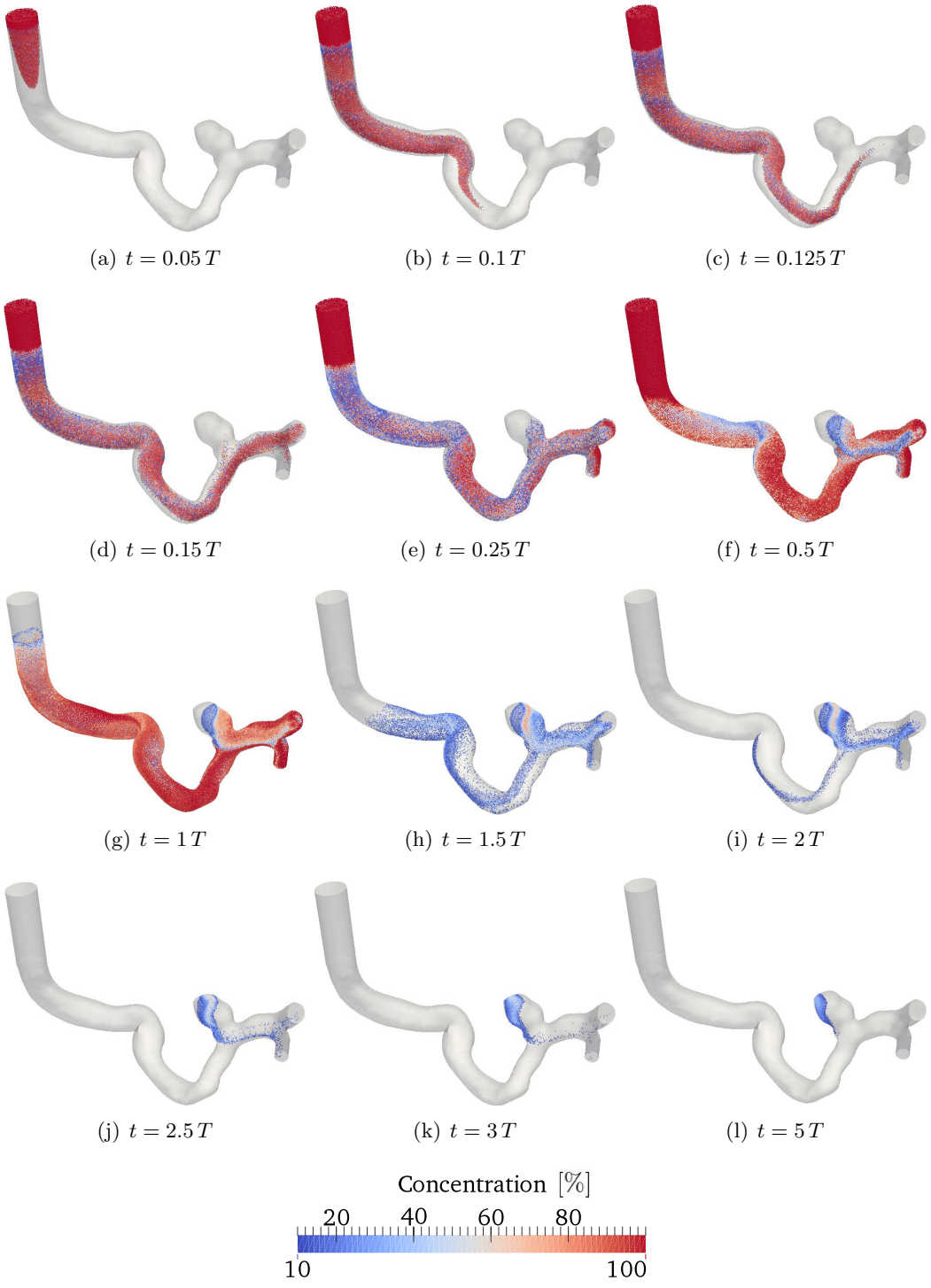
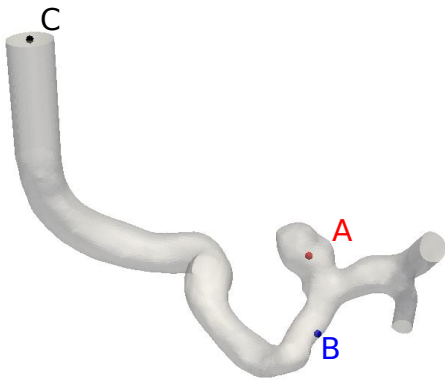
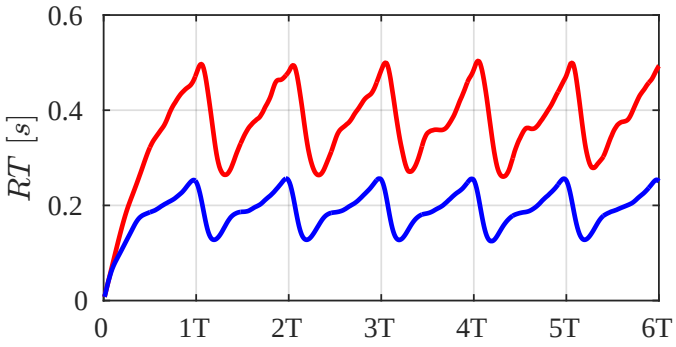


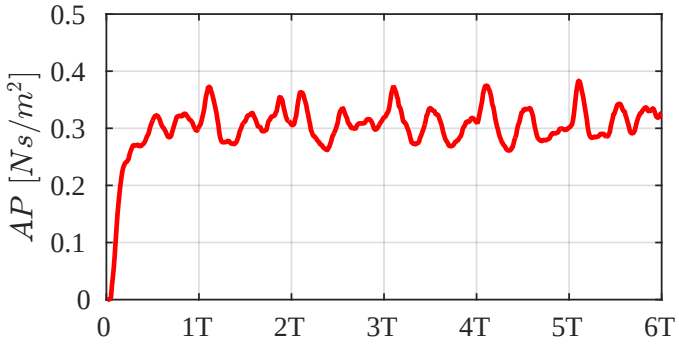
Figure 7.22: *Test case* - Sec. 7.5.2. Tracer concentration at different times. Gray area: domain contour.



(a)



(b)



(c)

Figure 7.23: *Test case* - Sec. 7.5.2. a) Scheme of the points; b) *residence time* at point A (red line) and at point B (blue line); c) *activation potential* at point A.

Chapter 8

Concluding Remarks and Future Work

8.1 Conclusions

Although mechanisms of aneurysm rupture have not been completely explained yet, the effects of blood flow on the vascular walls are widely accepted as risk factors.

Computational fluid dynamics can be a powerful tool for simulating hemodynamics in *patient-specific* cerebral aneurysm models. Numerical tools may help to evaluate the impact of various clinical interventions and to identify hemodynamic factors affecting treatment outcomes.

In this study the *SPH* method has been used to study hemodynamics in cerebral aneurysms. Due to its Lagrangian nature, *SPH* is able to ease up the treatment of geometrically complex domains such as cerebral vessels. The open-source *PANORMUS-SPH* code has been specialized for the analysis of cerebral aneurysm hemodynamics.

The *Incompressible SPH* approach has been employed which implies the resolution of a *PPE*. Since the fast and accurate solution of the *PPE* is a fundamental issue in the *ISPH* approach, a peculiar attention has been paid on the *PPE* solution method. The numerical model accuracy and efficiency have been improved through the solution of a unique equation system made of a *PPE* for each particle of the computational domain. In order to save memory and to accelerate matrix-vector multiplications, the matrix system has been saved using the *CRS* format where only the non-null terms are stored. The iterative *BiCGSTAB* method has been implemented due to the non-symmetricity of the coefficient matrix. In order to speed-up the computation of the solution method, a preconditioning algorithm has been employed. The performance evaluation (described in Sec. 2.7.2) has shown that the *BiCGSTAB* method provides a more accurate solution than those obtained with the semi-implicit *SOR* algorithm, where some difficulties to achieve convergence can occur (see Fig. 2.10). Moreover, the preconditioning algorithm noticeably accelerates convergence of the optimization procedure (see Fig. 2.11).

An inflow/outflow procedure has been developed to account for the treatment of *open-boundaries* in *SPH* (Chap. 3). This technique allows to set steady and unsteady pressure boundary conditions in the computational domain inlets and outlets or to impose the velocity profile (stationary or pulsatile) at the inlet when it is known. The leaving and entering the domain of *effective* particles have been dealt with (see Sec. 3.2.2 and Sec.

3.2.3). An important point is the handling of the frequency of generation that requires a dynamically adaptive procedure in order to suitably satisfy the continuity constraint.

Flows developing in relatively simple geometric domains have shown a very good agreement of numerical results with analytical solutions available in steady (Fig. 3.13) and unsteady conditions (Fig. 3.18). Furthermore, in the considered test cases the number of *effective* particles has been substantially unchanged during the simulations (Fig. 3.16) ensuring the global mass conservation. The method has been used to investigate the blood flow in cerebral vessels (see Sec. 6.3) and in cerebral aneurysms by imposing stationary (see Sec. 6.2) or pulsatile flow conditions (see Sec. 6.4 and Sec. 6.5).

A multi-resolution technique has been developed in *SPH*. This procedure allows to alleviate one of the main drawbacks of the *SPH* method related to the high computational costs with respect to *mesh-based* methods. With this technique, it has been possible to improve the accuracy of the simulation results while reducing the computational costs at the same time. The internal *interfaces* separating neighboring subdomains have been managed as *open-boundaries*. No particle splitting/coalescing methods have been employed to take into account the variable *smoothing length*. The movement of particles from one block to another has been handled using procedures similar to those employed at the inlet and outlet sections, as discussed in Sec. 4.2.4. The simultaneous solution of the *PPE* sub-systems of each block allowed to obtain a perfectly matched solution among the single subdomains, where different particle sizes are employed.

The analysis of benchmark test cases has been mostly focused on the checking of the seamless transition of the results to the interfaces between neighboring blocks (see Figs. 4.10, 4.13, 4.14 and 4.15). The local and global mass conservation have been accurately checked as well (see Fig. 4.11).

An accurate analysis of the reduction of the particle number achieved through the *Multi-Domain* (*MD*) approach has been performed too. To this aim, a giant cerebral aneurysm has been considered (see Sec. 6.5) where the reduction was about 50 times with respect to the corresponding *Single-Domain* approach. The resulting reduction of computational time is almost proportional to that of the particle number since the overloads due to the *interface* management is almost negligible.

The performed parallelization of the numerical solver on multiple *CPUs* using *MPI* libraries has been aimed again to reduce the computational efforts. The extremely complex procedures developed in both *Single-Domain* and *Multi-Domain* approaches have been accurately described, showing the good accuracy and computational efficiency of the employed algorithms. The results of the scalability tests have confirmed this efficiency up to the available number of processors (32 in the workstations which were available to use), as shown in Figs. 5.15 and 5.31.

Both the *mesh-less* feature of *SPH* and the numerical improvements introduced within this thesis allowed the employed model to accurately analyze the hemodynamics in cerebral aneurysms. Ideal and real aneurysm geometries have been investigated. The results have shown complex hemodynamic patterns in the neck and sac of the aneurysms (Figs. 6.14 and 6.15) where vortices form and rapidly dissolve during the cardiac cycle (Fig. 6.25). Furthermore, some indices mainly related to the *wall shear stress* have been calculated in order to characterize the stress state of the vessel walls (see Fig. 6.18). The solver has been successfully verified through comparison with other numerical solutions based on the use of finite-volume solvers (Fig. 6.4). Some comparisons with laboratory experiments

have been performed too (see Fig. 6.19).

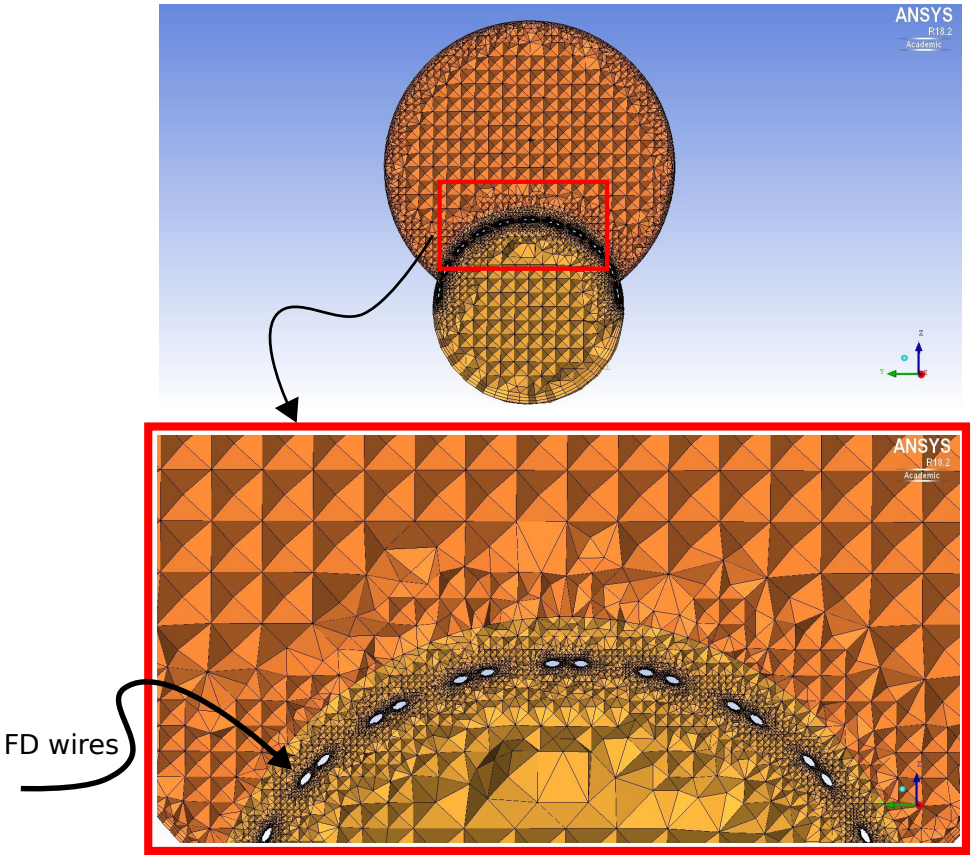
This research lays the foundation for the future development of a blood clot model in *SPH*. The *SPH* method is very suitable to model multi-phase processes like those due to *thrombi* formation inside the aneurysm sac and in the parent vessel which can occur after inserting an *endovascular* device. Specifically, in this research study a *tracer transport* model has been developed in *SPH* for flow visualization and prediction of intra-aneurysmal regions prone to *thrombus* deposition. The *tracer transport* model has been parallelized and implemented in the *Multi-Domain* approach. The model has been applied to analyze the transport of passive tracers throughout cerebral vessels with aneurysms (Figs. 7.14, 7.22) and the effect of different diffusion coefficients (Fig. 7.16). These simulations have reproduced the contrast agent injection during *angiography* tests.

