ctools - Feature #1363

Create ctulimit

11/12/2014 11:14 AM - Mayer Michael

Status:	Closed	**Start date:**	11/12/2014
Priority:	Normal	**Due date:**	
Assigned To:	Mayer Michael	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.0.0		

Description

Following #1263, a new ctool to compute upper limits is required.
The name of the tool should be ctulimit. When running the tool, the likelihood function could be inspected to derive the upper limit.
We might also support different calculation algorithms.

Related issues:

Duplicates ctools - Action # 1257: Implement upperlimit calculation	**Closed**	**07/09/2014**

History

#1 - 11/12/2014 11:56 AM - Mayer Michael

On branch *1363-create-ctulimit*, I created a skeleton for the ctulimit-tool. I started this a few days ago. It does not do anything yet but a suggested parfile is already there.
We have to think about where to implement the corresponding functionality in gammalib. I guess the GOptimizer class might be a good place.

#2 - 11/12/2014 05:08 PM - Mayer Michael

To compute an upper limit, there are several approaches.

- integrate the likelihood function up to a point where the confidence level is reached
- inspect the likelihood function to find the parameter value, where the likelihood has changed by {{latex(\Delta LogL)}}, where {{latex(\Delta LogL)}} is defined by the confidence interval
- and probably more

What would you think would be the algorithm best to start with? We probably also need meaningful names in the .par-file.

#3 - 11/14/2014 08:55 AM - Mayer Michael

> We have to think about where to implement the corresponding functionality in gammalib. I guess the GOptimizer class might be a good place.

Another alternative would be use a GFunction, which e.g. can already be integrated. We could add sth like GFunction::find() to get the proper value where the function has changed by a certain value.

#4 - 11/16/2014 03:07 PM - Sanchez David

Hi

both method have been implemented in the ST of Fermi. The simplest one is the second and has been implemented for issue https://cta-redmine.irap.omp.eu/issues/1263

D

#5 - 11/16/2014 03:49 PM - Mayer Michael

Great. Thanks for the info. Now, we just have to translate the functions to C++.

#6 - 11/19/2014 11:26 AM - Knödlseder Jürgen

There is just one problem with that method that we see for example in Fermi: in first approximation, the log-likelihood function is a parabola near the minimum. In some cases, however, the formal minimum may occur for negative flux values, but as we limit flux values to non-negative values, the "minimum" of the log likelihood function will be on the raising part of the parabola. The formal upper limit is then too small.

One way out of this is to determine the parameters of the parabola, and then shift the parabola minimum to 0 in case that the formal minimum is negative. This avoids at least to have too small upper limits in case that the flux boundary is reached.

#7 - 11/24/2014 10:39 PM - Sanchez David

If I remember well, the "Integral" method developed by S. Fegan for the Fermi ST was made for such purposes. It is based on helene 83 (right?). I think this can also be implemented here

#8 - 02/09/2015 10:38 AM - Mayer Michael

- Status changed from New to Feedback

- Assigned To set to Mayer Michael

- Target version set to 1.0.0

- % Done changed from 0 to 80

Computing upper limits is crucial to proceed writing spectral and light curve computation tools. I therefore implemented a first version of ctulimit, which is available on branch *1363-create-ctulimit*.
The algorithm uses a simple bisection method, similar as in the scripts provided by David. I furthermore added a rst-file to the doc folder and wrote a simple unit test.
I am still a bit uncertain about the output format: currently only three values are dumped into an output ascii file, namely the differential upper limit at the given reference energy, the integral flux limit in the given energy range and the energy flux upper limit (also in the given range).

What do you think?

#9 - 02/09/2015 03:51 PM - Sanchez David

Hi Michael

I am working on it also but I am looking at other methods and in particular one implemented in the fermi ST. Also we can thinks to use Feldman cousin as HESS

I will have a look at your code

cheers
David

#10 - 02/09/2015 06:35 PM - Mayer Michael

That's great David. It would be nice to add further functionality and sophistication to the tool. For now, it plainly searches for the root of the (modified) likelihood function. We might want to add more evolved mathematical methods and other UL calculation algorithms.
Nevertheless, it would be good to have the latest code in devel to be able to proceed with the spectrum and light curve tools.

