GammaLib - Feature #1459

function for calculating the containment radius for GCTAPsf and all inhereting classes

05/13/2015 07:05 PM - Kelley-Hoskins Nathan

Status:	Closed	**Start date:**	05/13/2015
Priority:	Normal	**Due date:**	
Assigned To:	Kelley-Hoskins Nathan	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	1.0.0		

Description

Comparing the psf in different classes, obserations, telescopes, and code formats can difficult, but the psf's containment radius is a pretty universal way to do this. However, it seems theres no native way in gammalib/ctools to calculate this, making comparisons and error checking difficult.

So, I propose an additional virtual function for GCTAPsf (and its inheriting classes), where a user gives a containment fraction(0.0-1.0), and the point in the energy/theta/phi/zenith/azimuth parameter space, and the containment radius containing that fraction of events is returned, in radians. I imagine the function prototype in GCTAPsf.hpp would look something like:

```
   virtual double      containment_radius( const double& fraction,
                         const double& logE,
                         const double& theta = 0.0,
                         const double& phi = 0.0,
                         const double& zenith = 0.0,
                         const double& azimuth = 0.0,
                         const bool&   etrue = true) const = 0;
```

I have written a python function that calculates the containment radius from the existing GCTAPsf::operator(), but it has to take a bunch of samples(slow), and then linearly interpolate between the two closest ones(inaccurate). Theres probably a better algorithm, or it may be directly computable from the psf function parameters, but the code below may still be useful to whomever is adding these functions.

```
import gammalib, numpy

def containment_radius( run, en, th, contain=0.5, npts=100 ) :
  """Calculate the containment radius that contains 'contain' fraction of the events.

  Args:
    contain: float, 0.0-1.0, containment fraction to find radius for
    run    : GCTAObservation object, should have psf irf loaded
    en     : float, energy, log10TeV
    th     : float, camera offset, radians
    npts : number of times to sample the psf between 0 and delta_max, higher npts = higher accuracy in the returned value

  Returns:
    float, radians, radius containing 'contain' fraction of events
  """
  samps = numpy.zeros(npts)
  dmax  = run.response().psf().delta_max( en, th ) # radians
  delta = dmax / npts # radians

  # loop over each point
  for i in range(npts) :
    d = i * delta # radians
    counts_per_sr = run.response().psf()( d, en, th )   # counts per sr
    if counts_per_sr == 0.0 :
      print 'warning: containment_radius(en=%f, th=%f): 0 counts at radius %.3f deg, probably no psf info' % ( en, th, d *
gammalib.rad2deg )
      return 0.0
```

```
  # integrate these counts around the circle of radius d
  # counts * perimeter of circle with that radius,
  # may get less accurate when d is large (>>10deg?)
  samps[i]      = counts_per_sr * 2 * gammalib.pi * d

# total counts in our psf from 0.0 to dmax
total_counts = numpy.sum(samps)

# loop over each sample, check if we cross the containment fraction
for j in range(npts-1) :
  partial_counts      = numpy.sum( samps[:j+1] )
  partial_counts_next = numpy.sum( samps[:j+2] )
  cfrac      = partial_counts      / total_counts
  cfrac_next = partial_counts_next / total_counts
  radius      =  j    * delta
  radius_next = (j+1) * delta

  # check if we have crossed the containment fraction
  if cfrac < contain and cfrac_next > contain :
    # if so, linearly interploate to find what radius has the target containment fraction
    # starting from http://en.wikipedia.org/wiki/Linear_equation , the Two-point form
    contain_radius = radius + ( (contain - cfrac) * (radius_next - radius) / (cfrac_next - cfrac ) )
    return contain_radius

raise LookupError('Could not find containment fraction %.2f within radii 0.0 and %.3f (radians)' % (contain, dmax) )
```

Related issues:

Duplicates GammaLib - Action # 1310: add GCTAMeanPsf::ctr(double percent)	**Closed**	**08/04/2014**

History

#1 - 05/19/2015 07:45 PM - Knödlseder Jürgen

- Project changed from ctools to GammaLib

#2 - 05/19/2015 07:50 PM - Knödlseder Jürgen

I fully support the addition of such a method to the virtual interface. Would you be willing to implement that method for the various PSF classes?

We could think about having this an implement method at the GCTAPsf level which simply integrates the PSF.

But ultimately a derived class implementation may provide faster computation.

#3 - 05/20/2015 12:26 AM - Kelley-Hoskins Nathan

I'm already pretty busy getting a veritas-ctools data converter functional for my research, and its the last 6 months of my PhD. I might try one weekend, once the converter is finished, but I don't think I can put my name down for it just yet.

#4 - 05/20/2015 11:10 AM - Mayer Michael

If we don't find the time and manpower to implement this now, this could be a nice and simple project for the coding sprint.

#5 - 05/20/2015 03:39 PM - Knödlseder Jürgen

Mayer Michael wrote:

> If we don't find the time and manpower to implement this now, this could be a nice and simple project for the coding sprint.

I agree that this could be a nice issue for the sprint.

#6 - 06/30/2015 02:37 PM - Kelley-Hoskins Nathan

- Assigned To set to Kelley-Hoskins Nathan

- Target version set to 1.0.0

#7 - 07/01/2015 05:38 PM - Kelley-Hoskins Nathan

- Status changed from New to Pull request

- % Done changed from 0 to 90

I verified that the values GCTAPsfKing::containment_radius() produces are correct, but someone should probably look over the calculations in GCTAPsf2D/GCTAPsfVector/GCTAPsfPerfTable::containment_radius() to make sure I didn't miss something, just to be safe.

#8 - 07/01/2015 06:14 PM - Knödlseder Jürgen

- Status changed from Pull request to In Progress

I integrated your code. It's in the integration branch and will move to devel once all tests are successful.

Concerning the unit tests, what I typically do is test specific values, so that any regression on the actual computations can be tracked. To do this, reference values are either known, or I just use the actual value of a computation so at least we track regression.

Could you add some tests that check for specific values?

#9 - 10/27/2015 10:20 PM - Knödlseder Jürgen

- Status changed from In Progress to Closed

- % Done changed from 90 to 100

Merged into devel.