BIOS 26210: Lab Exercise 6

Si Tang, 396904

November 28, 2009

Exercise 1

Preparation: Function Needed.

```
> plotCobweb <- function(f, max_iter) {
+     iVal <- 1
+     iiVal <- iVal/2
+     while (f(iVal) <= 0 | f(iiVal) == iiVal) {
+         iVal <- iVal/10
+         iiVal <- iVal/2
+     }
+     xRange <- numeric(length = 2)
+     xRange[1] <- 0
+     xMax <- iiVal
+     while (f(xMax) >= 0) {
+         xMax <- 1.1 * xMax
+     }
+     xRange[2] <- xMax
+     yArr <- numeric(length = max_iter * 2)
+     xArr <- yArr
+     xArr[1] <- 0
+     yArr[1] <- iiVal
+     for (ii in 1:max_iter) {
+         yArr[2 * ii] <- f(yArr[2 * ii - 1])
+         yArr[2 * ii + 1] <- yArr[2 * ii]
+         xArr[2 * ii] <- xArr[2 * ii - 1]
+         xArr[2 * ii + 1] <- yArr[2 * ii]
+     }
+     curve(f, xlim = xRange, bty = "n", fg = grey(0.6), xlab = expression(X[t]),
+         ylab = expression(X[t + 1]))
+     abline(h = 0)
+     abline(v = 0)
+     lines(c(xRange[1] - 100, xRange[2] + 100), c(xRange[1] -
+         100, xRange[2] + 100), col = "red")
+     lines(xArr, yArr, col = "green")
+ }
```

```
> plotNumSol <- function(f, x0, r = 2, nDiscard = 0, max_iter = 500) {
+     xArr <- numeric(length = max_iter + 1)
+     xArr[1] <- x0
+     for (i in 1:max_iter) {
+         xArr[i + 1] = f(xArr[i], r)
+     }
+     plot(0, 0, type = "n", bty = "n", fg = grey(0.6), xlim = range((nDiscard +
+         1):max_iter + 1), ylim = range(xArr), xlab = expression(X[t]),
+         ylab = expression(X[t + 1]))
+     lines((nDiscard + 1):(max_iter + 1), xArr[(nDiscard + 1):(max_iter +
+         1)], col = "blue", type = "b", pch = 1)
+ }
> NumSol <- function(f, x0, r = 2, max_iter) {
+     xArr <- numeric(length = max_iter + 1)
+     xArr[1] <- x0
+     for (i in 1:max_iter) {
+         xArr[i + 1] = f(xArr[i], r)
+     }
+     return(xArr)
+ }
```

1a.

Set $r = 2$ and plot the cobweb and the numerical solution as in Fig. 1 for model $X_{t+1} = rX_t(1 - X_t)$. Since this model is approaching its fixed point at $X_t = 0.5$ and stay there forever, I just plot the first 50 iterations.

```
> logisticModel <- function(x, r = 2) {
+     return(r * x * (1 - x))
+ }
> par(mfrow = c(1, 2))
> plotCobweb(logisticModel, 50)
> title(main = "Cobweb Plot", sub = "Si Tang-Problem 1(a)-1")
> plotNumSol(logisticModel, x0 = 0.01, r = 2, nDiscard = 0, max_iter = 50)
> title(main = "Numerical Solution", sub = "Si Tang-Problem 1(a)-2")
```

