Dynamisk Lab Test Bench Digital Twin

-

Actuators/FEA FMUs User Manual

Table des matières

[I. Some explanations, mechanical engineering, etc. 3](#_Toc111105156)

[a. The actuators 4](#_Toc111105157)

[i. The inputs 4](#_Toc111105158)

[ii. The outputs 4](#_Toc111105159)

[iii. Relations between inputs and outputs 4](#_Toc111105160)

[b. The cantilever beam 5](#_Toc111105161)

[i. The inputs 5](#_Toc111105162)

[ii. The outputs 6](#_Toc111105163)

[iii. Relations between inputs and outputs 6](#_Toc111105164)

[III. Further ideas to explore 17](#_Toc111105165)

1. Une image contenant intérieur

   Description générée automatiquementUne image contenant plancher, intérieur, fraise

   Description générée automatiquementSome explanations, mechanical engineering, etc.

So, here’s the test bench, and we split it into 3 parts:

* The 2 pistons linear actuators
* The cantilever beam
* The frame

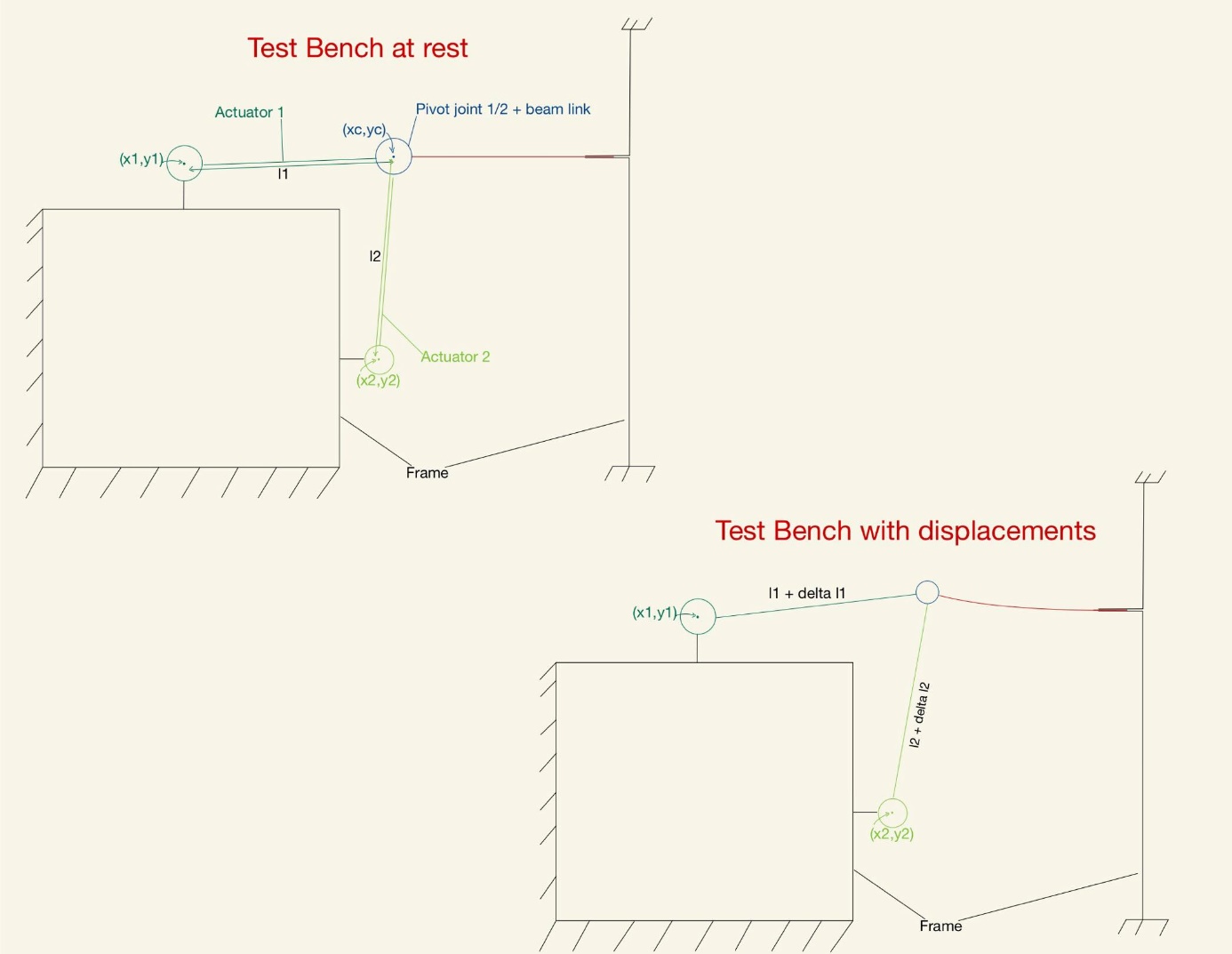
The frame was not studied, because it only serves as load, energy disperser.

The step-by-step analyses run on the digital shadow are static, meaning no inertia and damping are taken into account.

Same thing for gravity, supposing that the forces acted upon the beam are largely superior to that of the force of gravity on the beam(Weight of beam 95N approx.<<Actuator forces between 5.102 and 1.107 N). To be discussed with small vertical displacements though.

Une image contenant intérieur

Description générée automatiquementHere is a simplified planar schematic of the test Bench:

1. The actuators

Let’s first discuss parameters related to the actuators. Note that the following mentions of coordinates follow the general cartesian coordinates system.

1. The inputs
   * (x1,y1), (x2,y2)are the coordinates of the center of the pivot joints linking the actuators to the frame. Indexes 1 and 2 related to the horizontal and vertical actuators respectively
   * (xc,yc) represents the coordinates of the center of the pivot joint liking the actuators to the beam, when both are at rest. It is where the displacement on the beam is acted upon, as ux and uy, respectively horizontal and vertical displacement
   * l1 and l2 are the resting lengths of the actuators
   * Δ l1 and Δ l2  are the resulting differences between the resting length and the step length of each actuator respectively. So Δ l=l-l’, with l’, the length of the actuator for the step
2. The outputs
   * (ux,uy) are the calculated displacements applied from the actuators to the beam
3. Relations between inputs and outputs

Given that both actuators are fixed to the frame by a pivot joint, we can say that their endpoints coordinates satisfy the equation of a circle whose center has coordinates (x1,y1) and (x2,y2) respectively. Since those endpoints are connected to the same pivot, the center of that pivot, with coordinates (xc,yc), must then satisfy both previous equations, giving us the following systems:

* For the initialization
* For each step

Solving those systems thus gives us (xc,yc) and (ux,uy), which can be transferred to the next FMU.

1. Une image contenant mur, intérieur

   Description générée automatiquementThe cantilever beam

Same idea for the cantilever beam.

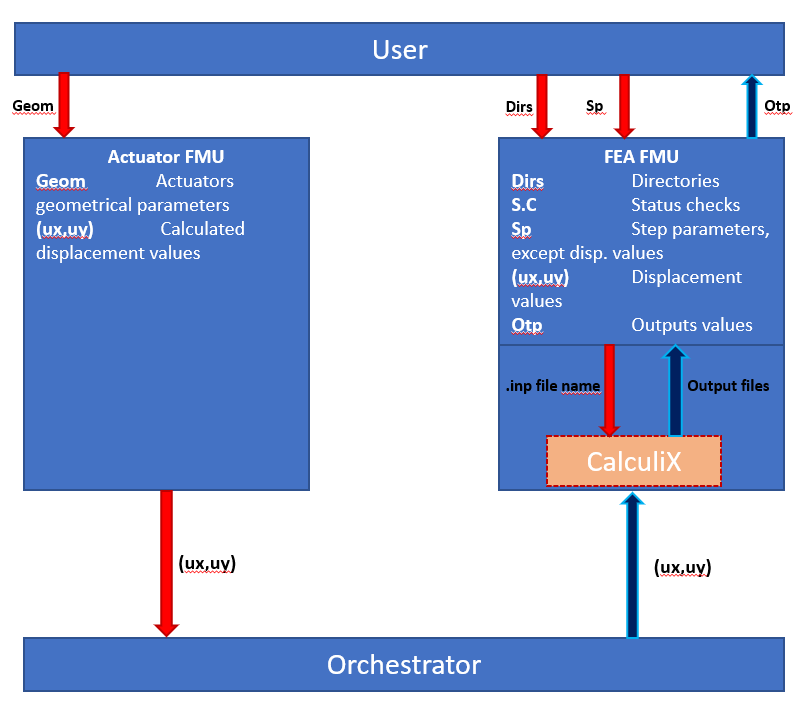
1. The inputs
   * ccx\_exe\_path is the raw path to the Calculix, the Finite Element Solver, ccx.exe path.  
     For further studies on the project, ask the user to directly add the ccx.exe to the OS Path environment variables
   * work\_dir is the raw path of the directory where all the step files will be saved.  
     For further studies on the project, it is preferable to use a temporary directory
   * rout\_dir is the directory to the .rout file used during the steps. Initially, it corresponds to the directory where the first .inp file is stored. It needs to be set by the user for the moment.  
     Further on, this file could be generated directly by the gmesh API, or something similar
   * nb\_steps\_prior serves as a status check for the simulation, especially for the initial step, where the procedure is slightly different
   * total\_steps is the number of steps to run for the simulation, defined by the user. Further on, that value needs to be calculated from the ratio of the length of the experiment and the step size.
   * Error is a status check for CCX. If there is an error with a run of CCX, that error variable because true, but procedure afterwards is yet to be determined
   * L is the length of the beam, used to determine the torque exerted from the beam to the frame

Then we have the Calculix static analysis step parameters, taken from the CCX manual, [\*STATIC (openaircraft.com)](https://www.openaircraft.com/ccx-doc/ccx/node331.html):

* + first\_increment\_value defines the initial increment for each step, regardless of previous steps. Need more information on necessity for inp file structure
  + step\_duration defines the time length of the step.
  + min\_increment\_value defines the minimum time increment allowed
  + max\_increment\_value defines the maximum time increment allowed
  + disp\_node\_set\_name relates to the name of the node set receiving the displacements, aka the node attached to the actuators
  + fixed\_node\_set\_name relates to the name of the node set fixed to the frame
  + analysis\_type is the type of FEA analysis to run. In this project only static analyses are possible. Further on, the rest of the capabilities of CCX could implemented to the FMU, but requires understanding on the other types of analyses
  + output\_type is else Forces or displacement, depending on what node set the user wants data on. For the moment, only forces have been implemented to the FMU
  + (ux,uy) are the calculated displacements applied from the actuators to the beam

1. The outputs
   * Fxbo Resulting horizontal force beam-->actuators[N]
   * Fybo Resulting vertical force beam-->actuators[N]
   * Fxfo Resulting horizontal force beam-->frame[N]
   * Fyfo Resulting vertical force beam-->frame[N]
   * Mzfo, Resulting z axis torque beam-->frame[N.m]
   * Dat, Displacement and force output filename
   * mass\_mat, Mass matrix output filename
   * stiff\_mat, Stiffness matrix output filename
2. Relations between inputs and outputs

To understand the previous inputs and outputs, here’s a simple schematic of the digital shadow’s working.

