The JETSCAPE Framework

Hands-on
Part I

Tianyu Dai
Duke University

RHIC & AGS Annual User's meeting
June 8th 2021

Outline

June 8th:

- Overview of JETSCAPE framework
- Run JETSCAPE using container
- Configure JETSCAPE
- Perform parton evolution in static medium
- Implement a new module in JETSCAPE*

June 9th:

- JETSCAPE for heavy ion collisions
- Configure JETSCAPE to use realistic hydrodynamic module
- Run p+p collisions
- Extend to A+A collisions and calculate RAA

Questions

Interact on Slack:

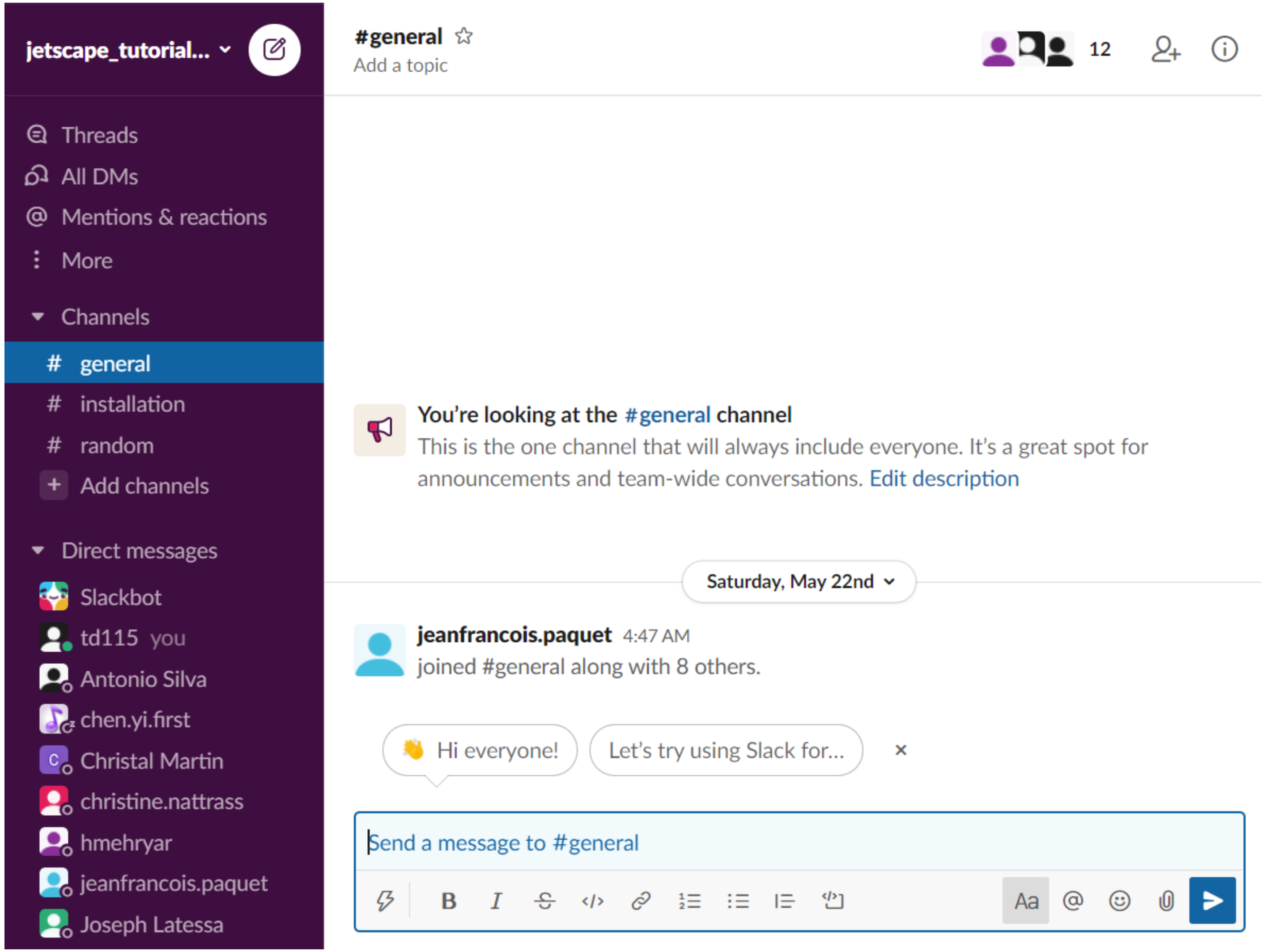
jetscape_tutorial_rhic_ags_2021

- Ask questions : # general
- Questions about installation:

