**15 Mayo 2018**

A priori tenía compilado Florence en la carpeta dev/florence/ pero no tenía claro con qué versión de Python había compilado. Al intentar correr el test usando python tests/test.py aparecían errores distintos en función si usaba python 3.5 o python 2.7. Al observar la línea de errores, parecía que al correr con python 2.7, el código era capaz de avanzar algunas líneas más.

He bajado de nuevo Florence en una nueva carpeta, de nombre dev/florence2/. En esta carpeta he probado de compilar usando python 2.7 (nota: por algún motivo, ahora el python por defecto es el 2.7, es decir, el python que aparece al teclear el comando “python” es el 2.7). Al intentar compilar con python 2.7, aparecían una serie de erorres relacionados con las flags de c++. Específicamente con las flags “-Wno-“, “-mllvm” y “-inline-threshold=$(INLINE\_THRESHOLD)”. Estas flags eran llamadas en distintos make situados en :

* FiniteElements/Assembly/\_Assembly\_
* MaterialLibrary/LLDispatch
* FinitElements/LocalAssembly/\_KinematicMeasures\_

He buscado qué partes específicas del código incluían estas flags y los he quitado (buscar por líneas comentadas con “(MAF)”). Después de esto el código se ha compilado sin problemas.

Al correr ahora python tests/test.py , el código entraba ya dentro de la simulación (cylinder), pero ahora se quejaba de la inexistencia de PostMesh, que diría haber instalado. La realidad es que lo había instalado dentro de la carpeta opt/ sin ningún criterio muy específico. He seguido las instrucciones de la página de PostMesh de Roman. Específicamente, dentro de cd ~, he seguido:

git clone https://github.com/romeric/PostMesh

cd PostMesh

make

[sudo] make install

To further build the C++ examples, (after building and installing PostMesh shared library) do

cd examples

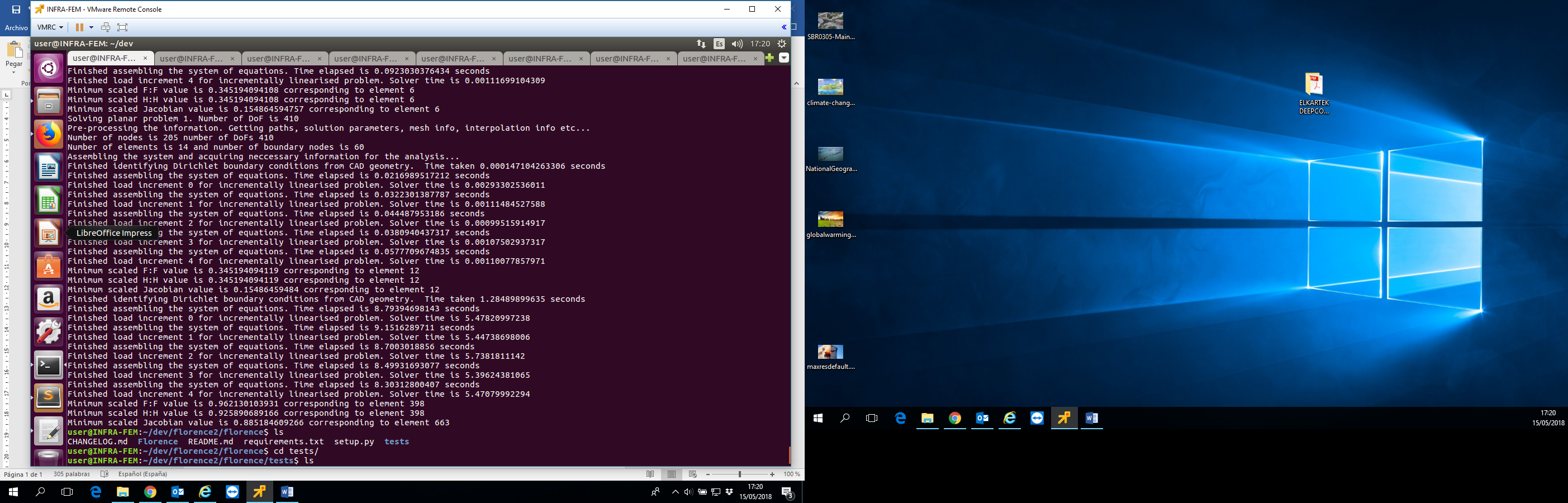
make

To build Cython/Python bindings (make sure you are in PostMesh directory)

cd PostMeshPy

[sudo] python setup.py install

Al finalizer he hecho sudo ldconfig y he probado de correr de nuevo el test y sí ha funcionado sin problema. Aquí un pantallazo



Entonces, parece que Florence está bien instalado. cd

Siguientes pasos:

* No tengo realmente claro como se llama al programa y como se puede hacer para crear output y pintarlo
* Tampoco tengo clara la estructura en paralelo y como se puede probar de correr en varios procesadores

**17 de Mayo 2018**

He intercambiado algunos emails con Roaman, que me ha ayudado en bastantes cosas. Aquí una copia de sus mensajes.

