README.md

Documentation for numerical method of characteristics

This is a documentation for numerical algorithm, which solves 1st order linear partial differential equations in $\mathbb{R}^2$. I designed this algorithm in my bachelor thesis (-url leading to my thesis after publication-).

Essential software

To use this programme, you need to own a licence to Matlab programming language. If you are from my university (Charles university, Faculty of Mathematics and physics), you have it free. More on this website: https://www.mff.cuni.cz/cs/math/vnitrni-zalezitosti/pocitace-a-technika/matlab. I wrote the programme on version Matlab 2024a. In might not work properly on other versions. So if that's your case, download Matlab 2024a.

Then you could use free graphics software Paraview, where you will be able to render 3D approximated solutions in pretty high quality. If you will decide to do so, download the latest version of Paraview from https://www.paraview.org/download/. However, downloading Paraview is an optional choice. You will see a little bit less detailed 3D graphs made in Matlab.

Contents of the .zip file

The .zip file contains this README file, where are all information and instructions for users.
Then it contains all essential matlab files (= " `xxxx.m` ") you need, in order to run the programme:

`Add_characteristics.m, Approximate_solution_inside.m, Approximate_solution_vertices.m,`

`B.m, Barycentric_interp.m, Create_characteristic.m, Create_net.m, createVTK.m,`

`Disc_entry_bdd.m, Entry_bdd.m, Find_closest_vertex.m, Find_position.m, Is_in_triangle.m,`

`Ivc.m, Main.m, MoC.m, Set_bdd.m, Set_def.m, writeVTKcell.m`

Extract all these files into one folder.

How does the algorithm work

Algorithm solves equations of following type:

$$\begin{cases} b_1 u_x + b_2 u_y + cu = f, & \text{in } \Omega \subset \mathbb{R}^2, \\ u = g, & \text{in } \partial\Omega^-. \end{cases}$$

Here $\Omega \subset \mathbb{R}^2$ is open bounded set. We define Ω as interior of a closed curve in $\mathbb{R}^2$ which divides the plane into only two components. First one has infinite diameter and the second one is the Ω.
Then $b_1,\ b_2,\ c,\ f$ are lipschitz continuous real functions defined on $\bar{\Omega}$.
We define $\partial\Omega^- = \{(x,\ y) \in \partial\Omega : b(x,\ y) \cdot n(x,\ y) < 0\}$, where $b = (b_1,\ b_2)$ and $n : \partial\Omega \to \mathbb{R}^2$ is a normal vector to $\partial\Omega$. Basically $\partial\Omega^-$ is a part of boundary through which characteristic curves enter.
Then g is real lipschitz continuous function defined on $\partial\Omega^-$.
Finally $u \in C^1(\bar{\Omega})$ is a real function for which we solve the equation.

The algorithm will discretize $\partial\Omega^-$, then calculate approximation of characteristic curves using Runge-Kutta method. While executing these steps, it will try to find the points on characteristic curves, so that they are not too far from each other. This is controlled using a fixed toleration constant. Then it will cover Ω with triangle net, but the vertices of triangles will lie on characteristic curves. It will calculate approximated solution in vertices of the triangles using composite trapezoidal rule. It will also find some approximated solution values inside the triangles using barycentric interpolation and then create a graph where will be boundary of Ω, characteristic curves and brief shape of the solution. Finally, it will eventually create .vtk file.

How to start the algorithm

To start the algorithm, start the function `Main.m` in Matlab (make sure that all other matlab files are with this function in the same folder).
Now you need to fill all the data in. Matlab will display input windows one at a time. At the end of this section, I will also show an example how to fill them in.
For decimal point use dot (e.g.: "3.6"), for multiplication use "*", for division "/".
Here is the pdf with table containg some elementary functions and other rules: https://www.mathworks.com/content/dam/mathworks/fact-sheet/matlab-basic-functions-reference.pdf.

For functions $b_1,\ b_2,\ c,\ f$ use variables $x,\ y$. If they are equal to zero, just write 0. You need to press "OK" to insert the input.

Then it will ask "How many set defining functions?". Write a number of functions (technically inequalities) which are going to define set Ω, click "OK". After that you will insert 1st function f_1, click "OK", and then you insert functions g_1 and h_1, such that $g_1 < f_1 < h_1$. Insert them in format "g, h". Again use variables x, y. Take a side from which f_1 is not bounded and write (appropriatelly) "Inf" or "-Inf". It will be always possible and you NEED to do it, click "OK". After that you insert 2nd function f_2 etc...
Algorithm need this, so it can control whether some points are out of bounds or not.