  # installation
- School material: # school_material

Wiki page:

https://github.com/TianyuDai/JETSCAPE-rhic-ags/wiki



Questions

Interact on Slack:

jetscape_tutorial_rhic_ags_2021

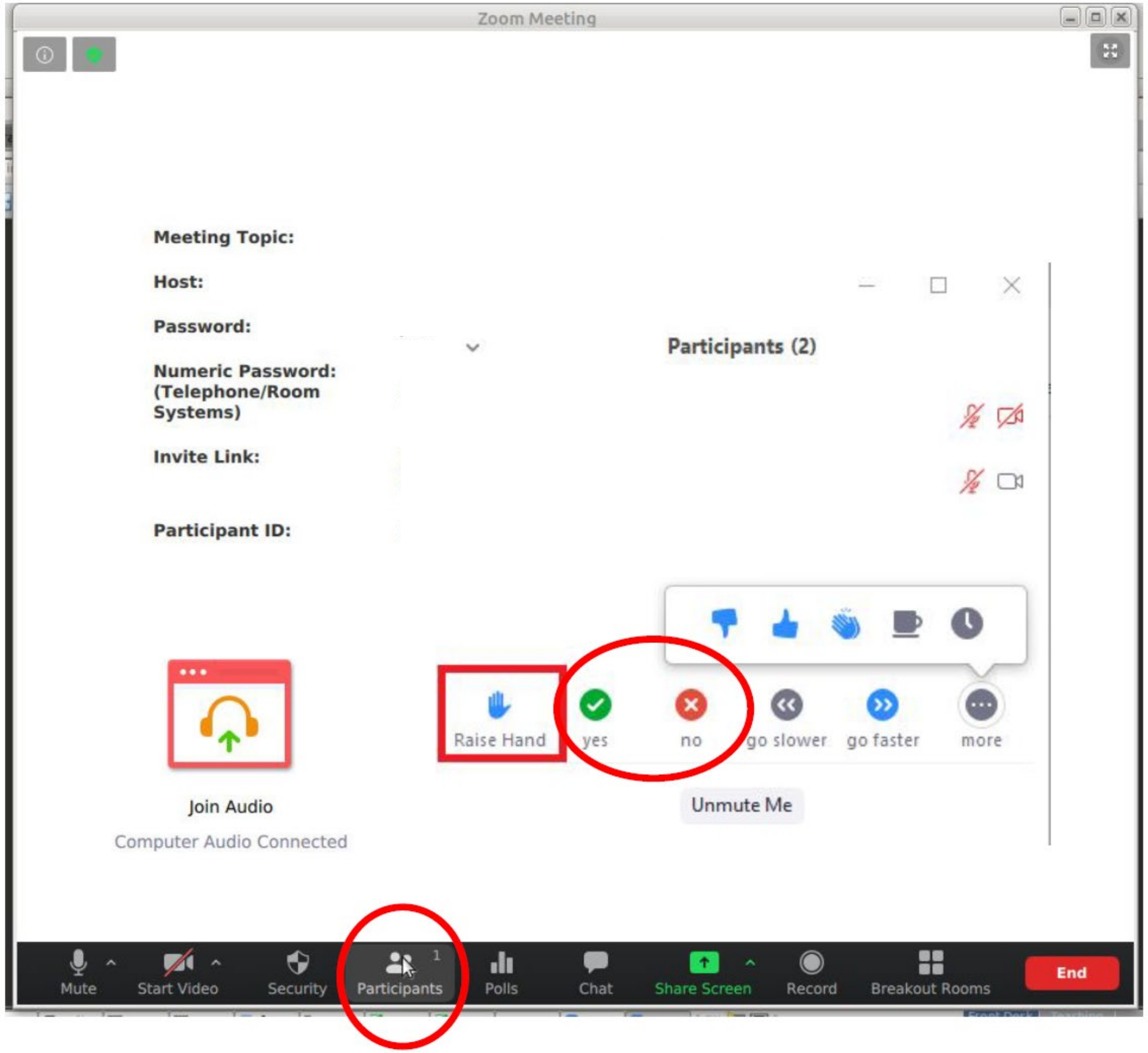
- Ask questions : # general
- Questions about installation:

  # installation
- School material: # school_material

Wiki page:

https://github.com/TianyuDai/JETSCAPE-rhic-ags/wiki

React on Zoom:



Steps to run JETSCAPE

1. Install and build JETSCAPE using container: Singularity / Docker
   One line of code to create environment with all software pre-requisites

Run only in Linux
Not require root access

https://github.com/TianyuDai/JETSCAPE-rhic-ags/wiki/Installing-JETSCAPE

Install JETSCAPE in a Singularity Container

Create a Singularity container of JETSCAPE built from the Docker container available on Docker Hub. Note that v1.4 does not refer to the version of JETSCAPE. The versioning on Docker Hub is different. Using the --sandbox flag creates the JETSCAPE Singularity container as a writable directory. The directory name chosen here is "jetbox". You can create any directory name you prefer.

