TFY4235/FYS8904

Solution problemset 7 Spring 2015

NTNU

Institutt for fysikk

Problem 1.

The programs that generate the sequence of random numbers are

Listing 1: r16807.f

```
      program r16807
      rinv=0.5/(2.**31-1)
      ibm=4711333
      do i=1,1000
      ibm=ibm*16807
      enddo
      rold=0.
      rnew=0.
      do i=1,10000
100   rold=rnew
      ibm=ibm*16807
      rnew=float(ibm)*rinv+0.5
      if(rold.lt.0.001.and.rnew.lt.0.001) then
      write(*,*) rold,rnew
      else
      goto 100
      endif
      enddo
      end
```

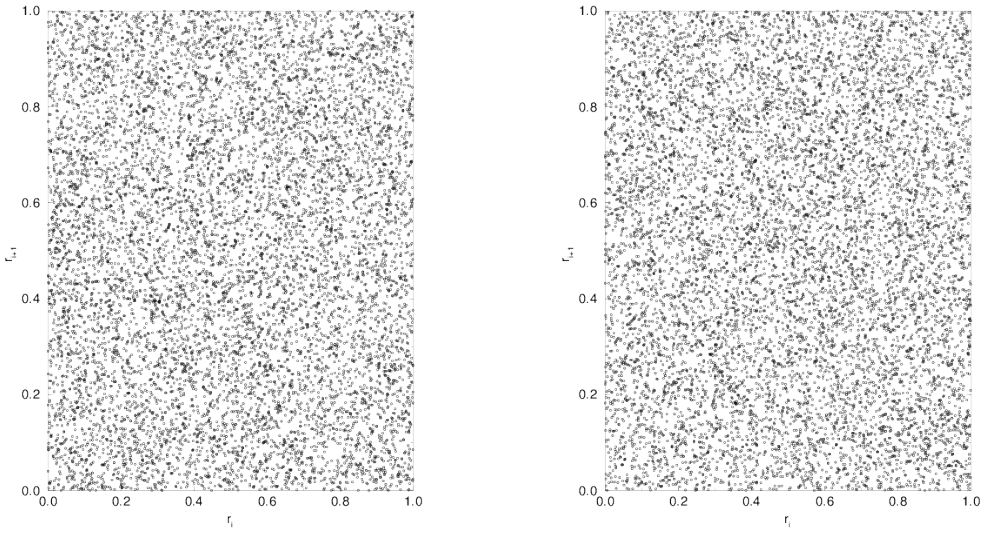
and

Listing 2: rinnb.f

```
      program rinnb
      rold=0.
      rnew=0.
      do i=1,10000
100   rold=rnew
      rnew=r_addran()
      if(rold.lt.0.001.and.rnew.lt.0.001) then
      write(*,*) rold,rnew
      else
      goto 100
      endif
      enddo
      end
```

Equivalent implementations in C are given in the *Code* subdirectory.

Figure 1a shows 10 000 random numbers generated with the "16807" algorithm. Similar results for the built-in generator is shown in figure 1b.



(a) 16807 (b) Built-in

Figure 1: 10000 random numbers generated with either 16807 or the built-in algorithm.

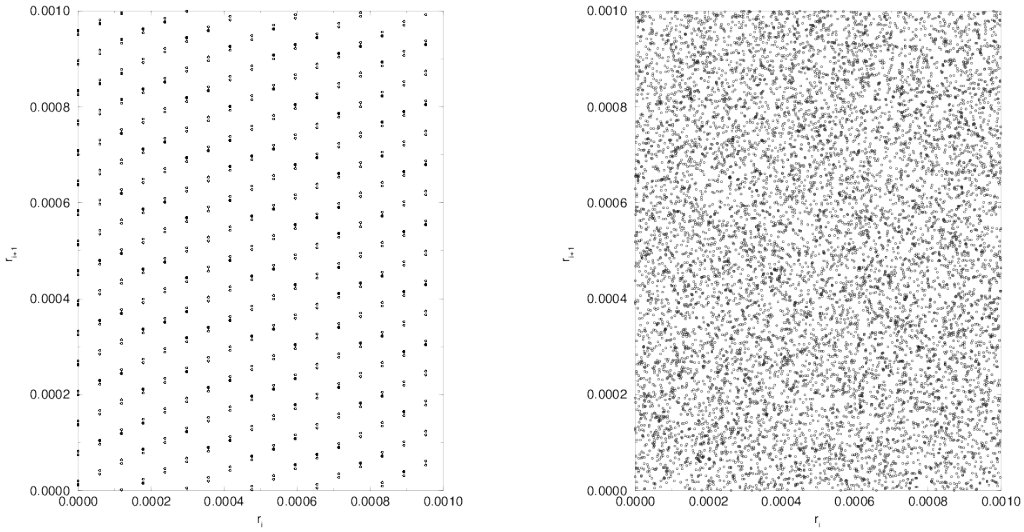
Figure 2a shows result of 10 000 numbers generated with the "16807" generator but limited to the square $(0,0)$, $(0,0.001)$, $(0.001,0)$, $(0.001,0.001)$.
The is not so nice.
The result from the build-in generator is shown in figure 2b. It is absolutely ok.
Should we conclude that the "16807" generator is not useable? No, the only conclusion we can make is that it is not good for this particular use. Any generator should be tested *in the context it is to be used.*

Problem 2.