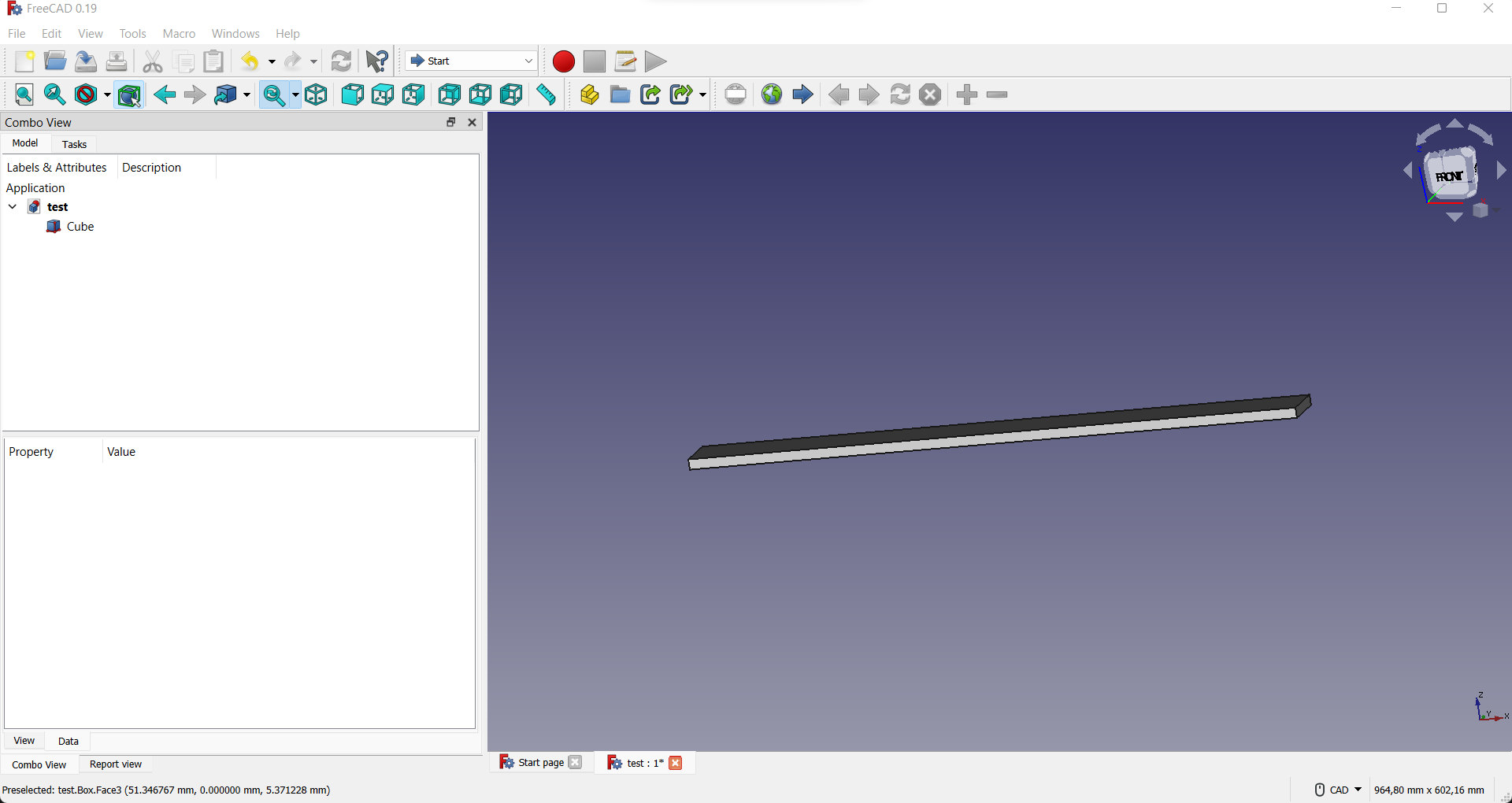


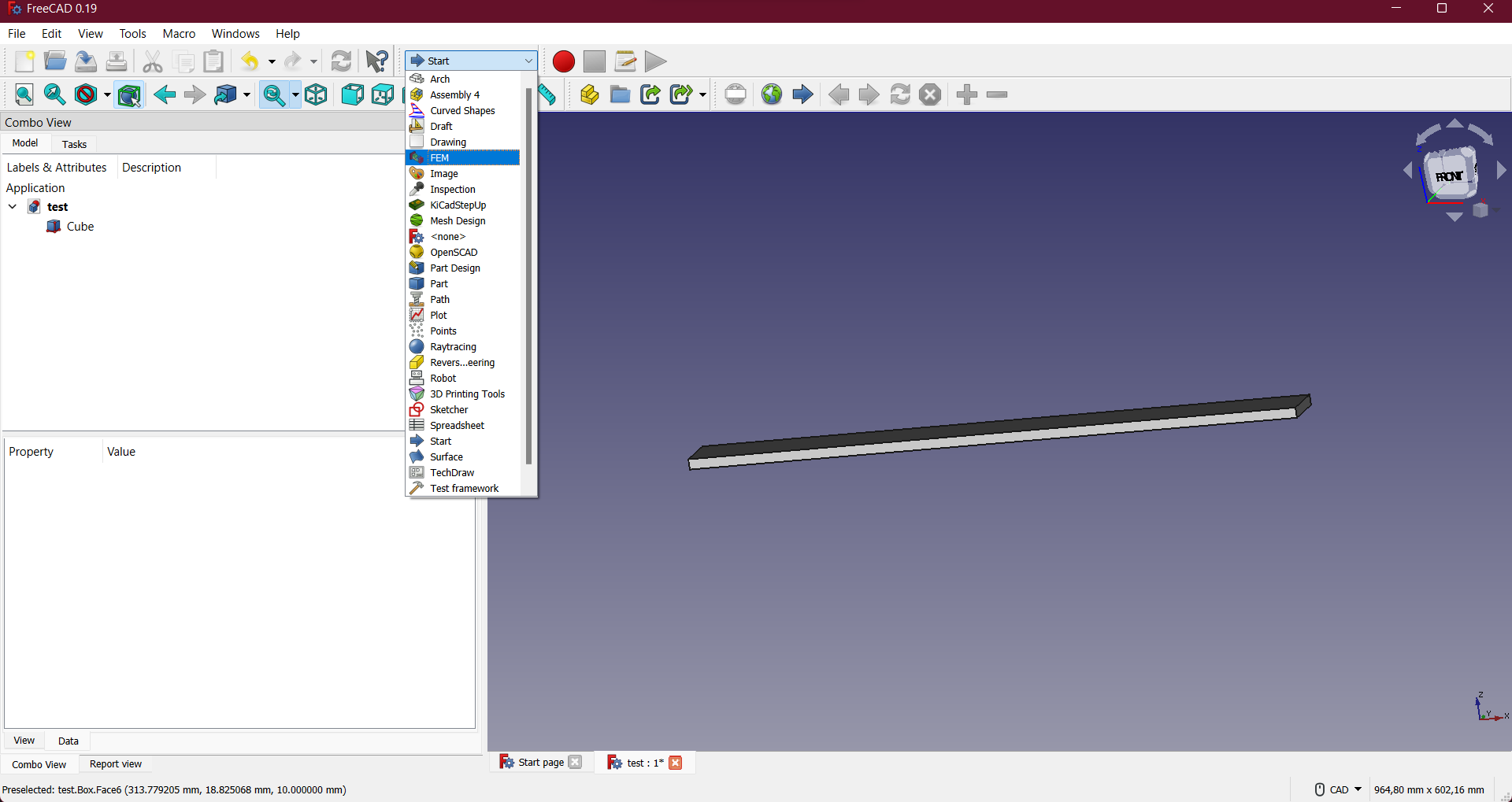
1. Installing and running the FMUs

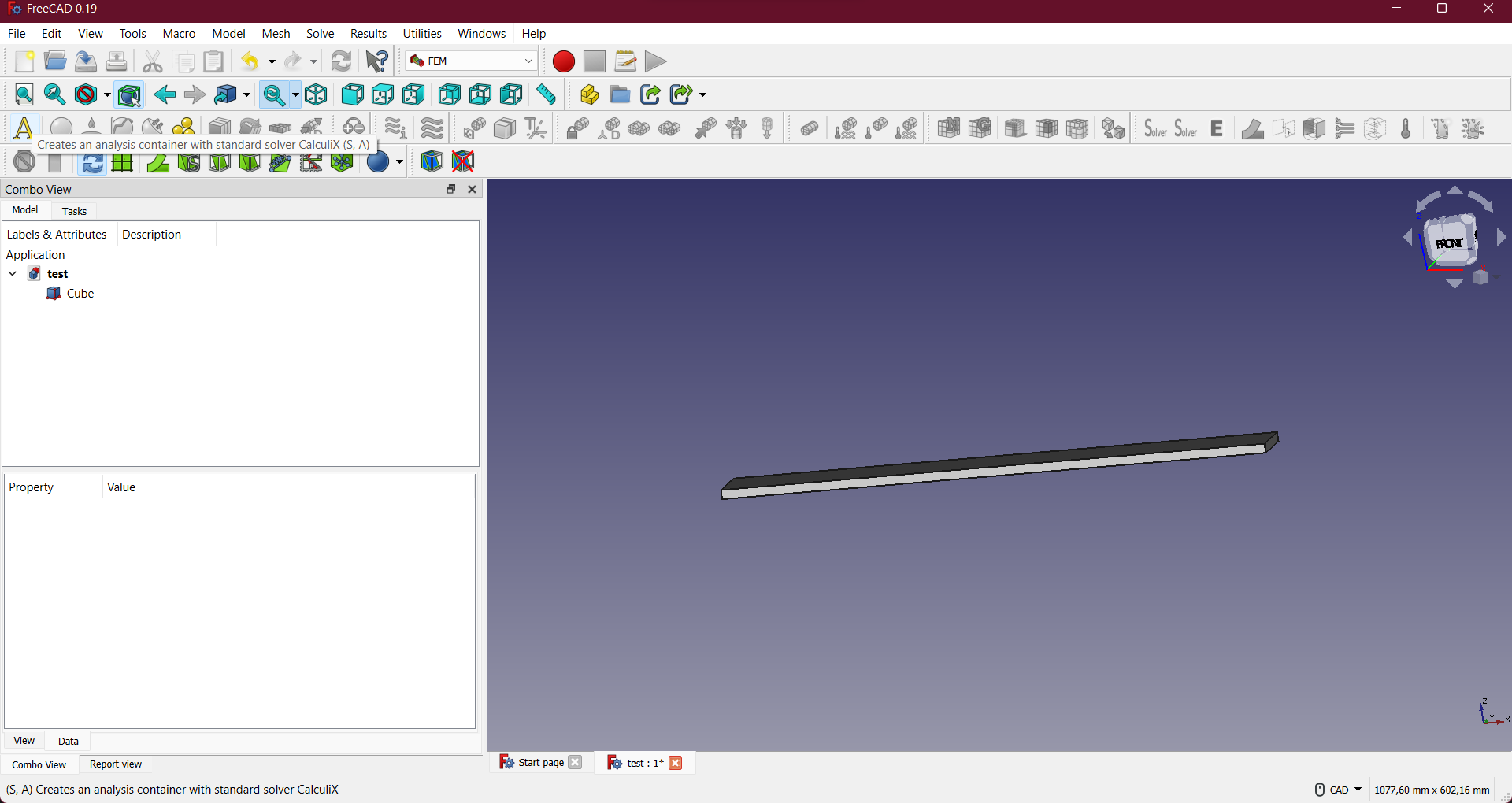
Step 1: Create your object geometry, on a CAD software of your choice. In my case, I chose FreeCAD: [FreeCAD: Your own 3D parametric modeler (freecadweb.org)](https://www.freecadweb.org/downloads.php)

