POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Homework report

Homework 2

Scientific computing tools for advanced mathematical modelling

Authors: Galvan Lorenzo, Leimer Saglio Caterina, Singh Randeep

Academic year: 2021-2022

---

1. Project description and goal

A stroke is a pathological condition in which the blood flow to the brain is stopped. The most common cause is the formation of a blood clot (thrombus). Most atrial thrombi form in the left atrial appendage (LAA), with greater or lesser incidence depending on the patient morphology. Many studies have investigated this variability using imaging techniques [1].
The problem can be split in three sub-problems:

1. Design a shape model able to parameterize the four LAA morphologies, using as base shape the provided template;
2. Implement a stable numerical solver, able to correctly simulate the blood flow for all shapes;
3. Define physical indicators to detect and predict the risk of ischemic stroke formation.

2. Mathematical formulation of the problem

Regarding the shape model, we decided to select as template just the boundary of the domain, ignoring the already fitted mesh. Thus, we create the meshes for the new shapes a posteriori, preserving the triangular geometry, in order not to compromise the performance of the numerical solver using deformed or invalid meshes.

2.1. Design of the Shape Model

The shape model is developed through *Radial Basis Functions.* RBF technique was preferred over PCA for two reasons:

- PCA-model assumes the existence of a "mean" shape and add weighted variations to it, through the use of modes. However, given the nature of our problem, it is not guaranteed the presence of maximum variability directions, due to the strong irregularity of the shapes. Therefore, it would probably be necessary to use a high number of modes, making this technique computationally cumbersome for this application.
- PCA works well, when you have a high number of shapes from which you can extract and deduce maximum variability directions. However, we have just a few structures, and even though it would be possible to add further forms through the use of some transformations (e.g., rotations, x-contractions, etc...), it would be too much laborious.

Let us recall the general structure of the RBF model, in the bi-dimensional case. We look for a map $\tau(\cdot;\mu)$, which, depending on a vector of parameters μ, could create the new shapes as:

$$\tau(\vec{x},\mu) = P(\vec{x}) + \sum_{i=1}^{k} \vec{\omega_i}(\mu)\phi(\|\vec{x}-\vec{x_i}\|)$$

where:

- $P(\vec{x})$ is an *affine transformation*;
- ϕ is the radial basis function *kernel*;
- $\{\vec{w}_i\}_{i=1}^k$ ensemble the *weight matrix* $W \in \mathbf{R}^{k\times 2}$;
- $\{\vec{x}_i\}_{i=1}^k$ ensemble the *control points matrix* $X \in \mathbf{R}^{k\times 2}$.

For our problem, we choose $P(\vec{x}) = \vec{x}$ and a *Gaussian* kernel $\phi(\|t\|) = e^{-\epsilon^2\|t\|^2}$. Control points are chosen in a way such that the model could parameterize the four morphologies in an efficient way. Therefore, they are not uniformly sampled from the template shape, but are selected by hand, taking into account the main characteristics of the LAA shapes (see section 3 for the detailed explanation).
At this point, our model becomes:

$$\tau(\vec{x};\mu) = \vec{x} + W^T \begin{bmatrix} \phi(\|\vec{x}-\vec{x}_1\|) \\ \vdots \\ \phi(\|\vec{x}-\vec{x}_k\|) \end{bmatrix}$$

and W is the only unknown of the problem. To find this latter, we exploit displacements $\{\delta_i\}_{i=1}^k$ of the control points, different for each shape, and impose:

$$\tau(\vec{x}_i;\mu) \overset{!}{=} \vec{x}_i + \vec{\delta}_i$$

so that W can be found solving the following linear system:

$$\begin{bmatrix} \phi(\|\vec{x}_1-\vec{x}_1\|) & \cdots & \phi(\|\vec{x}_k-\vec{x}_1\|) \\ \vdots & \ddots & \vdots \\ \phi(\|\vec{x}_k-\vec{x}_1\|) & \cdots & \phi(\|\vec{x}_k-\vec{x}_k\|) \end{bmatrix} W = \begin{bmatrix} \delta_{11} & \delta_{12} \\ \vdots & \vdots \\ \delta_{k1} & \delta_{k2} \end{bmatrix}$$

2.2. Numerical solver

Our numerical simulations of blood flow through the LAAs obtained with the shape model are done using `FreeFem++`, a PDE solver software.
Of course, our problem are the stationary Navier-Stokes equations in dimensionless form:

$$\begin{cases} -\frac{1}{Re}\Delta \boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + \nabla p = 0 & \text{in } \Omega \\ div(\boldsymbol{u}) = 0 & \text{in } \Omega \end{cases}$$

where:

- $\boldsymbol{u}$ is the velocity of the fluid,
- p is the pressure in the fluid,
- Re is the Reynolds number of our problem ($Re = \frac{LU}{\nu}$, with L, U reference length and velocity, ν kinematic viscosity of blood),
- We impose Dirichlet boundary condition, with fixed velocity at the inlet (left boundary) and no-slip conditions on the walls.