#11 - 02/20/2015 05:20 PM - Mayer Michael

- Status changed from Feedback to Pull request

Any thoughts on the upper limit computation code? Is it ready to get merged?

#12 - 02/20/2015 05:57 PM - Knödlseder Jürgen

I'm also late on that. Will check a.s.a.p.

#13 - 02/21/2015 12:16 PM - Sanchez David

Hi

I had a look and I think I have a couple of comments: * in the code we should check that the model is a power law, there is no reason to have other model * more important, the index should be fixed by the user. currently it is free right?

As I said, I am working on a other mathematical method but this is much more complicated that I was thinking. I 'll create another branch to propose you this piece of code

cheers
david

#14 - 02/21/2015 01:22 PM - Mayer Michael

Thanks for looking at the code.

> in the code we should check that the model is a power law, there is no reason to have other model * more important, the index should be fixed by the user. currently it is free right?

I tend to object on this. The upper limit code should not care about the spectral shape. The upper limit gets computed as a function of the normalising parameter only. All other parameters are kept as they are. There is no need to fix parameters in the code (which would confuse the user), since there is no fit performed which would check if a parameter is free. See for example the calculation of spectral points: the user can decide to give any spectral model. In each energy bin, the flux point as well as the upper limit are then computed with the assumption of a fixed spectral shape (given by the user). So I would argue that there is indeed the need to support all spectral models.

#15 - 02/21/2015 01:59 PM - Sanchez David

I don't see the point to compute an UL for a logparabola model or a power-law with a cut off. You can not find a source so you cannot constrain the other parameters. What is done usualy is that you assume a certain index for a power law model and then compute the UL. The assumed index is up

to the user who should explicitly decide the value. I have missunderstood the code and since there is no fit this is fine. I would nevertheless add the possibility for the user to put the power-law index to a certain value by adding a parameter in the par file.

As far as I know, I never see an UL computed for more complicated model than power-law or model with 1 parameter maybe I am wrong

#16 - 02/21/2015 02:22 PM - Mayer Michael

> I don't see the point to compute an UL for a logparabola model or a power-law with a cut off. You can not find a source so you cannot constrain the other parameters.

I agree that in case there is no source, you wouldn't compute an upper limit for a more complex model. The following example may, however, illustrate more clearly what I mean:
You make an overall fit for the Crab Nebula, say with a LogParabola model. Afterwards you want to calculate spectral points. In each energy bin, the index and curvature of the LogParabola stay fixed (to their best fit values) while the normalisation parameter gets fitted. For the latter parameter you want to compute an upper limit in each energy bin, too. The spectral parameters (index, curvature) stay completely the same, while the upper limit for the flux is being computed. Accordingly, the upper limit tool should be flexible to deal with any kind of model.

The functionality stays the same if you use a power law or not. If a user wants to compute an upper limit under the hypothesis of a curved power law (like e.g. for the Crab, see example above) this should be possible. The user may also want to compute the upper limit even though the source is significant (like we do it for spectral points).

#17 - 02/23/2015 08:58 AM - Sanchez David

well, I believe this will not change anything in such case, I would agree with you. Nevertheless, I think that the use should be aware about the used value of other parameter and how he can change this. Maybe adding some info in the log?

You exemple make think also that, the UL might be computed for another parameter like the Ecut of a PL with a cut off og the curvature parameter for a logparabola.

#18 - 02/24/2015 01:29 AM - Knödlseder Jürgen

- Status changed from Pull request to Feedback

- % Done changed from 80 to 90

I finally found the time to integrate your code. It looks fine. Here a list of things that I adapted:

- cl for the confidence level is very cryptic, I'd propose to rename to confidence
- I implemented a general confidence level computation
- typically, log-likelihood is abbreviated by logL in ctools and GammaLib, hence I propose to rename the corresponding variables
- I streamlined the code a bit to avoid shuffling of model containers (in fact as all methods pass either pointers or reference you have direct access to the model parameter without any need to remove and append model components)
- I'm not sure why you used the factor value instead of the value. I changed the code to directly use the value (avoids handling of scale)
- I did some reformatting of the output (using the standard gammalib::parformat function)

- Instead of using always component 0 of a spectral model I now search explicitly for Normalization, Prefactor or Integral. The tool throws an error if none of them is found.

