Subject: Re: Randomu Period
Posted by jargoogle on Sat, 01 May 2004 18:50:37 GMT
View Forum Message <> Reply to Message

I picked up the article which the randomu help page refers to, went through my Numerical Recipes book (also referred to), and here's basically what I think randomu is doing:

1) Based on the Lehmer generator (of the class of prime modulus multiplicative linear congruential generators), the basic algorithm is:

(i) modulus: m - a large prime number
(ii) multiplier: a - an integer in the range 2 to m-1
(iii) z(n+1) = f(z(n)) for n=1,2...
(iv) f(z) = az mod m

Most likely, RSI/IDL has followed the Park/Miller article and used

m=2147483647='7fffffff'x
a=16807

This particular implementation has a period of 2^31-1, though there are some problems with serial correlations which are compensated by a modification borrowed from or hinted by Numerical Recipes' ran1 function (See next point).

2) Bays-Durham shuffle. Basically, to break up low order correlations, instead of generating a single step in the sequence of 1(iv) above, generate N steps, filling an array. The N+1th step is then used to select one of the N previously computed values. That previously computed value becomes the output while the N+1th value takes its place in the array. Keep track of the N+1th value to seed the N+2th step.

```
if (firstTime) then begin
   a[0] = seed
   for i = 1, N-1 do a[i] = f(a[i-1])
   aNext = f(a[N-1])
endif else begin
   aNext = f(seed)
endelse
index = someConversion(aNext) ; Scales aNext to between 0 and N-1
output = a[index]
a[index] = aNext
seed = aNext
```

According to Numerical Recipes, when the above modification is made to the Lehmer generator AND Schrage's method is employed (it's not clear whether IDL uses Schrage's technique), then the output is statistically random as long as the number of calls to the routine are kept below 1 billion or so.

3) Implementation. The following is reverse engineering on my part. I actually am not sure what IDL is doing here, but it appears that in randomu (IDL 5.3 and 5.6):

```
m = '7fffffff'x
a = ?
N = 34?  36?
```

i) The seed. The seed is used to initialize the sequence, basically giving a starting z in f(z). Those who use randomu know that you only set the seed once:

```
seed = 1
num = randomu(seed,1)
```

and never again modify seed yourself. IDL takes over updating the seed at that point.

ii) The shuffle array. The array that comes back by way of seed has 36 elements. If you print the elements you find something like:

```
seed[0] = ANext
seed[1] = long(output * '7fffffff'x)
seed[i] = random long, i=2 to 35 (one of which is aNext)
```

I think that elements 2 through 35 actually store the shuffle array. IDL seems to have chosen to implement the shuffle with 34 elements. The first element tracks the seed value so that the overall sequence proceeds forward. The second element just seems to be a place holder for the output.

JG.