We implemented a fixed point method to numerically solve the problem, which is slightly more inefficient than the originally proposed Newton method, but it does not need a solution close to the actual one in order to converge (as Newton does).
In every fixed point iteration of our method we solve the following problem (which comes from the weak formulation of the Navier-Stokes equations):
Given $\boldsymbol{u}^0$ initial guess, find $(\boldsymbol{u}^k, p^k) \in V \times Q$ such that:

$$\begin{cases} a(\boldsymbol{u}^k,\boldsymbol{v}) + c(\boldsymbol{u}^{k-1},\boldsymbol{u}^k,\boldsymbol{v}) + b(\boldsymbol{v},p^k) = 0 & \forall \boldsymbol{v} \in V_0 \\ b(\boldsymbol{u}^{\boldsymbol{k}},q) = 0 & \forall q \in Q \end{cases}$$

where:

- a, b, c are the variational forms associated to each Navier-Stokes term:
  $a(\boldsymbol{u},\boldsymbol{v}) = \int_\Omega \nabla\boldsymbol{u}\cdot\nabla\boldsymbol{v}$
  $c(\boldsymbol{u},\boldsymbol{v},\boldsymbol{w}) = \int_\Omega (\boldsymbol{u}\cdot\nabla)\boldsymbol{v}\cdot\boldsymbol{w}$
  $b(\boldsymbol{v},p) = -\int_\Omega div(\boldsymbol{v})\, p$
- V, V_0, Q are the functional spaces for the problem:

$V = H^1(\Omega)$ space of velocity,
$V_0 = H^1_0(\Omega)$ space of test velocity,
$Q = L^2(\Omega)$ space of pressure.

Now, Re is stimated between 2000 and 3000 for blood flow in arteries, so we chose ours as 2300. Since it is rather high, we need some stabilization term so our solver will not encounter any numerical problem for this advection dominated flow, and we used SUPG (streamlined upwind Petrov Galerkin) stabilization, a very common technique used for this kind of problems.
In this case, we add new terms that depend on a stabilization parameter δ_k, where

$$\delta_k = \begin{cases} \delta \frac{h_k}{\boldsymbol{u}_k} \\ \delta \frac{h_k^2}{\nu} \end{cases} = \delta \frac{h_k}{\boldsymbol{u}_k} min(1, Re_l)$$

where:

- $\delta \in (0.1, 1)$ constant,
- h_k is the element k mesh size,
- $Re_l = \frac{\boldsymbol{u}_k h_k}{\nu}$ is the local Reynolds number,
- ν is the kinematic viscosity of blood.

Then we can use the `FreeFem++` solver using $P_2 - P_1$ elements for velocity and pressure fields (they are inf-sup stable to avoid additional stabilization problems) and we can calculate our indicator based on the solution we get from `FreeFem++` simulations.

2.3. Indicators

Starting from the results of the `FreeFem++` simulations, we can now construct several indicators in order to assess the thrombi formation risk associated with each mesh.
Our first approach is to analyze the velocity of the blood in every point of the domain and look for areas in which it is not moving. That is because of the piling up of red blood cells, which increases the chance of forming a thrombus. So in our analysis, we will look for shapes with a small area percentage of still blood and choose them as our healthiest shapes. It is important to note that this is very dependent on the volume and length of our meshes. In fact, we observe that the bigger the volume (more precisely the bigger the distance from the inlet), the bigger the area with still blood. That is to be expected (along with other shape features), since vortices cannot propagate indefinitely from the inlet. To eliminate the problem of bigger volumes (our shapes have very different dimensions), we will normalize the size of the shapes, and we will discuss this in further paragraphs.
Defining $\mathbf{x}_{ij} = (x_i, y_j)$ as the general point of the generated mesh, we have that we can define a sub domain $\bar{\Omega}$ of the mesh such as

$$\bar{\Omega}_{\text{shape}} = \{x_{ij} : v(x_{ij})_{\text{shape}} < \epsilon\} \tag{1}$$

And in particular we can then define the indicator such as the ratio between the sub domain previously defined and the total area of the mesh. This indicator measure the ratio between the area of the shape with steady blood over the total area. To obtain a fair comparison, we ought to have similar total area in the different meshes, to solely consider the effect of the shapes on the indicator. See Results (4) section for further analysis.

$$I_{\text{shape}} := \frac{|\bar{\Omega}_{shape}|}{|\Omega_{shape}|}$$

Given the simplified settings of our problem, namely the fact that the shapes cannot undergo contractions (that will naturally occur during a heartbeat) and that we impose a constant Dirichlet boundary condition, we carry out a further analysis after the normalization of the areas, observing that the results depend strongly on the structure of the mesh near the left boundary, rather than the specific LAA morphology. In particular, the two main features are the *funnel* effect that the mesh could have in the input, and the initial *slope* that the mesh has in the lower part of the shape. These two characteristics could heavily limit the entrance of the blood in the shape, compromising the blood flow in the entire morphology.

