GAMERA

PyGamma Workshop
MPIK Heidelberg, 16.11.15

Joachim Hahn

Max-Planck-Institut für Kernphysik

What is GAMERA?

- A Japanese Kaiju
- Now also a toolbox for modeling in relativistic astrophysics that aims for a user-friendly
  - calculation of radiation spectra from underlying particle distribution
  - calculation of particle spectrum time evolution in the presence of losses and escape
  - creation of population syntheses
  - calculation of source dynamics
  - ...

GAMERA structure

- Written in **`C++`**, with **`GSL`** as dependency
- Four classes
  - **`Particles:`** calculates time evolution of particle distributions
  - **`Radiation:`** calculates the SED from particle distributions
  - **`Astro:`** collection of astrophysical models (spiral arms, Galactic gas distributions, dynamical models etc.)
  - **`Utils:`** utility functions (random number distributions, integration functions etc.)
- Object-orientation allows for easy multi-zone modeling
- Communication between objects via **`Get`**/**`Set`** functions

Particles class

solves the following equation

$$\frac{\partial N}{\partial t} = \frac{\partial}{\partial \gamma}(PN) - \frac{N}{\tau} + Q$$

N = spectral particle density
γ = particle lorentz factor
P = -(dγ/dt) is the energy loss rate
τ = 'catastrophic' loss time-scale
(e.g. escape)
Q = source term

- P constant: a semi-analytical solver is called
- P = P(t): numerical solution of the problem

- Time-depended spectral modeling with arbitrary parameter and energy loss evolution, particle escape
- No spatial dependence in equation
  $\rightarrow$ no spatial modeling with this class for now

Radiation class

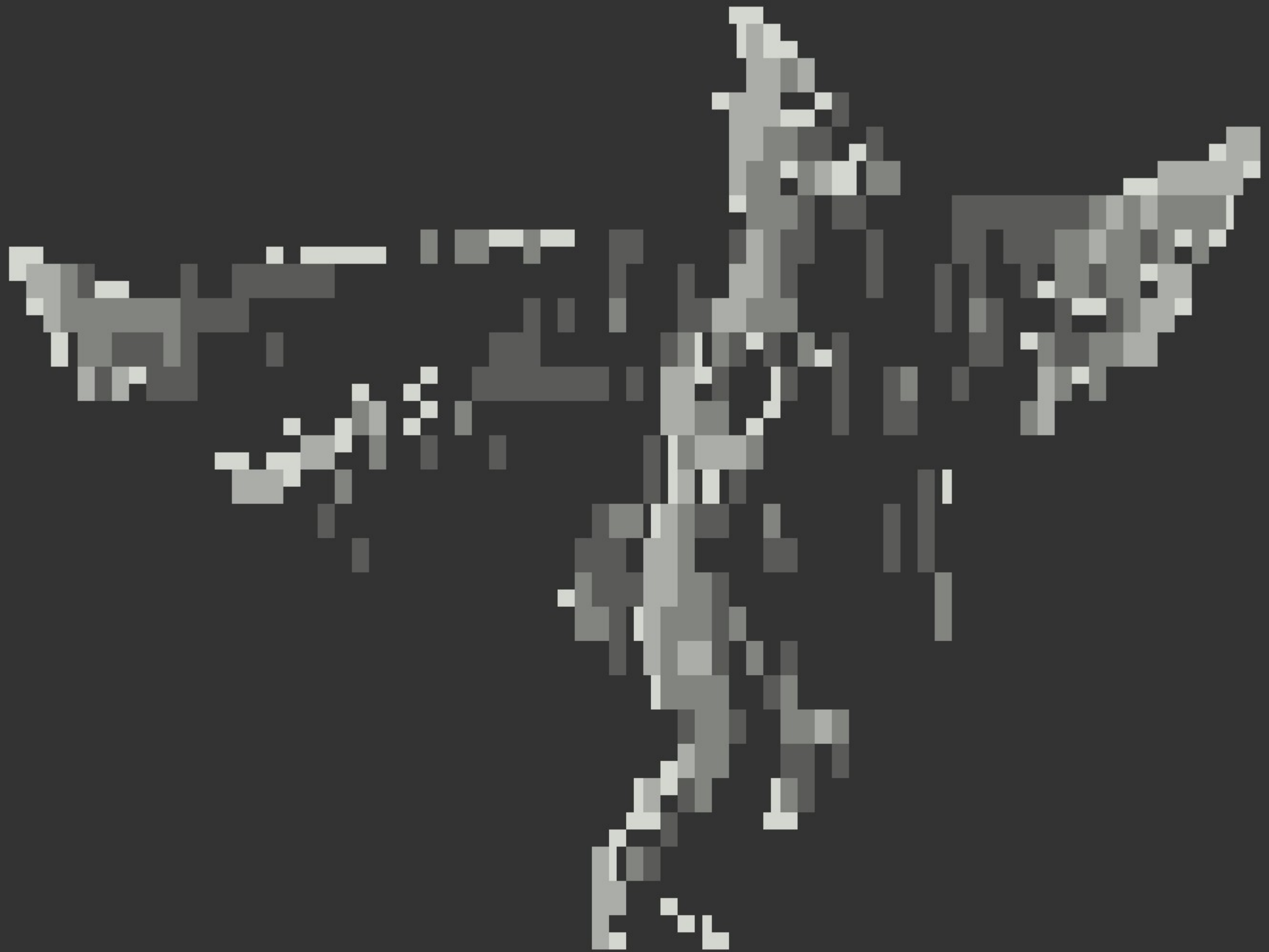
calculates the radiation
processes from relativistic particles

