Installation of SU2 on a Virtual Box

For the tutorials and the projects of the course, we will use the software **SU2** (https://su2code.github.io/). On the **SU2** website https://su2code.github.io/, you can find much information about this collection of simulation software tools. It is *open-source software*, this is why you can download, use, and even modify, it for free. However, open-source does not mean that you can do whatever you want with the software: terms and conditions for copying, distribution, and modification are defined under the GNU Lesser General Public License (for more info, https://github.com/su2code/SU2/blob/master/LICENSE.md).
There are different ways to get the code, as illustrated on the page Download: the first one is the pre-compiled version (*Binary Executables*), which can be convenient if you want just to use the software as provided, without the latest features or modifications. Conversely, we will compile the software, i.e., we will build the executables, which are machine-dependent. To do that, we can use the sources available in section *Source Code*, (but in this case, we will be stuck in the current version) or we can **clone** the latest version from **GitHub** https://github.com/su2code/SU2, where the branch "*master*", the default one, always contains the latest stable version. This means that, in the future, if some interesting features or bug fixes will be published online, we can easily include them in the version we will have on our laptop (more about the git repository of **SU2** on https://su2code.github.io/docs_v7/Gitting-Started/)
Probably, the most useful sections of the website are Docs and Tutorials. The former explains the basic usage of the software, the fundamental theory behind it, as well as how to set up a simulation, and includes the FAQs, with a link to the CFD online forum. The *Tutorials Collection* includes different examples, some of which will represent the starting point for our laboratories.

This tutorial shows how to install **SU2** under Windows, using a virtual machine (VM) with a Linux operating system (OS). In this way, we will start the installation from scratch in an environment that should be the same for everyone.
The same procedure applies to installing the VM on Linux and Mac OS. However, if you are already using a Linux distribution (like Ubuntu) or Mac OS, you can compile **SU2** there, without the VM. The process is similar to the one described for the VM with Ubuntu 20.04, but you can find instructions on the page https://su2code.github.io/docs_v7/Installation/. If you are having trouble running the procedure on your OS, you can install the Virtual Machine and follow the tutorial step-by-step.

If interested, you can also create a partition in your hard disk and install **Ubuntu 20.04** on it, using the dual boot to choose which OS starts when you turn on the computer. In this case, you need first to create a bootable USB stick, following, for instance, the tutorial for Ubuntu or Windows. Then, you can find online several tutorials that explain how to install Ubuntu alongside Windows. Finally, for the compilation of **SU2**, you can follow the instructions in the dedicated section below.

Download an ISO image of Ubuntu OS

The first thing you need to do, either to install **SU2** in a VM or to create a dual boot, is to download an ISO image of Ubuntu. We will use version 20.04 because we noticed some problems using the 22.04 in parallel under the virtual machine.
It can be downloaded from this link (it may require 30-40 minutes, so do it in advance):
https://releases.ubuntu.com/20.04.5/ubuntu-20.04.5-desktop-amd64.iso

Install the Virtual Machine on your OS

1 Download the installer files:

We will use the *Oracle VM Virtual Box.*
Download it from the website https://www.virtualbox.org/wiki/Downloads
Select the "*VirtualBox 6.1.38 platform packages*" and then your OS, e.g. "*Windows hosts*".
The main steps needed to create and configure the VM are briefly summarized in the followings. If you want to know more, or you encounter some problems, you can have a look at the comprehensive documentation in the *User Manual*: https://www.virtualbox.org/manual/UserManual.html

2 Install the Virtual Box

Run the executable you have just downloaded and allow it to make modifications to the system.

- A wizard, as for the installation of any standard package in Windows, is started, with a welcome message. Go to the next step.
- Leave the default settings in this step (Fig 1). In the next tab, you can choose if you want a shortcut in Start Menu, Taskbar, Desktop, etc. Then, a warning message says that, during the installation, the internet will be temporarily switched off: that's OK, go on. Then, **Start** the installation.

At the end of the installation, launch the installed Virtual Box.

