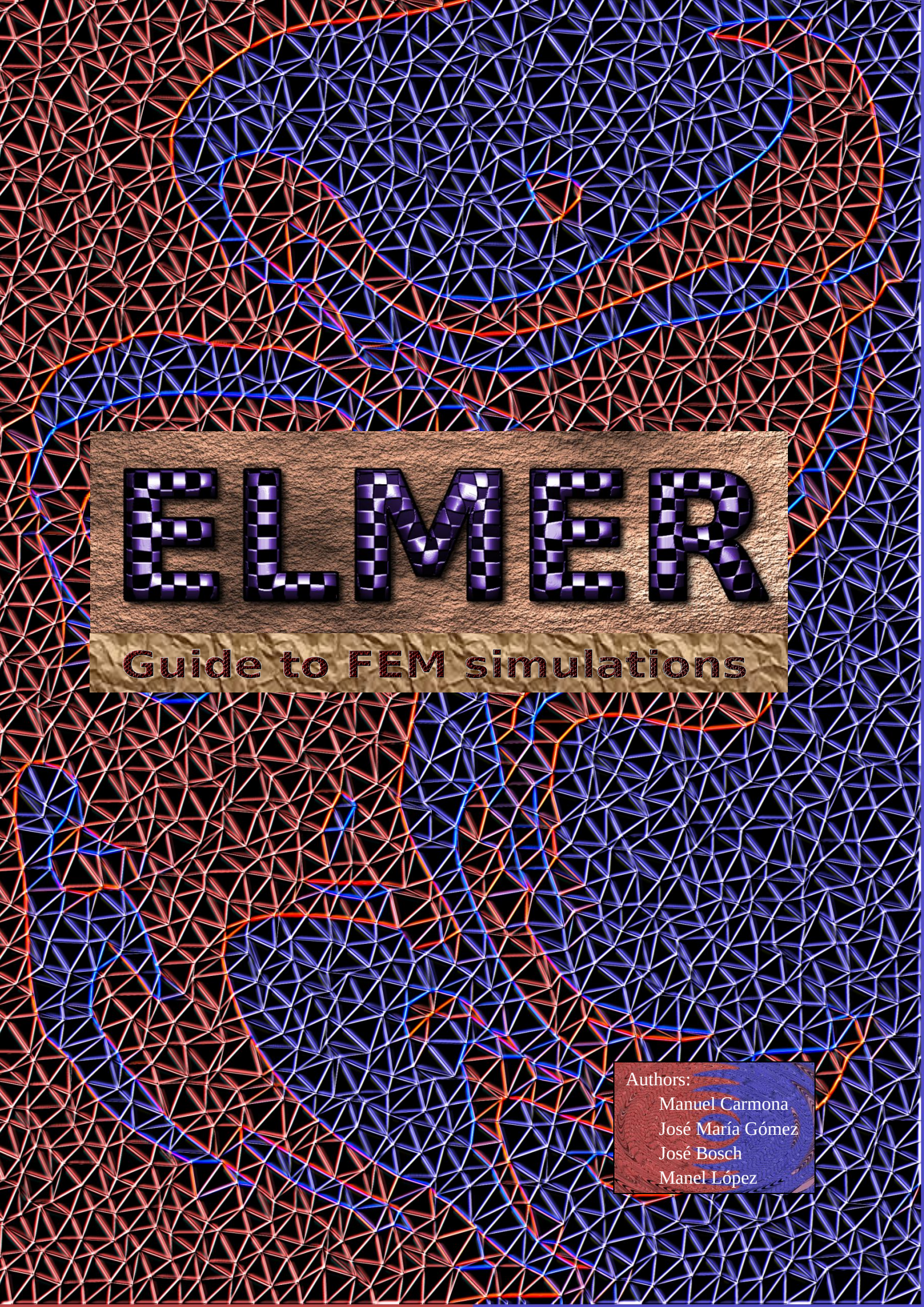

Authors:
Manuel Carmona
José María Gómez
José Bosch
Manel López

December/2014

Barcelona, Spain.

Table of contents

I. Introduction to Elmer

Elmer [1] is a combination of different softwares aimed at the simulation of multiphysics problems using the Finite Element Method (FEM). Three of these softwares are: ElmerGUI, ElmerSolver, ElmerPost. Elmer is an open source software, released under the GNU General Public License (GPL).

Elmer can be used in two different ways:

- By using its Graphical User Interface (GUI). (A command text file can be generated after a GUI session).
- By using a command text file.

Elmer has not good capabilities for geometry generation and meshing. Therefore, as a general procedure, the geometry and mesh should be imported into Elmer. It accepts different geometry and mesh formats. Among them, it accepts the GMSH mesh format.

II. Elmer GUI

Most of the things that can be done in Elmer can be accessed by its GUI. Nevertheless, there are some specific things that will have to be done by manipulating its input script (explained in later chapters).

This chapter is going to explained the most common options used to define the boundaries and loads, and the definition of the simulation parameters (type of simulation and its options).

We will assume that the geometry and the mesh of the problem has been generated by an external software. Indeed, it will be assumed that Gmsh was used for this purpose.

Before running ElmerGUI, the best thing is to create a directory, where we will create the Elmer project and put there the mesh file from Gmsh. We will import this mesh in Elmer, although once imported this file will never be used again by Elmer. Now, we can run ElmerGUI. It will appear the GUI, like shown in *Figure 1*.

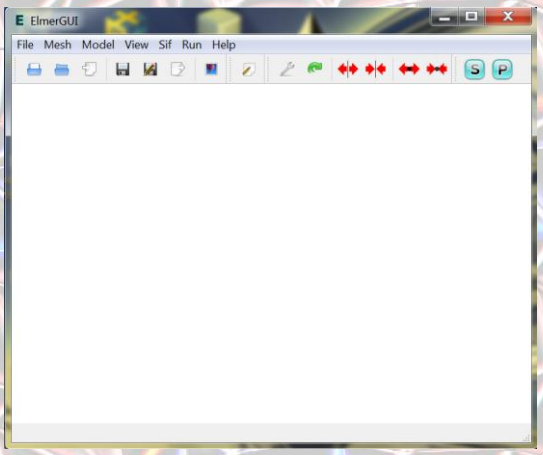

***Figure 1*. Elmer GUI.**

First, we will import the mesh file, with *File* → *Open*. You should be able to view the mesh that was generated with Gmsh:

