Subject: Re: random integers between 0 and 1,000,000
Posted by James Kuyper on Mon, 24 Oct 2005 14:04:42 GMT
View Forum Message <> Reply to Message

Norbert Hahn wrote:
> "Peter Albert" <peter.albert@gmx.de> wrote:
>
>> Hi Mike,
>>
>> I'd guess you got the negative numbers because you defined the array
>
> Unfortunately your guess does not correspond to the docu:

I don't follow you. How does his guess fail to correspond to the docu[mentation?]?

>> weights beforehand to be of integer type. Assigning it with values of
>> type long will lead to negative numbers if the values are larger than
>> 2^15-1.
>
> fix uses 15+1 bits for integer numbers (one bit is used for sign)
> long uses 31+1 bits for integer numbers ...
> ulong uses 32 bits for unsigned numbers. There is no sign.
> long64...

Agreed. Therefore, if the code the original poster showed us were preceeded by the following statement:

weights = intarr(29)

then the expression:

weights[i] = ULONG(1000000 * RANDOMU(seed, 1))

takes a 32 bit unsigned long with a value somewhere in the range from 0 to 1000000, and converts it into a 16 bit signed int, with a range from -32768 to 32767. This is the only way that weights[i] could ever gain a negative value from that expression. For any other data types, all of the values would be positive.

> So the original problem comes from interpreting the internal bits of
> unsigned numbers. The preferable function for transforming 0...1 float
> to integer without sign would be long.

Using 'long' rather than 'ulong' wouldn't do any good if the type of "weights" itself is INT.

Note to original poster: loops are very inefficient in IDL. You'd be

better off writing:

weights[*] = LONG(1000000*RANDOMU(seed, 29))

If the length of "weights" happens to be 29, and not something longer,
you're even better off using:

weights = LONG(1000000*RANDOMU(seed,29))