3. Methods

The construction of the different meshes is initially done by defining the edges in order to generate the different shapes, then using the `MeshPy` library to define the triangulation.

Editing borders

For the generation of the different meshes we decided to use the mesh template provided and, starting from that, generate the control points to obtain the different shapes. Starting from the edge of the mesh template, we have extracted the edge points in the file called `boundary_points_template.csv` that we use to obtain the actual control points used for the generation of the other meshes. The selection of these control points is done through the function :

```
def find_control_points_for_template(x, y, shape_type, plot_flag=False):
```

in which we select an edge point every 5 found and identify it as a control point and return all the control points identified in that way for the template boundary. Observe that, in the vertical line we decided to keep only 3 control points because the changes on the edges take place in the curve not corresponding to the vertical, so not many control points are needed on the vertical.

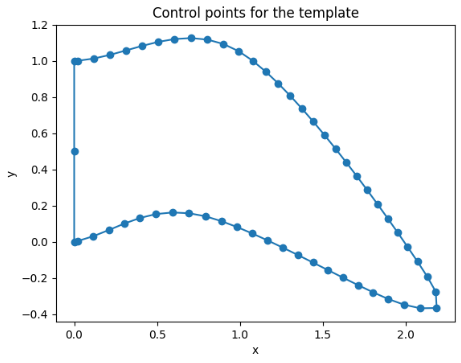


Figure 1: Control points

Since the number of control points obtained with this method is exaggeratedly high, we decide to further reduce the total number of these points for the generation of subsequent meshes. This is done by means of the function :

```
def subsample_control_points(df, shape_type, list_by_hand,
plot_flag=False):
```

in which only a part of the previously identified control points is kept, obtaining the points as in the figure 2. For the definition of the other meshes we initially use the GMSH program, which allows us to identify manually and indicatively the desired shape, obtaining a list of points that is saved in the file `punti_*.txt`. By defining these files, it is then possible to obtain the data frame of the control points of all the different shapes. This is carried out using the function :

```
def construct_cp(name_file, shape_type, plot_flag=False):
```

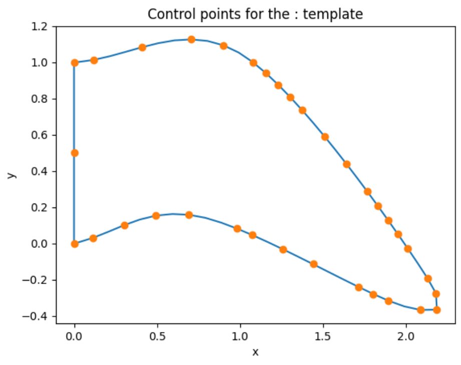
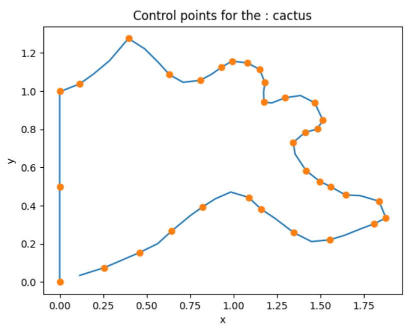
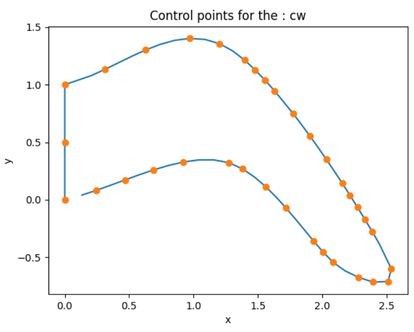


Figure 2: Control points of the template

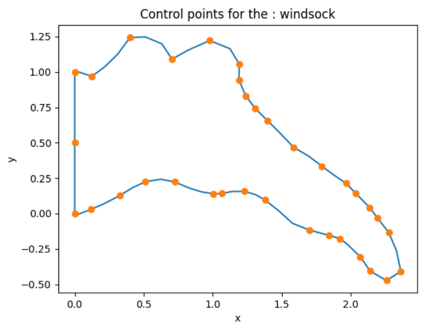
And we finally obtain the extended list of control points for each type of mesh and in particular then it is possible, as in the case of the template, to exploit the function `subsample_control_points()` to identify the new coordinates of the control points for each mesh. The definitions of the new point coordinates are shown in the figure 3.