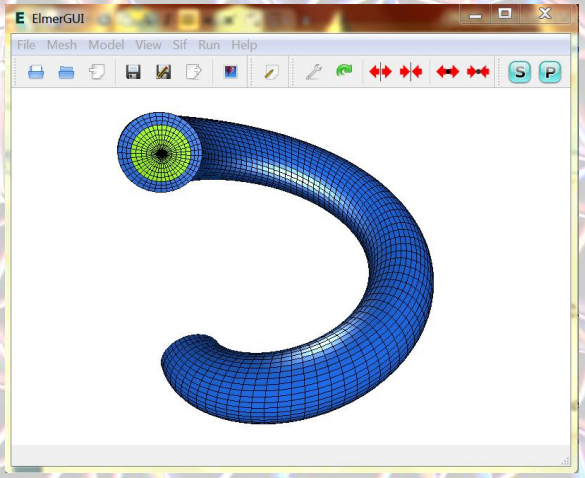

***Figure 2*. Importing mesh.**

By using the mouse, we can change the view of the model: Left (rotation), center (pan), wheel (zoom). You can reset the view parameters in *View → Model reset view*.

In order to define all the boundaries and loads of the model, we will have to go through the option of the *Model* section of the menu, as shown in *Figure 3*.

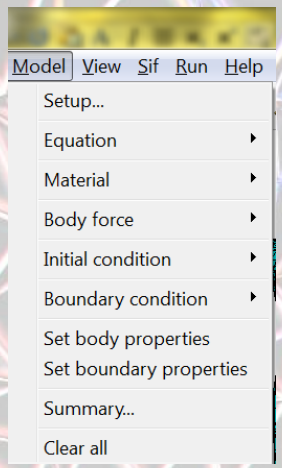

***Figure 3*. Model submenu.**

First of all, we will go to *Model → Setup*. There are several parameters that we will have to set, but most of them we will see them after at the end of this section. By now we will just change the names of the files: '*Solver input file*' and '*Post file*':

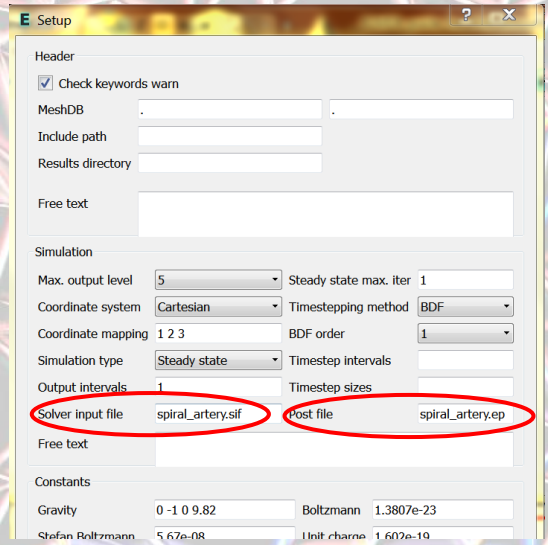

***Figure 4*. Setup.**

Maybe now it is a good moment to save the project, and we will repeat this action at the end of the process. For this, just go to *File → Save Project* and select the directory we already created previously.

The next step consists in defining which equations we want to solve in every part (body) of our model. We have to go to *Model → Equation→ Add*. It appears a window with different tabs corresponding each one to a specific equation of a specific field (fluidic, thermal, etc.). We have to activate each equation we want to solve, specifying to which body applies these activated equations. More than one equation can be solved in the same body.

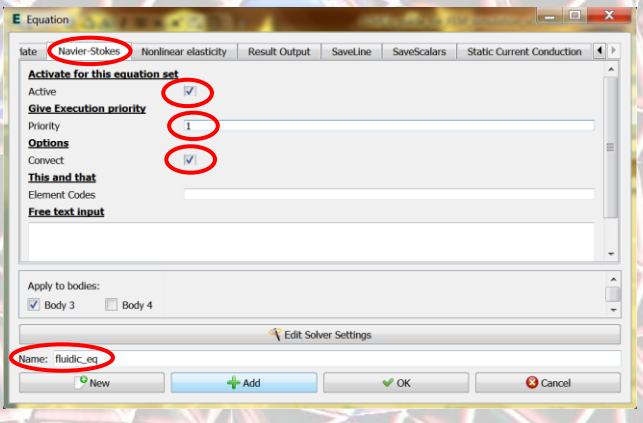

***Figure 5*. Equations setup.**

In this case, we have shown the application of the fluidic equation to Body 3 (the fluidic volume). We activate the solve for Body 3, set an appropriate name for this equation setup and change the solver parameters that apply in each case (for example, we have set the priority number to 1). We have to apply this also to the Body 4, but with another equations (like 'NonLinear elasticity', for example).

Now we can set the material properties of our bodies: blood and membrane. We have to go to *Model → Material→ Add*. In general, we have to set the corresponding properties in the General tab and in those corresponding to the activated equations tabs.

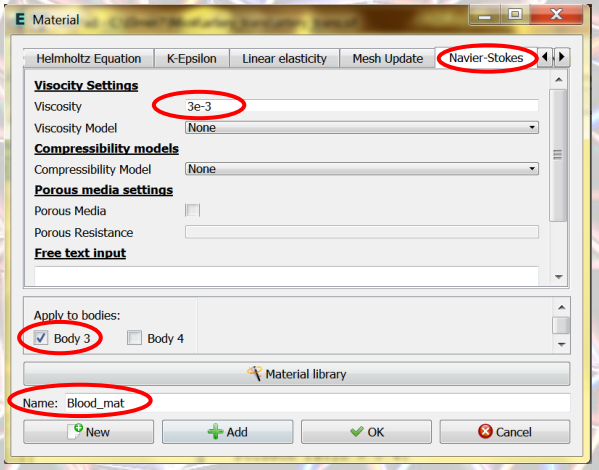

***Figure 6*. Materials setup.**

Next two steps are the definition of body forces and initial conditions. In our case, we are not going to set them.

And now we set the boundary conditions. We have to go to *Model* → *Boundary* → *Add*. If we have some doubts on the number of the surfaces, we can double-click to show at the bottom of the window which surface number has been selected.