Through the *tracer transport* model, the *residence time* (RT) parameter has been analyzed. The employed test cases have shown how RT reaches higher values inside the aneurysm sac where the forming vortex traps the fluid particles (Fig. 7.17). The RT has been investigated while imposing stationary flow. After some transitory instants, RT reaches the steady-state becoming a spacial field which indicates the mean *blood age* in each region of the domain. When analyzing pulsatile flows, it has been shown how RT periodically changes following the pattern of the cardiac cycle (Fig. 7.23).

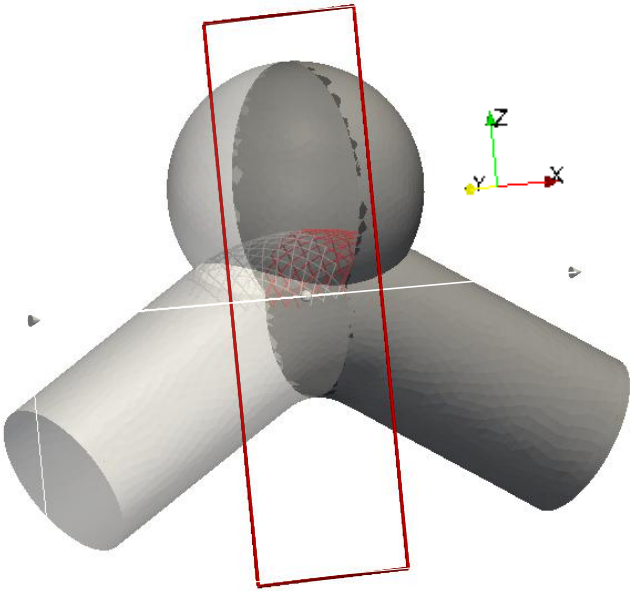
A *time-exposure stress* model has been implemented too, in order to simulate *mechanical platelet activation*. Untreated aneurysms has been analyzed. The results have shown that using the threshold value of 3.5 Ns/m^2 (Hellums, 1994) no platelet has been activated (Fig. 7.20).

8.2 Future developments

CFD is a promising method to study the impact of *flow diverter* (*FD*) devices on intra-aneurysmal hemodynamics. However, a well-known issue in "classical" numerical simulations of blood flow past *FD* (or *endovascular* devices in general) is the meshing process due to the geometrically complexity of these devices. Moreover, another relevant issue is the large difference scale between the size of the *FD* strut thickness and the cerebral vessels in the vicinity of the aneurysm. The Fig. 8.1.a shows an example of a finite-volume mesh used to discretize an ideal vessel with a spherical aneurysm and a *FD* device placed across the aneurysm neck. As it is seen in the figure (which shows a cross-section whose scheme is represented in Fig. 8.1.b) the mesh around the *FD* strut (whose wires are indicated in the figure) is extremely complex. The *SPH* method, due to its *mesh-less* nature, can be considered a very convenient tool for modeling the highly complex geometries of the *FD* struts. Future developments of the present work will address the employment of the *SPH* method for modeling aneurysms treated with *FD* devices. Specifically, analyses can be performed on the hemodynamic changes induced inside the aneurysm sac by the devices. To this aim, the *Multi-Domain* approach is fundamental since a simulation performed with a constant *smoothing length* value in the whole domain would be prohibitive in term of computational efforts and memory requirements. Moreover, *parallel computing* is essential to perform simulations in reasonable time. An example of the suitability of the *Multi-Domain* technique is shown in Fig. 8.2 where a domain decomposition into 5 blocks is performed. In order to show the reduction of the particle number allowed by the *MD* approach, kh values ranging between $6 \cdot 10^{-5}\ m \div 4 \cdot 10^{-4}\ m$ are used in the figure. In this example the lowest kh value of $6 \cdot 10^{-5}\ m$ has been selected to allow the distribution of at least 15 particles in each direction between the *FD* wires (D_h in the



(a)



(b)

Figure 8.1: a) Example of mesh for finite-volume simulation of aneurysm with *FD* device. From: Prof. Frangi's research group (University of Sheffield); b) cross-section.

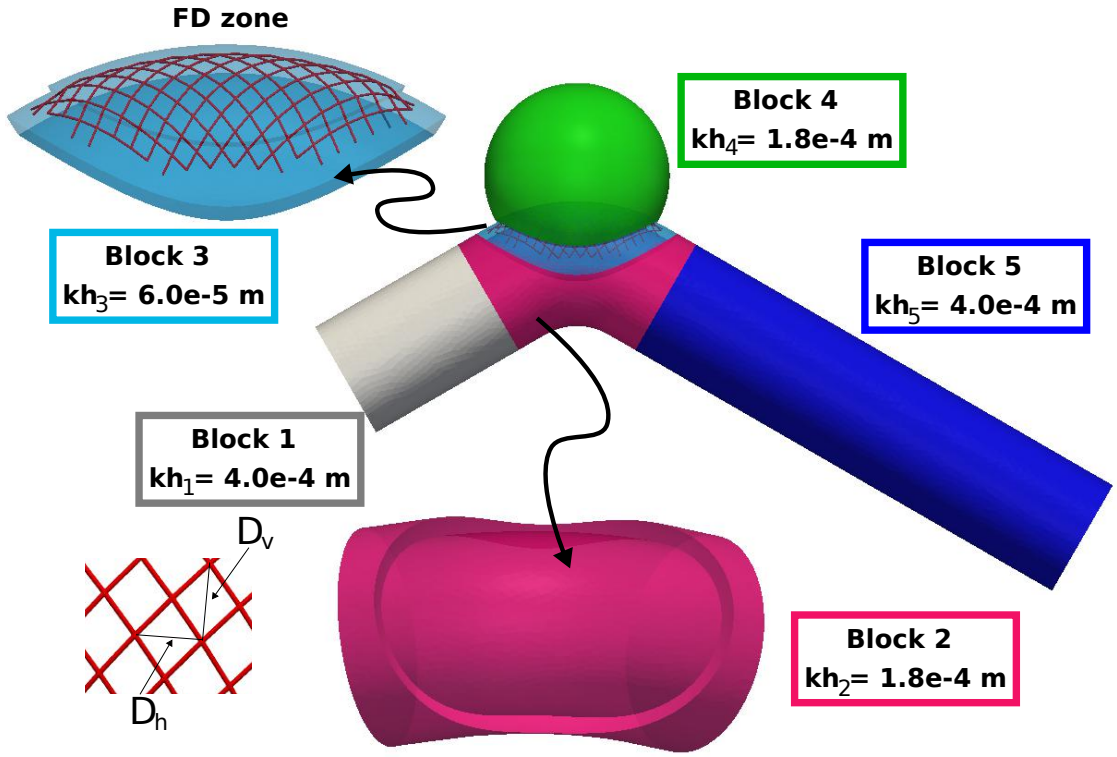


Figure 8.2: MD decomposition of an ideal aneurysm geometry with *FD* device.

figure equal to $4.5\,10^{-4}$ m). The resulting total number of particles is 840 183, about 4% of the value that would have been obtained using a constant value of the smaller kh value (equal to $kh_{min} = 6 \cdot 10^{-5}$ m) in the whole computational domain (as shown in Tab. 8.1). Much higher reductions of the computational efforts could be achieved using the proposed *Multi-Domain* approach with higher values of the ratios of the employed *smoothing lengths*.

Moreover, it is known that both *FD* size and compaction level remarkable affect the treatment outcomes. High content of activated platelets in the clots forming inside the aneurysm sac promotes the generation of *white thrombi* that, due to their stability and well-organized structure, could facilitate the handling process after aneurysm flow-diversion. Future research studies could employ the implemented *mechanical platelet activation* model in order to predict if blood shear stresses near the *FD* struts can mechanically activate platelets. The effects of different sizes of *FD* wires and *FD* compaction could be investigated analyzing both the flow reduction inside the aneurysm sac and the level of platelet activation due to shear stress generated near the *FD* struts.

Furthermore, by including aneurysmal biochemistry in the implemented *tracer transport* model, future work could be devoted to study the *chemical platelet activation* process.

Further, interesting applications of the *SPH* technique are those related to the modeling of multi-phase processes of *thrombus* formation and deposition inside the aneurysm sac.

In this study blood has been modeled as *Newtonian* fluid (as discussed in Sec. 6.1). Future research activities should take into account the *shear-thinning* behavior of the blood. Specifically, the particle viscosity (that in the present study has been considered constant) could be dynamically updated as function of the shear strain acting on the particle.

B_n	$N_{e,Bn}\,(kh_{Bn})$	$N_{e,Bn}\,(kh_{min})$
B_1	9 835	2 914 075
B_2	108 613	2 932 551
B_3	538 354	538 345
B_4	154 679	4 176 333
B_5	28 702	8 504 296
$N_{e,Tot}$	840 183	19 065 608

Table 8.1: First column: block number; second column: number of *effective* particles with kh_{Bn} (*MD* approach); third column: number of *effective* particles with $kh_{min} = kh_{B3} = 6 \cdot 10^{-5}$ m (*SD* approach); last row: total number of *effective* particles

Finally, since the *SPH* method is very suitable to model moving boundaries in the future it could be used to analyze the life cycle of cerebral aneurysms which includes formation, growth and rupture.

Appendix A
Coupled FVM-SPH method

During the PhD program, the author focused not only on the *SPH* method but also on the development of a procedure based on the combining *FVM* and *SPH* techniques (the method will be named *Coupled FVM-SPH* in the following).

The *Coupled FVM-SPH* method, that is particularly suitable for studying coastal engineering problem, is discussed in this chapter relying on the paper of Napoli et al. (2016).

As discussed in the Methodological note, the *PANORMUS* code contains both *FVM* and *SPH* solvers. These solvers have been combined for developing the coupled procedure. The *PANORMUS-FVM* code is briefly outlined to describe the *Coupled* technique. Regarding the *SPH* solver please refers to Chap. 2.

In order to show the performance of the method in both confined and free-surface flow and in $3D$ and $2D$ approximations, two test cases are shown: the lid-driven cavity problem and the solitary wave *run-up* and *overtopping* on a seawall.

A1 The *Coupled FVM-SPH* procedure

The *Coupled FVM-SPH* method combines the larger computational efficiency of *grid-based* approaches with the flexibility of the *SPH* method.

To this aim, the computational domain is partitioned in order to use the finite-volume method in some portions of the domain, while employing the *mesh-less* Lagrangian approach in the regions with the higher geometric complexity and/or the presence of moving boundaries, rapidly evolving free-surfaces and multiphase processes.

A1.1 The finite-volume solver

The *PANORMUS-FVM* solver allows to solve the momentum equations for incompressible flows using curvilinear non-orthogonal structured grids made of hexahedral cells, with a cell-centered discretization.

The finite-volume approximation of the integral momentum equation in the α-direction can be written for the generic cell of volume V as

$$V \frac{u_\alpha^{k+1} - u_\alpha^k}{\Delta t} + \sum_f u_{f_\alpha} \Phi_f - \nu \sum_f \left. \frac{\partial u_\alpha}{\partial n} \right|_f A_f + \\ + \sum_f \psi_f A_{f_\alpha} + V \; g_\alpha = 0 \qquad (\text{with } \alpha = 1, 2, 3) \tag{A.1}$$

where the summations with index f (indicating values on the f-th cell face) are extended to the six cell faces of the hexahedral element, u_α is the α-th component of the cell-

averaged velocity, Δt is the time step, the index k is used to indicate the variables at the k-th time step, Φ_f is the volume flux through the f-th cell face, ν is the fluid kinematic viscosity, n is the normal direction to the considered cell face pointing outward, ψ is the kinematic pressure, A_{f_α} is the projection in the α-direction of the cell face area and g_α is the α-th component of the gravity acceleration. When considering turbulent flows in the framework of the statistical approach based on the *Reynolds* averaging of the momentum and continuity equations (*RANS*), in the eqns. A.1 the velocity u_α must be substituted with the *Reynolds* average $\overline{u}_\alpha$ and the term

$$\sum_f \tau_{f_{\alpha\beta}} A_{f_\beta} \qquad (\text{with } \alpha, \beta = 1, 2, 3)$$

must be added to the left-hand side of eqns. A.1, where $\tau_{\alpha\beta}$ is the *Reynolds* stress tensor.

An explicit second-order accurate in time *Adams-Bashforth* scheme is used for the solution time marching.

As explained for the *SPH* solver in Chap. 2, a *fractional-step* method is used to solve eqns. A.1 (for $\alpha = 1, 2, 3$).

In the *predictor-step*, the *intermediate* velocity u^*_α is obtained removing the pressure terms from the momentum equations

$$V \frac{u^*_\alpha - u^k_\alpha}{\Delta t} + \sum_f u_{f_\alpha} \Phi_f - \nu \sum_f \left. \frac{\partial u_\alpha}{\partial n} \right|_f A_f + V \; g_\alpha = 0 \tag{A.2}$$

In order to correct the u^*_α velocities while imposing the continuity constraint for incompressible flows ($\partial u_\beta / \partial x_\beta = 0$), an irrotational corrective velocity field u^c_α must be calculated. The potential $-\psi \Delta t$ of u^c_α ($u^c_\alpha = -\Delta t \partial \psi / \partial x_\alpha$) is obtained solving a *PPE* which in the finite-volume approximation read as

$$\sum_f \left. \frac{\partial \psi}{\partial n} \right|_f A_f = \frac{\sum_f u^*_{f_n} A_f}{\Delta t} \tag{A.3}$$

where $u^*_{f_n}$ is the *intermediate* velocity component in the direction normal to the f-th cell face. The boundary conditions for eqn. A.3 at inflow sections or solid walls are obtained as

$$\left. \frac{\partial \psi}{\partial n} \right|_f = -\frac{1}{A_f} \frac{\Phi^{k+1}_f - \Phi^*_f}{\Delta t} \tag{A.4}$$

to be imposed on the cell faces lying on the boundaries, where Φ^{k+1} and Φ^* are the volume fluxes through the faces at the $(k+1)$-th time step (null on impermeable walls) and at the *intermediate* level (obtained extrapolating the u^* velocities from the interior flow towards the wall). The boundary conditions over a free-surface are of *Dirichlet* type, with null values of the potential ψ.