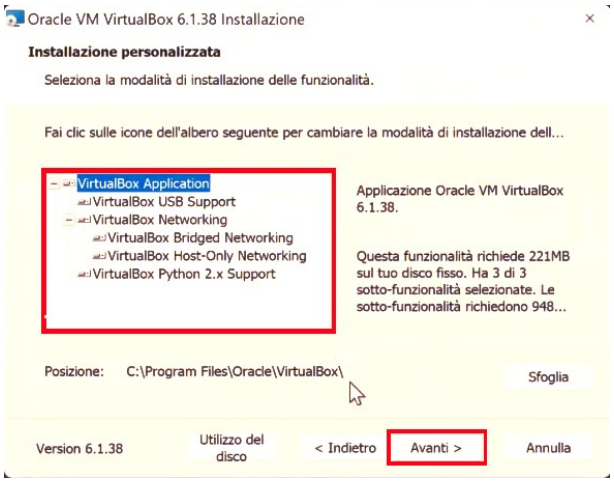


Figure 1: Oracle VM installation wizard, default settings in the first tab of install.

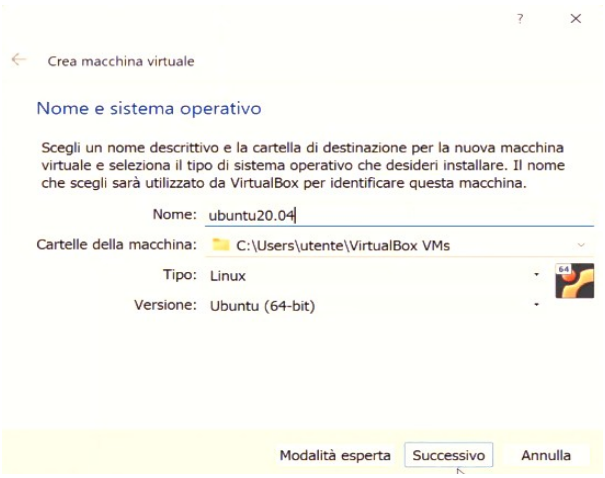


Figure 2: VM creation, assign name, type, and version.

3 Create the Virtual Machine (VM)

Create a new virtual machine, with the blue button **New**. Follow the wizard to create the VM:

- In the first step, you need to give a name to the VM. Let's call it "*ubuntu20.04*": the software recognizes that we are asking for a *Linux* type, with *Ubuntu* version (Fig 2).
- Choose the size of RAM memory assigned to the VM. Suggestions are:
  - 4096 MB if you have 8 GB of RAM;
  - 8192 MB if you have 16 GB of RAM
- Create the memory of the VM. Let's generate a virtual hard disk. Click **Create** and use the default settings: "*VDI (VirtualBox Disk Image)*" type, "*Dynamically allocated*". Select at least 20.00 GB of space.
- Click on **Create**.

4 Setup the ubuntu 20.04 VM

Now, the **ubuntu20.04** VM is listed in the left column, but it needs to be configured. Click on it and then click the yellow gear icon **Settings** (Fig. 3). The *Settings* window opens:

- In the **General** category, the *Basic* tab lists the name of the VM, and the type and version of the OS you want to install. They should be Linux and Ubuntu.
- In the **System** category, go to the *Processor* tab. Set the number of CPU cores. Suggestions:
  - assign 4 CPU cores to the VM, if you have 8 physical cores,
  - assign 2 CPU cores to the VM, if you have 4 physical cores,
  - you should not assign more CPU cores than are available physically.