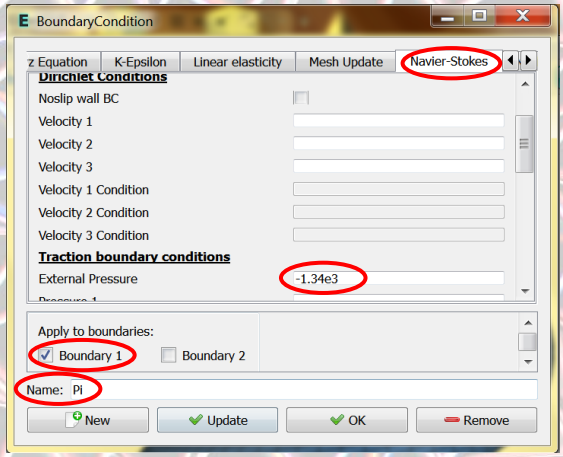

***Figure 7*. Boundaries setup.**

Here we have to apply the boundaries in all the appropriate tabs, depending on the ones that were selected for the equations for each body.

Once we have defined all these steps, we can generate the Sif file (Elmer script file) (states for Solver Input File) with *Sif* → *Generate* and save again the project.

And finally, we could already start running the solution with *Run* → *Start solver*.

If we have to modify the Sif file manually, we will have to avoid generating the SIF file again and even save the project (because it generates also the Sif file). When you open the project, try also to have de Sif file open with an editor to avoid overriding it.

III. Elmer commands

The Elmer commands file has a defined structure for defining the different aspects related to the simulation of a defined mesh. Therefore, these commands define aspects like:

- Boundary conditions.
- Loads.
- Solvers.
- Simulation parameters.

This file can be generated from zero, modified from another already existing file or create it graphically with ElmerGUI after specifying all needed options. We will make an introduction to ElmerGUI in the next chapter. In this chapter we will show some of the commands that we will find in this file.

The command file has a default extension SIF (Solver Input File). This file is organized in sections. Each section starts with the name of the section (header), followed by the commands applied to this section and it is ended by and *End* command. The different headers can be:

- *Header*: Generally just used for indicating the location of the mesh files.
- *Constants*: For defining constants.
- *Simulation*: To provide parameters related to simulation, like the type of simulation, name of output file, etc.
- *Body n*: Provides the material, body forces, equation, solver, boundary conditions and initial conditions used for simulating the body number n. This n number identifies the body in the mesh file.
- *Material n*: For defining material properties.
- *Body Force n*: For defining body forces.
- *Initial Condition n*: For defining initial conditions.
- *Boundary Condition n*: For defining a boundary condition.
- *Equation n*: For defining an equation set to be solved.
- *Solver n*: For defining a solver to be used.

All headers containing '*n*' can be found multiple times. For distinguishing each one, we have to provide a different integer number (*n*).

There are only a few things that can be put outside a section:

- *Line Comments*: They must start with a '*!*' symbol.
- *MATC expressions*: MATC allows to define mathematical expressions in Elmer. They are also used to define variables that will be used afterwards in sections. In general, MATC expressions start with a '*$*' symbol.
- *Check Keywords "Warn"*: This commands activates outputting warning messages.
- *echo on* and *echo off*.

Many commands in Elmer consist of assigning a value to an Elmer keyword, by using the equal sign ('='). For the assigned value, we can specify the type of variable (Real, Integer, Logical, String

or File) before the value is given. The value can also be a vector (example: *temps(3)= 300 320 340*) or a 2D array (example: cond(2,3) = Real 3.0 2.0 1.0 \
4.0 3.0 2.0).
(The slash is not needed in last versions of Elmer)

In some occasions, parameters can depend on variables like time, position, temperature, or on a variable that we want to solve for. These dependencies can be defined by a table or by a MATC expression. An example using a table for defining a thermal conductivity as function of temperature is the following:

```
Conductivity = Variable Temperature
  Real
        273    95.2
        273 1000
        300 1020
        400 1000
  End
```

We could do a similar thing by using a MATC expression:

```
Conductivity = Variable Temperature
  MATC "k0*(1+alpha*tx)"
```

'tx' is used for the independent variable (in this case the temperature). We could have more than one variable. In that case, we use the vector notation (example: tx(1) or tx(2))

Examples of variables that can be used are: *Time*, *Temperature*, *Pressure*, *Displacement 1*, *Coordinate 1*, *Electric Current 1*, *Magnetic Field 1*, etc.

Other language aspects:

- '::': Two semicolons are used for separating two instructions in the same line.
- *RUN*: Instruction for executing the FEM solution.

The mesh is divided in parts called bodies. For GMSH, these bodies are related to the physical groups created during the mesh generation.

Let's see more specific commands for each section.

III.1. Header section

In this section we will define the location and names of the different files related with Elmer. We can just put a command for declaring the location of the mesh files:

```
Header
Mesh DB "directory" "meshfilename"
```

End

Another example:

Header
CHECK KEYWORDS Warn
Mesh DB "." "."
Include Path ""
Results Directory ""
End

We can define more things in this section, but we will not use it. Just for mentioning them, we can indicate how many of the other sections exist in this file, we can also indicate another directory for placing the results (*Results Directory "directory"*) and include other paths if needed (*Include Path "directory"*). We can also place here the command *CHECK KEYWORDS Warn*.

III.2. Constants section

In this section we define constants if we need it. The specific model that we want to solve can require the definition of some constants. This is explained in the Models Manual of Elmer for each specific simulation field (fluidic, thermal, etc).

Constants
Gravity(4) = 0 -1 0 9.82
Stefan Boltzmann = 5.67e-08
Permittivity of Vacuum = 8.8542e-12
Boltzmann Constant = 1.3807e-23
Unit Charge = 1.602e-19
End

III.3. Simulation section

In this section we provide some parameters defining general aspects of the simulation procedure itself (independently of the specific field: thermal, mechanical, etc.) that we want to perform. For example, if the simulation is transient of stationary, coordinate system, time steps, etc. More specifically, we can define:

- *Simulation Type*: With the keywords *Transient* or *Steady State*.
- *Coordinate Mapping*: It is a vector of numbers providing the relation between the coordinates in the mesh file and the coordinates in the simulation. If they are the same, we will use *1, 2* and *3*. for a 3-dimensional case.