By the way I forgot to mention. The easiest way to get around module dependencies is to perhaps install a good package manager.

Have a look at anaconda: <https://www.anaconda.com/download/>

Also if you have anaconda installed, then florence can use the built-in intel solver (Pardiso) shipped with anaconda to solve really big problems which standard Python/MATLAB backslash cannot solve.

Best regards,

Roman

On Wed, May 16, 2018 at 6:06 PM, Roman Poya <[romanpoya@gmail.com](mailto:romanpoya@gmail.com)> wrote:

Hi Miquel,

You are right. The errors comes from Florence but its a purely Python issue. I am sure you have incorrect environment variables. Or you don't have a proper virtual environment set up.

Can you query in your command prompt (the same prompt from which you are trying to run the example) the following:

>>> which python

>>> python --version

>>> which pip

Use the result of the last command to pip install the package. If your python version is 3.5 then you surely don't have a virtualenv set up. The recommended way to do deal with python packages is by using virtualenv.

By the way what problem(s) did you have installing florence under python3.5. I have three installation of florence with python 3 and they all work fine.

Also in your code, don't give the .dat extension to a paraview output file. Paraview extension is typically .vtu for generic unstructured data.

Let me know if you have any further problems.

Best regards,

Roman

On Wed, May 16, 2018 at 5:01 PM, Aguirre Font, Miquel <[miquel.aguirre@tecnalia.com](mailto:miquel.aguirre@tecnalia.com)> wrote:

Hi Roman,

 Many thanks for the detailed response. It is really helpful!!

 And also thank you very much for uploading the new examples.

 I have been trying to plot the results for the example hyperelastic\_explicit\_dynamics.py, by adding the line

 solution.WriteVTK(“output.dat”,quantity=”all”)

 The simulation runs well, but when it hits this new line, I get an error which comes from PostProcess.py

 raise ImportError(“Could not import evtk. Install it using ‘pip install pyevtk’”)

 The thing is that I have done pip install pyevtk (also pip2.7 install pyevtk) and it seems that it is only installed in python3.5. I can see pyevtk inside the dist-packages folder of python3.5 but not inside the dist-packages folder in python2.7. Also if once inside python3.5 I type “import pyevtk”, no error is shown. If I do the same inside python2.7, I get ImportError: No module named pyevtk. Sorry because I believe this is a problem more related to Python than to Florence, but do you know  why this could be happening?

 I will indeed create a github profile in order to get your updates.

 Thank you again!

 Miquel

**De:** Roman Poya [mailto:[romanpoya@gmail.com](mailto:romanpoya@gmail.com)]   
**Enviado el:** miércoles, 16 de mayo de 2018 11:42  
**Para:** Aguirre Font, Miquel <[miquel.aguirre@tecnalia.com](mailto:miquel.aguirre@tecnalia.com)>  
**Asunto:** Re: question about Florence

 Hi Miquel,

 Great to know you are have decided to use florence for your simulations and you have been able to build and compile it!

The test cases in the florence repository are outdated and are not representative of how you should prepare your data files for the analysis. They are there mainly because they test essential components of florence every time someone submits new changes to the repository. Essentially, they safeguard the code against bad submissions/regressions.

 I have added four examples in the florence repository just today. Have a look at them, they will definitely get you started. Florence has a rather powerful in-built visualiser that can plot/animate millions of DoFs in no time (you need to VTK engine and mayavi installed for it though). However, for you it would be easier to write the results into a VTK file and visualise them using paraview. You can also save the results in HDF5 file format and read them from matlab. The workflow for writing paraview files is:

 my\_solution = FEMSolver.Solve(\*\*my\_solver\_arguments)

my\_solution.WriteVTK("my\_vtk\_file\_name",quantity=0) . # quantity to plot ux, uy, uz etc

 If you are unsure of the quantity, just type (quantity="all") however, that might take quite a bit of time if you have really big meshes. Alternatively, you can query a quantity's name based on its value, for instance

 my\_solution.QuantityNamer(4) # returns a string named F\_xx (that is the a component of the deformation gradient tensor).

 Let me know if you have further questions, I will be happy to assist you.

Meanwhile, if you want to add your own changes or request new features to florence, then you should have an active github profile. I submit new changes to florence very frequently (when I have the time) and if you are not on github then you might not have some of the latest features/bug fixes.

 Best regards,

Roman

On Tue, May 15, 2018 at 4:43 PM, Aguirre Font, Miquel <[miquel.aguirre@tecnalia.com](mailto:miquel.aguirre@tecnalia.com)> wrote:

Dear Roman,

 I hope you are doing well.

 If you had some time, I have a question about the Florence software that you are developing….

 Here in Tecnalia we were looking for an open-source FEM code and Antonio recommended Florence which you have developed.

 I have downloaded the code with all the dependencies (PostMesh, Fastor) from your github webpage, compiled it and run the tests and it seems to work well (I had problems with python 3.5 but python 2.7 seems to work). Yet, it is a bit difficult for me to understand how the code works. I see some calls you make from the TestCaseCylinder and I get a general idea about how to call the input but, for example, I cannot see how to generate the output and postprocess it.

