**Prandtl-Meyer Expansion**[1]

*Simulation Course*

*Escola Tècnica Superior d'Enginyeria de Telecomunicacions i Aeroespacial de Castelldefels*

*Universitat Politecnica de Catalunya*

León Enrique Prieto Bailo [leon.enrique.prieto@estudiantat.upc.edu](mailto:leon.enrique.prieto@estudiantat.upc.edu) Verónica Sastre Rojo [veronica.sastre@estudiantat.upc.edu](mailto:veronica.sastre@estudiantat.upc.edu)

**Abstract**

The Project report contains information about the physical phenomenon, the mathematical resolution and the implementation of the problem in a software designed for the user.

First, the problem situation will be raised, what is the need to simulate this physical phenomenon? How can we apply it in a practical case?

Later, in the mathematical field, we will talk about how we should treat the scenario that arises in order to be able to generate a resolution algorithm that allows solving the problem and how to derive it to implement it in a computer.

Finally, the needs that the software must meet and how the required implementations should be carried out so that the user can use the simulator in a robust and friendly way will be discussed. Additionally, in the document, you can find documentation related to the structure of open source and free use classes.

**Resumen**

El reporte del Proyecto contiene información acerca del fenómeno físico, la resolución matemática y la implementación del problema en un software diseñado para el usuario.

Primeramente, se planteará la situación del problema, ¿cuál es la necesidad de simular este fenómeno físico? ¿Como podemos aplicarlo en un caso práctico?

Posteriormente, en el ámbito matemático, se hablará de cómo debemos tratar el escenario que se plantea para poder lograr generar un algoritmo de resolución que permita resolver el problema y como aproximarlo para implementarlo en un computador.

Finalmente, se discutirán las necesidades que debe cumplir el software y como se deben realizar las implementaciones requeridas para que el usuario pueda utilizar de manera robusta y amigable el simulador. Adicionalmente, en el documento, se puede hallar documentación relacionada con la estructura de clases de código abierto y de libre uso.

**Resum**

El report del projecte conté informació sobre el fenomen físic, la resolució matemàtica i la implementació del problema en un programari dissenyat per a l'usuari.

Primerament, es plantejarà la situació del problema, quina és la necessitat de simular aquest fenomen físic? Com podem aplicar-ho en un cas pràctic?

Posteriorment, a l'àmbit matemàtic, es parlarà de com hem de tractar l'escenari que es planteja per poder generar un algorisme de resolució que permeti solucionar el problema i com aproximar-lo per implementar-lo en un computador.

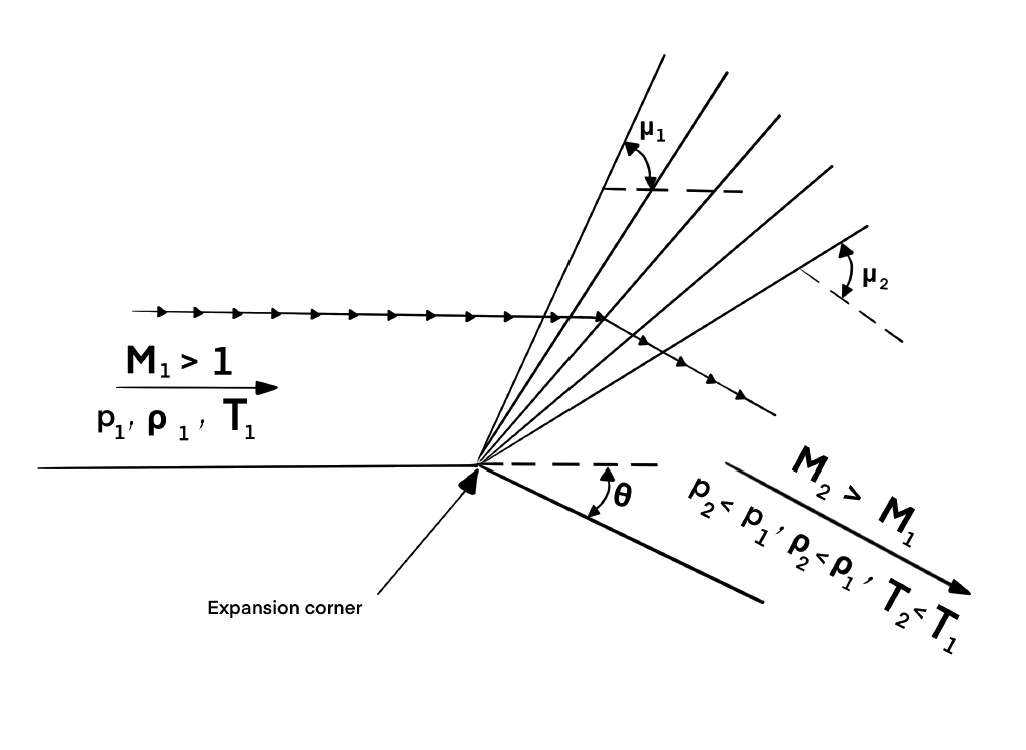
Finalment, es discutiran les necessitats que ha de complir el programari i com cal fer les implementacions requerides perquè l'usuari pugui utilitzar de manera robusta i amigable el simulador. Addicionalment, al document, es pot trobar documentació relacionada amb l'estructura de classes de codi obert i de lliure ús.

1. **Description of the problem. Why this simulation is important?**

The Prandtl-Meyer expansion is the process that occurs when a supersonic flow turns around a convex corner, forming a divergent fan consisting of an infinite number of Mach waves. A two-dimensional, inviscid and supersonic flow will be analyzed in this project.

Across the expansion fan, the flow accelerates so, the Mach number increases and the static pressure, temperature and density decreases.

Since the flow turns in small angles and the changes across each expansion wave are small, the whole process is isentropic. Due to the fact that the whole process is isentropic, the stagnation pressure, temperature and density remain constant across the fan. This fact simplifies the calculations of the flow properties significantly.



*Figure 1. Prandtl Meyer expansion wave.*[1]

Due to the fact that the inviscid flow must easily notice the shape of the surface over which it is flowing, it is vital to couple the surface boundary condition into the flow-field calculation. For this reason, a numerical mathematics adjustment will be done.

The space marching technique that will be applied for the solution of the two-dimensional supersonic flow problem is MacCormack's. It is a second order discretization scheme for the numerical solution of hyperbolic partial differential equations.

The relevance of the project lies in the understanding of the physical phenomenon. It is important to know how the fluid behaves when certain geometric conditions are met. The Prandtl Meyer expansion can be found in any type of aircraft, rocket or object that moves in a supersonic regime.

That is why it is of vital importance and of great help to use numerical methods to represent the phenomenon and simulate the behavior of the flow for a geometry of interest, as well as to obtain the properties and magnitudes of the flow in this scenario.

1. **Prandtl-Meyer Expansion: Mathematical Analysis**
   1. **Problem SetUp.**

The angles of the expansion wave with respect to the horizontal component is denoted as , the upper and lower angle boundaries of the wave are denoted and , respectively. Therefore, we can define:

The Prandtl-Meyer function for a calorically perfect gas, denoted as , follows:

This allows us, by implicitly solving (trial and error), to find the Mach number along the wave. The process is as follows: is calculated by adding to the previous expression, then, for a given , is computed and finally, by solving implicity, is found.

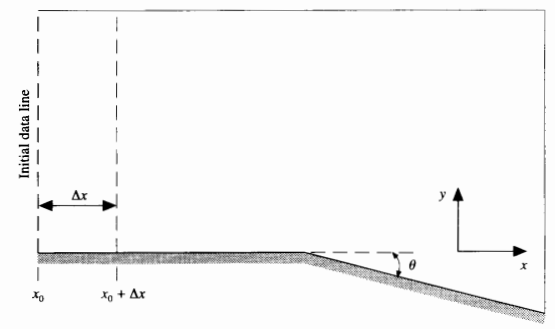
Once the Mach number is found, the pressure, temperature and density downstream can be found from the isentropic flow relations.

The governing Euler equations for a steady, two-dimensional flow can be expressed in the generic form of hyperbolic partial differential equation, due to isentropic flow consideration, the value of is equal to zero.