In the *corrector-step*, the divergence-free updated velocity field is obtained as

$$u^{k+1}_\alpha = u^*_\alpha + u^c_\alpha = u^*_\alpha - \frac{\sum_f \psi_f A_{f_\alpha}}{V} \Delta t \tag{A.5}$$

For free-surface flows, the velocity fluxes through the cell faces lying at the air-liquid interface are obtained from the integral continuity condition $\sum_f \Phi_f = 0$ applied to the surface cells. These *free-surface* faces are thus moved upward or downward at the end of each time step according to the *kinematic condition* for the free-surface as discussed by

(Lipari and Napoli, 2008). The *kinematic condition* in finite-volume approximation can be expressed as

$$\frac{\eta^{k+1} - \eta^k}{\Delta t} - \frac{\Phi_S}{A_S n_{S_v}} = 0 \tag{A.6}$$

where η is the free-surface level (measured from a reference horizontal plane), the index S indicates faces lying on the free-surface, A_S is the face area and n_{S_v} is the projection in the vertical direction of its normal unit vector.

A1.2 The treatment of *hybrid interfaces*

The Fig. A3 shows the computational domain subdivision into region discretized through a structured grid of hexahedral cells (the *FVM-domain*) and another covered by Lagrangian particles (the *SPH-domain*). In the sketch of the figure, the *starting particle distance* Δx is half the kh value and the FVM cells are cubes of side kh.

The *FVM-domain* and *SPH-domain* are separated by surfaces named *hybrid interfaces* (or simply *h-interfaces*). The *h-interface* is composed by the faces of the FVM-grid cells neighboring to the *SPH-domain* which are divided into two triangles (the *h-interface triangles* represented in the figure with red bold line filled by gray region) following the procedure discussed in Sec. 2.5.

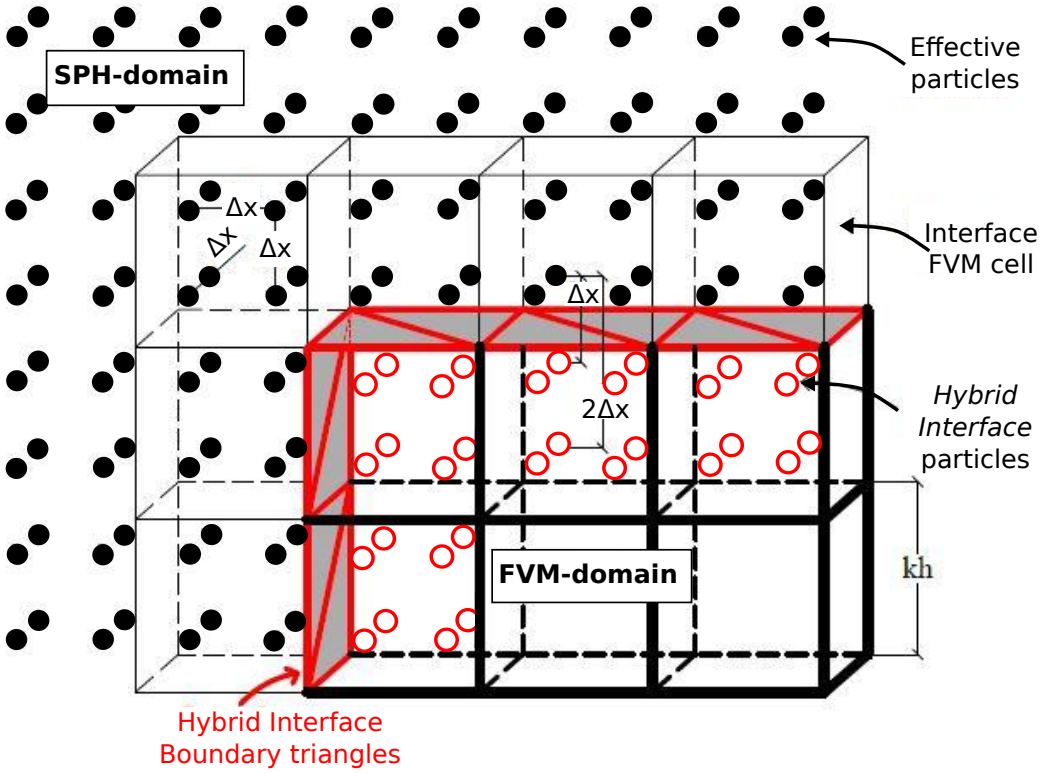


Figure A3: *Coupled FVM-SPH.* Domain decomposition into *FVM* (bottom-right area) and *SPH* (left and top areas) domains. Bold and thin black lines: *effective FVM* cells (FVM_e) and *hybrid interface FVM* cells (FVM_{hi}), respectively; full black circles: *effective* particles; empty red circles: *hybrid interface* particles (HI); red bold line: *hybrid interface triangles*. Taken from: Napoli et al. (2016), 678, fig. 2.

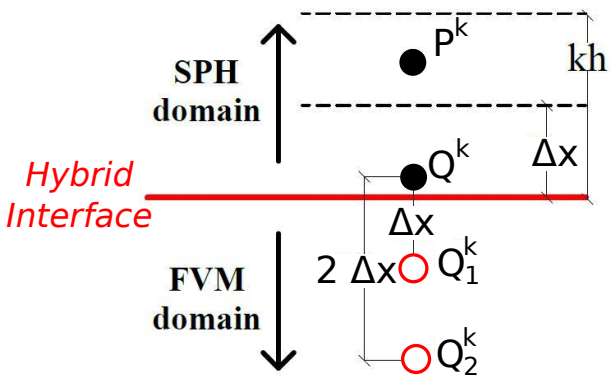


Figure A4: *Coupled FVM-SPH.* Generation of the HI particles. Full black circles: *effective* particles; empty red circles: HI particles. Taken from: Napoli et al. (2016), 679, fig. 3.

In order to match the solution in the neighboring domains, a layer of FVM-cells named *hybrid interface FVM* (indicated with the symbol FVM_{hi} to distinguish them from *effective FVM* cells, FVM_e), is placed near the *h-interface* in the *SPH-domain*. Moreover, the *hybrid interface triangles* are used to generate *ghost* particles (named *h-interface*, HI) in the *FVM-domain*. The HI particles are generated using a similar procedure explained for the IO and IP particles (see Chap. 3 and Chap. 4, respectively). Specifically, they are generated starting from the *effective* ones with distances from the *hybrid interface* lower than Δx. These *ghost* particles are placed along the line connecting each of the aforementioned *effective* particle and normal to the *hybrid interface*. In Fig. A4 the particle Q at the k-th time step lies at a distance lower than Δx from the *h-interface* and thus generates the HI particles Q_1 and Q_2 at distances from Q equal to Δx and $2\,\Delta x$, respectively, in the direction normal to the *h-interface*. No HI particles are generated with reference to the particle P lying at distance larger than Δx from the *h-interface*.

At the *h-interface* an interpolation procedure (whose scheme is shown in Fig. A5) is used allowing to obtain a smooth transition of the solution between the grid-discretized domain and the particle-covered region.
The set of FVM eqns. (A.2 - A.5) is solved only on the *effective FVM* cells (bold black line in the figure). The hydrodynamic variables in the centroids $\mathbf{x}_{int}$ of the *hybrid interface FVM* cells can be obtained following two different procedure. The former (see scheme in Fig. A5.b) is based on the *SPH kernel approximation* at the $\mathbf{x}_{int}$ point

$$f(\mathbf{x}_{int}) = \frac{\sum_j \frac{m_j}{\rho_j} f_j W_{cj}}{\sum_j \frac{m_j}{\rho_j} W_{cj}} \tag{A.7}$$

where the sum is extended to the particles having distance d_{cj} from $\mathbf{x}_{int}$ shorter than kh (particles inside the *support domain* Ω_c of the selected FVM_{hi} cell which is indicated with a dashed circle in the figure) and W_{cj} is the *kernel function* considering the same d_{cj} distance. An alternative procedure to the eqn. A.7 (not reported in Napoli et al. (2016)) is based on a Taylor series expansion around the closest *effective* particle (blue particles in Fig. A5.c)

$$f(\mathbf{x}_{int}) = f_R + \sum_{j=1}^{N_R} \frac{m_j}{\rho_j} (f_j - f_R) \frac{\partial W_{Rj}}{\partial x_\alpha} (x_{int,\alpha} - x_{R,\alpha}) \tag{A.8}$$

where R is the closest *effective* particle to the point $\mathbf{x}_{int}$, the summation convention on repeated indices is used for the index α, N_R is the number of particles inside the *support domain* of R (Ω_R) and W_{Rj} is the *kernel function* considering the distance (d_{Rj}) between the particle R and its neighboring particle j.

Correspondingly, the *SPH* eqns. 2.20 - 2.24 (explained in Chap. 2) are solved on the

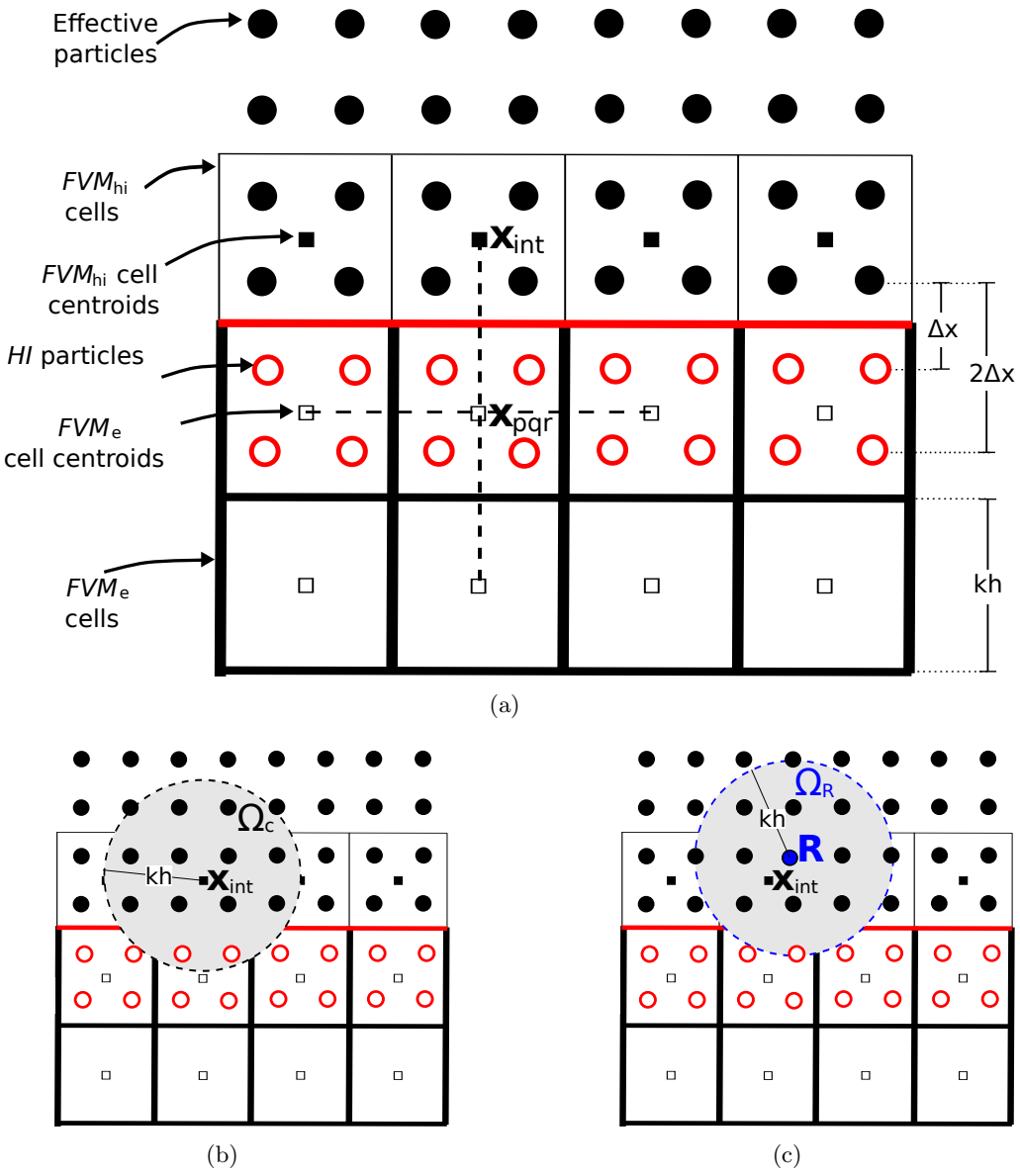


Figure A5: *Coupled FVM-SPH*. Scheme of the interpolation procedure at the *h-interface*. Full black circles: *effective* particles; empty red circles: *HI* particles; red bold line: *h-interface*; bold and thin black line: *effective* (FVM_e) and *hybrid interface* (FVM_{hi}) FVM grid cells, respectively; full and empty black squares: centroid of the *effective* and *interface* FVM grid cells, respectively. a) General scheme. Taken from: Napoli et al. (2016), 680, fig. 4; b) scheme for eqn. A.7. Dashed black circle: *support domain* Ω_c of the centroid $\mathbf{x}_{int}$; c) scheme for eqn. A.8. Dashed blue circle: *support domain* Ω_R of the closest particle R.

effective particles only, while the f_i variables at the i-th HI particle are obtained through an interpolation from the FVM solution based on the second-order accurate Taylor series expansion

$$f_i = f(\mathbf{x}_{pqr}) + \left.\frac{\partial f}{\partial x_\alpha}\right|_{pqr} (x_{i,\alpha} - x_{pqr,\alpha}) \tag{A.9}$$

where pqr are the indices of the FVM cell nearest to the current HI particle (see Fig. A4.a).

It is important to highlight that the sums in eqn. (A.7) to obtain the values on the *interface FVM cells* involve HI particles, as it is seen in Fig. A5 where the HI particles are indicated in red. On the other hand, the Fig. A4.a shows that the dashed-line cross centered in the centroid $\mathbf{x}_{pqr}$ (which indicates the cells used to calculate the derivatives for interpolating the variables on the HI particles close to $\mathbf{x}_{pqr}$ through eqn. A.9) involves the *interface cell* $\mathbf{x}_{int}$.

In Napoli et al. (2016) an iterative procedure is used at the *h-interface* for solving the eqns. A.7 and A.9 for each variable (*intermediate* and *corrected* velocities and *pseudo-pressure*) till achieving convergence of the solutions on both the *interface FVM cells* and HI particles. The iterative procedure is applied first on the *predictor-step* velocities after solving the eqn. A.2 on each FVM grid cell and the eqn. 2.20 on each *effective* particle. Further, the system made of eqns. A.7 and A.9 is solved to obtain the ψ values of the HI particles after each internal step of the *Poisson* resolution on the FVM grid and the SPH particles (eqns. A.3 and 2.21, respectively). The iterative procedure is applied again, at the end of each time step, to obtain the *corrected* velocities.