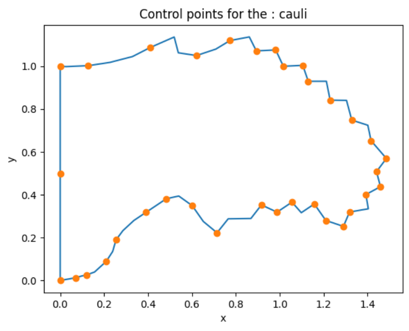
(a) Cactus



(b) Cw



(c) Windsock



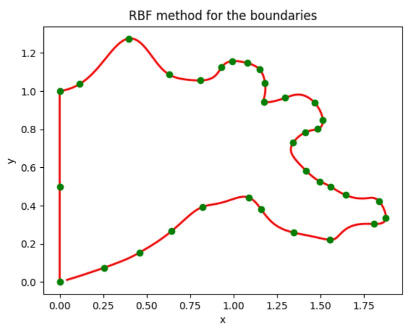
(d) Cauli

Figure 3: Control points for different meshes

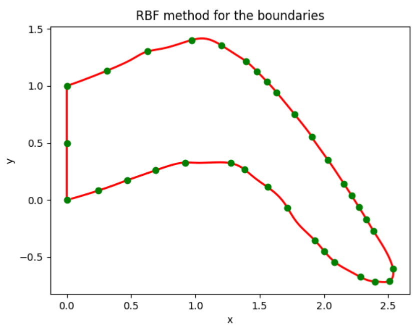
Once the list of new coordinates for each mesh type has been obtained, it is possible to exploit the Radial Basis Function to generate the new edge for each shape by exploiting the control points identified earlier. As explained before, the function used for the modelling of this case is the standard Gaussian, with $\epsilon = 4.9$

$$\phi(\underline{\mathbf{x}}) = e^{-\epsilon^2||\underline{\mathbf{x}}||^2}$$

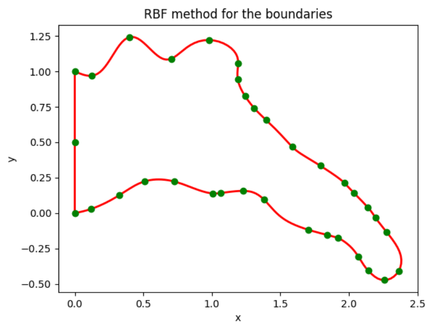
In particular, the use of this particular function allows us to obtain regular and better defined edges compared to the edges that were previously obtained by simply connecting the control points.



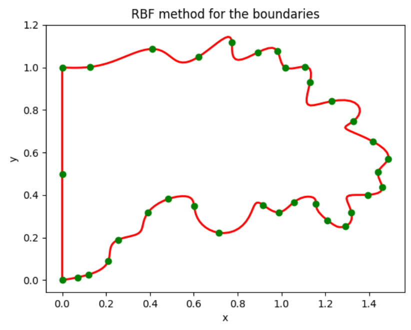
(a) Cactus



(b) Cw



(c) Windsock



(d) Cauli

Figure 4: Shapes obtained through the RBF model

Triangulation

Once the edges have been defined for each individual mesh type, we can then move on to defining the triangulation starting with the edge. We then define a class for generating the triangulation. The class is identified by a constructor with the variables identifying the mesh.
Where `markers` are identifiers of edge points, there is 1 if the respective point is on the edge, 0 otherwise. The library `MeshPy`, which is a library created for the generation of meshes, is used to implement the function :

```
    def create_mesh(self):
```

in which, starting from the previously defined edge, a triangulation is obtained for each shape. It is then possible to exploit the `refinement_func()` function, based on the control of the area of each triangle, to obtain different refinements of the triangulation in order to obtain more or less precise simulations with the chosen `FreeFem++` solver. The characteristics of the different mesh shapes are identified in the table 2.

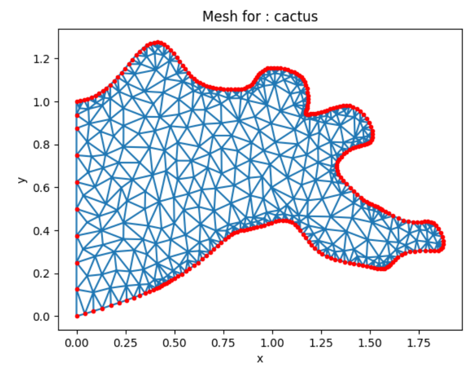
Mesh	Points	Elements	Boundary points
Cactus	1236	2190	280
Cw	1974	3663	283
Windsock	1560	2832	286
Cauli	1079	1875	281

Table 1: Main characteristics of meshes

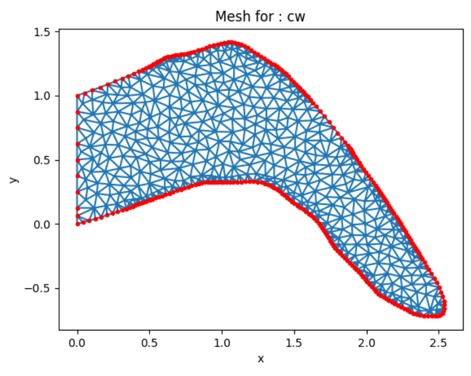
Finally, to be able to use these meshes and analyze possible simulations, we use the function :

```
    def write_msh(self,flag_trans=False, shape_type = ''):
```

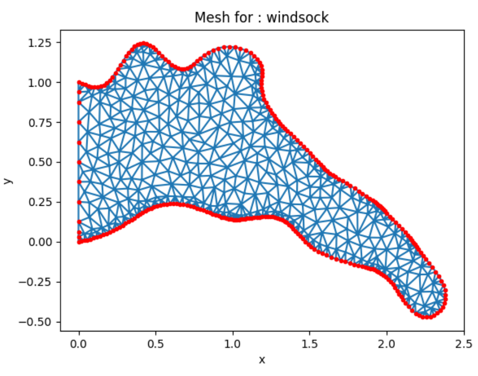
which basically writes the .msh file that can be used in `FreeFem++`.



