Subject: Unavoidable looping?
Posted by Chris[5] on Sat, 17 May 2008 00:16:06 GMT
View Forum Message <> Reply to Message

I've seen IDL accomplish some mighty impressive tasks using vectors and arrays when other languages would use loops. As such, like a good IDL child I have been sold on the philosophy that, whenever possible, a task that avoids a loop will outperform one that does loop (though there are some posts here that argue otherwise).

However, there are some situations where loops seem unavoidable to me. A category that comes to mind are problems where elements in a list are dependent on the previous list elements. Differential equations, for example, are solved by advancing a function in tiny steps and continuously updating the size (and direction) of the next step. Such a problem has to be evaluated sequentially. Is there any spiffy IDL technique to solve such a problem without running into the efficiency problems inherent in loops? Or is IDL simply not competitive with other languages for such tasks?