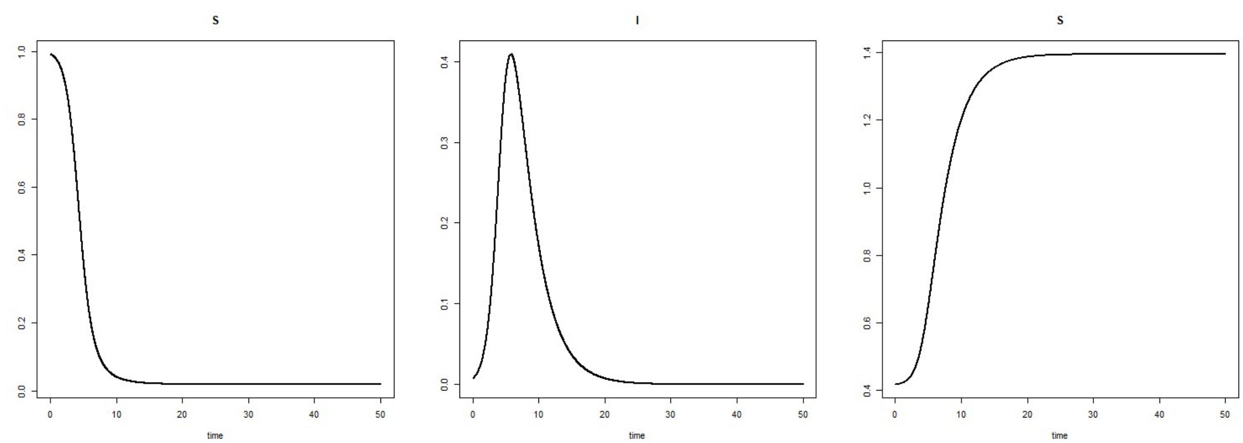
PROBLEM SET 7 (teamwork with Nguyen Dang Tien Loi)

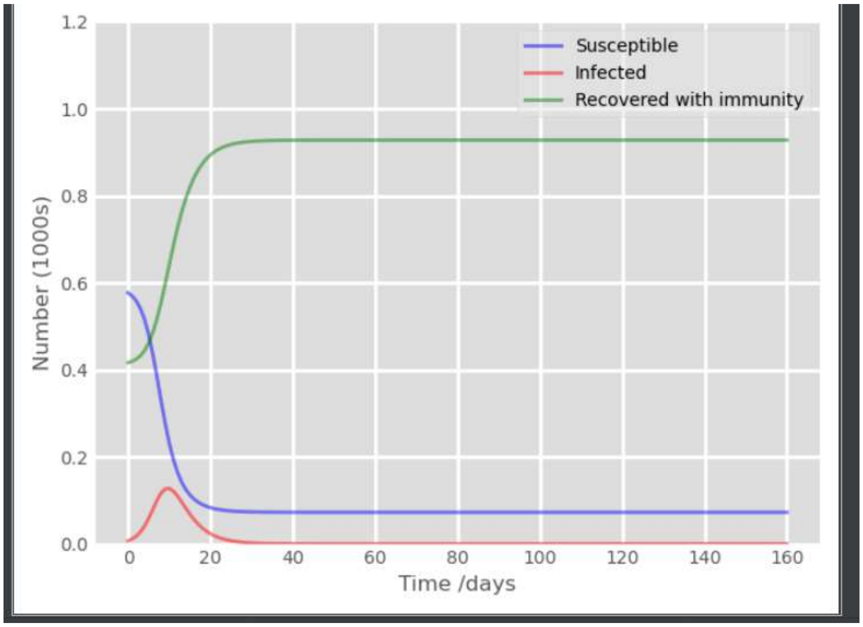
Exercise 1.