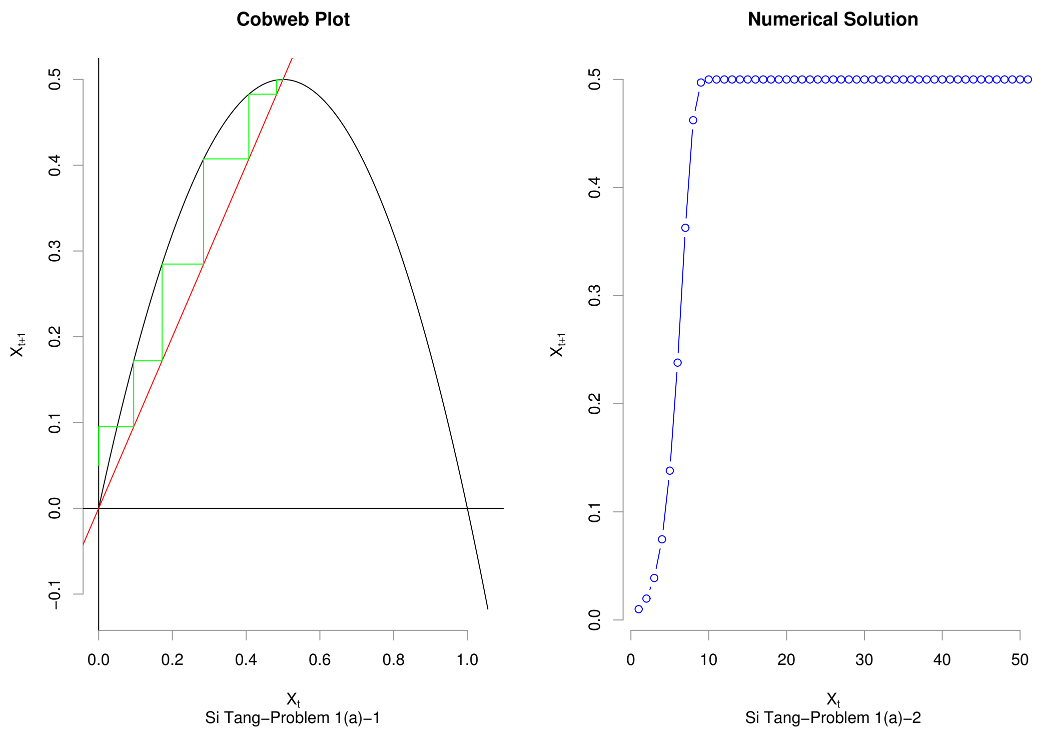

Fig. 1 Cobweb and numerical solution of the logistic model $X_{t+1} = rX_t(1 - X_t)$

1b.

```
> solver <- function(f, x0, r_min, r_max, dr, max_iter) {
+     r <- seq(r_min, r_max, by = dr)
+     x <- array(NA, c(length(r), max_iter + 1))
+     for (i in 1:length(r)) {
+         x[i, ] <- NumSol(f, x0, r[i], max_iter)
+     }
+     return(list(r = r, x = x))
+ }
```

1c.

```
> max_iter <- 500
> rmin <- 0
> rmax <- 4
> dr <- 0.01
> x0 <- 0.01
> nDiscard <- 200
> result <- solver(logisticModel, x0, rmin, rmax, dr, max_iter)
> r <- result$r
> x <- result$x
> plot(0, 0, type = "n", bty = "n", fg = grey(0.6), xlab = "r",
+     ylab = expression(X[t]), xlim = range(r), ylim = range(x))
> for (i in 1:length(r)) {
+     points(rep(r[i], max_iter - nDiscard + 1), x[i, (nDiscard +
+         1):(max_iter + 1)], col = "blue", pch = ".")
+ }
```

