Subject: Re: Randomu in 5.2? Not working
Posted by landsman on Tue, 06 Apr 1999 07:00:00 GMT
View Forum Message <> Reply to Message

In article <7edl5u$7kd@netline.jpl.nasa.gov>, "Mathew Yeates" <mathew@fugue.jpl.nasa.gov> writes...
> Hi-
>
> We just went from 5.1 to 5.2 and I'm noticing different behavior.
> Here's my 5.2 session
> IDL> res=randomu(a,1) ; set seed using system clock
> IDL> print,res
> 0.415999
>
> I then exit IDL and start a new session..... and get the same result!
>
> I verified that this does not happen when I use 5.1.

Mathew,

The behavour of RANDOMU has changed three times since V4.0.1, and each new version has introduced a bug of some sort. Below is a summary of the sordid history of RANDOMU problems that I've pieced together from previous posts to comp.lang.idl-pvwave, and from information from Pat Broos. Corrections welcome.

--Wayne Landsman landsman@mpb.gsfc.nasa.gov

V4.0.1: No problems?

V5.0: RANDOMU could yield a non-random distribution if two programs using RANDOMU are interleaved. For example, in the program demo.pro given at the bottom of this message, the command demo,/breakit will show a significant excess in the distribution of random numbers between 0 and 0.03.

V5.1: A *negative* seed value must be specified if you want to preserve the same "random" sequence

```
IDL> seed = 2 & print, randomu(seed, 3)
     0.0594004     0.982075     0.358593
IDL> seed = 2 & print, randomu(seed, 3)
     0.831999     0.303037     0.506712
```

but

```
IDL>  seed = -2 & print, randomu(seed, 3)
```

```
      0.342299     0.402381     0.307838
IDL> seed = -2 & print, randomu(seed, 3)
      0.342299     0.402381     0.307838
```

This isn't necessarily a bug, but it means that RANDOMU works differently in V5.1 than in all other IDL versions.

V5.1.1 and V5.2: The seed variable is now initialized to the same value at the start of each session rather than the system clock. Thus, Monte Carlo simulations from different IDL sessions, might yield decidedly unrandom results. Perhaps more insidious, only the first call to RANDOMU is initialized inside a program. Thus, if one calls the following program test.pro multiple times, you will see that the "random" vector is simply the vector on the previous call, shifted by one.

```
PRO test
print, randomu(seed)
print, randomu(seed,6)
return
end
```

For this last problem, Pat suggests using the following wrapper program to RANDOMU to store the seed value in a common block.

```
FUNCTION random, n1, n2, n3, NEW_SEED=new_seed, _EXTRA=extra

COMMON random_seed, seed

if keyword_set(new_seed) then seed = long(new_seed)

case n_params() of
    0: return, randomu(seed,          _EXTRA=extra)
    1: return, randomu(seed,n1,       _EXTRA=extra)
    2: return, randomu(seed,n1,n2,    _EXTRA=extra)
    3: return, randomu(seed,n1,n2,n3, _EXTRA=extra)
endcase

end
```

********** **********

```
FUNCTION lib_random

return, randomu(other_seed,1)
end
```

```
PRO demo, x, BREAK_IT=break_it

; Type demo,/breakit to see the "non-random" distribution that can result in
; V5.0.   Works correctly in earlier and later IDL versions

x = fltarr(100000)

for ii = 0L, n_elements(x)-1 do begin
  x(ii) = randomu(seed,1)

  if keyword_set(break_it) then dummy = lib_random()
endfor

h = histogram( x, MIN=0.0, BIN=0.01 )
plot, h, PSYM=10
print, h
return
end
```