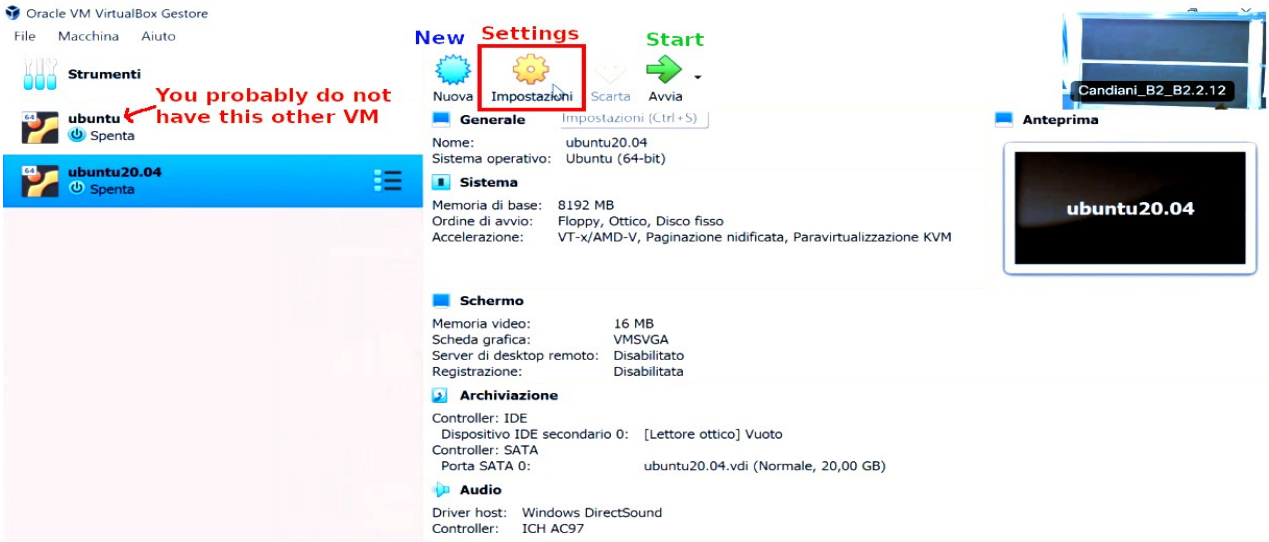
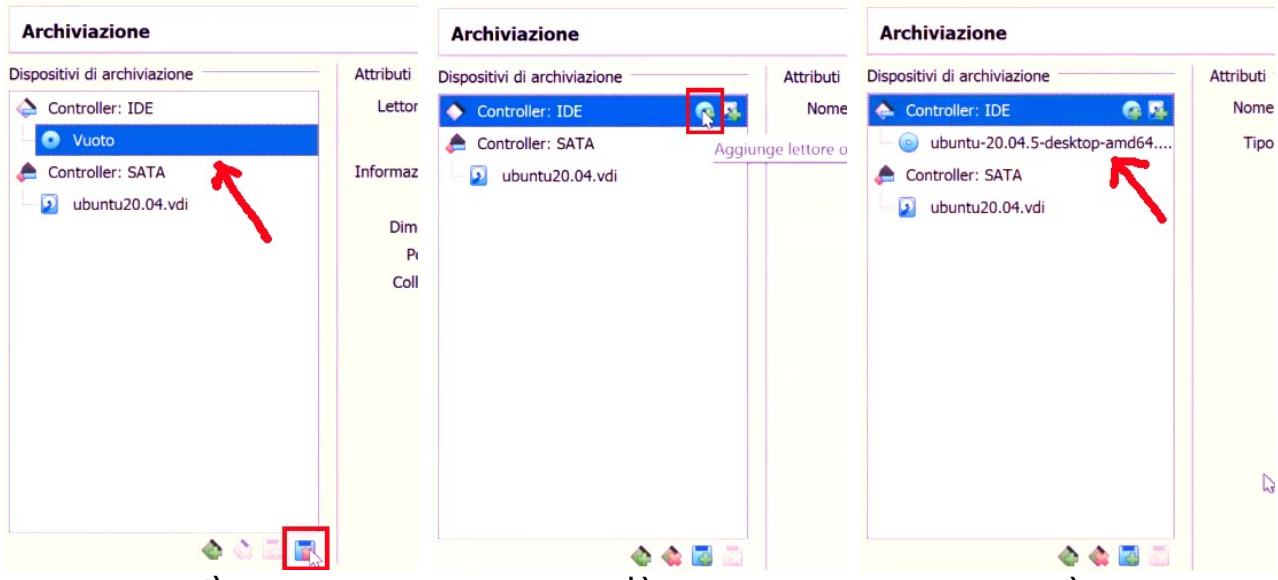


Figure 3: Oracle VM overview. In the left column, the available VMs are listed (in blue the selected one). On the right, at the top the icons New, Settings, and Starts; below, the summary information about the selected VM.

