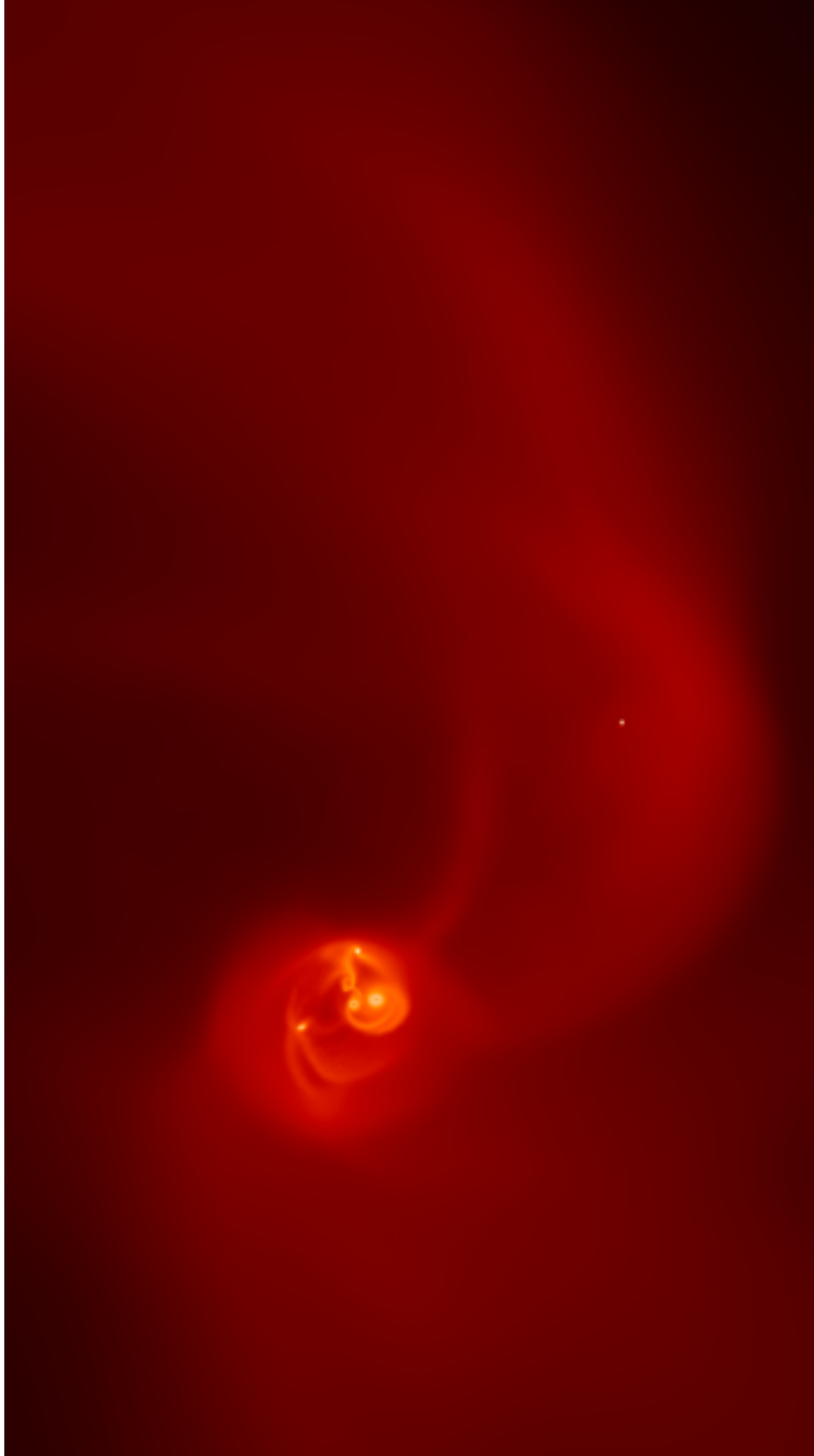
Initial conditions in Python

Giovanni Rosotti

IoA Cambridge

29th October 2015



Numpy

- Most of you said you are familiar in python
- Numpy is a library providing array support for python
- Lot of functions for operating on arrays
- Only recommendations is: never use direct loops in python as they are horrendously slow. Always use a numpy function
- If you are not familiar with numpy, unfortunately we don't have time at the moment. It's plenty of tutorials online. For example: http://cs231n.github.io/python-numpy-tutorial/#numpy

Why use python for IC?