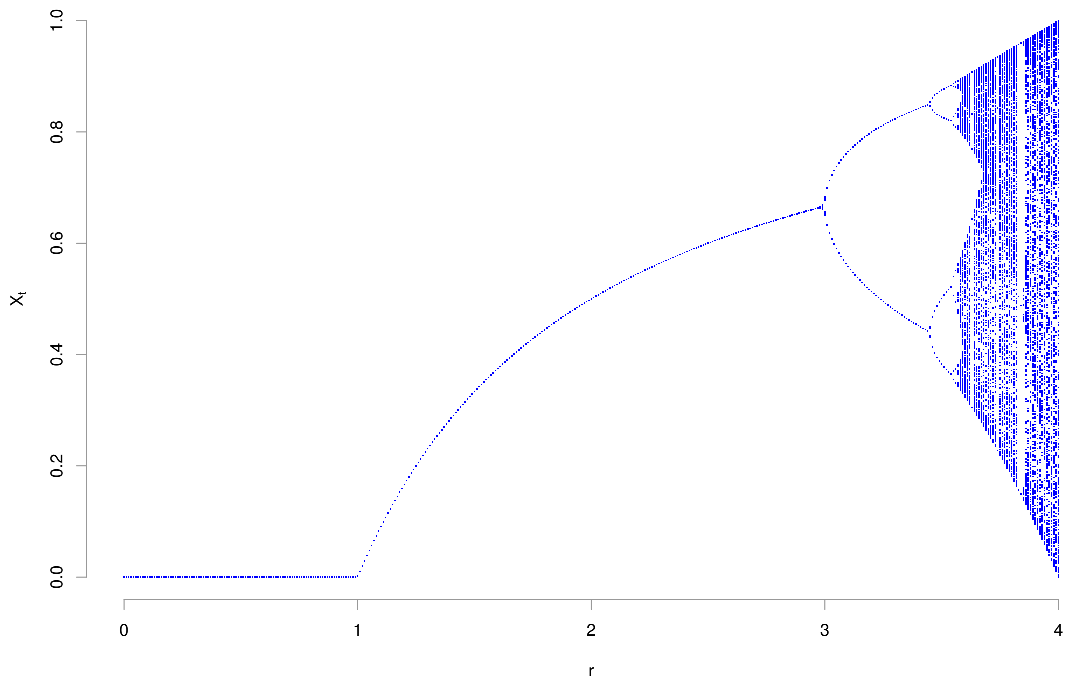

Fig. 2 Bifurcation diagram of the logistic model $X_{t+1} = rX_t(1 - X_t)$ over r

1d.

```
> par(mfrow = c(4, 1))
> r <- c(2, 3.3, 3.5, 4)
> for (i in 1:length(r)) {
+     plotNumSol(logisticModel, x0 = 0.01, r = r[i], nDiscard = 200,
+         max_iter = 500)
+     title(main = paste("Numerical Solution: r=", r[i]), sub = paste("Si
    Tang-Problem 1(d) -",
+         i))
+ }
```

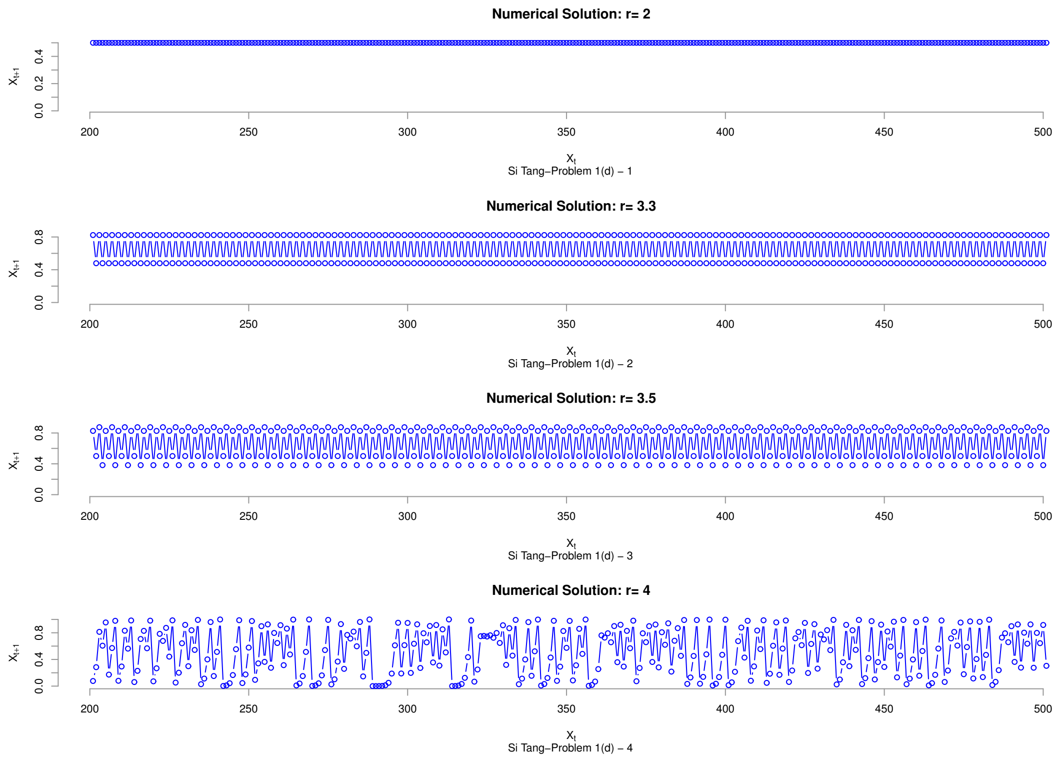

Fig. 3 Numerical solution over time of the logistic model $X_{t+1} = rX_t(1 - X_t)$, with different values of parameter r. From upper to lower: $r = 2,\ 3.3,\ 3.5,\ 4$

Exercise 2

2a.