- Let's go to the **Storage** category, in order to enable the connection between the VM and the virtual hard disk, where you want to have the Ubuntu system.
  - Under the *Storage Device* list, you should have two storage devices: an IDE controller and a SATA controller. The former one has no devices attached. Click on the label "*Empty*" below it, and remove it using the **Remove** icon (a floppy disk with a red X, in the bottom right corner, Fig. 4a)
  - Now you need to say the virtual machine to use the disk image of Ubuntu (the ISO file you have downloaded before) to be used as a disk. To do so, select the IDE Controller and click the **Add Disk** icon (a CD with a green + sign, on its right, Fig. 4b). A dialog is displayed: you can already have the ISO file listed there, or you may need to add. In this second case, click the **Add** icon and browse to the location where the ISO file is, select it and click **Open**. Then, click **Select**.
  - Now, under the IDE controller, you should have the ISO image of Ubuntu 20.04, as in Fig. 4c.
- We have finished the configuration, click **OK** to close the *Setting* windows.



a) b) c)

Figure 4: Storage settings, assignment of the ISO image as a disk. a) the initial configuration with the "Empty" device to the deleted; b) the icon to add a new disk; c) the final configuration where the ISO image of Ubuntu 20.04 is listed.

5 Start the virtual machine

Let's start the virtual machine, by clicking the **Start** icon (green right arrow).

6 Install Ubuntu on the VM

When you start the VM just configured, it asks to select the disk with the OS. Check that the ISO image of Ubuntu 20.04 is selected. Then, click **Start**. A disk check may be performed.

The wizard will guide you through the installation. In addition, you can find several guides online, e.g., from here on https://ubuntu.com/tutorials/install-ubuntu-desktop#4-boot-from-usb-flash-drive:

- In the Welcome tab, choose to **Install Ubuntu**. Then, choose the keyboard layout.
- Next, you will be prompted to choose between the *Normal installation* and *Minimal installation* options. The **Minimal installation** is sufficient for our scopes.
- In the *Installation Type* tab, select **Erase disk and install Ubuntu**, because we want to have only Ubuntu on the VM disk.

  Important note: if you are not on the VM, but you are creating a dual-boot, you probably do not want to erase the disk, so you need to select **Something else** to manage the partitions.
- Click **Install Now**, a popup asks for a confirmation about the partitions on which Ubuntu will be installed. If everything looks fine, click on **Continue**.
- Select the location and time zone, then you need to enter your name, which is used to generate a computer's name (you can modify it if you want). Then, you need to decide username and password. Since you are installing Ubuntu on the VM, you can choose to log in automatically, without entering the password because you probably have a password in Windows. Anyhow, remind that password, because it is required for other tasks, e.g., to run commands with **sudo**. When you have entered all details, as in Fig. 5, click **Continue**.

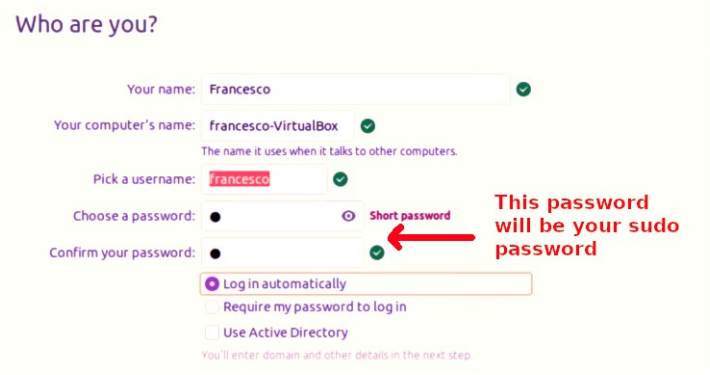


Figure 5: Ubuntu installation, assign user name and password.

