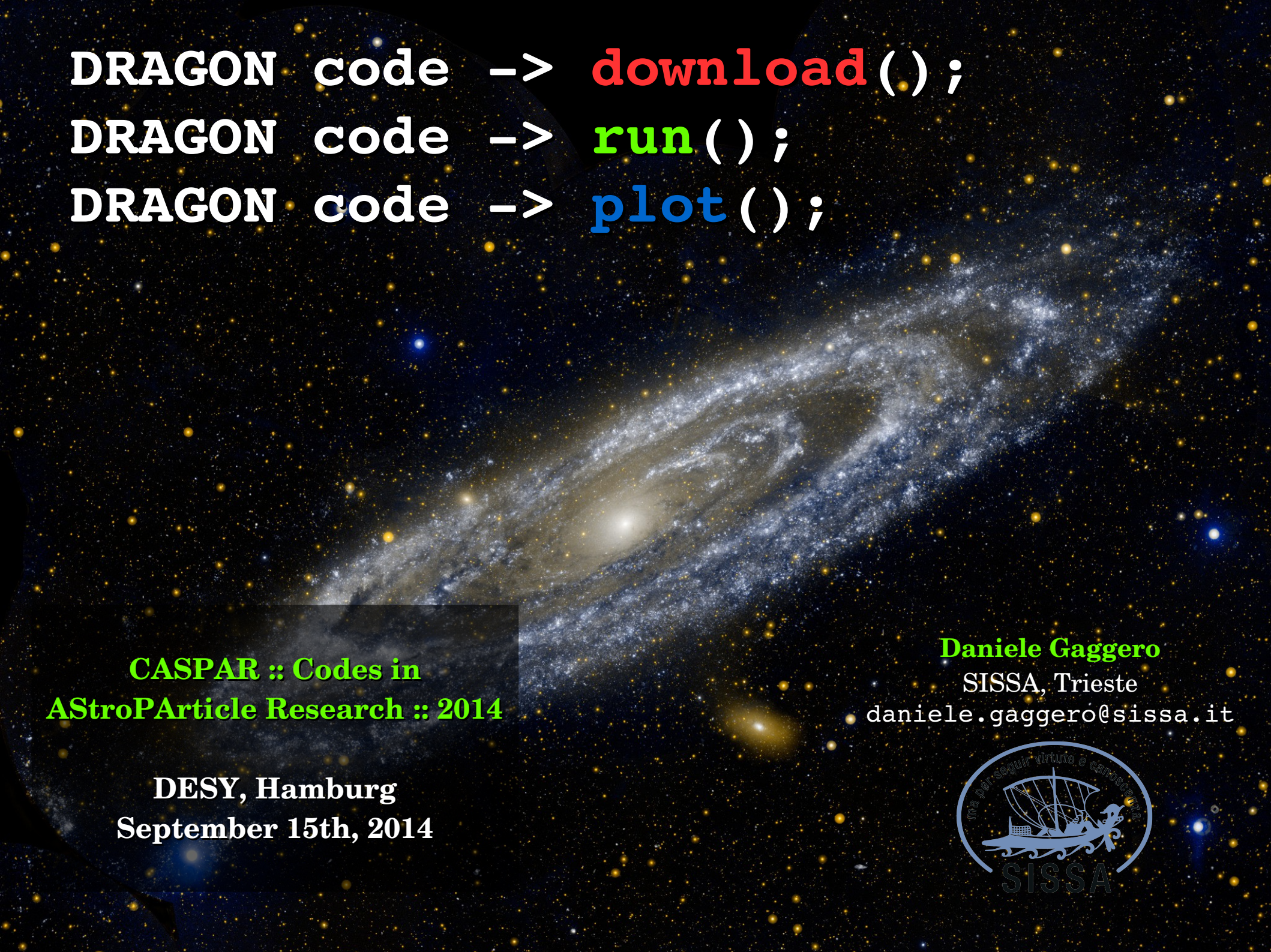


CASPAR :: Codes in AStroPArticle Research :: 2014

DESY, Hamburg
September 15th, 2014

Daniele Gaggero
SISSA, Trieste
daniele.gaggero@sissa.it

1 -> Downloading DRAGON

DRAGON can be downloaded from

→ http://www.dragonproject.org/Download.html

The following libraries are required:

- GSL (http://www.gnu.org/software/gsl/)

- **cfistio** (http://heasarc.gsfc.nasa.gov/fitsio/fitsio.html)

A note on CFITSIO: when you unpack the routine, do

```
$ ./configure –prefix=path_you_prefer
$ make
$ make install
```

and make sure that in path_you_prefer the library (`libcfitsio.a`) is correcly placed in `lib/` and the header files (`*.h`) are located in `include/`

2 -> Installing DRAGON

1) Before installing the code launch the script to initialize installation tools:

```
$ ./start.sh
```

→ a note for MAC users: you may need to edit `start.sh` and put "`glibtoolize`" instead of "`libtoolize`"

2) Configure the code, a typical command line is:

```
$ ./configure --with-cfitsio=$CFITSIO_DIR --with-numcpu=NUMCPU
```

where "**`$CFITSIO_DIR`**" is the path of your `cfitsio` library and "**`NUMCPU`**" is the machine core number.

3) Finally create the executable:

```
$ make
```

3 -> Preparing the xml

I will now explain how to set all the relevant parameters for a DRAGON run;
the parameters for the run must be coded in a single XML file.
There are several sample files provided with the code. Let's see all the blocks one by one.

1. The “Output” block

```
<Output>
    <partialstore />   <!-- if present, the code writes the spectrum at Sun position for each species on a FITS file -->
    <fullstore />      <!-- if present, the code writes the complete (r,z,p) grid of propagated particles for each species on a FITS file; the complete file can be quite large (10-100 MB), depending of the resolution and number of species -->
    <feedback value="2" />
<!-- 0 → no output on screen: ideal for a long set of well tested runs
     2 → a lot of output on screen: ideal for debugging -->
</Output>
```

3 -> Preparing the xml

I will now explain how to set all the relevant parameters for a DRAGON run;
the parameters for the run must be coded in a single XML file.
There are several sample files provided with the code. Let's see all the blocks one by one.

2. The “Grid” block

```
<Grid type="2D">      <!-- Number of spatial dimensions. Options: 2D, 3D -->

        <Observer>  <!-- Position of the Solar System; values are in kpc; the FITS with the spectrum will refer to these coordinates -->
           <x value="8.3" />
           <y value="0.0" />
           <z value="0.0" />
        </Observer>

        <Rmax value="12" />    <!-- Maximum value of Galacto/centric radius (R) in kpc -->
        <L value="4" />        <!-- Halo size in kpc. The Galaxy extends from -L to L -->
        <DimR value="41" />    <!-- Number of grid points along R -->
        <DimZ value="81" />    <!-- Number of grid points along vertical axis -->

        <Ekmin value=".1" />         <!-- Minimum kinetic energy of propagated particles in GeV -->
        <Ekmax value="1000." />      <!-- Maximum kinetic energy of propagated particles in GeV -->

        <Ekfactor value="1.2" />
<!-- Logarithmic spacing of energy grid. E[i] = exp( ln(Ekmin) + i ln(Ekfactor) );
     values closer to 1 result in a more refined grid!
     1.2  is a standard values for runs with little of no reacceleration;
     for runs with large values of v_Alfvén we recommend more refined energy grids, in order to follow the evolution in momentum space at low energies with more accuracies: values around 1.1 or less should be used
-->
```

3 -> Preparing the xml

I will now explain how to set all the relevant parameters for a DRAGON run;
the parameters for the run must be coded in a single XML file.
There are several sample files provided with the code. Let's see all the blocks one by one.

2. The “Grid” block – continued

```
        <NuclearChain>

            <Zmax value="14" />
<!-- Maximum atomic number of propagated particles;
```

if one wants to focus on leptonic species, it is enough to propagate from He (Zmax = 2);

instead, if one wants to compute the B/C ratio it is recommended to start the chain from Silicon in order to get a precision O(1%);

starting from a much heavier nucleus is useless and the corresponding run will take a long time to finish!!!

→

```
            <Zmin value="1" />      <!-- Minimum atomic number of propagated particles -->
            <PropLepton />          <!-- if present, the code propagates leptonic species (optional) -->
            <PropExtraComponent /> <!-- if present, the code propagates a primary extra component of leptons
with a different slope and normalization with respect to the conventional primary electron component (optional) -->

        </NuclearChain>

    </Grid>
```

3 -> Preparing the xml

I will now explain how to set all the relevant parameters for a DRAGON run;
the parameters for the run must be coded in a single XML file.
There are several sample files provided with the code. Let's see all the blocks one by one.