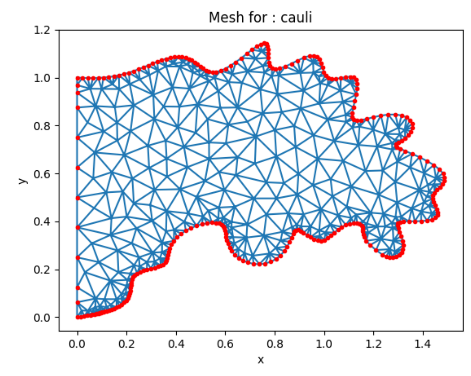
(a) Cactus



(b) Cw



(c) Windsock



(d) Cauli

Figure 5: Meshes

Transformations

In order to better analyze the actual goodness of the identified indicator and to have a complete analysis of the indicator on each shape, we decided to perform transformations on the basic shapes. In this way we can obtain different information on the different meshes of the same shape. The transformations we have chosen to exploit to modify mesh edges are :

- Stretching/Shrinking on x-axis :

```
def x_transform(coordinates, scale):
```

The expansion/restriction function on the x-axis simply modifies the entire mesh with respect to the axis with respect to a `scale`

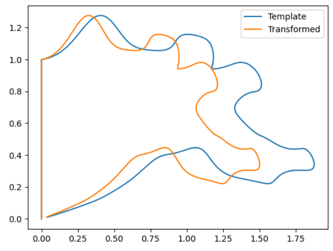


Figure 6: x_transformation

- Rotation :

```
def rotation_transform(coordinates, theta, shape_type, x_min=0.75):
```

The rotation function defines a scheme in which, depending on the minimum abscissa from which to start the rotation, the angle of the rotation itself will depend on the distance of the abscissa of the point from the minimum abscissa. This is to avoid irregular intervals and possible discrepancies on the edge defined by the non-standardised rotation.

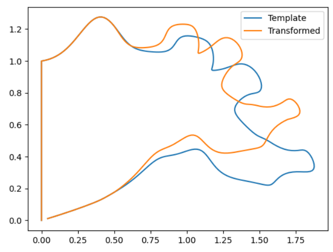


Figure 7: rotation

- Noise :

```
    def noise(coordinates):
```

For the definition of Noise the function is implemented using the `np.random` library in order to move the control points depending on a random datum for both coordinates.

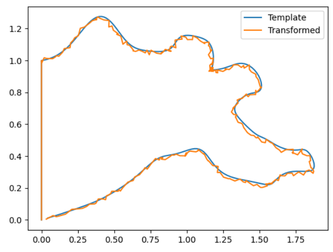


Figure 8: noisy boundary

By exploiting these 3 particular transformations, individually or even one after the other, it is possible to generate a set of meshes that will be used for the analysis of the goodness of the chosen indicator; in this way it is possible to derive information to obtain a clearer and deeper answer even for different meshes of the same type.

Normalization

As described above, once the meshes have been defined appropriately, the task is to analyze the evolution of vorticity and velocity within the different shapes using the chosen solver. In this way we can obtain feedback on the increased possibility of thrombus formation in the different shapes. To do this, and to be able to define an indicator that is "fair" with respect to each shape, it is necessary to ensure that the indicator itself depends solely on the different mesh shapes and not on factors such as area or size difference.

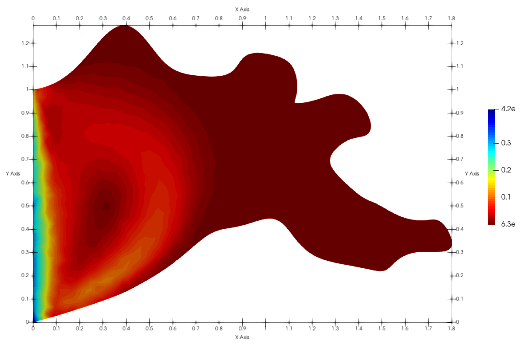
In our case, in fact, it was necessary to intervene on both axial coordinates in order to make the 4 different meshes fall within the same maximum and minimum values of the two spatial coordinates, so that the area subtended by the edge is similar for the different shapes generated. In this way it will then be possible to make a correct comparison of the different indicators for our analysis on comparable meshes, by studying only the effect of the shapes and not of their area or size.
Thanks to these transformations, it is possible to develop and construct new meshes for each shape and by running the simulation on the generated mesh, calculate the indicator for these different meshes as well. This procedure is done to try to determine which components and characteristics of each shape have the greatest influence on the chosen indicator.

