Subject: Re: Bizarre (?) behavior of randomu
Posted by weitkamp on Fri, 04 Jan 2008 14:44:24 GMT
View Forum Message <> Reply to Message

John,

As the two previous posters have already mentioned, it is true that the heart of the problem is that every time you run "rannummers" anew, the "seed" value is undefined at the beginning, i.e., when the loop starts at k=0. Allard de Wit is right in that in this case (a call to RANDOMU with the seed variable undefined) "the seed value will be taken from some system values (like the systime or similar)".

It is strange indeed that this leads to the repetition pattern you describe (btw, I reproduced this behavior on two different Linux systems with IDL 6.3 and 6.4). But since the behavior you observe results from an ill-defined initial state, there is probably no point in reasoning why it turns out just this way and not another.

As the two previous posters have pointed out, the solution is to carefully select the value of "seed" each time you call RANDOMU, depeding on what you want:

* To obtain a first pseudo-random number (or number sequence), input the seed variable with a well-defined long integer scalar.

* "To continue the pseudo-random number sequence, input the variable containing the saved state [i.e., the value of "seed" given back by RANDOMU] as the Seed argument in the next call to RANDOMN or RANDOMU." (quoting the IDL 6.4 online help).

* If you want to obtain an exact repetition of a number sequence obtained in this way, start the same code again, with the same seed value passed in the initial call to RANDOMU as before.

I certainly don't think that a common block or a wrapper object is needed for this -- adding a single positional parameter "seed" to the procedure does the job:

```
pro rannummers, seed
nn=100000
kmax=4
for k=0,kmax-1 do begin
      rnum=randomu(seed,nn)
      print,rnum[0:5],format='(6f10.5)'
endfor
end
```

```
IDL> seed = 478319L
IDL> rannummers, seed
% Compiled module: RANNUMMERS.
   0.55458   0.71883   0.50224   0.45372   0.04597   0.01121
   0.55780   0.51784   0.57281   0.17167   0.53495   0.70408
   0.31777   0.72242   0.59234   0.98761   0.35242   0.87670
   0.55746   0.99774   0.43512   0.06631   0.96045   0.63092
IDL> rannummers, seed
   0.29667   0.92210   0.57792   0.39947   0.91916   0.63813
   0.76520   0.73514   0.77973   0.11008   0.32726   0.94418
   0.96893   0.69209   0.11459   0.27642   0.73942   0.57509
   0.37667   0.64760   0.04895   0.66316   0.75020   0.77216
IDL> rannummers, seed
   0.10873   0.68996   0.19374   0.47691   0.84283   0.73352
   0.99465   0.27692   0.61346   0.59465   0.13965   0.38272
   0.01367   0.72033   0.74071   0.48875   0.66167   0.99068
   0.10014   0.64916   0.70723   0.11969   0.66077   0.59047
IDL> seed = 478319L
IDL> rannummers, seed
   0.55458   0.71883   0.50224   0.45372   0.04597   0.01121
   0.55780   0.51784   0.57281   0.17167   0.53495   0.70408
   0.31777   0.72242   0.59234   0.98761   0.35242   0.87670
   0.55746   0.99774   0.43512   0.06631   0.96045   0.63092
```

In the above example, the three first calls produce a true pseudo-random series of numbers (without any visible repetition scheme). The last call produces the same series as the first one because we seeded the number generator with the same value.

Cheers,
Timm

Allard de Wit wrote:

```
>  The trick is indeed to specify a random seed during the first call to
>  RANDOMU and preserve the seed value through subsequent calls to
>  RANDOMU. If you do not specify it in the first call (like in your
>  example) the seed value will be taken from some system values (like
>  the systime or similar) and RANDOMU will produce different sets of
>  random numbers with each call. This is generally undesirable,
>  if you want your results to be reproducible.
>
>  example:
>  IDL> seed=1
>  IDL> print, randomu(seed, 10)
>       0.415999    0.0919649     0.756410     0.529700     0.930436
>  0.383502     0.653919    0.0668422     0.722660     0.671149
>  IDL> print, randomu(seed, 10)
```

> 0.383416 0.631635 0.884707 0.519416 0.651519
> 0.237774 0.262453 0.762198 0.753356 0.909208
> IDL> seed=1
> IDL> print, randomu(seed, 10)
> 0.415999 0.0919649 0.756410 0.529700 0.930436
> 0.383502 0.653919 0.0668422 0.722660 0.671149
> IDL> print, randomu(seed, 10)
> 0.383416 0.631635 0.884707 0.519416 0.651519
> 0.237774 0.262453 0.762198 0.753356 0.909208
>
> Note that while I specify seed=1, it will be replaced by a LONARR(36)
> after the first call to RANDOMU.
>
> A more elegant solution without using COMMON blocks is to wrap the
> random number generator in an IDL object.
>
> with best regards,
>
> Allard