- *Coordinate System*: With a text keyword for defining the type of coordinate system (*Cartesian 1D*, *Cartesian 2D*, *Cartesian 3D*, *Polar 2D*, *Polar 3D*, *Cylindric*, *Cylindric Symmetric*, *Axi Symmetric*).
- *Timestepping Method*: String with five possible options: *BDF*, *Newmark*, *Implicit Euler*, *Explicit Euler* and *Crank-Nicolson*.
- *Timestep Intervals*: Vector of integers defining the number of intervals (or substeps) inside every time step.
- *Timestep Sizes*: Vector providing the size in the time units of every time step.
- *Output File*: Name of the results output file (.dat).
- *Output Intervals*: Vector of integers providing the frequency of the obtained results that will be saved to the output file.
- *Post File*: Name of the results file that is understood by Elmer Post. (.ep)
- *Steady State Max Iterations*: Maximum number of iterations of every time calculation in order to get a converged solution.

```
Simulation
  Max Output Level = 5
  Coordinate System = Cartesian
  Coordinate Mapping(3) = 1 2 3
  Simulation Type = Steady state
  Steady State Max Iterations = 1
  Output Intervals = 1
  Timestepping Method = BDF
  BDF Order = 1
  Solver Input File = tube.sif
  Post File = tube.ep
End
```

III.4. Body section

It is used to define, for each body of the mesh file (or created with Elmer), which other sections apply for it. We have to specify which *Equation*, *Material*, *Body Force* and *Initial Condition* sections applies.

```
Body 1
  Target Bodies(1) = 1
  Name = "Body 1"
  Equation = 1
  Material = 1
```

End

III.5. Material section

Here we define the properties of the material. The properties that we have to define depends on the kind of simulation that we want to perform, like for example thermal or fluidic. These properties have to be checked on the Elmer Models Manual.

Material 1
Name = "Fluido"
Viscosity = 1.0
Density = 1e3
End

III.6. Body Force section

In this section we specified the loads applied. The loads that we have to specify depends on the kind of simulation. We have to obtain this information from the Elmer Models Manual.

III.7. Initial Condition section

Similary to body forces, we have to check the Elmer Models Manual to obtain the initial conditions that can be applied to our specific simulation.

III.8. Boundary Condition section

In this section, we define first the boundary where it will be applied (assigning their number to the vector *Target Boundaries*). And afterwards, we define the boundaries conditions that apply to them. The possible options have to be found in the Elmer Models Manual.

Boundary Condition 1
Target Boundaries(1) = 2
Name = "Output P"

```
  Pressure 1 = 0
  Pressure 3 = 0
  Pressure 2 = 0
End
```

III.9. Equation section

Each equation is related to a specific physical model (thermal, fluidic, etc). In this section we have to indicate which equation(s) will apply to a body. It can be one, or more than one. We refer to each equation by using the number(s) of the respective solver sections. We associate these numbers to the *Active Solvers* vector.

```
Equation 1
  Name = "Fluidic equation"
  NS Convect = False
  Active Solvers(1) = 1
End
```

III.10. Solver section

Here we specify one physical model to be solved and some options related to this physical model and which method will be used to solve it (solver). The name of each equation and the different options can be obtained in the Elmer Models Manual. The options for solvers and their options can be found in the Solvers Manual.

General options in this section are:

- *Variable = Variable_name*: For defining a variable with name Variable_name. A vector can also be defined with names of subcomponents. Example: *Variable = vp[Vel:3, P:1]* (defines a variable called *vp* with four components, the first three are called *Vel* and the fourth component is called *P*).
  If we only need to specify a variable with three components: *Variable = -dofs 3 vp*.
- We can define when to execute a solver. By default, if we have more than one solver, they are executed in the order that they are defined. With the command *Exec Solver order* we can change this. *order* is a string that can have values of: *never*, *always*, *before timestep*, *after timestep*, *before all*, *after all*, *before saving* and *after saving*.

```
Solver 1
  Equation = Navier-Stokes
  Procedure = "FlowSolve" "FlowSolver"
```

```
  Variable = Flow Solution[Velocity:3 Pressure:1]
  Exec Solver = Always
  Stabilize = True
  Bubbles = False
  Lumped Mass Matrix = False
  Optimize Bandwidth = True
  Steady State Convergence Tolerance = 1.0e-5
  Nonlinear System Convergence Tolerance = 1.0e-7
  Nonlinear System Max Iterations = 20
  Nonlinear System Newton After Iterations = 3
  Nonlinear System Newton After Tolerance = 1.0e-3
  Nonlinear System Relaxation Factor = 1
  Linear System Solver = Iterative
  Linear System Iterative Method = BiCGStab
  Linear System Max Iterations = 500
  Linear System Convergence Tolerance = 1.0e-10
  Linear System Preconditioning = ILU0
  Linear System ILUT Tolerance = 1.0e-3
  Linear System Abort Not Converged = False
  Linear System Residual Output = 1
  Linear System Precondition Recompute = 1
End
```

III.11. Examples

i) Flow through a circular tube

```
Header
  CHECK KEYWORDS Warn
  Mesh DB "." "."
  Include Path ""
  Results Directory ""
End

Simulation
  Max Output Level = 5
  Coordinate System = Cartesian
  Coordinate Mapping(3) = 1 2 3
  Simulation Type = Steady state
  Steady State Max Iterations = 1
```

```
  Output Intervals = 1
  Timestepping Method = BDF
  BDF Order = 1
  Solver Input File = tube.sif
  Post File = tube.ep
End

Constants
  Gravity(4) = 0 -1 0 9.82
  Stefan Boltzmann = 5.67e-08
  Permittivity of Vacuum = 8.8542e-12
  Boltzmann Constant = 1.3807e-23
  Unit Charge = 1.602e-19
End

Body 1
  Target Bodies(1) = 1
  Name = "Body 1"
  Equation = 1
  Material = 1
End

Solver 1
  Equation = Navier-Stokes
  Procedure = "FlowSolve" "FlowSolver"
  Variable = Flow Solution[Velocity:3 Pressure:1]
  Exec Solver = Always
  Stabilize = True
  Bubbles = False
  Lumped Mass Matrix = False
  Optimize Bandwidth = True
  Steady State Convergence Tolerance = 1.0e-5
  Nonlinear System Convergence Tolerance = 1.0e-7
  Nonlinear System Max Iterations = 20
  Nonlinear System Newton After Iterations = 3
  Nonlinear System Newton After Tolerance = 1.0e-3
  Nonlinear System Relaxation Factor = 1
  Linear System Solver = Iterative
  Linear System Iterative Method = BiCGStab
  Linear System Max Iterations = 500
  Linear System Convergence Tolerance = 1.0e-10
  Linear System Preconditioning = ILU0
  Linear System ILUT Tolerance = 1.0e-3
  Linear System Abort Not Converged = False
  Linear System Residual Output = 1
```

```
  Linear System Precondition Recompute = 1
End

Equation 1
  Name = "Fluidic equation"
  NS Convect = False
  Active Solvers(1) = 1
End

Material 1
  Name = "Fluido"
  Viscosity = 1.0
  Density = 1e3
End

Boundary Condition 1
  Target Boundaries(1) = 2
  Name = "Output P"
  External Pressure = 0
End

Boundary Condition 2
  Target Boundaries(1) = 3
  Name = "wall"
  Noslip wall BC = True
End

Boundary Condition 3
  Target Boundaries(1) = 1
  Name = "Input vz"
  Velocity 3 = 1.0e-3
End
```