The population model is:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix} = \begin{bmatrix} 0.5 & 2 \\ 0.1 & 0 \end{bmatrix} \times \begin{bmatrix} x_t \\ y_t \end{bmatrix}$$

According to the Leslie model, x denotes the young age group, which does not produce, with the survival rate 0.5 to grow up and rate 0.1 to become mature, and y denotes the mature age group , which repoduces with mean fecundity of 2 and then dies. So in this population, half of the young age group will grow up and 10% of the young age group will become mature, thus 40% of the young age group will die.

2b.

The matix equation above can be written as the following two equations:

$$\begin{aligned} x_{t+1} &= 0.5x_t + 2y_t \\ y_{t+1} &= 0.1x_t \end{aligned}$$

Write the code (see box below) and plot the population vector as in Fig.4.

```
> leslieModel <- function(x, y) {
+     x1 <- 0.5 * x + 2 * y
+     y1 <- 0.1 * x
+     return(c(x1, y1))
+ }
> solveLeslie <- function(f, x0, y0, t_max) {
+     pop <- array(NA, c((t_max + 1), 2))
+     pop[1, 1] <- x0
+     pop[1, 2] <- y0
+     for (i in 2:(t_max + 1)) {
+         pop[i, ] <- leslieModel(pop[i - 1, 1], pop[i - 1, 2])
+     }
+     return(pop)
+ }
> tmax2b <- 50
> pop <- solveLeslie(leslieModel, 10, 10, tmax2b)
> plot(1:(tmax2b + 1), pop[1:(tmax2b + 1), 1] + pop[1:(tmax2b +
+     1), 2], bty = "n", type = "b", fg = grey(0.6), col = 4, pch = 19,
+     xlab = "t", ylab = "Population", main = "Population vector Plot",
+     sub = "Si Tang-Problem 2(b)-1")
> lines(1:(tmax2b + 1), pop[1:(tmax2b + 1), 1], pch = 19, bty = "n",
+     type = "b", fg = grey(0.6), col = 2)
> lines(1:(tmax2b + 1), pop[1:(tmax2b + 1), 2], pch = 19, bty = "n",
+     type = "b", fg = grey(0.6), col = 3)
> legend(30, 8, paste(c(expression(x[t]), expression(y[t]), expression(N[t])),
+     "~t"), col = c(2:4), lwd = 1, bty = "n")
> abline(h = 0)
> abline(v = 0)
```

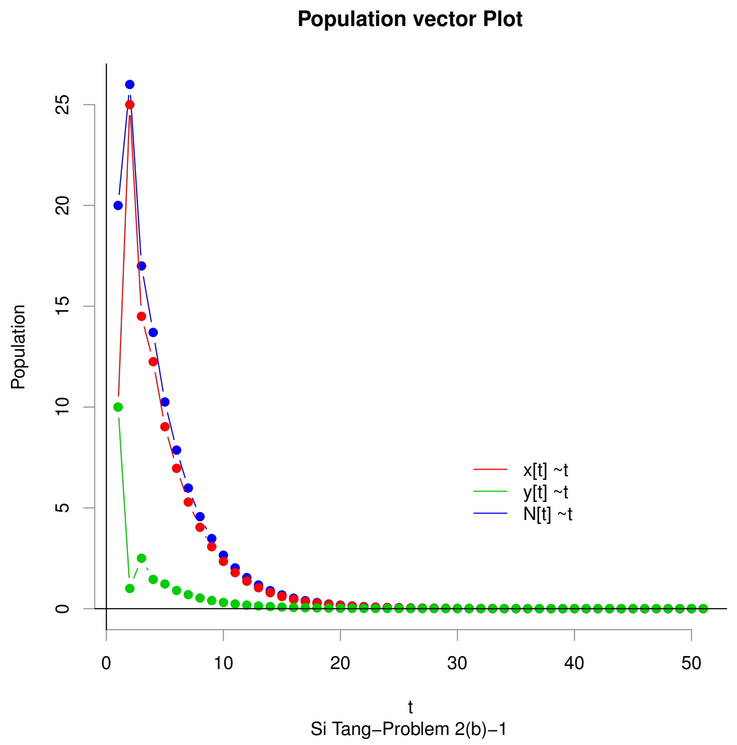

Fig. 4 The population vector plot, starting at $(x_0 = 10, y_0 = 10)$ with $t_{max} = 50$

2c.

Write code to solve this population model and plot x_t and y_t in Fig.5. Also plot the ratio of x_{t+1}/x_t and y_{t+1}/y_t over the time course in Fig. 6.
From Fig. 5, we can see this population will gradually go extinct, no matter what the initial condition is.