3. The “Algorithm” box

these values may change a lot the performance of the code and more refined values may slow down the run a lot!

```
<Algorithm>
      <OpSplit>   <!-- The code starts with dt = Dtmax; after Nrept iterations, the code rescales dt by the factor Dtfactor; this process is iterated until Dtmin is reached -->

            <Nrept value="30" />            <!-- Number of iterations before changing timestep. 30 is a standard values for runs with little of no reacceleration; for runs with large values of v_Alfvén we recommend larger values, around 60 - 80 -->
            <Dtfactor value=".25" />        <!-- Rescaling factor of the time step     -->
            <Dtmin value="0.001" />         <!-- Minimum time step in Myr -->
            <Dtmax value="64." />           <!-- Maximum time step in Myr -->
      </OpSplit>
</Algorithm>
```

3 -> Preparing the xml

I will now explain how to set all the relevant parameters for a DRAGON run;
the parameters for the run must be coded in a single XML file.
There are several sample files provided with the code. Let's see all the blocks one by one.

4. The “Galaxy” box

```
<Galaxy>

        <Gas type="Galprop" />
<!-- Gas model; options: BronfFerr, NS, Galprop, Uniform -->
        <SNR type="Ferriere" />
<!-- Source distribution for the primary components; options: Lorimer, Galprop, Ferriere, OneRing, Rings -->
        <SNR_Extra type="Ferriere" />
<!-- Source distribution for the extra component; options: the same as SNRType  (optional) -->
        <XCOmode type="SM96" />         <!-- Model for the X_CO factor; options: SM96, galprop_2004, galprop_2010,
constant -->

        <Diffusion type="Constant">
<!-- this is the one of the most important blocks, with the most relevant parameters -->
<!-- type: the spatial distribution of the diffusion coefficient; options: Constant, Exp, Qtau -->
            <D0_1e28 value="2.7" />
<!-- Normalization of the diffusion coefficient at reference rigidity DiffRefRig Unit: 10^28 cm^2/s -->
            <DiffRefRig value = "4" />
<!-- Reference rigidity for the normalization of the diffusion coefficient -->
            <Delta value="0.6" />
<!-- Slope of the diffusion coefficient spectrum -->
            <zt value="4" />
<!-- Scale heigth of the diffusion coefficient, useful in Exp mode: D(z) \propto exp(z/zt) (optional) -->
            <etaT value="1." />
<!-- Low energy correction factor of the diffusion coefficient: D \propto beta^etaT  -->
        </Diffusion>
```

3 -> Preparing the xml

I will now explain how to set all the relevant parameters for a DRAGON run;
the parameters for the run must be coded in a single XML file.
There are several sample files provided with the code. Let's see all the blocks one by one.

4. The “Galaxy” box - continued

```
        <Reacceleration type="Ptuskin94"> <!-- Optional block -->
            <vA_kms value="0." />  <!-- Alfvén velocity in km/s -->
        </Reacceleration>

        <CrossSection type="GalpropXSec" leptopt="Kamae" apopt="GalpropFunction" ApCs="2" />  <!-- Model for cross sections. leptopt is the model for electron and positron production; options: Kamae, GalpropTable -->

        <MagneticField type="Pshirkov"> <!-- Model for the magnetic field. Options: Pshirkov, Farrar, Uniform, Toymodel -->
                <B0disk value="2.e-06" /> <!-- Useful for Pshirkov field: halo regular field normalization in Gauss -->
                <B0halo value="4.e-06" /> <!-- Useful for Pshirkov field: turbulent regular field normalization in Gauss -->
                <B0turb value="7.5e-06" />
        </MagneticField>

    </Galaxy>
```

3 -> Preparing the xml

I will now explain how to set all the relevant parameters for a DRAGON run;
the parameters for the run must be coded in a single XML file.
There are several sample files provided with the code. Let's see all the blocks one by one.

5. The “CR” box

Here we put the normalizations of primary protons, primary electrons, extra component (if present), and the injection indexes for all these species

```
<CR>

        <ProtNormEn_GeV value="100" />   <!-- Reference energy for nuclei normalization in GeV -->
        <ElNormEn_GeV value="33." />     <!-- Reference energy for primary electron normalization in GeV
-->
        <ProtNormFlux value="5.e-2" />   <!-- Proton flux at reference energy   for normalization; in
DRAGON units: GeV^-1 m^-2 s^-1 sr^-1 -->
        <ElNormFlux value="0.004" />     <!-- Electron flux at reference energy for normalization; in DRAGON
units: GeV^-1 m^-2 s^-1 sr^-1 -->
        <ElNormEnExtra_GeV value="300" />       <!-- Reference energy for primary electron extra component
normalization in GeV -->
        <ElNormFluxExtra value="1.e-06" />      <!-- Extra component flux at reference energy; in DRAGON
units: GeV^-1 m^-2 s^-1 sr^-1 -->
```

3 -> Preparing the xml

I will now explain how to set all the relevant parameters for a DRAGON run;
the parameters for the run must be coded in a single XML file.
There are several sample files provided with the code. Let's see all the blocks one by one.

5. The “CR” box - continued

Here we put the normalizations of primary protons, primary electrons, extra component (if present), and the injection indexes for all these species

```
<InjectionIndexAllNuclei>  <!-- You can add an arbitrary number of breaks!! -->
    <alpha_0 value="2.22" />  <!-- First injection slope for nuclei -->
    <rho_0 value="1." />      <!-- Position of first break (rigidity) in GV -->
    <alpha_1 value="2.22" />  <!-- Second injection slope for nuclei -->
    <rho_1 value="11." />     <!-- Position of second break (rigidity) in GV -->
    <alpha_2 value="2.22" />
    <rho_2 value="111." />
    <alpha_3 value="2.22" />
</InjectionIndexAllNuclei>
<InjectionIndexElectrons> <!-- You can add an arbitrary number of breaks!! -->
     <rho_0 value="1." />    <!-- Position of first break (rigidity) in GV -->
     <rho_1 value="5." />
     <rho_2 value="10." />
     <alpha_0 value="1.80" /> <!-- First injection slope for electrons -->
     <alpha_1 value="1.80" />
     <alpha_2 value="2.50" />
     <alpha_3 value="2.50" />
     <CutoffRigEl value="20000." />
</InjectionIndexElectrons>
<InjectionIndexExtraComponent>
     <rho_0 value="1." />
     <alpha_0 value="1.85" />
     <alpha_1 value="1.85" />
     <CutoffRigExtra value="10000." />
</InjectionIndexExtraComponent>


</CR>
```

3 -> Preparing the xml

I will now explain how to set all the relevant parameters for a DRAGON run;
the parameters for the run must be coded in a single XML file.
There are several sample files provided with the code. Let's see all the blocks one by one.

6. The DM block (optional)

Put some description

```
<DarkMatter Reaction="Annihilation" Model="SelfTable" Profile="NFW">
    <!-- Reaction can be "Annihilation" or "Decay", (spectrum) Model can be "SelfTable" or "Delta", (density) Profile can be "Iso", "NFW", "Kra", "Moore", "Einasto" -->
    <PropDMLepton />        <!-- If this flag is specified, leptons originating from DM annihilation/decay are computed -->
    <PropDMAntiProton />    <!-- If this flag is specified, antiprotons originating from DM annihilation/decay are computed -->
    <Mass value="1000." />  <!-- DM particle mass in GeV -->
    <!-- ********** -->
    <!-- LifeTime value="1e26" -->    <!-- if Decay is specified, the lifetime in seconds -->
    <SigmaV value="2.e-23" />          <!-- if Annihilation is specified, the <sigma v> in cm^3/2 -->
    <!-- ********** -->
    <SSDensity value="0.41" />        <!-- Dark Matter local energy density in GeV/cm^3 -->
    <EkDelta value="1000." />          <!-- if Delta is specified as a spectrum model, this is the energy in GeV at which particles are injected -->
    <LeptonDatafile     value="DM/mumu_1000gev_pos.txt" />
    <!-- if SelfTable is specified as a spectrum model, this is the file with the inj spectrum in GeV^-1 for leptons  -->
    <AntiprotonDatafile value="DM/mumu_1000gev_ap.txt" />
    <!-- if SelfTable is specified as a spectrum model, this is the file with the inj spectrum in GeV^-1 for pbar -->
    <!--Channel value="17" /-->
</DarkMatter>
```

4 -> Running and plotting the output

It is easy to perform a run. Once the xml is ready, just launch DRAGON with the xml as parameter:

```
$ ./DRAGON examples/run_2D.xml
```

The code will produce one or two FITS files, depending on the XML settings.

If the <fullstore> flag is on, you will find in output/run_2D.fits the full dataset: for each species there is a block in the FITS file with the CR flux matrix as a function of *(r, z, kin. energy)*

If the <partialstore> flag is on, you will find in output/run_2D_spectrum.fits the spectra at Sun position: for each species there is a block in the FITS file with the CR spectrum as a function of the kinetik energy

4 -> Running and plotting the output

It is easy to perform a run. Once the xml is ready, just launch DRAGON with the xml as parameter:

```
$ ./DRAGON examples/run_2D.xml
```

--- ---

A note on DRAGON FITS files (please look in the web if you need a complete documentation of FITS format)

The FITS files permit to store data and metadata in the same file.