- The installation process takes a while. In the end, you have Ubuntu installed in the VM.
- The system may ask you to install updates, especially if you do not have downloaded them during the installation. You can accept, however, if the system asks to update to Ubuntu 22.04, do not accept it.
- Familiarize yourself with Ubuntu and with the terminal. For instance, you can follow this tutorial: https://ubuntu.com/tutorials/command-line-for-beginners, but you can find many similar ones online.

7 How to switch off

You can shut down Ubuntu, from the top right corner menu.

Otherwise, you can **Close** the Oracle VM. Then, it asks you whether you want to save or power off:

- **Save state**: leave Ubuntu OS as it is now, you can restart exactly from where you are; it is similar to suspending a PC.
- **Send Shutdown**: it properly shuts down Ubuntu; it is the same as clicking *Power off* in Ubuntu.
- **Power off the machine**: it stops abruptly the VM; it is the same as pulling out the power connector on a desktop PC, so it can potentially cause data losses.

An alternative way to use Ubuntu under Windows:

For Windows 11 (Build 22000 or later), you can use Windows Subsystem for Linux (**WSL**, https://ubuntu.com/wsl), version 2.
If you don't use Windows 11, it may be impossible to have a graphical interface.
For the installation, you can follow one of the guides online, for instance: https://ubuntu.com/tutorials/install-ubuntu-on-wsl2-on-windows-10#1-overview

SU2 compilation

Preliminary steps

The compilation of **SU2** requires some preliminary steps to install the required software packages. To install them, open a terminal and use the **apt-get** software. During the installation of these packages, it may ask to continue, say "yes" prompting Y. You can run this terminal from any folder on your computer, as it installs the packages in the File System (this is why you need the **sudo** in from of the command)

- Install the program **git**, with:
  ```
  sudo apt-get install git
  ```
- Install some essential libraries for Ubuntu
  ```
  sudo apt-get install build-essential
  ```
- Install **openMPI** for parallel execution (and compilation)
  ```
  sudo apt-get install openmpi-bin libopenmpi-dev
  ```
- Install a collection of enhancements for the management of Python packages
  ```
  sudo apt-get install python3-setuptools
  ```

Note: you can install them sequentially, writing all of them in a single line:
```
sudo apt-get install git build-essential openmpi-bin libopenmpi-dev python3-setuptools
```

Now, we need to copy the source files from the online repository to your local machine. Create and navigate to a folder where you want to install the code.

- For instance, we can create a folder *Codes* in the home:
  ```
  cd
  mkdir Codes
  cd Codes
  ```
  In this way, we are in the folder *Codes*. You can check with the command **pwd**.
  Note: folder and file names in Ubuntu are case-sensitive: *Codes* is not the same as *codes* or *CODES*.
- Let's clone the repository of **SU2**:
  ```
  git clone https://github.com/su2code/SU2.git
  ```

It creates a folder *SU2* where all source files of **SU2** are copied. It may take a while.

Compilation of SU2 from sources

Reference for the compilation process: https://su2code.github.io/docs_v7/Build-SU2-Linux-MacOS/
Let's start the compilation and installation of **SU2** on your laptop.

First of all, we need to decide where we want the executable of **SU2** will be created. Let's create a new folder, called *SU2_bin* in the folder *Codes*:
```
cd ~/Codes
mkdir SU2_bin
```
Within this folder, a subfolder *bin* will be created during the installation, with the executable file.

Then, we can setup the compilation.

- Go to the folder where you have cloned **SU2** and enter the folder *SU2*.
- To check you are in the right folder, use **ls**: you should see several files and folders, among those there is a file **meson.py** (usually written in green, because it is an executable file). This file is a script to configure the environment within which SU2 will be built.
- We can customize some options (see the full list on the webpage mentioned before). We recommend switching off the support for *TecIO* because this library generates the files for *TecPlot*, which is proprietary software for post-process, not available to us. Then, we want to say that the executable file of **SU2** should be created in the folder *SU2_bin*:

```
cd ~/Codes/SU2
./meson.py build -Denable-tecio=false --prefix=/home/username/Codes/SU2_bin
```

  Note 1: the prefix should be edited to match your folder organization. There is a dot in front of /meson.py (it says that the file you want to use is in the folder where you are), a single dash in front of the first option (Denable-tecio), and two dashes in front of prefix.