- Synchrotron radiation
  (Blumenthal&Gould1970, Ghisellini1988)
- Bremsstrahlung
  (Baring 1999)
- Inverse-Compton radiation
  (Blumenthal&Gould1970)
  - Arbitrary radiation fields possible, self-consistently used for energy losses in Particles class
  - Also SSC support
- Inelastic pp scattering
  (Kafexhiu2014)

Astro class

holds astrophysical models useful
for source modeling and population syntheses

- galactic structures like arm models
  (Valée2005, Taylor&Cordes1993)
- Magnetic field, gas models
  (Jaffe2010, Ferriere2001)
- VHE-source progenitor model functions
  (Salpeter1955,Weaver1977,Castor1975,Chevalier1989 etc.)
- VHE-source dynamical models (so far SNRs only)
  (Truelove&McKee1999,
  Thin-Shell approximation from Ptuskin2005)



GAPPA

(GAMERA Python PAckage)

gappa

`gappa` is the `Swig`-wrapped `GAMERA` library

- Code is `C++` based
  → object orientation,
  `Set`/`Get` functions heavily used
- Input/output types :

  → `float`: constant input/output parameters

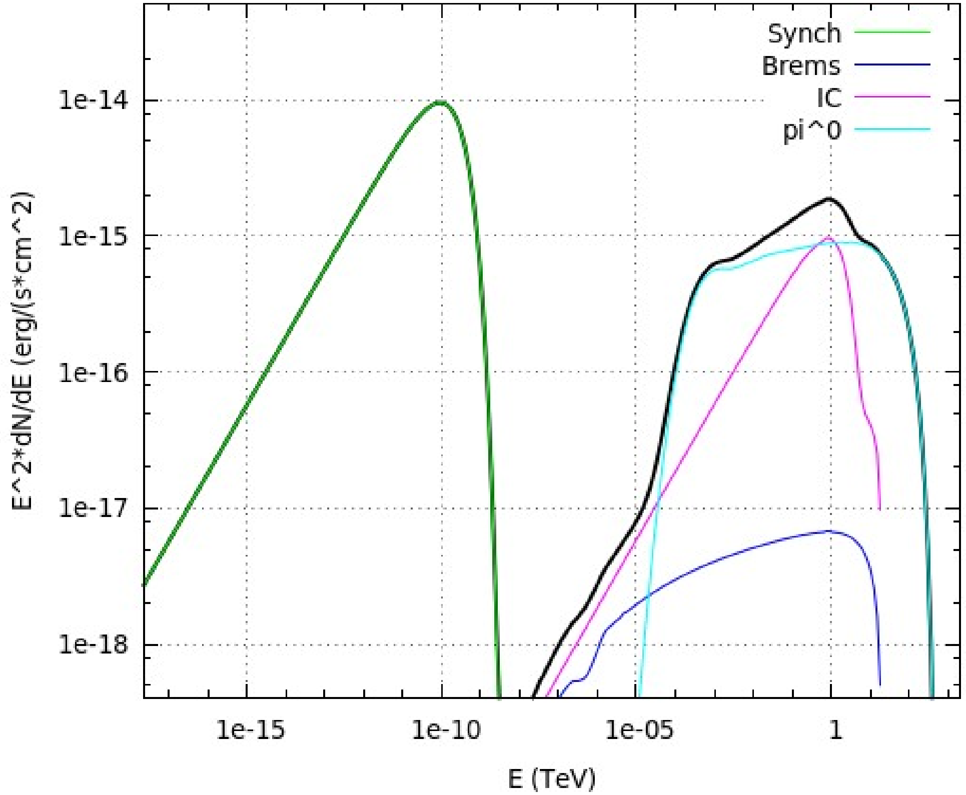
  → `lists`/`numpy-arrays`:
  time evolution of parameters,
  energy distribution of particles

Some examples

1. Calculate SED from pre-defined particle distribution

`Radiation` class

```
fr = gamerapy.Radiation()
fr.SetDistance(d)
fr.SetAmbientDensity(n)
fr.SetBField(b)
fr.AddThermalTargetPhotons(temp1,edens1)
fr.AddThermalTargetPhotons(temp2,edens2)
fr.SetElectrons(electrons)
fr.SetProtons(protons)
fr.CalculateDifferentialPhotonSpectrum()

total_sed = np.array(fr.GetTotalSED())
ic_sed = np.array(fr.GetICSED())
brems_sed = np.array(fr.GetBremsstrahlungSED())
synch_sed = np.array(fr.GetSynchrotronSED())
pp_sed = np.array(fr.GetPPSED())
```