IV. ElmerPost

Although Elmer comes with two post-processors (ElmerPost and VTK), we will just keep viewing results with ElmerPost. We can run ElmerPost through the ElmerGUI (*Run → Start postprocessor*) or directly through its icon. In the first case, the current results file will be read directly, while in the second case we will have to read the results file (.ep). After running the postprocessor, we will get two windows: the graphics window and the commands window (*Figure 8*). Here we are just going to explain some key aspects about using ElmerPost. Additionally, we can see results in other post-processing softwares, like ParaView [2].

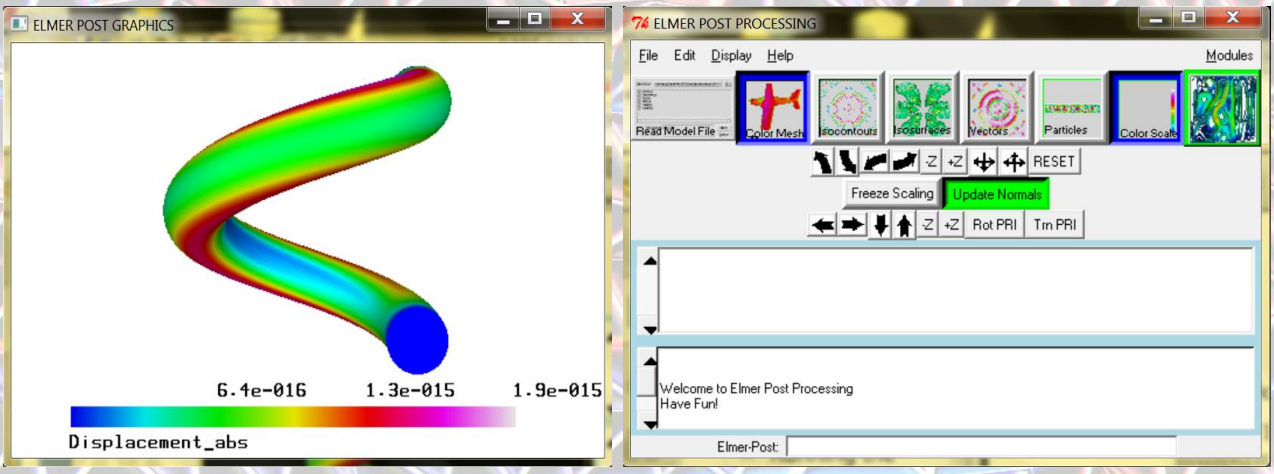

***Figure 8*. Post-processor windows.**

IV.1. Graphics window

In this window, we can do the typical mouse actions that can be expected: rotation (right), zoom (left & right), Pan (left).

In the commands window, we have some buttons controlling the view (Graphics commands), shown in *Figure 9*.

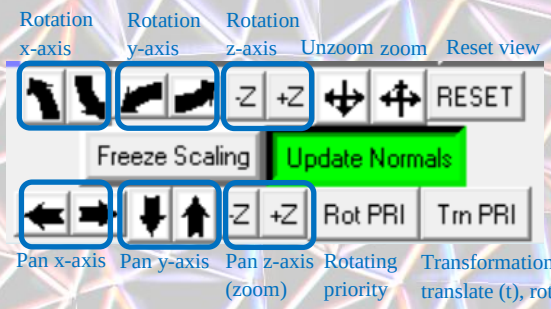

***Figure 9*. Graphics commands.**

By default, when something is plotted in the graphics window, the view is rescaled in order to fit it in the window (this is referred as 'Update Normals'). With the 'Freeze Scaling' option, we can avoid changing the scale, keeping the last used scaling factor.

IV.2. Commands window

We have several regions on this commands window, as shown in .

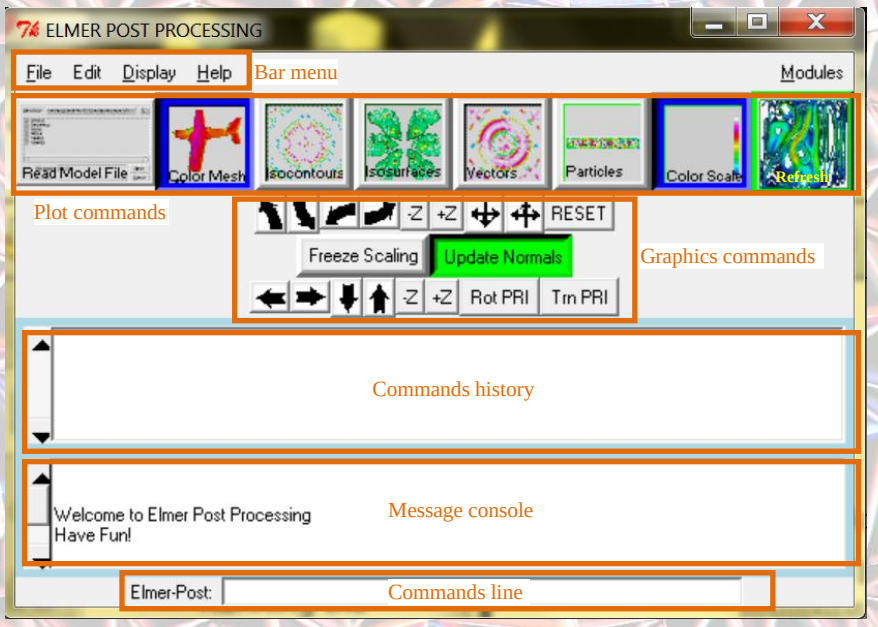

***Figure 10*. Commands window.**

In the bar menu, we can access most of the ElmerPost functionalities and configuration parameters. The different submenus are shown in *Figure 11*.