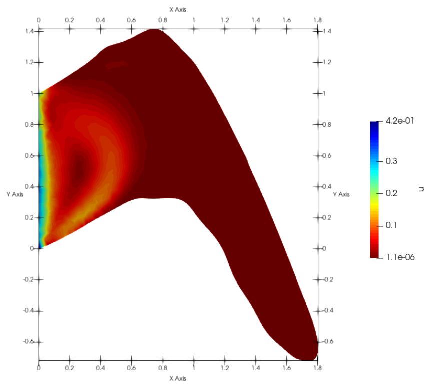
4. Numerical results

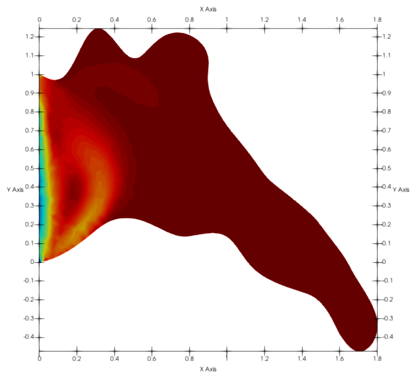
We decided to generate and use different meshes and domains to perform our simulations. These resulted in different rankings and conclusions, which we will discuss in the following chapters.

4.1. Rankings - Same area normalization

Exploiting the solver defined previously in `FreeFem++` we obtain the simulations of velocity on the different shapes. In particular we observe that in the forms elongated with respect to the x-axis there is a lower vorticity and consequently a higher probability for the formation of thrombi. We then perform the analysis on the shapes normalized with respect to the x-axis and with respect to the area, so that the evolution of the velocity in the shapes is influenced only by the shapes themselves (see figure 9)


(a) Cactus


(b) Cw


(c) Windsock

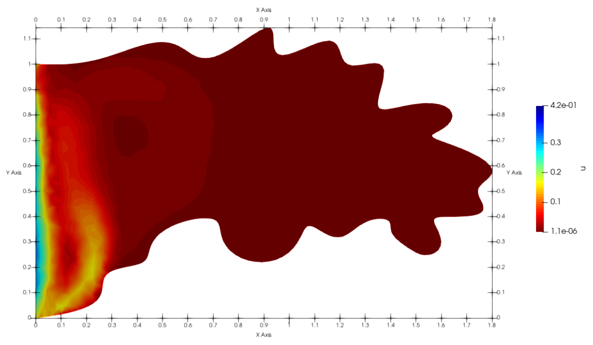

(d) Cauliflower

Figure 9: Paraview visualization

A first observation can be made on the 'Cauli' shape in which there is an obvious restriction of the mesh that channels the generated vortex. We observe in particular the different average velocities, vorticities and areas of the different shapes in the table 2. In the figure 10 we visually observe the definition of the sub-domain $\bar{\Omega}$ for the four different meshes.

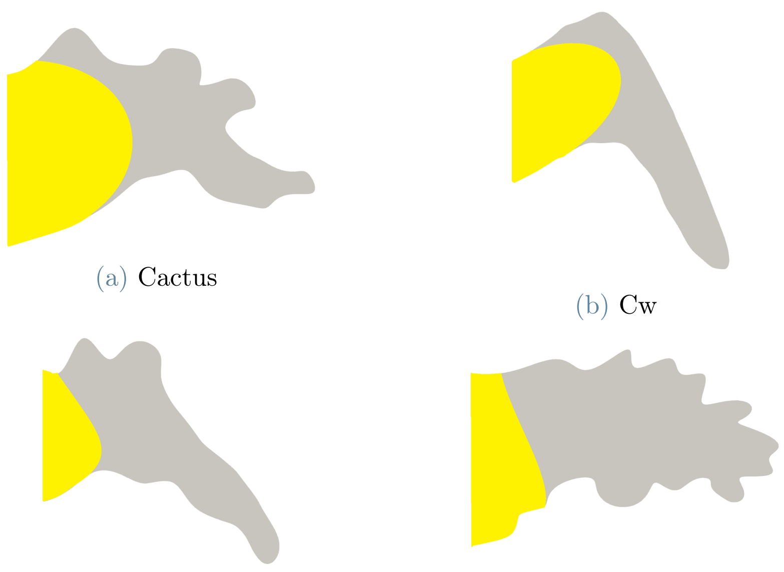

(a) Cactus (b) Cw (c) Windsock (d) Cauliflower

Figure 10: $\bar{\Omega}$ (see (1)) in yellow for the different shapes

Mesh	Mean velocity	Vorticity	N. small u	N. high u	Mean high u	Area
Cactus	0.0331954	0.300378	3644	6914	0.0367943	1.30518
Cw	0.0192699	0.171149	7431	9763	0.0280713	2.30121
Windsock	0.0237908	0.225882	5661	7801	0.0320957	1.74756
Cauli	0.0256439	0.37792	3850	5165	0.0325663	1.04189

Table 2: Before normalization

However, to make a fair comparison, we ought to analyze our results shapes having comparable areas. So normalizing the shapes, and performing again the simulation we obtain the final ranking:

Order	Mesh	Value of the indicator	Area
1	Cactus	34.88 %	1.32418
2	Cw	36.07 %	1.3300
3	Windsock	41.86 %	1.3291
4	Cauli	49 %	1.33189

Table 3: Scale normalized area

4.2. Explanation of the results

As mentioned an the end of section 2.3, results heavily dependent on the *slope* of the lower part, and on the *funnel* effect of the shapes, which create a sort of barrier for the blood flow. Since these two characteristics dominate the behavior of the simulations, we can obtain very different results even for the same shape just changing the two properties above.