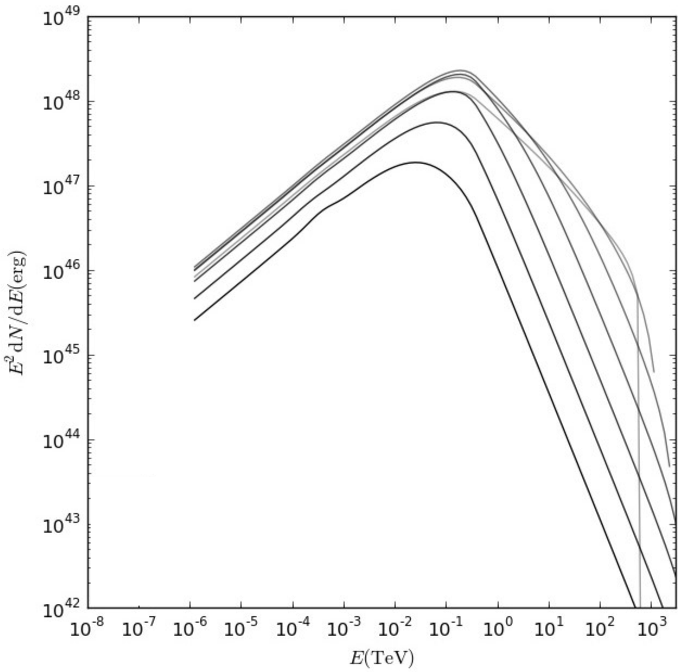
MAX-PLANCK-INSTITUT FÜR KERNPHYSIK

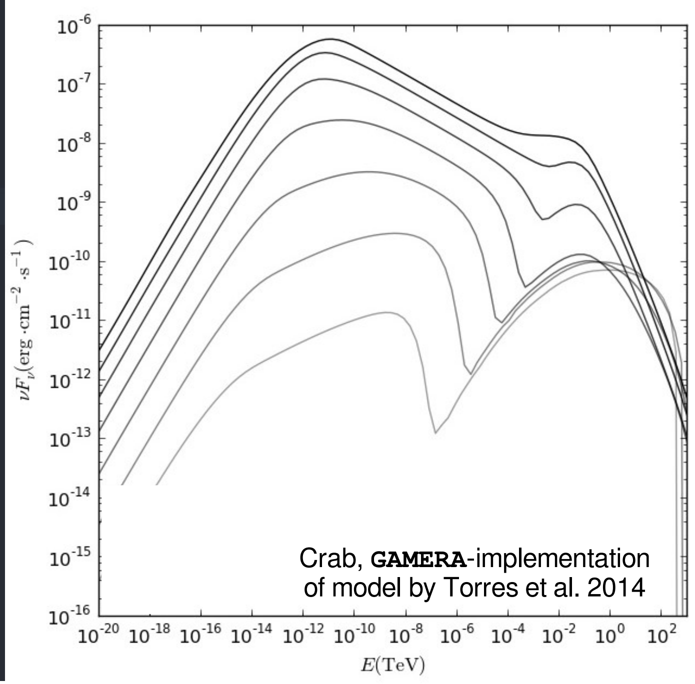
Some examples

2. Electron evolution with variable energy losses and parameters + and subsequent photon emission

`Particles` + **`Radiation`** class

```python
# set particle stuff
fp = gamerapy.Particles()
fp.SetLuminosityLookup(lumt)
fp.SetBFieldLookup(bt)
fp.SetEmaxLookup(emaxt)
fp.SetRadiusLookup(r)
fp.SetVelocityLookup(v)
fp.SetAmbientDensity(dens)
fp.SetEmin(emin)
fp.SetBreakEnergy(ebreak)
fp.SetLowSpectralIndex(spindlow)
fp.SetSpectralIndex(spindhigh)
fp.SetEnergyBinsForNumericalSolver(ebins)
fp.SetAge(age)

# set radiation stuff
fr = gamerapy.Radiation()
fr.SetDistance(dist)
fr.AddThermalTargetPhotons(temp1,edens1)
fr.AddThermalTargetPhotons(temp2,edens2)
fr.AddThermalTargetPhotons(temp3,edens3)
fr.SetAmbientDensity(fp.GetAmbientDensity())
fr.CreateICLossLookup()
fp.SetICLossLookup(fr.GetICLossLookup())

# calculate stuff
fp.SetAge(age)
fr.SetBField(fp.GetBField())
fp.CalculateParticleSpectrum("electrons")
fr.SetElectrons(fp.GetParticleSpectrum())
fr.AddSSCTargetPhotons(fp.GetRadius())
fr.CalculateDifferentialPhotonSpectrum()
ElectronSED = np.array(fr.GetElectronSED())
TotalSED = np.array(fr.GetTotalSED())
ICSED = np.array(fr.GetICSED())
BremsSED = np.array(fr.GetBremsstrahlungSED())
SynchSED = np.array(fr.GetSynchrotronSED())
```