R




```
# There are three model parameters: a, b, and c that are defined thirst. Parameters are stored
# as a vector with assigned names and values:
parameters <- c(a = 1.42, b = 0.35)

# The three state variables are also created as a vector, and their initial values given:
state1 <- c(S = 0.9913406, I = 7.072e-3,S = 0.416)
state2 <- state1 + c(S = 576356, I = 7072, R=416200)

# Model equations
SIR <- function(t, state, parameters)
  { with(as.list(c(state, parameters)),
          {
            dS <- -a*I*S
            dI <- a*I*S-b*I
            dR <- b*I
            list(c(dS, dI, dR))
            })
  }

# Time specification
times <- seq(0, 50, by = 0.01)

# Model integration
# Function ode returns an object of class deSolve with a matrix that contains
# the values of the state variables (columns) at the requested output times.
out1 <- ode(state1, times, SIR, parameters)
```

Python

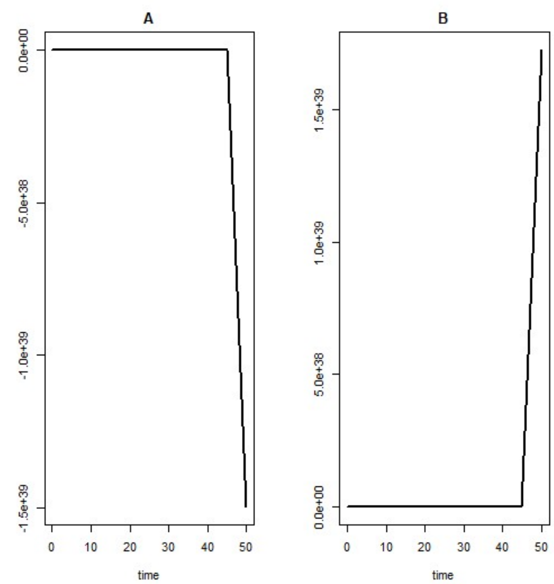



```python
r = 0.1
K = 10

a= 1.42
b= 0.35

t = np.linspace(0,160,160)
N = 1000
I0,R0 = 7.072,416.200
S0= N - I0 -R0

def SIR(y,t,N,a,b):
    S,I,R = y
    dSdt = -a*S*I/N
    dIdt = a*S*I/N - b*I
    dRdt = b*I

    return dSdt, dIdt, dRdt

y0 = S0,I0,R0

ys = odeint(SIR, y0, t, args=(N,a,b))
S,I,R = ys.T
```

Exercise 2.

a.