Note: FreeCAD version 0.19

The following will only focus on the FreeCAD software, other methods have yet to be studied!!

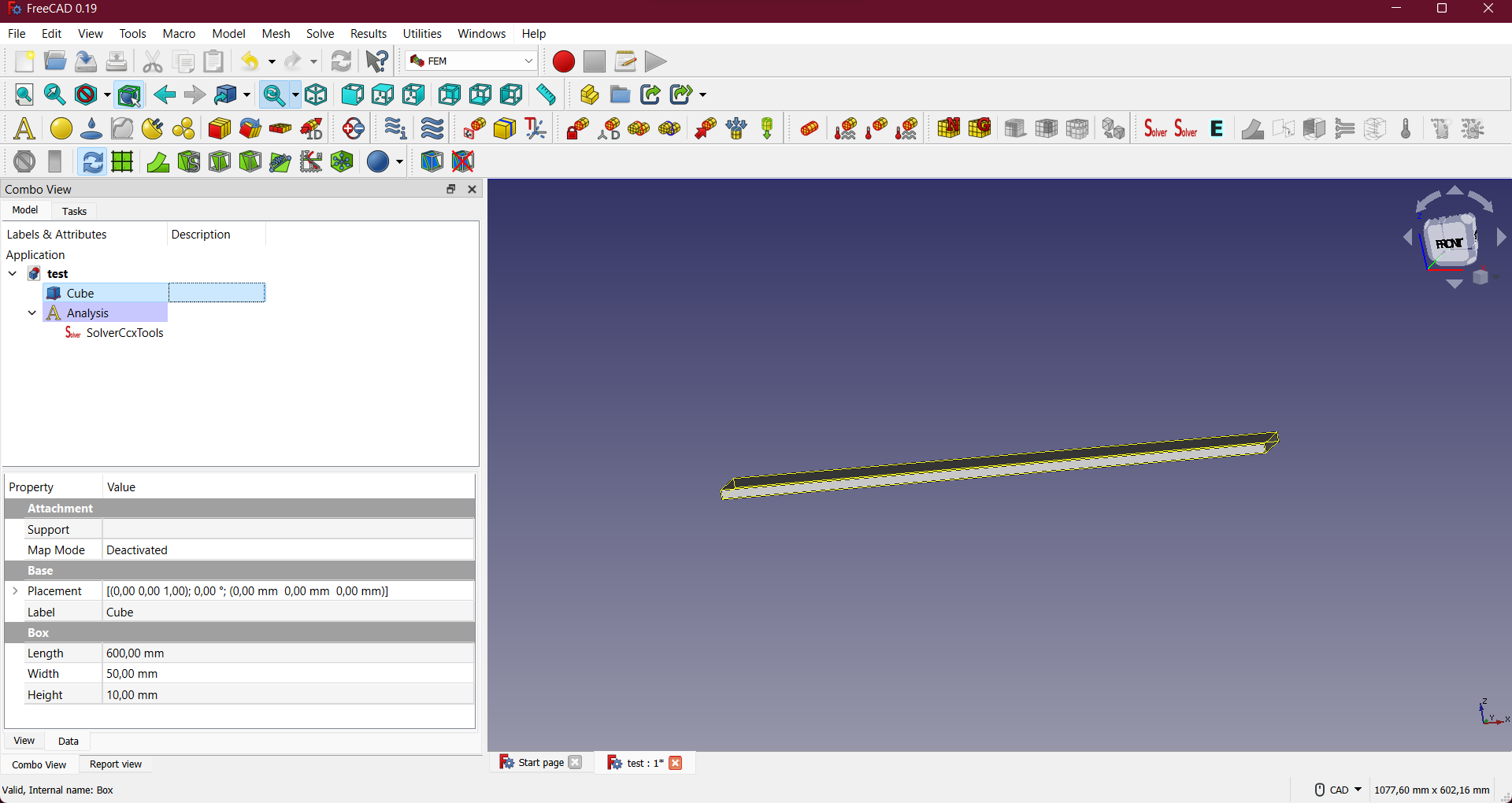
Try making a simple cantilever beam, and it should look something like this