Some examples

2. Electron evolution with variable energy loss
par
and
sub
pho
em

Pa

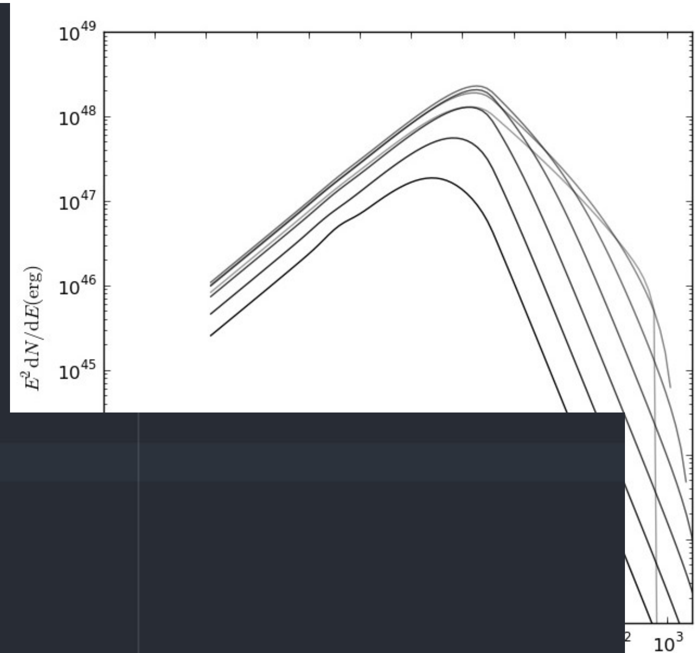
+

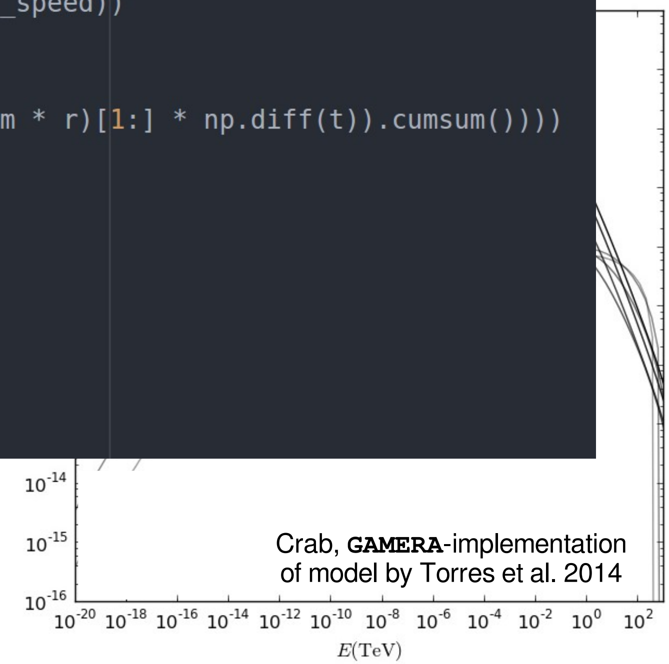
Radiation class

```
# set particle stuff
fp = gamerapy.Particles()
fp.SetLuminosityLookup(lumt)
fp.SetBFieldLookup(bt)
fp.SetEmaxLookup(emaxt)
fp.SetRadiusLookup(r)
fp.SetVelocityLookup(v)
fp.SetAmbientDensity(dens)
fp.SetEmin(emin)
fp.SetBreakEnergy(ebreak)
fp.SetLowSpectralIndex(spindlow)
```

```
def CalculateTimeDependentStuff():
    t = np.logspace(0,math.log10(1.e6*age),80)
    gammap = 1.3333
    vej = math.sqrt(10.*e0/(3.*mej))
    c = math.pow((6./(15.*(gammap-1.)))+289./240.,-0.2);

    lum = (1.-etab)*lum0*(1.+t/tc)**(-1.*(brind+1.)/(brind-1.))
    emax = 3.*eps*gamerapy.el_charge*np.sqrt(etab*lum/((1.-etab)*gamerapy.c_speed))
    r = c*(lum0*t*gamerapy.yr_to_sec/e0)**0.2 * vej*t*gamerapy.yr_to_sec
    v = 1.2*r/(gamerapy.yr_to_sec*t)
    b = np.sqrt(gamerapy.yr_to_sec*etab*6./r**4 * np.concatenate(([0], ((lum * r)[1:] * np.diff(t)).cumsum())))

    lum = np.vstack((t, lum)).T
    b = np.vstack((t, b)).T
    emax = np.vstack((t, emax)).T
    r = np.vstack((t, r)).T
    v = np.vstack((t, v)).T

    return lum, b, emax, r, v
```

```
ElectronSED = np.array(fr.GetElectronSED())
TotalSED = np.array(fr.GetTotalSED())
ICSED = np.array(fr.GetICSED())
BremsSED = np.array(fr.GetBremsstrahlungSED())
SynchSED = np.array(fr.GetSynchrotronSED())
```