In order to solve the previous equation, the MacCormack’s predictor-corrector explicit finite-difference method will be applied.

and are the column vectors whose values are defined as follows:

An initial data line will be set for and then the solution will be carried out by marching in steps of

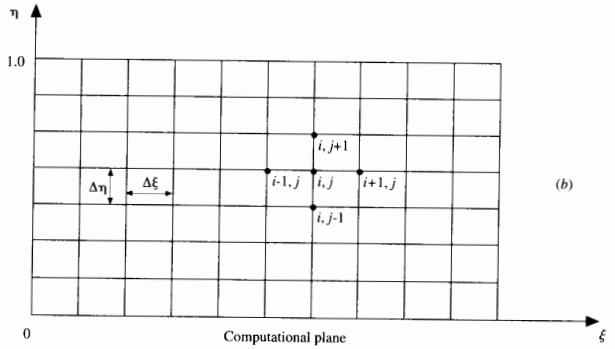
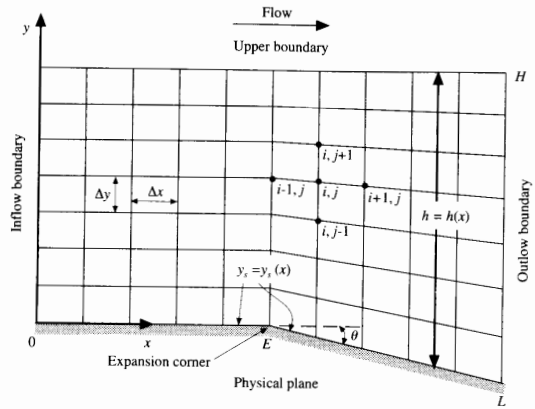


*Figure 2. Geometry of the problem.*[1]

Primitive variables can be obtained by applying the following expressions with the column vector.

And the column vectors can be obtained from the primitive and column vectors as follows:

To simplify the computational problem, a computational plane has to be generated from the physical one, then find the primitive variables and revert the change to the physical plane.



*Figure 3. Physical and computational plane.*[1]

The change of variable to be performed is:

The coordinate corresponding to the lower boundary is denoted as , this value will be and then, after the expansion corner, it will start decreasing. The height or the distance between and the is defined as .

For

For

Then, the value of can be written as:

Then, the continuity, x-momentum, y-momentum and energy governing flow differential equations that must be solved in the computational plane follow:

Now, the problem is fully set up and the differential equations can be solved by using MacCormack’s method.

* 1. **Predictor step: Written using forward differences.**

The discontinuity tends to produce oscillations in the solution of the flow field, these can be eliminated by including artificial viscosity in the solution.

The predicted values of are obtained as follows:

With the predicted values, predicted column vector can be obtained by finding the value of .

then is found by:

* 1. **Corrector step: Written using rearward differences:**

Forming the average derivatives:

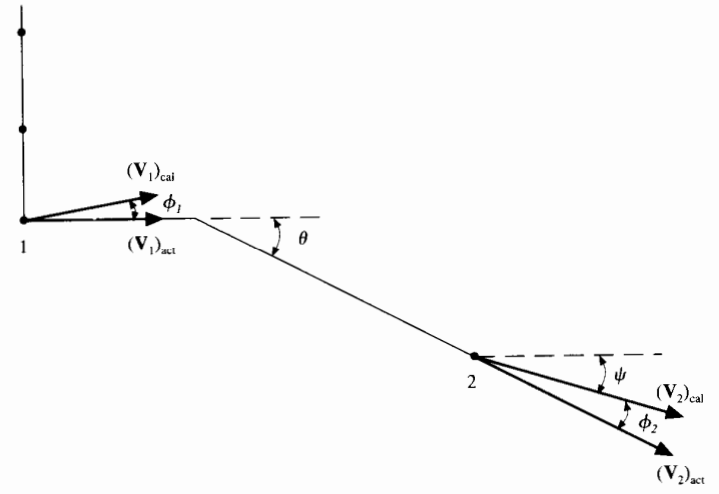
In this corrector step it is also needed to add the artificial viscosity, thus:

And then, it’s finally possible to recover , where the primitive variables can be recovered from.

* 1. **Boundary Conditions**

The same process is repeated for the boundaries, for the rearward differences are changed to forward differences and for the forward differences are changed to backwards. The artificial viscosity (computed by using central differences) is not used for the boundaries.

Sketch of the velocity vectors in the boundary:



*Figure 4. Boundary velocity vectors before and after the expansion corner.*[1]

The direction of the resultant velocity at the wall will not necessarily be tangent to the wall due to numerical inaccuracy. This can be quantified and corrected. The angle between the velocity vector and the wall before the expansion corner is denoted as whose value is:

The calculated Mach number at the wall will be:

Before the, there could be a rotation of the velocity vector through the Prandtl-Meyer expansion wave where the deflection angle through the wave is , this yields to a new velocity vector which is assumed to be the actual velocity tangent to the wall. This can be computed by finding the associated Mach number obtained implicitly from the Prandtl-Meyer relationship.

Solving by trial and error, and obtaining the Mach number derives us into finding the primitive actual variables as follows:

The same thing happens after the expansion wave, the velocity vector is rotated to the angle , and then the Prandtl-Meyer function becomes:

Where:

Finally, the computation of the downstream marching step size can be obtained by:

1. **Software design: describe objects, databases, method for visualization of results**
   1. **Class description**

The project is formed as a whole by the combination of the following three classes.

* + 1. **Rules Class**

In this class the initial parameters of the simulation, the constants involved into the phenomenon as well as the baseline of the Grid are set, this class will be used as a query class, therefore, it will be recursively be used during the simulation. The class contains basic getters and a constructor which initialize the object, this is the first class that is going to be used since Cell and Grid rely on Rules.

* + 1. **Constructor**
* **public Rules(double u, double v, double ro, double p, double T, double M, double Cy, double gamma, double R, double E, double theta, int j, double xMax, double H, double Courant):** Initial constructor which initializes the object using user input variables.
  + 1. **Attributes**
* **double u0:** Initial horizontal velocity of the fluid.
* **double v0:** Initial vertical velocity of the fluid.
* **double ro0:** Initial density of the fluid.
* **double p0:** Initial pressure of the fluid.
* **double T0:** Initial temperature of the fluid.
* **double M0:** Initial Mach number of the fluid.
* **double Cy0:** Viscous courant constant.
* **double gamma0:** Adiabatic gas constant.
* **double R0:** Ideal gas constant.
* **double E0:** Location of the expansion corner.
* **double Theta0:** Angle of the expansion deflection.
* **double xMax0:** Maximum length of the experiment.
* **double H0:** Initial height of the experiment.
* **double Courant0:** Courant constant.
* **int J0:** Number of vertical cells of the experiment.
  + 1. **Methods**
* **public double getU(), getV(), getRO(), getP(), getT(), getM(), getCy(), getGamma(), getR(), getH(), getxMax(), getCourant(), getE(), getTheta(), public int getJ():** Methods used to recover the variables of the rules class.
  1. **Cell Class**

This class contains the fluid attributes for a specified point of the grid. The combination of numerous cell objects will create the grid as a whole. The class is also capable of computing its own status by calling the “Preditor Step” and “Corrector Step” functions, which perform the MacCormack method with the surroundings provided in grid format. It also has its getters in order to read the information of the fluid properties as well as a constructor for initialization.

* + 1. **Constructor.**
* **public Cell(Rules rIn, int veIn, int hoIn):** Initialises the cell object with the rule class, also tells it on which position is located of the grid.
  + 1. **Attributes.**
* **double u, v, ro, p, T, M:** Magnitudes of the cell fluid.
* **double F1, F2, F3, F4, G1, G2, G3, G4:** F and G values of the fluid point.
* **double F1pre, F2pre, F3pre, F4pre, G1pre, G2pre, G3pre, G4pre:** Predicted F and G values of the fluid point.
* **double roPre, pPre:** Predicted pressure and density values of the fluid point.
* **double dF1de, dF2de, dF3de, dF4de:** Differentials of F values with respect to the differential of the horizontal component.
* **rules r:** Stores the rules which rule on the fluid point.
  + 1. Methods.
* **public void PredictorStep(double dEtadX, double dEta, double h, double dXi, Grid g):** The own cell computes it’s next predicted values by feeding him initialized variables on the grid.
* **public void CorrectorStep(double dEtadX, double dEta, double h, double dXi, double Xi, Grid g):** The own cell computes it’s final values after applying the predictor step before.
* **public double getU(), getV(), getRO(), getP(), getT(), getM():** Getters which return the magnitudes of the fluid point.
* **public double getF(int i), getG(int i):** Getters which return the F and G values.
* **public double getFpre (int i), getGpre(int i):** Getters which return the F and G values obtained with the predictor step.
* **public double getPpre():** Getter which returns the predicted pressure.
  1. **Grid Class**

This class will generate the structure and define the location of the cells in our simulation. Grid class has the method which performs the whole simulation by calling the cell functions explained before, it also has two functions which are used to initialize and define the computational plane.