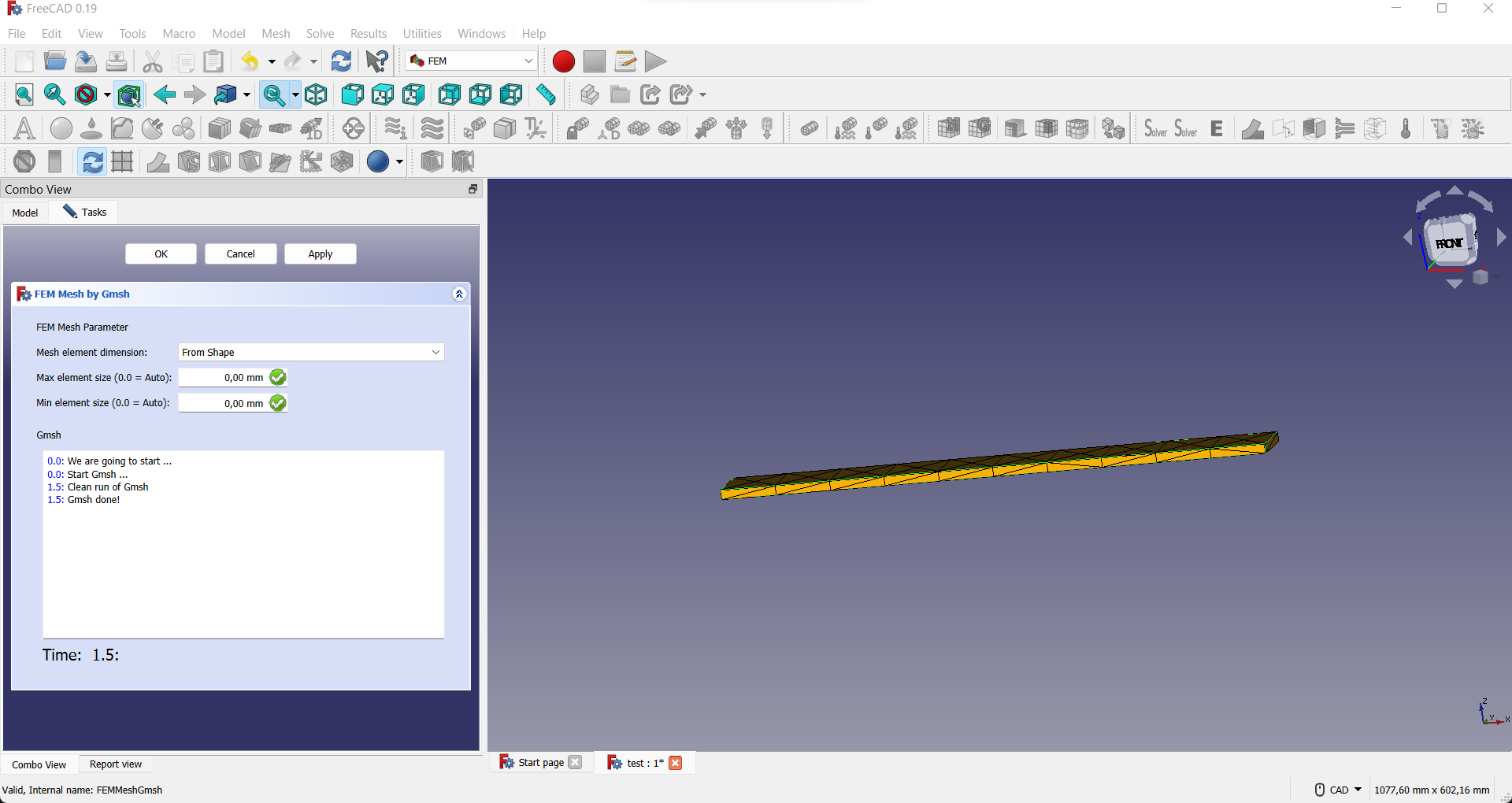
Open the FEM workbench as such

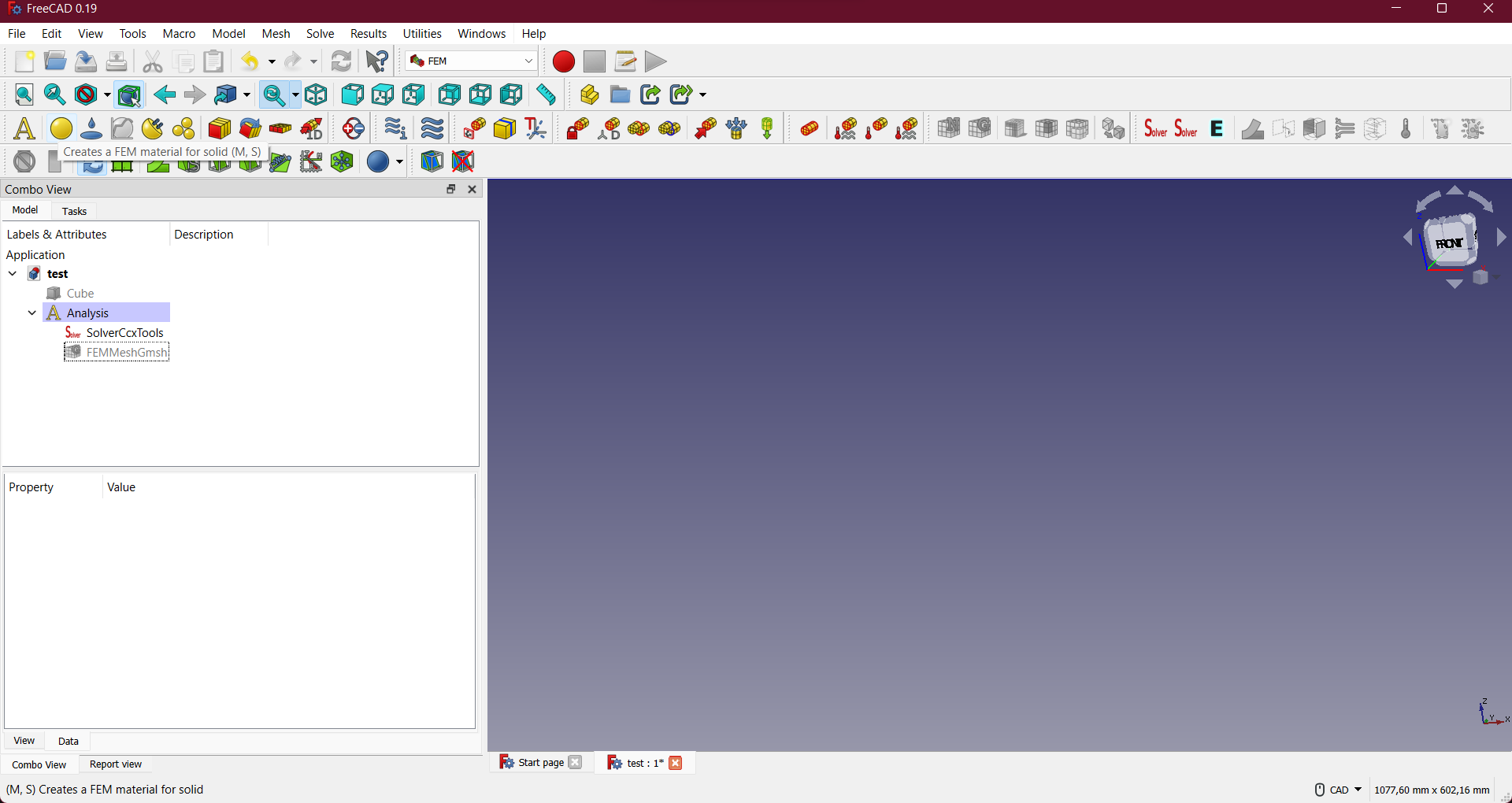
Click on analysis on the top right of the window

And click and open the analysis menu that was added to your combo view.

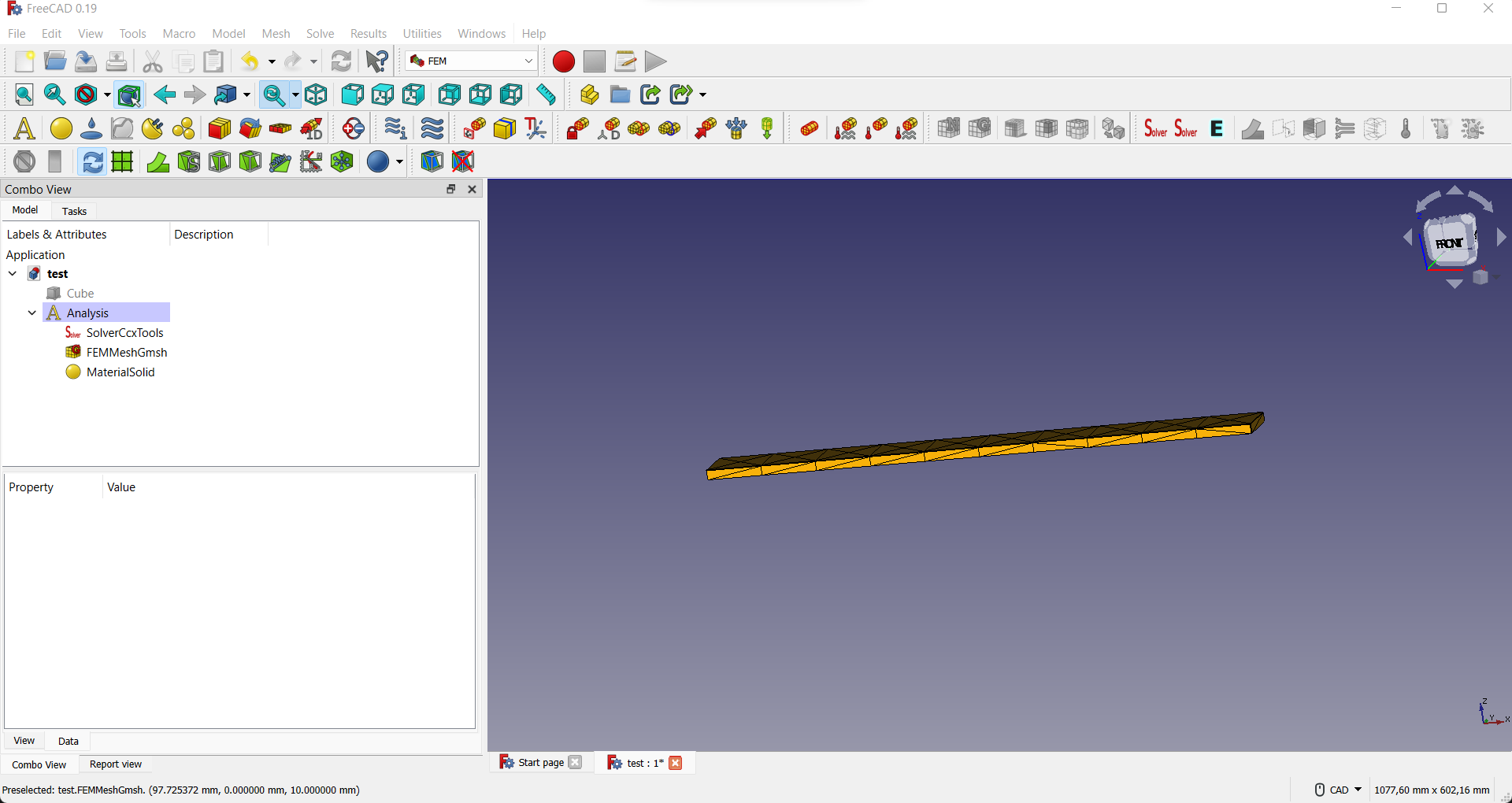
Click again on the geometry model in the combo view, the cube in our case. And add a mesh with the netgen or gmesh icon, setting the parameters to your liking, here I’ll use Gmesh.



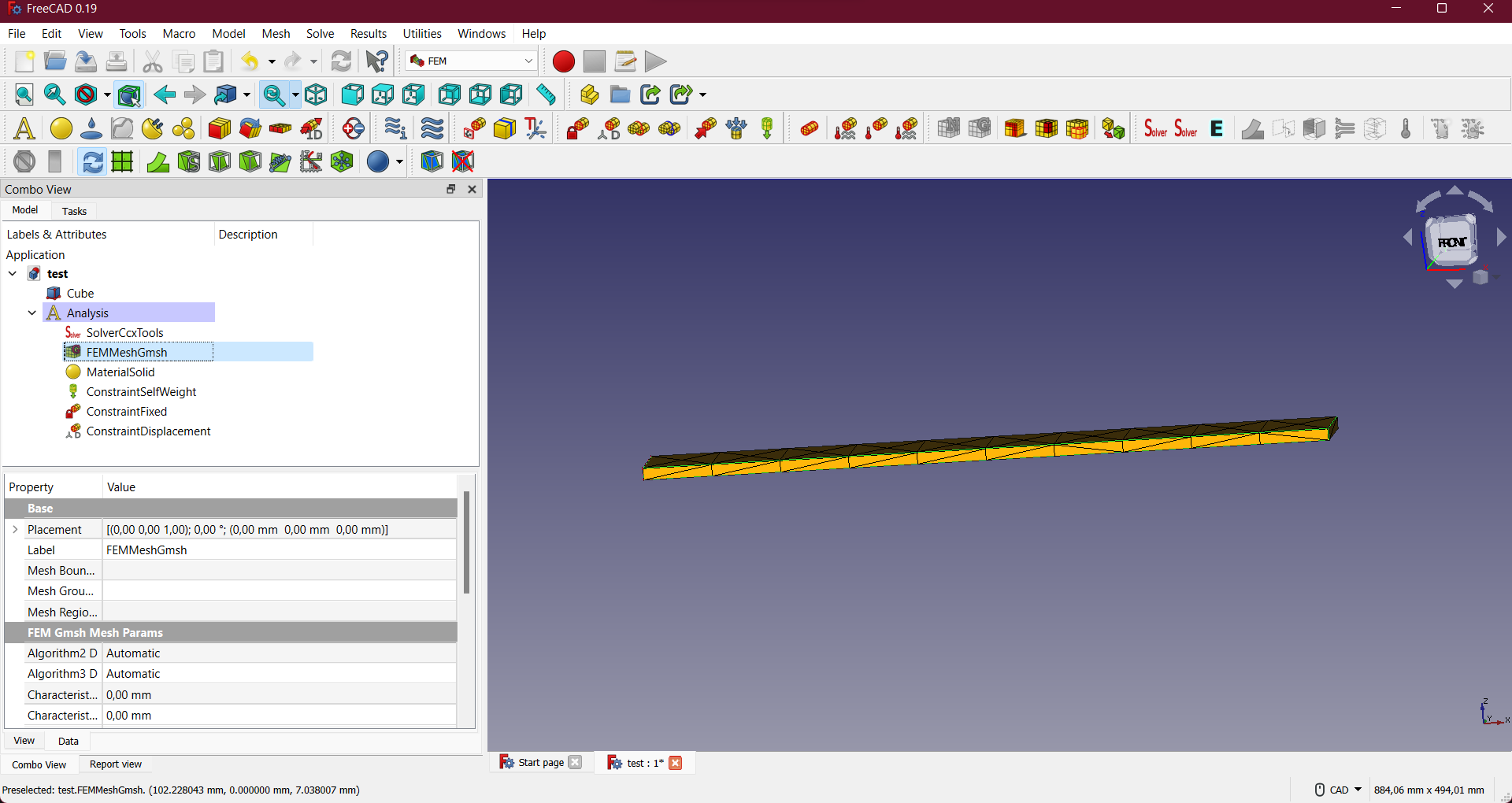
Remember to select apply before selecting the Ok button and you should have the mesh built for you

Then click on the material icon and chose your material. Going with generic steel in my case

Une image contenant texte

Description générée automatiquement

Then click on the mesh in the combo view and add your constraints with the icons in the FEM workbench



