

## GammaLib - Feature #102

### Optimize GNodeArray computations

03/05/2012 10:26 PM - Knödlseider Jürgen

<b>Status:</b>	Closed	<b>Start date:</b>	03/05/2012
<b>Priority:</b>	Normal	<b>Due date:</b>	
<b>Assigned To:</b>	Knödlseider Jürgen	<b>% Done:</b>	100%
<b>Category:</b>		<b>Estimated time:</b>	40.00 hours
<b>Target version:</b>	00-07-00		
<b>Description</b>			
<p>GNodeArray is the central method for interpolations of GammaLib, hence it should be optimized as well as possible.</p> <p>The GNodeArray::set_value() method sets the indices for linear interpolation within an array. The index computation is done each time the set_value() method is called, hence computations are also done when the same value is used repetitively. This can be avoided by caching the last value used for computations in a mutable member. If the argument passed is identical to this mutable member, indices do not need to be updated.</p> <p>We should also study if bisection is the most rapid method for finding the correct indices. In case that the actual value is near the last value, starting from the actual indices should be much more efficient. Here, some reading is needed to find out what the most efficient search methods are.</p> <p>We also should declare the set_value() method as const. This can be done by declaring the m_inx_left and m_inx_right members mutable.</p> <p>Finally, we should implement an extensive unit test for this class, making sure that it perfectly works for linear and non-linear spacings.</p>			

#### History

##### #1 - 12/04/2012 11:08 PM - Knödlseider Jürgen

- Assigned To set to Knödlseider Jürgen
- % Done changed from 0 to 10

The set\_value method has been declared as const, and all members except of the nodes are now declared mutable.

##### #2 - 12/04/2012 11:08 PM - Knödlseider Jürgen

- Target version set to 00-07-00

##### #3 - 12/13/2012 05:12 PM - Knödlseider Jürgen

- Status changed from New to In Progress

##### #4 - 12/13/2012 05:40 PM - Knödlseider Jürgen

- % Done changed from 10 to 70

Here some reading about the search algorithm: [http://en.wikipedia.org/wiki/Binary\\_search](http://en.wikipedia.org/wiki/Binary_search)

I'm not sure that the actual bisection code can be made much faster. I did some minor update, removing basically an unnecessary if statement, as shown below.

Before:

```
int low = 0;
int high = nodes - 1;
while ((high - low) > 1) {
    int mid = (low+high) / 2;
    if (m_node[mid] > value) {
        high = mid;
    }
}
```

```
    }  
    else if (m_node[mid] <= value) {  
        low = mid;  
    }  
}  
m_inx_left = low;
```

After:

```
int low = 0;  
int high = nodes - 1;  
while ((high - low) > 1) {  
    int mid = (low+high) / 2;  
    if (m_node[mid] > value) {  
        high = mid;  
    }  
    else {  
        low = mid;  
    }  
}  
m_inx_left = low;
```

I also added caching of the last value, which should also speed up things.

I'm still need to work on the unit tests.

**#5 - 12/14/2012 05:55 PM - Knödlseider Jürgen**

- Status changed from *In Progress* to *Feedback*

- % Done changed from 70 to 100

Unit tests have been added.

**#6 - 12/21/2012 12:17 AM - Knödlseider Jürgen**

- Status changed from *Feedback* to *Closed*