In this research study another procedure has been implemented in order to solve a whole system (made of equation for the cells and for the particles) using the *preconditioned BiCGSTAB* method. In particular, the values at the *interface FVM cell* are obtained using eqn. A.8 rather than eqn. A.7. The procedure allows to speed up the method without influencing the results. The details of the new procedure related to the velocity and PPE systems are shown below.

Solution matching at *h-interfaces* for the velocities

In the new procedure, the velocity system is made by the equations for the *hybrid interface FVM* cells and the equations for the *hybrid interface* particles.

For the velocity system, eqn. A.8 can be rewritten as

$$u_c = u_R + \sum_{j=1}^{N_R} \frac{m_j}{\rho_j} (u_j - u_R) \nabla W_{Rj}(\mathbf{x}_c - \mathbf{x}_R) \tag{A.10}$$

where for simplicity the variables at the centroid $\mathbf{x}_{int}$ are indicated with the subscript c (thus u_c is the velocity at the point $\mathbf{x}_{int}$) and u_R is the velocity of the closest *effective* particle R. Some manipulations can be made to eqn. A.10 in order to explicit the coefficient matrix and the *right-and-side* of the portion of the velocity system related to the FVM_{hi} cells. To this aim, the particles inside the *support domain* of R could be *effective* (whose number is indicated with N_R^e) or HI (whose number is N_R^{HI}) particles. Therefore, the summation in eqn. A.10 can be splitted in the following to summations

$$u_c = u_R + \sum_{j=1}^{N_R^e} C'_{cr} (u_j - u_R) + \sum_{j=1}^{N_R^{HI}} C'_{cr} (u_j - u_R) \tag{A.11}$$

where $C'_{cr} = \frac{m_j}{\rho_j}\nabla W_{Rj}(\mathbf{x}_c - \mathbf{x}_R)$.

Finally, using eqn. A.11, the equation system for the *interface FVM* cells can be written as

$$u_c - \sum_{j=1}^{N_R^{HI}} C'_{cr} u_j = RHS_c \ , \ c = 1, .., N_{hc} \tag{A.12}$$

$$\text{with} \qquad RHS_c = u_R + \sum_{j=1}^{N_R^e} C'_{cr}\,(u_j - u_R)$$

where N_{hc} is the number of *interface FVM* cells and the RHS_c is *right-and-side*.

The eqn. A.9 for the velocity at the centroid of the FVM_{hi} cell can be written as

$$u_i = u(\mathbf{x}_{pqr}) - \frac{\sum_f u_f A_f}{V} \ , \ i = 1, .., N_{HI} \tag{A.13}$$

where N_{HI} is the total number of *hybrid interface* particles and u_f is a suitable approximation of the velocity value at the cell face obtained using the values of the surrounding cells. Obviously, the neighboring cells could be FVM_e or FVM_{hi} and thus will effect on the RHS and on the coefficient matrix of the system, respectively.

Therefore, the velocity system is made of $N_{hc} + N_{HI}$ unknowns values. The velocity system is solved after the *predictor-step* (eqns. A.2 and 2.20) to obtain the *intermediate* velocities and after the *corrector-step* (eqn. A.5 and 2.24).

Solution matching at *h-interfaces* for the ψ values

The eqns. A.8 and A.9 can be rewritten for the *Poisson* system as

$$\psi_c = \psi_R + \sum_{j=1}^{N_R} \frac{m_j}{\rho_j}\,(\psi_j - \psi_R)\,\nabla W_{Rj}(\mathbf{x}_c - \mathbf{x}_R), \ c = 1, .., N_{hc} \tag{A.14}$$

$$\psi_i = \psi(\mathbf{x}_{pqr}) - \frac{\sum_f \psi_f A_f}{V}, \ i = 1, .., N_{HI} \tag{A.15}$$

Differently from the velocity system, the values of the *effective* particles (R and its neighboring particles j) in eqn. A.14 are unknowns. Likewise, in eqn. A.15 ψ_f is an approximation of the *pseudo-pressure* at the cell faces obtained using the values of the surrounding cells that are unknowns (both for the *effective FVM* and for the *interface FVM* cells). Therefore, eqns. A.14 and A.15 must be solved simultaneously with the *Poisson* eqns. 2.21 and A.3. As a result, the global system that must be solved is made of the N_{ec} and N_e Pressure Poisson equations (where N_{ec} and N_e are the numbers of the *effective* cells and particles in the computational domain, respectively) and N_{hc} and N_{HI} Taylor series expansions (related to the *hybrid interface* cells and particle, respectively).

The inflow/outflow procedure through h-interfaces

The *h-interface triangles* (as explained for the outflow boundaries in Chap. 3) can be freely gone through by the particles (thus "virtually" entering into the *FVM-domain*). Whenever it occurs, these particles must be excluded from the computation and are included in a

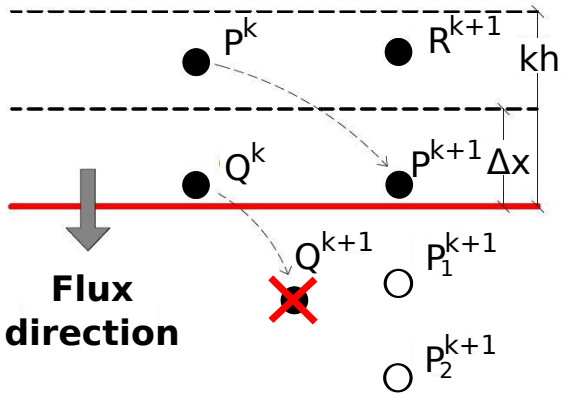


Figure A6: *Coupled FVM-SPH*. Particle leaving the *SPH-domain*. Full and empty black circles: *effective* and HI particles. Taken from: Napoli et al. (2016), 680, fig. 5.

storage list. In Fig. A6 the particle Q during the $(k+1)$-th time step goes through the *h-interface* and it is deactivated. The distance of the particle P from the *h-interface* at the time level $(k+1)$ becomes lower than Δx and the HI particles P_1 and P_2 are thus generated.

On the other hand, when the velocities near the *h-interface* are directed from the *FVM-* to the *SPH-domain*, new particles must be released to avoid emptying of the latter domain and to ensure global mass conservation. In order to save the memory avoiding a continuous increase of the particle number, the index of the new particles is taken from the aforementioned *storage list*.

A similar procedure explained for the new particles generated through *inflow* boundaries and *block interfaces* (see Chap. 3 and Chap. 4) is used for the inlet at the *h-interface*. Specifically, a new particle is released whenever no effective particles are found in the circular conical region (*scan region*) with vertex on an effective particle lying at a distance from the *h-interface* between Δx and kh and axis normal to the *h-interface*, as it is shown in Fig. A7.a. The position of the new particle (indicated with the symbol S) is shown in Fig. A7.b, together with its HI particles.

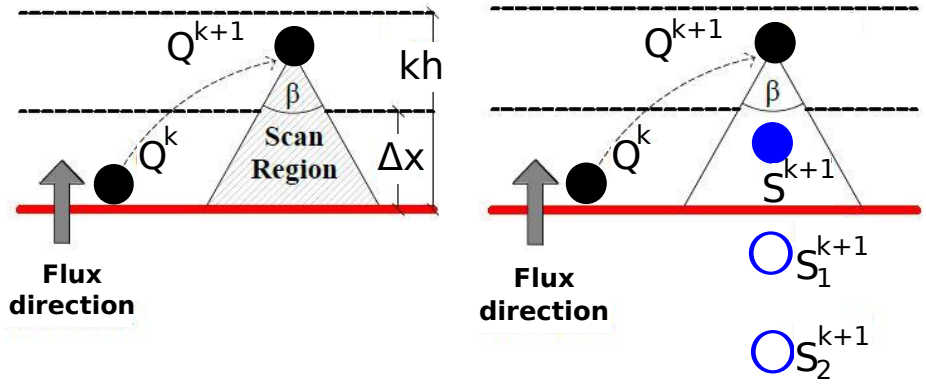
A2 Results and discussion

A2.1 Test case 1: Lid-driven cavity

The *lid-driven cavity* problem has been considered to show the performance of the *Coupled FVM-SPH* model in $3D$ viscous flows. The domain is made of a cubic box of side $d = 1\ m$, in which the flow is driven by the top face sliding in its own plane with a velocity u_s of $0.01\ m/s$. The fluid kinematic viscosity is set to $10^{-4}\ \ m^2/s$, thus obtaining a *Reynolds number* $u_s d/\nu$ equal to 100. *Adherence BCs* are applied at all the walls.

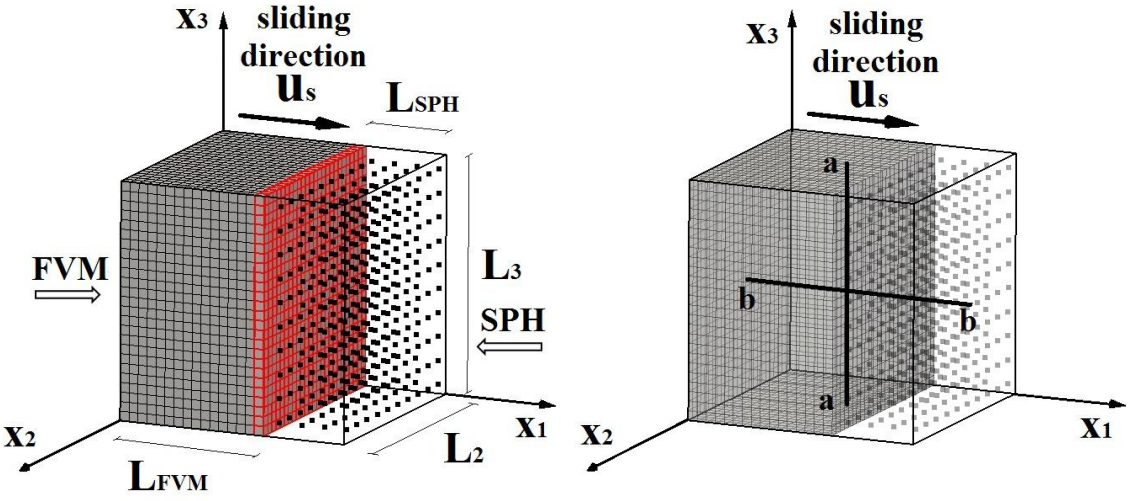
The Fig. A8 shows the computational domain partitioned in the *SPH-domain* and *FVM-domain* with the plane $x_1 = L_{FVM}$ normal to the sliding direction. A Cartesian grid of cubic cells with side length equal to $0.04\ m$ is used to discretize the *FVM-domain* (box on the left in the figure) into 16 x 25 x 25 elements in the x_1, x_2 and x_3 directions, respectively. In the *SPH-domain* the *smoothing length* has been set equal to the *FVM* grid spacing, with a *smoothing length* $h = 0.02\ m$, resulting in 45 000 particles.

The results of Albensoeder and Kuhlmann (2005) have been used for comparison. The



(a) (b)

Figure A7: *Coupled FVM-SPH*. Release of new particle through *h-interface*. Full black circles: particle Q at time k and $k+1$. a) Check inside the *scan region*; b) Release new particle scheme. Full blue circle: new *effective* particle S; empty blue circles: HI particles generated by S.



(a) (b)

Figure A8: *Coupled FVM-SPH*. Sketch of the lid-driven cavity. *a)* FVM cells and Lagrangian particles. The *interface* cells are indicated in red. $L_{FVM} = 0.64$ m; $L_{SPH} = 0.36$ m; $L_2 = 1$ m; $L_3 = 1$ m; b) vertical (a-a) and horizontal (b-b) axes. Taken from: Napoli et al. (2016), 685, fig. 13.

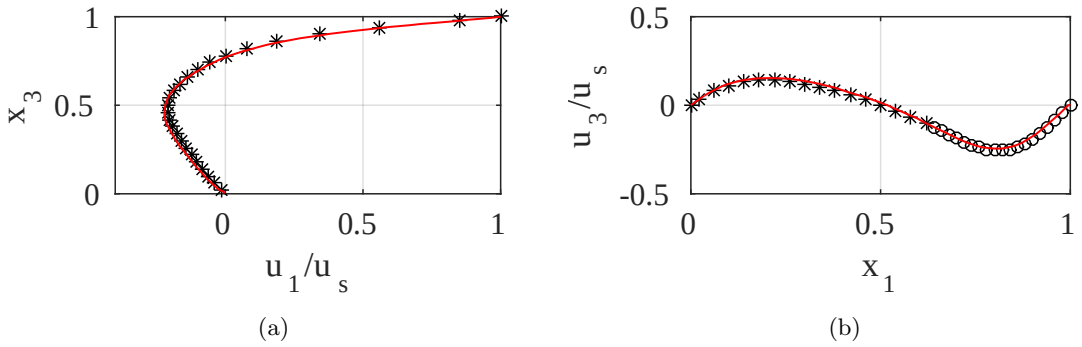


Figure A9: *Coupled FVM-SPH*. Lid-driven cavity results. Non-dimensional velocity profiles along the vertical (a-a) and horizontal (b-b) axes; a) u_1/u_s profile along the (a-a) axis; b) u_3/u_s profile along the (b-b) axis. Continuous red lines: reference data from Albensoeder and Kuhlmann (2005); stars: FVM results; circles: SPH results.

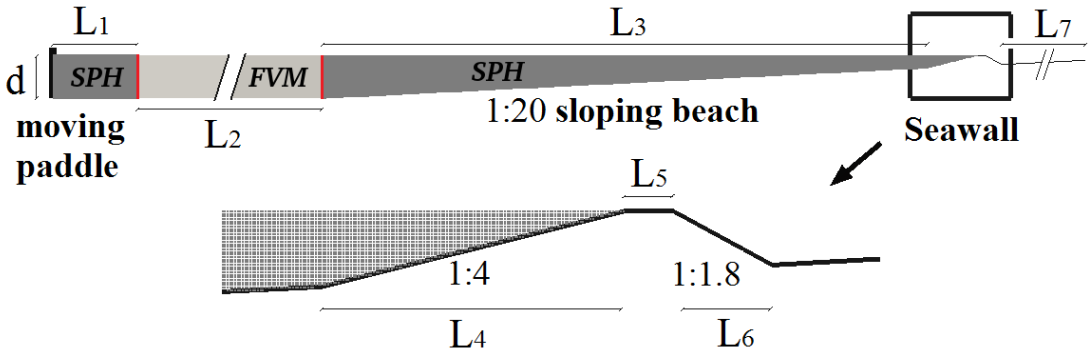


Figure A10: *Coupled FVM-SPH*. Sketch of the solitary wave channel. Bold red lines: *h-interfaces*. $d = 0.255$ m; $L_1 = 0.5$ m; $L_2 = 9.5$ m; $L_3 = 3.6$ m; $L_4 = 0.097$ m; $L_5 = 0.048$ m; $L_6 = 0.3$ m; $L_7 = 5.955$ m. Taken from: Napoli et al. (2016), 689, fig. 19.

horizontal (u_1) and vertical (u_3) velocity profiles along the axes (a-a) and (b-b) (see Fig. A8.b), respectively, are plotted in Fig. A9, together with the results of Albensoeder and Kuhlmann (2005). While the axis (a-a) entirely lies into the *FVM-domain*, (b-b) crosses the interface, thus allowing to show the matching of the *FVM* and *SPH* solutions. In both cases the velocities are made non-dimensional with the lid velocity u_s and an excellent agreement is obtained with the reference results.

