# Assignment 6

Stochastic Models

*Juan Carlos Villaseñor-Derbez*

*2018-02-27*

The system is given by:

$$
\begin{aligned}
D &\xrightarrow{k_1} D + M \\
M &\xrightarrow{k_2} M + P \\
M &\xrightarrow{k_3} . \\
P &\xrightarrow{k_4} .
\end{aligned}
$$

In ODE form:

$$
\begin{aligned}
\frac{dM}{dt} &= k_1 D \breve{} k_3 M \\
\frac{dP}{dt} &= k_2 M \breve{} k_4 P
\end{aligned}
$$

## Part 1 - Dividing the system into discrete timesteps

The system can be divided into discrete steps expressed as probabilities of gaining $M$, loosing $M$, or $M$ staying the same:

- $P(M \rightarrow M + 1)$: $k_1 D \Delta t$
- $P(M \rightarrow M - 1)$: $k_3 i \Delta t$
- $P(M \rightarrow M)$: $(1 - k_1 D \Delta t - k_3 i \Delta t)$

```r
# Set a seed for consistent random realizations
set.seed(43)

# Define initial values
D <- 1
M <- 20

# Defien parameters
k1 <- 0.1
k3 <- 0.001
dt <- 0.1
Tend <- 10000
time <- seq(0, Tend, dt)

# Define teh cutoff probabilities for each event
p1 <- k1*dt
p2 <- k3*M*dt
```

```r
# Define vectore where M_t will be saved
Msave <- numeric(length(time))

for(i in 1:length(time)){
  Msave[i] <- M

  p2 <- k3*M*dt

  u <- runif(1)

  if(u < p1){
    M <- M + 1
  } else if(u < (p1 + p2)){
    M <- M-1
  } else {
    M <- M
  }
}

plot(time, Msave, type = "l")
```