Then it will ask "How many boundary functions?". Write number of curves representing the whole boundary of Ω, click "OK". It wants parametric expression of the curves. The curve $\omega = (v(s),\, w(s))$ represented by functions v, w has to be inserted with variable s. Insert functions v, w together seperated by comma: "v, w". Don't use any space after the comma! Click "OK". Then insert interval parametrizing the curve. For $I = [a,\, b]$ insert "a, b", click "OK". Now you can insert another curve etc...
Algorithm need this, so it can graph the boundary at the end.

Then it will ask "How many ENTRY boundary functions?". It is the same process as in the paragraph above. Insert how many curves will define entry boundary and then write them the same way as curves representing the whole boundary. You will basically just chose some curves you already inserted, insert them again and maybe change parametrization interval.

Afterwards you will insert initial value condition, which is function g.
After that you can write "y" or "n" if you want or don't want to generate .vtk file for Paraview (more on that later).

Finally you choose if you want to adjust toleration constant. I recommend leaving it fixed for first use. It is equal to 0.15. If you reduce it, you will get more accurate solution, but the programme will take longer time to execute.

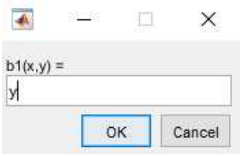
After clicking final "OK", the algorithm will start. You can see it progressing in the terminal. At the end, it will plot a graph with boundary of Ω, characteristic curves and a brief shape of the solution. Eventually, it will generate you .vtk file in the folder.

Example

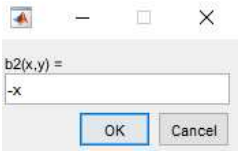
$$\begin{cases} yu_x - xu_y = 0, & \text{v } \Omega = \{(x,\, y) \in \mathbb{R}^2 : x^2 + y^2 < 4;\; y > 0\}, \\ u(x,\, y) = \cos(x + \pi/2), & \text{pro } (x,\, y) \in [-2,\, 0] \times \{0\}. \end{cases}$$

For this example, it will look like this:

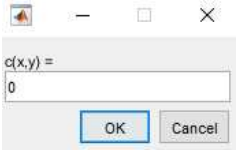
1.



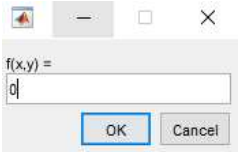
2.



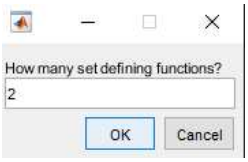
3.



4.

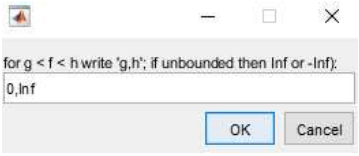


5.

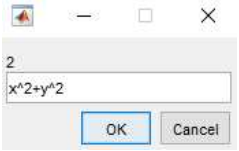


6.

7.

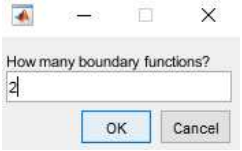


8.

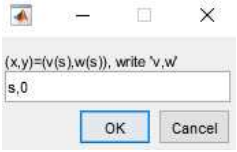


9.

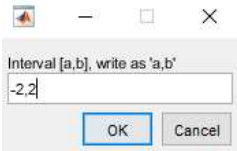
10.



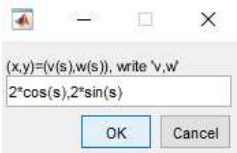
11.



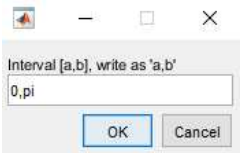
12.



13.

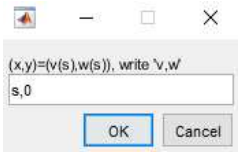


14.

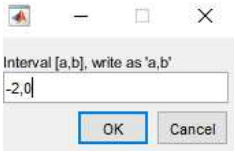


15.

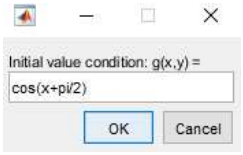
16.



