Scientific Computing Lab

Kevin Francis

April 1, 2021

Equations and Numerical Methods

In this lab we will be comparing a linear congruent random number generator to the built in random() function within python. The linear congruent random number generator (LCRNG) is below:

$$r_{i+1} = (ar_i + c) mod M$$

This equation says the the next term is the remainder of $\frac{ar_i+c}{M}$ where the initial values are provided by the user. To test the built in random number generator we find the moment given by the equation below:

$$\frac{1}{N}\sum_{i=1}^{N} x_i^k \tag{1}$$

The moment should be approximately $\frac{1}{k+1}$, this will look for a uniform distribution. Another test that will be done is

$$\sqrt{N}\left|\frac{1}{N}\sum_{i=1}^{N} x_i^k - \frac{1}{k+1}\right| \tag{2}$$

Part 1

The LCRNG was used with $(a, c, M, r_i) = (57, 1, 256, 10)$ to produce the the a period of 256.

Code

```
def lcg(r,a,c,M):
    r = (a*r+c)%M
    return r
def lct_test_period(ri,a,c,M):
    r = lcg(ri,a,c,M)
    l = []
    n = 0
    m = 1
    while r != ri:
        r = lcg(r,a,c,M)
        l.append(r)
        if len(l) >= n+1:
            xi = l[n]
            n += 2
        if len(l) >= m+1:
            yi = l[m]
            m += 2
            figure(1)
            title('Plot of random Xi vs Yi')
            scatter(xi, yi)
    print "The period is: " + str(len(l) + 1)
```

Part 4

Using equation (1) from above the moment for the built in random number generator was found. Below is the code and output for $k = 1, 2$:

Code

```python
def moment(k, N):
    s = 0
    for i in range(1,N+1):
        s += random()**k
        i += 1
    return float(s/N)
```

Output

```
#Enter kth value: 1
#Enter number or random values: 100
#The moment is: 0.483706656003
#Enter kth value: 2
#Enter number or random values: 100
#The moment is: 0.29886473275
```

Part 5

Again the built in generator is tested, this time using equation (2) for $k = 1, 3, 5, 7, 9$ and $N = 100, 10000, 100000$. The code and output are below. Note that this function also generates a plot of the data.

Code

```
def moment_test():
    for k in range(1,10,2):
        A = logspace(2,6,5,endpoint=True,base=10.0,dtype=int)
        B = zeros(len(A))
        for i in range(len(A)):
            N = A[i]
            B[i] = (abs(moment(k,N) - (1./(k + 1))))
        loglog(A,B,'-o',label='k = ' + str(k))
        print 'For k=' + str(k) + ' and N=' + str(N) + ' the moment
        is: ' + str(sqrt(N) * B[i])
    legend(loc='upper right')
    xlabel('log(N)')
    ylabel('Logarithimc Moment')
    title('Log-Log Plot of N vs. Moment')
    show()
```

Output

```
#For k=1 and N=100 the moment is: 0.172173695813
#For k=1 and N=1000 the moment is: 0.244804103983
#For k=1 and N=10000 the moment is: 0.126801801512
#For k=1 and N=100000 the moment is: 0.280455526914
#For k=1 and N=1000000 the moment is: 0.0379595745481
#For k=3 and N=100 the moment is: 0.0145407578826
#For k=3 and N=1000 the moment is: 0.35550769236
#For k=3 and N=10000 the moment is: 0.22544470438
#For k=3 and N=100000 the moment is: 0.566764371839
#For k=3 and N=1000000 the moment is: 0.222209612911
#For k=5 and N=100 the moment is: 0.149459058773
#For k=5 and N=1000 the moment is: 0.00612313363216
#For k=5 and N=10000 the moment is: 0.272787889419
#For k=5 and N=100000 the moment is: 0.0661647396608
#For k=5 and N=1000000 the moment is: 0.0366521550892
#For k=7 and N=100 the moment is: 0.0135674950122
#For k=7 and N=1000 the moment is: 0.0022796610533
#For k=7 and N=10000 the moment is: 0.176992812933
#For k=7 and N=100000 the moment is: 0.188511622279
#For k=7 and N=1000000 the moment is: 0.128399677635
#For k=9 and N=100 the moment is: 0.0994364484568
#For k=9 and N=1000 the moment is: 0.00882861284156
#For k=9 and N=10000 the moment is: 0.0361793510962
#For k=9 and N=100000 the moment is: 0.127785488699
#For k=9 and N=1000000 the moment is: 0.122669017877
```

Around $x = 6$ is where the series begins to lose accuracy. The larger x gets the worse it becomes.