This command creates the Singularity container with the required software prerequisites to run JETSCAPE. **Only run this command in a directory that has a couple Gb of disk space, since this directory will be used for everything that follows.**

```
singularity build --sandbox jetbox docker://jetscape/base:v1.4
```

Shell into the newly built container.

```
singularity shell --writable jetbox
```

This last command essentially created a virtual machine where all the libraries required are available to compile and run JETSCAPE.

Run in Linux, Windows, Mac
Require root access

https://github.com/JETSCAPE/JETSCAPE/tree/master/docker

Using JETSCAPE via Docker

Docker is a software tool that allows one to deploy an application in a portable environment. A docker "image" can be created for the application, allowing any user to run a docker "container" from this image. We have prepared a docker image for the JETSCAPE environment, which allows you to use JETSCAPE on macOS or linux without installing a long list of pre-reqs or worrying about interference with software you already have installed. Step-by-step instructions are provided below.

Create and start a docker container that contains all of the JETSCAPE pre-reqs:

macOS:

```
docker run -it -v ~/jetscape-docker:/home/jetscape-user --name myJetscape jetscape/base:v1
```

linux:

```
docker run -it -v ~/jetscape-docker:/home/jetscape-user --name myJetscape --user $(id -u):$(id -g)
```

Steps to run JETSCAPE

Instructions for this step: Find in Github wiki page!
https://github.com/TianyuDai/JETSCAPE-rhic-ags/wiki/Installing-JETSCAPE

Load Singularity/Docker environment

Inside container:
Building and running JETSCAPE

```
tianyu@tianyu-XPS-13-9360:~/jetscape-docker$ docker start -ai myJetscape
I have no name!@2a53f2746d21:~$ ls
JETSCAPE
I have no name!@2a53f2746d21:~$ cd JETSCAPE/build/
I have no name!@2a53f2746d21:~/JETSCAPE/build$
```

Outside container:
Editing text files

```
tianyu@tianyu-XPS-13-9360:~/jetscape-docker$ ls
JETSCAPE
tianyu@tianyu-XPS-13-9360:~/jetscape-docker$ cd JETSCAPE/build/
tianyu@tianyu-XPS-13-9360:~/jetscape-docker/JETSCAPE/build$
```

Tip:

Keep two terminals open: One inside the container, and one outside

When building/running JETSCAPE — **inside container**
When editing text files — **outside container**

Steps to run JETSCAPE

2. Download and build JETSCAPE

   - Create a directory called "rhic_ags_school" and CD into it.
     We assume /rhic-ags-school as the working directory during the school.

```
mkdir /rhic-ags-school
cd /rhic-ags-school
```

   - Dowload JETSCAPE

```
git clone https://github.com/TianyuDai/JETSCAPE-rhic-ags.git
```

   - Enter into the cloned repository. Get external packages

```
cd JETSCAPE-rhic-ags/external_packages
bash get_music.sh
```

   - Create a directory called build. Enter into the build directory and build JETSCAPE.

```
mkdir /rhic-ags-school/ JETSCAPE-rhic-ags/build
cd ../build
cmake –DUSE_MUSIC=ON ..
make –j4
```

Configuring JETSCAPE

JETSCAPE is configured via two XML files

- Master XML file — *you don't modify this*
  Contains default values for every possible module and parameter
- User XML file — *you provide this*
  List of which modules to run, and which default parameter values to override

