GammaLib - Feature #3638

GModelSpectral class for Dark matter annihilations

05/04/2021 06:50 AM - Hernández Cadena Sergio

Status:	Rejected	**Start date:**	05/04/2021
Priority:	Normal	**Due date:**	
Assigned To:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

Add spectral class for dark matter annihilations

History

#1 - 05/04/2021 08:19 AM - Knödlseder Jürgen

- Project changed from ctools to GammaLib

Could you please explain a bit what type of spectral model do you need? Specifically, a description of the formulae underlying the model would be helpful.

Please also note that for complex models we now have the GModelSpectralTable class, see http://cta.irap.omp.eu/ctools/users/user_manual/models_spectral.html#table-model. This type of model allows pre-computation of parametric spectral models by an external package, and providing the model through a FITS file. Eventually, this type of model may fit your needs.

#2 - 05/05/2021 03:34 AM - Hernández Cadena Sergio

user#3 wrote:

> Could you please explain a bit what type of spectral model do you need? Specifically, a description of the formulae underlying the model would be helpful.
>
> Please also note that for complex models we now have the GModelSpectralTable class, see http://cta.irap.omp.eu/ctools/users/user_manual/models_spectral.html#table-model. This type of model allows pre-computation of parametric spectral models by an external package, and providing the model through a FITS file. Eventually, this type of model may fit your needs.

Hi!

The reasons to include a spectral class for dark matter are:

1. The GModelSpectralTable assumes that I have the spectra precomputed for a fixed range of energies, which is not the case for dark matter and resulting in loss of energy resolution in the spectrum. For dark matter, the spectra are usually computed for the logarithm of the ratio of the gamma-ray energy and the mass of the dark matter particle.

2. I will introduce, at some point, the possibility to compute the spectrum resulting from contributions from several channels (branching ratios). Honestly I didn't see how to manage them [branching ratios] with GModelSpectralTable.

3. I have the tables already in ASCII files, and read the files using GCsv is a possibility, I think

#3 - 05/05/2021 08:48 AM - Knödlseder Jürgen

I would expect that dark matter particle mass and the branching ratio are parameters of the table model. Varying dark matter particle mass and the branching ratio over the relevant parameter ranges should then allow you to fill the table model.

You would loop over these parameters, and for each parameter combination compute the expected spectrum (according to your point 1 this should be a simple transformation to go from energy to the variable of your DM spectrum). CTA's energy resolution is at best 5%, hence once you chose an energy binning smaller than that there should be no loss of information (the table model interpolates between values).

#4 - 05/05/2021 08:51 AM - Knödlseder Jürgen

- File test.py added

Attached a script that I used to created an exponentially cut-off power law table model. I'm pretty sure that you can adapt it to the format of your dark matter models.

attachment:test.py

#5 - 05/05/2021 05:51 PM - Hernández Cadena Sergio

Hi,

Thank you. I will check the script.

I just want to mention that at this moment the data I have for the spectrum (in fact, dN/dlogx) is organized in a mesh grid of MDM and log10 (E/m). The grid in log10(E/m) is equally spaced. So, passing the spectrum in this way, I still need to tune the GModelSpectralTable class to properly convert to the spectrum (dN/dE).

#6 - 05/05/2021 06:41 PM - Tibaldo Luigi

The easiest solution would actually be to use the data you have to calculate dN/dE and fill that into the table so that no modifications of GammaLib are needed.

#7 - 05/05/2021 07:08 PM - Hernández Cadena Sergio

Hi!

To exemplify my case:

My data is in this mesh grid:

```
logx1      logx2     logx3     . . . logxm
m1  dN/dlogx1  dN/dlogx2  dN/dlogx3       dN/dlogxm
m2
m3
'
'
'
mn dN/dlogx1   dN/dlogx2 dN/dlogx3  . . . dN/dlogxm
```

Then, transforming to the Energy space needed to pass the spectrum in GModelSpectralTable means that I will have a table like this:

```
E1      E2      E3 . . .           Em
m1  dN/dE1   0       ...
m2  dN/dE1   dN/dE2   0      ...
```

```
m3  dN/dE1   dN/dE2   dN/dE3   0 ...
'
'
'
mn   dN/dE1   dN/dE2   dN/dE3   dN/dE4 ... dN/dEm
```

Hope this helps to exemplify what I mean with "losing resolution" when interpolating for small values of masses.

#8 - 05/06/2021 08:52 AM - Knödlseder Jürgen

Here I would recommend to extrapolate the spectrum for small energies to avoid zeros in the table.

#9 - 05/06/2021 03:21 PM - Hernández Cadena Sergio

The problem is that for energies greater than the mass of the dark matter particle, the spectrum is zero.

#10 - 05/06/2021 04:07 PM - Knödlseder Jürgen

I think you can just set the corresponding spectral values to zero.

#11 - 05/06/2021 04:19 PM - Tibaldo Luigi

For the "resolution loss" problem I think that you can just use the finest energy grid that suits the smallest masses you want to study. For the other masses the grid will be finer in the table than what you have computed but that is not an issue.

#12 - 05/06/2021 06:22 PM - Hernández Cadena Sergio

Ok.

I will test this approach, and let you know if works properly

#13 - 05/07/2021 06:34 AM - Hernández Cadena Sergio

Hi!

I created a fits file using the GModelSpectralTable. It looks like ~150 energy bins are sufficient to compute properly the spectrum for masses around a hundred GeV. I am a little worried about memory and time consumption (because of the number of channels and energy bins), but probably it's not a big problem considering that normalization is the only free parameter.

Another reason why I was considering creating the spectral class for dark matter is to incorporate the other two parameters that are encapsulated in the normalization, in particular, if I want to perform studies in one of these parameters. I am going to work with normalization at this moment

#14 - 05/07/2021 07:46 AM - Tibaldo Luigi

Hi Sergio, you can use more than one parameter, and there is no need that one of the parameters is the normalization. I guess in your case the two parameters you want to use are mass and cross section? Just make a primary table hdu as a function of these two parameters, and then declare their names in the 'PARAMETERS' HDU.
I do not know how computing intensive this will be, I have not used extensively the Table model yet.

#15 - 05/07/2021 09:34 AM - Knödlseder Jürgen

I don't think that memory will be an issue, compared to the available RAM on modern computers the FITS table is rather small. On the computational side it should be rather fast since only interpolations are involved, but as Luigi pointed out, the table model is rather new and was not used extensively, hence your feedback will be valuable.

#16 - 05/07/2021 03:21 PM - Hernández Cadena Sergio

Thanks Jürgen

I will test with some simulations during the weekend. I will let you know if I find another issue.

#17 - 05/07/2021 03:25 PM - Hernández Cadena Sergio

Hi Luigi

user#266 wrote:

> Hi Sergio, you can use more than one parameter, and there is no need that one of the parameters is the normalization. I guess in your case the two parameters you want to use are mass and cross section? Just make a primary table hdu as a function of these two parameters, and then declare their names in the 'PARAMETERS' HDU.

In fact, are cross-section and J-factor. Both normalizations. I was thinking in having a table-parameter with just one value, I will test later.

> I do not know how computing intensive this will be, I have not used extensively the Table model yet.

#18 - 05/10/2021 10:10 PM - Hernández Cadena Sergio

I will close this feature.

Thanks for the help!

#19 - 05/11/2021 10:55 AM - Knödlseder Jürgen

- Status changed from New to Rejected

Files

test.py	3.93 KB	05/05/2021	Knödlseder Jürgen