In addition, from Fig. 6, the growth rate (or actually the 'decay' rate) of the mature age group, the young age group and the whole population will approach a constant 0.7623475.

```
> tmax2c <- 50
> x0 <- c(1, 10, 10)
> y0 <- c(5, 10, 1)
> pop <- array(NA, c(length(x0), (tmax2c + 1), 2))
> for (i in 1:length(x0)) {
+     pop[i, 1:(tmax2c + 1), 1:2] <- solveLeslie(leslieModel, x0[i],
+         y0[i], tmax2c)
+ }
> plot(0, 0, type = "n", bty = "n", fg = grey(0.6), xlim = c(0,
+     tmax2c), ylim = c(0, 10), xlab = "t", ylab = "Population",
+     main = "Population vector Plot", sub = "Si Tang-Problem 2(c)-1")
> color = 2
> for (i in 1:length(x0)) {
+     lines(1:(tmax2c + 1), pop[i, 1:(tmax2c + 1), 1], lty = 1,
+         cex = 0.7, type = "p", fg = grey(0.6), col = color, pch = 16)
+     lines(1:(tmax2c + 1), pop[i, 1:(tmax2c + 1), 1], lty = 1,
+         cex = 0.7, type = "l", fg = grey(0.6), col = color, pch = 16)
+     color <- color + 1
+ }
> legend(30, 8, paste("X~t   , ", "x0=", x0, ", y0=", y0), col = 2:4,
+     ncol = 1, lwd = 1, bty = "n", pch = 16)
> abline(h = 0)
> abline(v = 0)
> for (i in 1:length(y0)) {
+     lines(1:(tmax2c + 1), pop[i, 1:(tmax2c + 1), 2], lty = 1,
+         cex = 0.7, type = "p", fg = grey(0.6), col = color, pch = 17)
+     lines(1:(tmax2c + 1), pop[i, 1:(tmax2c + 1), 2], lty = 1,
+         cex = 0.7, type = "l", fg = grey(0.6), col = color, pch = 17)
+     color <- color + 1
+ }
> legend(30, 4, paste("Y~t   , ", "x0=", x0, ", y0=", y0), col = 5:7,
+     ncol = 1, lwd = 1, bty = "n", pch = 17)
```

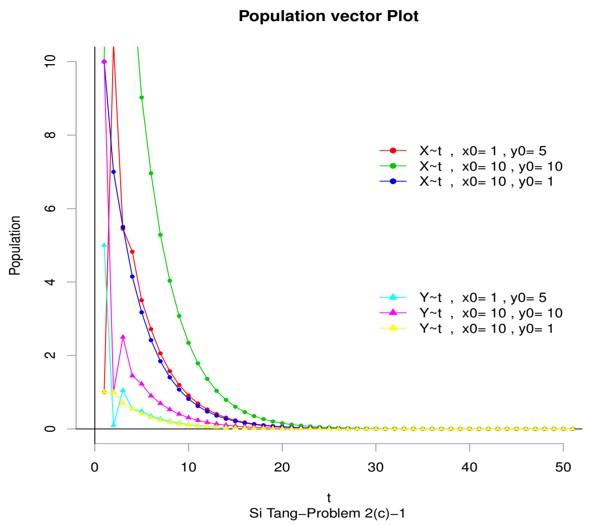

Fig. 5 The plot of the population vector and the ratio between the mature and young age group over the time course.

```
> plot(0, 0, type = "n", bty = "n", fg = grey(0.6), xlim = c(0,
+     tmax2c), ylim = c(0, 2), xlab = "t", ylab = "Population",
+     main = "", sub = "Si Tang-Problem 2(c)-2")
> color = 2
> for (i in 1:length(x0)) {
+     lines(1:(tmax2c), pop[i, 2:(tmax2c + 1), 1]/pop[i, 1:(tmax2c),
+         1], lty = 1, cex = 0.7, type = "p", fg = grey(0.6), col = color,
+         pch = 16)
+     lines(1:(tmax2c), pop[i, 2:(tmax2c + 1), 1]/pop[i, 1:(tmax2c),
+         1], lty = 1, cex = 0.7, type = "l", fg = grey(0.6), col = color,
+         pch = 16)
+     color <- color + 1
+ }
> legend(20, 2, paste("X[t+1]/X[t] ~ t, ", "x0=", x0, ", y0=",
+     y0), col = 2:4, ncol = 1, lwd = 1, bty = "n", pch = 16)
> abline(h = 0)
> abline(v = 0)
> for (i in 1:length(y0)) {
+     lines(1:(tmax2c), pop[i, 2:(tmax2c + 1), 2]/pop[i, 1:(tmax2c),
+         2], lty = 1, cex = 0.7, type = "p", fg = grey(0.6), col = color,
+         pch = 17)
+     lines(1:(tmax2c), pop[i, 2:(tmax2c + 1), 2]/pop[i, 1:(tmax2c),
+         2], lty = 1, cex = 0.7, type = "l", fg = grey(0.6), col = color,
+         pch = 17)
+     color <- color + 1
+ }
> legend(20, 1.5, paste("Y[t+1]/Y[t] ~ t, ", "x0=", x0, ", y0=",
+     y0), col = 5:7, ncol = 1, lwd = 1, bty = "n", pch = 17)
```