4.2.1 Funnel effect

On a first approximation, we choose the area of the cut meshes as an indicator for the funnel effect. The areas of the four shapes are listed in the table 4 and we observe that their are consistent with the ranking defined on the basis of the general indicator. In fact, the larger the area of a shape the more distributed the flow entering (and exiting) the appendix, allowing good blood circulation and preventing the possible formation of thrombi.

So, to study their consequences on the simulations, we cut the meshes till a certain a certain threshold x_{max} (see figure 11). From now on, $x_{max} = 0.6$ unless differently specified.

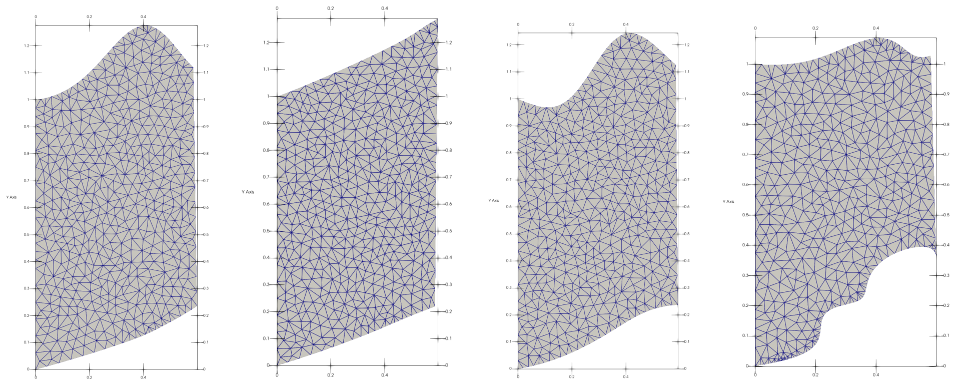


Figure 11: Cut Meshes for : Cactus/Cw/Windsock/Cauli

In particular, the "cauliflower" shape is the most dangerous precisely because the smaller area indicates a larger funnel shape that channels the fluid.

Mesh	Funnel Area
Cactus	0.622506
Cw	0.613605
Windsock	0.574017
Cauli	0.525053

Table 4: Area of the cut meshes

Remark: To be precise, this indicator could be further improved by partitioning the cut meshes in a certain number of bins, and then computing a weighted sum of the bins area, giving more importance to what happens near the left boundary. However, as a first approximation, it is sufficient to use the plain area.

4.2.2 Slope Effect

For the slope we define the indicator as follow:

$$I_{slope}(shape) = \frac{1}{N-1} \sum_{i=1}^{N-1} \frac{y_{i+1} - y_i}{x_{i+1} - x_i} (x_{max} - x_{i+1})$$

where $(x_i, y_i)_{i=1}^{N}$ are the coordinates of the lower part of the chosen mesh, and x_{max} is the threshold defined before. Observe that the bigger this indicator, the worse the performance of the mesh. The first term in the sum takes into account the finite differences (approximation of the first derivative in each point) and the second term ensembles the weights which give more importance on what happens near the left boundary (near zero). In other words, this indicator could be seen ad a rough approximation of the weighted integral of the first derivative of a function interpolating the points in the lower part of the mesh. In Table 1, you can see the results obtained for the very same meshes analyzed before:

Mesh	I_{slope}
Cactus	0.086759
Cw	0.169103
Windsock	0.152034
Cauli	0.319923

Table 5: Slope indicator for the different morphologies

Note that, contrary to the ranking and the funnel indicator we have an inconsistency: Cw and Windsock are swapped (even though the difference is very small, and almost negligible for our purposes). However, this is not an error since Windsock has indeed a bit smaller "climbing gradient" than the Cw, and we can reckon that the funnel effect is determinant in this case (indeed, windsock has a smaller funnel indicator, and overall it is worse than cw).