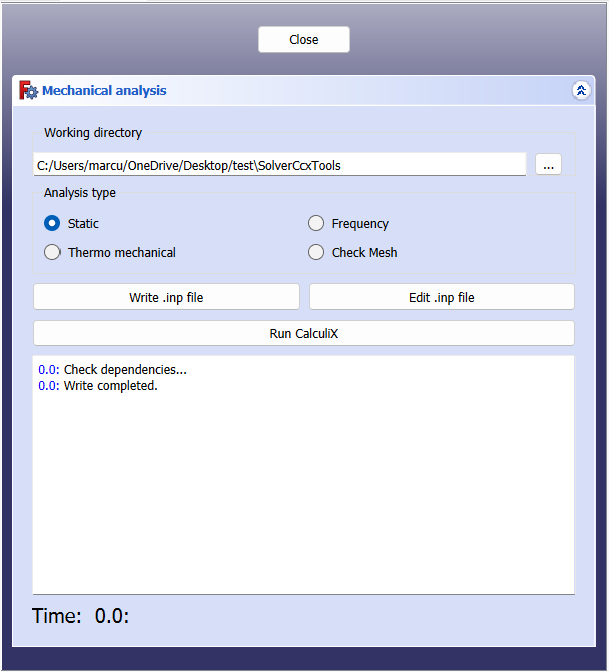
Once all your constraints have been set, you need to write the first inp file

Go to the SolverCcxTools icon on the right side and this should pop up.Une image contenant texte, capture d’écran, moniteur

Description générée automatiquement

Remember the directory, so that you run your FEA FMU afterwards!!

Click on Write .inp file and this should be the state of your window.



Click on close and you’re now done with FreeCAD.

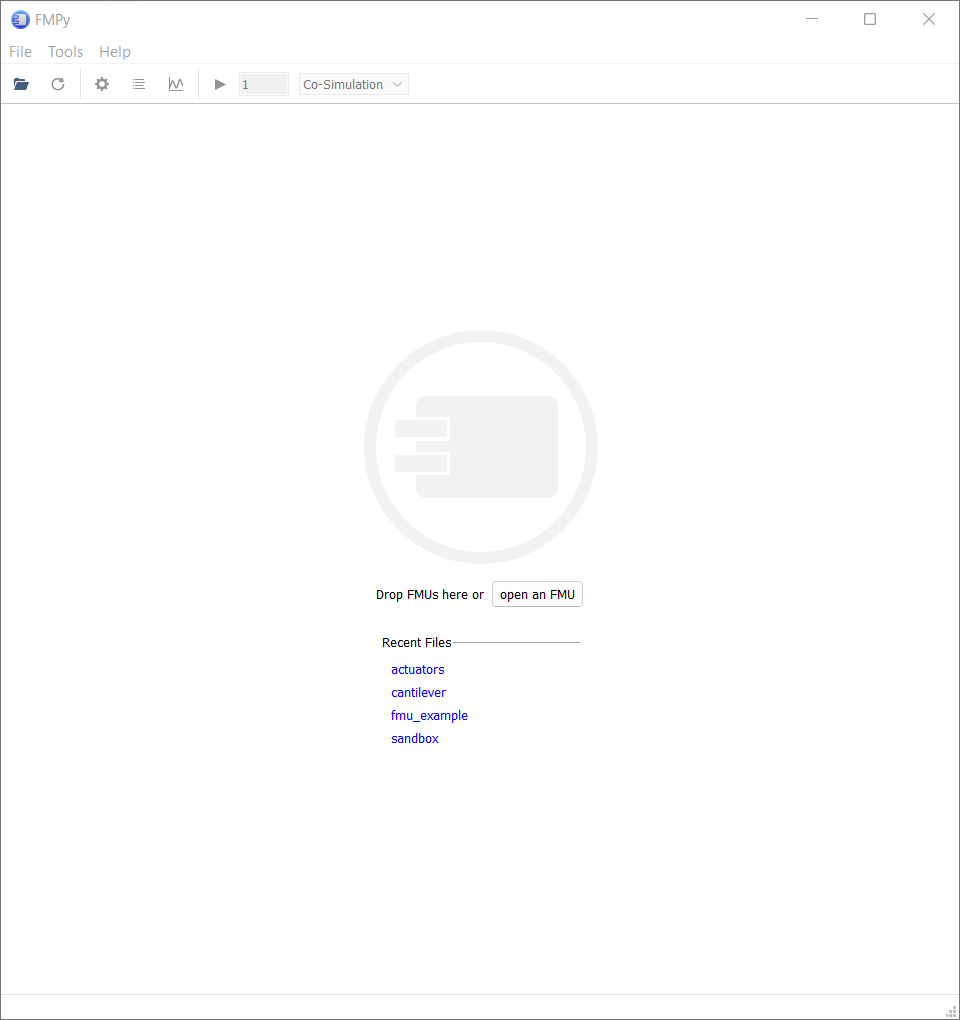
Step 2: Go and find the generated inp file and copy it to the directory where you want to save your results to.

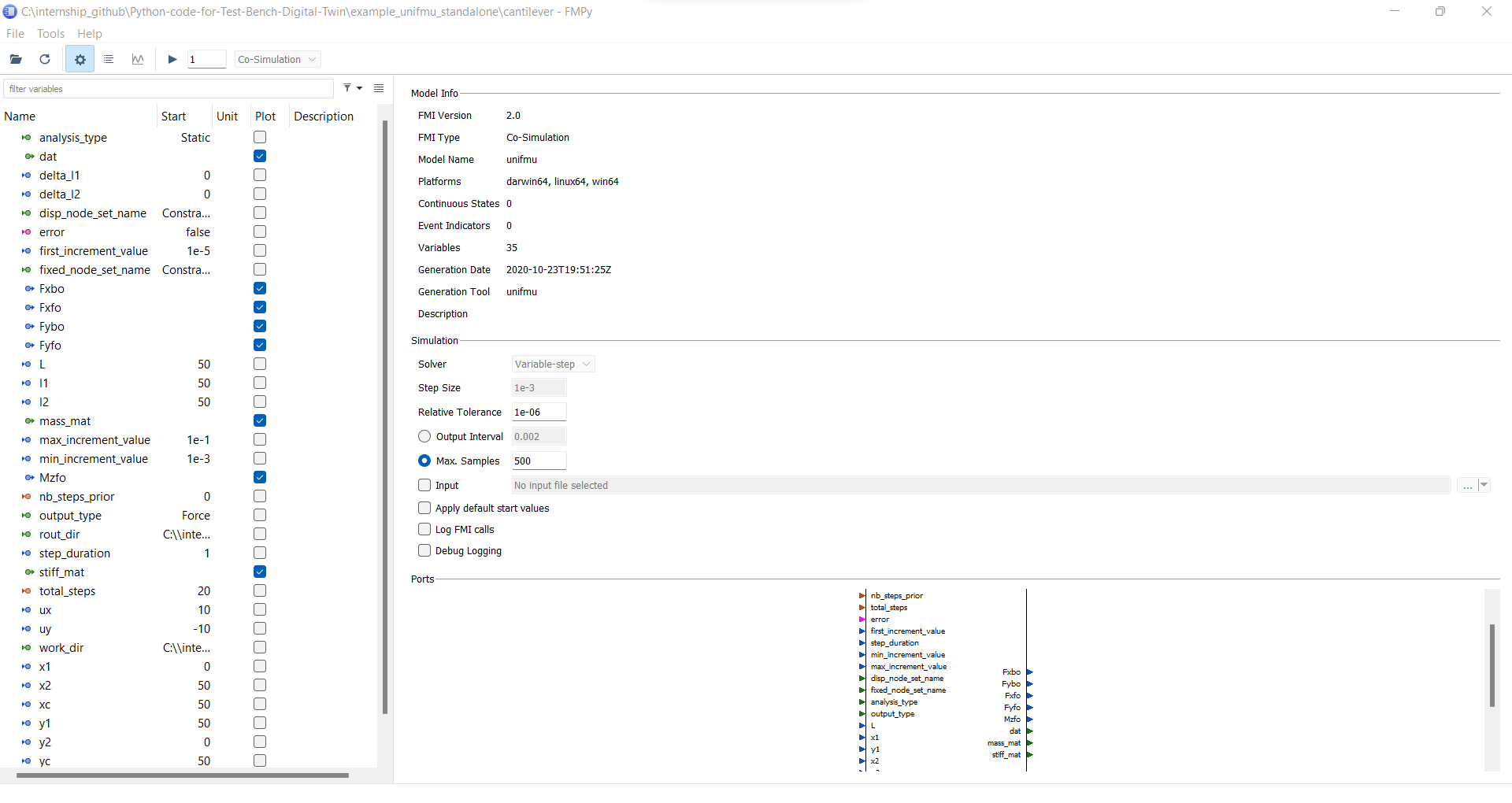
Step 3: Go to this link and download the code for the FMUs: [MarcusAdielsonGuichet/Test-Bench-Digital-Twin (github.com)](https://github.com/MarcusAdielsonGuichet/Test-Bench-Digital-Twin)