A2.2 Test case 2: Solitary wave run-up and overtopping on a seawall

In order to show the performance of the *Coupled FVM-SPH* with free-surface flow with moving grid, the generation and propagation of a solitary wave in a channel and the run-up and overtopping on a seawall have been considered. The Fig. A10 shows the channel scheme where a seawall is placed on the beach to reduce the run-up. The laboratory experimental measures of Lin et al. (2012) have been used for comparison. In the first half of the channel the solitary wave propagates over a horizontal bottom, while a $1:20$ sloping beach is placed in the second half, with a trapezoidal seawall whose geometrical features are shown in the figure.

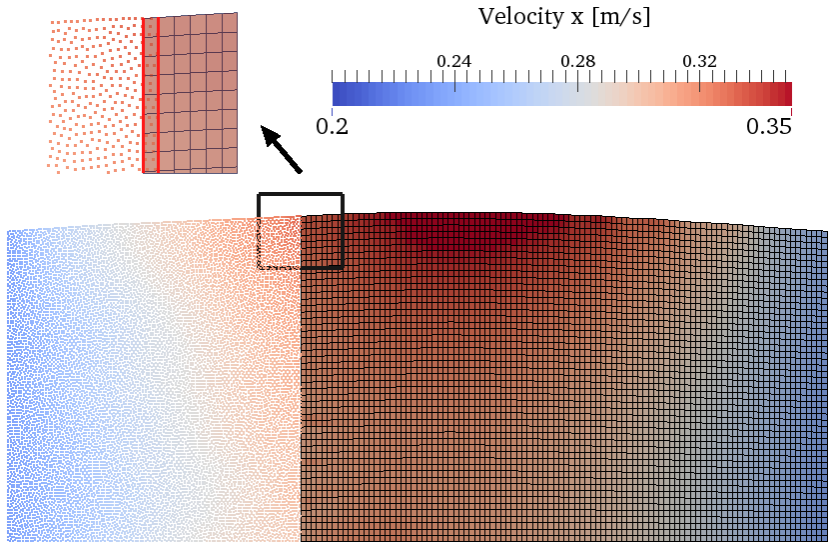


Figure A11: *Coupled FVM-SPH*. First *SPH-domain* and upstream portion of the *FMV-domain* separated by the first vertical interface. The color scale indicates the streamwise velocity in both the domains. In the detail of the free-surface region the bold red lines indicate the portion of the *FVM-domain* covered with HI particles. Taken from: Napoli et al. (2016), 689, fig. 20.

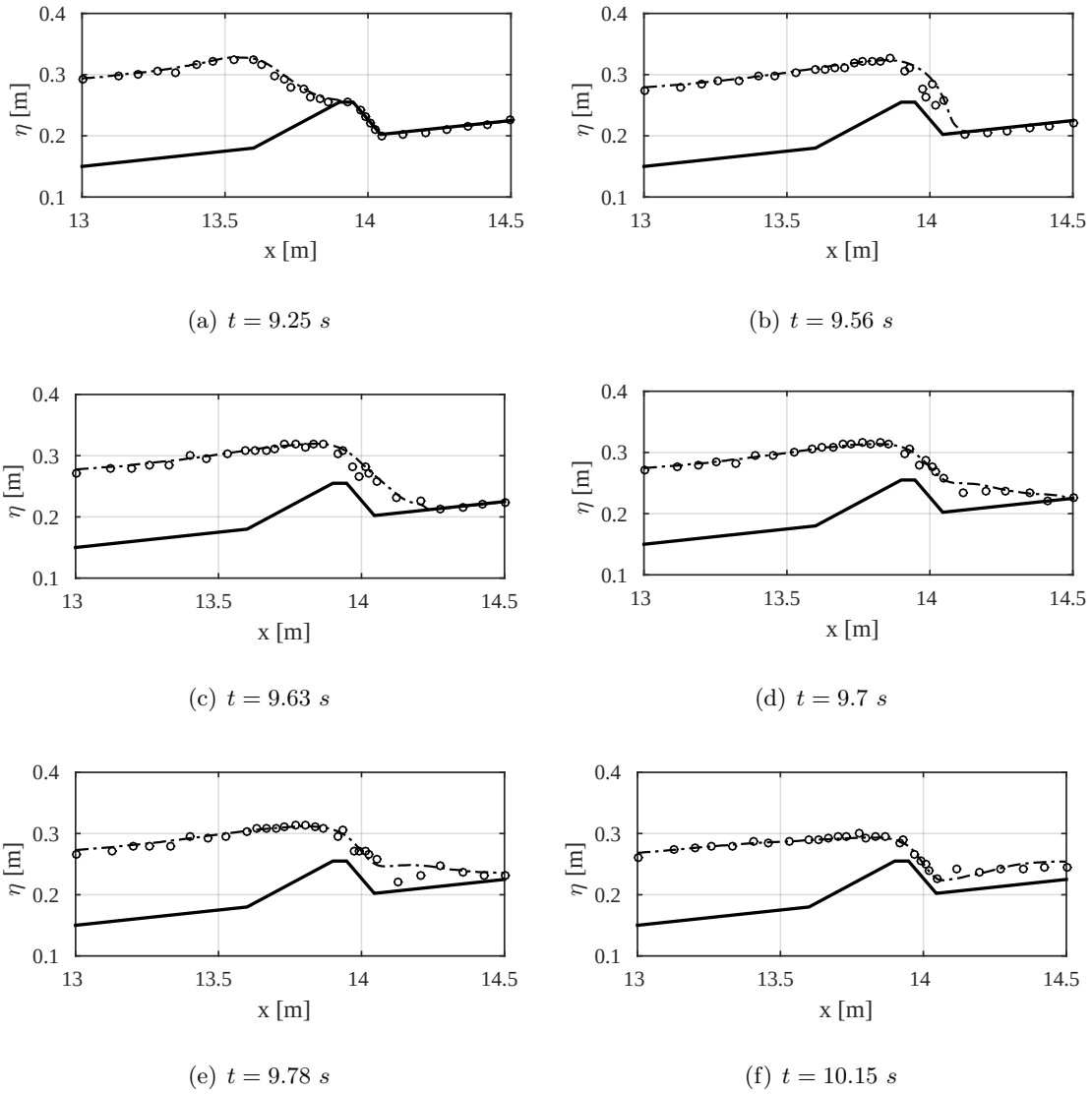
Two vertical *h-interfaces* (red lines in the figure) segregate an intermediate *FVM-domain* from an upstream and a downstream *SPH-domain*. The first *SPH-domain* is close to the wave-generating paddle that is a moving boundary, whilst the second covers the sloping beach region containing the seawall. The *FVM* is employed to propagate the wave over the horizontal bottom, while the *SPH* method is used, in addition to the paddle region, in the wave run-up/overtopping portion of the domain.

The still water depth d is set to 0.255 m as in the experiment, while the solitary wave is generated by the vertical paddle, moving in the x_1 direction according to the law proposed by Goring (1978) to obtain an offshore wave height of 0.23 d.

The *FVM* grid is made of 1900 x 51 (in the streamwise and vertical directions, respectively) square cells of side 0.005 m. During the simulation the grid is modified at each time step updating the cell heights according to the free-surface level changes, as obtained from eqn. A.6.

In both the *SPH-domains* the *smoothing length* is set to $h = 0.0025\ m$ (half the cell side). The number of *effective* particles in the first *SPH-domain* (neighboring the moving paddle) is 20 400, while 97 000 particles are used in the run-up region. *Free-slip* boundary conditions are applied at the walls (moving paddle and channel bottom). The $2D$ results are obtained considering plane flows with one cell only (in the *FVM-domain*) and one single layer of particles (in the *SPH-domains*) in the direction normal to the plane where *periodic BCs* are imposed.

The Fig. A11 shows a portion of the computational domain near the first *h-interface* during the passage of the solitary wave; the colors indicate the streamwise velocity component. In the figure it is clearly seen the perfect matching of the free-surface levels and of the streamwise velocity at the *h-interface*. The free-surface levels (η) near the seawall at different instants are plotted in Fig. A12. The numerical results are in excellent agreement with the reference experimental data, showing that the paddle-generated wave correctly



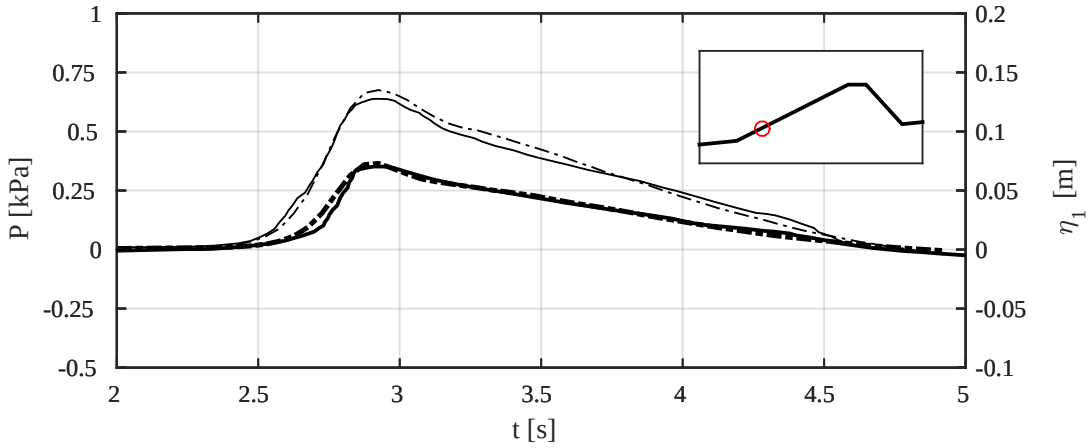
(a) $t = 9.25\ s$ (b) $t = 9.56\ s$

(c) $t = 9.63\ s$ (d) $t = 9.7\ s$

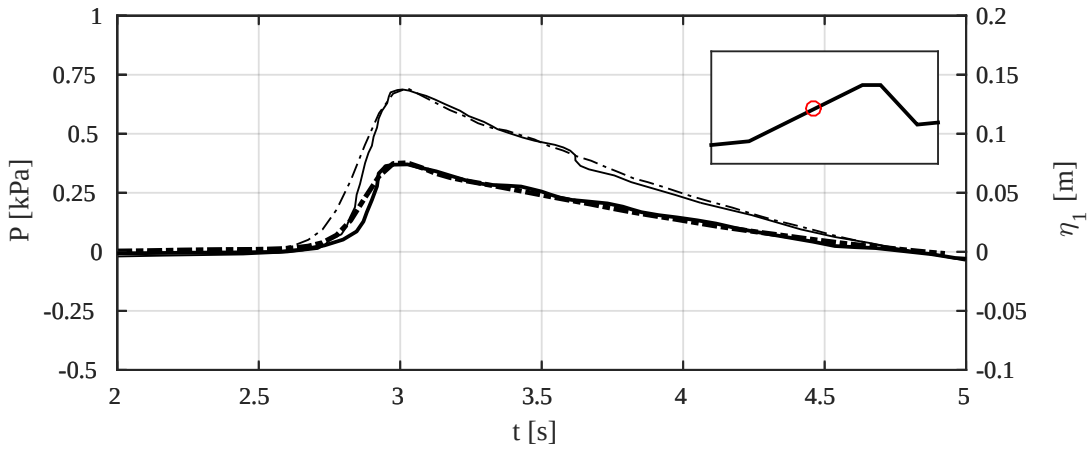
(e) $t = 9.78\ s$ (f) $t = 10.15\ s$

Figure A12: *Coupled FVM-SPH*. Free-surface evolution at different time instants close to the seawall. Bold line: channel bottom. Dashed line: FVM-SPH coupling results; circles: experimental data of Lin et al. (2012). Taken from: Napoli et al. (2016), 690, fig. 21.

propagates through the *SPH-* and *FVM-domains*, without significant perturbations due to the passage through the two *h-interfaces*. The Fig. A13 shows the time evolutions of the bottom dynamic pressure and of the free-surface level at two channel cross-sections on the seawall. The very good agreement of the numerical results with the experimental data confirms the optimal performance of the *Coupled FVM-SPH* method.



(a) $x = 13.63\ m$



(b) $x = 13.77\ m$

Figure A13: *Coupled FVM-SPH.* Dynamic pressure and free-surface level time evolution at two cross-sections (a) and (b) whose position on the seawall is indicated with the red circles. Dashed and continuous thin lines: numerical and experimental dynamic pressure. Dashed and continuous bold lines: numerical and experimental free-surface levels ($\eta_1 = \eta - z_b$ with z_b the bottom level). Taken from: Napoli et al. (2016), 691, fig. 22.

Publications

The publications during the *PhD* program are listed as follows.

1. Enrico Napoli, Mauro De Marchis, Chiara Gianguzzi, Barbara Milici, Alessandra Monteleone. *A coupled Finite Volume–Smoothed Particle Hydrodynamics method for incompressible flows.* In: Comput. Methods Appl. Mech. Engrg. 310 (2016), pp. 674–693.
   https://doi.org/10.1016/j.cma.2016.07.034

2. Alessandra Monteleone, Massimiliano Monteforte, Enrico Napoli. *Inflow/outflow pressure boundary conditions for smoothed particle hydrodynamics simulations of incompressible flows.* In: Computers and Fluids 159 (2017), pp. 9–22.
   https://doi.org/10.1016/j.compfluid.2017.09.011

3. Alessandra Monteleone, Mauro De Marchis, Barbara Milici, Enrico Napoli. *A multi-domain approach for smoothed particle hydrodynamics simulations of highly complex flows.* In: Comput. Methods Appl. Mech. Engrg.
   https://doi.org/10.1016/j.cma.2018.06.029

Bibliography

Akenine-Möllser, T. (2001). "Fast 3D triangle-box overlap testing". In: *Journal of graphics tools* 6.1, pp. 29–33.

Albensoeder, S. and H.C. Kuhlmann (2005). "Accurate three-dimensional lid-driven cavity flow". In: *Journal of Computational Physics* 206.2, pp. 536–558.

Alemu, Y. and D. Bluestein (2007). "Flow-induced platelet activation and damage accumulation in a mechanical heart valve: numerical studies". In: *Artificial organs* 31.9, pp. 677–688.