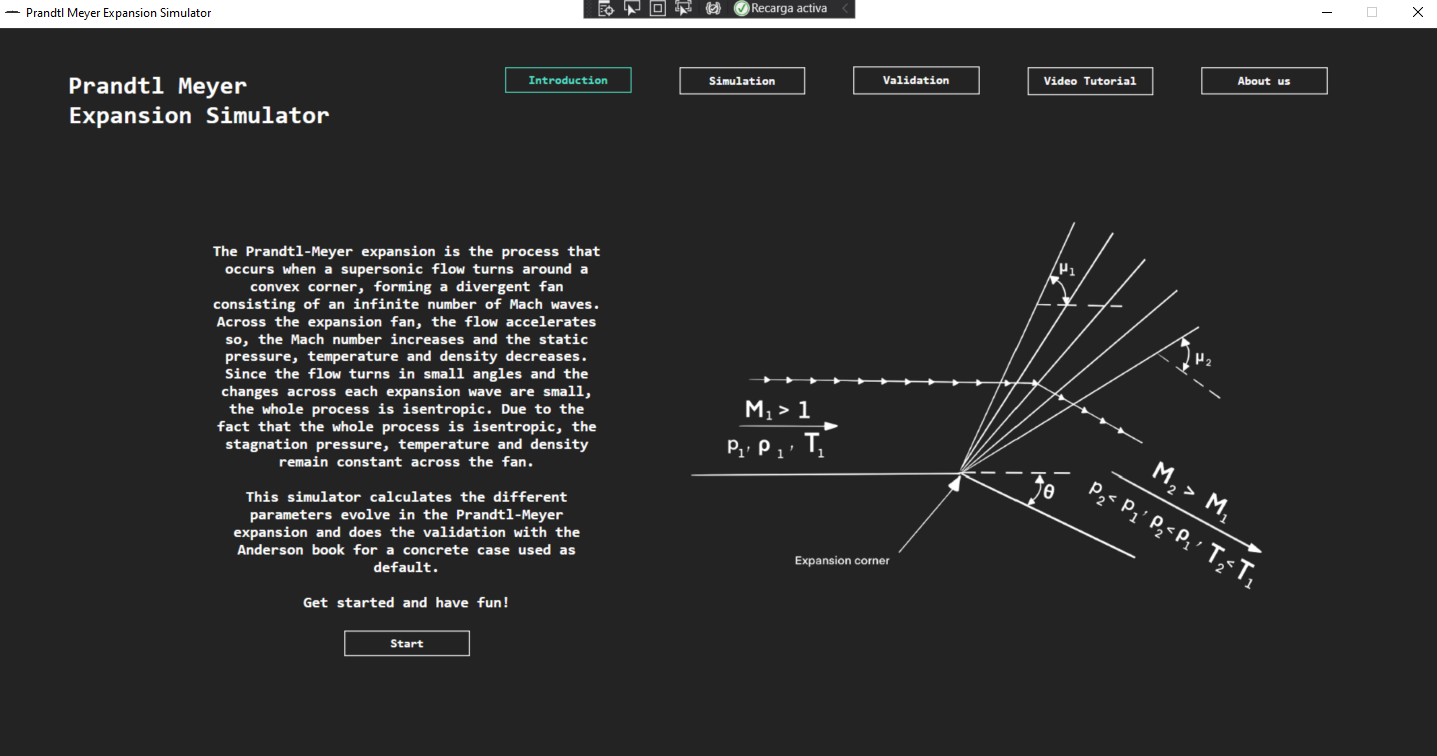
* + 1. **Constructor**
* **public Grid(Rules rIn):** Constructor that allows us to initialize the grid with information provided by a Rules object.
  + 1. **Attributes**
* **List<Cell[]> Mesh:** List of Cell class object vectors which stores every cell of the grid.
* **List<double> xP:** List of horizontal coordinates.
* **Rules r:** Stored rules variable.
* **Double dEta:** Vertical differential of the computational plane.
* **Double h:** Height of the current column.
* **Double dy:** Vertical differential of the physical plane.
* **Double xi:** Actual position of the column.
* **Double dXi:** Horizontal differential of the computational/physical plane.
* **int ho:** Stores horizontal index of the row.
* **List<double[]> yP:** List of the position of the cells in the physical plane.
* **double[] dEtadX:** Stores the differential of the vertical component in the computational plane w.r.t. the horizontal differential of the computational/physical plane.
  + 1. **Methods**
* **public void PrandtlMeyerExpansion():** Computes the solution of the experiment by using MacCormack's predictor and corrector step.
* **private void InitVars():** Initalises the variables needed for every row.
* **private void ComputeStepSize():** Computes the horizontal differential of the computational/physical plane.
* **public (double u, double v, double ro, double p , double T, double M) getDownstream():** Returns downstream fluid magnitudes.
* **public Cell GetCell(int ve, int ho):** Returns the cell located in vertical and horizontal index.
* **public List<double[]> GetYP():** Getter for the vertical positions of the grid.
* **public List<double> GetXP():** Getter for the horizontal positions of the grid.
  1. **Forms description**

The project has only one window form with four buttons: Introduction, Simulation (Canvas Grid and Research Grid), Video Tutorial and About us. Every button is associated to a particular grid and if the user presses the button the grid associated with it will be the one on the screen. The program starts by default with the Introduction Button pressed and it shows the Introduction Grid.

The grids that the simulator has are:

* + 1. **Introduction Grid.**

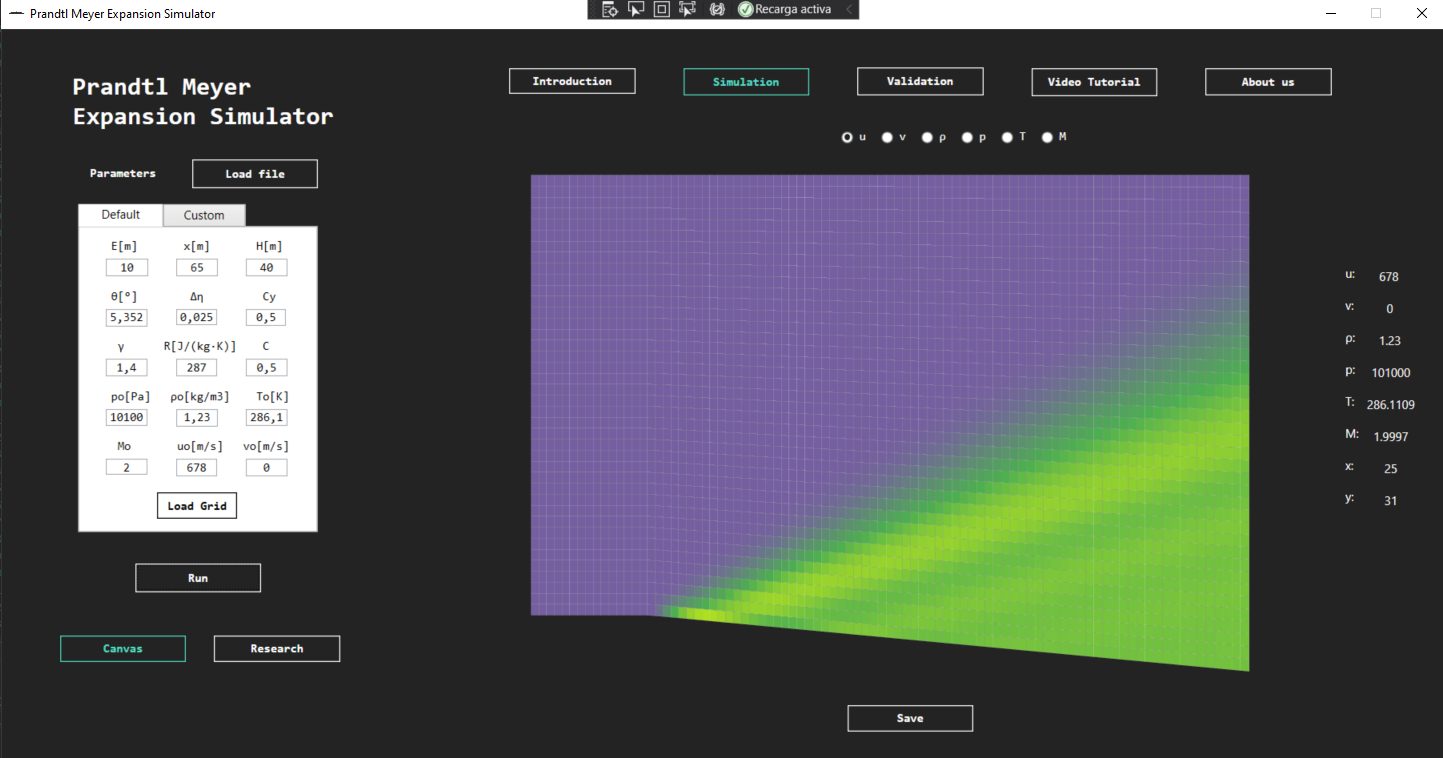
In this grid, a short introduction to the topic, what the simulator does and a scheme of the phenomenon is presented.