Configuring JETSCAPE

Master XML file

you don't modify this

A "database" of all possible modules and parameters

All possible basic settings

All possible initial state module parameters

...

```xml
<jetscape>

 <!-- General settings -->
 <nEvents> 100 </nEvents>
 <setReuseHydro> true </setReuseHydro>
 <nReuseHydro> 10 </nReuseHydro>

 <!-- Technical settings -->
 <debug> on </debug>
 <remark> off </remark>
 <vlevel> 0 </vlevel>
 <enableAutomaticTaskListDetermination> true </enableAutomaticTaskListDetermination>

 <!--  JetScape Writer Settings -->
 <outputFilename>test_out</outputFilename>
 <JetScapeWriterAscii> off </JetScapeWriterAscii>
 <JetScapeWriterAsciiGZ> off </JetScapeWriterAsciiGZ>
 <JetScapeWriterHepMC> off </JetScapeWriterHepMC>

 <!--  Random Settings. For now, just a global  seed. -->
 <Random>
   <seed>1</seed>
 </Random>

 <!-- Inital State Module  -->
 <IS>
   <!-- x range [-grid_max_x, grid_max_x] -->
   <!-- y range [-grid_max_y, grid_max_y]-->
   <!-- longitudinal range [-grid_max_z, grid_max_z]-->
   <!-- in units of [fm] -->
   <grid_max_x> 10 </grid_max_x>
   <grid_max_y> 10 </grid_max_y>
   <grid_max_z> 0 </grid_max_z>
   <grid_step_x> 0.2 </grid_step_x>
   <grid_step_y> 0.2 </grid_step_y>
   <grid_step_z> 0.2 </grid_step_z>

   <Trento use_module="pre_defined">
     <!-- pre-defined system: default collisions have auau200, pbpb2760, pbpb5020, more in
        the future -->
     <pre_defined collision_system="auau200" centrality_min="30" centrality_max="40" />
     <!-- user-defined system: to get one event in 0-100% centrality range -->
     <user_defined projectile="Au" target="Au" sqrts="200" cross_section="4.2" />
   </Trento>

   <!-- Options to read initial conditions from saved file  -->
   <initial_profile_path>../examples/test_hydro_files</initial_profile_path>
 </IS>

...
```

Configuring JETSCAPE

User XML file

you provide this

Specify which modules to run
Specify parameter values (otherwise taken from master)

Activate modules (in order)

config/jetscape_user_brick.xml

Set nEvents

Set Writer

Set initial condition

Set medium

Set energy loss module

```xml
<jetscape>

  <nEvents> 10000 </nEvents>

  <!--  JetScape Writer Settings -->
  <outputFilename>brick_out</outputFilename>
  <JetScapeWriterAscii> on </JetScapeWriterAscii>

  <!-- Hard Process -->
  <Hard>
    <PGun>
      <pT>120</pT>
      <parID>21</parID>
    </PGun>
  </Hard>

  <!-- Hydro  Module  -->
  <Hydro>
    <Brick>
      <T>0.3</T>
    </Brick>
  </Hydro>

  <!--Eloss Modules -->
  <Eloss>

    <deltaT>0.01</deltaT>
    <maxT>1</maxT>

    <Martini>
      <alpha_s> 0.3 </alpha_s>
      <pcut> 1.0 </pcut>
    </Martini>

  </Eloss>

</jetscape>
```

Running JETSCAPE

There is one central executable: runJetscape.cc

- Modules are automatically added according to User XML
- You don't ever need to re-compile this executable

Pass your user configuration file as a command line argument:

```
cd /rhic-ags-school/JETSCAPE-rhic-ags/build
./runJetscape ../config/jetscape_user_brick.xml
```

JETSCAPE Output

JETSCAPE output contains:

- Final state hadrons
- Final state partons
- Full parton-shower history

Ascii output: build/brick_out.dat

```
2 Event
# sigmaGen 1
# sigmaErr 0
# weight 1
# HardProcess Parton List: PGun
0 21 0 120 0 0 120 0 0 0 0
# Energy loss Shower Initating Parton: JetEnergyLoss
0 21 0 120 0 0 120 0 0 0 0
# Parton Shower in JetScape format to be used later by GTL graph:
[0] V 0 0 0 0
[1] V 0 0 0 0
[2] V 0 0 0 0.7
[3] V 0 0 0 0.7
[0]=>[1] P 0 21 0 120 0 0 120 0 0 0 0
[1]=>[2] P 3 21 0 118.743 0 0 118.743 0.71 0 0 0.71
[1]=>[3] P 4 21 0 1.27016 0 0 1.27016 0.71 0 0 0.71
```

Event info

Initial parton

Shower-initiating parton

Parton shower history

You can produce JETSCAPE output in two formats:

Ascii

Custom JETSCAPE format

Executables are provided to extract final-state hadrons/partons

Can also directly write gzipped ascii

HepMC3

Standard event format (larger size)

Compatible with Rivet

JETSCAPE Output

Tutotial for this practice: Find in Github wiki page!
https://github.com/TianyuDai/JETSCAPE-rhic-ags/wiki/A-single-parton-propagating-in-a-brick-of-fixed-temperature-and-length

Generate final state partons using Ascii output:

```
cd /rhic-ags-school/JETSCAPE-rhic-ags/build
mkdir ../../JETSCAPE-output
./FinalStatePartons brick_out.dat ../../JETSCAPE-output/my_final_state_partons.txt
```

JETSCAPE-rhic-ags/examples/FinalStatePartons.cc
has two input parameters:

- *brick_out.dat is the filename of the Ascii output file*
- *my_final_state_partons.txt is the file to save the final state parton information*