Aneurisk-Team (2012). *Aneurisk repository*. Web Site. URL: http://ecm2.mathcs.emory.edu/aneuriskweb.

Appanaboyina, S., F. Mut, R. Löhner, C.M. Putman, and J.R. Cebral (2008). "Computational fluid dynamics of stented intracranial aneurysms using adaptive embedded unstructured grids". In: *International Journal for Numerical Methods in Fluids* 57.5, pp. 475–493.

Augsburger, L., P. Reymond, D.A. Rufenacht, and N. Stergiopulos (2011). "Intracranial stents being modeled as a porous medium: flow simulation in stented cerebral aneurysms". In: *Annals of biomedical engineering* 39.2, pp. 850–863.

Badry, A., R. Elshafey, and M. Khalil (2014). "Detection, characterization and endovascular therapy planning of intracranial aneurysms with 16-channel multidetector row CT angiography". In: *The Egyptian Journal of Radiology and Nuclear Medicine* 45.1, pp. 151–158.

Barcarolo, D.A., D. Le Touzé, G. Oger, and F. De Vuyst (2014). "Adaptive particle refinement and derefinement applied to the smoothed particle hydrodynamics method". In: *Journal of Computational Physics* 273, pp. 640–657.

Becske, T., D.F. Kallmes, I. Saatci, C.G. McDougall, I. Szikora, G. Lanzino, C.J. Moran, H.H. Woo, D.K. Lopes, A.L. Berez, et al. (2013). "Pipeline for uncoilable or failed aneurysms: results from a multicenter clinical trial". In: *Radiology* 267.3, pp. 858–868.

Benz, W. (1988). "Applications of smooth particle hydrodynamics (SPH) to astrophysical problems". In: *Computer Physics Communications* 48.1, pp. 97–105.

Bisseling, R.H. (2004). *Parallel Scientific Computation: A structured approach using BSP and MPI*. Oxford University Press on Demand.

Bluestein, D., L. Niu, R.T. Schoephoerster, and M.K. Dewanjee (1997). "Fluid mechanics of arterial stenosis: relationship to the development of mural thrombus". In: *Annals of biomedical engineering* 25.2, p. 344.

Bonet, J. and S. Kulasegaram (2000). "Correction and stabilization of smooth particle hydrodynamics methods with applications in metal forming simulations". In: *International journal for numerical methods in engineering* 47.6, pp. 1189–1214.

Bouscasse, B., A. Colagrossi, S. Marrone, and M. Antuono (2013a). "Nonlinear water wave interaction with floating bodies in SPH". In: *Journal of Fluids and Structures* 42, pp. 112–129.

Bouscasse, B., S. Marrone, A. Colagrossi, and A. Di Mascio (2013b). "Multi-purpose interfaces for coupling SPH with other solvers". In: *Proceedings of the 8th International SPHERIC Workshop*.

Boussel, L., V. Rayz, C. McCulloch, A. Martin, G. Acevedo-Bolton, M. Lawton, R. Higashida, W.S. Smith, W.L. Young, and D. Saloner (2008). "Aneurysm growth occurs at region of low wall shear stress: patient-specific correlation of hemodynamics and growth in a longitudinal study". In: *Stroke* 39.11, pp. 2997–3002.

Briganti, F., G. Leone, M. Napoli, W. Lauriola, F. Florio, and F. Maiuri (2015). "Early fatal hemorrhage after endovascular treatment of a giant aneurysm with flow diverter device and coils". In: *Clinical neuroradiology* 25.2, pp. 201–205.

Brown C.H., III, R.F. Lemuth, J.D. Hellums, L.B. Leverett, and C.P. Alfrey (1975). "Response of human platelets to shear stress". In: *ASAIO Journal* 21.1, pp. 35–39.

Caballero, A., W. Mao, L. Liang, J. Oshinski, C. Primiano, R. McKay, S. Kodali, and W. Sun (2017). "Modeling Left Ventricular Blood Flow Using Smoothed Particle Hydrodynamics". In: *Cardiovascular engineering and technology* 8.4, pp. 465–479.

Calamante, F., P.J. Yim, and J.R. Cebral (2003). "Estimation of bolus dispersion effects in perfusion MRI using image-based computational fluid dynamics". In: *Neuroimage* 19.2, pp. 341–353.

Castro, M.A, C.M. Putman, and J.R. Cebral (2006). "Patient-specific computational fluid dynamics modeling of anterior communicating artery aneurysms: a study of the sensitivity of intra-aneurysmal flow patterns to flow conditions in the carotid arteries". In: *American Journal of Neuroradiology* 27.10, pp. 2061–2068.

Cebral, J.R., M.A. Castro, J.E. Burgess, R.S. Pergolizzi, M.J. Sheridan, and C.M. Putman (2005). "Characterization of cerebral aneurysms for assessing risk of rupture by using patient-specific computational hemodynamics models". In: *American Journal of Neuroradiology* 26.10, pp. 2550–2559.

Cebral, J.R., M.A. Castro, C.M. Putman, and N. Alperin (2008). "Flow–area relationship in internal carotid and vertebral arteries". In: *Physiological measurement* 29.5, p. 585.

Cebral, J.R. and R. Lohner (2005). "Efficient simulation of blood flow past complex endovascular devices using an adaptive embedding technique". In: *IEEE transactions on medical imaging* 24.4, pp. 468–476.

Cebral, J.R., F. Mut, J. Weir, and C. Putman (2011). "Quantitative characterization of the hemodynamic environment in ruptured and unruptured brain aneurysms". In: *American Journal of Neuroradiology* 32.1, pp. 145–151.

Chason, J.L. (1958). "Berry aneurysms of the circle of Willis: results of a planned autopsy study." In: *Neurology* 8, pp. 41–44.

Chong, K., C. Jiang, D. Ram, A. Santhanam, D. Terzopoulos, P. Benharash, E. Dutson, J. Teran, and J.D. Eldredge (2017). "Visualization of vascular injuries in extremity trauma". In: *Medical & biological engineering & computing* 55.9, pp. 1709–1718.

Chorin, A.J. (1968). "Numerical solution of the Navier-Stokes equations". In: *Mathematics of computation* 22.104, pp. 745–762.

Chow, A.D., B.D. Rogers, S.J. Lind, and P.K. Stansby (2018). "Incompressible SPH (ISPH) with fast Poisson solver on a GPU". In: *Computer Physics Communications* 226, pp. 81–103.

Chui, Y.P. and P.A. Heng (2010). "A meshless rheological model for blood-vessel interaction in endovascular simulation". In: *Progress in biophysics and molecular biology* 103.2-3, pp. 252–261.

Colagrossi, A. and M. Landrini (2003). "Numerical simulation of interfacial flows by smoothed particle hydrodynamics". In: *Journal of computational physics* 191.2, pp. 448–475.

Colin, F., R. Egli, and F.Y. Lin (2006). "Computing a null divergence velocity field using smoothed particle hydrodynamics". In: *Journal of Computational Physics* 217.2, pp. 680–692.

Cowan J., John A., J. Ziewacz, J.B. Dimick, Gilbert R. Upchurch J., and B.G. Thompson (2007). "Use of endovascular coil embolization and surgical clip occlusion for cerebral artery aneurysms". In:

Crespo, A.J.C., J.M. Domínguez, B.D. Rogers, M. Gómez-Gesteira, S. Longshaw, R. Canelas, R. Vacondio, A. Barreiro, and O. García-Feal (2015). "DualSPHysics: Open-source parallel CFD solver based on Smoothed Particle Hydrodynamics (SPH)". In: *Computer Physics Communications* 187, pp. 204–216.

Cummins, S.J. and M. Rudman (1999). "An SPH projection method". In: *Journal of computational physics* 152.2, pp. 584–607.

Damiano, Robert J, Ding Ma, Jianping Xiang, Adnan H Siddiqui, Kenneth V Snyder, and Hui Meng (2015). "Finite element modeling of endovascular coiling and flow diversion enables hemodynamic prediction of complex treatment strategies for intracranial aneurysm". In: *Journal of biomechanics* 48.12, pp. 3332–3340.

De Padova, D., M. Mossa, S. Sibilla, and E. Torti (2013). "3D SPH modelling of hydraulic jump in a very large channel". In: *Journal of Hydraulic Research* 51.2, pp. 158–173.

Dempere-Marco, L., E. Oubel, M. Castro, C. Putman, A. Frangi, and J.R. Cebral (2006). "CFD analysis incorporating the influence of wall motion: application to intracranial aneurysms". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, pp. 438–445.

Dilts, G.A. (1999). "Moving-least-squares-particle hydrodynamics—I. Consistency and stability". In: *International Journal for Numerical Methods in Engineering* 44.8, pp. 1115–1155.

Domínguez, José M, Alejandro JC Crespo, Daniel Valdez-Balderas, Benedict D Rogers, and Moncho Gómez-Gesteira (2013). "New multi-GPU implementation for smoothed particle hydrodynamics on heterogeneous clusters". In: *Computer Physics Communications* 184.8, pp. 1848–1860.

Dyka, C.T., P.W. Randles, and R.P. Ingel (1997). "Stress points for tension instability in SPH". In: *International Journal for Numerical Methods in Engineering* 40.13, pp. 2325–2341.

Federico, I., S. Marrone, A. Colagrossi, F. Aristodemo, and M. Antuono (2012). "Simulating 2D open-channel flows through an SPH model". In: *European Journal of Mechanics-B/Fluids* 34, pp. 35–46.

Feigin, V.L., G.J. Rinkel, C.M. Lawes, A. Algra, D.A. Bennett, J. van Gijn, and C.S. Anderson (2005). "Risk factors for subarachnoid hemorrhage: an updated systematic review of epidemiological studies". In: *Stroke* 36.12, pp. 2773–2780.

Feldman, J. and J. Bonet (2007). "Dynamic refinement and boundary contact forces in SPH with applications in fluid flow problems". In: *International Journal for Numerical Methods in Engineering* 72.3, pp. 295–324.

Fiorella, D., H.H. Woo, F.C. Albuquerque, and P.K. Nelson (2008). "Definitive reconstruction of circumferential, fusiform intracranial aneurysms with the pipeline embolization device". In: *Neurosurgery* 62.5, pp. 1115–1121.

Ford, M.D., G.R. Stuhne, H.N. Nikolov, D.F. Habets, S.P. Lownie, D.W. Holdsworth, and D.A. Steinman (2005). "Virtual angiography for visualization and validation of computational models of aneurysm hemodynamics". In: *IEEE transactions on medical imaging* 24.12, pp. 1586–1592.

Foutrakis, G.N., H. Yonas, and R.J. Sclabassi (1999). "Saccular aneurysm formation in curved and bifurcating arteries". In: *American Journal of Neuroradiology* 20.7, pp. 1309–1317.

Geers, A.J., H.G. Morales, I. Larrabide, C. Butakoff, P. Bijlenga, and A. Frangi (2017). "Wall shear stress at the initiation site of cerebral aneurysms". In: *Biomechanics and modeling in mechanobiology* 16.1, pp. 97–115.

Gester, K., I. Lüchtefeld, M. Büsen, S.J. Sonntag, T. Linde, U. Steinseifer, and G. Cattaneo (2015). "In vitro evaluation of intra-aneurysmal, flow-diverter-induced thrombus formation: a feasibility study". In: *American Journal of Neuroradiology*.

Gingold, R.A. and J.J. Monaghan (1977). "Smoothed particle hydrodynamics: theory and application to non-spherical stars". In: *Monthly notices of the royal astronomical society* 181.3, pp. 375–389.

Goertz, L., C. Hamisch, S. Telentschak, C. Kabbasch, N. von Spreckelsen, P. Stavrinou, M. Timmer, R. Goldbrunner, G. Brinker, and B. Krischek (2018). "Impact of Aneurysm Shape on Intraoperative Rupture During Clipping of Ruptured Intracranial Aneurysms". In: *World neurosurgery* 118, e806–e812.

Gomez-Gesteira, M., B.D. Rogers, R.A. Dalrymple, and A.J.C. Crespo (2010). "State-of-the-art of classical SPH for free-surface flows". In: *Journal of Hydraulic Research* 48.S1, pp. 6–27.

Goring, D.G. (1978). "Tsunamis–the propagation of long waves onto a shelf". In:

Gray, J.P., J.J. Monaghan, and R.P. Swift (2001). "SPH elastic dynamics". In: *Computer methods in applied mechanics and engineering* 190.49-50, pp. 6641–6662.

Gropp, W., E. Lusk, N. Doss, and A. Skjellum (1996). "A high-performance, portable implementation of the MPI message passing interface standard". In: *Parallel computing* 22.6, pp. 789–828.

Guo, X., B.D. Rogers, S. Lind, and P.K. Stansby (2018). "New massively parallel scheme for Incompressible Smoothed Particle Hydrodynamics (ISPH) for highly nonlinear and distorted flow". In: *Computer Physics Communications* 233, pp. 16–28.

Hansen, K.B., A. Arzani, and S.C. Shadden (2015). "Mechanical platelet activation potential in abdominal aortic aneurysms". In: *Journal of biomechanical engineering* 137.4, p. 041005.

Hellums, J.D. (1994). "1993 Whitaker Lecture: biorheology in thrombosis research". In: *Annals of biomedical engineering* 22.5, pp. 445–455.

Hendrikse, J., A.F. van Raamt, Y. van der Graaf, W.PTM Mali, and J. van der Grond (2005). "Distribution of cerebral blood flow in the circle of Willis". In: *Radiology* 235.1, pp. 184–189.

Hirschler, M., P. Kunz, M. Huber, F. Hahn, and U. Nieken (2016). "Open boundary conditions for ISPH and their application to micro-flow". In: *Journal of Computational Physics* 307, pp. 614–633.

Holme, P.A., U. Ørvim, M.J.A.G. Hamers, N.O. Solum, F.R. Brosstad, R.M. Barstad, and K.S. Sakariassen (1997). "Shear-induced platelet activation and platelet microparticle

formation at blood flow conditions as in arteries with a severe stenosis". In: *Arteriosclerosis, thrombosis, and vascular biology* 17.4, pp. 646–653.

Hosseini, S.M. and J.J. Feng (2011). "Pressure boundary conditions for computing incompressible flows with SPH". In: *Journal of Computational physics* 230.19, pp. 7473–7487.

Hu, W., W. Pan, M. Rakhsha, Q. Tian, H. Hu, and D. Negrut (2017). "A consistent multi-resolution smoothed particle hydrodynamics method". In: *Computer Methods in Applied Mechanics and Engineering* 324, pp. 278–299.

Janiga, G., L. Daróczy, P. Berg, D. Thévenin, M. Skalej, and O. Beuing (2015). "An automatic CFD-based flow diverter optimization principle for patient-specific intracranial aneurysms". In: *Journal of biomechanics* 48.14, pp. 3846–3852.