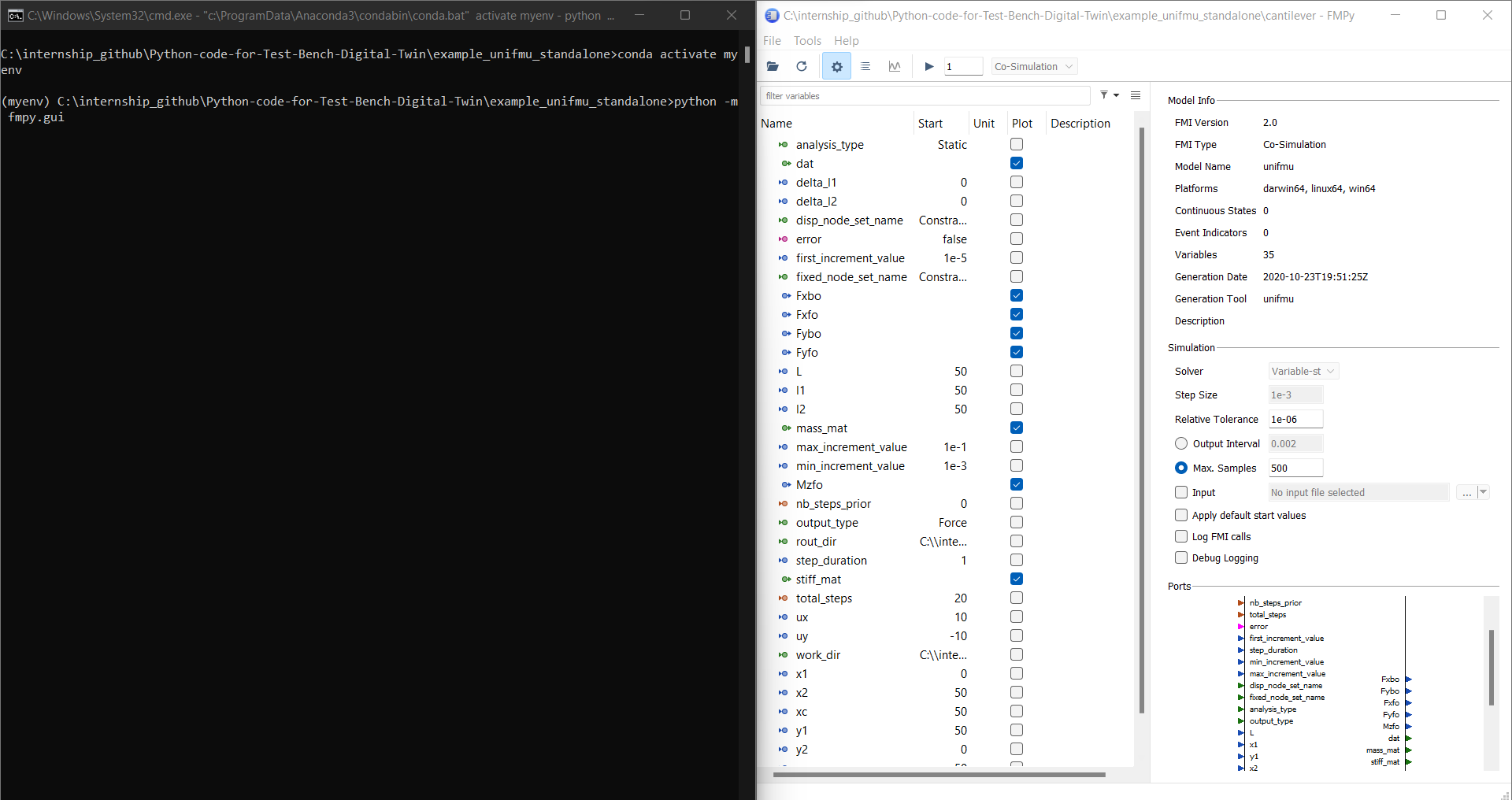
Note:

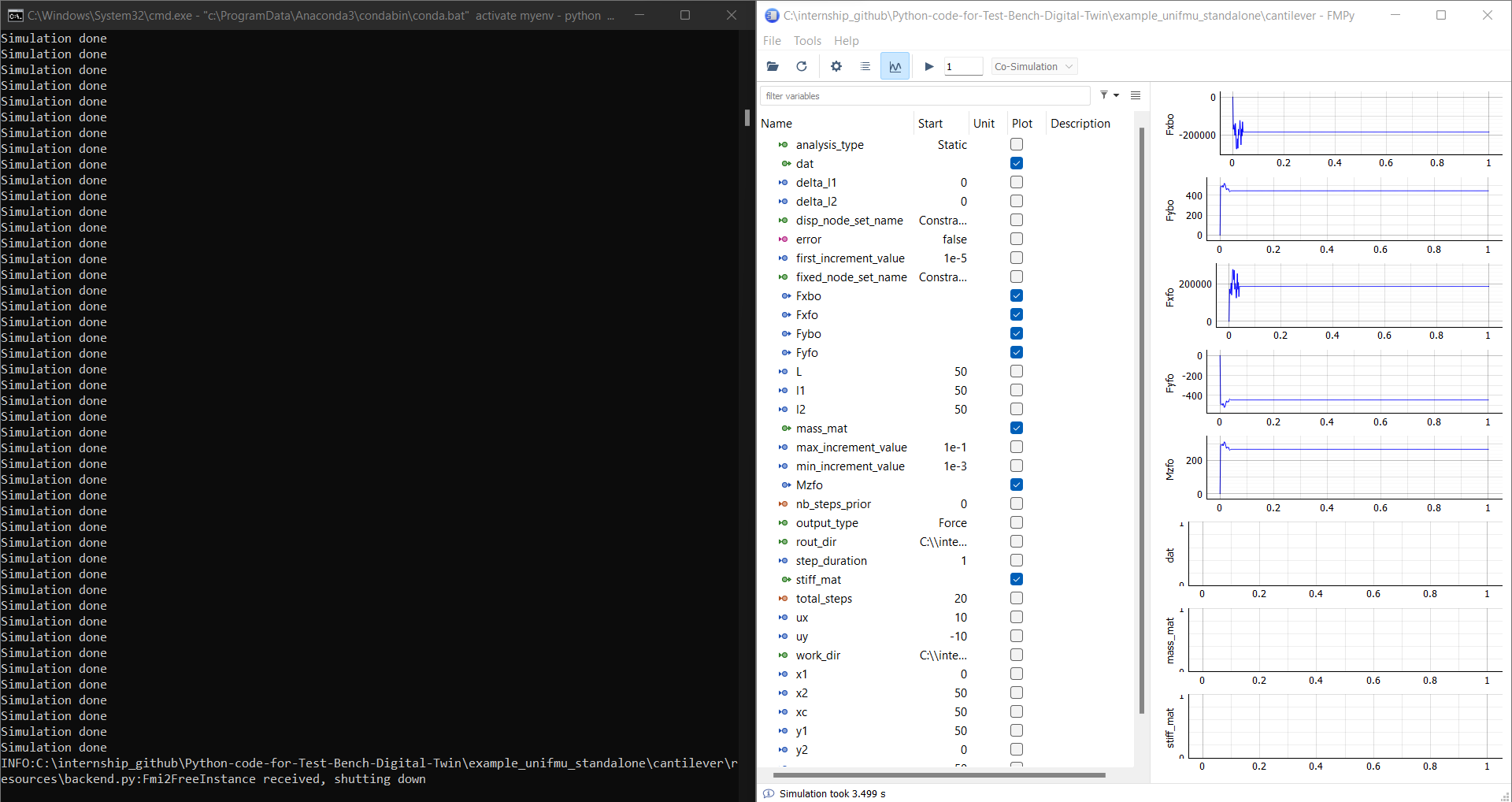
* CCX 2.19 binaries location: [Windows executable ver calculix\_2.19 - CalculiX](https://calculix.discourse.group/t/windows-executable-ver-calculix-2-19/938)
* UniFMU 0.0.7 files: [Release 0.0.7 · INTO-CPS-Association/unifmu (github.com)](https://github.com/INTO-CPS-Association/unifmu/releases/tag/0.0.7)
* Into Cps 4.0.5: [Download (into-cps-association.github.io)](https://into-cps-association.github.io/download/)

Step 4 : Open cmd/Powershell in the example\_unifmu\_standalone folder and follow the README instructions in the same folder. Careful, Fmpy only run on up to Python 3.9, so make sure you have the right python

Step 5: Once the fmpy GUI is open, open/drag and drop the cantilever folder into this window

Step 6 : If everything works this should be the state of the fmpy window

And here with the command line window next to it:

Then if you press the button on the top this should be the result:

If that is the case you can now play around with the parameters, so that they satisfy your study case. Don’t forget to press enter with the last parameter change, so that the Fmpy incorporates it into the next simulation.

The ccx\_exe\_path is: ..\CCX Files\calculix2.19win64\ccx\ccx\_219.exe, to be set by the user?

Step 7: Open the actuators folder in the same fashion as the cantilever one, and repeat the Fmpy test to get these windowsUne image contenant texte

Description générée automatiquement

Step 8: Co-simulation

Close all fmpy gui windows and cmd windows.

Download into-cps.exe from this link: [Download (into-cps-association.github.io)](https://into-cps-association.github.io/download/)

Extract it in a favored directory

Open a cmd window and reactivate the conda env with: conda activate myenv

Launch the into-cps.exe from the cmd window: INTO-CPS-Application.4.0.5.exe   
Needs to have the python 3.9 or the uniFMU won’t work, that’s why we’re relaunching the conda env!!!

Please note that you also need to remove the random input generator in both model.py for the FMUs, if you have your own set of inputs for a co-sim. The function to take out and all its calls is update\_inputs() in the cantilever FMUs. For the actuator FMU, the actuators\_output() needs to be modified to stop the random input generation.

This should be the next window appearing on your screen. It takes a little while don’t worry…

Une image contenant texte

Description générée automatiquement

Go to file/New project, name your project and select the folder in which the save the Co-simulation files, and validate with the create button.Une image contenant texte