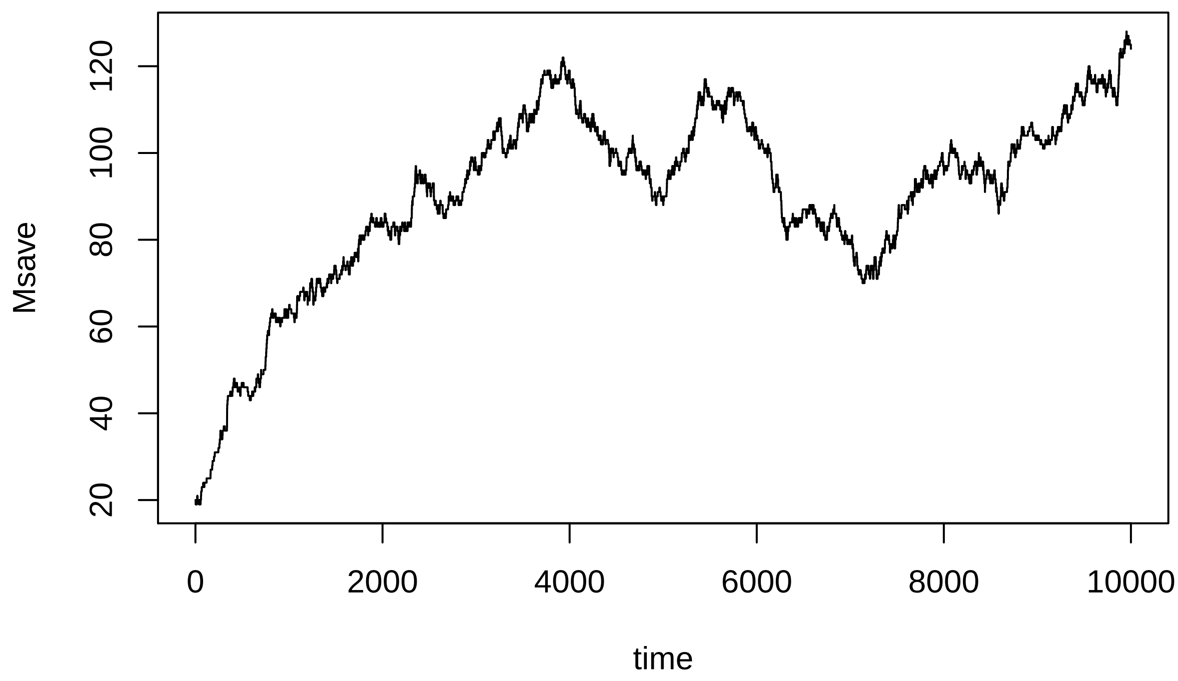



Figure 1: One run of a discretized stochastic process

### Different realizations of this system

```r
# Set a seed
set.seed(43)

# Define number of realizations and pre-define a vector to store the values
reps <- 1000
M_last <- numeric(reps)

for(j in 1:reps){
  # Define initial M as 20 vor every realization
  M <- 20

  # Define p1 and p2
  p1 <- k1*dt
  p2 <- k3*M*dt
# iterate through time
  for(i in 1:length(time)){
    p2 <- k3*M*dt

    u <- runif(1)
# Define what happens to M based on P1 and P2
    if(u < p1){
      M <- M + 1
    } else if(u < (p1 + p2)){
      M <- M-1
    } else {
      M <- M
    }
  }
  # Store the last value of M
  M_last[j] <- M
}
```

#### Histogram of the last value of M

```r
hist(M_last, freq = F, col = "grey")
x <- seq(min(M_last), max(M_last))
y <- dpois(x, (k1/k3))
lines(x,y,col="red",lwd=2)
```

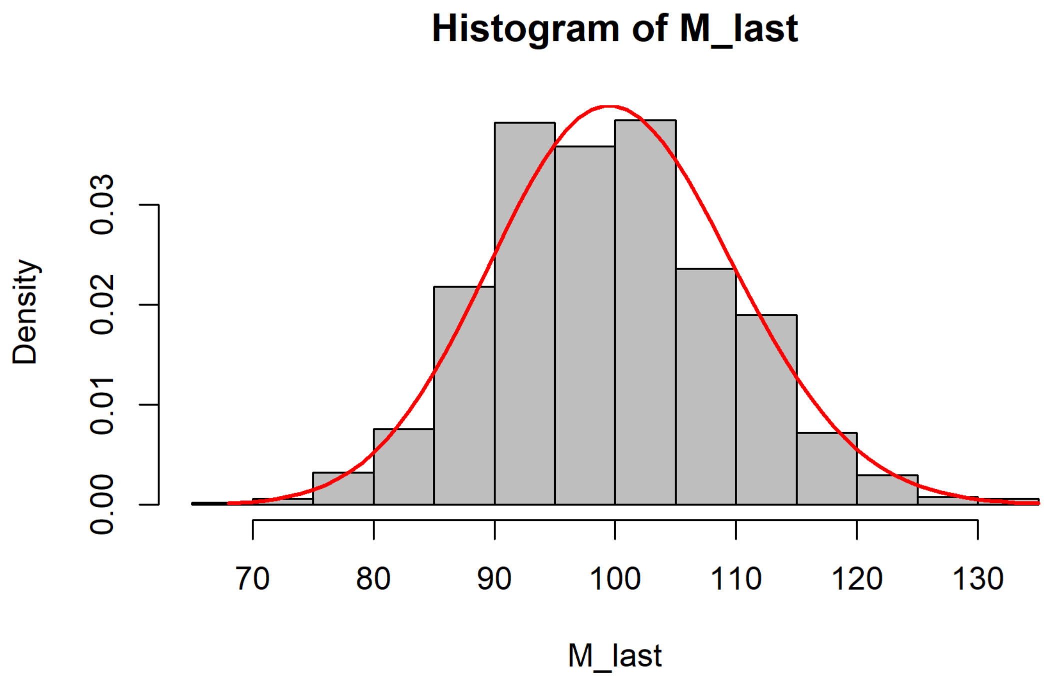



Figure 2: Histogram of the last value of 1000 realizations of a discretized stochastic system. The red line follows a poisson distribution with mean at $\frac{k_1}{k_3}$

## Modify k1

```r
k1 <- 1

M_last <- numeric(reps)

for(j in 1:reps){
  M <- 20

  p1 <- k1*dt
  p2 <- k3*M*dt

  for(i in 1:length(time)){
    p2 <- k3*M*dt

    u <- runif(1)

    if(u < p1){
      M <- M + 1
    } else if(u < (p1 + p2)){
      M <- M-1
    } else {
      M <- M
    }
  }
  M_last[j] <- M
}

hist(M_last, freq = F, col = "grey", main = "Histogram of M_last with k1 = 1")
x <- seq(min(M_last), max(M_last))
y <- dpois(x, (k1/k3))
lines(x,y,col="red",lwd=2)
```