Jeong, W. and K. Rhee (2012). "Hemodynamics of cerebral aneurysms: computational analyses of aneurysm progress and treatment". In: *Computational and mathematical methods in medicine* 2012.

Jianming, W., G. Na, and G. Wenjun (2010). "Abrasive waterjet machining simulation by SPH method". In: *The International Journal of Advanced Manufacturing Technology* 50.1-4, pp. 227–234.

Jou, L-D., D.H. Lee, H. Morsi, and M.E. Mawad (2008). "Wall shear stress on ruptured and unruptured intracranial aneurysms at the internal carotid artery". In: *American Journal of Neuroradiology* 29.9, pp. 1761–1767.

Jou, L.D., C.M. Quick, W.L. Young, M.T. Lawton, R. Higashida, A. Martin, and D. Saloner (2003). "Computational approach to quantifying hemodynamic forces in giant cerebral aneurysms". In: *American Journal of Neuroradiology* 24.9, pp. 1804–1810.

Kamath, Sylvia (1981). "Observations on the length and diameter of vessels forming the circle of Willis." In: *Journal of anatomy* 133.Pt 3, p. 419.

Karmonik, C., C. Yen, O. Diaz, R. Klucznik, R.G. Grossman, and G. Benndorf (2010). "Temporal variations of wall shear stress parameters in intracranial aneurysms-importance of patient-specific inflow waveforms for CFD calculations". In: *Acta neurochirurgica* 152.8, pp. 1391–1398.

Karmonik, C., C. Yen, R.G. Grossman, R. Klucznik, and G. Benndorf (2009). "Intraaneurysmal flow patterns and wall shear stresses calculated with computational flow dynamics in an anterior communicating artery aneurysm depend on knowledge of patient-specific inflow rates". In: *Acta neurochirurgica* 151.5, pp. 479–485.

Khayyer, A., H. Gotoh, and Y. Shimizu (2017). "Comparative study on accuracy and conservation properties of two particle regularization schemes and proposal of an optimized particle shifting scheme in ISPH context". In: *Journal of Computational Physics* 332, pp. 236–256.

Khorasanizade, S. and J.M.M. Sousa (2016a). "A two-dimensional Segmented Boundary Algorithm for complex moving solid boundaries in Smoothed Particle Hydrodynamics". In: *Computer Physics Communications* 200, pp. 66–75.

— (2016b). "An innovative open boundary treatment for incompressible SPH". In: *International Journal for Numerical Methods in Fluids* 80.3, pp. 161–180.

Kim, T., A.Y. Cheer, and H.A. Dwyer (2004). "A simulated dye method for flow visualization with a computational model for blood flow". In: *Journal of biomechanics* 37.8, pp. 1125–1136.

Kiselev, R., K. Orlov, A. Dubovoy, V. Berestov, A. Gorbatykh, D. Kislitsin, T. Shayakhmetov, A. Tasenko, P. Seleznev, N. Strelnikov, et al. (2018). "Flow diversion versus parent artery occlusion with bypass in the treatment of complex intracranial aneurysms: Im-

mediate and short-term outcomes of the randomized trial". In: *Clinical neurology and neurosurgery* 172, pp. 183–189.

Ku, David N (1997). "Blood flow in arteries". In: *Annual review of fluid mechanics* 29.1, pp. 399–434.

Kundu, Pijush K, Ira M Cohen, and David R Dowling (2016). *Fluid mechanics*.

Kunz, P., M. Hirschler, M. Huber, and U. Nieken (2016). "Inflow/outflow with Dirichlet boundary conditions for pressure in ISPH". In: *Journal of Computational Physics* 326, pp. 171–187.

Lang, Johannes (2001). *Skull base and related structures: atlas of clinical anatomy*. Schattauer Verlag.

Lee, E-S., C. Moulinec, Rui Xu, D. Violeau, D. Laurence, and P. Stansby (2008). "Comparisons of weakly compressible and truly incompressible algorithms for the SPH mesh free particle method". In: *Journal of computational Physics* 227.18, pp. 8417–8436.

Lei, M., D.P. Giddens, S.A. Jones, F. Loth, and H. Bassiouny (2001). "Pulsatile flow in an end-to-side vascular graft model: comparison of computations with experimental data". In: *Journal of Biomechanical Engineering* 123.1, pp. 80–87.

Leroy, A., D. Violeau, M. Ferrand, L. Fratter, and A. Joly (2016). "A new open boundary formulation for incompressible SPH". In: *Computers & Mathematics with Applications* 72.9, pp. 2417–2432.

Lin, T.C., K.S. Hwang, S.C. Hsiao, and R.Y. Yang (2012). "An experimental observation of a solitary wave impingement, run-up and overtopping on a seawall". In: *Journal of Hydrodynamics* 24.1, pp. 76–85.

Lind, S.J., R. Xu, P.K. Stansby, and B.D. Rogers (2012). "Incompressible smoothed particle hydrodynamics for free-surface flows: A generalised diffusion-based algorithm for stability and validations for impulsive flows and propagating waves". In: *Journal of Computational Physics* 231.4, pp. 1499–1523.

Lipari, G. and E. Napoli (2008). "The impacts of the ALE and hydrostatic-pressure approaches on the energy budget of unsteady free-surface flows". In: *Computers & fluids* 37.6, pp. 656–673.

Liu, G.R. and M.B. Liu (2003). *Smoothed particle hydrodynamics: a meshfree particle method*. World Scientific.

Liu, M.B. and G.R. Liu (2010). "Smoothed particle hydrodynamics (SPH): an overview and recent developments". In: *Archives of computational methods in engineering* 17.1, pp. 25–76.

López, Y. R., D. Roose, and C.R. Morfa (2013). "Dynamic particle refinement in SPH: application to free surface flow and non-cohesive soil simulations". In: *Computational Mechanics* 51.5, pp. 731–741.

Lucy, L.B. (1977). "A numerical approach to the testing of the fission hypothesis". In: *The astronomical journal* 82, pp. 1013–1024.

Mao, W., K. Li, and W. Sun (2016). "Fluid–structure interaction study of transcatheter aortic valve dynamics using smoothed particle hydrodynamics". In: *Cardiovascular engineering and technology* 7.4, pp. 374–388.

Marrone, S., M. Antuono, A. Colagrossi, G. Colicchio, D. Le Touzé, and G. Graziani (2011). "δ-SPH model for simulating violent impact flows". In: *Computer Methods in Applied Mechanics and Engineering* 200.13-16, pp. 1526–1542.

Marshall, I., P. Papathanasopoulou, and K. Wartolowska (2004). "Carotid flow rates and flow division at the bifurcation in healthy volunteers". In: *Physiological measurement* 25.3, p. 691.

Marzo, A., P. Singh, I. Larrabide, A. Radaelli, S. Coley, M. Gwilliam, I.D. Wilkinson, P. Lawford, P. Reymond, U. Patel, et al. (2011). "Computational hemodynamics in cerebral aneurysms: the effects of modeled versus measured boundary conditions". In: *Annals of biomedical engineering* 39.2, pp. 884–896.

Meng, H., V.M. Tutino, J. Xiang, and A. Siddiqui (2013). "High WSS or low WSS? Complex interactions of hemodynamics with intracranial aneurysm initiation, growth, and rupture: toward a unifying hypothesis". In: *American Journal of Neuroradiology*.

Meng, H., Z. Wang, M. Kim, R.D. Ecker, and L.N. Hopkins (2006). "Saccular aneurysms on straight and curved vessels are subject to different hemodynamics: implications of intravascular stenting". In: *American Journal of Neuroradiology* 27.9, pp. 1861–1865.

Menichini, C. and X.Y. Xu (2016). "Mathematical modeling of thrombus formation in idealized models of aortic dissection: initial findings and potential applications". In: *Journal of mathematical biology* 73.5, pp. 1205–1226.

Meringolo, D.D., A. Colagrossi, S. Marrone, and F. Aristodemo (2017). "On the filtering of acoustic components in weakly-compressible SPH simulations". In: *Journal of Fluids and Structures* 70, pp. 1–23.

Mittal, R. and G. Iaccarino (2005). "Immersed boundary methods". In: *Annu. Rev. Fluid Mech.* 37, pp. 239–261.

Miura, Y., F. Ishida, Y. Umeda, H. Tanemura, H. Suzuki, S. Matsushima, S. Shimosaka, and W. Taki (2013). "Low wall shear stress is independently associated with the rupture status of middle cerebral artery aneurysms". In: *Stroke* 44.2, pp. 519–521.

Monaghan, J.J. (1992). "Smoothed particle hydrodynamics". In: *Annual review of astronomy and astrophysics* 30.1, pp. 543–574.

— (1994). "Simulating free surface flows with SPH". In: *Journal of computational physics* 110.2, pp. 399–406.

— (2000). "SPH without a tensile instability". In: *Journal of Computational Physics* 159.2, pp. 290–311.

— (2012). "Smoothed particle hydrodynamics and its diverse applications". In: *Annual Review of Fluid Mechanics* 44, pp. 323–346.

Monteleone, A., M. De Marchis, B. Milici, and E. Napoli (2018). "A multi-domain approach for smoothed particle hydrodynamics simulations of highly complex flows". In: *Computer Methods in Applied Mechanics and Engineering*.

Monteleone, A., M. Monteforte, and E. Napoli (2017). "Inflow/outflow pressure boundary conditions for smoothed particle hydrodynamics simulations of incompressible flows". In: *Computers & Fluids* 159, pp. 9–22.

Morales, H.G., I. Larrabide, A.J. Geers, M.L. Aguilar, and A. Frangi (2013). "Newtonian and non-Newtonian blood flow in coiled cerebral aneurysms". In: *Journal of biomechanics* 46.13, pp. 2158–2164.

Morley, T.P. (1969). "Giant intracranial aneurysms: diagnosis, course, and management". In: *Clin Neurosurg* 16, pp. 73–94.

Morris, J.P., P.J. Fox, and Y. Zhu (1997). "Modeling low Reynolds number incompressible flows using SPH". In: *Journal of computational physics* 136.1, pp. 214–226.

Müller, M., S. Schirm, and M. Teschner (2004). "Interactive blood simulation for virtual surgery based on smoothed particle hydrodynamics". In: *Technology and Health Care* 12.1, pp. 25–31.

Munarriz, P.M., P.A. Gómez, I. Paredes, A.M. Castaño-Leon, S. Cepeda, and A. Lagares (2016). "Basic principles of hemodynamics and cerebral aneurysms". In: *World neurosurgery* 88, pp. 311–319.

Napoli, E. (2011). "PANORMUS User's manual". In: *University of Palermo, Palermo, Italy, www.panormus_3d.org*, pp. 1–74.

Napoli, E., M. De Marchis, C. Gianguzzi, B. Milici, and A. Monteleone (2016). "A coupled finite volume–smoothed particle hydrodynamics method for incompressible flows". In: *Computer Methods in Applied Mechanics and Engineering* 310, pp. 674–693.

Napoli, E., M. De Marchis, and E. Vitanza (2015). "PANORMUS-SPH. A new Smoothed Particle Hydrodynamics solver for incompressible flows". In: *Computers & Fluids* 106, pp. 185–195.

Narayanaswamy, M., A.J.C. Crespo, M. Gómez-Gesteira, and R.A. Dalrymple (2010). "SPHysics-FUNWAVE hybrid model for coastal wave propagation". In: *Journal of Hydraulic Research* 48.S1, pp. 85–93.

Nelson, R.P. and J.C. Papaloizou (1994). "Variable smoothing lengths and energy conservation in smoothed particle hydrodynamics". In: *Monthly Notices of the Royal Astronomical Society* 270.1, pp. 1–20.

Nolin, S., V. Roubtsova, B. Morse, and T. Quach (2009). "Smoothed particle hydrodynamics hybrid model of ice-jam formation and release". In: *Canadian Journal of Civil Engineering* 36.7, pp. 1133–1143.

Oger, G., M. Doring, B. Alessandrini, and P. Ferrant (2007). "An improved SPH method: Towards higher order convergence". In: *Journal of Computational Physics* 225.2, pp. 1472–1492.

Ohta, M., S.G. Wetzel, P. Dantan, C. Bachelet, K.O. Lovblad, H. Yilmaz, P. Flaud, and D.A. Rüfenacht (2005). "Rheological changes after stenting of a cerebral aneurysm: a finite element modeling approach". In: *Cardiovascular and interventional radiology* 28.6, pp. 768–772.

Oka, S. and M. Nakai (1987). "Optimality principle in vascular bifurcation". In: *Biorheology* 24.6, pp. 737–751.

Owen, J.M., J.V. Villumsen, P.R. Shapiro, and H. Martel (1998). "Adaptive smoothed particle hydrodynamics: Methodology. II." In: *The Astrophysical Journal Supplement Series* 116.2, p. 155.

Passerini, T., L.M. Sangalli, S. Vantini, M. Piccinelli, S. Bacigaluppi, L. Antiga, E. Boccardi, P. Secchi, and A. Veneziani (2012). "An integrated statistical investigation of internal carotid arteries of patients affected by cerebral aneurysms". In: *Cardiovascular Engineering and Technology* 3.1, pp. 26–40.

Pierot, L. (2011). "Flow diverter stents in the treatment of intracranial aneurysms: Where are we?" In: *Journal of Neuroradiology* 38.1, pp. 40–46.

Quinlan, N.J., M. Basa, and M. Lastiwka (2006). "Truncation error in mesh-free particle methods". In: *International Journal for Numerical Methods in Engineering* 66.13, pp. 2064–2085.

Radaelli, A.G., L.a Augsburger, J.R. Cebral, M. Ohta, D.A. Rüfenacht, R. Balossino, G. Benndorf, D.R. Hose, A. Marzo, R. Metcalfe, et al. (2008). "Reproducibility of haemodynamical simulations in a subject-specific stented aneurysm model—a report on the Virtual Intracranial Stenting Challenge 2007". In: *Journal of biomechanics* 41.10, pp. 2069–2081.

Ramstack, J.M., L. Zuckerman, and L.F. Mockros (1979). "Shear-induced activation of platelets". In: *Journal of biomechanics* 12.2, pp. 113–125.

Raymond, J., J.C. Gentric, T.E. Darsaut, D. Iancu, M. Chagnon, A. Weill, and D. Roy (2016). "Flow diversion in the treatment of aneurysms: a randomized care trial and registry". In: *Journal of neurosurgery* 127.3, pp. 454–462.

Rayz, V.L., L. Boussel, L. Ge, J.R. Leach, A.J. Martin, M.T. Lawton, C. McCulloch, and D. Saloner (2010). "Flow residence time and regions of intraluminal thrombus deposition in intracranial aneurysms". In: *Annals of biomedical engineering* 38.10, pp. 3058–3069.

