Subject: Re: IDL random number generator
Posted by James Kuyper on Fri, 09 May 2003 14:13:26 GMT
View Forum Message <> Reply to Message

krijger@astro.uu.nl wrote:
>
> Hi,
> I know that randomn is pseudo-random, how many numbers can you
> generate before the non-randomness kicks in?
>
> Thijs Krijger

None. The non-randomness is there from the very beginning. You could make a true random number generator by running it off of the radioactive decay of atoms, or some similar hardware-based approach. However, software random number generators are absolutely deterministic, once you've set up the seed. You can set the seed form a clock setting, which means that the precise sequence of random numbers generated depends upon the precise time at which the program reads the clock. But even the very first number can be absolutely predicted from the seed value.

Every random number generator has a period, after which it starts repeating the same exact sequence. How long that period is depends upon the quality of the algorithm used. Commonly used algorithms have periods in the range of 100,000 numbers or better. Very sophisticated generators can have periods that are so long that your computer will become obsolete before the sequence repeats.

Subject: Re: IDL random number generator
Posted by tandp on Sat, 10 May 2003 22:17:44 GMT
View Forum Message <> Reply to Message

In article <3EBBB786.9C52F5F3@saicmodis.com>, James Kuyper <kuyper@saicmodis.com> wrote:

> krijger@astro.uu.nl wrote:
>>
>> Hi,
>> I know that randomn is pseudo-random, how many numbers can you
>> generate before the non-randomness kicks in?
>>
>> Thijs Krijger
>
> None. The non-randomness is there from the very beginning. You could
> make a true random number generator by running it off of the radioactive
> decay of atoms, or some similar hardware-based approach. However,
> software random number generators are absolutely deterministic, once

> you've set up the seed. You can set the seed form a clock setting, which
> means that the precise sequence of random numbers generated depends upon
> the precise time at which the program reads the clock. But even the very
^
| and the seed

> first number can be absolutely predicted from the seed value.
>
> Every random number generator has a period, after which it starts
> repeating the same exact sequence. How long that period is depends upon
> the quality of the algorithm used. Commonly used algorithms have periods
> in the range of 100,000 numbers or better. Very sophisticated generators

The best pseudorandom number generator (congruence method and seed chosen
to be the largest prime ineteger representable in a word) will have a
period equal to the seed value.

> can have periods that are so long that your computer will become
> obsolete before the sequence repeats.

Subject: Re: IDL random number generator
Posted by James Kuyper on Sun, 11 May 2003 06:30:46 GMT
View Forum Message <> Reply to Message

Mike wrote:
>
> In article <3EBBB786.9C52F5F3@saicmodis.com>, James Kuyper
> <kuyper@saicmodis.com> wrote:
>
>> krijger@astro.uu.nl wrote:
>>>
>>> Hi,
>>> I know that randomn is pseudo-random, how many numbers can you
>>> generate before the non-randomness kicks in?
>>>
>>> Thijs Krijger
>>
>> None. The non-randomness is there from the very beginning. You could
>> make a true random number generator by running it off of the radioactive
>> decay of atoms, or some similar hardware-based approach. However,
>> software random number generators are absolutely deterministic, once
>> you've set up the seed. You can set the seed form a clock setting, which
>> means that the precise sequence of random numbers generated depends upon
>> the precise time at which the program reads the clock. But even the very
>
^
>
| and the seed
>

>> first number can be absolutely predicted from the seed value.
>>
>> Every random number generator has a period, after which it starts
>> repeating the same exact sequence. How long that period is depends upon
>> the quality of the algorithm used. Commonly used algorithms have periods
>> in the range of 100,000 numbers or better. Very sophisticated generators
>
> The best pseudorandom number generator (congruence method and seed chosen
> to be the largest prime ineteger representable in a word) will have a
> period equal to the seed value.

The seed value? Thus, if the seed value is 1, it repeats indefinitely? I
think you're mistaken on that.

My understanding is that, if care is taken it choosing the parameters of
the algorithm, the period is precisely the best that it could be,
independent of the seed value chosen. It's 2^n-1, where n is the number
of bits in the seed. However, there are other algorithms that set up an
initial state which is much larger than the seed itself, often
represented as a array of integers. Ideally, such generators could have
a period as long as 2^(n*m), where m is the number of n-bit integers in
the array. Let 'm' be as small as 128, and you've got a period that
could never possibly be measured before the machines they run on become
obsolete.