*Figure 5. Introduction Grid.*

* **“Start” Button:** button that shows the Simulation Grid. Also, you can access the Simulation Grid by pressing the Simulation button.
  + 1. **Simulation grid.**

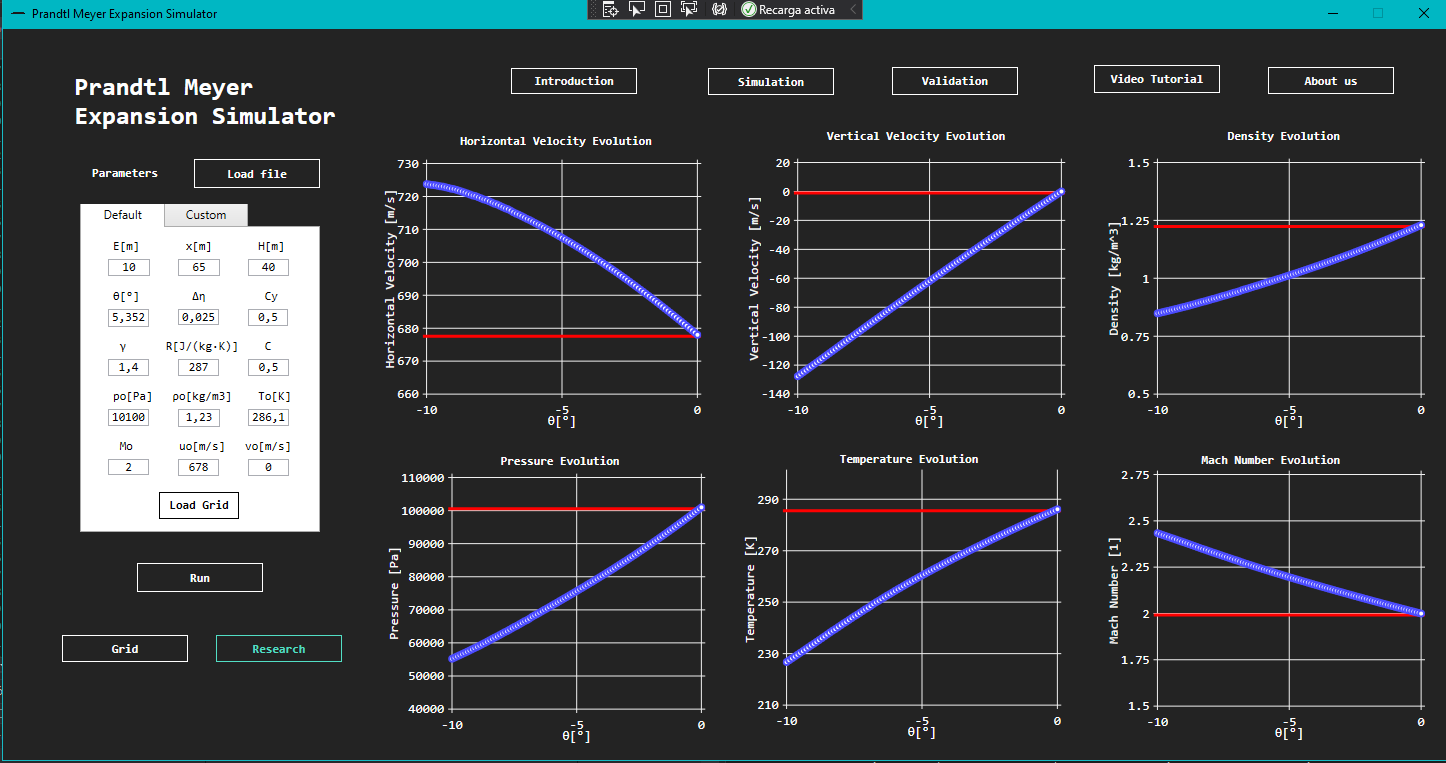
In this grid, the simulation takes place.



*Figure 6. Simulation Grid (Canvas Grid).*

* **“Load File” Button:** it loads a file previously done with the simulation data.
* **Parameters Tab Control:** it allows you to load the parameters for the simulation.
  + **Default:** parameters used in the Anderson book to do the simulation.
  + **Custom:** the user can select each initial parameter to do the simulation.
  + **“Load Grid” Button:** it creates the boundary and horizontal divisions of the canvas.
* **“Run” Button:** it calculates the Prandtl Meyer expansion with the parameters introduced and shows the result into the canvas.
* **“Canvas” Button:** it shows the canvas grid.
* **“Research” Button:** it shows the research grid.
* **Canvas Grid.**
  + **Canvas:** it shows the values of the different parameters calculated in the entire expansion corner.
  + **“Save” Button:** it saves the parameters of the simulation into a CSV file.
  + **“u, v, ρ, p, T, M” Radio Buttons:** they show the results for the variable selected into the canvas.
  + **“u, v, ρ, p, T, M, x, y” Labels:** they show the value of each variable of each cell of the canvas when the user puts the mouse over the canvas.
    1. **Research Grid.**

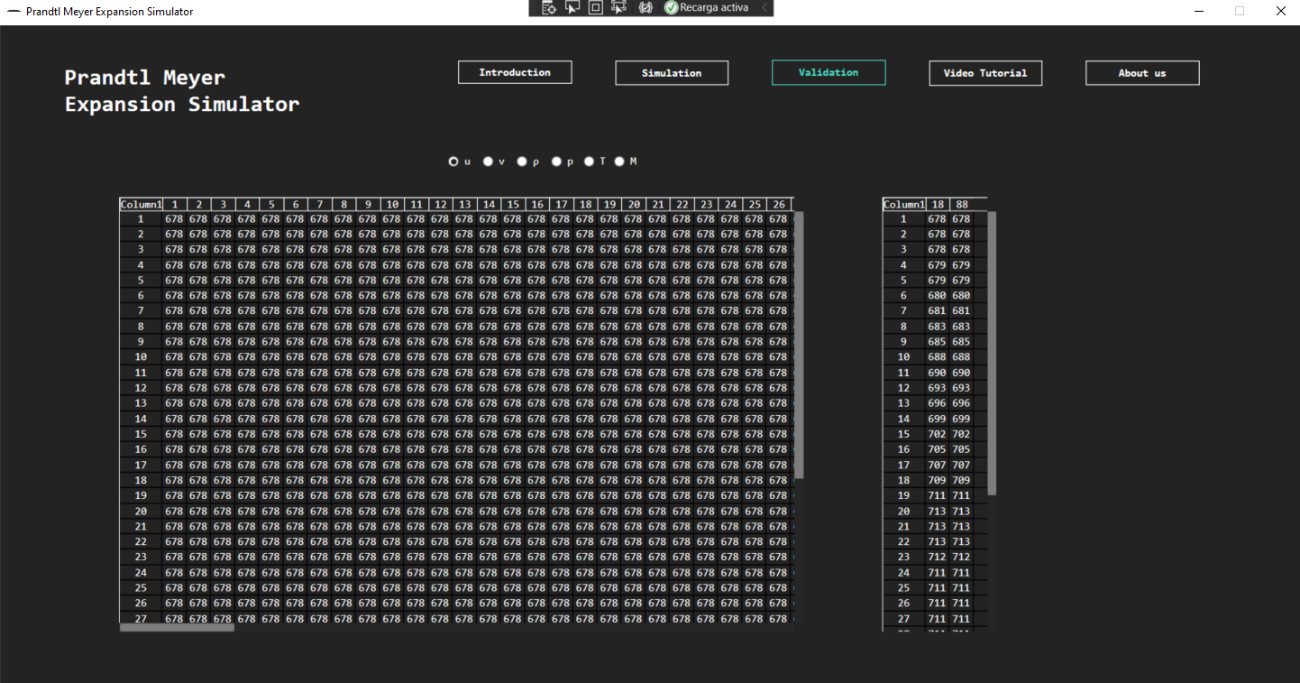
This grid contains the advanced research of the project.