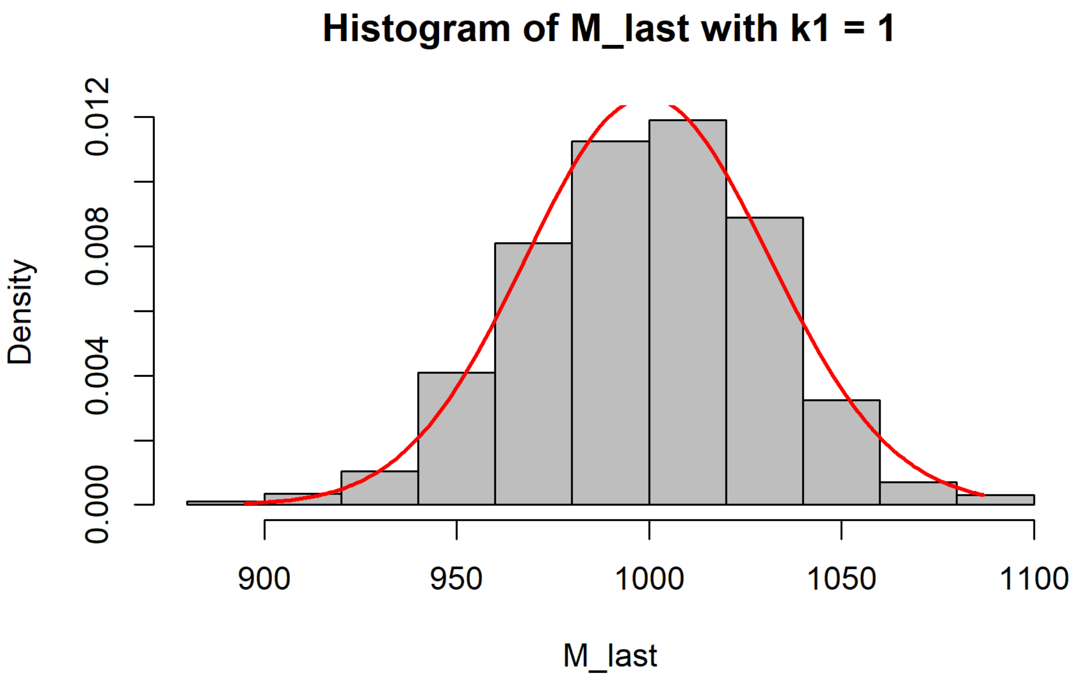
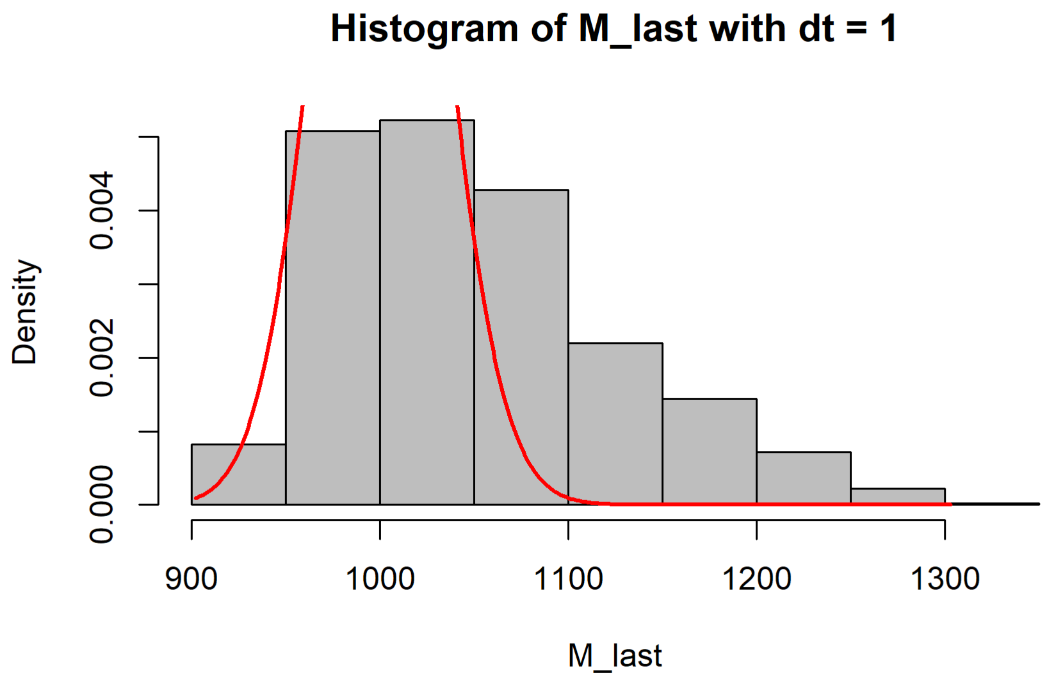