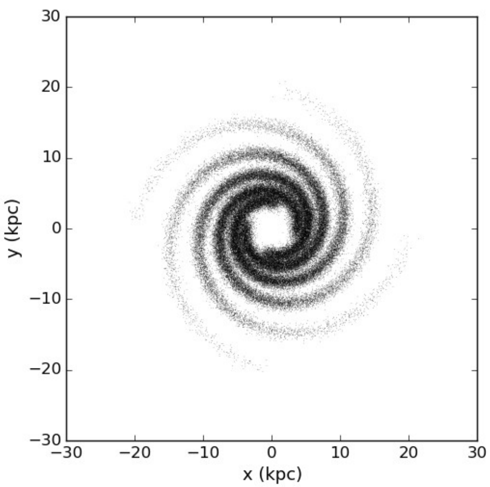


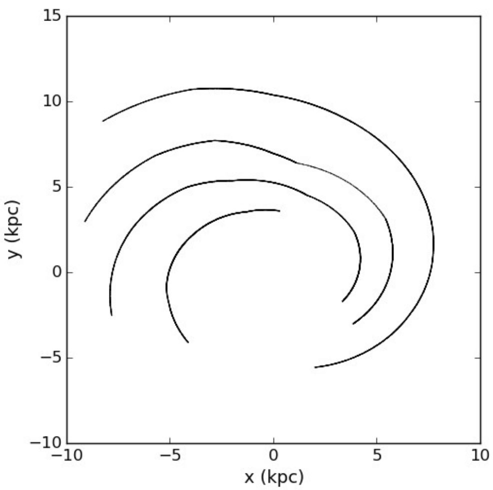


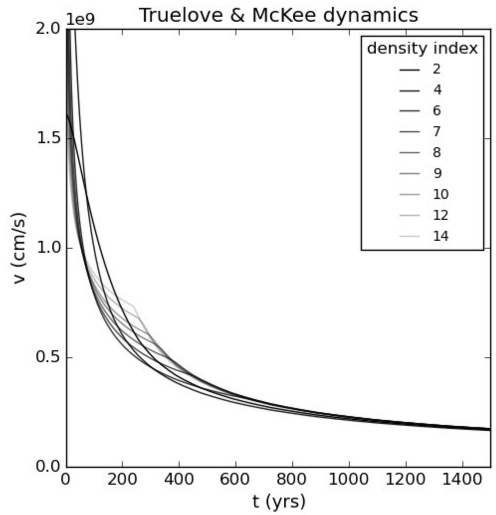
Crab, **GAMERA**-implementation of model by Torres et al. 2014

