

GammaLib Tech-Note

Response computations

GammaLib-TN0003

Version: 1.0
11 April 2020

Author: Jürgen Knödlseider
Approved by: Jürgen Knödlseider

Institut de Recherche en Astrophysique et Planétologie (IRAP)
9, avenue du Colonel-Roche
31028 Toulouse Cedex 4
FRANCE

This page intentionally left blank

Contents

1	Scope	1
2	Current implementation	1
3	Modifications for binned analysis	2
4	Implementing pre computation	2
5	Gradients of spatial parameters	3
5.1	Point source	4
5.2	Gaussian source	4

1 Scope

This technical note suggests a modification of the response computation that is useful for binned analysis and that can lead to a speed-up of the model fitting of extended models. The note relates to the Redmine issue 3203 (see <https://cta-redmine.irap.omp.eu/issues/3203>).

2 Current implementation

The `G0bservation::likelihood_poisson_binned()` method computes the log-likelihood, gradients and curvature matrix using the following formulae:

$$-\ln L = \sum_i n_i \ln e_i - e_i \quad (1)$$

$$\frac{\partial \ln L}{\partial a_l} = \sum_i \left(1 - \frac{n_i}{e_i}\right) \frac{\partial e_i}{\partial a_l} \quad (2)$$

$$\frac{\partial^2 \ln L}{\partial a_l \partial a_m} = \sum_i \left(\frac{n_i}{e_i^2}\right) \frac{\partial e_i}{\partial a_l} \frac{\partial e_i}{\partial a_m} \quad (3)$$

with n_i being the number of observed events and e_i the number of predicted events, and a_l the model parameters. Now

$$e_i = \tilde{e}_i \times \Delta_i \quad (4)$$

$$\frac{\partial e_i}{\partial a_l} = \frac{\partial \tilde{e}_i}{\partial a_l} \times \Delta_i \quad (5)$$

where Δ_i is the bin size of event i and

$$\tilde{e}_i = \sum_k M_{k,e}(E) \times M_{k,t}(t) \times \int_p M_{k,p}(p) R_i(p, E, t) dp \quad (6)$$

for a sky model, with $R_i(p, E, t)$ being the Instrument Response Function, and $M_{k,e}(E)$, $M_{k,t}(t)$ and $M_{k,p}(p)$ the spectral, temporal and spatial components of the sky model. The gradients for spectral parameters are computed using

$$\frac{\partial \tilde{e}_i}{\partial a_l} = \sum_k M_{k,t}(t) \times \frac{\partial M_{k,e}(E)}{\partial a_l} \times \int_p M_{k,p}(p) R_i(p, E, t) dp \quad (7)$$

and for temporal parameters using

$$\frac{\partial \tilde{e}_i}{\partial a_l} = \sum_k M_{k,e}(E) \times \frac{\partial M_{k,t}(t)}{\partial a_l} \times \int_p M_{k,p}(p) R_i(p, E, t) dp \quad (8)$$

For spatial parameters no analytical computation is possible and the gradients are computed using a difference method

$$\frac{\partial \tilde{e}_i}{\partial a_l} = \sum_k \frac{M_{k,e}(E) M_{k,t}(t) \int_p M_{k,p}(p|a_l + h) R_i(p, E, t) dp - M_{k,e}(E) M_{k,t}(t) \int_p M_{k,p}(p|a_l - h) R_i(p, E, t) dp}{2h} \quad (9)$$

The costly factor in model fitting of extended models is the evaluation of the integrals

$$I_{i,k} = \int_p M_{k,p}(p) R_i(p, E, t) dp \quad (10)$$

The most time-consuming part in the evaluation of the integral are the coordinate transforms that map p' into p , and to transform from the sky system to a local system of the model (or the instrument). For binned analysis, many events share the same p' (for example all events in a same spatial pixel but with different energies). Hence computation time can be saved when doing the coordinate transform only once for all events that share the same spatial direction. Eq. 10 should therefore be evaluated for bunches of events.

Here is a possible scheme that could be implemented:

- Before entering the event loop, `GObservation::likelihood_poisson_binned()` pre computes the integrals $I_{i,k}$ for all sky models. In case that a sky model has free parameters, also the gradients $\frac{\partial I_{i,k}}{\partial a_l}$ with respect to all free parameters a_l are pre computed.
- The `GResponse::eval_prob()` method checks whether for a given model pre computed $I_{i,k}$ are available. If yes, instead of calling `GResponse::irf_spatial` the method will use the pre computed $I_{i,k}$.
- The `GResponse::eval_prob()` method deals also with the numerical computation of the gradients $\frac{\partial I_{i,k}}{\partial a_l}$ for all free sky model parameters. This allows to use the pre computed gradients $\frac{\partial I_{i,k}}{\partial a_l}$ instead. For all parameters for which spatial gradients exist, `GResponse::eval_prob()` will set the `has_grad()` flag, so that numerical gradients are no longer computed in `GObservation::model()`.