The DRAGON fits files are structured as follows:

1) ***a header*** with the most relevant parameter of the run, e.g. “Emin”, “Emax”, “Rmin”, “Rmax”… these keywords are easily readable with the standard FITS I/O routines with C++, python, IDL ecc.

2) ***a data unit for each species***, with either the spectrum at Sun position or the full matrix.

4 -> Running and plotting the output

In order to visualize the results, we prepared a set of `python` routines to do the most relevant plots.

For those who are not familiar with `python`, don't worry, for each type of plot there is a full working routine and you just have to do some very basic editing for most stuff you'll need!

4 -> Running and plotting the output

In order to visualize the results, we prepared a set of python routines to do the most relevant plots.

Each routine permits to visualize an observable; it is possible to compare *an arbitrary number of runs*, and for every run it's possible to alter the normalization, modulate with a force-field potential and choose the color. Here is an example:

plot_protons.py

```
#/*****************************************************************************
# FILENAMES
#/*****************************************************************************

Input_Folder    = os.path.join(os.environ['HOME'],'python','runs')
Output_Folder   = os.path.join(os.environ['HOME'],'python','plots')
Data_Folder     = os.path.join(os.environ['HOME'],'python','data')

Run_Names = ['run_1', 'run_2']
Number_of_Runs = len(Run_Names)
File_Names = []
for i in range(Number_of_Runs):
  File_Names.append(Input_Folder + '/' + Run_Names[i] + '_spectrum.fits.gz')

norm = [1.0, 1.0]                # normalization factor
phi  = [0.450, 0.550]                  # modulation potential
colors = ['red', 'blue']
```

---> Introductory exercise <---

Run a sample XML and plot the proton spectrum

1) run the code with this sample file:

```
$ ./DRAGON examples/run_2D_simple.xml
```

2) check that the output file has been created correctly

3) edit the python routine `plots/plot_protons_simple.py` plugging the path and name of the run

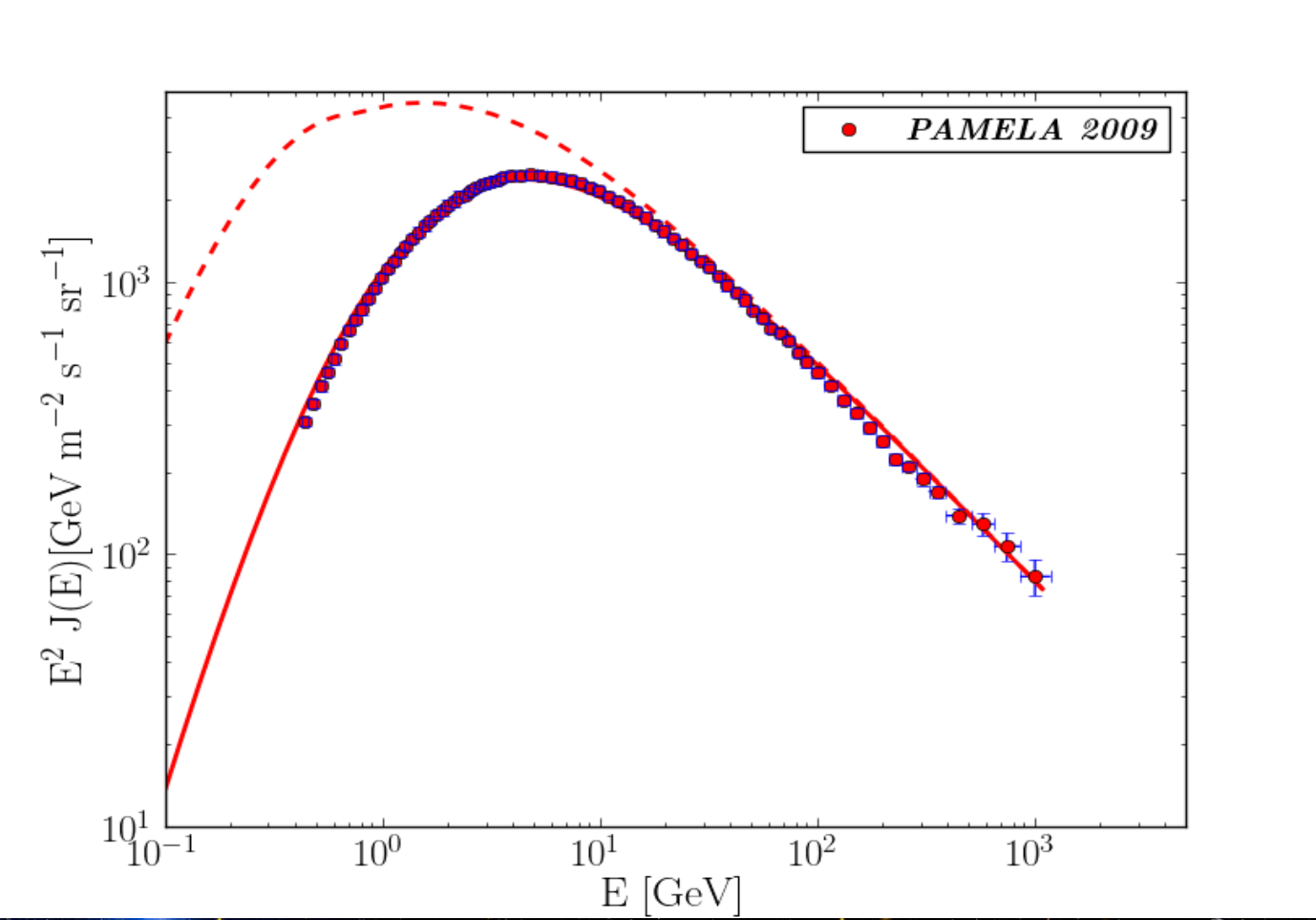
4) run the python plotting routine – just edit it in order to put the correct path and the correct name of the run and you're ready to plot the proton spectrum, with and without solar modulation (the mod. potential is set to 0.52 GV)

```
$ cd plots
$ python plot_protons_simple.py
```

---> Introductory exercise <---

Run a sample XML and plot the proton spectrum

--> this is what you should get:



-->First exercise<--

A reference 2D run in which CR nuclei and leptons are propagated

"examples/run_2D.xml"

```
<?xml version="3.0.2" ?>
<Output>
  <partialstore />  <!-- The code writes the spectrum at Sun position for each species on a FITS file
(optional) -->
  <fullstore />    <!-- The code writes the complete (r,z,p) grid of propagated particles for each
species on a FITS file (optional) -->
  <feedback value="2" />
</Output>
<Grid type="2D">          <!-- Number of spatial dimensions. Options: 2D, 3D -->
  <Observer>
        <x value="8.3" />
        <y value="0.0" />
        <z value="0.0" />
  </Observer>
  <Rmax value="12" />   <!-- Maximum value of Galactocentric radius (R) in kpc -->
  <L value="4" />       <!-- Halo size in kpc. The Galaxy extends from -L to L -->
  <DimR value="41" />   <!-- Number of grid points along R -->
  <DimZ value="81" />   <!-- Number of grid points along vertical axis -->
  <Ekmin value=".1" />       <!-- Miminum kinetic energy of propagated particles in GeV -->
  <Ekmax value="1000." />   <!-- Maximum kinetic energy of propagated particles in GeV -->
  <Ekfactor value="1.2" />  <!-- Logaritmic spacing of energy grid. E[i] = exp( ln(Ekmin) + i ln
(Ekfactor) ) -->
  <NuclearChain>
    <Zmax value="14" />    <!-- Maximum atomic number of propagated particles -->
    <Zmin value="1" />     <!-- Minimum atomic number of propagated particles -->
    <PropLepton />         <!-- The code propagates leptonic species (optional) -->
    <PropExtraComponent />
  </NuclearChain>
</Grid>
```