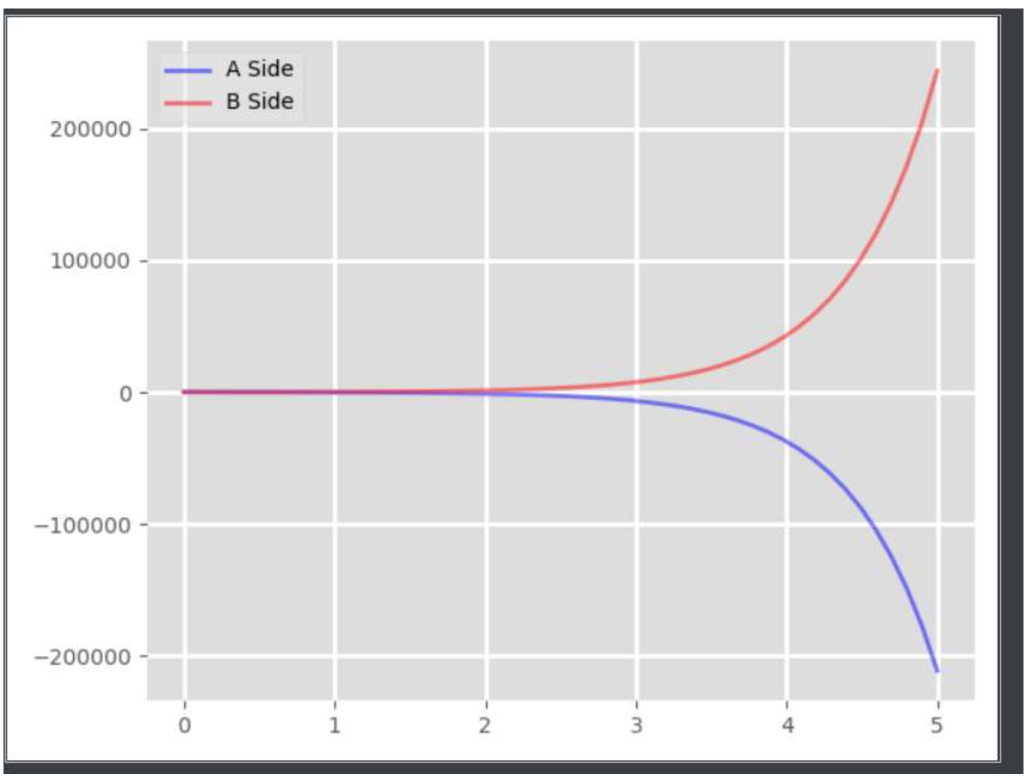
R




```r
#LANCHESTER
times <- seq(0, 50, by = 5)
parameters <- c(a = 2, b = 1.5)
state2 <- c(A = 100, B = 200)

LANCH <- function(t, state, parameters)
{ with(as.list(c(state, parameters)),
       {
         dA <- -b*B
         dB <- -a*A
         list(c(dA, dB))
       })
}


out2 <- ode(state2, times, LANCH, parameters)

# Plotting results
par(oma = c(1, 1, 1, 1)) # Set outside margins
par(mar = c(4, 5, 2, 1)) # Set plot margins
plot(out2, xlab = "time", ylab = "", lwd = 2, lty = 1, mfrow = c(1, 3))
```

Python

```
t = np.linspace(0,5,50)

A0,B0 = 100,200

a = 2
b = 1.5
y0 = A0,B0

def LAN(y,t,a,b):
    A,B = y
    dAdt = -b*B
    dBdt = -a*A

    return dAdt, dBdt

ys = odeint(LAN, y0, t, args=(a,b))
A,B = ys.T
```

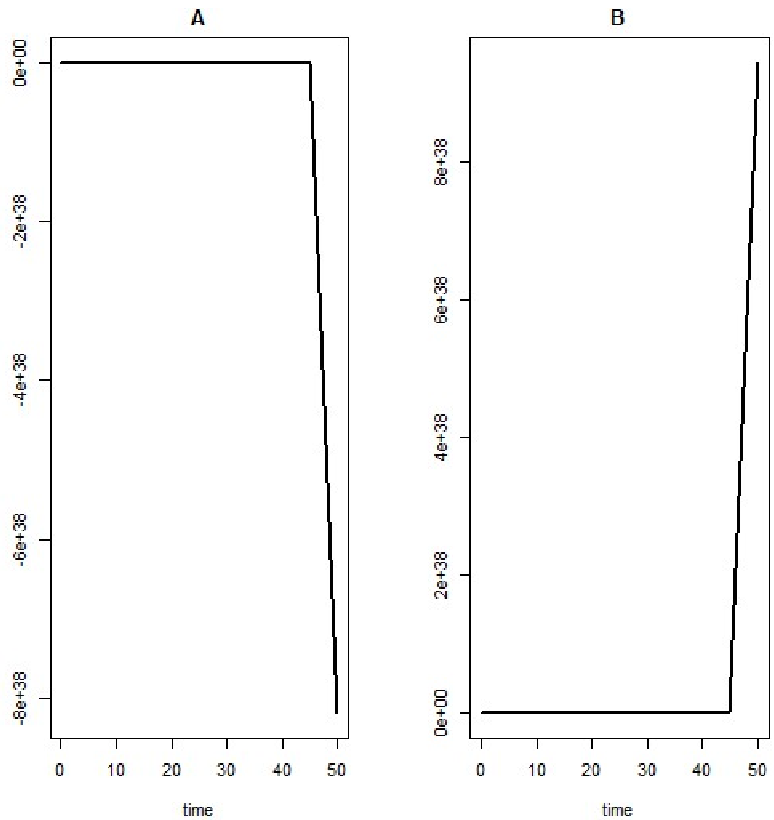


b.