Set input and output file

```
auto reader=make_shared<JetScapeReaderAscii>(argv[1]);
std::ofstream dist_output (argv[2]); //Format is SN, PID, E, Px, Py, Pz, Eta, Phi
int SN=0, TotalPartons=0;
while (!reader->Finished())
  {

    reader->Next();

    // cout<<"Analyze current event: "<<reader->GetCurrentEvent()<<endl;
    auto mShowers=reader->GetPartonShowers();

    TotalPartons =0;
    for (int i=0;i<mShowers.size();i++)
      {
        TotalPartons = TotalPartons + mShowers[i]->GetFinalPartons().size();
      }
```

Get parton shower information

JETSCAPE Output

Tutotial for this practice: Find in Github wiki page!
https://github.com/TianyuDai/JETSCAPE-rhic-ags/wiki/A-single-parton-propagating-in-a-brick-of-fixed-temperature-and-length

Generate final state partons using Ascii output:

```
cd /rhic-ags-school/JETSCAPE-rhic-ags/build
makedir ../../JETSCAPE-output
./FinalStatePartons brick_out.dat ../../JETSCAPE-output/my_final_state_partons.txt
```

JETSCAPE-rhic-ags/examples/FinalStatePartons has two input parameters:

- *brick_out.dat is the filename of the Ascii output file*
- *my_final_state_partons.txt is the file to save the final state parton information*

```
for (int i=0;i<mShowers.size();i++)
  {
    //cout<<" Analyze parton shower: "<<i<<endl;
    // Let's create a file
    for ( int ipart = 0; ipart< mShowers[i]->GetFinalPartons().size(); ++ipart){
      Parton p = *mShowers[i]->GetFinalPartons().at(ipart);
      //          if(abs(p.pid())!=5) continue;

      dist_output << ipart   << "\t"
                  << p.pid() << "\t"
                  << p.pstat() << "\t"
                  << p.e()   << "\t"
                  << p.px()  << "\t"
                  << p.py()  << "\t"
                  << p.pz()  << "\t"
                  << p.eta()<<  "\t"<< p.phi() << endl;
    }
  }
```

Define output format

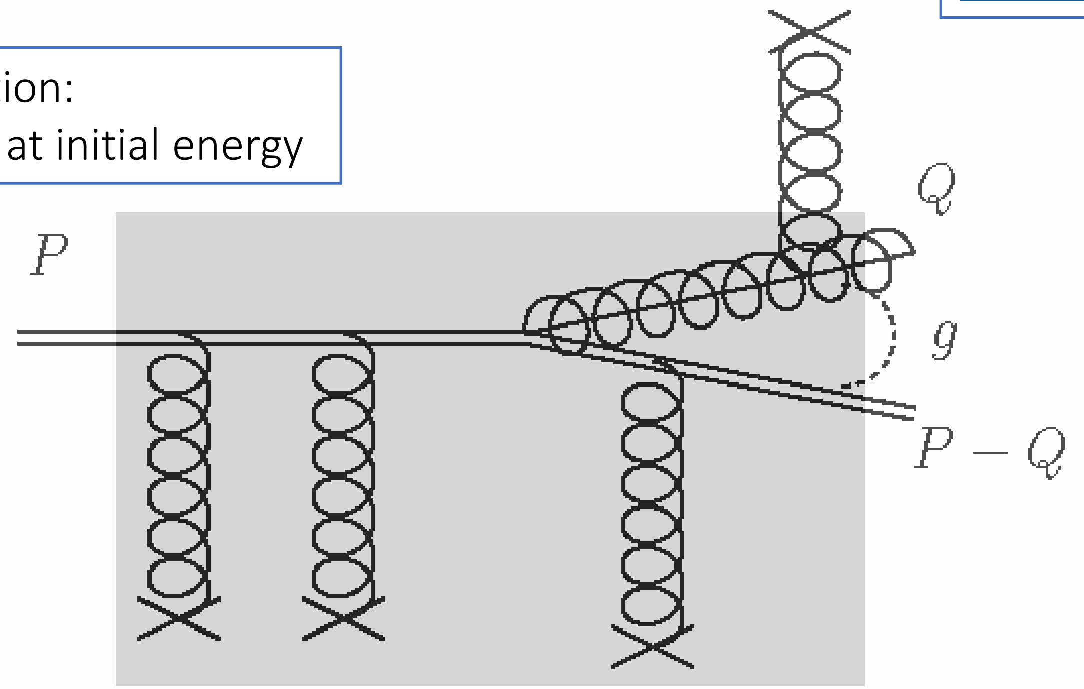
Calculating parton energy distribution

Run the python script to plot the final parton energy distribution:

```
cd /rhic-ags-school/JETSCAPE-rhic-ags/analysis/
python3 plot_energy_distribution.py
```