Description générée automatiquement

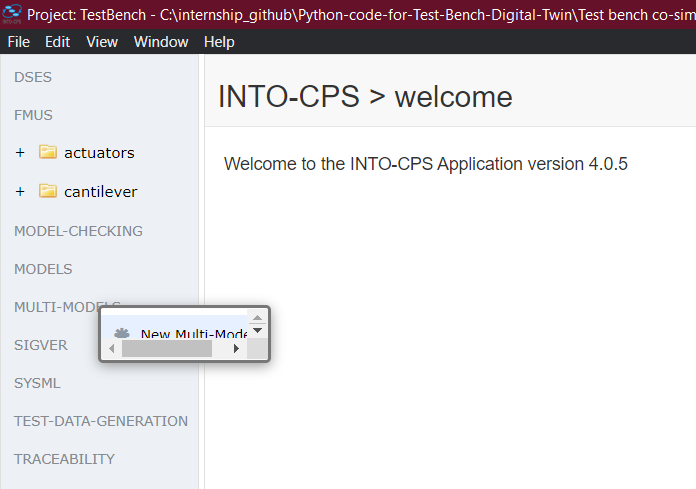
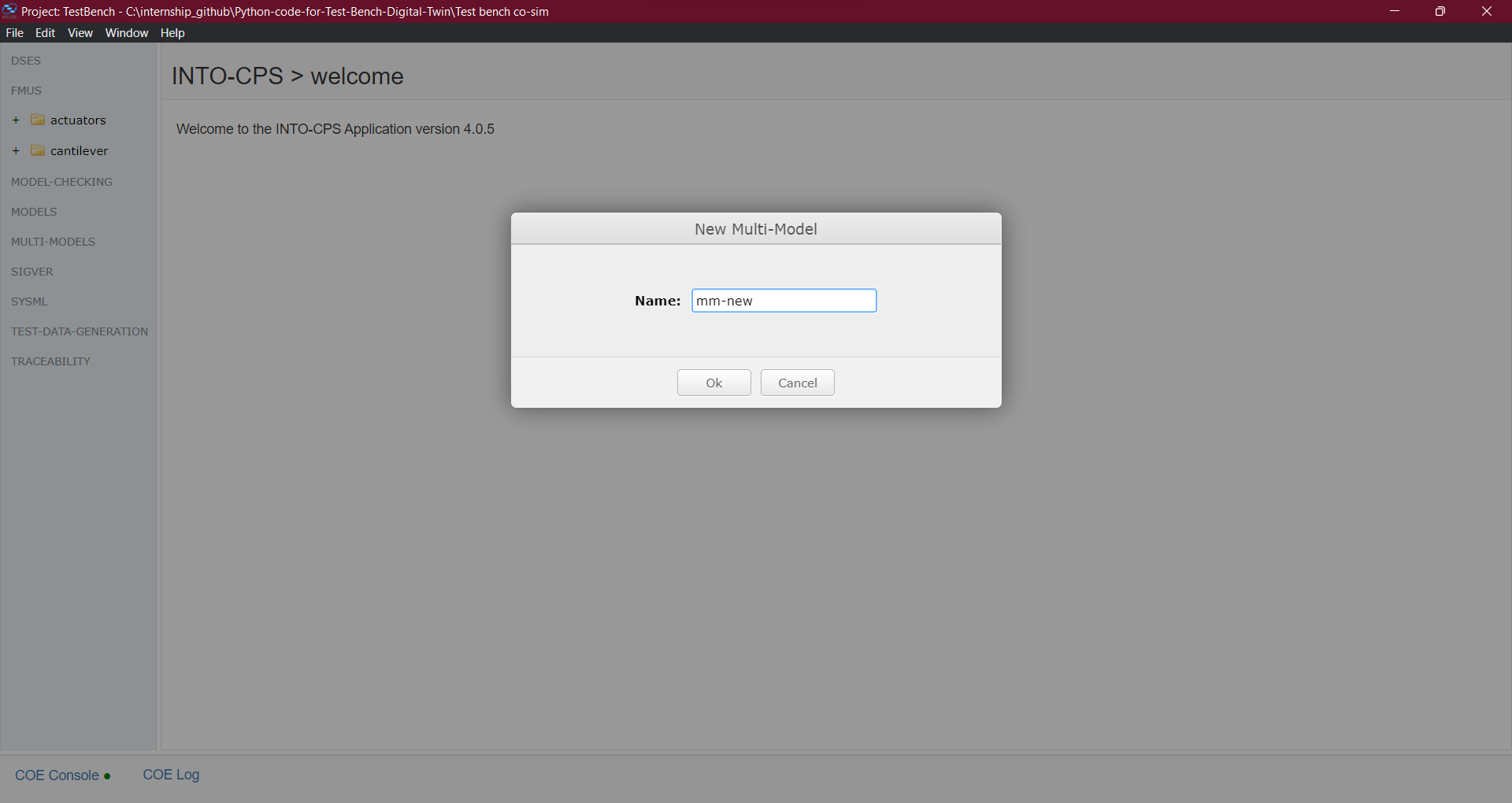
Une image contenant texte

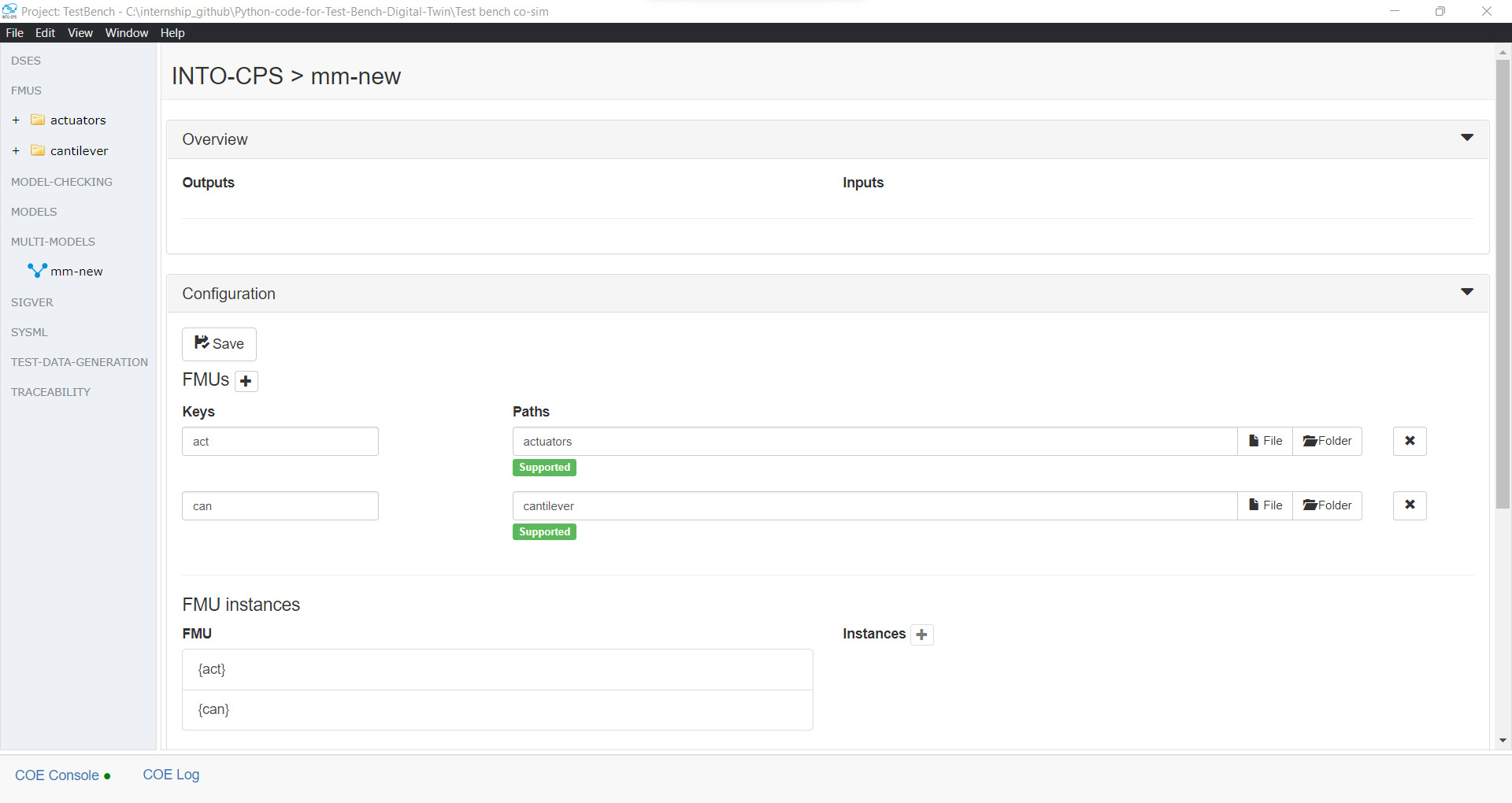
Description générée automatiquement

Then you go to the example\_unifmu\_standalone and copy the FMU folders, to the FMUs folder inside of your project folder.Une image contenant texte, moniteur, noir, capture d’écran

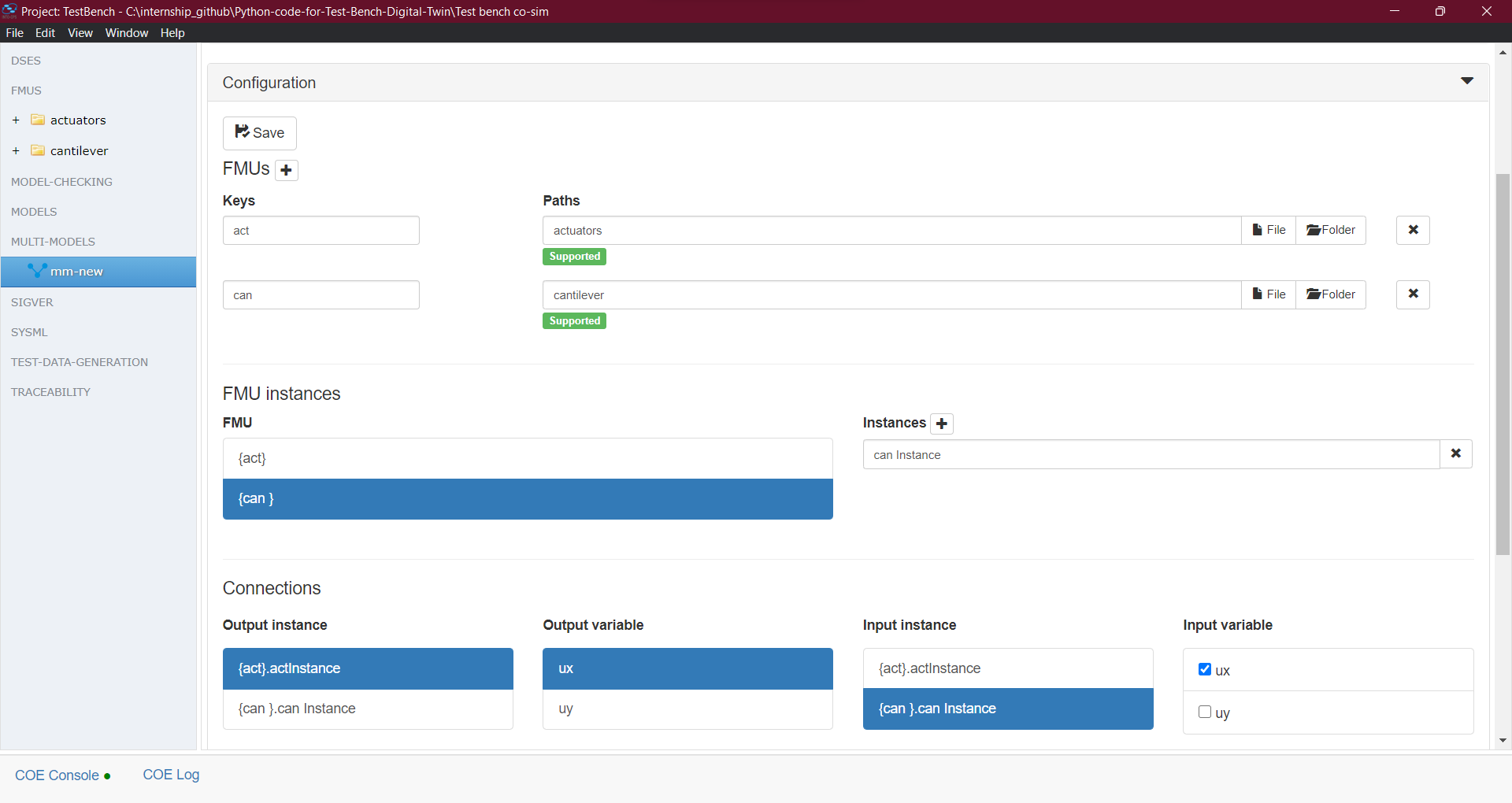
Description générée automatiquementUne image contenant texte, capture d’écran, moniteur