$$A' = -b * B + r1 * A_0 - b_1$$

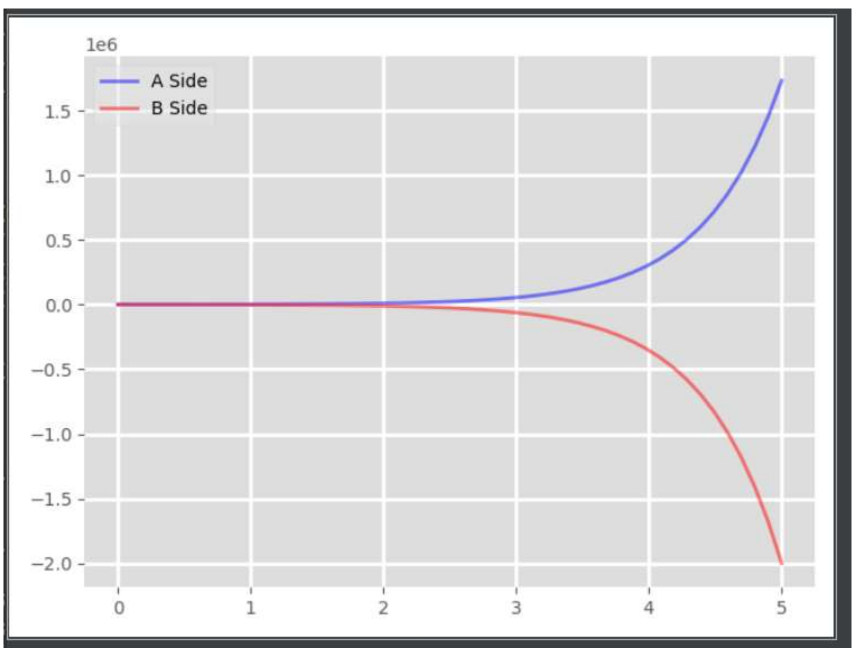
$$B' = -a * B + r2 * B_0 - b_2$$

```
#LANCHESTER
times <- seq(0, 50, by = 5)
parameters <- c(a = 2, b = 1.5, r1=0.2, r2=0.3, b1=30, b2= 20, A0=1000, B0=500)
state2 <- c(A = 100, B = 200)

LANCH <- function(t, state, parameters)
{ with(as.list(c(state, parameters)),
       {
         dA <- -b*B + r1*A0 - b1
         dB <- -a*A + r2*B0 - b2
         list(c(dA, dB))
       })
}


out2 <- ode(state2, times, LANCH, parameters)

# Plotting results
par(oma = c(1, 1, 1, 1)) # Set outside margins
par(mar = c(4, 5, 2, 1)) # Set plot margins
plot(out2, xlab = "time", ylab = "", lwd = 2, lty = 1, mfrow = c(1, 3))
```



Python

```python
a = 2
b = 1.5
r1 = 0.2
r2 = 0.3
b1 = 30
b2 = 20
A0 = 1000
B0 = 500
y0 = A0,B0

def LAN(y,t,a,b):
    A,B = y
    dAdt = -b*B + r1*A0 - b1
    dBdt = -a*A + r2*B0 - b2

    return dAdt, dBdt

ys = odeint(LAN, y0, t, args=(a,b))
A,B = ys.T
```



Exercise 3.