 If it is not too much asking, would you have a more general example that I could run, where also output is being generated? What software do you use to plot the results?

 Many thanks in advance.

 Best regards,

 Miquel

Siguiendo las recomendaciones de Roman, intenté realizar las simulaciones que él subió, con mayor detalle del proceso de cálculo. El problema corría sin problemas pero parecía que había problemas en cuanto a usar pyevtk para escribir el output para paraview. Roman ha comentado que lo mejor es trabajar con un virtual environment, donde instalas todos los paquetes necesarios para una distribución específica de python dentro de una carpeta. Cuando quireas activar esa distribución específica, activas el virtual environment y allí todo está hecho con los links correctos (se evita tener links de paquetes entre distintas distribuciones de python).

He creado un virtual environment para python 2.7, que es el que de momento me ha servido para poder correr Florence.

python -m pip install --user virtualenv

En el directorio dev/ he creado la carpeta env, que es la del virtual environment

python -m virtualenv env

Y para activarlo

source env/bin/actívate

Ver más información en esta web: <https://packaging.python.org/guides/installing-using-pip-and-virtualenv/> (está explicado para python3, pero a mi me ha funcionado sin problemas para python 2.7)

Una vez activado el virtual environment, he probado de correr alguno de los ejemplos de nuevo. El hecho de haber creado el virtual environment ha hecho que no sirvan ninguno de los paquetes instalados anteriormente. Florence ha ido lanzado errores y yo he ido instalando los paquetes que faltaban vía pip.

En general no ha habido problemas excepto para el paquete PyQt4, necesario para pintar el ejemplo del wrinkling del soft dielectric film. Para ello, he seguido las instrucciones de esta página web

<https://stackoverflow.com/questions/1961997/is-it-possible-to-add-pyqt4-pyside-packages-on-a-virtualenv-sandbox>

Y específicamente esta respuesta:

Linux debian, python 2.7:

Install python-qt4 globaly: sudo apt-get install python-qt4

Create symbolic link of PyQt4 to your virtual env ln -s /usr/lib/python2.7/dist-packages/PyQt4/ ~/.virtualenvs/myEnv/lib/python2.7/site-packages/

Create symbolic link of sip.so to your virtual envln -s /usr/lib/python2.7/dist-packages/sip.so ~/.virtualenvs/myEnv/lib/python2.7/site-packages/

Por otro lado, al querer correr el ejemplo de curved mesh generation, python se quejaba que no podía importar PostMesh. He tenido que repetir las instrucciones de Roman en su web, pero ahora estando en el virtual environment, i.e. :

cd PostMeshPy

python setup.py build\_ext

python setup.py bdist\_wheel

cd ../../ && pip install PostMesh/PostMeshPy/dist/\*.whl

**31 de Mayo de 2018**

Pasos seguidos para crear repositorio de github a partir del repositorio Florence, así como crear una copia local en la máquina virtual.

*Paso 1: crear fork de Florence en Github*

Una fork es una copia idéntica de un repositorio existente en Github, para poder hacer modificaciones ya sea por errores encontrados o por implementar nuevas capacidades. La fork a pesar de ser una copia de un repositorio original, pertenece a la persona que ha hecho la copia como un repositorio cualquiera. Como cualquier repositorio el usuario puede invitar colaboradores. Una vez el usuario esté satisfecho con los cambios que ha hecho en la fork, puede proponer un “pull request” al propietario del repositorio original, para que absorba los cambios que ha hecho en la fork. Asimismo, la fork se puede mantener sincronizada con el repositorio orignal.

Se han seguido los pasos descritos en esta web: <https://help.github.com/articles/fork-a-repo/>

Desde la cuenta de Aratz (aratzga), se ha ido al repositorio florence de Roman (romeric) y se ha creado una fork. En la fork de Artz, se ha invitiado a Miquel (miquelaguirre) para que sea colaborador de esa fork. Más adelante, Aratz podrá invitar a nuevos colaboradores de la misma manera.

Después se ha trabajado para configurar Git en Ubuntu de la máquina virtual (https://help.github.com/articles/set-up-git/) . Se han seguido los pasos 1, 2 y 3 sin problema del apartado “Setting up Git”. No queda claro como hacer la autenticación automática del paso “Next steps: Authenticating…”. De la experiencia de Michigan, recuerdo que teníamos que usar HTTPS, pero no tengo claro como hacer el cache del password.

Paso 2: crear un local clone del fork

*Paso XX instalar una GUI par GIT*

Había varias opciones (<https://git-scm.com/download/gui/linux> ), pareciendo Gitkraken la mejor, pero parece limitada a uso no comercial. No me queda claro que se pueda instalar en el servidor de Tecnalia. En su lugar, Git Extensions, que es libre parece una alternativa buena.

He seguido los pasos de esta web: <https://github.com/gitextensions/gitextensions/wiki/How-To:-run-Git-Extensions-on-Linux>

En realidad ha hecho falta seguir estos pasos también:

<https://www.codeofclimber.ru/2015/how-to-install-gitextensions-in-ubuntu/>

TEST PARA COMMIT