Week 10
21.3 Random Numbers and Monte Carlo Simulation

- The procedure of generating variables from a given probability distributions is known as Monte Carlo sampling.

Random number generators

- Random number generators must have several other important characteristics if they are to be used efficiently within computer Simulations.

1. computationally fast,
2. requires little computer memory,
3. is sufficiently spread out,
4. is identically replicable and
5. has a long cycle.

Lineêr kongruensiële kansgetal generator / Lin congruential random number generator

Every random number R_i, $i \leq n$, in the random number series of n random numbers , are calculated with the formula:

$$R_i = \frac{x_i}{m} \qquad \text{where} \qquad x_i = ax_{i-1} + c \mod m.$$

- a is the constant multiplier,
- c is the increment,
- m is the modulus,
- x_0 is seed number.

The initial value of x_0 is called the seed, a is the constant multiplier, c is the increment, and m is the modulus. These four variables are called the parameters of the generator. Using this relation, the value of x_{i+1} equals the remainder from the division of $ax_i + c$ by m. The random number between 0 and 1 is then generated using the equation

$$R_i = \frac{x_i}{m} \qquad (i = 1, 2, 3, \ldots)$$

For example, if $x_0 = 35$, $a = 13$, $c = 65$, and $m = 100$, the algorithm works as follows:

Iteration 0 Set $x_0 = 35$, $a = 13$, $c = 65$, and $m = 100$.

Iteration 1 Compute

$$\begin{aligned} x_1 &= (ax_0 + c) \text{ modulo } m \\ &= [13(35) + 65] \text{ modulo } 100 \\ &= 20 \end{aligned}$$

Deliver

$$\begin{aligned} R_1 &= \frac{x_1}{m} \\ &= \frac{20}{100} \\ &= 0.20 \end{aligned}$$

Iteration 2 Compute

$$\begin{aligned} x_2 &= (ax_1 + c) \text{ modulo } m \\ &= [13(20) + 65] \text{ modulo } 100 \\ &= 25 \end{aligned}$$

Deliver

$$\begin{aligned} R_2 &= \frac{x_2}{m} \\ &= \frac{25}{100} \\ &= 0.25 \end{aligned}$$

Iteration 3 Compute

$$x_3 = (ax_2 + c) \text{ modulo } m$$

$$\vdots$$

and so on.

Inverse transform

1. Given a random variable X with probability density function $f(x)$ - find the cumulative probability density function
$$F(x) = \int_{-\infty}^{x} f(t)\, dt,$$
2. Create a random number R,
3. Set $F(x) = R$ and solve for x with $x = F^{-1}(R)$. The value of x then is a random number of the distribution with probability density function $f(x)$.
4. $x_i = F^{-1}(R_i)$ then is a random variable generator or process generator.

Continuous probability distributions of importance

There are three continuous probability dsitributions of importance in this course.

- the uniform distribution,
- the exponential distribution and the
- the normal distribution.

Exponential distribution

$$t \sim \text{Expon}(\lambda)$$

Waarskynlikheidsdigtheidsfunksie:

Probability density function:

$$f(t) = \begin{cases} 0 & t < 0 \\ \lambda e^{-\lambda x} & t \geq 0, \lambda > 0 \end{cases}$$

Kumulatiewe waarskynlikheidsdigtheidfunksie:

Cumulative probability density function:

$$F(t) = \begin{cases} 0 & t < 0 \\ 1 - e^{-\lambda t} & t \geq 0, \lambda > 0 \end{cases}$$

Inverse transform:

Inverse transform:

$$t_i = -\frac{1}{\lambda} \ln(1 - R_i)$$

LINGO

```
########Exponential distribution

# First create set of random numbers - I've named it Rs

m=2^31-1
a=7^5
x=987654321
c=100000

Rs=c()

#num is the number of random numbers to create
num=10
for(i in 1:num){
  x=(a*x+c)%%m
  R=x/m
  Rs[i]=R
}

Rs

# create the set of num-number of random variables based on the elements of Rs
# Say the random variables are exponential distributed with lambda=1/5 customers per minute
# in other words the mean time between arrivals is 5 minutes

get_Exponvar=function(R)return(-(1/lambda)*log(1-R))

lambda=1/5 #arrivals/minutes

MyIATS=sapply(Rs,get_Exponvar)

MyIATS

#Check for yourself - the larger the number random variables the closer the mean will get
# to 1/lambda
mean(MyIATS)

#The same set of interarrival times could also be generated by the qexp function of R

MyIATs_Alt=qexp(Rs,lambda)
MyIATs_Alt
```