Tutotial for this practice:
Find in Github wiki page!
https://github.com/TianyuDai/JETSCAPE-rhic-ags/wiki/A-single-parton-propagating-in-a-brick-of-fixed-temperature-and-length

Initial energy distribution:
a Dirac-delta function at initial energy

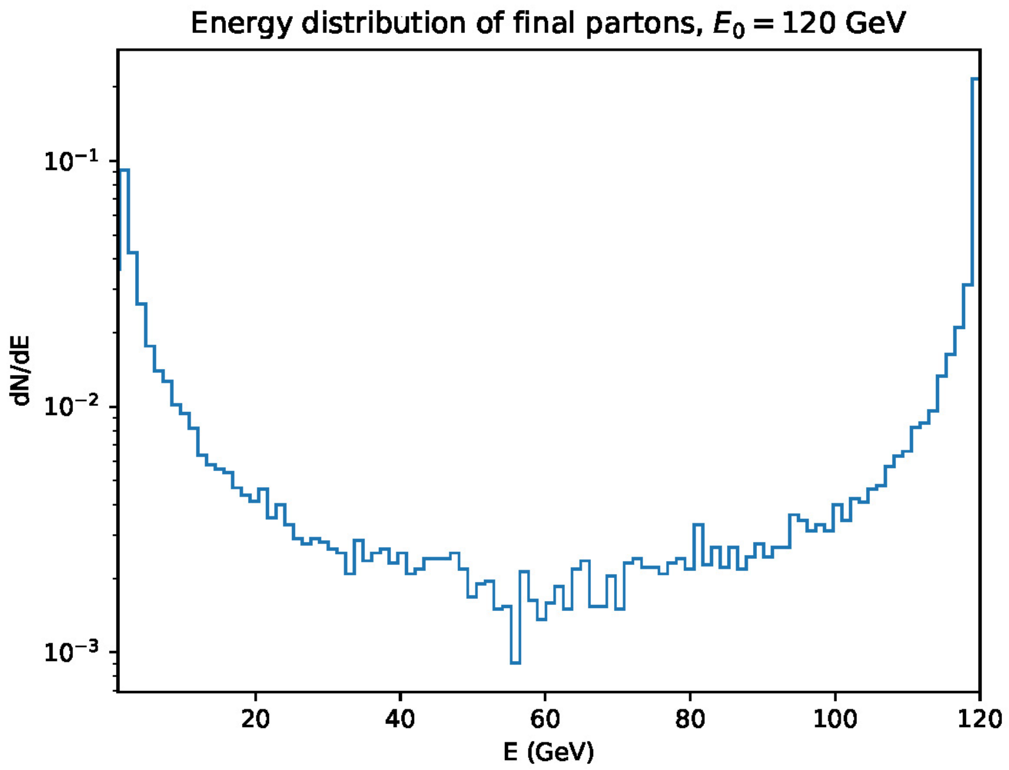


Final energy distribution:
a distribution with a wider range

Calculating parton energy distribution

Run the python script to plot the final parton energy distribution:

```
cd /rhic-ags-school/JETSCAPE-rhic-ags/analysis/
python3 plot_energy_distribution
```



Tutotial for this practice:
Find in Github wiki page!
https://github.com/TianyuDai/JETSCAPE-rhic-ags/wiki/A-single-parton-propagating-in-a-brick-of-fixed-temperature-and-length

Implement a new module

```
#ifndef MYJEL_H
#define MYJEL_H

#include "JetEnergyLossModule.h"

using namespace Jetscape;

class MyJEL : public JetEnergyLossModule<MyJEL>
{
 public:

  MyJEL();
  virtual ~MyJEL();

  void Init();
  void DoEnergyLoss(double deltaT,double time, double Q2,
                    vector<Parton>& pIn, vector<Parton>& pOut);
  void WriteTask(weak_ptr<JetScapeWriter> w);

 private:
  // Allows the registration of the module so that it is available
  // to be used by the Jetscape framework.
  static RegisterJetScapeModule<MyJEL> reg;

};

#endif // MyJEL
```

Implement your physics in these standard functions, which will be called by the framework

Note: Which function(s) to implement depends on the type of the module
For details, see arXiv:1903.07706

Implement a new module

To register a module with the framework, we should do with the following steps:

- Note the following in your CustomModule.h

```
private:
    // Allows the registration of the module so that it is available to be used by the Jetscape framework
    static RegisterJetScapeModule<CustomModule> reg;
```

- Note the following in your CustomModule.cc

Module class name

Module XML name

```
// Register the module with the base class
RegisterJetScapeModule<CustomModule> CustomModule::reg("CustomModule");
```