First exercise

```
<Algorithm>
  <OpSplit>
    <!-- The code starts with dt = Dtmax; after Nrept iterations, the code rescales dt by the factor
Dtfactor; this process is iterated until Dtmin is reached -->
    <Nrept value="30" />          <!-- Number of iterations before changing timestep -->
    <Dtfactor value=".25" />      <!-- Rescaling factor of the time step     -->
    <Dtmin value="0.001" />       <!-- Minimum time step in Myr -->
    <Dtmax value="64." />         <!-- Maximum time step in Myr -->
  </OpSplit>
</Algorithm>
<Galaxy>
  <Gas type="Galprop" />   <!-- Gas model; options: BronfFerr, NS, Galprop, Uniform -->
  <SNR type="Ferriere" />  <!-- Source distribution for the primary components; options: Lorimer,
Galprop, Ferriere, OneRing, Rings -->
  <SNR_Extra type="Ferriere" /> <!-- Source distribution for the extra component; options: the same as
SNRType  (optional) -->
  <XCOmode type="SM96" />        <!-- Model for the X_CO factor; options: SM96, galprop_2004,
galprop_2010, constant -->
  <Diffusion type="Constant"> <!-- Spatial distribution of the diffusion coefficient; options:
Constant, Exp, Qtau -->
    <D0_1e28 value="2.7" />   <!-- Normalization of the diffusion coefficient at reference rigidity
DiffRefRig Unit: 10^28 cm^2/s -->
    <DiffRefRig value = "4" /> <!-- Reference rigidity for the normalization of the diffusion
coefficient -->
    <Delta value="0.6" />    <!-- Slope of the diffusion coefficient spectrum -->
    <zt value="4" />          <!-- Scale heigth of the diffusion coefficient, useful in Exp mode: D(z)
\propto exp(z/zt) (optional) -->
    <etaT value="1." />       <!-- Low energy correction factor of the diffusion coefficient: D
\propto beta^etaT  -->
  </Diffusion>
  <Reacceleration type="Ptuskin94"> <!-- Optional block -->
    <vA_kms value="0." />          <!-- Alfvén velocity in km/s -->
  </Reacceleration>
  <CrossSection type="GalpropXSec" leptopt="Kamae" apopt="GalpropFunction" ApCs="2" />  <!-- Model for
cross sections. leptopt is the model for electron and positron production; options: Kamae,
GalpropTable -->
  <MagneticField type="Pshirkov"> <!-- Model for the magnetic field. Options: Pshirkov, Farrar,
Uniform, Toymodel -->
     <B0disk value="2.e-06" /> <!-- Useful for Pshirkov field: halo regular field normalization in
Gauss -->
     <B0halo value="4.e-06" /> <!-- Useful for Pshirkov field: turbulent regular field normalization
in Gauss -->
     <B0turb value="7.5e-06" />
  </MagneticField>
</Galaxy>
```

First exercise

```
<CR>
  <ProtNormEn_GeV value="100" />  <!-- Reference energy for nuclei normalization in GeV -->
  <ElNormEn_GeV value="33." />   <!-- Reference energy for primary electron normalization in GeV -->
  <ProtNormFlux value="5.e-2" />  <!-- Proton flux at reference energy   for normalization; in DRAGON
units: GeV^-1 m^-2 s^-1 sr^-1 -->
  <ElNormFlux value="0.004" />   <!-- Electron flux at reference energy for normalization; in DRAGON
units: GeV^-1 m^-2 s^-1 sr^-1 -->
  <ElNormEnExtra_GeV value="300" />     <!-- Reference energy for primary electron extra component
normalization in GeV -->
  <ElNormFluxExtra value="1.e-06" />  <!-- Extra component flux at reference energy; in DRAGON units:
GeV^-1 m^-2 s^-1 sr^-1 -->

  <!-- ************************** -->
  <InjectionIndexAllNuclei>  <!-- You can add an arbitrary number of breaks!! -->
    <alpha_0 value="2.22" />  <!-- First injection slope for nuclei -->
    <rho_0 value="1." />     <!-- Position of first break (rigidity) in GV -->
    <alpha_1 value="2.22" />  <!-- Second injection slope for nuclei -->
    <rho_1 value="11." />    <!-- Position of second break (rigidity) in GV -->
    <alpha_2 value="2.22" />
    <rho_2 value="111." />
    <alpha_3 value="2.22" />
  </InjectionIndexAllNuclei>
  <!-- ************************** -->
  <InjectionIndexElectrons> <!-- You can add an arbitrary number of breaks!! -->
      <rho_0 value="1." />   <!-- Position of first break (rigidity) in GV -->
      <rho_1 value="5." />
      <rho_2 value="10." />
      <alpha_0 value="1.80" /> <!-- First injection slope for electrons -->
      <alpha_1 value="1.80" />
      <alpha_2 value="2.50" />
      <alpha_3 value="2.50" />
      <CutoffRigEl value="20000." />
  </InjectionIndexElectrons>
  <!-- ************************** -->
  <InjectionIndexExtraComponent>
       <rho_0 value="1." />
       <alpha_0 value="1.85" />
       <alpha_1 value="1.85" />
       <CutoffRigExtra value="10000." />
  </InjectionIndexExtraComponent>
  <!-- ************************** -->
</CR>
```

-->First exercise<--

1-> run the code

```
$ ./DRAGON examples/run_2D.xml
```
(this will take a couple of minutes)

2-> edit the python routine `plots/plot_protons.py` in order to plot the ***proton spectrum*** corresponding to this run with 3 different modulation potentials: 0.2 - 0.4 – 0.6 GV

3-> edit the python routine `plots/plot_BC.py` in order to plot the **B/C** corresponding to this run with 3 different modulation potentials: 0.2 - 0.4 – 0.6 GV

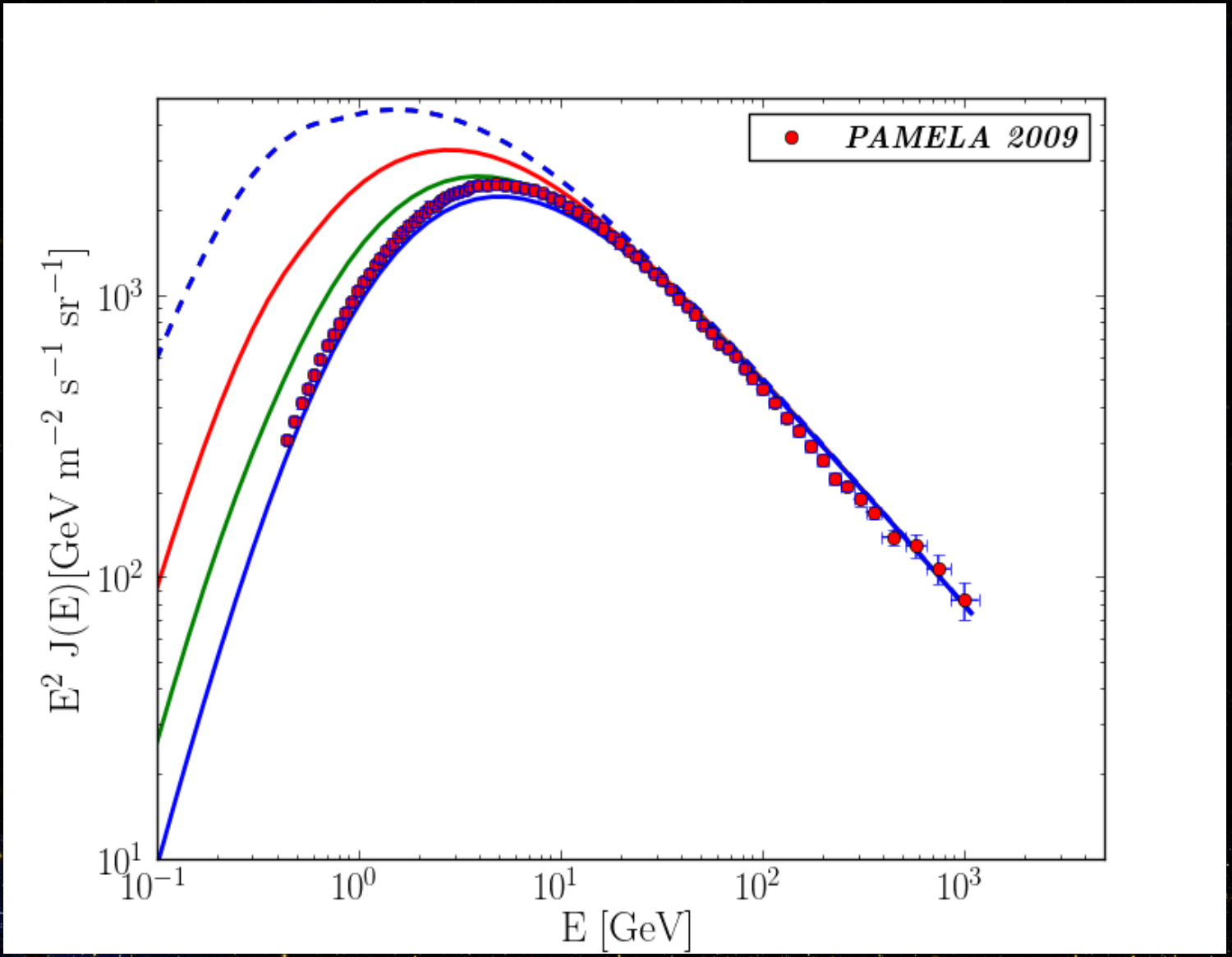
4-> edit the python routine `plots/plot_lepton_fluxes.py` in order to plot the ***leptonic fluxes*** corresponding to this run with 3 different modulation potentials: 0.2 - 0.4 – 0.6 GV

4bis-> edit the python routine `plots/plot_lepton_fraction.py` in order to plot the ***positron fraction*** corresponding to this run with 3 different modulation potentials: 0.2 - 0.4 – 0.6 GV

5-> ***fix the normalization*** of the electrons to 1.2 and the normalization of the extra component to 0.8 in both the flux and fraction routines and plot again!