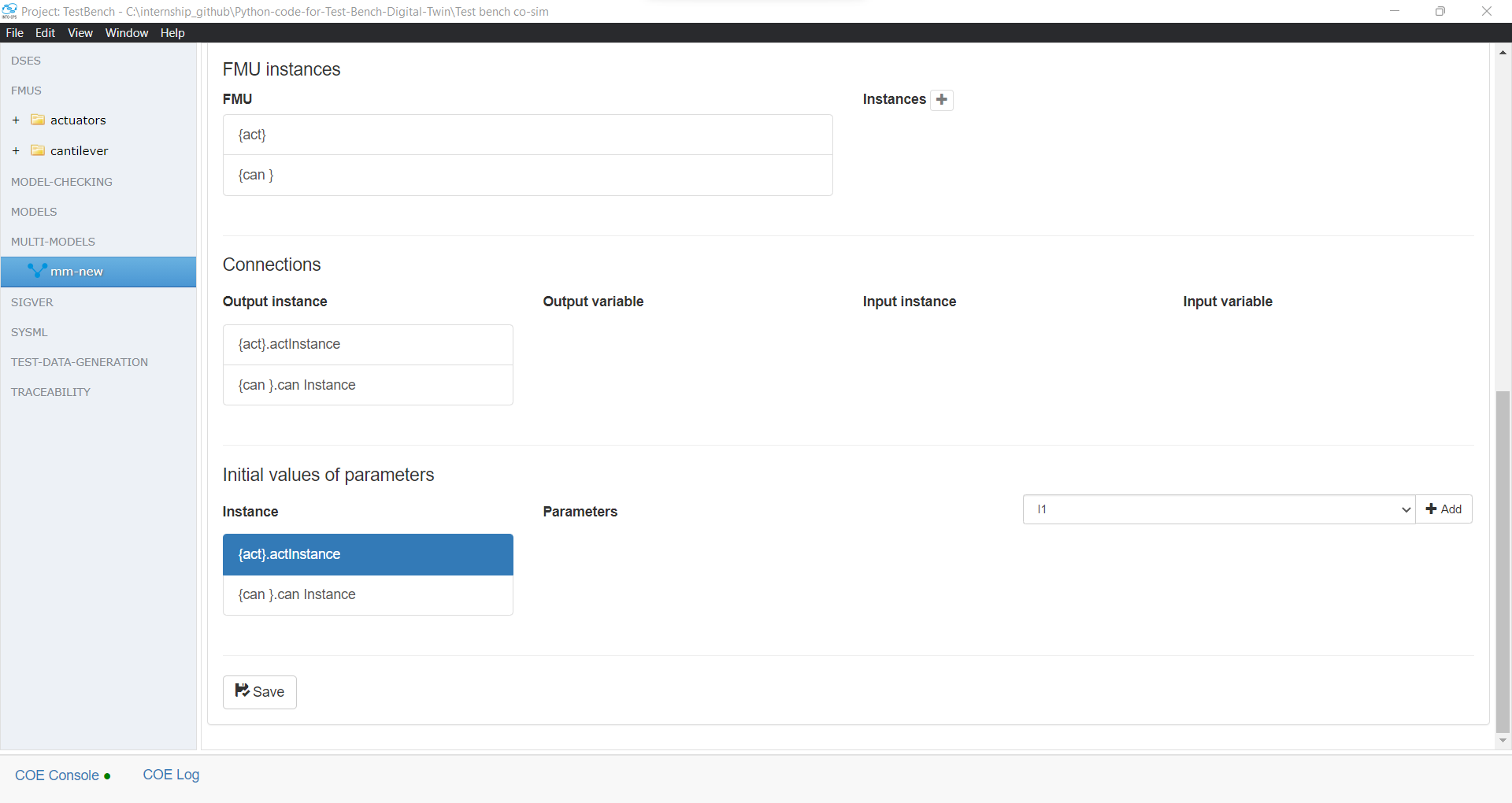
Description générée automatiquement

Then return to the into-cps window and right click on Multi-Model and select New Multi-Model and name it.  
  


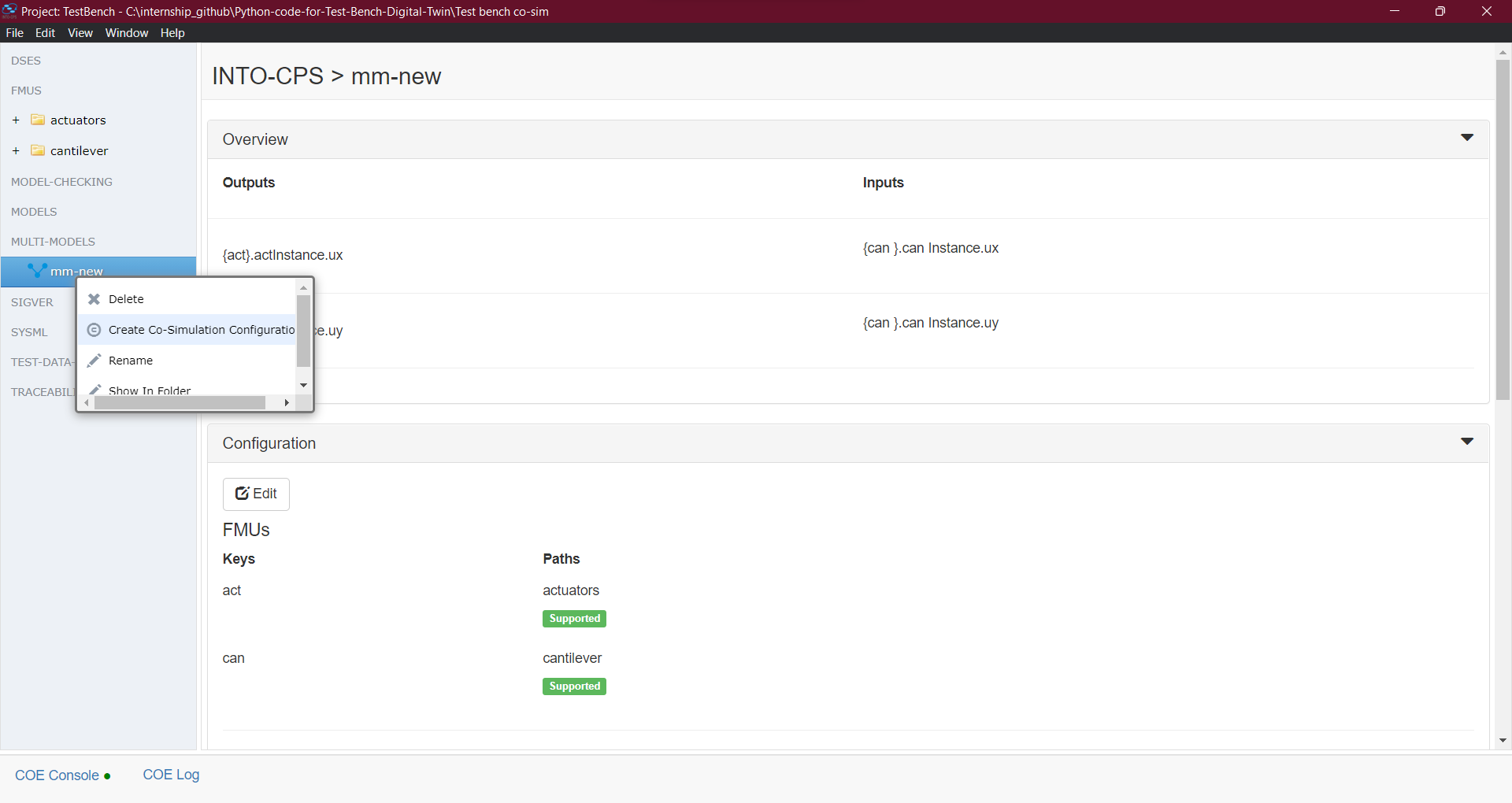
Open configuration, click Edit, add each FMU folder from the FMUs folder, name them accordingly in the keys text space.  


Add one instance of each fmu

And add every link between inputs and outputs of the FMUs with the connection section. Here the ux ouput from act is linked to the ux input from can  


And then you can also add some initial values to your parameters with the last section  


Remember to click save when you’re done.

Now right click on the multi-model and select Create a Co-Simulation Configuration  


And configurate your co-simulation before saving.  
Une image contenant texte

Description générée automatiquement

Close the configuration tab and now only the simulation tab is extended and visible.  
Une image contenant texte

Description générée automatiquement

Go to the Window/Show Download Manager and download the latest Maestro  
Une image contenant texte

Description générée automatiquement  
Une image contenant texte

Description générée automatiquement

Once it is download, close the Download manager and click on launch  
Une image contenant texte

Description générée automatiquement

After we fix the issues, this is where the co-simulation works

Step 9: Dynamic visuals

1. Further ideas to explore

So far, only the solver part is done and both the preprocessing as well as the post processing need to be automated maybe, or at least have a smoother transition between them. This is the link to the CCX binaries to get the right excecutable: [Windows executable ver calculix\_2.19 - CalculiX](https://calculix.discourse.group/t/windows-executable-ver-calculix-2-19/938)

Here are several software for pre/post processing, mostly from this CCX forum thread, [Pre/Post Processing Open Software - CalculiX](https://calculix.discourse.group/t/pre-post-processing-open-software/630):

* PrePoMax, easy to use, very intuitive only on Windows, but requires knowing the CCX keyword library for more complex simulations. Can also be used for preprocessing, fairly simple of usage
* CalMED conversor + Paraview for MED, requires the converter for Paraview to read the .frd files generated by CCX, as it only reads .vtk files. Also, loss of variables, mis conversion as well
* CalculiX GraphiX, native to CCX, but Windows’ version is very limited, requiring Linux use knowledge to run for full capabilities. Steep learning curve, more advanced FEA and CCX background necessary
* Salome Mecway Code Aster, very steep curve, difficult installation, but apparently more performant than CCX, for very large simulations

I recommend having a look at this thread: [Where to start with open source FEA - CalculiX](https://calculix.discourse.group/t/where-to-start-with-open-source-fea/969). It gives extensive insights on what to use depending on the types of analysis and data treatments you want as a FEA user.

Note that all of this software are open source, so taking a look at their code on GitHub could shorten the time required to code a new multi-step cosimulation post-processor.

Regarding the pre-processing, using FreeCAD for the moment is a good and very simple use process, but it might lack the compatibility with other software files. So exploration on that idea also needs to be looked into.

Also, FreeCAD is harder to work with than paid software when it comes to assembled systems, even if it does work. So more extensive research and knowledge on FreeCAD is required to finalize the generalization of these FMUs for test benches.

Gmesh was not delved into much during the internship, given how FreeCAD handles it fine, but maybe it possesses that compatibility, and the CGX launcher can convert any .unv, which is Gmesh’s output type, into a .inp file, so