  Note 2: if you make some errors with this command, to redo the configuration you need to use the flag **reconfigure**, which however takes only one option at once. So for instance, to change the prefix:

```
./meson.py build --reconfigure --prefix=/home/username/Codes/SU2_bin
```

- The last rows of the output of this command should be similar to those in Fig. 6. They suggest you also the next two steps, which are explained below.

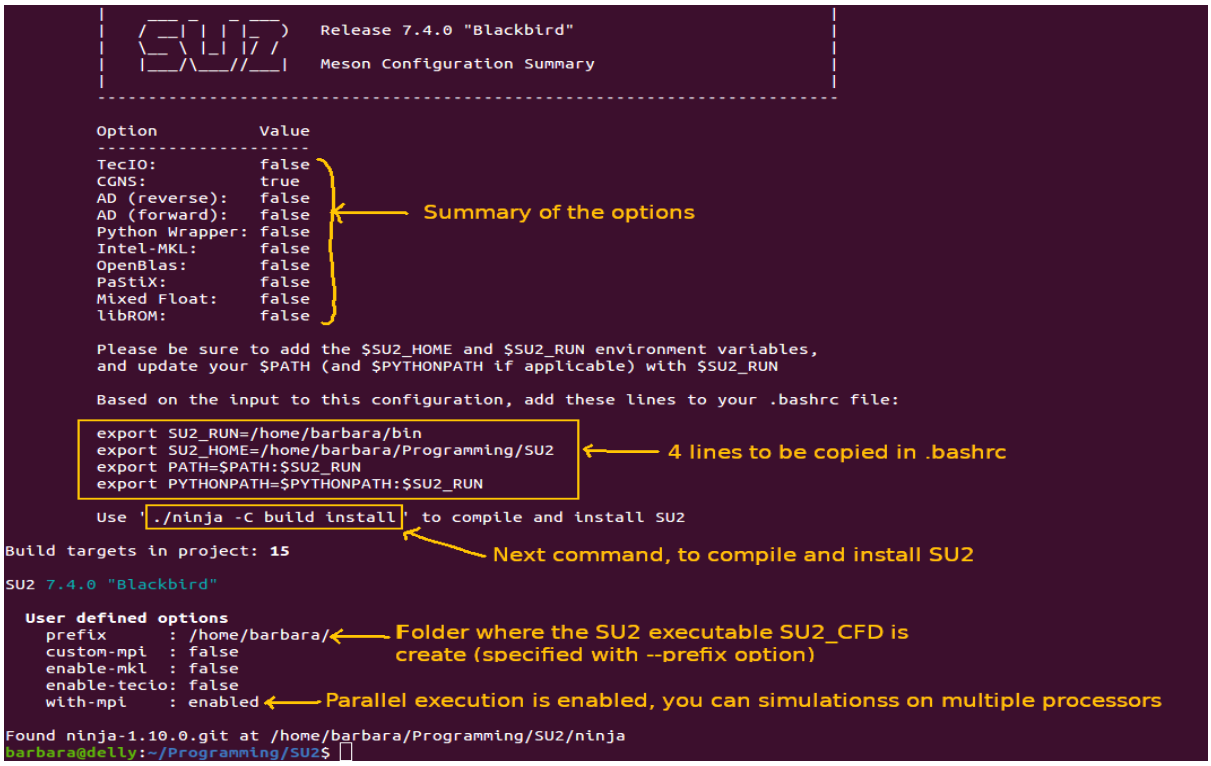


Figure 6: SU2 configuration, last part of the output of the command ./meson.py build -Denable-tecio=false –prefix=/home/barbara
Orange signs highlight and comment the most interesting parts.