*Figure 7. Simulation Grid (Research Grid).*

* **Plots:** they show the evolution of the parameters evolve in the Prandtl-Meyer expansion changing the theta angle value.
  + 1. **Validation Grid.**

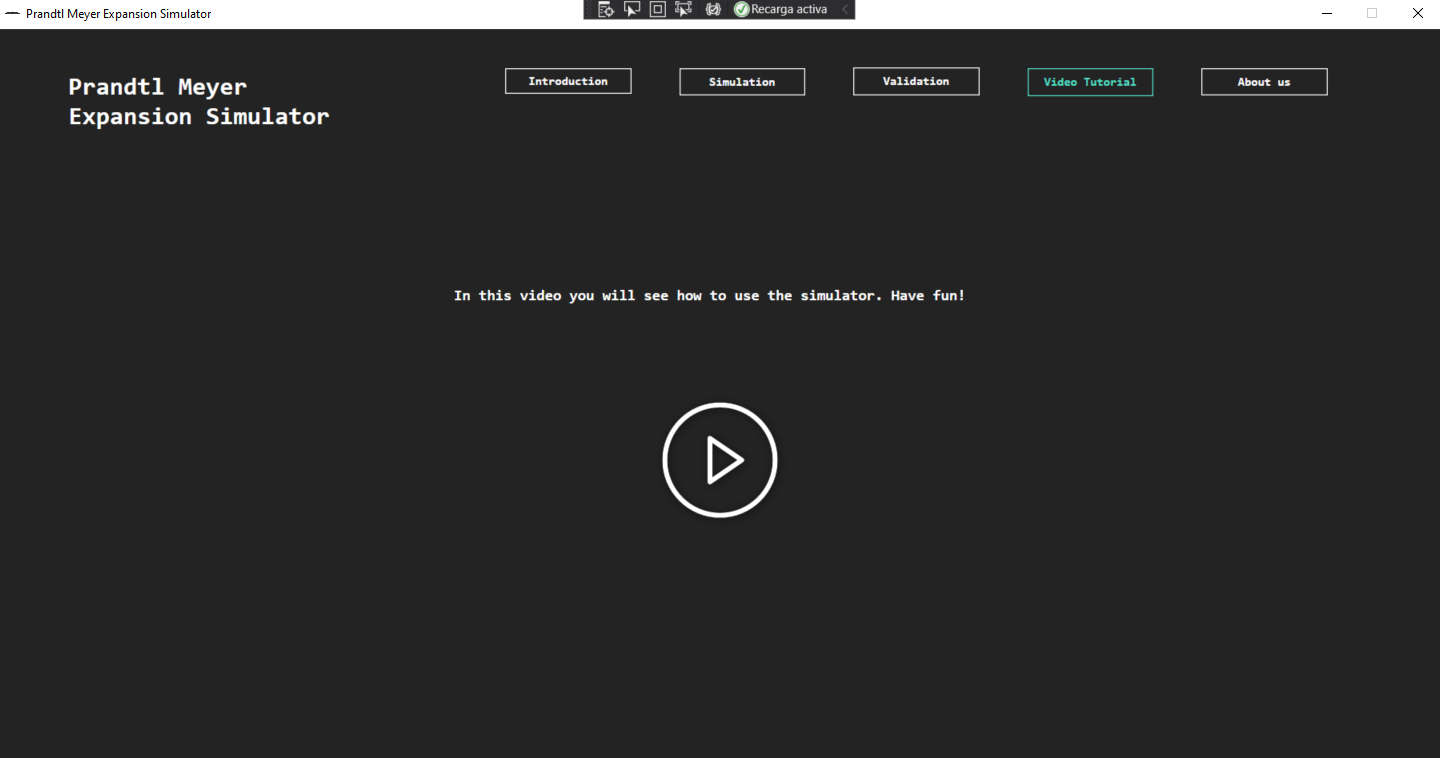
This grid allows the user to compare the results of the software with the Anderson’s CFD.



*Figure 8. Validation Grid.*

* **“u, v, ρ, p, T, M” Radio Buttons:** they show the results for the variable selected into the Data Table.
* **Big Data Table:** it contains the calculated fluid magnitudes.
* **Small Data Table:** it contains the Anderson fluid magnitudes.
  + 1. **Video Tutorial Grid.**

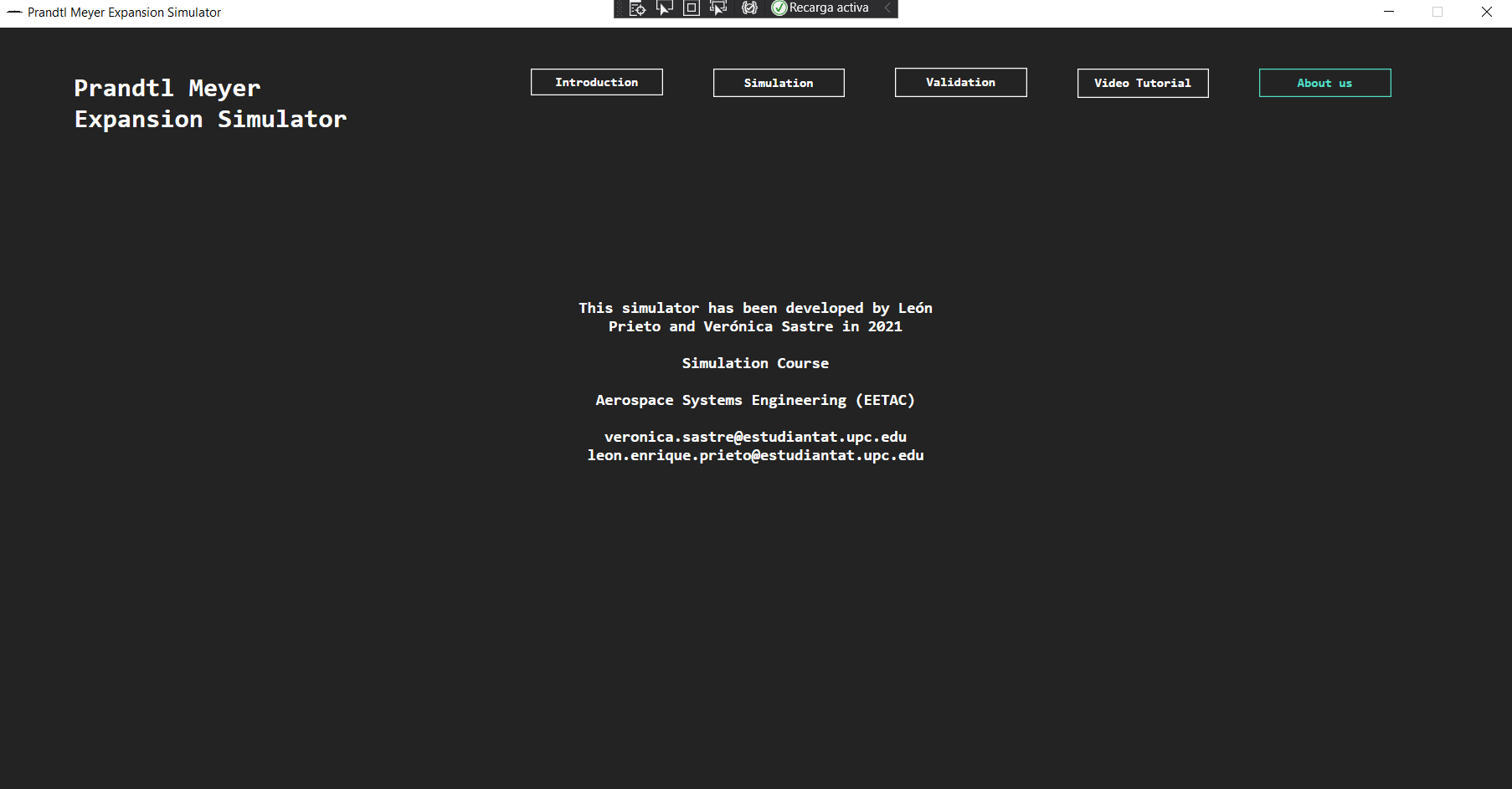
In this grid, a video of how to use the simulator will be presented.