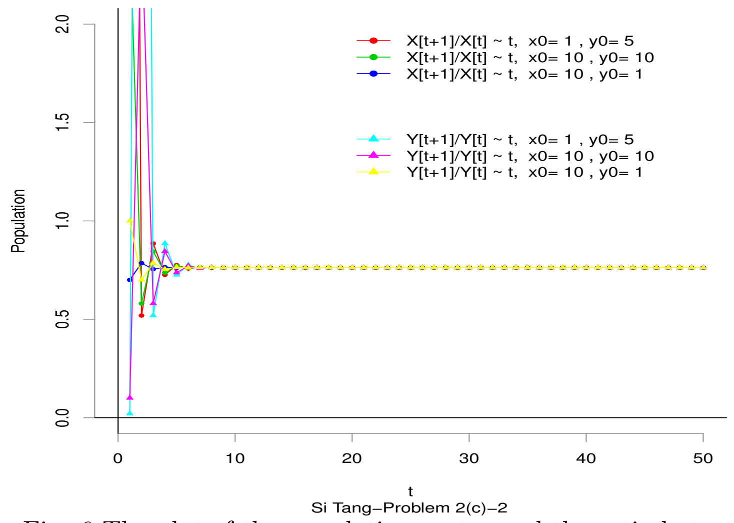

Fig. 6 The plot of the population vector and the ratio between the mature and young age group over the time course.

2d.

Since the solution to the Leslie model is like the following expressions:

$$\begin{aligned} x_{t+1} &= C_1\lambda_1^t + C_2\lambda_2^t, \quad (\lambda_2 < \lambda_1) \\ y_{t+1} &= C_3\lambda_1^t + C_4\lambda_2^t, \quad (\lambda_2 < \lambda_1) \end{aligned}$$

When $t \to \infty$, $\lim_{t\to\infty} \frac{x_{t+1}}{x_t} = \lambda_1$, then I can estimate from Fig. 6 that the larger eigenvalue of the matrix is $\lambda_1 = 0.7623475$.

Exercise 3

3a.

On adding a parameter a, the population model changes from a Leslie model to a Usher model. The parameter a here denotes the percentage of the mature age group (y) that survives from time t to time $t + 1$.