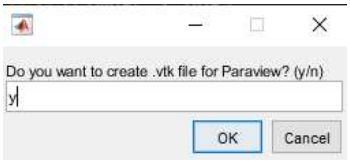
17.



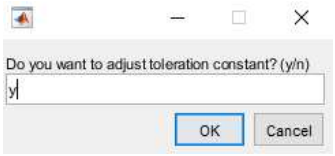
18.



19.

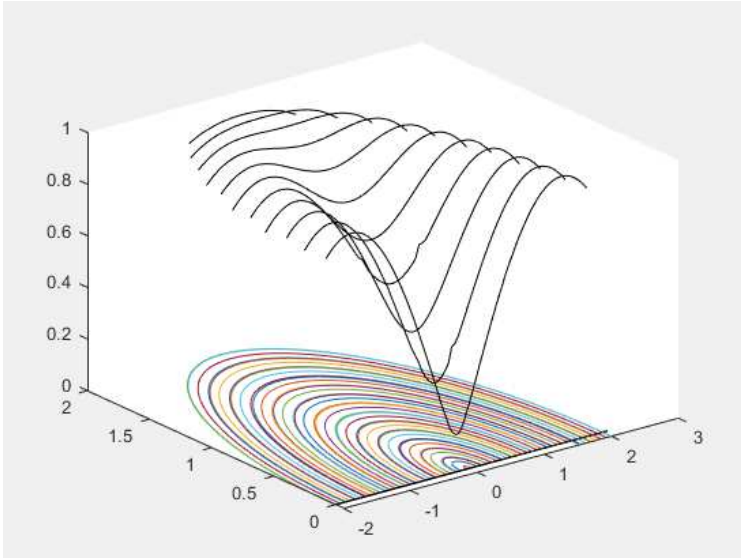


20.



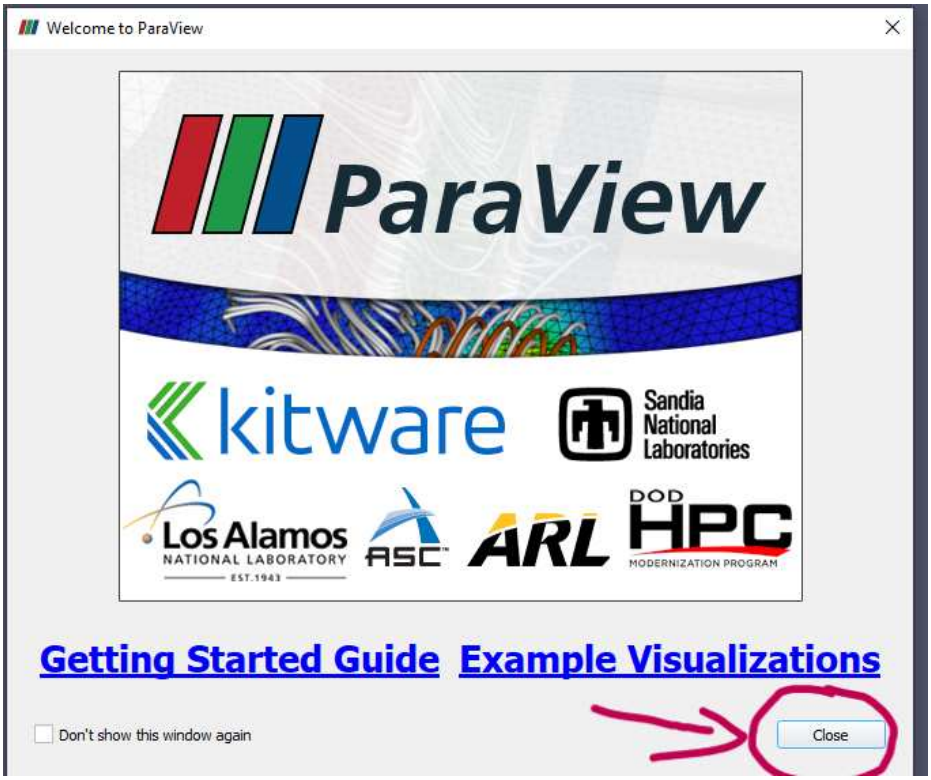
21.