Good practice suggests keeping in separate folders the software and the simulations. To be able to run **SU2** from any folder in your computer, you need to export the environment variables of **SU2**. To do this, you need to copy the four lines of the output of the previous command starting with **export**, and paste them in the *.bashrc* file, which is located in the home.

- To open it, you can use the editor you prefer, e.g., **gedit**

```
gedit ~/.bashrc
```

then, scroll down until the end, paste the four lines, and save. It is good practice to add a comment before them to explain what they are, e.g., "#SU2 environment variables" (A line starting with # is a comment in bash), as shown in Fig. 7.

Note: on Mac, instead of the *.bashrc*, you have a *.zshrc* file. You need to copy and paste the same lines.

- The file *.bashrc* is read only when a new terminal is opened, so you have to re-read it with

```
source ~/.bashrc
```

Now, we can finally install **SU2**:

```
./ninja -C build install
```

Note: if you want to speed up the compilation using NN processors, you can add the option -jNN, e.g.:

```
./ninja -C build install -j4
```

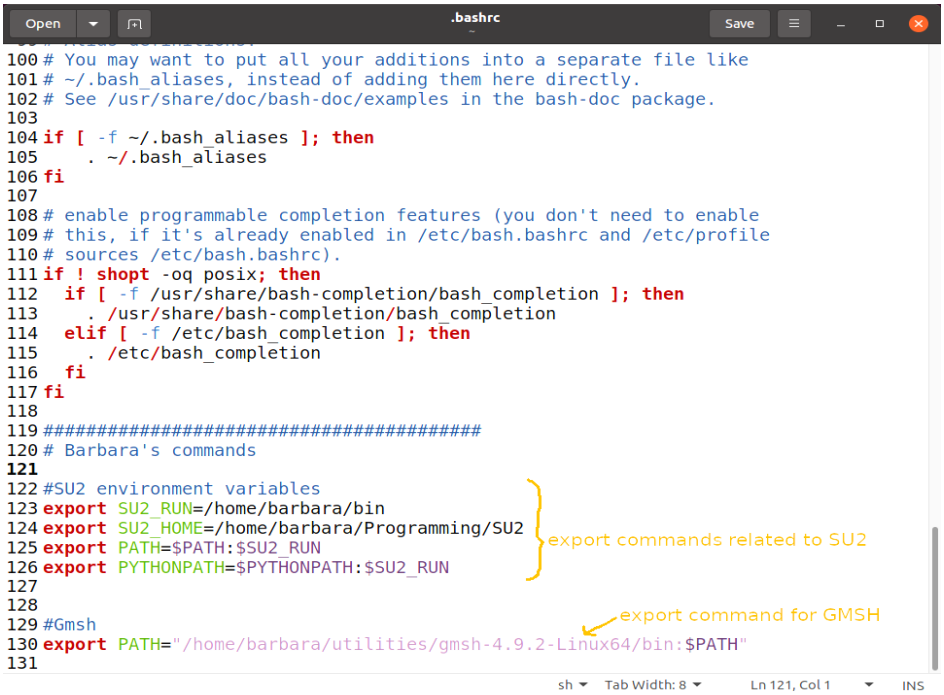


Figure 7: Sample of .bashrc, with the export commands for SU2 (copied from Fig. 6) and GMSH (see sec. Additional software, even if here version 4.9.2 is used instead of 4.10.5). The paths may differ from yours.

Quick start

To check the installation has been successful, you can run the quick-start test case, described on the web-page https://su2code.github.io/docs_v7/Quick-Start/.

From the home folder of **SU2** (which now can be reached also with **cd $SU2_HOME**), enter the subfolder *QuickStart*, where there are two files:

- *inv_NACA0012.cfg*: the configuration file of **SU2**, which is the input file for **SU2** where it reads the options to run the simulations. We will see how to customize it in the next labs
- *mesh_NACA0012_inv.su2*: the file of the mesh in the **SU2** format (note that this is not the .geo file we explained in the lab, but it is the actual mesh file, where nodes, triangles, etc are defined).