Subject: Re: IDL random number generator
Posted by krijger on Mon, 12 May 2003 09:29:54 GMT
View Forum Message <> Reply to Message

James Kuyper <kuyper@saicmodis.com> wrote in message
news:<3EBBB786.9C52F5F3@saicmodis.com>...
> krijger@astro.uu.nl wrote:
>>
>> Hi,
>> I know that randomn is pseudo-random, how many numbers can you
>> generate before the non-randomness kicks in?
>>
>> Thijs Krijger
>
> None. The non-randomness is there from the very beginning. You could
> make a true random number generator by running it off of the radioactive
> decay of atoms, or some similar hardware-based approach. However,
> software random number generators are absolutely deterministic, once
> you've set up the seed. You can set the seed form a clock setting, which
> means that the precise sequence of random numbers generated depends upon
> the precise time at which the program reads the clock. But even the very
> first number can be absolutely predicted from the seed value.

>
> Every random number generator has a period, after which it starts
> repeating the same exact sequence. How long that period is depends upon
> the quality of the algorithm used. Commonly used algorithms have periods
> in the range of 100,000 numbers or better. Very sophisticated generators
> can have periods that are so long that your computer will become
> obsolete before the sequence repeats.

So, if in IDL I use the data=randomn(seed,N), then how big can N be
(and I can make the claim that the numbers are still random (compared
to each other))?

Thijs Krijger

Subject: Re: IDL random number generator
Posted by James Kuyper on Mon, 12 May 2003 15:15:34 GMT
View Forum Message <> Reply to Message

krijger@astro.uu.nl wrote:
>
> James Kuyper <kuyper@saicmodis.com> wrote in message
news:<3EBBB786.9C52F5F3@saicmodis.com>...
>> krijger@astro.uu.nl wrote:
>>>
>>> Hi,
>>> I know that randomn is pseudo-random, how many numbers can you
>>> generate before the non-randomness kicks in?
>>>
>>> Thijs Krijger
>>
>> None. The non-randomness is there from the very beginning. You could
>> make a true random number generator by running it off of the radioactive
>> decay of atoms, or some similar hardware-based approach. However,
>> software random number generators are absolutely deterministic, once
>> you've set up the seed. You can set the seed form a clock setting, which
>> means that the precise sequence of random numbers generated depends upon
>> the precise time at which the program reads the clock. But even the very
>> first number can be absolutely predicted from the seed value.
>>
>> Every random number generator has a period, after which it starts
>> repeating the same exact sequence. How long that period is depends upon
>> the quality of the algorithm used. Commonly used algorithms have periods
>> in the range of 100,000 numbers or better. Very sophisticated generators
>> can have periods that are so long that your computer will become
>> obsolete before the sequence repeats.
>
> So, if in IDL I use the data=randomn(seed,N), then how big can N be

> (and I can make the claim that the numbers are still random (compared
> to each other))?

The numbers will never be truly random. They will always be pseudo-random, no matter how big N is. There's no special point at which the cease to be pseudo-random. That's a meaningless question, like asking how many ducks are equivalent to one psuedonym. A meaningful question to ask is how long will it be before the pseudo-random sequence repeats. I don't know the answer to that one for this particular generator. According to the online help:

"The random number generator is taken from: "Random Number Generators: Good Ones are Hard to Find", Park and Miller, Communications of the ACM, Oct 1988, Vol 31, No. 10, p. 1192. To remove low-order serial correlations, a Bays-Durham shuffle is added, resulting in a random number generator similar to ran1() in Section 7.1 of Numerical Recipes in C: The Art of Scientific Computing (Second Edition), published by Cambridge University Press."

If it's important to you, then you should probably track down those references and read them.

However, if you call randomu(seed), where 'seed' has not been defined, it will create the state array in a variable named 'seed'. That state array is a 36-element array of long integers. In principle, if their algorithm uses that array efficiently, the repeat period could be as long as 2**(36*32)= 3.2E798, which should be long enough for most purposes. :-) Let's put it this way. If you encoded the pseudo-random numbers on the energy levels of atoms, the entire visible universe isn't large enough (by many orders of magnitude) to record the entire sequence.