After execution of algorithm, we get following graph:

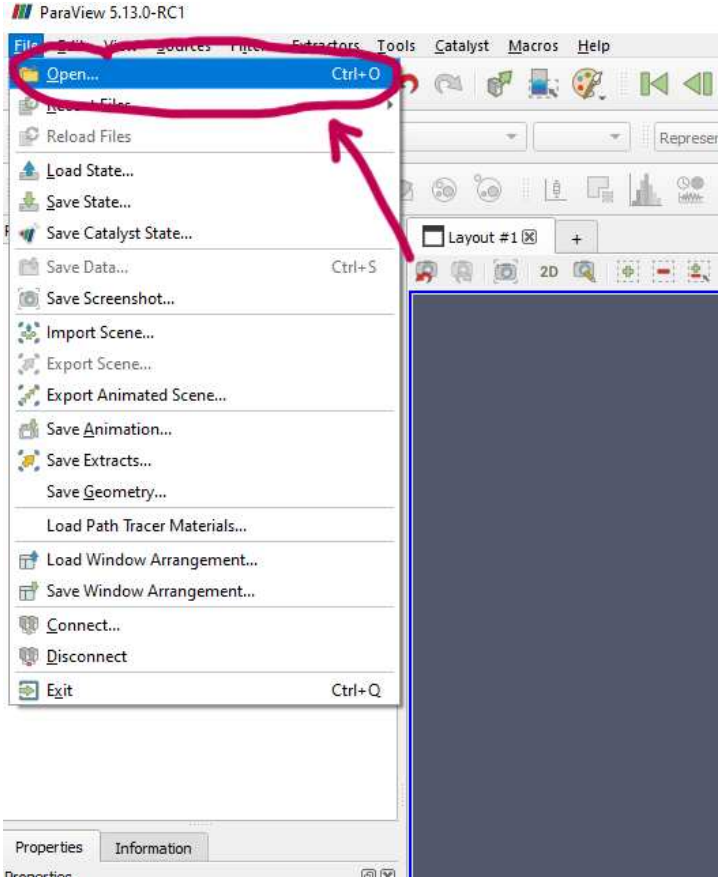


Paraview

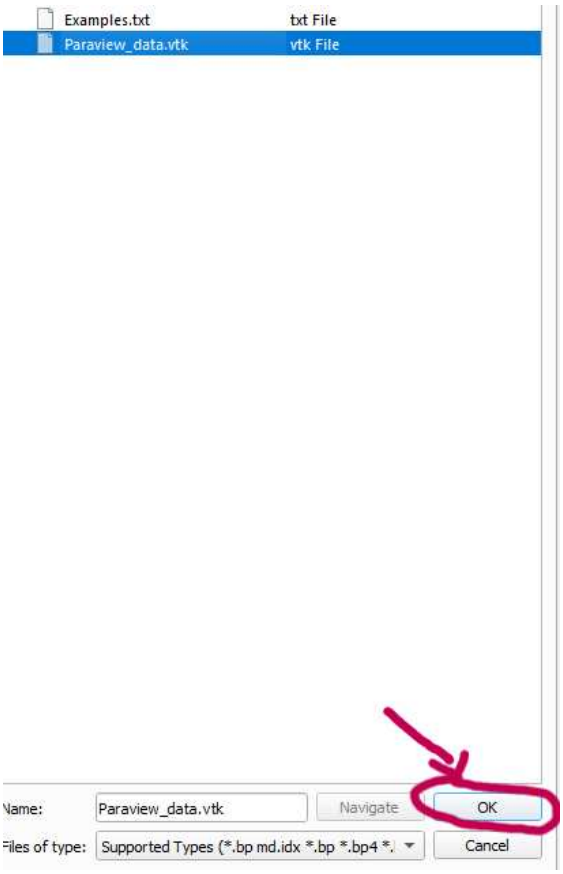
If we generated .vtk file as well, I will describe how to easilly graph the approximated solution in Paraview. After opening the software, close the followinng window:



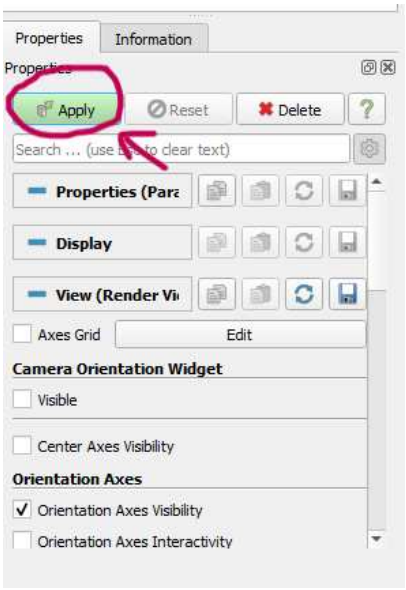
Then insert the Paraview_data.vtk file which is in the same folder as `Main.m` . You do that by "CTRL + O" or in the top left corner is File-->Open...