To run the test case in serial

```
SU2_CFD inv_NACA0012.cfg
```

or in parallel with NN processors

```
mpirun -n NN SU2_CFD inv_NACA0012.cfg
```

Note: if you are using Ubuntu 22.04, you are probably unable to run the serial command, and you receive an error about MPI. In this case, try the parallel version.

It takes a couple of minutes, it should converge in about 180 iterations. The final part of the log, printed on screen, will be similar to Fig. 8.

Some new files are present in the *QuickStart* folder, among them **flow.vtu** which can be read by **ParaView** to visualize the flow field.

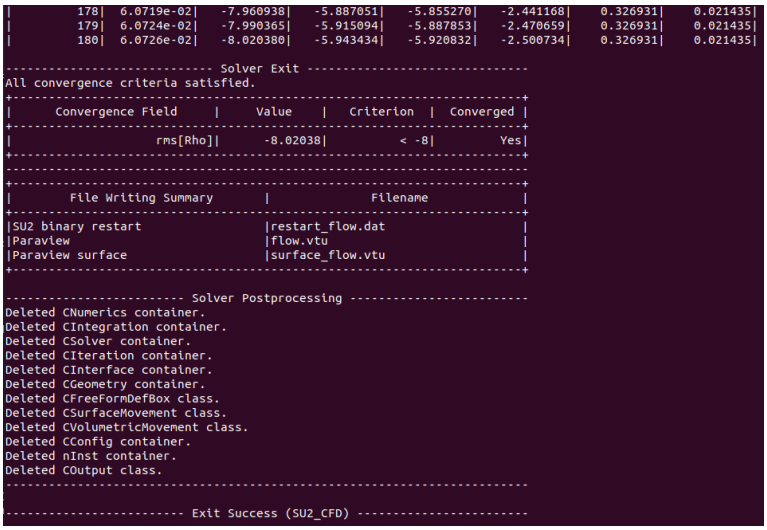


Figure 8: QuickStart test, final part of the log.

Additional software

GMSH

Gmsh is the mesh generator we will use to create the meshes for our simulations. It can be installed in Windows or Ubuntu. Here, we will explain how to install it under Ubuntu.

- Download the binaries from the website https://gmsh.info/:
  In section *Download*, under "*Current stable release (version 4.10.5, 1 July 2022)*": select "*Download Gmsh for **Linux***".
- The file *gmsh-4.10.5-Linux64.tgz* will be saved in the folder *Download*, under the home.
  You can move the file to another folder, for instance to the same folder *Codes*.

```
cd ~/Download
mv gmsh-4.10.5-Linux64.tgz ~/Codes
```

- If you are not already there, navigate to the folder where the file *gmsh-4.10.5-Linux64.tgz* is, e.g.:

```
cd ~/Codes
```

- Un-package the file using the program **tar** (find more about it here: https://www.tecmint.com/18-tar-command-examples-in-linux/):

```
tar -zxvf  gmsh-4.10.5-Linux64.tgz
```

- The executable file is in the subfolder *bin*. To run Gmsh from any folder in your computer, you have to export the path to the binary of **Gmsh**, which is called **gmsh** (in green in the output of **ls**) and should be in the subfolder *bin* of *gmsh-4.10.5-Linux64*. Similarly to what done with the environment variables of **SU2**, you need to add a line to the file .bashrc to export this path. Suppose you have the folder of **Gmsh** under *Codes* in your home, the line to add should be (see also Fig. 7)

```
export PATH="/home/username/Codes/gmsh-4.10.5-Linux64/bin:$PATH"
```

ParaView

ParaView is an open-source platform for data visualization. It can be installed in Windows or Ubuntu.
All information is on the website: https://www.paraview.org/

You can install the latest stable version for your OS, which you can find on the Download page. The procedure is similar to the one described for **Gmsh**. Also in this case, you need to export the path of the executable by adding a line to the *.bashrc* file.
As an alternative, in Ubuntu, you can use **apt-get** and get a slightly older version:

```
sudo apt-get install paraview
```

To launch it, simply prompt **paraview** in the terminal.