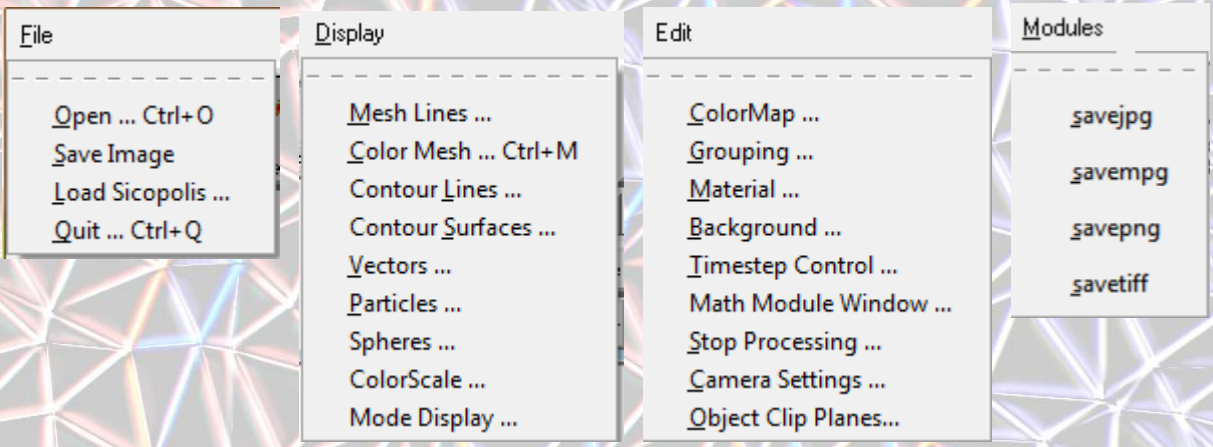

***Figure 11*. Bar submenus.**

In the plot commands buttons, we have mainly different ways to visualize the obtained results from simulation. Nevertheless, the first button corresponds to the reading of the results file (generally with extension ep, although other formats are also accepted). The options for reading the file is shown in *Figure 12*.

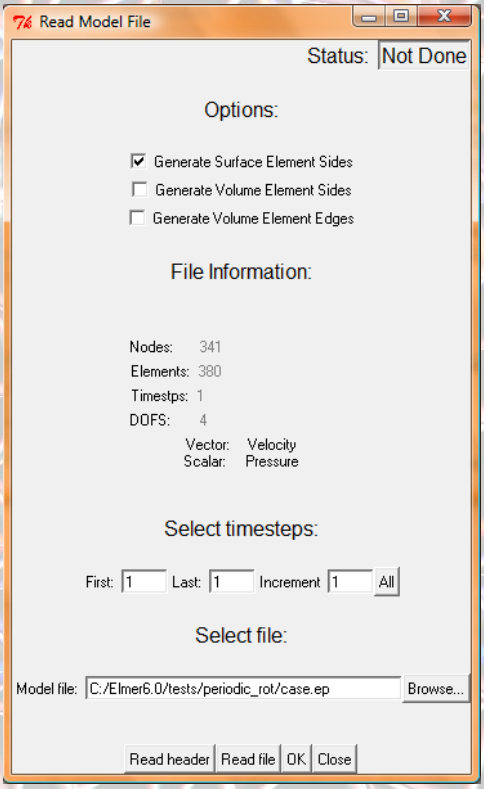

***Figure 12.* Read model.**

The first way to visualize results is called 'Color Mesh'. Basically, it paints the mesh with colors depending on the values of the results obtained from a variable. In *Figure 13*, it is shown the different options. In 'Color Variable', we can choose which variable to use for painting.

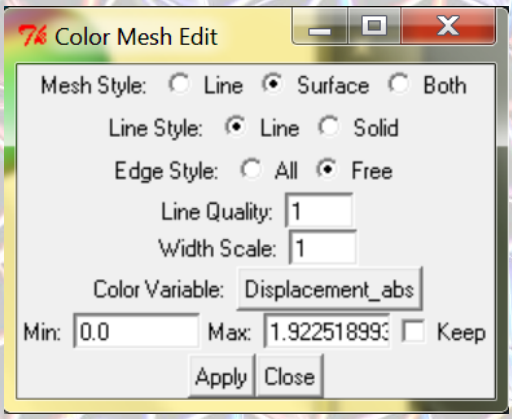

***Figure 13.* Color mesh.**

An example is shown in *Figure 14*.

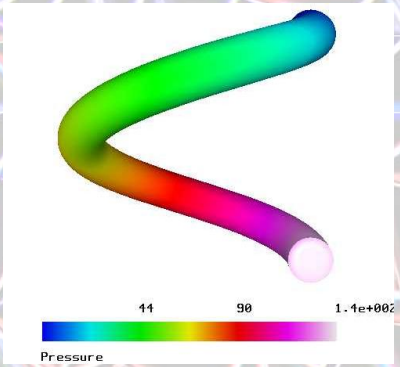

Figure 14. **Color mesh example (pressure distribution).**

With isocontours, we can observe lines having the same value of a certain variable. With isosurfaces, we can obtain surfaces having the same variable value. Isosurfaces can be used for obtained cross-sections of the model, defining the contour variable as position and another variable for 'Color Variable'.

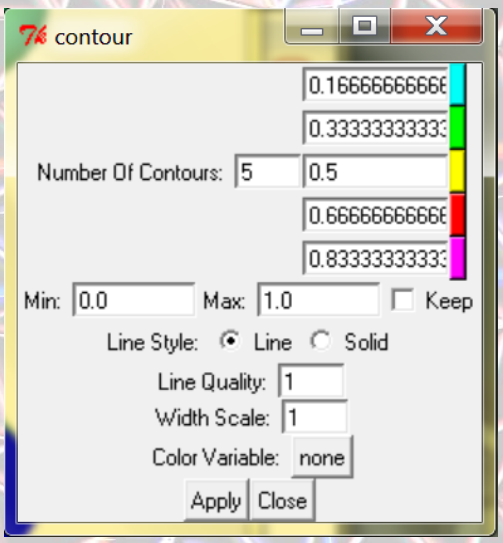

Figure 15. **Isocontours.**

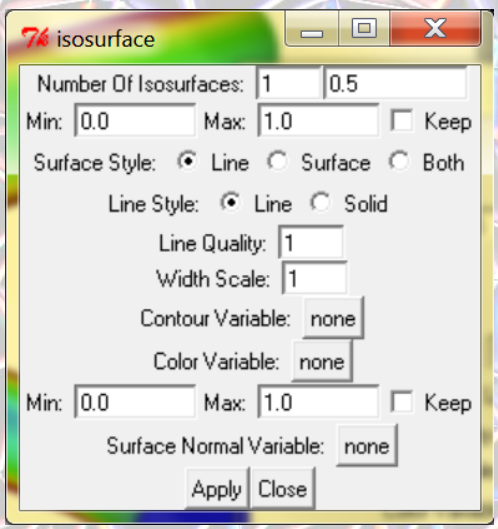

Figure 16. **Isosurface.**

An example for isosurfaces can be seen in *Figure 17*.