3b.

```
> UsherModel <- function(x, y, a) {
+     x1 <- 0.5 * x + 2 * y
+     y1 <- 0.1 * x + a * y
+     return(c(x1, y1))
+ }
> solveUsher <- function(f, x0, y0, a_min, a_max, da, t_max) {
+     a <- seq(a_min, a_max, by = da)
+     pop <- array(NA, c(length(a), (t_max + 1), 2))
+     for (k in 1:length(a)) {
+         pop[k, 1, 1] <- x0
+         pop[k, 1, 2] <- y0
+         for (i in 2:(t_max + 1)) {
+             pop[k, i, ] <- f(pop[k, i - 1, 1], pop[k, i - 1,
+                 2], a[k])
+         }
+     }
+     return(pop)
+ }
```

```
> tmax <- 500
> amin <- 0
> amax <- 0.8
> da <- 0.01
> x0 <- 10
> y0 <- 8
> a = seq(amin, amax, by = da)
> total <- numeric(length = length(a))
> total2 <- numeric(length = length(a))
> result <- solveUsher(UsherModel, x0, y0, amin, amax, da, tmax)
> for (i in 1:length(a)) {
+     total[i] <- result[i, tmax, 1] + result[i, tmax, 2]
+     total2[i] <- result[i, tmax - 1, 1] + result[i, tmax - 1,
+         2]
+ }
> par(mfrow = c(1, 3))
> plot(a, total, type = "l", xlim = c(0, 1), bty = "n", xlab = "a",
+     ylab = expression(x[500] + y[500]), fg = grey(0.6), col = "blue",
+     main = "Convergence Behavior of the Usher Population", sub = "Si
    Tang-Problem 3(b)-1")
> abline(v = 0)
> abline(h = 0)
> a2 <- a[a <= 0.6]
> plot(a2, total[1:length(a2)], type = "l", xlim = c(0, 1), bty = "n",
+     xlab = "a", ylab = expression(x[500] + y[500]), fg = grey(0.6),
+     col = "blue", main = "Convergence Behavior of the Usher Population\n (
    Scaled Figure)",
+     sub = "Si Tang-Problem 3(b)-2")
> abline(v = 0)
> abline(h = 0)
> plot(a, total/total2, type = "l", xlim = c(0, 1), bty = "n",
+     xlab = "a", ylab = expression(lambda), fg = grey(0.6), col = "blue",
+     main = "Eigenvalue versus a", sub = "Si Tang-Problem 3(b)-3")
> abline(v = 0)
> abline(h = 0)
```

Plot the convergence behavior of the Usher population model versus different values of a in Fig. 7.

From Fig.7, we could see that when $a < 0.6$, the largest eigenvalue of the model is less than 1 and the population will go extinction; when $a = 0.6$, the largest eigenvalue of the model is equal to 1 and the population will stay at a equilibrium; when $a > 0.6$, the largest eigenvalue of the model is greater than 1, and population will grow without bound.

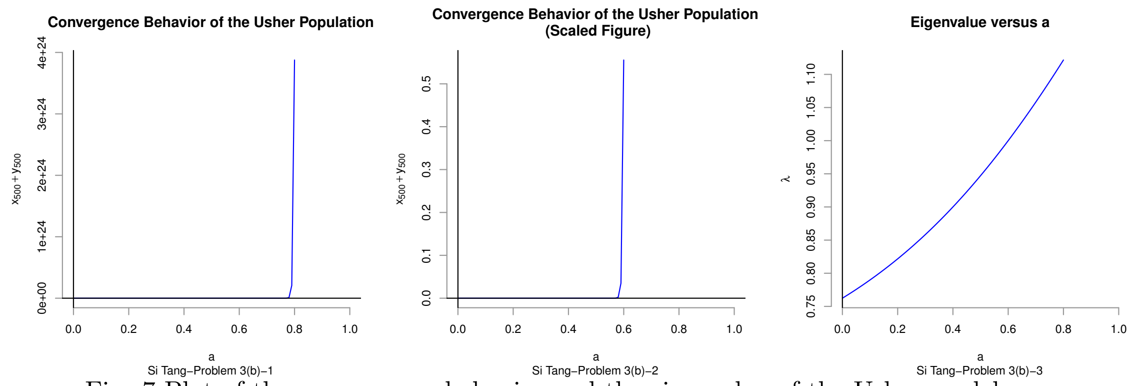

Fig. 7 Plot of the convergence behavior and the eigenvalue of the Usher model versus different values of a: left, convergence behavior of the population versus a, $a \in [0, 0.8]$; middle, scale the left figure to $a \in [0, 0.6]$; right, the change of the eigenvalue over a.

3c.

From the analysis above, the bifurcation point of this Usher model is at $a = 0.6$, and the largest eigenvalue at the bifurcation is $\lambda = 1.0$. Plot the representative solutions on both sides of $a = 0.6$: (1) $a = 0.55$, (2) $a = 0.6$, (3). $a = 0.62$.(See Fig. 8, next page).

```
> par(mfrow = c(3, 1))
> aVal <- c(0.55, 0.6, 0.62)
> for (ii in 1:length(aVal)) {
+     plot(0:tmax, result[which(a == aVal[ii]), 1:(tmax + 1), 1] +
+         result[which(a == aVal[ii]), 1:(tmax + 1), 2], xlab = "t",
+         ylab = "Population", type = "l", col = "blue", ylim = c(0,
+             max(result[which(a == aVal[ii]), , 1] + result[which(a ==
+                 aVal[ii]), , 2])), bty = "n", fg = grey(0.6),
+         main = "Population Behavior of the Usher Model", sub = paste("Si
    Tang-Problem 3(b)-",
+             ii))
+     lines(0:tmax, result[which(a == aVal[ii]), 1:(tmax + 1),
+         1], xlab = "t", ylab = "Population", type = "l", col = "red",
+         bty = "n", fg = grey(0.6))
+     lines(0:tmax, result[which(a == aVal[ii]), 1:(tmax + 1),
+         2], xlab = "t", ylab = "Population", type = "l", col = "green",
+         bty = "n", fg = grey(0.6))
+     abline(v = 0)
+     abline(h = 0)
+     legend(350, max(result[which(a == aVal[ii]), 1:(tmax + 1),
+         1] + result[which(a == aVal[ii]), 1:(tmax + 1), 2]),
+         paste(c("N", "X", "Y"), "~ t, a= ", aVal[ii]), col = c("blue",
+             "red", "green"), lwd = 1, bty = "n")
+ }
```

