

# Computational Physics - Exercise 9

Maurice Donner

Lukas Häffner

June 25, 2019

# 1 Random Numbers – Rolling Dice

We write a simple portable random number generator, using linear congruences:

$$I_{j+1} = aI_j + c \pmod{m} \quad (1)$$

For that, we create an initiation function, that creates a global Variable.

```
def init(initVal):  
    global rand  
    rand = initVal
```

This variable will now be rewritten over and over again by a number generated by (1):

```
def generate_random(a,m,c,initVal):  
    global rand  
    rand = (a*rand+c)%m  
    return rand
```

This function produces homogeneously distributed numbers between 0 and m-1. It can be normalized by  $r_i = I_j/(m - 1)$ , to get homogeneously distributed real numbers between 0 and 1. We try it for example for  $a = 106, m = 6075, c = 1283$ :

```
Generating random number... a = 106, m = 6075, c = 1283  
Random number sequence:  
0: 1389  
1: 2717  
2: 3760  
3: 4968  
4: 5441  
5: 904  
6: 5982  
7: 3575  
8: 3583  
9: 4431  
  
Normalizing...  
0: 0.22867961804412248  
1: 0.4473164306881791  
2: 0.6190319394138953  
3: 0.8179124135660191  
4: 0.8957853144550544  
5: 0.14883108330589398  
6: 0.9848534738228515  
7: 0.5885742509054989  
8: 0.5898913401382944  
9: 0.7295027988146197
```

A simple way to check 'by eye' that the random number generator really does produce homogeneously distributed numbers, is to create two sequences with different initial values  $I_0$  and  $J_0$ . Let  $\mathbf{r}$  be a random normalized number sequence generated with  $I_0 = 1$  and  $\mathbf{s}$  generated with  $J_0 = 2$ . We plot all pairs  $(r_i, s_i)$  of generated numbers in a number plane:

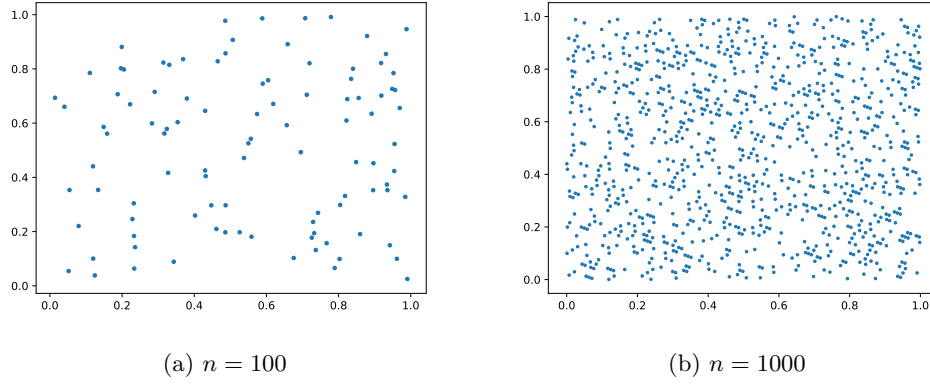


Figure 1: Creating a set of random numbers

Since the eye is quite sensitive to see a good distribution. For  $m = 100$  there is no visible pattern. Taking 1000 samples, will result in numbers, that appear random at first, but this time there are small patterns forming, that look like little lines all pointing towards the same direction. To see the deterministic character of this random number sequence, we plot  $I_{j+1}$  against  $I_j$  for  $I_0 = 1$ , and  $n = 1000$ :

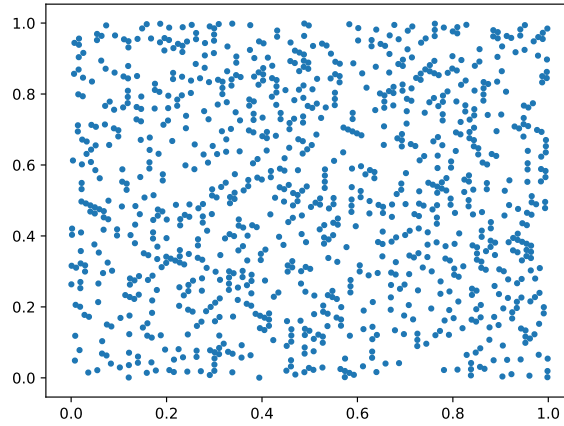


Figure 2: Deterministic Character of random number sequence

Again, overall the generator appears random, yet small patterns can still be seen. This effect can be mitigated, by choosing larger values for  $a$ ,  $m$ , and  $c$ . For example, for  $a = 1060$ ,  $m = 60751$ ,  $c = 12835$ , the small patterns, formerly visible in the deterministic character of our sequence vanishes:

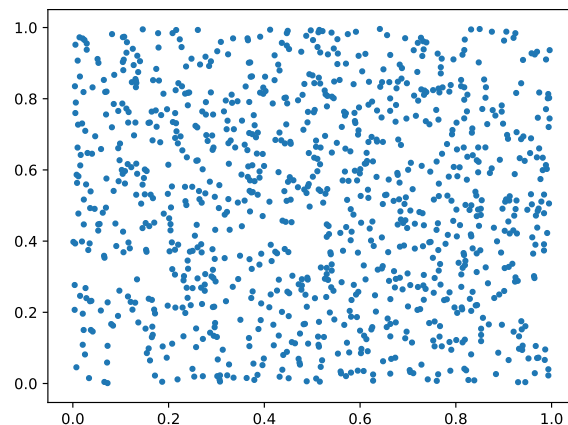


Figure 3: Choosing larger values  $\rightarrow$  The plot becomes "more random"

Next we create an experiment where the generator rolls dices. For that, we normalize our generator to 6, instead of 1 (This will be done in an extra `.py` file to avoid confusion). In this configuration, our random number generator generates numbers between 0 and 6. Because those numbers are equally distributed, all we need to do is an `int`-conversion. With that, python completely neglects any digits behind the comma. And our random numbers will hence be integer values between 0 and 5. After adding up packets of 10 rolls each, we get a distribution:

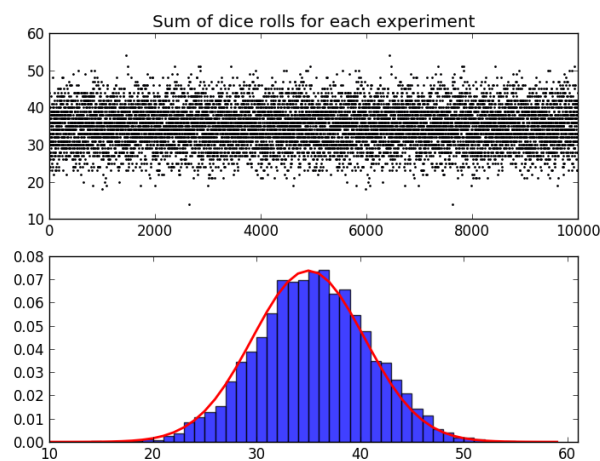


Figure 4: Adding 10 Dice rolls – Distribution

The Dice rolls follow a Gaussian Distribution according to the Central Limit Theorem.