Figure 3: Histogram of the last value of 1000 realizations of a discretized stochastic system with $K_1 = 1$. The red line follows a poisson distribution with mean at $\frac{k_1}{k_3}$

### Modify dt

```
K <- 0.1
dt <- 0.5
Tend <- 10000
time <- seq(0, Tend, dt)

M_last <- numeric(reps)

for(j in 1:reps){
  M <- 20

  p1 <- k1*dt
  p2 <- k3*M*dt

  for(i in 1:length(time)){
    p2 <- k3*M*dt

    u <- runif(1)

    if(u < p1){
      M <- M + 1
    } else if(u < (p1 + p2)){
      M <- M-1
    } else {
      M <- M
    }
  }
  M_last[j] <- M
}

hist(M_last, freq = F, col = "grey", main = "Histogram of M_last with dt = 1")
x <- seq(min(M_last), max(M_last))
y <- dpois(x, (k1/k3))
lines(x,y,col="red",lwd=2)
```

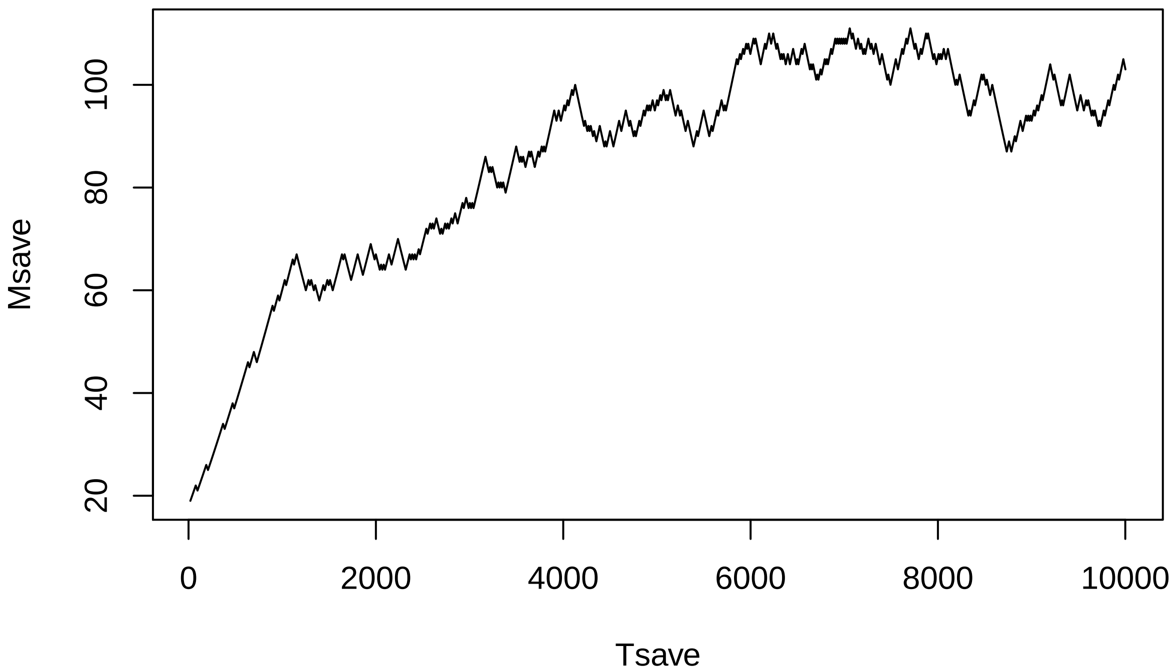


Histogram of M_last with dt = 1
Density
0.000
0.002
0.004
900
1000
1100
1200
1300
M_last