The following program uses the Box-Müller algorithm to generate random numbers following a gaussian with unit variance and zero mean. The programs groups the numbers into batches of size `nnum`$= N$. It identifies the largest number in each batch and then averages over these numbers to produce the average of the largest number among N numbers, $\langle x_{\max}\rangle$.

Listing 3: bboxm.f

```
1       program bboxm
2 c
3 c Box-Mueller algorithm for generating gaussian random numbers
4 c
5       parameter(msamp=10000,nnum=50000)
6 c
7       ibm=1955
8       rinv=0.5/(2.**31-1.)
9       do i=1,1000
```



(a) 16807 (b) Built-in

Figure 2: 10000 random numbers generated with either 16807 or the built-in algorithm limited to the square $(0, 0)$, $(0, 0.001)$, $(0.001, 0)$, $(0.001, 0.001)$.

```
10          ibm=ibm*16807
11          enddo
12    c
13          pi2=8*atan(1.)
14    c
15          avemax=0.
16    c
17          do isamp=1,msamp
18    c
19          xmax=0.
20    c
21          do i=1,nnum/2
22          ibm=ibm*16807
23          y1=rinv*float(ibm)+0.5
24          ibm=ibm*16807
25          y2=rinv*float(ibm)+0.5
26          x1=sqrt(-2.*log(y1))*cos(pi2*y2)
27          x2=sqrt(-2.*log(y1))*sin(pi2*y2)
28          xmax=max(xmax,x1)
29          xmax=max(xmax,x2)
30          enddo
31    c
32          avemax=avemax+xmax
33    c
34          enddo
```

```
35 c
36       avemax=avemax/msamp
37       print *,nnum,avemax
38       end
```

The task was to find $\langle x_{\max}\rangle$ as a function of N. It so happens that this is a classical problem of *extreme statistics*. If the random numbers x are distributed according to a cumulative probability $P(x) = \int_{-\infty}^{x} p(x')\,dx'$, then we have that

$$P(\langle x_{\max}\rangle) = 1 - \frac{1}{N}\ . \tag{1}$$

The cumulative probability of a Gaussian is

$$P(x) = \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)\right]\ . \tag{2}$$

For large x, this expression is

$$P(x \to \infty) = 1 - \frac{e^{-x^2/2\sigma^2}}{x\sqrt{2\pi/\sigma^2}} \tag{3}$$

Combining Eqs. (1) and (3), we find

$$\langle x_{\max}\rangle = \sqrt{2\sigma^2 \ln N}\ . \tag{4}$$

We show the result of our simulation in the figure below. The straight line in the plot indicates that we find exactly the behavior expected from Eq. (4). I let N vary between 100 and 100 000, averaging each run over 10 000 samples.

I was in the lucky position here to know the functional form (4). Trying to fit $\langle x_{\max}\rangle$ vs. N without knowing this, is not so easy — but a typical task in computational physics.

The average height of people (from Wikipedia) is around 175 cm with a (guestimated) variance of around 10 cm. There are 6×10^9 people around. Plugging this into Eq. (4) gives $\langle x_{\max}\rangle = 175\ \mathrm{cm}\ + 10\ \mathrm{cm}\ \sqrt{2\ln 6 \times 10^9} = 242$ cm. This is right between the tallest and the second tallest person alive today. Hence, we may conclude that even the extremely tall follow the same Gaussian distribution as everybody else. Their height is in fact not unexpected. If they had not been that tall, somebody else would.

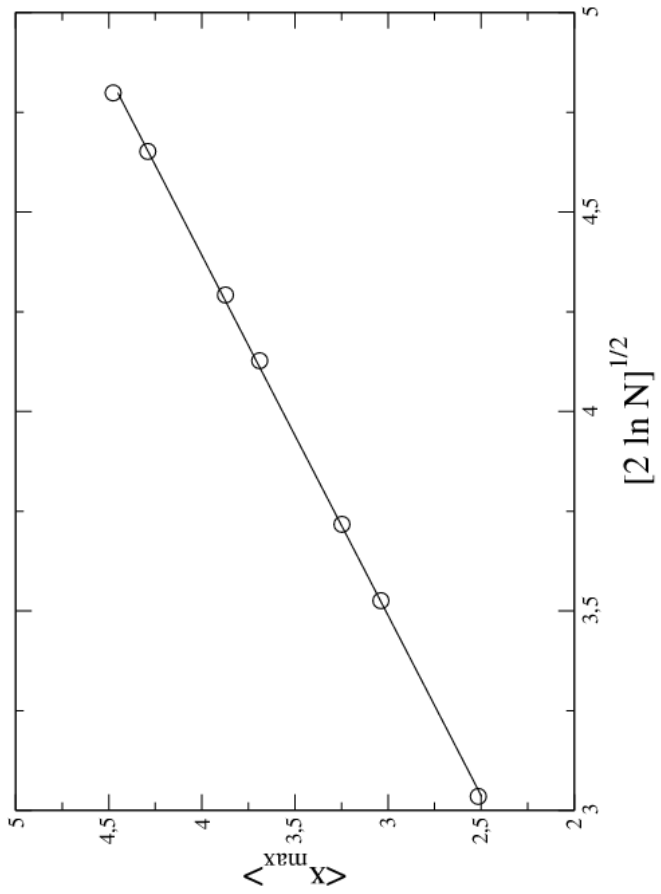


Figure 3: Plot of x_{max}