Uniform distribution

$$t \sim \text{Uniform}\,(a, b)$$

Waarskynlikheidsdigtheidsfunksie: Probability density function:

$$f(t) = \begin{cases} 0 & t < a, t > b \\ \frac{1}{b-a} & a \leq t \leq b \end{cases}$$

Kumulatiewe waarskynlikheidsdigtheidfunksie: Cumulative probability density function:

$$F(t) = \begin{cases} 0 & t < a \\ \frac{x-a}{b-a} & a \leq t \leq b \\ 1 & t > b \end{cases}$$

Inverse transform: Inverse transform:

```
########Uniform distribution

# First create set of random numbers - I've named it Rs again

m=2^31-1
a=7^5
x=654321987
c=100000

Rs=c()

#num is the number of random numbers to create
num=20
for(i in 1:num){
  x=(a*x+c)%%m
  R=x/m
  Rs[i]=R
}

Rs

# create the set of num-number of random variables based on the elements of Rs
# Say the random variables service times which are uniform distributed
# with a=15 minutes and b=45 minutes

get_Unifvar=function(R)return(a+(b-a)*R)


a=15
b=45

My_STs=sapply(Rs,get_Unifvar)

My_STs
```

```
# Buta discrete demands

dem=matrix(c(100,150,200,250,300,350,400,0.05,0.15,0.2,0.25,0.2,0.1,0.05),ncol=2)
dem

lastub=dem[1,2]

ubs=c()
ubs[1]=lastub

numintervals=dim(dem)[1]

for(i in 2:numintervals){
  lastub=lastub+dem[i,2]
  ubs[i]=lastub
}
ubs

lbs=c(0)
for(i in 2:numintervals){
  lbs[i]=ubs[i-1]
}
lbs

bounds=data.frame(lbs,ubs)

bounds

num=100

sdem=c()

x=111222333

for(i in 1:num){
  x=(a*x+c)%%m
  R=x/m
  j=1
  found=FALSE
  while(found==FALSE){
    if(lbs[j] <= R & R <ubs[j]){
      found=TRUE
      sdem[i]=dem[j,1]
    }
    j=j+1
  }
  cat("\n",R,"  ",sdem[i])
}
sdem

#hist(sdem)

table(sdem)

t(dem)


plot(table(sdem))
```

```
  x=(a*x+c)%%m
  R=x/m
  if(R<=p){S[i]=1}else{S[i]=0}
}
S


# Binomial trials simulated

# A Binomial random variable is the number of successes Y
# of n-number of independent identical Bernoulli trials with
# probability of success p delivering X_j, j in {1..n},
# Y=X_1+X_2+...+X_n.


num=10

p=0.2
n=5

binoms=c()
x=1312322333
for(j in 1:num){
  S=c()
  for(i in 1:n){
    x=(a*x+c)%%m
    R=x/m
    if(R<=p){S[i]=1}else{S[i]=0}
    binoms[j]=sum(S)

  }
  cat("\n",S,"  ",sum[S])
}
binoms

mean(binoms)

# Number of arrivals per day


lambda=20

AT=0
A=0
while(AT<=1){
  A=A+1
  x=(a*x+c)%%m
  R=x/m
  AT=AT+(-1*(1/lambda)*log(1-R))
  cat("\n",AT," ",A)
  }
A-1


num=15

lambda=20

ARRs=c()
```

```
for(i in 1:num){

AT=0
A=0
while(AT<=1){
  A=A+1
  x=(a*x+c)%%m
  R=x/m
  AT=AT+(-1*(1/lambda)*log(1-R))
  cat("\n",AT," ",A)
}

ARRs[i]=A-1
}
ARRs

hist(ARRs)

qpois

hist(rpois(num,20))

qpois(0.5,lambda)

num=20

m=2^31-1
a=7^5
c=123456
x=987654321

# Simulating number of arrivals per time unit with
# the inverse Poisson function of R.

ARRs=c()
for(i in 1: num){
  x=(a*x+c)%%m
  R=x/m
  ARRs[i]=qpois(R,lambda)
}
ARRs
```