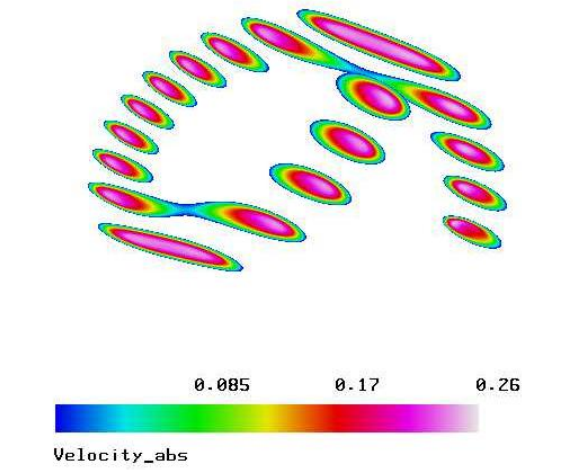

***Figure 17*. Isosurface example (absolute velocity).**

Many times, we will want to draw vector magnitudes as vectors. This can be done with the 'Vectors' button.

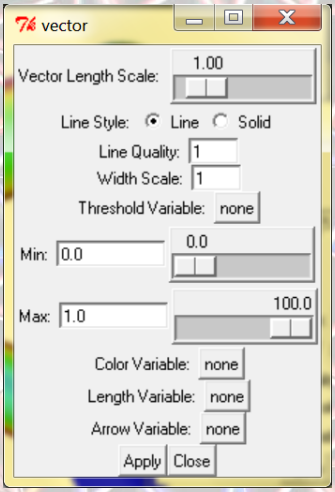

***Figure 18*. Vectors.**

An example is shown in *Figure 19*, showing the velocity vectors.

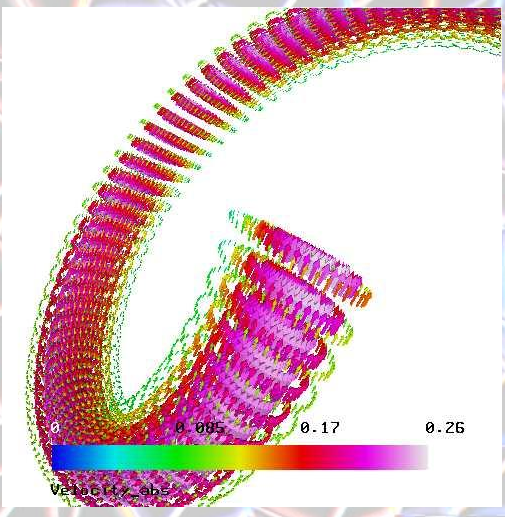

***Figure 19*. Vectors example (velocity).**

The particles button allows us to obtain the trajectories of particles within a fluid. For this purpose we have to set a variable ('*Particle Variable*') containing the initial positions of the particles to be considered. This variable is a matrix, where the second index is referred to the particle number (starting from 0). The first index is for the different parameters that have to be provided for each particle. This parameters are: x, y, z, initial color guess (for example, 0), guess of the initial element number containing the particle (for example 0).

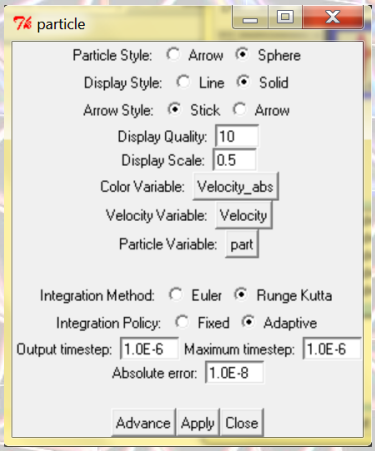

***Figure 20*. Particles.**

As an example, we have used a simple 2D axisymmetric fluidic problem. We have defined the '*Particle Variable*' named as *part* in the following way:

```
math nps=5;
math rad=5e-4;
do i 0 (nps-1) {
        math posx=$i*rad/nps;
        math part(0,$i)=posx;
        math part(1,$i)=0;
        math part(2,$i)=0;
        math part(3,$i)=0;
        math part(4,$i)=0;
        }
```

Click '*Apply*' and then '*Advance*' repeatedly in order to obtain the particle positions as time goes on. One intermediate result is shown in *Figure 21*.

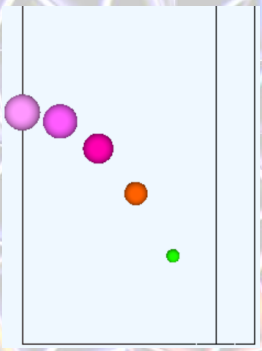

***Figure 21*. Particles position after some time, starting all at the input.**

The color scale, provided with the colored bar in the graphics window, can be setup with the 'Color Scale' button.

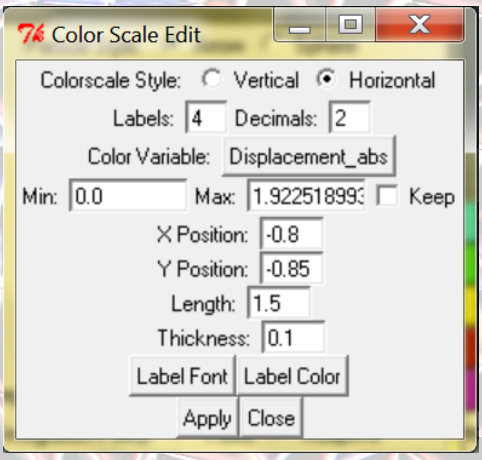

***Figure 22*. Color scale.**

The graphics commands are used to change orientation, pan, zoom, etc. We can also fix the scale of the plotted results (with the 'Freeze Scaling' button).

We have a commands line. We can introduce commands in the TCL/TK language and also in the MATC language. MATC is used for managing mathematical calculations using matrices (generally filled with the results data). In this last case (MATC) we have to start the command with '*math*'. We are just going to use it for creating some plots. A more detailed explanation is given in a next section.