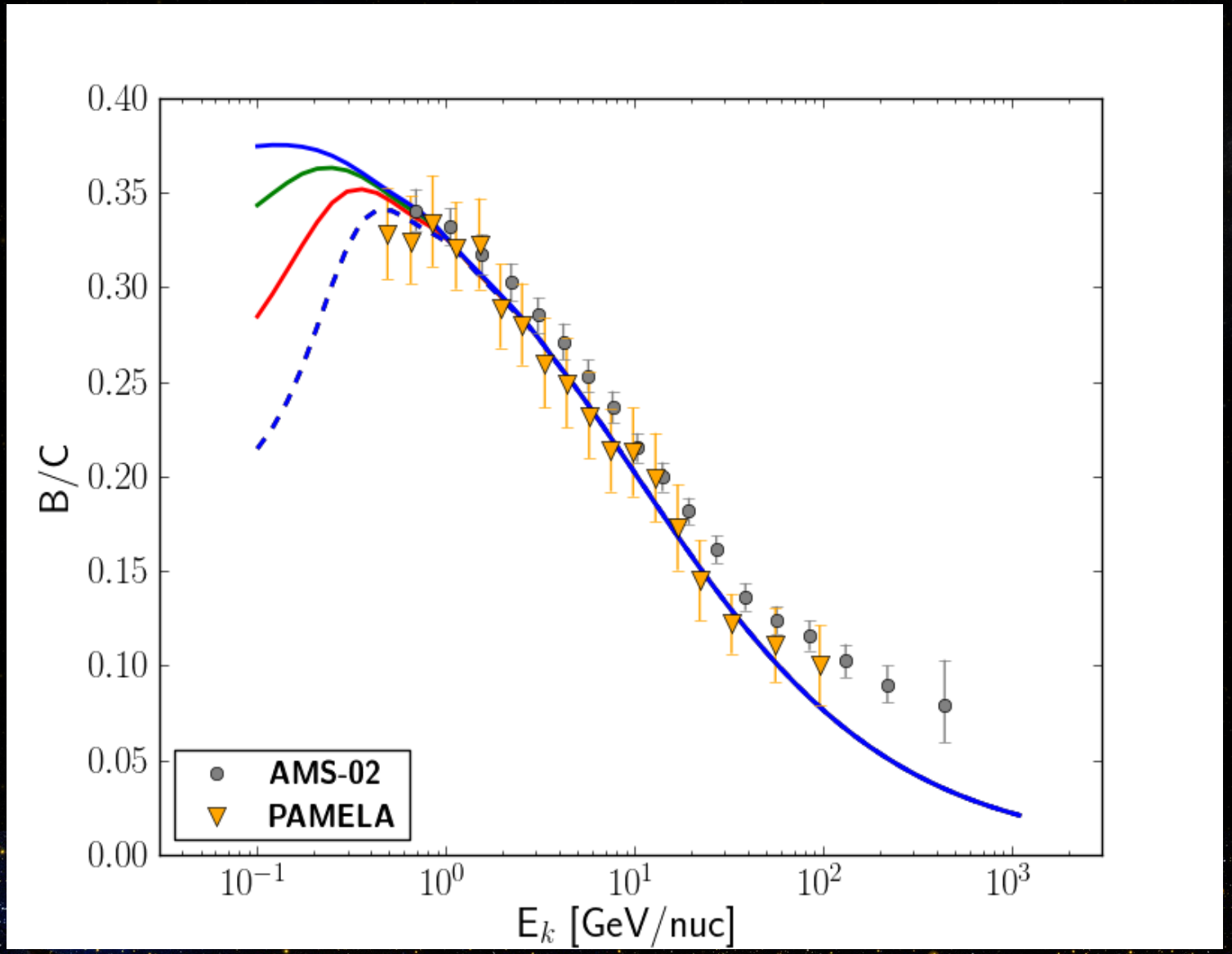
First exercise

-> This is what you should get:



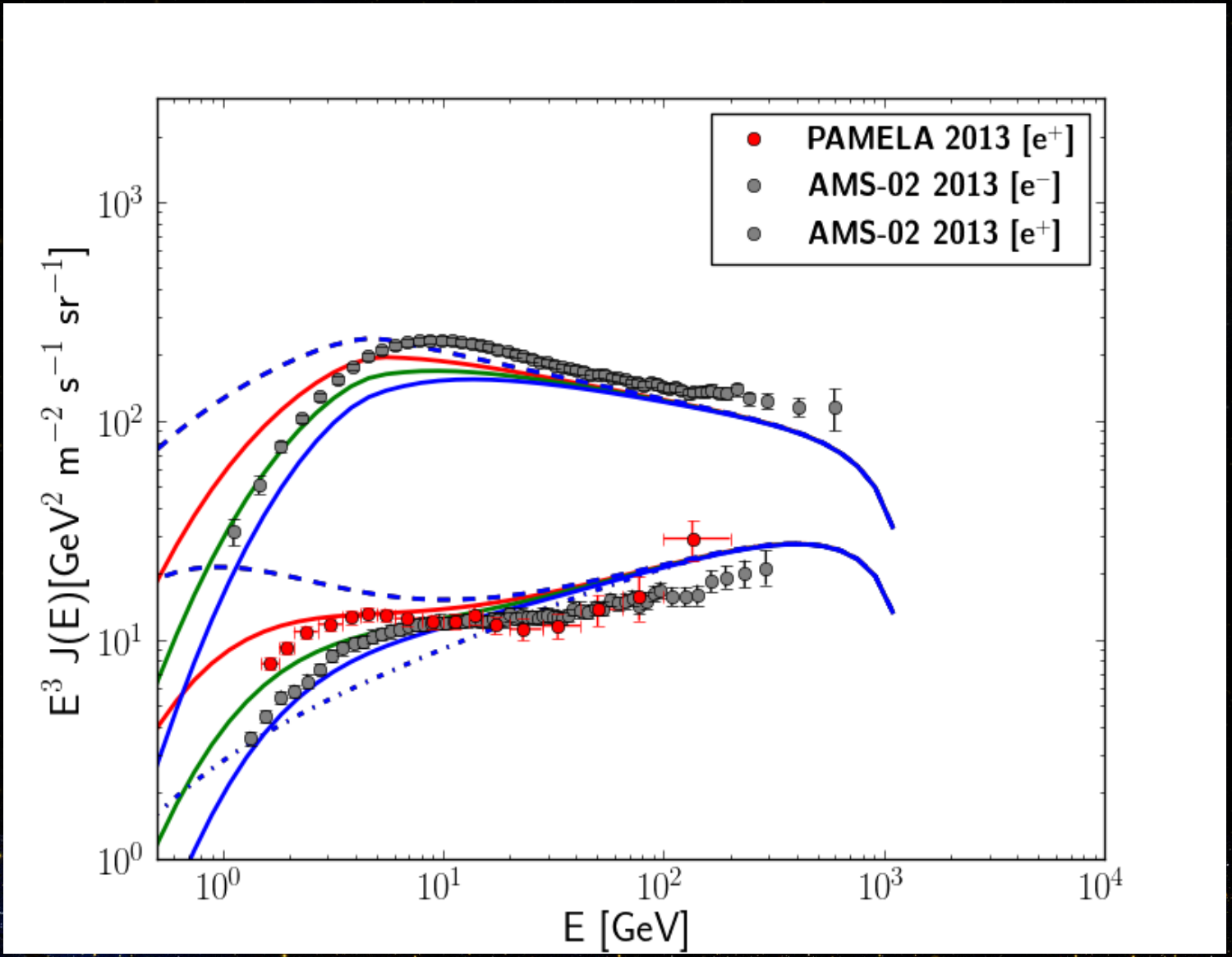
First exercise

-> This is what you should get:



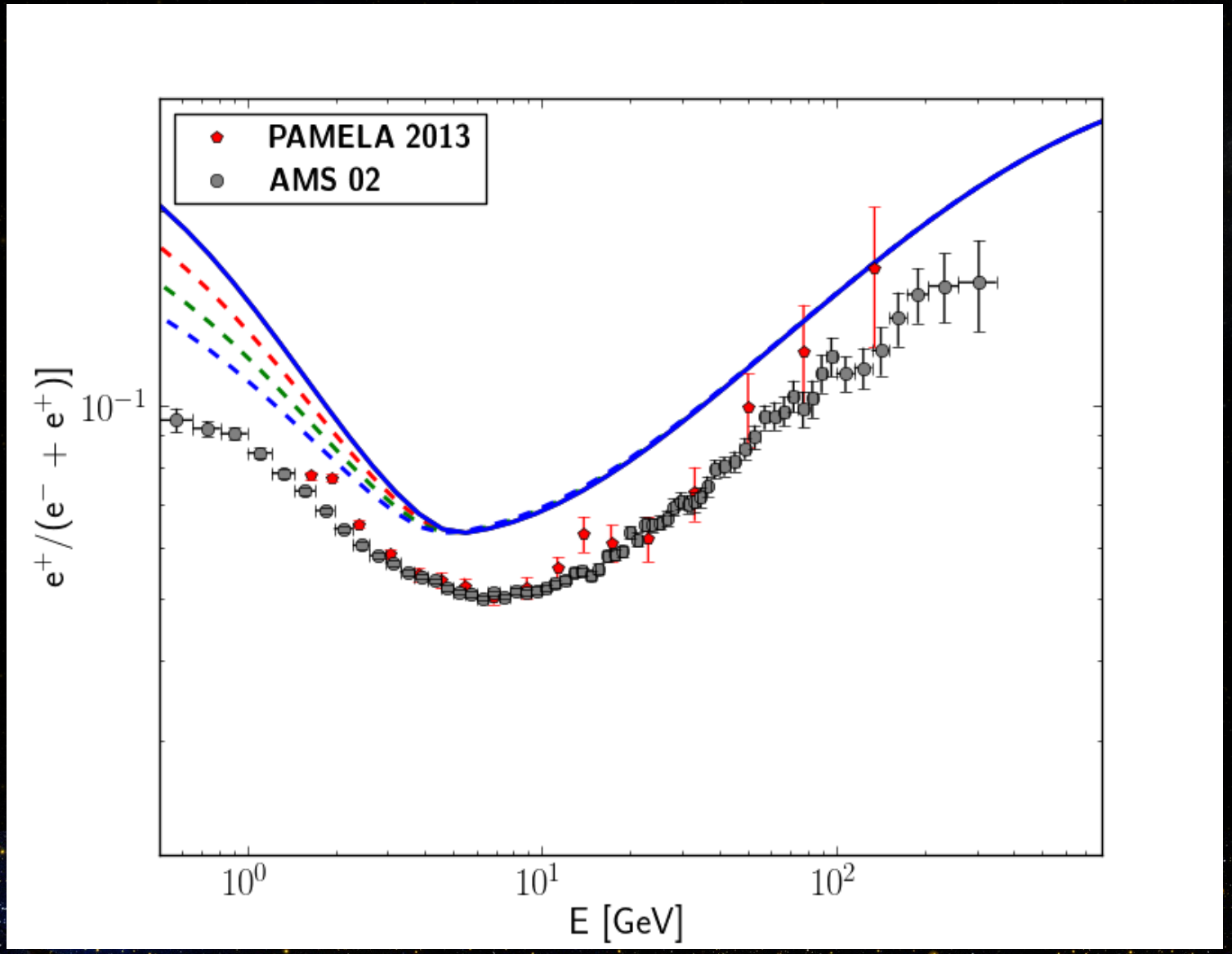
First exercise

-> This is what you should get:



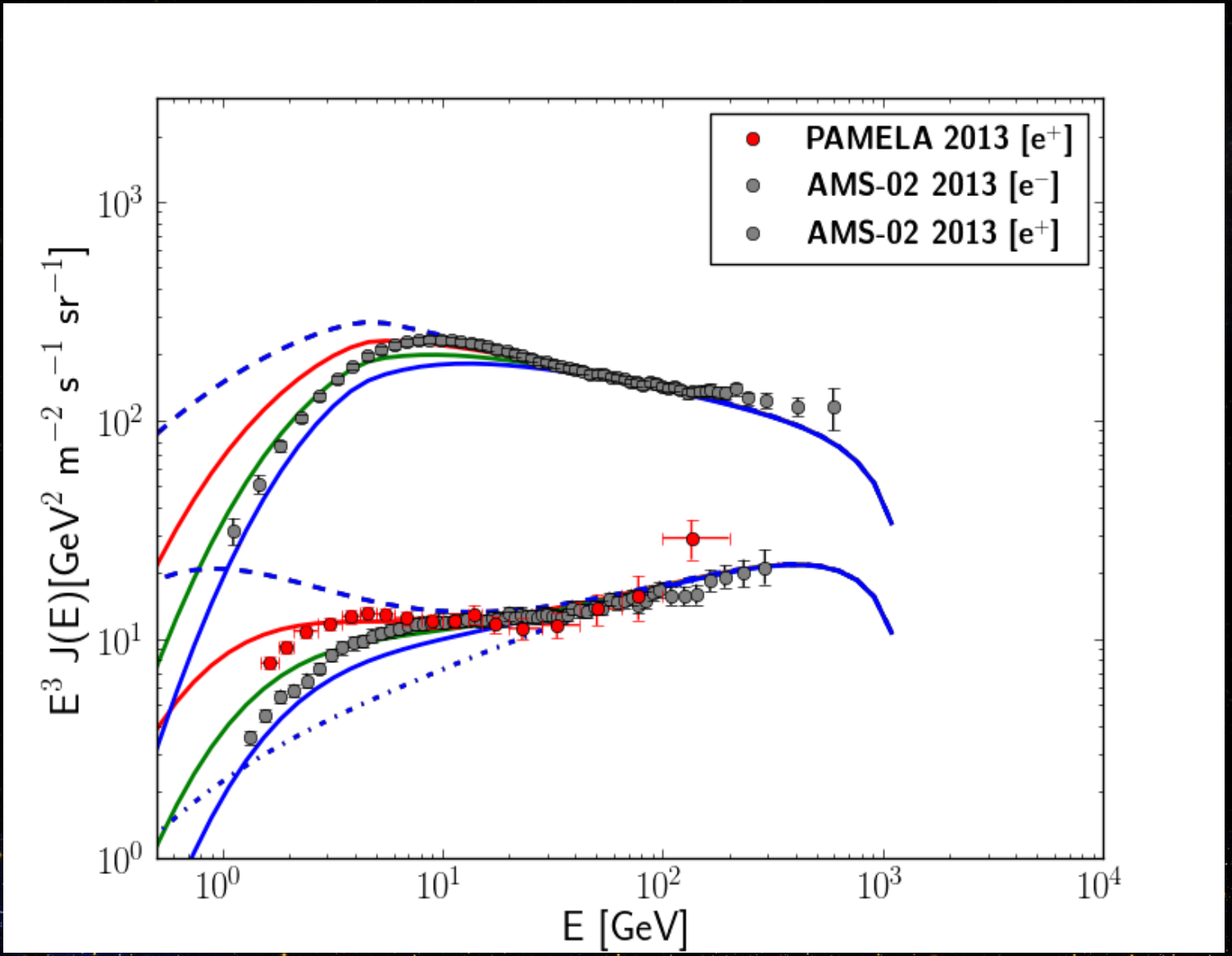
First exercise

-> This is what you should get:



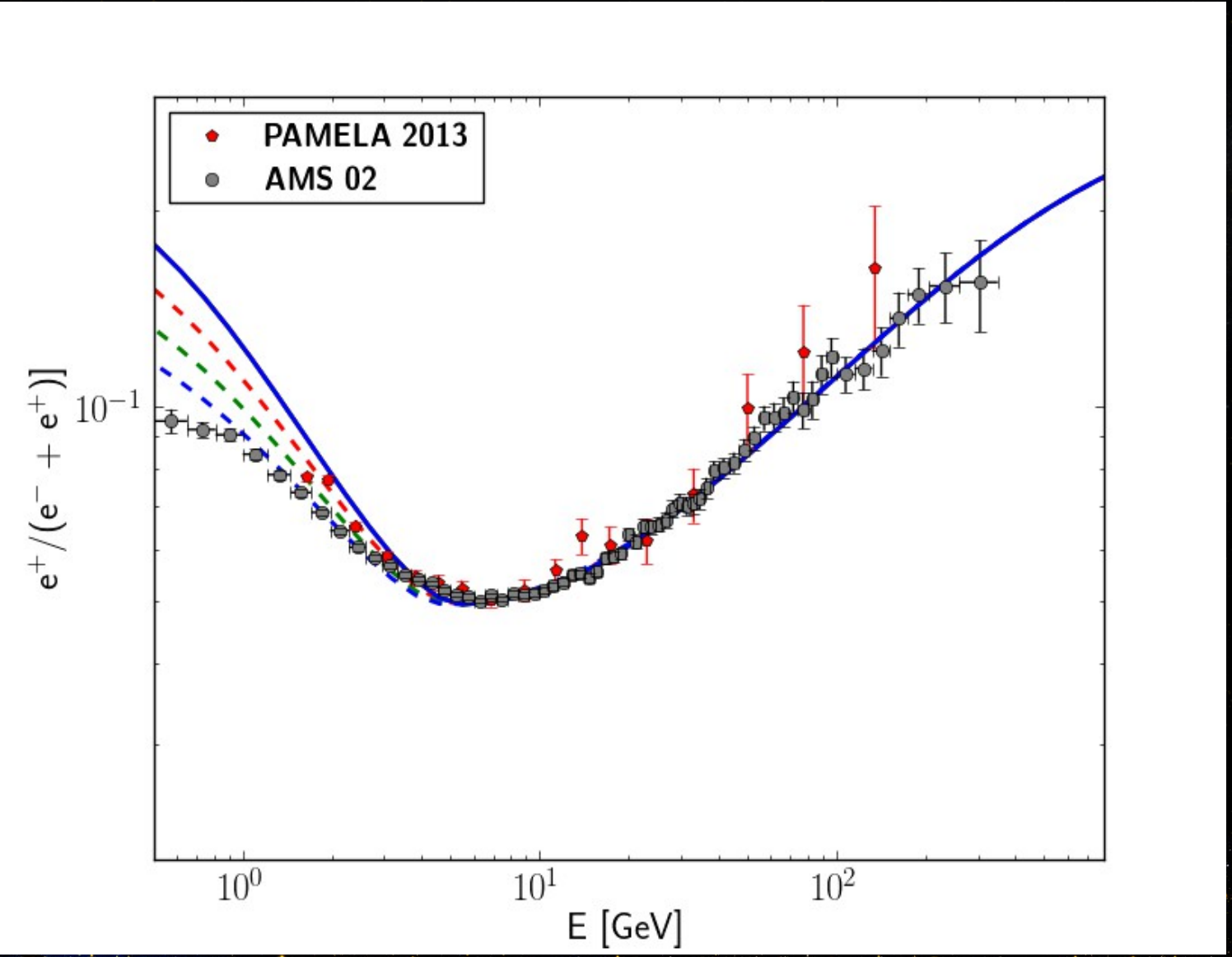
First exercise

-> This is what you should get:



First exercise

-> This is what you should get:



-->Second exercise<--

A reference 2D run in which antiprotons and leptons originating from DM annihilation are propagated

`"examples/run_2D_DM.xml"`

-->Second exercise<--

1-> run the code with the DM xml

```
$ ./DRAGON examples/run_2D_DM.xml
```

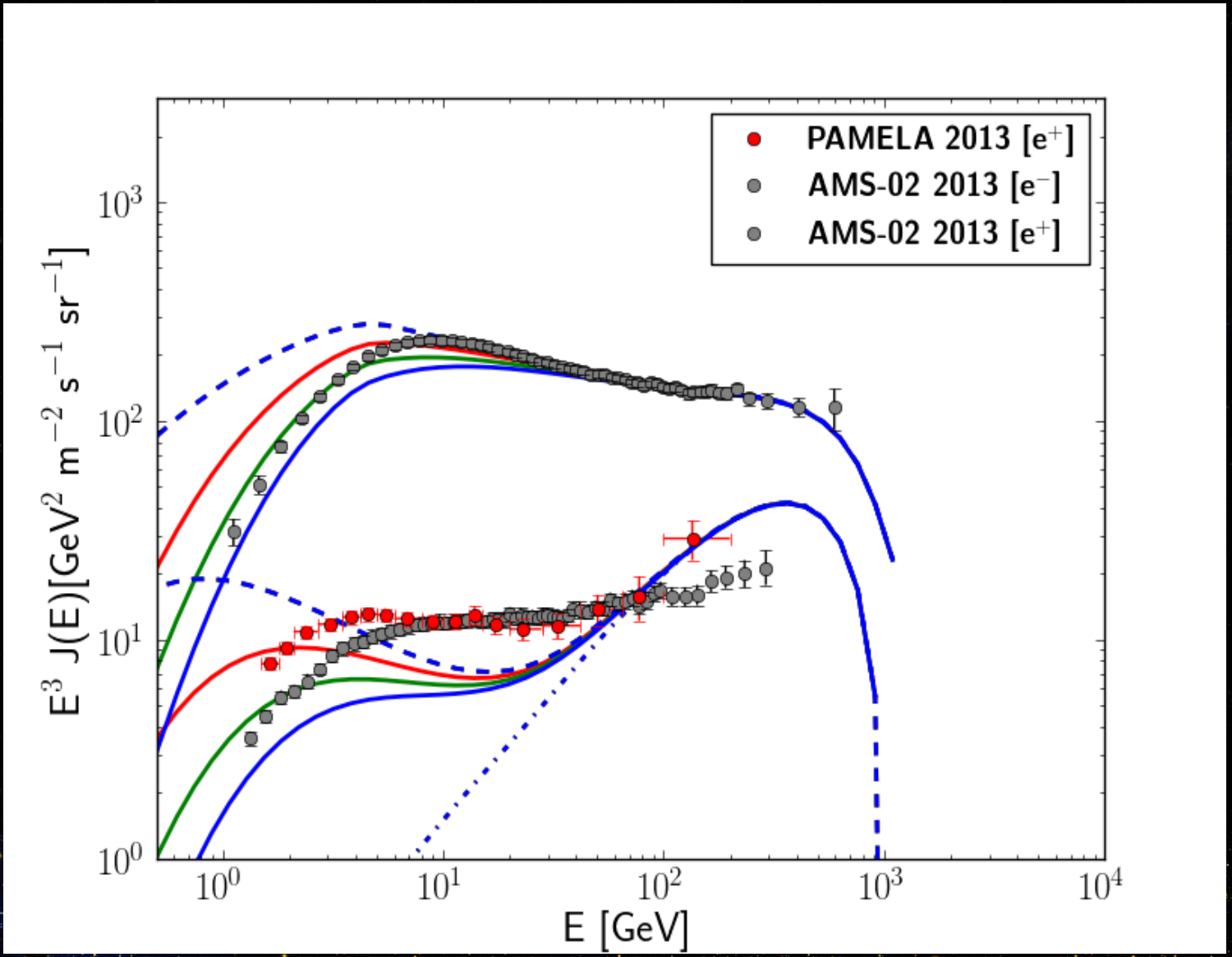
(this will take a couple of minutes)

2-> edit the python routine `plots/plot_lepton_fluxes.py` in order to plot the ***leptonic fluxes*** corresponding to this run

- with 3 different modulation potentials: 0.2 - 0.4 – 0.6 GV
- and setting the normalization of the electrons to 1.2 and the normalization of the extra component to 0.0

Second exercise

-> This is what you should get:



Third exercise

Comparing two different propagation models, with Kolmogorov-like and Kraichnan-like diffusion

The runs are `run_2D_KRA.xml` and `run_2D_KOL.xml`

→ The first is a model with moderate reacceleration ($v\_A = 15\ km/s$) and $\delta = 0.5$

```
    <D0_1e28 value="2.7" />
    <DiffRefRig value = "4" />
    <Delta value="0.5" />
    <zt value="4" />
    <etaT value="-0.4" />
...
    <vA_kms value="15." />
```

→ The second is similar to the so-called "conventional" model in `Galprop`-based papers; it features a high level of reacceleration ($v\_A = 30\ km/s$) and $\delta = 0.33$

```
    <D0_1e28 value="2.7" />
    <DiffRefRig value = "4" />
    <Delta value="0.33" />
    <zt value="4" />
    <etaT value="-1" />
...
    <vA_kms value="30." />
```

These runs will take longer! Since there is a non negligible reacceleration, the number of iteration per timestep is larger – we set 90 instead of 30 as in plain diffusion runs, so the results, in particular those regarding the leptonic fluxes, are more reliable ***(thanks to KIT group for pointing this out)***

Third exercise

Comparing two different propagation models, with Kolmogorov-like and Kraichnan-like diffusion

The runs are `run_2D_KRA.xml` and `run_2D_KOL.xml`

1-> run the code

```
$ ./DRAGON examples/run_2D_KRA.xml && ./DRAGON
examples/run_2D_KOL.xml
```