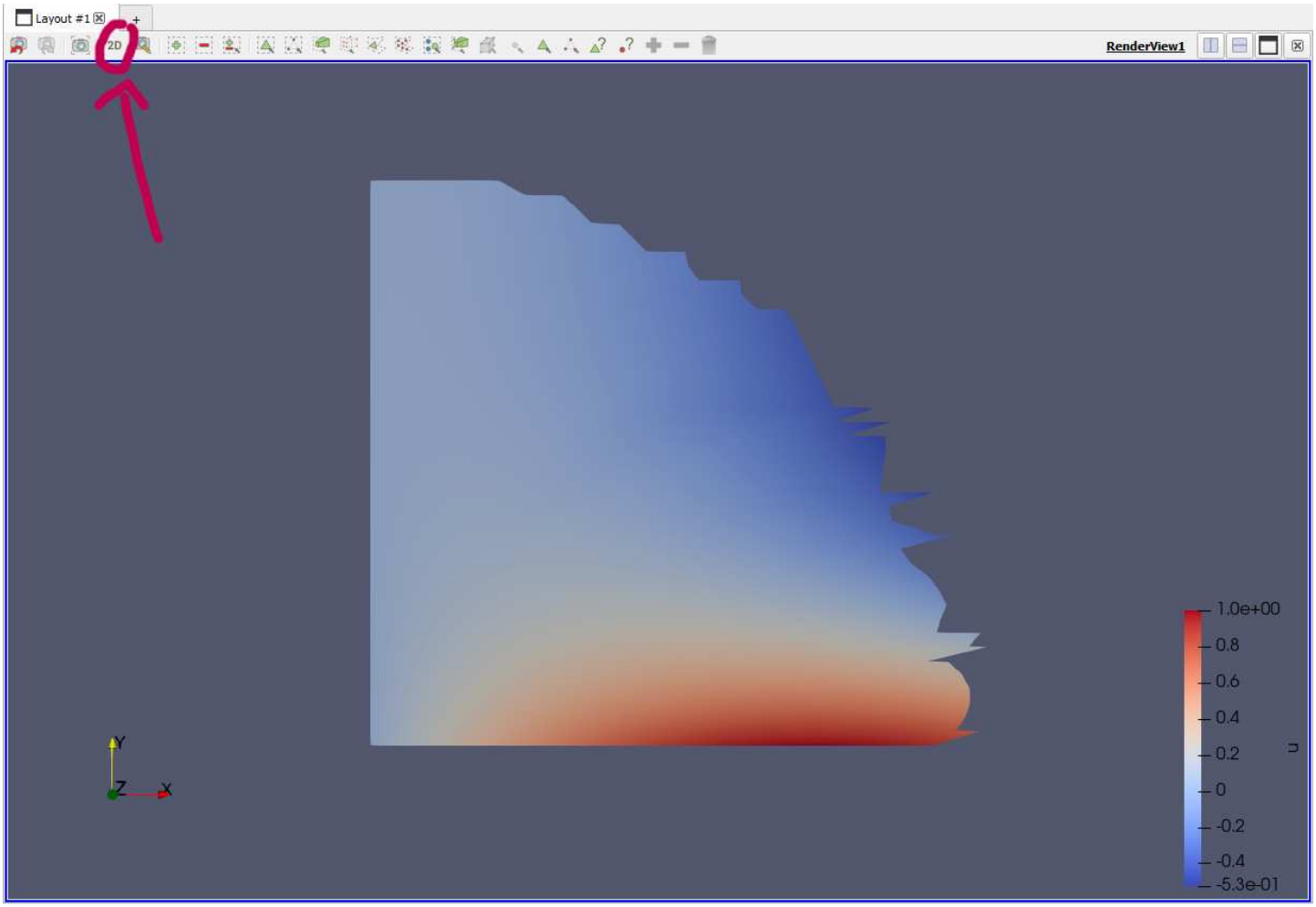
After founding Paraview_data.vtk file and selecting it, click "OK":