Note that I implemented a has_par() method for spectral model components in GammaLib, hence you need to update also GammaLib to make the code compile.

I merged the code into devel but keep the issue on Feedback so that you can cross-check the code.

I wonder whether the plain ASCII file is very useful as output. Do we need in fact the results in an output file? Or should we implement a way to add this information into the XML file? For example by adding an ulimit attribute to the parameter, equivalent to the error attribute?

#19 - 02/24/2015 09:27 AM - Mayer Michael

Knödlseder Jürgen wrote:

> I finally found the time to integrate your code. It looks fine. Here a list of things that I adapted:
>
> - cl for the confidence level is very cryptic, I'd propose to rename to confidence

Agree

> - I implemented a general confidence level computation

Great, this provides now much more flexibility. I wasn't aware of the computation of this value.

> - typically, log-likelihood is abbreviated by logL in ctools and GammaLib, hence I propose to rename the corresponding variables

Agree

> - I streamlined the code a bit to avoid shuffling of model containers (in fact as all methods pass either pointers or reference you have direct access to the model parameter without any need to remove and append model components)

Also agree. I guess this was still a relic from cttsmap from where I copied some parts of the code (which I know is bad habit). Code is much cleaner now.

> - I'm not sure why you used the factor value instead of the value. I changed the code to directly use the value (avoids handling of scale)

I thought it might be numerically more stable to use the factor value because the code does not have to deal with small numbers like, e.g., the prefactor. But this does not seem to be an issue after all.

> - I did some reformatting of the output (using the standard gammalib::parformat function)

Output looks good.

> - Instead of using always component 0 of a spectral model I now search explicitly for Normalization, Prefactor or Integral. The tool throws an error if none of them is found.

Yes that makes sense. I quickly checked all available spectral models. With this new functionality GModelSpectralConst and GModelSpectralNodes cannot be used for upper limit computation ("Value" and "Intensity#" parameter names respectively). I think for the latter, this makes sense since we have multiple normalisations. I cannot think of a case where one wants to compute an upper limit of GModelSpectralConst, nevertheless, we might want to keep this possibility? I propose adding "Value" to the allowed parameter names or change the parameter name to "Normalization".

> Note that I implemented a has_par() method for spectral model components in GammaLib, hence you need to update also GammaLib to make the code compile.
>
> I merged the code into devel but keep the issue on Feedback so that you can cross-check the code.
>
> I wonder whether the plain ASCII file is very useful as output. Do we need in fact the results in an output file? Or should we implement a way to add this information into the XML file? For example by adding an ulimit attribute to the parameter, equivalent to the error attribute?

I am also not happy with the output format but couldn't think of a better idea. I agree that adding it to the xml file or to the model container would make sense. Would you add the ulimit attribute directly to GModelPar and not to GModelSky (analogous to the TS-value)? Still, an upper limit value needs to come with a reference energy or energy boundaries (in case of flux limits). I am not sure how to keep all this information together.

Thanks a lot for your effort. I also made some quick checks which looked fine. Apart from the output format and the parameter name conflict with GModelSpectralConst, this tool seems to be finished.

#20 - 02/24/2015 10:40 AM - Knödlseder Jürgen

Mayer Michael wrote:

> I thought it might be numerically more stable to use the factor value because the code does not have to deal with small numbers like, e.g., the prefactor. But this does not seem to be an issue after all.

Internally the multiplication will be made anyways, hence the numerical result is the same.

> Yes that makes sense. I quickly checked all available spectral models. With this new functionality GModelSpectralConst and GModelSpectralNodes cannot be used for upper limit computation ("Value" and "Intensity#" parameter names respectively). I think for the latter, this makes sense since we have multiple normalisations. I cannot think of a case where one wants to compute an upper limit of GModelSpectralConst, nevertheless, we might want to keep this possibility? I propose adding "Value" to the allowed parameter names or change the parameter name to "Normalization".

Agree. I think I did this check yesterday but apparently I have overlooked GModelSpectralConst. Indeed, GModelSpectralNodes cannot be used. I must say I don't know how this could used in an upper limit computation.

> I wonder whether the plain ASCII file is very useful as output. Do we need in fact the results in an output file? Or should we implement a way to add this information into the XML file? For example by adding an ulimit attribute to the parameter, equivalent to the error attribute?