Additionally, we have a console where the commands history is plotted and another console for messages.

IV.3. Commands line

As mentioned previously, we have a commands line where we can add instructions in TCL/TK language and also in MATC language. These commands can be entered manually, or included in a file, executing it with the command '*source file*'. The first language (TCL/TK) is an "external" language, quite used in some other softwares. Detailed information can be found in books and through internet. The second language (MATC) is a quite specific language for Elmer, that is used for performing calculations with matrices. Elmer documentation comes with a MATC tutorial. We are just going to comment some of the most important aspects of these two languages and apply it to some examples.

Before starting looking at these two languages, we have to know that, when results are read by ElmerPost, the results are saved in matrices. The names of the variables can be seen in the "Read Model" window. As an example, for a fluidic simulation we will have the variables *Velocity* and *Pressure*. Obviously, these matrices contains the velocity and pressure results. They have as many

rows as components (*Velocity* has three rows, each corresponding to the velocities in directions x, y and z respectively, while *Pressure* only have one row). And they have as many columns as nodes we have in the model. Depending on the type of simulation (for example, for transient simulations) there can be more indexes. The relationship between the nodes and the column index is given by another matrix called *nodes*. This matrix contains the position of the nodes of our model. It has three rows corresponding to the x, y and z positions. And it has as many columns as nodes in our model. If we modify the nodes matrix, we will change the position of the nodes, and so it will be shown in the graphics windows. This can be useful sometimes for a more convenient visualization of the model/results, as we will see later.

Let's see some basics of the TCL/TK programming language:

- Comments: #
- Command syntax: *command arg1 arg2 arg3 ... .*
- Variables:
  - Not declared.
  - Assign a value: *set var val*.
  - In order to obtain the value of a variable, we have to use *$var*.
  - Command *info* to obtain information about variables (among other things).
  - Text strings can be grouped with quotes (*""*) or rounded brackets (*{}*).
  - All variables are strings. If not, they have to be indicated explicitly with the command *expr*.
- Expressions can be grouped with square brackets (*[]*).
- Mathematical functions: abs(x), acos(x), asn(x), atan(x), atan2(y,x), ceil (x), cos(x), cosh(x), double(x), exp (x), floor(x), fmod(x,y), hypot(x,y), int(x), log(x), log10(x), pow(x,y), rand(), round(x), sin(x), sinh(x), sqrt(x), srand(arg), tan(x), tanh(x).
- Conditional*: if condition then action.*
- For loop: *for {inicialization} {condition} {increment} instruction*.
- Foreach loop: *foreach variable list instruction*.
- While loop: *while condition instruction*.

MATC is a library used for evaluating mathematical expressions with matrices. These expressions can be used in the SIF file, also can be evaluated during simulation, and it can also be used in ElmerPost. Some basic aspects of MATC are:

- Two types of variables: matrices and strings.
- Ranges can be specified: Examples: 0:5, 5:0.
- Conditional if: *if (expr) expr; else expr;*.
- For loop: *for (i=vector) expr;*.
- While loop: *while(expr) expr;*.
- Function:

```
function name(arg1,arg2,...)
! Optional function description (seen with help("name"))
        import var1     #import global variable
        export var2     #convert a local variable to global
        expr;
```

_name = value #Returned value

- Operators: ’, @, ~, ^, *, #, /, +, -, ==, <>, <, >, <=, >=, :, &, |, ?, %, =.
- Functions: funcdel(name), funclist(name), sprintf(fmt[,vec]), sscanf(str,fmt), matcvt(matrix, type), cvtmat(special,type), eval(str), **source(name)**, help or help(”symbol”), fread(fp,n), fscanf(fp,fmt), fgets(fp), fwrite(fp,buf,n), fprintf(fp,fmt[,vec]), fputs(fp,str), fopen(name, mode), freopen(fp,name,mode), fclose(fp), save(name,a[,ascii_flag]), load(name), min(matrix), max(matrix), sum(matrix), trace(matrix), det(matrix), inv(matrix), tril(x), triu(x), eig(matrix), jacob(a,b,eps), lud(matrix), hesse(matrix), eye(n), zeros(n,m), ones(n, m), rand(n,m), diag(matrix) or diag(vector), **vector(start,end,inc)**, **size(matrix)**, resize(matrix,n,m), where(a), exists(name), **who**, format(precision).
- Mathematical functions: abs(x), acos(x), asin(x), atan(x), ceil(x), cos(x), cosh(x), exp(x), floor(x), ln(x), log(x), pow(x,y), sin(x), sinh(x), sqrt(x), tan(x), tanh(x).

V. Types of simulations

V.1. Transient

We can divide a transient analysis in diferent timesteps, each one with a total time size. For defining the total time size of every timestep we use the '*Timestep Sizes'* command. Each timestep will have a number of substeps. The number of substeps is defined with the '*Timestep Intervals*' command. Both commands are vectors. We have to take into account that timesteps refers to increase of time. As an example, if we want to make a transient simulation that is lasting 1s in total but with two timesteps, the first one lasting 1s second (solved every 0.1s) and the second timestep arriving up to 10s (solved every 1s) we could set:

```
Timestep Sizes(2) = 0.1 1
Timestep Intervals(2) = 10 9
```

We can set a non-uniform time step by using the command '*Timestep Size*' with *TimeStep* as variable (this variable takes the values from 1 up to the number of intervals). Example:

```
Timestep Size = Variable TimeStep
  Real MATC "t0*1.05^(tx-1)"
```

For setting parameters depending on time, we have to create tables, with the variable *Time*.

```
Body Force 1
  Stress Bodyforce 2 = Variable Time
    Real
     0.0  -1.0
     4.0  -1.0
     4.01  0.0
     5.0   0.0
    End
End
```

Initial conditions are generally needed for transient simulation. Nevertheless, usually these conditions are taken zero.

V.2. Coupled simulations

Coupled simulations refer to the problems where more than one field interacting each other has to be solved in a common domain or in the boundary between two domains. They are specified in solvers and boundary conditions.

V.3. Axisymmetric models-simulations

In Elmer, the axis of rotation is always the y axis. We have to take it into account for building the model. Also, we have to add the boundary conditions associated with the rotation axis.

Acronyms:

2D: Two-dimensional

3D: Three-dimensional

CAD: Computer Aided Design

FEM: Finite Element Method

GPL: General Public License

---

[1] http://www.csc.fi/english/pages/elmer.
[2] http://www.paraview.org.