## Part 2 - Use the Gillespie Algorithm

### Get one realization

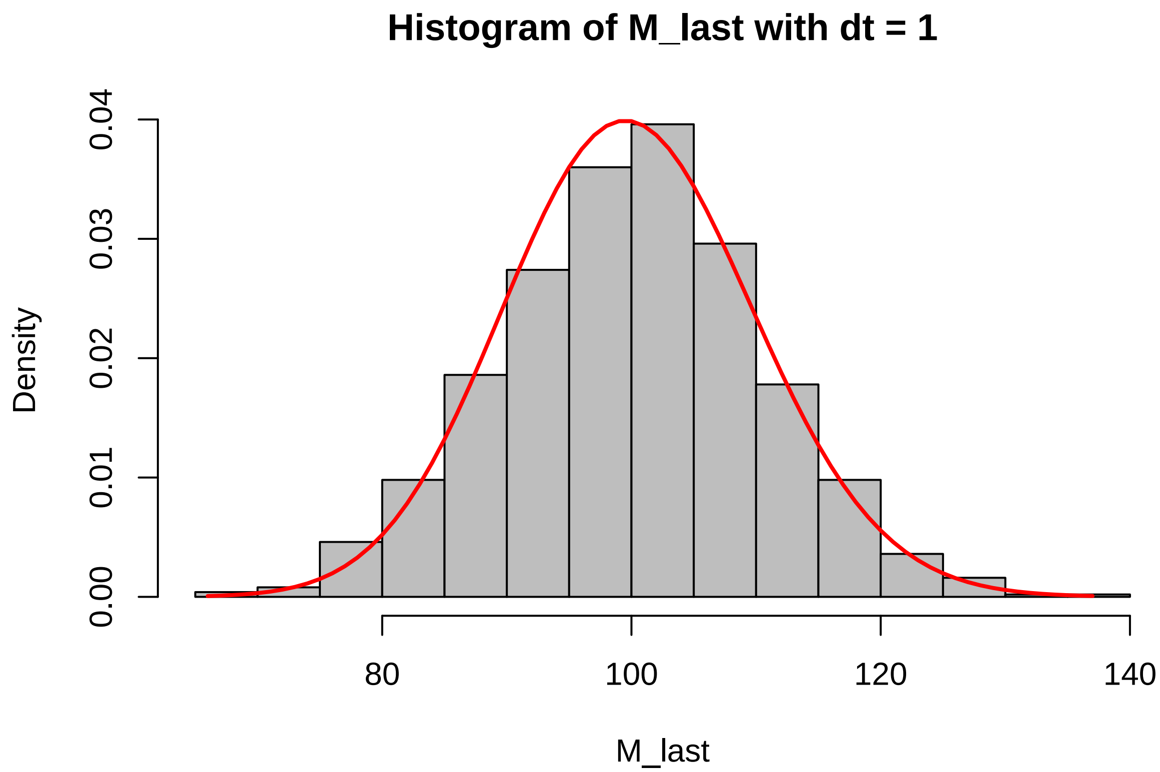
```
set.seed(43)

D <- 1
k1 <- 0.1
k3 <- 0.001
dt <- 0.1
Tend <- 10000
Msave <- numeric(0)
Tsave <- numeric(0)

M <- 20

i <- 0
t <- 0

while(t<Tend){
  i <- i+1
  a1  <- k1*D
  a2 <- k3*M
  atot <- a1 + a2
  u1 <- runif(1)
  u2 <- runif(1)

  tau <- log(1/a1)/atot
  if(u2 < (a1/atot)){
    M <- M + 1
  } else {
      M <- M-1
  }
  t <- t + tau
  Msave[i] <- M
  Tsave[i] <- t
}

plot(Tsave, Msave, type = "l")
```



Msave
20
40
60
80
100
0
2000
4000
6000
8000
10000
Tsave

### Get a histogram of the last value of 1000 realizations

```r
set.seed(43)

reps <- 1000
M <- 20
M_last <- numeric(reps)

for(j in 1:1000){
  i <- 0
  t <- 0
  while(t<Tend){
    i <- i+1
    a1  <- k1*D
    a2 <- k3*M
    atot <- a1 + a2
    u1 <- runif(1)
    u2 <- runif(1)

    tau <- log(1/a1)/atot

    if(u2 < (a1/atot)){
      M <- M + 1
    } else {
      M <- M-1
    }

    t <- t + tau
  }
  M_last[j] <- M
}

hist(M_last, freq = F, col = "grey", main = "Histogram of M_last with dt = 1")

x <- seq(min(M_last), max(M_last))
y <- dpois(x, (k1/k3))

lines(x,y,col="red",lwd=2)
```




The Gillespie Algorithm provides a better fit to the poisson distribution and, for this case, runs faster than the discretization.