- Note this part in src/framework/JetScape.cc

```
//    - Custom module
else if (((int)childElementName.find("CustomModule") >= 0)) {
  auto customModule =
      JetScapeModuleFactory::createInstance(childElementName);
  if (customModule) {
    jloss->Add(customModule);
    JSINFO << "JetScape::DetermineTaskList() -- Eloss: Added "
           << childElementName << " to Eloss list.";
  }
}
```

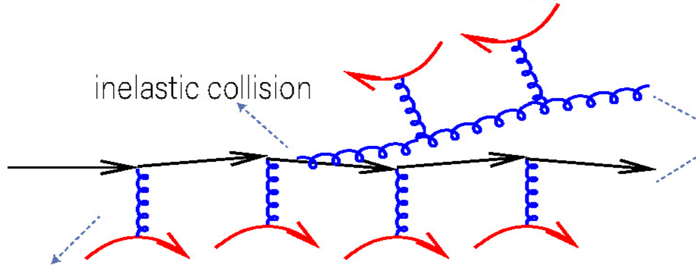
Set your new module's name as CustomModule, or modify /src/framework/JetScape.cc

Implement a new module

Practice: implement **a diffusion process** in the energy loss module

Interactions between the parton and the medium can be described using a Boltzmann equation

$$\left(\frac{\partial}{\partial t}+\frac{\mathbf{p}}{p}\cdot\nabla\right)\delta f_a=-\mathcal{C}[\delta f_a, n_a]$$



Soft and frequent interactions can be treated stochastically as a diffusion process

$$\boxed{C_a^{\mathrm{diff}}[\delta f]\equiv-\frac{\partial}{\partial p^i}\left[\eta_D(p)p^i\delta f^a(\boldsymbol{p})\right]-\frac{1}{2}\frac{\partial^2}{\partial p^i\partial p^j}\left[\left(\hat{p}^i\hat{p}^j\hat{q}_L(p)+\frac{1}{2}(\delta^{ij}-\hat{p}^i\hat{p}^j)\hat{q}(p)\right)\delta f^a(\boldsymbol{p})\right].}$$

Put a new module performing diffusion process in JETSCAPE

use trasnsport parameters given in Ghiglieri, J., Moore, G.D. & Teaney, D., J. High Energ. Phys. ***2016,*** 95 (2016). *in this practice*

Implement a new module

Tutotial for this practice: Find in Github wiki page!
https://github.com/TianyuDai/JETSCAPE-rhic-ags/wiki/Implement-a-new-energy-loss-module-in-JETSCAPE

Practice: implement a diffusion process in the energy loss module

Use analysis/new_modules/CustomModule.cc to perform diffusion process

Initialization:

```
void CustomModule::Init()
{
        JSINFO<<"Intialize CustomModule ...";

        double deltaT = 0.0;
        deltaT = GetXMLElementDouble({"Eloss", "deltaT"});

        string s = GetXMLElementText({"Eloss", "CustomModule", "name"});
        JSDEBUG << s << " to be initilizied ...";

        alpha_s = 0.3;
        alpha_s = GetXMLElementDouble({"Eloss", "CustomModule", "alpha_s"});

        pcut = 1.;
        pcut = GetXMLElementDouble({"Eloss", "CustomModule", "pcut"});

        mu = 1.;
        mu = GetXMLElementDouble({"Eloss", "CustomModule", "cutoff"});

        g = sqrt(4.*M_PI*alpha_s);
        std::normal_distribution<double> white_noise(0.,1.);
}
```

Read information from hydro module:

```
std::unique_ptr<FluidCellInfo> check_fluid_info_ptr;
GetHydroCellSignal(Time, xx, yy, zz, check_fluid_info_ptr);
VERBOSE(8)<< MAGENTA<<"Temperature from Brick (Signal) = "
         <<check_fluid_info_ptr->temperature;

vx = check_fluid_info_ptr->vx;
vy = check_fluid_info_ptr->vy;
vz = check_fluid_info_ptr->vz;
T = check_fluid_info_ptr->temperature;
```

Add particles after energy loss back to parton list:

```
if (pVecNew.t() > pcut)
{
    pOut.push_back(Parton(0, Id, 0, pVecNew, xVec));
    pOut[pOut.size()-1].set_form_time(0.);
}
return;
```

Do energy loss:

```
void CustomModule::DoEnergyLoss(double deltaT, double Time, double Q2, vector<Parton>& pIn, vector<Parton>& pOut)
```

Implement a new module

Tutotial for this practice: Find in Github wiki page!
https://github.com/TianyuDai/JETSCAPE-rhic-ags/wiki/Implement-a-new-energy-loss-module-in-JETSCAPE