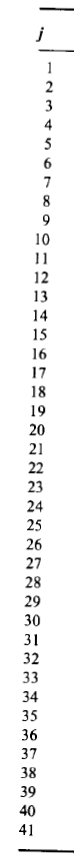
*Figure 9. Video Tutorial Grid.*

* + 1. **About us**

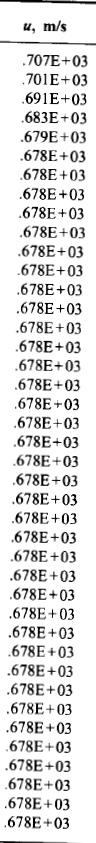
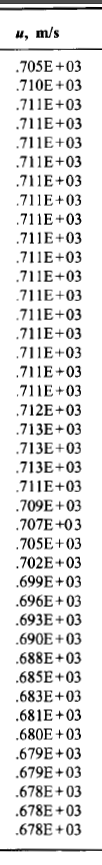
In this grid, developers contact information will be shown.



*Figure 10. About Us Grid.*

1. **Validation.**
   1. **Results verification.**
      1. **Horizontal velocity verification.**

|  |  |  |
| --- | --- | --- |
| **Results Obtained (u)** | | |
| **j** |  |  |
| **1** | 707,0175910193931 | 709,1274101137019 |
| **2** | 691,5102733839332 | 709,4755197132239 |
| **3** | 679,4003630397951 | 709,8342342754601 |
| **4** | 678,0061274046702 | 709,8077873416149 |
| **5** | 678,0000000000008 | 709,6194530937026 |
| **6** | 678,0000000000005 | 709,6990823554756 |
| **7** | 678 | 710,1695024179563 |
| **8** | 677,9999999999999 | 710,4592499967969 |
| **9** | 677,9999999999999 | 710,0151146461761 |
| **10** | 678,0000000000001 | 709,3281211368769 |
| **11** | 678 | 709,4279247948019 |
| **12** | 678 | 710,4734505741114 |
| **13** | 678,0000000000003 | 711,3020907241689 |
| **14** | 678,0000000000001 | 710,7577050249807 |
| **15** | 678,0000000000005 | 709,1523315158371 |
| **16** | 678,0000000000001 | 708,0468494490777 |
| **17** | 678 | 708,724575793118 |
| **18** | 678 | 710,9572828001355 |
| **19** | 678 | 713,1817576208108 |
| **20** | 678 | 714,1183111949113 |
| **21** | 678 | 713,5998566670103 |
| **22** | 678 | 712,0165587266041 |
| **23** | 678 | 709,7676825934487 |
| **24** | 678 | 707,1187239597331 |
| **25** | 678 | 704,2286996900648 |
| **26** | 678 | 701,1966243051612 |
| **27** | 678 | 698,0938381024791 |
| **28** | 678 | 694,9839664815751 |
| **29** | 678 | 691,9354330833953 |
| **30** | 678 | 689,0282927357667 |
| **31** | 678 | 686,3541475096165 |
| **32** | 678 | 684,0068676836984 |
| **33** | 678 | 682,0642879797228 |
| **34** | 678 | 680,5666408297302 |
| **35** | 678 | 679,5024459589597 |
| **36** | 678 | 678,8111813257566 |
| **37** | 678 | 678,4028020185356 |
| **38** | 678 | 678,1837968951883 |
| **39** | 678 | 678,0771487648168 |
| **40** | 678 | 678,0295675330162 |
| **41** | 678 | 678,0121007044002 |

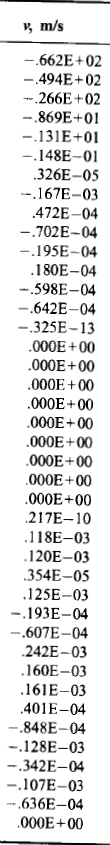
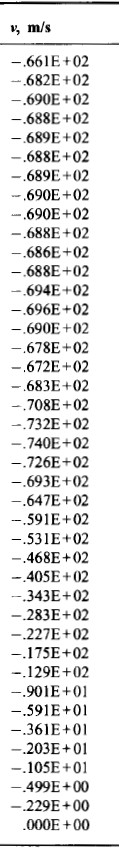
****

*Figure 12. Computed horizontal velocity.*

*Figure 11. Anderson’s horizontal velocity.*

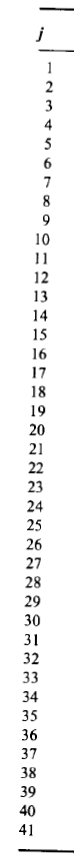
* + 1. **Vertical velocity verification.**

|  |  |  |
| --- | --- | --- |
| **Results Obtained (v)** | | |
| **j** |  |  |
| **1** | -66,23528440463622 | -66,43293785700985 |
| **2** | -14,74591418464073 | -66,56374874864568 |
| **3** | -3,4198257289445864 | -66,84976316719452 |
| **4** | -0,020965687561915747 | -66,63959367462849 |
| **5** | -7,84861250030259E-13 | -65,87886331954998 |
| **6** | -4,744898167792978E-13 | -65,93077052552152 |
| **7** | -3,510955941301523E-13 | -67,0912376470314 |
| **8** | -3,9923938012422023E-13 | -67,74019096644416 |
| **9** | -1,170183862973475E-13 | -66,5582338529275 |
| **10** | -2,4566507267495638E-15 | -64,80048494806864 |
| **11** | -8,660482269912863E-14 | -65,04877813221556 |
| **12** | -7,091839529252694E-14 | -67,69185847149433 |
| **13** | -2,91502036814555E-14 | -69,7651517804007 |
| **14** | -5,595782769072313E-14 | -68,34519505281132 |
| **15** | -1,487154255861338E-13 | -64,27503255626767 |
| **16** | -1,5594405732571295E-13 | -61,51552944802818 |
| **17** | -1,0589122906306326E-15 | -63,25164279179104 |
| **18** | 0 | -68,93605513843612 |
| **19** | 0 | -74,67862470421291 |
| **20** | 0 | -77,11309652696544 |
| **21** | 0 | -75,70207962916567 |
| **22** | 0 | -71,53103661240831 |
| **23** | 0 | -65,7844376007981 |
| **24** | 0 | -59,25755522097469 |
| **25** | 0 | -52,41078455949933 |
| **26** | 0 | -45,50815359228321 |
| **27** | 0 | -38,715687542906316 |
| **28** | 0 | -32,158902267184615 |
| **29** | 0 | -25,955128042730202 |
| **30** | 0 | -20,22932072759667 |
| **31** | 0 | -15,114997666962408 |
| **32** | 0 | -10,7393342462426 |
| **33** | 0 | -7,194948160746212 |
| **34** | 0 | -4,50876704259942 |
| **35** | 0 | -2,6244939144264476 |
| **36** | 0 | -1,4116063092752065 |
| **37** | 0 | -0,6993023344102796 |
| **38** | 0 | -0,3186646497473578 |
| **39** | 0 | -0,13366795172248086 |
| **40** | 0 | -0,05121244693439195 |
| **41** | 0 | -0,020956246605945514 |

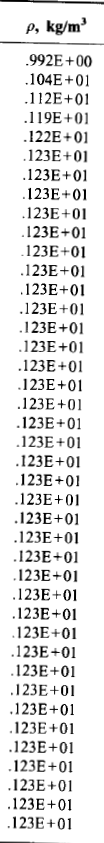
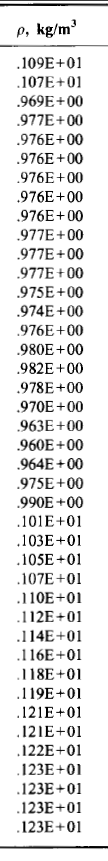