Subject: Re: IDL random number generator
Posted by tandp on Mon, 12 May 2003 17:12:54 GMT

View Forum Message <> Reply to Message

In article <3EBDEE16.F2E96EAD@saicmodis.com>, James Kuyper <kuyper@saicmodis.com> wrote:

> Mike wrote:
>>
>> In article <3EBBB786.9C52F5F3@saicmodis.com>, James Kuyper
>> <kuyper@saicmodis.com> wrote:
>>
>>> krijger@astro.uu.nl wrote:
>>>>
>>>> Hi,

>>>> I know that randomn is pseudo-random, how many numbers can you
>>>> generate before the non-randomness kicks in?
>>>>
>>>> Thijs Krijger
>>>
>>> None. The non-randomness is there from the very beginning. You could
>>> make a true random number generator by running it off of the radioactive
>>> decay of atoms, or some similar hardware-based approach. However,
>>> software random number generators are absolutely deterministic, once
>>> you've set up the seed. You can set the seed form a clock setting, which
>>> means that the precise sequence of random numbers generated depends upon
>>> the precise time at which the program reads the clock. But even the very
>> ^
>> | and the seed
>>
>>> first number can be absolutely predicted from the seed value.
>>>
>>> Every random number generator has a period, after which it starts
>>> repeating the same exact sequence. How long that period is depends upon
>>> the quality of the algorithm used. Commonly used algorithms have periods
>>> in the range of 100,000 numbers or better. Very sophisticated generators
>>
>> The best pseudorandom number generator (congruence method and seed chosen
>> to be the largest prime ineteger representable in a word) will have a
>> period equal to the seed value.
>
> The seed value? Thus, if the seed value is 1, it repeats indefinitely? I
> think you're mistaken on that.

A faulty memory led me to describe teh seed as needing to be the largets
prime number available on the given system. Actually it is the modulus
value that should be chosen this way. The choice of seed, multiplier and
modulus numbers is discussed a bit in Numerical Recipes which refers to
Knuth's book. If you need reliable details readup on it there. Don;t trust
the internet.

>
> My understanding is that, if care is taken it choosing the parameters of
> the algorithm, the period is precisely the best that it could be,
> independent of the seed value chosen. It's 2^n-1, where n is the number
> of bits in the seed. However, there are other algorithms that set up an
> initial state which is much larger than the seed itself, often
> represented as a array of integers. Ideally, such generators could have
> a period as long as 2^(n*m), where m is the number of n-bit integers in
> the array. Let 'm' be as small as 128, and you've got a period that
> could never possibly be measured before the machines they run on become
> obsolete.

Subject: Re: IDL random number generator
Posted by Matt Feinstein on Mon, 12 May 2003 18:06:37 GMT
View Forum Message <> Reply to Message

On 12 May 2003 02:29:54 -0700, krijger@astro.uu.nl wrote:

> James Kuyper <kuyper@saicmodis.com> wrote in message
news:<3EBBB786.9C52F5F3@saicmodis.com>...
>> krijger@astro.uu.nl wrote:
>>>
>>> Hi,
>>> I know that randomn is pseudo-random, how many numbers can you
>>> generate before the non-randomness kicks in?
>>>
>>> Thijs Krijger
>>
>> None. The non-randomness is there from the very beginning. You could
>> make a true random number generator by running it off of the radioactive
>> decay of atoms, or some similar hardware-based approach. However,
>> software random number generators are absolutely deterministic, once
>> you've set up the seed. You can set the seed form a clock setting, which
>> means that the precise sequence of random numbers generated depends upon
>> the precise time at which the program reads the clock. But even the very
>> first number can be absolutely predicted from the seed value.
>>
>> Every random number generator has a period, after which it starts
>> repeating the same exact sequence. How long that period is depends upon
>> the quality of the algorithm used. Commonly used algorithms have periods
>> in the range of 100,000 numbers or better. Very sophisticated generators
>> can have periods that are so long that your computer will become
>> obsolete before the sequence repeats.
>
> So, if in IDL I use the data=randomn(seed,N), then how big can N be
> (and I can make the claim that the numbers are still random (compared
> to each other))?
>
> Thijs Krijger

The period of a reasonable random number generator should be at least
2^32 (around four billion) and could be larger-- you need to know the
details of the algorithm to be sure. Table 1 in Knuth's chapter on
random numbers has one with an effective modulus of 2^1376, which is a
pretty big number by most standards. It's worth pointing out that the
subject of random number generators has been worked over rather
heavily since the 60's, when some notorious algorithms produced
numbers that weren't very random. Modern algorithms must pass a
battery of stringent tests for non-correlation and non-periodicity in
all low dimensions. The moral is that you can't select a random number
generator at random.

Matt Feinstein

--
The Law of Polarity: The probability of wiring a battery with
the correct polarity is (1/2)^N, where N is the number of times
you try to connect it.