Practice: implement a diffusion process in the energy loss module

- Find CustomModule.cc, CustomModule.h, Lorenz.h in analysis/new_modules, and copy them to src/jet/

```
cp /rhic-ags-school/JETSCAPE-rhic-ags/analysis/* ../src/jet
```

- Re-compile JETSCAPE

```
cd  /rhic-ags-school/JETSCAPE-rhic-ags/build
cmake –DUSE_MUSIC=ON ..
make -j
```

- Create a new XML file, config/jetscape_user_custom.xml, add the new module into it

```
<!--Eloss Modules -->
<Eloss>

  <deltaT>0.01</deltaT>
  <maxT>1</maxT>

  <CustomModule>
    <name> CustomModule </name>
    <alpha_s> 0.3 </alpha_s>
    <pcut> 1.0 </pcut>
    <cutoff> 1.0 </cutoff>
  </CustomModule>

</Eloss>
```

- Run JETSCAPE with the new energy loss module

```
./runJetscape ../config/jetscape_user_custom.xml
```

- Run FinalStatePartons to process Ascii output file, and plot the final parton energy distribution

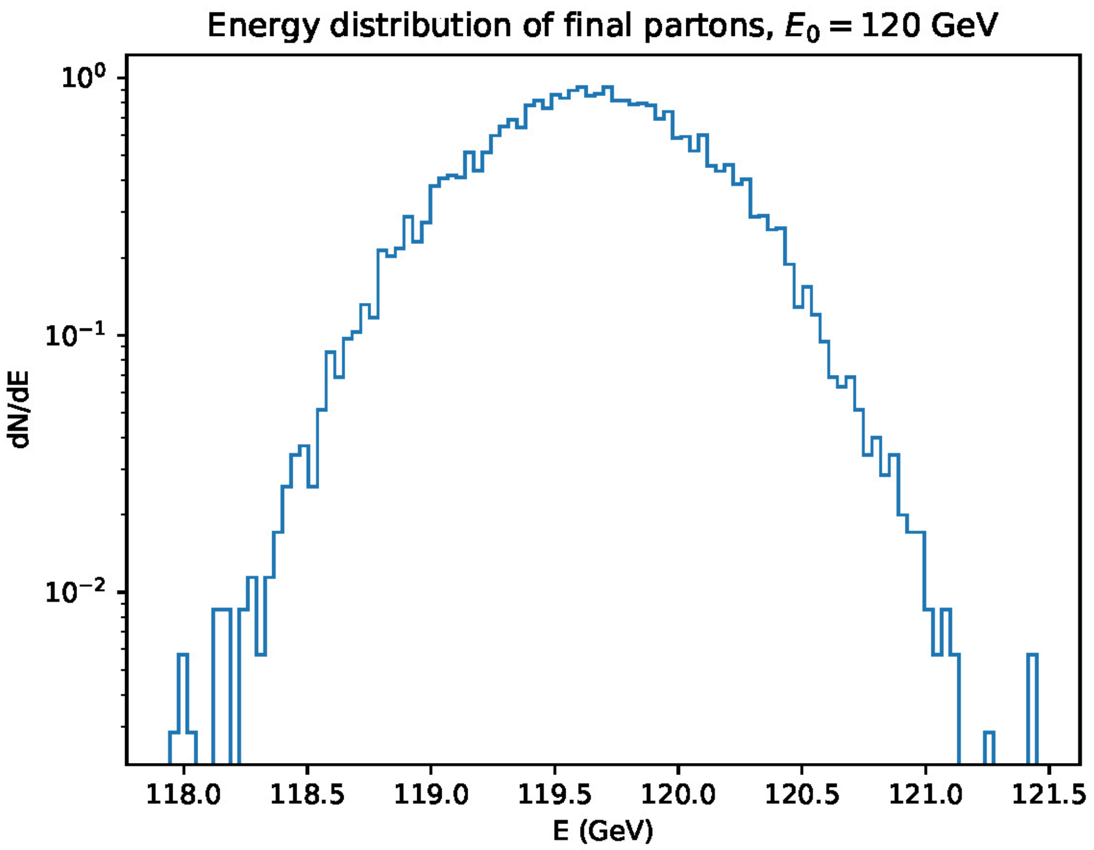
```
./FinalStatePartons custom_brick_out.dat ../../JETSCAPE-output/custom_final_state_partons.txt
cd ../analysis
python3 plot_energy_distribution.py
```

Remember to modify the output file name

Implement a new module

Practice: implement a diffusion process in the energy loss module

Energy distribution of final partons using the new added custom energy loss module



Tutotial for this practice:
Find in Github wiki page!
https://github.com/TianyuDai/JETSCAPE-rhic-ags/wiki/Implement-a-new-energy-loss-module-in-JETSCAPE