Rinkel, G.J. (2008). "Natural history, epidemiology and screening of unruptured intracranial aneurysms". In: *Revue neurologique* 164.10, pp. 781–786.

Rinkel, G.J. et al. (2001). "Subarachnoid haemorrhage: diagnosis, causes and management." In: *Brain: a journal of neurology* 124.Pt 2, pp. 249–278.

Rinkel, G.J., M. Djibuti, A. Algra, and J. Van Gijn (1998). "Prevalence and risk of rupture of intracranial aneurysms: a systematic review". In: *Stroke* 29.1, pp. 251–256.

Roman, F., E. Napoli, B. Milici, and V. Armenio (2009). "An improved immersed boundary method for curvilinear grids". In: *Computers & Fluids* 38.8, pp. 1510–1527.

Rooij, W.J. van, M.E. Sprengers, A.N. de Gast, J.P. Peluso, and M. Sluzewski (2008). "3D rotational angiography: the new gold standard in the detection of additional intracranial aneurysms". In: *American Journal of Neuroradiology* 29.5, pp. 976–979.

Rouchaud, A., W. Brinjikji, G. Lanzino, H.J. Cloft, R. Kadirvel, and D.F. Kallmes (2016). "Delayed hemorrhagic complications after flow diversion for intracranial aneurysms: a literature overview". In: *Neuroradiology* 58.2, pp. 171–177.

Saad, Y. (2003). *Iterative methods for sparse linear systems*. Vol. 82. siam.

Schäfer, M., S. Turek, F. Durst, E. Krause, and R. Rannacher (1996). "Benchmark computations of laminar flow around a cylinder". In: *Flow simulation with high-performance computers II*. Springer, pp. 547–566.

Shadden, S.C. and A. Arzani (2015). "Lagrangian postprocessing of computational hemodynamics". In: *Annals of biomedical engineering* 43.1, pp. 41–58.

Shadden, S.C. and S. Hendabadi (2013). "Potential fluid mechanic pathways of platelet activation". In: *Biomechanics and modeling in mechanobiology* 12.3, pp. 467–474.

Shahriari, S., H. Maleki, I. Hassan, and L. Kadem (2012). "Evaluation of shear stress accumulation on blood components in normal and dysfunctional bileaflet mechanical heart valves using smoothed particle hydrodynamics". In: *Journal of biomechanics* 45.15, pp. 2637–2644.

Shamloo, A., M.A. Nejad, and M. Saeedi (2017). "Fluid–structure interaction simulation of a cerebral aneurysm: Effects of endovascular coiling treatment and aneurysm wall thickening". In: *Journal of the mechanical behavior of biomedical materials* 74, pp. 72–83.

Shi, W., M. Zheng, and P.X. Liu (2017). "Virtual surgical bleeding simulation with navier-stokes equation and modified smooth particle hydrodynamics method". In: *Information and Automation (ICIA), 2017 IEEE International Conference on*. IEEE, pp. 276–281.

Shibata, K., S. Koshizuka, T. Matsunaga, and I. Masaie (2017). "The overlapping particle technique for multi-resolution simulation of particle methods". In: *Computer Methods in Applied Mechanics and Engineering* 325, pp. 434–462.

Shimogonya, Y., T. Ishikawa, Y. Imai, N. Matsuki, and T. Yamaguchi (2009). "Can temporal fluctuation in spatial wall shear stress gradient initiate a cerebral aneurysm? A proposed novel hemodynamic index, the gradient oscillatory number (GON)". In: *Journal of biomechanics* 42.4, pp. 550–554.

Shojima, M., M. Oshima, K. Takagi, R. Torii, K. Nagata, I. Shirouzu, A. Morita, and T. Kirino (2005). "Role of the bloodstream impacting force and the local pressure elevation in the rupture of cerebral aneurysms". In: *Stroke* 36.9, pp. 1933–1938.

Siddiqui, A.H., P. Kan, A.A. Abla, L. N. Hopkins, and E.I. Levy (2012). "Complications after treatment with pipeline embolization for giant distal intracranial aneurysms with or without coil embolization". In: *Neurosurgery* 71.2, E509–E513.

Skillen, A., S. Lind, P.K. Stansby, and B.D. Rogers (2013). "Incompressible smoothed particle hydrodynamics (SPH) with reduced temporal noise and generalised Fickian smoothing applied to body–water slam and efficient wave–body interaction". In: *Computer Methods in Applied Mechanics and Engineering* 265, pp. 163–173.

Soares, J.S., J. Sheriff, and D. Bluestein (2013). "A novel mathematical model of activation and sensitization of platelets subjected to dynamic stress histories". In: *Biomechanics and modeling in mechanobiology* 12.6, pp. 1127–1141.

Souto-Iglesias, A., F. Macià, L.M. González, and J.L. Cercos-Pita (2013). "On the consistency of MPS". In: *Computer Physics Communications* 184.3, pp. 732–745.

Spreng, F., D. Schnabel, A. Mueller, and P. Eberhard (2014). "A local adaptive discretization algorithm for Smoothed Particle Hydrodynamics". In: *Computational Particle Mechanics* 1.2, pp. 131–145.

Stehbens, W.E. (1972). *Pathology of the cerebral blood vessels*. CV Mosby.

Steinman, D.A., J.S. Milner, C.J. Norley, S.P. Lownie, and D.W. Holdsworth (2003). "Image-based computational simulation of flow dynamics in a giant intracranial aneurysm". In: *American Journal of Neuroradiology* 24.4, pp. 559–566.

Stock, K.W., S.G. Wetzel, P.A. Lyrer, and E.W. Radü (2000). "Quantification of blood flow in the middle cerebral artery with phase-contrast MR imaging". In: *European radiology* 10.11, pp. 1795–1800.

Swegle, J.W., D.L. Hicks, and S.W. Attaway (1995). "Smoothed particle hydrodynamics stability analysis". In: *Journal of computational physics* 116.1, pp. 123–134.

Szikora, I., G. Paál, A. Ugron, F. Nasztanovics, M. Marosfoi, Z. Berentei, Z. Kulcsar, W. Lee, I. Bojtar, and I. Nyary (2008). "Impact of aneurysmal geometry on intraaneurysmal flow: a computerized flow simulation study". In: *Neuroradiology* 50.5, pp. 411–421.

Szymanski, G. (1932). "Quelques solutions exactes des équations d'hydrody- namique du fluide visqueux dans le cas d'un tube cylindrique". In: *Journal de Mathematiques pures et Appliquees* 11, pp. 67–108.

Tafuni, A., J.M. Domínguez, R. Vacondio, and A.J.C. Crespo (2018). "A versatile algorithm for the treatment of open boundary conditions in Smoothed particle hydrodynamics GPU models". In: *Computer Methods in Applied Mechanics and Engineering* 342, pp. 604–624.

Takeda, H., S.M. Miyama, and M. Sekiya (1994). "Numerical simulation of viscous flow by smoothed particle hydrodynamics". In: *Progress of theoretical physics* 92.5, pp. 939–960.

Tambasco, M. and D.A. Steinman (2003). "Path-dependent hemodynamics of the stenosed carotid bifurcation". In: *Annals of biomedical engineering* 31.9, pp. 1054–1065.

Tan, S.K., N.S. Cheng, Y. Xie, and S. Shao (2015). "Incompressible SPH simulation of open channel flow over smooth bed". In: *Journal of Hydro-environment Research* 9.3, pp. 340–353.

Taylor, C.A., T.J.R. Hughes, and C.K. Zarins (1998). "Finite element modeling of blood flow in arteries". In: *Computer methods in applied mechanics and engineering* 158.1-2, pp. 155–196.

Taylor, J.O., R.S. Meyer, S. Deutsch, and K.B. Manning (2016). "Development of a computational model for macroscopic predictions of device-induced thrombosis". In: *Biomechanics and modeling in mechanobiology* 15.6, pp. 1713–1731.

Taylor, J.O., K.P. Witmer, T. Neuberger, B.A. Craven, R.S. Meyer, S. Deutsch, and K.B. Manning (2014). "In vitro quantification of time dependent thrombus size using magnetic resonance imaging and computational simulations of thrombus surface shear stresses". In: *Journal of biomechanical engineering* 136.7, p. 071012.

Toth, G. and R. Cerejo (2018). "Intracranial aneurysms: Review of current science and management". In: *Vascular Medicine* 23.3, pp. 276–288.

Turan, N., R.A. Heider, A.K. Roy, B.A. Miller, M.E. Mullins, D.L. Barrow, J. Grossberg, and G. Pradilla (2018). "Current perspectives in imaging modalities for the assessment of unruptured intracranial aneurysms: a comparative analysis and review". In: *World neurosurgery* 113, pp. 280–292.

Turowski, B., S. Macht, Z. Kulcsár, D. Hänggi, and W. Stummer (2011). "Early fatal hemorrhage after endovascular cerebral aneurysm treatment with a flow diverter (SILK-Stent)". In: *Neuroradiology* 53.1, pp. 37–41.

Vacondio, R., B.D. Rogers, P.K. Stansby, and P. Mignosa (2011). "SPH modeling of shallow flow with open boundaries for practical flood simulation". In: *Journal of Hydraulic Engineering* 138.6, pp. 530–541.

— (2013a). "Shallow water SPH for flooding with dynamic particle coalescing and splitting". In: *Advances in Water Resources* 58, pp. 10–23.

Vacondio, R., B.D. Rogers, P.K. Stansby, P. Mignosa, and J. Feldman (2013b). "Variable resolution for SPH: a dynamic particle coalescing and splitting scheme". In: *Computer Methods in Applied Mechanics and Engineering* 256, pp. 132–148.

Vacondio, R., B.D. Rogers, P.K Stansby, and P. Mignosa (2016). "Variable resolution for SPH in three dimensions: Towards optimal splitting and coalescing for dynamic adaptivity". In: *Computer Methods in Applied Mechanics and Engineering* 300, pp. 442–460.

Vali, A., A.A. Abla, M.T. Lawton, D. Saloner, and V.L. Rayz (2017). "Computational Fluid Dynamics modeling of contrast transport in basilar aneurysms following flow-altering surgeries". In: *Journal of biomechanics* 50, pp. 195–201.

Van der Vorst, H.A. (1992). "Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems". In: *SIAM Journal on scientific and Statistical Computing* 13.2, pp. 631–644.

Vanninen, R., H. Manninen, and A. Ronkainen (2003). "Broad-based intracranial aneurysms: thrombosis induced by stent placement". In: *American Journal of Neuroradiology* 24.2, pp. 263–266.

Walkoff, L., W. Brinjikji, A. Rouchaud, J. Caroff, and D.F. Kallmes (2016). "Comparing magnetic resonance angiography (MRA) and computed tomography angiography (CTA) with conventional angiography in the detection of distal territory cerebral mycotic and oncotic aneurysms". In: *Interventional Neuroradiology* 22.5, pp. 524–528.

Wang, P., A-M. Zhang, F. Ming, P. Sun, and H. Cheng (2019). "A novel non-reflecting boundary condition for fluid dynamics solved by smoothed particle hydrodynamics". In: *Journal of Fluid Mechanics* 860, pp. 81–114.

Wendland, H. (1995). "Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree". In: *Advances in computational Mathematics* 4.1, pp. 389–396.

White, A.C., D.A. Kumpe, C.D. Roark, D.E. Case, and J. Seinfeld (2018). "Patterns, Predictors, and Outcomes of Postprocedure Delayed Hemorrhage Following Flow Diversion for Intracranial Aneurysm Treatment". In: *World neurosurgery*.

Wiebers, D.O. (2003). "International Study of Unruptured Intracranial Aneurysms Investigators: Unruptured intracranial aneurysms: Natural history, clinical outcome, and risks of surgical and endovascular treatment". In: *Lancet* 362, pp. 103–110.

Wiebers, D.O. et al. (2003). "Unruptured intracranial aneurysms: natural history, clinical outcome, and risks of surgical and endovascular treatment". In: *The Lancet* 362.9378, pp. 103–110.

Wiebers, D.O. and R. Marsh (1998). "Unruptured intracranial aneurysms - Risk of rupture and risks of surgical intervention". In: 339.24, pp. 1725–1733.

Womersley, J.R. (1955). "Method for the calculation of velocity, rate of flow and viscous drag in arteries when the pressure gradient is known". In: *The Journal of physiology* 127.3, pp. 553–563.

Wurzinger, L.J., R. Opitz, P. Blasberg, and H. Schmid-Schönbein (1985). "Platelet and coagulation parameters following millisecond exposure to laminar shear stress". In: *Thrombosis and haemostasis* 53.02, pp. 381–386.

Xiang, J., D. Ma, K.V. Snyder, E.I. Levy, A.H. Siddiqui, and H. Meng (2014). "Increasing flow diversion for cerebral aneurysm treatment using a single flow diverter". In: *Neurosurgery* 75.3, pp. 286–294.

Xiang, J., S.K. Natarajan, M. Tremmel, D. Ma, J. Mocco, L.N. Hopkins, A.H. Siddiqui, E.I. Levy, and H. Meng (2011). "Hemodynamic–morphologic discriminants for intracranial aneurysm rupture". In: *Stroke* 42.1, pp. 144–152.

Xiong, Q., B. Li, and J. Xu (2013). "GPU-accelerated adaptive particle splitting and merging in SPH". In: *Computer Physics Communications* 184.7, pp. 1701–1707.

Xu, F., B. Xu, L. Huang, J. Xiong, Y. Gu, and M.T. Lawton (2018). "Surgical Treatment of Large or Giant Fusiform Middle Cerebral Artery Aneurysms: A Case Series". In: *World neurosurgery*.

Xu, R., P. Stansby, and D. Laurence (2009). "Accuracy and stability in incompressible SPH (ISPH) based on the projection method and a new approach". In: *Journal of computational Physics* 228.18, pp. 6703–6725.

Yeylaghi, S., B. Moa, P. Oshkai, B. Buckham, and C. Crawford (2016). "ISPH modelling of an oscillating wave surge converter using an OpenMP-based parallel approach". In: *Journal of Ocean Engineering and Marine Energy* 2.3, pp. 301–312.

Zang, Y., R.L. Street, and J.R. Koseff (1994). "A non-staggered grid, fractional step method for time-dependent incompressible Navier-Stokes equations in curvilinear coordinates". In: *Journal of Computational Physics* 114.1, pp. 18–33.

Zhang, M., Y. Li, X. Zhao, D.I. Verrelli, W. Chong, M. Ohta, and Y. Qian (2017). "Haemodynamic effects of stent diameter and compaction ratio on flow-diversion treatment of intracranial aneurysms: A numerical study of a successful and an unsuccessful case". In: *Journal of biomechanics* 58, pp. 179–186.