*Figure 14. Computed vertical velocity.*

*Figure 13. Anderson’s vertical velocity.*

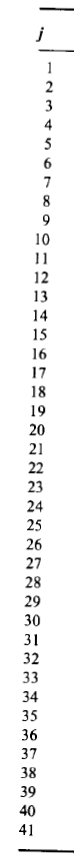
* + 1. **Density verification.**

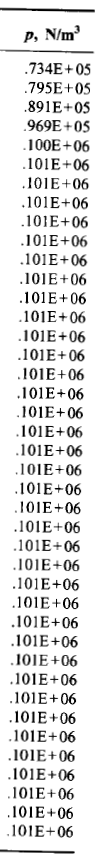
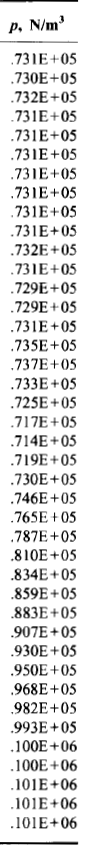
|  |  |  |
| --- | --- | --- |
| **Results Obtained (ρ)** | | |
| **j** |  |  |
| **1** | 1,0299322248642808 | 0,9998087183187994 |
| **2** | 1,1236552738728538 | 0,9809801138422459 |
| **3** | 1,2192698986992712 | 0,9811424408860641 |
| **4** | 1,2299543394354537 | 0,9827624412869433 |
| **5** | 1,2299999999999975 | 0,9853792055541374 |
| **6** | 1,229999999999998 | 0,9852226058389654 |
| **7** | 1,2299999999999995 | 0,9816724938004855 |
| **8** | 1,2299999999999995 | 0,9795924370540747 |
| **9** | 1,2300000000000002 | 0,9834963030216541 |
| **10** | 1,2299999999999995 | 0,9893179547800386 |
| **11** | 1,23 | 0,9884996267185956 |
| **12** | 1,2299999999999993 | 0,979862602344393 |
| **13** | 1,2299999999999995 | 0,9731915984408545 |
| **14** | 1,2299999999999995 | 0,9778057825990569 |
| **15** | 1,229999999999999 | 0,9911503006131691 |
| **16** | 1,2299999999999995 | 1,0003227350072932 |
| **17** | 1,23 | 0,9944892453149944 |
| **18** | 1,23 | 0,9758142760908959 |
| **19** | 1,23 | 0,9574236802067861 |
| **20** | 1,23 | 0,9497603309258087 |
| **21** | 1,23 | 0,9542302436920866 |
| **22** | 1,23 | 0,9675498930945371 |
| **23** | 1,23 | 0,9862228762662056 |
| **24** | 1,23 | 1,0078888528104737 |
| **25** | 1,23 | 1,0311526793554802 |
| **26** | 1,23 | 1,0551813964110417 |
| **27** | 1,23 | 1,0794132298588026 |
| **28** | 1,23 | 1,1033801106790426 |
| **29** | 1,23 | 1,1266002858934356 |
| **30** | 1,23 | 1,1485206091191176 |
| **31** | 1,23 | 1,1685127146534433 |
| **32** | 1,23 | 1,1859371864847115 |
| **33** | 1,23 | 1,200273502211559 |
| **34** | 1,23 | 1,2112734174790036 |
| **35** | 1,23 | 1,2190595279646723 |
| **36** | 1,23 | 1,224101884076395 |
| **37** | 1,23 | 1,2270741900937718 |
| **38** | 1,23 | 1,2286658077744235 |
| **39** | 1,23 | 1,2294401712886172 |
| **40** | 1,23 | 1,229785480896668 |
| **41** | 1,23 | 1,2299122136746885 |



*Figure 16. Computed density.*

*Figure 15. Anderson’s density.*

* + 1. **Pressure verification**

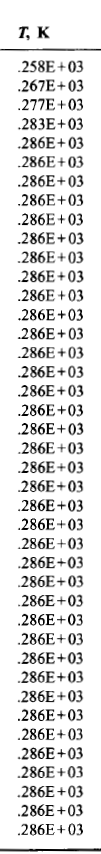
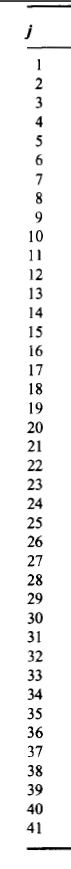


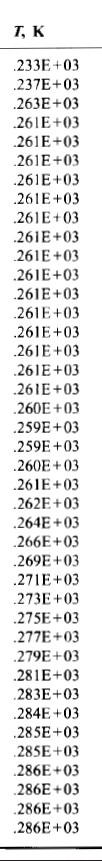
|  |  |  |
| --- | --- | --- |
| **Results Obtained (p)** | | |
| **j** |  |  |
| **1** | 78274,31885850022 | 74123,94464044756 |
| **2** | 89259,94663167186 | 73935,19573257177 |
| **3** | 99785,7378308681 | 73792,57915259013 |
| **4** | 100994,79022266401 | 73941,71781918657 |
| **5** | 100999,99999999953 | 74179,16200316098 |
| **6** | 100999,99999999965 | 74156,75444776478 |
| **7** | 100999,99999999988 | 73775,70752679021 |
| **8** | 100999,99999999988 | 73548,35874709213 |
| **9** | 101000 | 73946,24746033846 |
| **10** | 100999,99999999988 | 74552,43513482739 |
| **11** | 101000 | 74472,1072160356 |
| **12** | 100999,99999999988 | 73572,86035378726 |
| **13** | 100999,99999999988 | 72868,2162936785 |
| **14** | 100999,99999999988 | 73338,54656138335 |
| **15** | 100999,99999999977 | 74728,46393829421 |
| **16** | 100999,99999999988 | 75693,75056912511 |
| **17** | 101000 | 75097,71939550713 |
| **18** | 101000 | 73159,5270119228 |
| **19** | 101000 | 71246,81411441485 |
| **20** | 101000 | 70443,9322604987 |
| **21** | 101000 | 70894,45826899039 |
| **22** | 101000 | 72265,4339392815 |
| **23** | 101000 | 74206,24744307378 |
| **24** | 101000 | 76480,81203772384 |
| **25** | 101000 | 78948,6434923262 |
| **26** | 101000 | 81524,31179468305 |
| **27** | 101000 | 84148,28079374833 |
| **28** | 101000 | 86768,85588983772 |
| **29** | 101000 | 89330,93574515544 |
| **30** | 101000 | 91769,7804492967 |
| **31** | 101000 | 94010,66511177365 |
| **32** | 101000 | 95976,35415735643 |
| **33** | 101000 | 97602,39013558114 |
| **34** | 101000 | 98855,4028352393 |
| **35** | 101000 | 99745,24421615875 |
| **36** | 101000 | 100322,86788799998 |
| **37** | 101000 | 100663,88971685688 |
| **38** | 101000 | 100846,6751166688 |
| **39** | 101000 | 100935,65238430118 |
| **40** | 101000 | 100975,34061742376 |
| **41** | 101000 | 100989,90841494396 |

*Figure 18. Computed pressure.*

*Figure 17. Anderson’s pressure.*

|  |  |  |
| --- | --- | --- |
| **Results Obtained (T)** | | |
| **j** |  |  |
| **1** | 264,8065685664314 | 258,3209961874979 |
| **2** | 276,7844144382147 | 262,60871018489337 |
| **3** | 285,1587643059541 | 262,0587896488017 |
| **4** | 286,10673785977207 | 262,15556947438114 |
| **5** | 286,11087504603194 | 262,2989988506413 |
| **6** | 286,11087504603216 | 262,26144474136595 |
| **7** | 286,1108750460325 | 261,857407640989 |
| **8** | 286,1108750460325 | 261,6047742496926 |
| **9** | 286,1108750460326 | 261,9760027402182 |
| **10** | 286,1108750460325 | 262,5693601295956 |
| **11** | 286,1108750460327 | 262,5035833285123 |
| **12** | 286,1108750460325 | 261,61977117077146 |
| **13** | 286,1108750460325 | 260,8902752824332 |
| **14** | 286,1108750460325 | 261,3351342238284 |
| **15** | 286,1108750460322 | 262,70276361603663 |
| **16** | 286,1108750460325 | 263,6562000959598 |
| **17** | 286,1108750460327 | 263,1144869018651 |
| **18** | 286,1108750460327 | 261,22926277032377 |
| **19** | 286,1108750460327 | 259,2861908543973 |
| **20** | 286,1108750460327 | 258,4328207164855 |
| **21** | 286,1108750460327 | 258,8673121521395 |
| **22** | 286,1108750460327 | 260,240778142554 |
| **23** | 286,1108750460327 | 262,17030620814376 |
| **24** | 286,1108750460327 | 264,3978697587389 |
| **25** | 286,1108750460327 | 266,7717274209119 |
| **26** | 286,1108750460327 | 269,2018988277258 |
| **27** | 286,1108750460327 | 271,6286742988688 |
| **28** | 286,1108750460327 | 274,00394968230876 |
| **29** | 286,1108750460327 | 276,2804339162223 |
| **30** | 286,1108750460327 | 278,40626457821236 |
| **31** | 286,1108750460327 | 280,32497119171745 |
| **32** | 286,1108750460327 | 281,9815312603427 |
| **33** | 286,1108750460327 | 283,33376833120207 |
| **34** | 286,1108750460327 | 284,365121323962 |
| **35** | 286,1108750460327 | 285,0922345971137 |
| **36** | 286,1108750460327 | 285,56203982349683 |
| **37** | 286,1108750460327 | 285,8386747672337 |
| **38** | 286,1108750460327 | 285,9867516783427 |
| **39** | 286,1108750460327 | 286,0587906589997 |
| **40** | 286,1108750460327 | 286,0909163250539 |
| **41** | 286,1108750460327 | 286,1027072314482 |