- Python is slower than C++. Writing GANDALF in python wouldn't have been a good idea
- But initial condition generation and analysis takes much less time than running a simulation! So it's ok to pay some price
- On the other hand, it is much easier and faster to code in python
- The numpy/scipy stack has a lot of pre-made tools - no need to reinvent the wheel
- Most of you already know python - no need to learn c++ if you don't need to change the code
- Real life scenario is that you use python to generate IC, save them in a file and run the simulation from the executable on a supercomputer

The basic skeleton

```
from seren.analysis.facade import *

paramfile='testimport.dat' #change this to your needs
sim=newsim(paramfile)

#you can still change some parameters by calling SetParam if you want

#before uploading the initial condition, you need to call PreSetupForPython
#if you forget, you will get an error
#After calling this function, you can no longer change the parameters
#(if you try, you will get an error)
sim.PreSetupForPython()


#--------------------------------
# DO YOUR INITIALIZATION HERE

# To import an array, you have to do like that:
# sim.ImportArray(array, string),
# where array is a numpy array and string defines the quantity
# that you are importing (e.g., 'x' or 'vy').
# At minimum, you need to import the coordinates, the mass arrays
# and the internal energies
# (quantities not imported are set to zero)
#--------------------------------

#Once you are finished, call setup
#If you forget, run will do it for you, but you can't
#do plots before calling run (if you try, you will get
#an error)
setupsim()

#You can now do plots, that will be updated as the simulation runs
plot('x','y')


#Now you can call run
run()
#Block does not exit when the script ends
block()
```

GANDALF-Python interface

- from gandalf.analysis.facade import *
- sim=newsim(2,type='sph') -> ndim and sph are the only parameters that you can't change afterwards. All the parameters are initialised to their default values
- sim=newsim('test.dat') -> alternatively you can also start from an existing parameter file
- sim.SetParam('ic','python') -> tells GANDALF that the C++ part does not need to initialise the particle arrays
- sim.SetParam('foo','bar') -> you can change in this way all the parameters that you would normally set in the parameter file
- sim.PreSetupForPython() -> allocate memory etc. before the actual initialisation. After this call you CANNOT change the parameters anymore
- sim.ImportArray(name, array) -> actual initialisation
- sim.SetupSimulation()
- sim.run()

A simple example

file example08.py

```
#=============================================================================
from gandalf.analysis.facade import *
import numpy as np
import time

# Set basic parameters for generating initial conditions
Nhydro = 200
vfluid = 4.0
xmin = -1.5
xmax = 1.5

# Set uniform line of Nhydro particles between the limits of xmin and xmax
# in local numpy arrays
deltax = (xmax - xmin) / Nhydro
x = np.linspace(xmin + 0.5*deltax,xmax - 0.5*deltax,num=Nhydro)
m = np.ones(Nhydro)*(xmax - xmin)/Nhydro

# Set velocities of shock-tube so v = vfluid for x < 0 and -vfluid for x > 0
vx = np.ones(Nhydro)*vfluid
vx[x > 0.0] = -vfluid
```

…continued

```
(other stuff)

# Call setup routines and import particle data
sim.PreSetupForPython()
sim.ImportArray(x,'x')
sim.ImportArray(vx,'vx')
sim.ImportArray(m,'m')
sim.SetupSimulation()
```

Save the file

- If you don't want to run the simulation in python, just save the output in a file
- Call the function

```
sim.WriteSnapshotFile(name,format)
```

- You don't need to have called SetupSimulation() to do it
- If then you want to use the file as initial condition, use the following parameters:

```
:   ic=file
:   in_file=name
:   in_file_form=format
```

- …and of course all the other ones you need!!!!
- For example you might generate the IC on your laptop and then run the simulation on a super-computer