NUPYCEE

The NuGrid collaboration

http://forum.astro.keele.ac.uk:8080/nugrid

Received ______________________; accepted ______________________

Document name: NUPYCEE

SVN directory: svn://forum.astro.keele.ac.uk/utils/GCE/NUPYCEE

Contributors: Christian Ritter, Benoit Côté

Please send any feedback to critter@uvic.ca or bcote@uvic.ca. Thank you.

Abstract: The Nugrid-Python-Chemical-evolution environment, short NuPyCEE, provides a framework to simulate the ejecta of simple stellar populations with the 1-zone code Stellar Yields for Galactic Modeling Applications as well as galaxies with the One-zone Model for the Evolution of Galaxies. Various chemical evolution input parameter can be specified through a python-based interface. Both tools provide extraction and plotting methods. We also provide a web interface with limited capabilities which allows to download yield tables containing the ejecta of SSP's. To compare the results to observational data the user can utilize the STELLAB module which provides various spectroscopic abundance ratios of stars such as Milky Way and dwarf galaxies.

1. Introduction

Welcome to the NUPYCEE user Guide. The purpose of NUPYCEE is to provide the user with a framework to perform chemical evolution calculations. The focus lies on simplification and user-friendliness. The user can choose between the *Stellar Yields for Galactic Modeling Applications* (SYGMA) module and the *One-zone Model for the Evolution of Galaxies* (OMEGA) module. SYGMA allows to follow the ejecta of simple stellar populations in a closed box while OMEGA deals with the simulation of galaxies with inflows and outflows. Both tools can be used in a similar manner through the ipython command line or ipython notebooks.

1.1. Fastest possible instruction

To download the code you need to have git installed on your local machine. Then download the code from github at

```
https://github.com/NuGrid/NuPyCEE
```

into your local directory via

```
git clone https://nupycee@bitbucket.org/nupycee/nupycee.bitbucket.org.git .
```

Install ipython with numpy and matplotlib and nugridpy via

```
pip install nugridpy
```

Start ipython in the NUPYCEE directory with

```
ipython -pylab -p numpy
```

To run a SYGMA calculation:

```
import sygma as s
s1=s.sygma()
```

That's it! To plot the total mass ejected:

```
s1.plot_totmasses(source='all')
```

To run OMEGA:

```
import omega as o
o1=o.omega()
```

```
o1.plot_spectro(xaxis='[Fe/H]',yaxis='[O/Fe]')
```

Now you might want to compare the prediction with observations through the STELLAB module.

```
import stellab as st

st1=st.stellab()

st1.plot_spectro(xaxis='[Fe/H]', yaxis='[O/Fe]')
```

1.2. Preparing the environment

To take advantage of the interactive ipython environment SYGMA and OMEGA are both launched after starting ipython with

```
ipython -pylab -p numpy
```

where the modules numpy and matploblib have to be installed. Also you might need to install other python modules (e.g., h5py).

A necessary dependence is NuGrid's pylib toolbox nugridpy which contains a number of python scripts. It can simply be installed via pip:

```
pip install nugridpy
```

1.3. SYGMA

1.3.1. General concepts

In order to follow the chemical enrichment via gas particles in N-body simulations one has to describe the ejecta of material from such star particles, essentially simple stellar populations. With SYGMA one can follow the ejecta of such populations via diagrams and you can extract the composition of the ejecta via tables.

1.3.2. Getting started

If one wants to start SYGMA from outside the SYGMA directory one has to specify the *SYGMADIR* variable which points to the SYGMA dir.

Import the module:

```
import sygma as s
```

Next, we create an SSP instance, initiate a class instance $s1$ of SYGMA:

```
s1=sygma.sygma(iniZ=0.0001,tend=1e10,mgal=1e9)
```

A single stellar population of mass $1 \times 10^9\,\mathrm{M}_\odot$ and metal fraction of 0.0001 is created and evolved in time steps of 10^8 years to the final time of 1e10 years. Other not specified and hence default parameter include for example the choice of the initial-mass function.

One can check the definition of these input parameter online via `http://nugrid.github.io/NuPyCEE/SPHINX/build/html/sygma.html`.

No yield tables are defined and hence the NuGrid tables as the default choice were selected. These are the Set1 and Set1extension yields which are AGB and massive star yields for five different metallicities. In this mode the user specifies *iniZ* which can be either $Z = 2e-2, 1e-2, 6e-3, 1e-3, 1e-4$ and 0.0. Scaled abundances for Z=1e-t65, 1e-6 will be available soon. Currently for $Z = 0$ the

PopIII stars from **?** are set as default.

1.3.3. Providing your own yield tables

Yield tables are available in the NUPYCEE subdirectory yield_tables. Add your yield tables to this directory and SYGMA will be able read the table if you have specified the *table* variable. Only for table of Z=0 the variable *pop3_table* is used. Both tables need yields specified in the SYGMA (and OMEGA) yield input format. See for the structure the default table. It is important to provide an initial abundance file which has to match the number of species provided in the yield tables. Provide the file in the iniAbu directory inside the directory yield_tables. The input variable with which the table file can be specified is *iniabu_table*. For the necessary structure see again the default choice of that variable.

1.3.4. What do I get out?

The output during execution shows some initial parameters, e.g. the isotopes being part of chemical evolution, as well as the time and metallicity evolution. Also shown is the contribution time of stars of certain initial mass.

To analyze the run afterwards we provide a variety of plotting functions which are described in detail at `http://nugrid.github.io/NuPyCEE/SPHINX/build/html/sygma.html`. In following we give some examples. The function plot_mass() shows the evolution of the yield in solar masses.

```
s1.plot_mass()
```

With SYGMA you can look at different contributions for AGB stars, massive stars, and SN Ia.

```
s1.plot_mass(source='agb')
s1.plot_mass(source='massive')
s1.plot_mass(source='sn1a')
```

Note that those lines appear all in one figure.

The function *plot_mass_range_contributions*() shows the yield contribution of stars as a function of their initial mass. It allows to identify which mass range contributes to certain elements. To monitor the evolution of the mass of the gas reservoir ejected masses over time, use plot_totmasses(). The star formation rate as a function of time can be plotted with the function *plot_sfr*(). In the case you want to write out detailed chemical evolution tables of isotopes, use the function *write_evol_table*(). It writes out the mass of isotopes ejected by stars as a function of time (each line represents a timestep).

1.4. OMEGA

OMEGA is a classical one-zone galaxy model with an input star formation history where stars form and inject new elements within the same gas reservoir, using SYGMA to create a SSP at every timestep. A complete description of OMEGA's input parameters and functions can be found in the Sphinx documentation.

OMEGA can mimic known local galaxies, such as Sculptor and Fornax, by using their specific star formation history and current mass, which is taken from the literature. The code offers three different prescriptions for treating gas inflows and outflows (see the OMEGA Userguide ipython notebook).

All the parameters associated with simple stellar populations are treated as in the SYGMA module.

1.5. Getting Started with OMEGA

Steps what to do with omega.

inflows and outflow implementation; star formation input ...

1.6. Stellab

Cool stuff about stellab here.

2. Disclaimer

3. History

This document history complements the svn log.

Authors	yymmdd	Comment
CR	130926	Update of links and details
CR	160513	creation

3.1. Contact

If any bugs do appear or if there are any questions, please email critter@uvic.ca