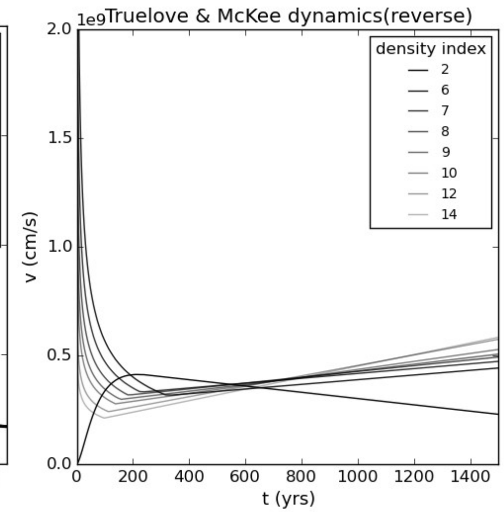
Some examples

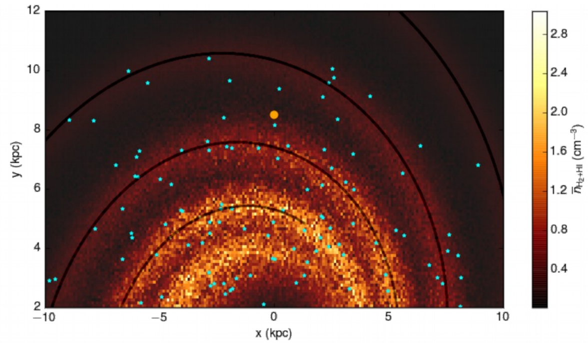
3. Spiral arms, gas density, magnetic field strength, gas column density from Earth, random source coordinates, dynamics `Astro` class

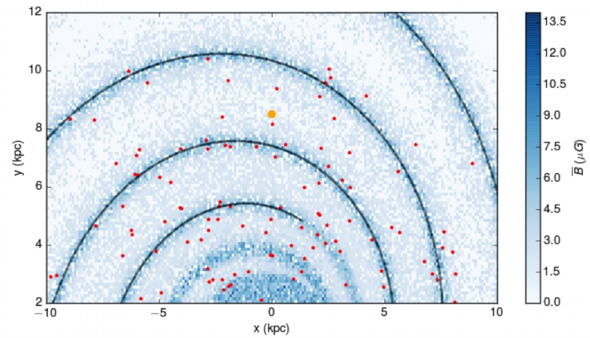


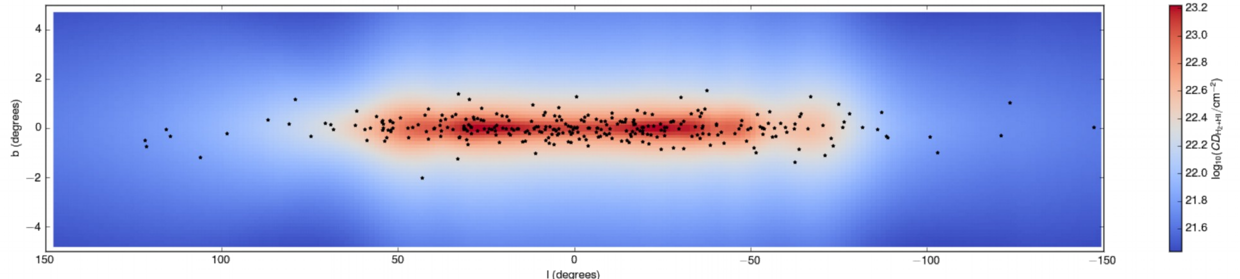












Docu & Download

You can get if from github

https://github.com/JoachimHahn/GAMERA.git

documentation is available on

http://joachimhahn.github.io/GAMERA/

**Note: This is new stuff.
The docu is under construction,
Bugs are being fixed and features
are being added.**

TODO

- Finish the docu
- Adding more features to the code
  - Secondary electrons
  - Synthetic observational data creation
  - Further speed optimisations
  - Grid-solver for energy-space-propagation
  - More dynamics models (PWNs +...)
  - ...
- Upload it to `pypi`
- Find bugs and fix them

Conclusions

- **`GAMERA`** is a new **`C++`** package for source modeling + more in gamma astrophysics
- Allows for straight-forward time-dependent and multi-zone modeling
- Holds tools for source dynamics and population syntheses
- Has been wrapped into a **`python`** package using **`Swig`**

  $\rightarrow$ Now easy to use in scripting
- **Work in progress!** Feel free to try it out and contribute!

There will be a tutorial on Thursday, 9:15!