In the case that energy dispersion should be considered, $I_{i,k}$ and $\frac{\partial I_{i,k}}{\partial a_i}$ would be needed for an important number of true photon energies (33 in the current implementation). This would be very likely unmanageable. An approximation can be made, however, that uses the measured energy E' instead of the true energy E for the computation of Eq. 10. Energy dispersion would still be applied for the spectral model component, yet it would be assumed that the $I_{i,k}$ and $\frac{\partial I_{i,k}}{\partial a_i}$ would be constant under small variations of E . In that case the energy dispersion integration should be moved from `GResponse::convolve()` to `GResponse::eval_prob()` so that it only applies to the spectral part of the model, and consequently, the spectral gradients should be also computed numerically in this modified `GResponse::eval_prob()` method. Then, the special treatment for energy dispersion in `GObservation::model()`, where for the moment all gradients are computed numerically, will not be needed anymore.

Currently, Eq. 10 is computed for every event using the method

[illegible]

To benefit from common computations for all events a method needs to be implemented returning a vector $I_{i,k}$ for all events in an observation

```
virtual GVector irf_spatial(const GSource&      source,
                           const GObservation& obs) const;
```

This would have the advantage that the event loop can be put in the innermost possible place, avoiding to repeat any computations that do not need to be repeated for every event. This means that the class **GIntegral**, that is widely used for the IRF integration, must be extended to support vector integration. Specifically, it must provide a method

```
GVector GIntegral::trapzd(const double& a,
                          const double& b,
                          const int&    n,
                          GVector      result);
```

and a method

```
GVector GIntegral::romberg_vector(const double& a,
                                  const double& b,
                                  const int&    order)
```

to support vector integration. Furthermore, a new class **GFunctions** is needed that provides

```
virtual GVector eval(const double& x) = 0;
```

and that can be used to implement the integration kernels. **GIntegral** must be able to hold a pointer to such a kernel, and constructor and kernel access methods need to be added:

```
explicit GIntegral(GFunctions* kernel);
void      kernel(GFunctions* kernel);
const GFunctions* kernel(void) const;
```

5 Gradients of spatial parameters

Using the definition of the gradient

$$\frac{\partial f(x)}{\partial x} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x-h)}{2h} \quad (11)$$

and applying the integral sum rule

$$\int f(x) + g(x) dx = \int f(x) dx + \int g(x) dx \quad (12)$$

one can write

$$\frac{\partial I_{i,k}}{\partial a_l} = \lim_{h \rightarrow 0} \frac{\int_p M_{k,p}(p|a_l + h) R_i(p, E, t) dp - \int_p M_{k,p}(p|a_l - h) R_i(p, E, t) dp}{2h} \quad (13)$$

$$= \lim_{h \rightarrow 0} \frac{\int_p (M_{k,p}(p|a_l + h) - M_{k,p}(p|a_l - h)) R_i(p, E, t) dp}{2h} \quad (14)$$

$$= \lim_{h \rightarrow 0} \int_p \frac{M_{k,p}(p|a_l + h) - M_{k,p}(p|a_l - h)}{2h} R_i(p, E, t) dp \quad (15)$$

$$= \int_p \frac{\partial M_{k,p}(p)}{\partial a_l} R_i(p, E, t) dp \quad (16)$$

In other words, the gradients of the spatial parameters are simply the model gradients integrated over the kernel of the response functions.

5.1 Point source

The point source can be considered as a Dirac function $\delta(x - x_0)$, where x_0 is the location of the source in one dimension. Using

$$f(x)\delta'(x) = -f'(x)\delta(x) \quad (17)$$

and

$$\int f(x)\delta(x) = f(0) \quad (18)$$

it follows

$$\int f(x)\delta'(x) = - \int f'(x)\delta(x) = -f'(0) \quad (19)$$

and hence

$$\frac{\partial I_{i,k}}{\partial a_l} = - \frac{\partial R_i(p, E, t)}{\partial p} \quad (20)$$

5.2 Gaussian source

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \quad (21)$$

$$\frac{\partial f(x)}{\partial \mu} = \frac{1}{\sigma^2\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} = \frac{1}{\sigma} f(x) \quad (22)$$

$$\frac{\partial f(x)}{\partial \sigma} = \frac{-1}{\sigma^2\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} + \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \frac{x-\mu}{\sigma^2} = f(x) \left(\frac{x-\mu}{\sigma^2} - \frac{1}{\sigma} \right) \quad (23)$$