> I am also not happy with the output format but couldn't think of a better idea. I agree that adding it to the xml file or to the model container would make sense. Would you add the ulimit attribute directly to GModelPar and not to GModelSky (analogous to the TS-value)? Still, an upper limit value needs to come with a reference energy or energy boundaries (in case of flux limits). I am not sure how to keep all this information together.

You are right, adding simply ulimit is not very helpful.

What are actually your use cases for the ASCII file? Identifying may help to find a better solution.

> Thanks a lot for your effort. I also made some quick checks which looked fine. Apart from the output format and the parameter name conflict with GModelSpectralConst, this tool seems to be finished.

I agree to add Value to the possible parameter names.

#21 - 02/24/2015 03:21 PM - Mayer Michael

> Indeed, GModelSpectralNodes cannot be used. I must say I don't know how this could used in an upper limit computation.

Since there is no overall normalisation parameter but multiple ones, I'd also say this model cannot be used for the limit computation. Maybe add a meaningful exception in that case?

> What are actually your use cases for the ASCII file? Identifying may help to find a better solution.

There are no real use cases yet. For csspec, the limit is computed without writing the value to an output file. It stays in memory and gets written to the output spectrum fits file. An ascii file with blank numbers as it is at the moment, is probably the worst thing to do. We might still think about a way to write the output to xml file:

```
<source name="Crab" type="PointSource">
  <uimit confidence="0.95" method="bisection">
    <differential eref="1e6" value="3.1e-17" unit="ph/cm2/s/MeV"/>
    <integral emin="1e6" emax="1e8" value="2e-11" unit="ph/cm2/s"/>
    <energy_integral emin="1e6" emax="1e8" value="3e-11" unit="erg/cm2/s"/>
  </ulimit>
</source>
```

This could be in a separate output file or added to the <source> section of the model xml file. We just have to make sure to not interfere with GModel::read(&xml). Maybe even have a GULimit class taking care of read/write. Or is that all an overkill?

#22 - 02/25/2015 01:33 AM - Knödlseder Jürgen

Mayer Michael wrote:

> Indeed, GModelSpectralNodes cannot be used. I must say I don't know how this could used in an upper limit computation.

> Since there is no overall normalisation parameter but multiple ones, I'd also say this model cannot be used for the limit computation. Maybe add a meaningful exception in that case?

There is already an exception that tells you that no valid flux parameter has been found. Do we need more?

> What are actually your use cases for the ASCII file? Identifying may help to find a better solution.

> There are no real use cases yet. For csspec, the limit is computed without writing the value to an output file. It stays in memory and gets written to the output spectrum fits file. An ascii file with blank numbers as it is at the moment, is probably the worst thing to do. We might still think about a way to write the output to xml file:
> [...]
> This could be in a separate output file or added to the <source> section of the model xml file. We just have to make sure to not interfere with GModel::read(&xml). Maybe even have a GULimit class taking care of read/write. Or is that all an overkill?

I won't add a special class just to write an output. How about postponing that issue to the moment where we have a real use case?

It then would be logical to remove for the moment the output file (we may keep the code, but remove the call and the parameter in the parfile) so that we don't have to change a format later (adding is always easier than changing).

#23 - 02/25/2015 09:22 AM - Mayer Michael

> There is already an exception that tells you that no valid flux parameter has been found. Do we need more?

Since this exception is only thrown if GModelSpectralNodes is given, we might throw an error with a message like: "you cannot use GModelSpectralNodes for upper limit computation since the normalisation parameter is ambiguous. Choose any other model instead". But I would say this is a rather unimportant detail.

> I won't add a special class just to write an output. How about postponing that issue to the moment where we have a real use case?

Agree

> It then would be logical to remove for the moment the output file (we may keep the code, but remove the call and the parameter in the parfile) so that we don't have to change a format later (adding is always easier than changing).

Fine with me, too. At the end, everything important is available in the log-file anyway.

#24 - 02/25/2015 11:45 PM - Knödlseder Jürgen

- % Done changed from 90 to 100

I made the modifications and merged into devel. Last chance for feedback before closing the issue.

#25 - 02/26/2015 12:23 AM - Mayer Michael

Looks good, ready to close.

#26 - 02/26/2015 08:35 AM - Knödlseder Jürgen

- Status changed from Feedback to Closed