Probability Train Late

```
# Train Late x-Mean > 1
phi_Train = 3
P = 1/sqrt(2*pi*(phi_Train**2))*exp(-1/(2*phi_Train**2))
P
```

P = 0.1257944

```
# Bus depart Before Train mean x from 8:44 to 8:50
phi_Train = 3
phi_Bus = 1
Mean_Train = 8:44
Mean_Bus = 8:50

delta_Train <- c(0,1,2,3,4,5,6)
delta_Bus <- c(6,5,4,3,2,1,0)
t <- c(1,2,3,4,5,6,7)
P_result = 0

for (i in t)
{
  P_Train = 1/sqrt(2*pi*(phi_Train**2))*exp(-(delta_Train[i]**2)/(2*phi_Train**2))
  P_Bus = 1/sqrt(2*pi*(phi_Bus**2))*exp(-(delta_Bus[i]**2)/(2*phi_Train**2))
  P_result = P_result + P_Train*P_Bus
}
P_result
```

```
for (i in t)
{
  P_Train = 1/sqrt(2*pi*(phi_Train**2))*exp(-(delta_Train[i]**2)/(2*phi_Train**2))
  P_Bus = 1/sqrt(2*pi*(phi_Bus**2))*exp(-(delta_Bus[i]**2)/(2*phi_Train**2))
  P_result = P_result + P_Train*P_Bus
}
P_result
1] 0.09383193
```

Python

```
phi_Train = 3
P = 1/math.sqrt(2*math.pi*(phi_Train**2))*math.exp(-1/(2*phi_Train**2))
print("Result"+str(P))
```

Result :0.12579440923099774

```
phi_Train = 3
phi_Bus = 1


delta_Train = [0,1,2,3,4,5,6]
delta_Bus = [6,5,4,3,2,1,0]

P_result = 0

for i in range(1,7):
  P_Train = 1/math.sqrt(2*math.pi*(phi_Train**2))*math.exp(-(delta_Train[i]**2)/(2*phi_Train
  P_Bus = 1/math.sqrt(2*math.pi*(phi_Bus**2))*math.exp(-(delta_Bus[i]**2)/(2*phi_Train**2))
  P_result = P_result + P_Train*P_Bus

print("Result:"+str(P_result))
```

Result:0.086652166915449

Exercise 4.

```
def rng1(seed, a, b, M, ntotal):
    data = np.zeros(ntotal)
    data[0] = seed
    for i in range(1,ntotal):
        data[i] = np.mod((a*data[i-1]+b), M)
    return data/np.float(M)

def rng2(seed, a, b, M, ntotal):
    data = np.zeros(ntotal)
    data[0] = seed
    for i in range(1,ntotal):
        data[i] = np.mod((a*data[i-1]+b), M)/np.float(M)
    return data

def rng3(seed, a, b, M, ntotal):
    data = np.zeros(ntotal)
    for i in range(1,ntotal):
        data[i] = np.mod((a*data[i-1]+b), M)
    return data

def rng4(seed, a, b, M, ntotal):
    data = np.zeros(ntotal)
```

Outcome:

```
Result 1: [0.0042 0.9963 0.0042 0.9963 0.0042]
Result 2: [42.          0.9963      0.99670037  0.9971007   0.99750099]
Result 3: [0. 5. 0. 5. 0.]
```

Code No 2 is the correct one

Exercise 5.

No Gaussian Distribution is determined of the probability of 0.683 for points lie within 1 standard deviation of the mean.

```python
ave = 0.0
std = 1.0
N = 100000
np.random.seed(0)
X = ave+std*np.random.randn(N)
plt.ylim(-10,10)
plt.xlabel(r'$i$',fontsize=16)
plt.ylabel(r'$x_i$',fontsize=16)
plt.plot(X,',')
plt.show()

count = 0

for i in range(100000):
    if X[i]>=5:
        count += 1

print("Count of Number out 5 phi:"+str(count))
```

```
"C:\Users\Tien Loi\Miniconda3\envs\untitled\python.exe" "C:/Users/Tien Loi/Desktop/simulation/ps7/random_gene.py"
Count of Number out 5 phi:0

Process finished with exit code 0
```