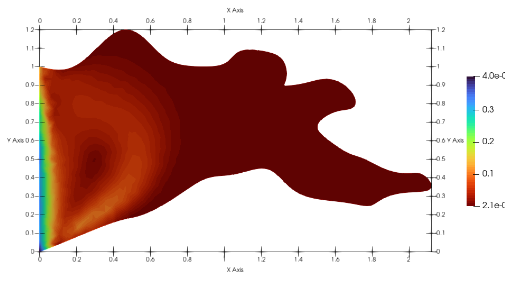
4.3. New rankings with transformed shapes

To prove our conjectures on the slope and funnel effects, we generate four new morphologies, which differ greatly from the original one based on the funnel and slope indicators, and show that the ranking can be totally turned upside down, meaning that these two characteristics hold a central role in the thrombi formation risk.
In fact, by performing new simulations we obtain these results:

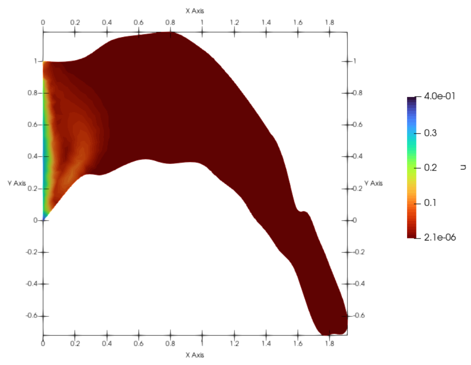
Order	Mesh	Value of the indicator	Area	Funnel Area	I_{slope}
1	Cauli	28.41 %	1.33012	0.584920	0.06330
2	Windsock	39.79 %	1.33271	0.610628	0.10780
3	Cactus	40.42 %	1.32910	0.574641	0.08228
4	Cw	50.63 %	1.33189	0.482942	0.18152

Table 6: New ranking on transformed shapes

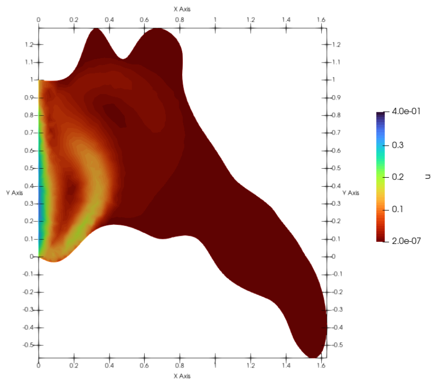
It appears clear that every shape, including the Cauli, can improve if the slope is relaxed or the funnel area is increased. In fact, in this case, Cauli became the least risk-prone shape. Moreover, observe that Windsock and Cactus are performing quite similarly with the funnel effect and slope effect compensating each other.
Nevertheless, we should underline new shapes may not reflect the actual shape of the LAAs in real human hearts, so we should refer to the indicator ranking for the original normalized shapes in Table (3). Anyhow, we reiterate that due to the very strong assumptions on the stationarity of the shapes (we don't analyze the contractions of the appendices), even the original results cannot be trusted blindly. What we wanted to point out from this study is that the thrombi risk assessment (in the static case, at least) cannot be done just basing upon the morphology type, but each appendix should be studied individually.



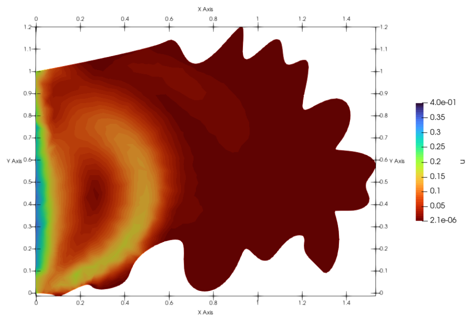
(a) Cactus



(b) Cw



(c) Windsock



(d) Cauliflower

Figure 12: Paraview visualization on the transformed shapes

Mesh	Original Funnel Area	Transformed Funnel Area
Cactus	0.622506	0.574641
Cw	0.613605	0.482942
Windsock	0.574017	0.610628
Cauli	0.525053	0.584920

Table 7: Comparison between funnel area of original and transformed meshes

Mesh	Original I_{slope}	Transformed I_{slope}
Cactus	0.086759	0.08228
Cw	0.169103	0.18152
Windsock	0.152034	0.10780
Cauli	0.319923	0.06330

Table 8: Comparison between slope indicator of original and transformed meshes

5. Conclusions

Above we proved that just changing the two properties discussed can bring to very different behaviours even in the same class of LAA morphologies.
However, notice that the underlying fluid dynamics problem is heavily non-linear, and we can't expect these two indicators to provide the same results given by the Navier-Stokes equations, even though we notice a very

strong correlation between the information given by the indicators and the numerical simulations. Nevertheless, in principle we could come up with a new statistical model based on the funnel indicator, slope indicator or a combination of them (defining the combination is not trivial though, it requires at least a clever normalization of the indicators), which can be then used to assess the severity of the shape replacing the speed indicator. This will allow to bypass the numerical simulations, dramatically improving computational efficiency of the process. Anyway, this go beyond the scope of this work, even though it would be interesting to implement such model.

References

[1] Otso Arponen Miika Korhonen, Antti Muuronen et al. Left atrial appendage morphology in patients with suspected cardiogenic stroke without known atrial fibrillation. `https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0118822#abstract09`.

- Our code repository: Repository
- MeshPy package website: Documentation