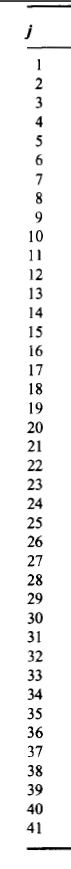
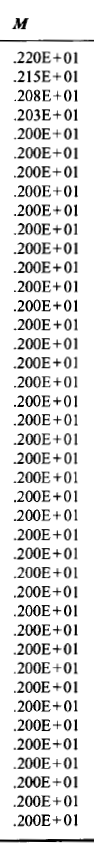
* + 1. **Temperature Verification**



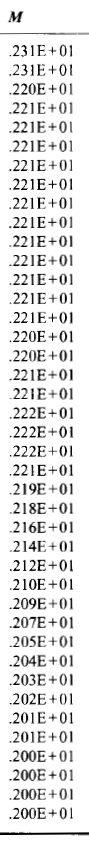
*Figure 20. Computed temperature.*

*Figure 19. Anderson’s temperature.*

* + 1. **Mach Number Verification**



|  |  |  |
| --- | --- | --- |
| **Results Obtained (M)** | | |
| **j** |  |  |
| **1** | 2,1419755995273593 | 2,2112551271915435 |
| **2** | 2,0740612318168363 | 2,1937186983536967 |
| **3** | 2,0071644121418277 | 2,1972023117438853 |
| **4** | 1,9996989103661562 | 2,1966549511741973 |
| **5** | 1,9996663795293717 | 2,1952588424455812 |
| **6** | 1,9996663795293699 | 2,1956750470177218 |
| **7** | 1,9996663795293677 | 2,199145978279324 |
| **8** | 1,9996663795293672 | 2,201286413911635 |
| **9** | 1,9996663795293663 | 2,1980204641675667 |
| **10** | 1,9996663795293679 | 2,192930719368804 |
| **11** | 1,9996663795293665 | 2,1935811584870035 |
| **12** | 1,9996663795293672 | 2,2012527776568853 |
| **13** | 1,9996663795293685 | 2,2074918900646776 |
| **14** | 1,9996663795293679 | 2,2035165409954613 |
| **15** | 1,9996663795293697 | 2,1916884406086203 |
| **16** | 1,9996663795293679 | 2,1835893021748194 |
| **17** | 1,9996663795293665 | 2,1883807378823064 |
| **18** | 1,9996663795293665 | 2,2047514763941423 |
| **19** | 1,9996663795293665 | 2,221639235925065 |
| **20** | 1,9996663795293665 | 2,228993278100987 |
| **21** | 1,9996663795293665 | 2,2250578279283504 |
| **22** | 1,9996663795293665 | 2,212982995139565 |
| **23** | 1,9996663795293665 | 2,196225457485399 |
| **24** | 1,9996663795293665 | 2,177098209684939 |
| **25** | 1,9996663795293665 | 2,156940995857123 |
| **26** | 1,9996663795293665 | 2,1365276432232423 |
| **27** | 1,9996663795293665 | 2,116351935961199 |
| **28** | 1,9996663795293665 | 2,0967944652293498 |
| **29** | 1,9996663795293665 | 2,078216871218542 |
| **30** | 1,9996663795293665 | 2,0610080093406165 |
| **31** | 1,9996663795293665 | 2,0455857451546247 |
| **32** | 1,9996663795293665 | 2,032350861860456 |
| **33** | 1,9996663795293665 | 2,0216005079949784 |
| **34** | 1,9996663795293665 | 2,0134324250902127 |
| **35** | 1,9996663795293665 | 2,0076897580298145 |
| **36** | 1,9996663795293665 | 2,0039861810525994 |
| **37** | 1,9996663795293665 | 2,001807917887871 |
| **38** | 1,9996663795293665 | 2,000642699723771 |
| **39** | 1,9996663795293665 | 2,0000760157748076 |
| **40** | 1,9996663795293665 | 1,9998233442947344 |
| **41** | 1,9996663795293665 | 1,99973061390698 |

****

*Figure 22. Computed Mach number.*

*Figure 21. Anderson’s Mach number.*

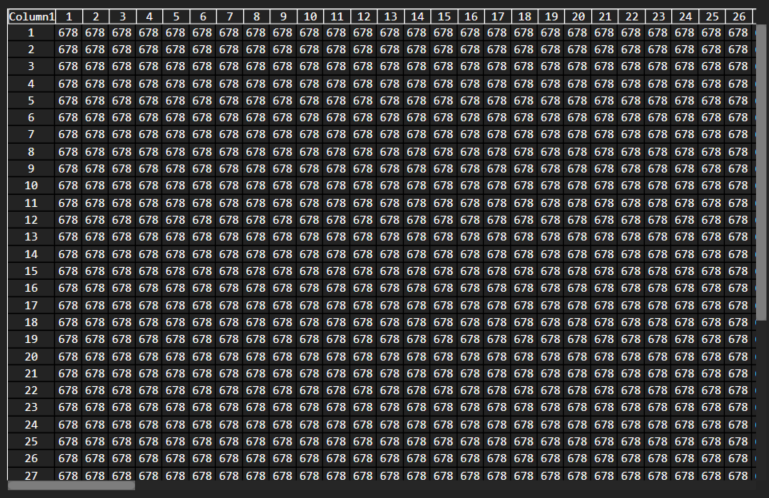
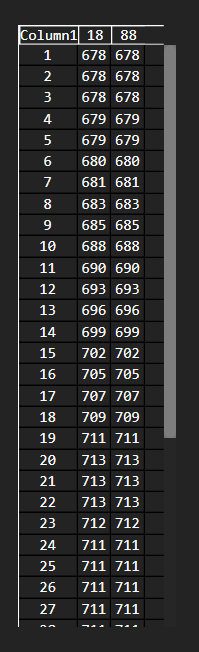
As it can be seen, the results obtained from the software are very close to the ones provided in the Anderson’s.

* 1. **How to check the results.**

The validation of the results is done in the Validation Grid. The fluid magnitude that is selected by default is the horizontal velocity but the user can change this magnitude by pressing the rest of radio buttons.

The values of the selected fluid magnitude will be shown in both Data Tables. In the Big Data Table, the values correspond to the ones obtained from the simulation and in the Small Data Table, the values correspond to the values obtained in the Anderson.

In order to check and validate the results, the user has only to select the fluid magnitude in the radio buttons and then check if the values shown in the Small Data Table are the same or similar to the ones obtained in the Small Data Table.



*Table 23. Horizontal Velocity Simulation results and Anderson’s.*

Since the Anderson only shows the values of the fluid magnitudes at and , only two columns (19 and 88) of the results obtained with the simulation can be validated comparing it with the Anderson values.

1. **Challenge**

The most challenging aspect of the project was the mathematical derivation, the workload of implementing all the equations from the Anderson CFD makes it difficult. Debugging and finding where the mistake is and why is it affecting everything else makes the development very hard since you don’t have almost anything to compare your early results with. Also, the procedure detailed in the Anderson is a bit hard to understand, it’s easy to follow in the early steps, but then it tells you to go back and add something and then go on… These successive changes make the implementation harder.

The most challenging aspects faced during the development of the software was the implementation of the canvas since each vertical division has its own length due to the variable and also due to the fact that it has a corner with a concrete angle, theta. Another difficult thing was to implement a color gradient with three colors.

References:

[1] J. Anderson, “Computational Fluid Dynamics: The Basics with Applications. 1995,” *McGrawhill Inc*, 1995.