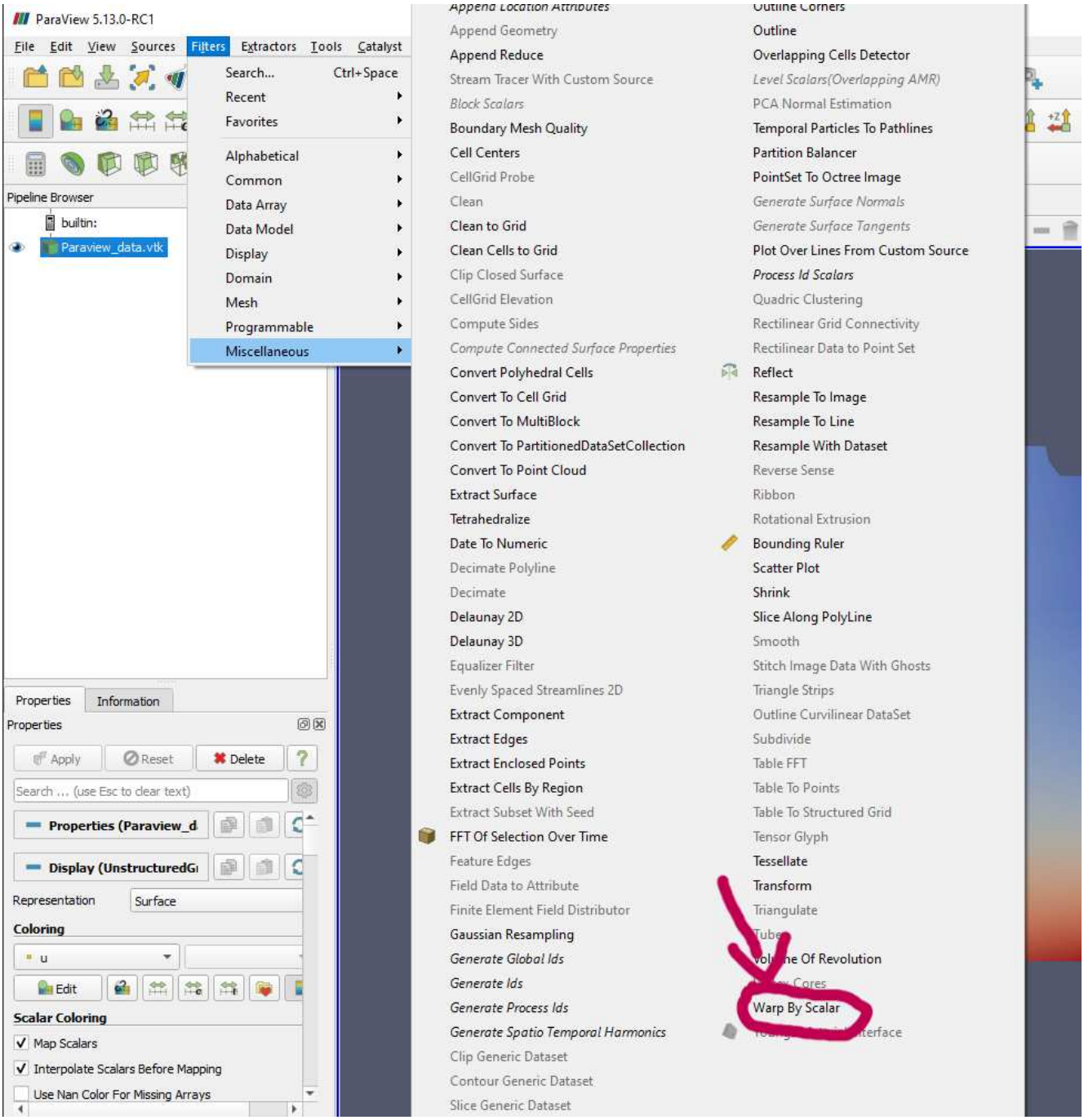
Then at the left side of the Paraview click on the green button "Apply":



You will already see some graph. Now click on the "2D" button above the graph which will change into "3D":



After that, select at the top menu in Paraview Filters-->Miscellaneous-->Warp By Scalar



Click "Apply" again and there you go!