Visualization

Below are three images, each plotted within the corresponding functions above. Note that figure one is using the values of the variables in part 1.

Figure 1:

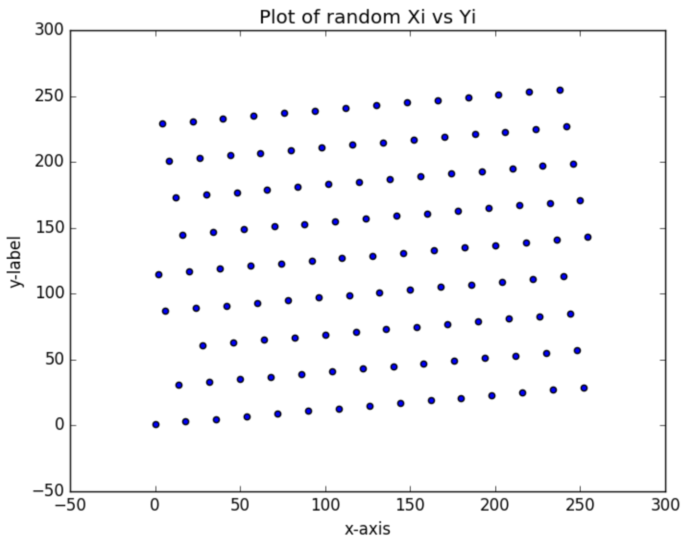


Figure 2:

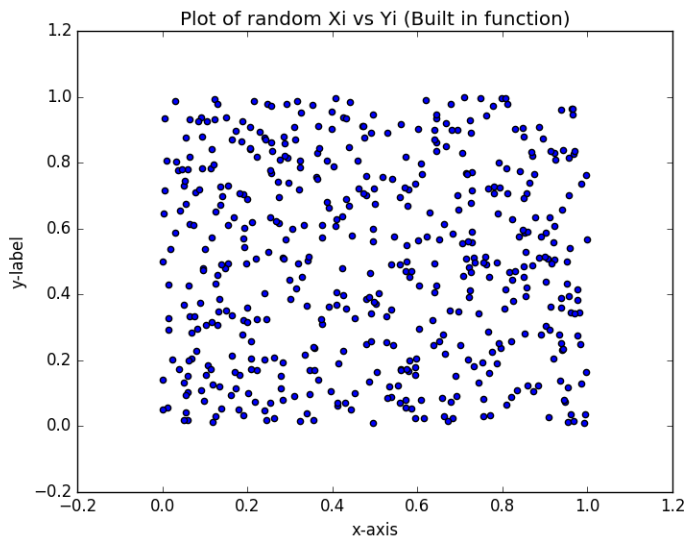
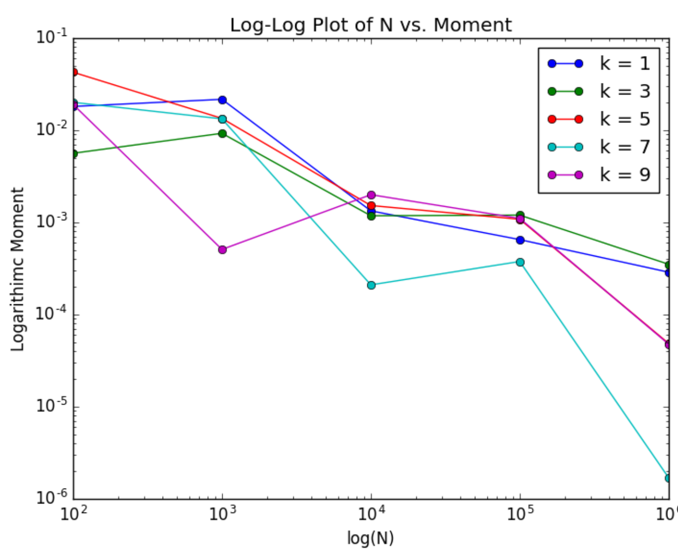


Figure 3:



.

Discussion

In this lab the LCRNG was compared to the built in function generator of python. From figure 1 above it can be seen that if certain numbers are picked the LCRNG will have a pattern. However the built in generator is more random. The built in function also have a uniform distribution, this can be seen in the output for part 4. The moment is approximately $\frac{1}{k+1}$ for the two k values tested. This lab was helpful with comparing a random number generator (LCRNG) that is commonly used to one supplied by python.