Subject: Re: VALUE_LOCATE tutorial
Posted by Jeremy Bailin on Wed, 24 Jun 2009 22:20:45 GMT
View Forum Message <> Reply to Message

There's one more important application for Value_Locate that I meant to include!

(David - you can stick this after the "Using Ranges For Partitioning" subsection)

- Using Ranges In An Interpolation Scheme -

Another convenient use of this property of Value_Locate is for interpolation. Most interpolation schemes work by fitting a low-order polynomial or similar function to the points near the desired location. How do you efficiently determine which points are the "near points"? Using Value_Locate!

The simplest example is a linear interpolation between the neighbouring points: if x[i] <= array[j] <= x[i+1] (where x is strictly increasing) then the interpolated value is:

interpolated_y[j] = y[i] * (x[i+1]-array[j])/(x[i+1]-x[i]) + y[i+1] * (array[j]-x[i])/(x[i+1]-x[i])

The trick is to figure out which i to use for a given j... but that's exactly what Value_Locate does! Here is some simple code that will calculate this interpolation (I haven't taken care to handle the edge cases correctly, but see the code of the library function Interpol for more details):

```
left = Value_Locate(x, array)
right = left+1
interpolated_y = y[left] * (x[right]-array)/(x[right]-x[left]) + y
[right] * (array-x[left])/(x[right-x[left])
```

This is equivalent to Interpol(y, x, array).

-Jeremy.