Practical 1: Installing Ubuntu, MadGraph5, and Testing with Python

Setup

To begin with, we will need a working Linux operating system installed on our computer. If you're a Windows user, the easiest way to do this is by using the official Windows Subsystem for Linux (WSL), where the instructions to install can be found here. Once installed, you should be able to run the program Ubuntu, which will open a command prompt Window which allows you to interact with the copy of Ubuntu now running on your computer.

To test to see if your installation is working, you can run the command `ls` to list the contents of the current directory. This will be empty, but should run without errors. You can then make a new folder (directory) named 'chep' by running

```
mkdir chep
```

If you run `ls` again, you should now see this folder in the output. To move into this folder, we can change directory

```
cd chep
```

Next, check to make sure Git is installed, by running

```
which git
```

This should tell you where your git installation is. If this doesn't work, make sure git is installed. If this runs without problem we should be able download the latest version of the course repository by running

```
git clone https://github.com/ValentinHirschi/ComputationalHEP.git
```

and navigate into its directory

```
cd ComputationalHEP/
```

We can run `ls` to see the newly downloaded contents. We can also update this repository week by week by using

```
git pull
```

To edit the code for the course, the simplest way is to use VS Code. To open the repository in VS Code, we can run from the current directory

```
code .
```

This should install and open a VS Code window.

Along with our course code, we will also be running MadGraph5. We install this using

```
wget https://launchpad.net/mg5amcnlo/3.0/3.6.x/+download/MG5_aMC_v3.5.7.tar.gz
```

We can unpack this tarball using

```
tar -xzf MG5_aMC_v3.5.7.tar.gz
```

We should now see the decompressed contents when we run `ls`. We can then remove the zipped file, as we no longer need it

```
rm -rf MG5_aMC_v3.5.7.tar.gz
```

We will now have a new directory that we can navigate into

```
cd MG5_aMC_v3_5_7/
```

We will also need few other things installed to get everything to run properly. To ensure this works, first run

```
sudo apt-get update
```

and enter your password when prompted. Then, we need to install Fortran

```
sudo apt update && sudo apt-get install gfortran
```

and type `Y` to proceed. We also need C++ installed, which can be done with

```
sudo apt install build-essential
```

For our python code that we will run later, we will also need a few packages. Firstly, check `python3 --version`, and make sure it is ≥ 3.12. If not, run

```
sudo add-apt-repository ppa:deadsnakes/ppa -y
sudo apt update
sudo apt install -y python3.12 python3.12-venv python3.12-dev
sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.12 1
sudo update-alternatives --config python3
```

To install packages, we need to use `pip`. To install this, run

```
sudo apt install python3-pip
python3 -m ensurepip --upgrade
python3 -m pip install --upgrade pip setuptools
```

Then, we install `symbolica`

```
pip install symbolica
```

and `numpy`

```
pip install numpy
```

Our First Process in Madgraph

Lets try and use Madgraph, and see how it works. To run it, use the command

```
./bin/mg5_aMC
```

You should now see the introduction text for MadGraph5. To see the commands that are part of the package, we can type

```
help
```

If you ever get stuck, you can use the `help` command to view documentation and instructions. Lets try and generate the cross section for a simple process, $e^+e^- \to \mu^+\mu^-$. To do this, we will generate the process:

```
generate e+ e- > mu+ mu-
```

The output should say that we have generated one process with two diagrams. These will correspond to scattering via an intermediate photon, and intermediate Z boson. To see these, we can run

```
display diagrams ./
```

This will generate a file with a drawing of the diagrams (you may not be able to open it though on the default Windows installation).

Obviously, if we want to study Quantum Electrodynamics, we don't want to consider the process mediated by the Z boson. We can ignore this process by using instead

```
generate e+ e- > mu+ mu- / z
```

To compute the result, we can firstly output the process

```
output
```

and then begin by typing

```
launch
```

This will begin running the code. We can choose now to adjust some parameters (we have 60 seconds to decide if we want to do this). We won't adjust any of the first options for now, so enter `0`. On the next screen, we would like to edit some run paramters, so enter `2`. This will enter an editor window. You can navigate the text using the arrow keys. To edit the text, we enter 'insert' mode by pressing `i`. To stop editing, press `Esc`. To save and exit this screen, type `:w` to write your edits, then `:q` to exit to the previous screen.

We would like to make a few edits.

Firstly, in the Standard Cuts section, we would like to remove the cutoffs. We should set also `ptl`, the minimum, to `0.0`.

We would also like to set the remove the upper bound on rapidity by setting `etal` to `-1`.

We will also remove the minimum distance between leptons by setting `drll` to `0.0`.

You can then exit to the previous screen, and run the process by entering `0`. This will run and produce a numerical calculation of the cross-section! It should be something like

```
Cross-section :   0.09286 +- 2.585e-05 pb
```

We can now exit Madgraph by typing

```
exit
```

Our First Process in Python

The course code we downloaded contains a Python script that we can use to compute the same process. Lets navigate back up to the project directory with `cd ..` to where our python code will be. Lets open again VS Code by typing `code .`. Inside the `experiments` folder, there is a file called `epem_lplm_fixed_order_LO.py`. This is a Python script that implements the same scattering process that we just computed in MadGraph. We will be trying to understand exactly what this code does, and how to modify it to model processes other than leptonic processes $e^+e^- \to l^+l^-$.

Let's first trying running the code as it is. To see how to do this, we can firstly go back to the terminal, and run

```
./run.py --help
```

This will give us some documentation. It tells us we can run the experiment `epem_lplm_fixed_order_LO`. Let's do this

```
./run.py epem_lplm_fixed_order_LO --seed 3
```

Here, for testing purposes, we specify the random number seed to use in order to get the same result each time. You should see the code run for a few iterations and produce a numerical result for the cross-section, for example

```
Iteration 9: 0.0929257 +- 0.000163142, chi=1.1246
```

This is quite close to the Madgraph value, indeed they lie within each others uncertainty bounds.