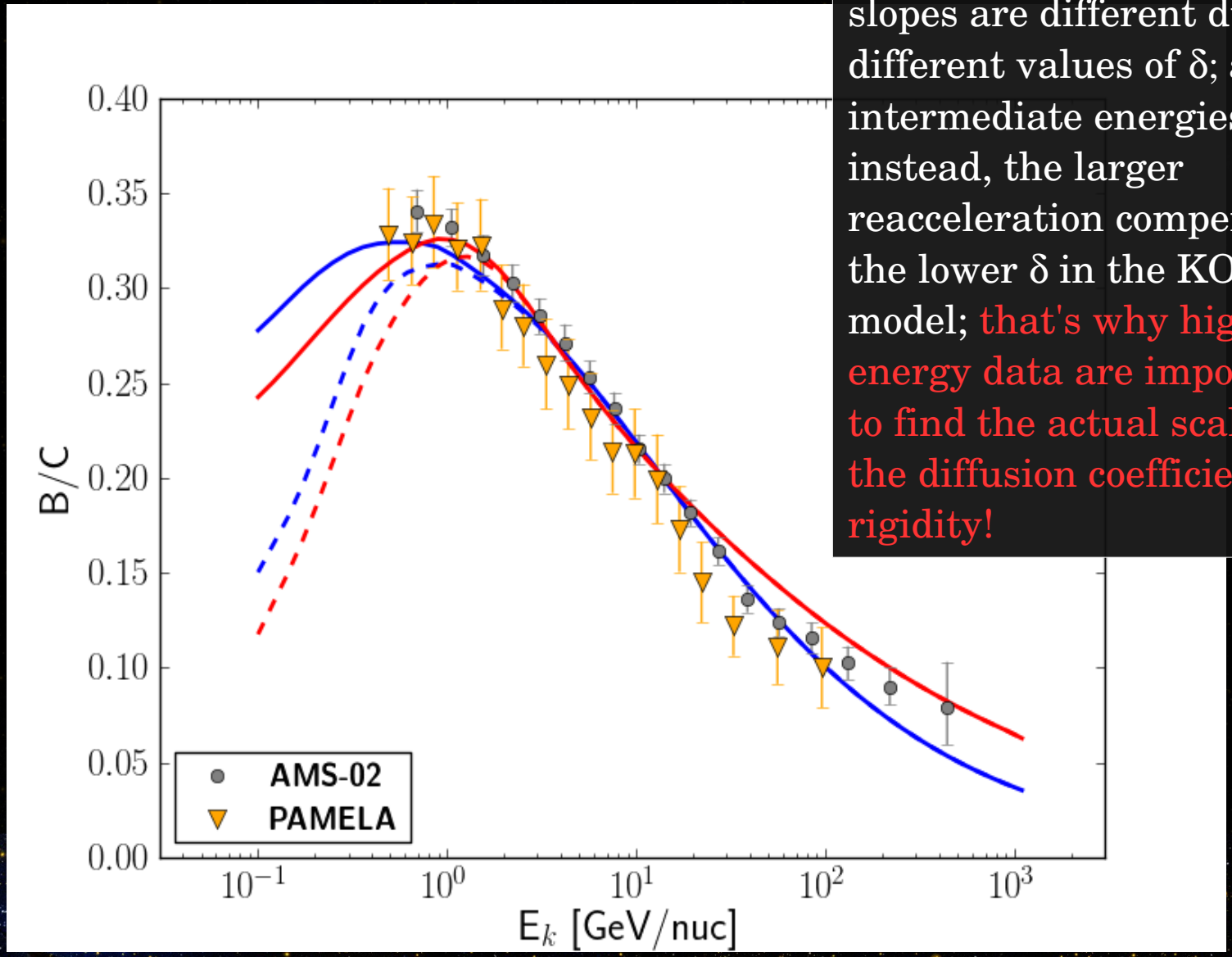
(this will take several minutes)

2-> edit the python routine `plots/plot_protons.py` in order to plot the ***proton spectrum*** corresponding to these runs trying to find the correct modulation potential

3-> edit the python routine `plots/plot_BC.py` in order to plot the **B/C** corresponding to these runs with the following modulation potentials:
0.4 – 0.6 GV

Third exercise

-> This is what you should get for the B/C:



Notice that the high energy slopes are different due to different values of δ; at intermediate energies, instead, the larger reacceleration compensates the lower δ in the KOL model; that's why high energy data are important to find the actual scaling of the diffusion coefficient with rigidity!

A few words about 3D

It's easy to run DRAGON in 3D mode!

The code runs on a Cartesian (x,y,z) grid; it's possible to have a non-equidistant binning (we acknowledge the Karlsruhe group for this feature) so regions where structures are present can be resolved more accurately

In the examples/ folder you can find run_3D.xml which is a good starting point.

The code allows the user to implement a spiral arm structure, whose parameters are specified in the XML. This spiral arm structure may be applied to any distribution:
source term Q(z,y,z), gas density, interstellar radiation field (ISRF) ecc.

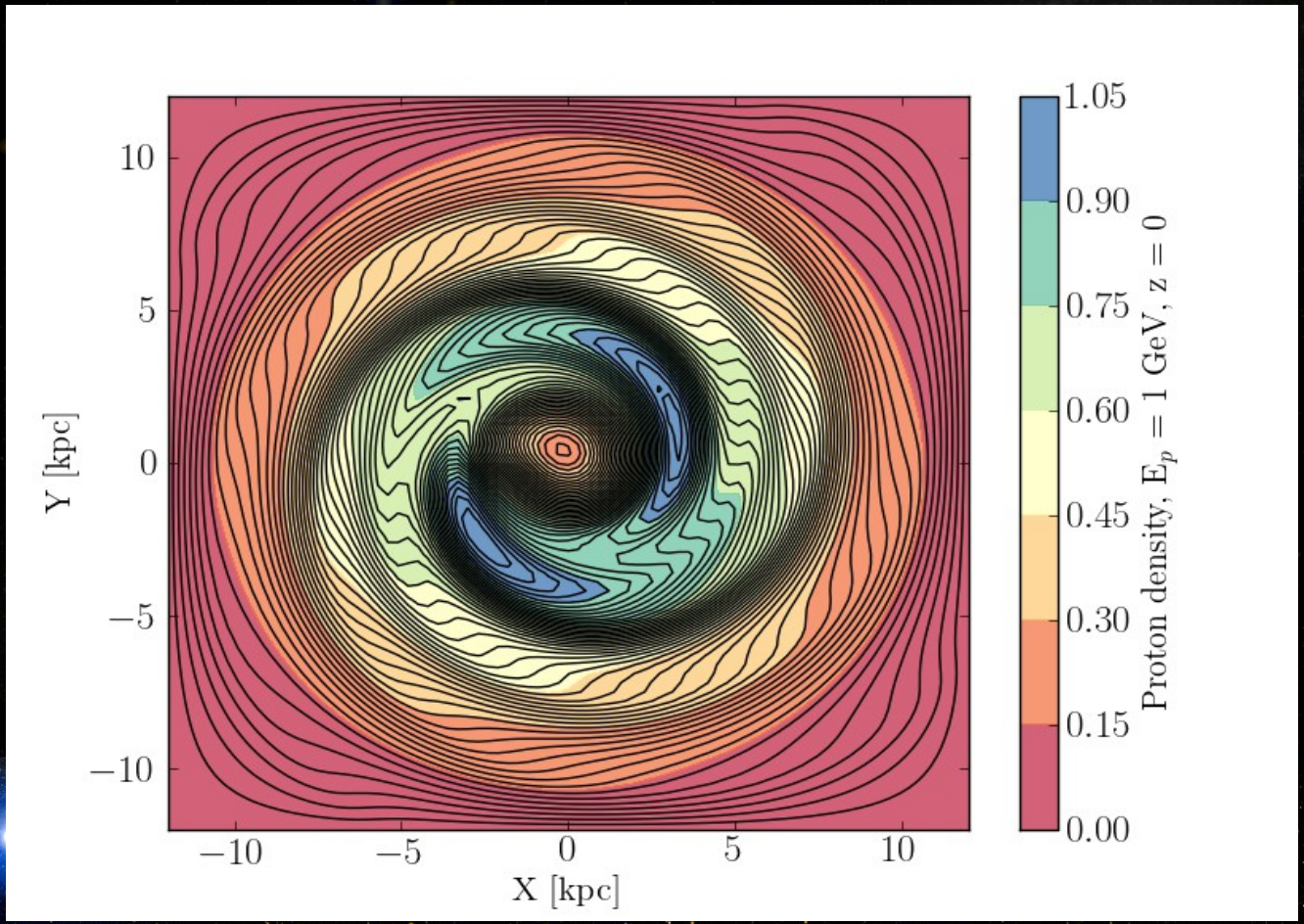
In this example, we apply the spiral arm distribution and see the effect on the propagated leptons.

Since the leptons, especially at high energy, suffer severe energy losses due to inverse Compton and Synchrotron emission, they follow more closely the source term, and the presence of a spiral pattern in this term is evident in the propagated fluxes.

A few words about 3D

The routine `plot_contours_3D.py` allows to visualize face-on maps of the propagated fluxes.

Using this routine on the standard 3D run you should get this result.



Summary

– we learned how to install and run DRAGON

– we learned how to prepare a meaningful parameter file

– we have a set of exercises that may be useful to get started with the code:

0) plotting the proton spectrum for a simple 2D run

1) plotting all the relevant species in a standard 2D run, playing with normalization and tuning the modulation potential

2) plotting a DM run

3) comparing different propagation models

4) running the code in 3D mode and plotting a face-on map of the Galaxy