Solutions problem set 1: AWK

Random number generators

Awk has a built-in random number generator called rand() which we will test in this exercise.

a. Generate 100000 random numbers and save the output. (What happens if you add "| sort" ?)

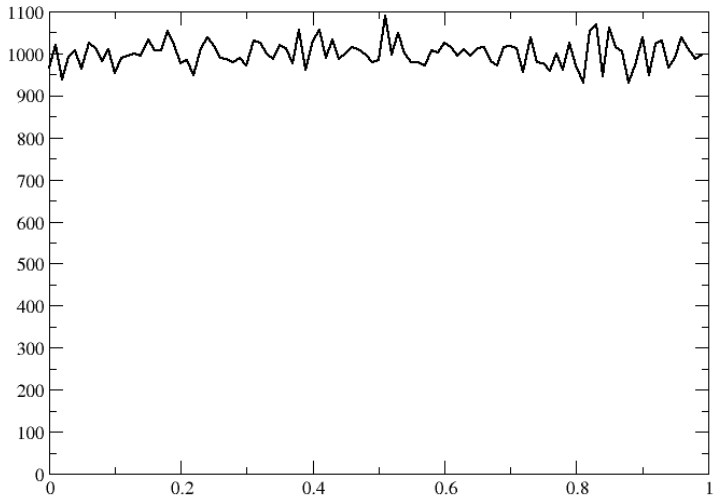
```
awk 'BEGIN {for(i=0;i<100000;i++) print rand() }' t1 > r_num
"|sort" will sort the random numbers, t1 can be an arbitrary file
```

b. Write an awk-program which calculates the average value, the variance and the error of the mean of the data.
Hint: VAR=($\langle X^2 \rangle - \langle X \rangle^2$), ERR=(sqrt(VAR/n)).

```
        { average+=$1;  average_square+=$1*$1;}
END     { average/=NR; average_square/=NR;
          VAR=average_square-average*average;
          ERR= sqrt(VAR/NR);
          print average, VAR, ERR; }
```

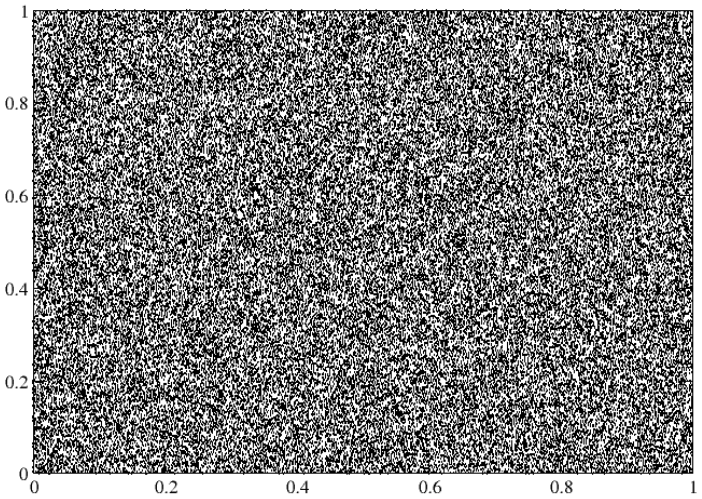
c. Uniformity Test: Create a histogram of the data with 100 bins and display it using xmgrace or gnuplot (Hint: Use the int() function)

```
        { histo[int($1*100)]++}
END     { for(i=0;i<100;i++)
                print i/100,histo[i]}
```



d. Screen Pixel Test: Save 2*100000 random numbers in two columns. Visualize the result.

```
awk '{BEGIN {for(i=0;i<100000;i++) print rand(),rand()}' t1 > r_num2
```

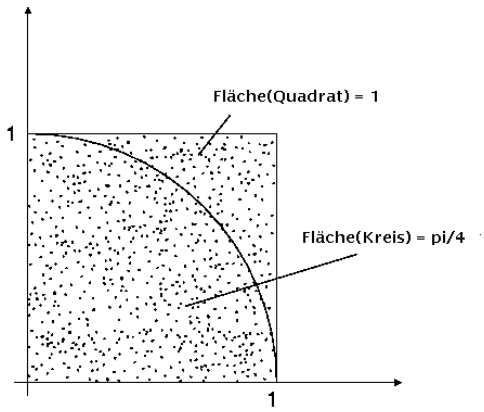


e. Program your own linear congruental random number generator:
Hints: $X_{n+1} = (a*X_n+c) \% m$
Numerical Recipes in C advocates:
a=1664525, c=1013904223, $m=2^{32}$.

```
BEGIN {x=1;a=1664525;c=1013904223;m=2^32;
       for(i=0;i<100000;i++) {x=(a*x+c)%m; print x/m}}
```

Your first Monte Carlo Simulation

Write a Monte Carlo Program with awk which calculates π.




```
awk '$1*$1+$2*$2<1 {F_Kreis ++} END
{print 4*F_Kreis}' r_num2
```

Homework: Extend 1b to calculate the auto-correlation as well.

```
# Program to determine auto-correlation function of a data-set
        { a [NR-1] = $1
          average = average + $1
          average_square = average_square + $1*$1}

END     { mean_sqr = (average*average) / (NR*NR)
          sqr_mean = average_square / NR

          for (ctr1 = 0; ctr1 < 10; ctr1++)
                { product = 0
                for (ctr2=0; ctr2 < NR - ctr1; ctr2++)
                        product = product + (a [ctr2+ctr1] * a [ctr2])
                        corr_product = product / ctr2
          print ctr1, (corr_product - mean_sqr) / (sqr_mean - mean_sqr)
                }
        }
```

Picture credits:
http://www.tuhh.de/rzt/tuinfo/programmentwicklung/parallel/Junglas/mpi-1-kurs/node9.html