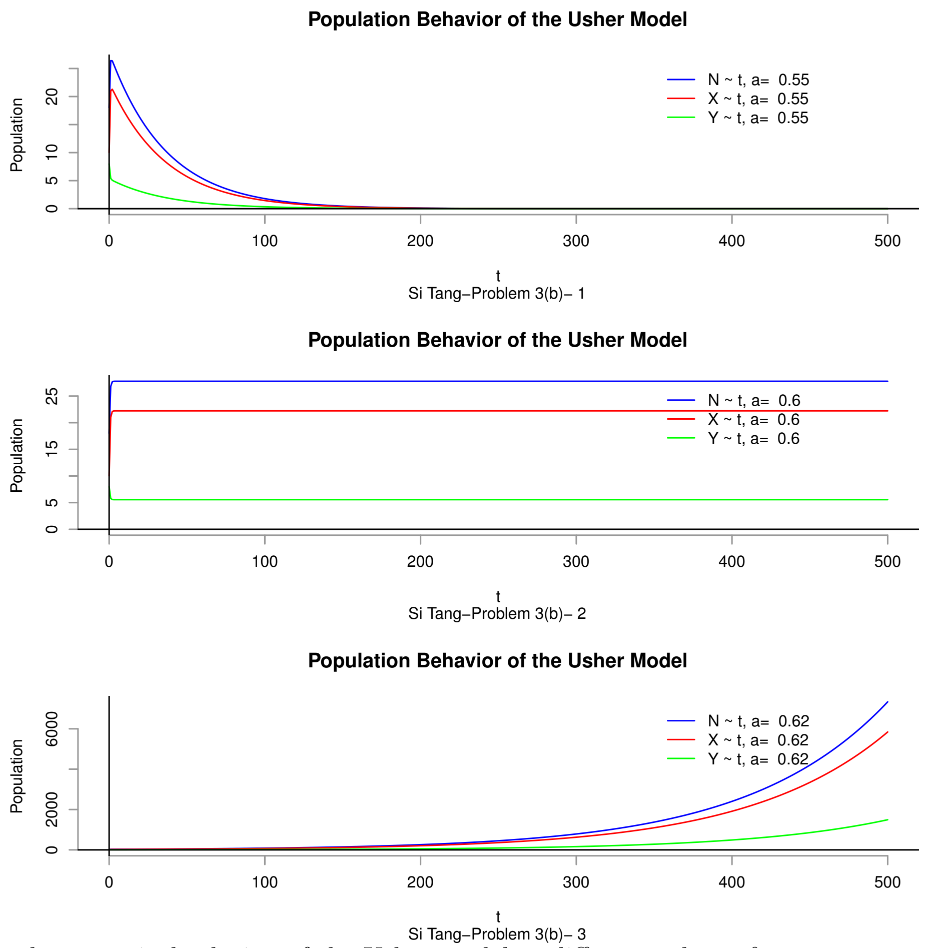

Fig. 8 Plot the numerical solution of the Usher model at different values of a: upper, $a = 0.55$; middle, $a = 0.60$; lower, $a = 0.62$, starting from $(x_0 = 10, y_0 = 8)$ with $t_{max} = 500$

Exercise 4.

4a.

```
> Vector2lList <- function(x) {
+     if (is.numeric(x) == FALSE) {
+         cat("The Input is not a numeric vector!")
+     }
+     else {
+         i <- length(x)
+         dataPart <- numeric()
+         pointPart <- numeric()
+         if (i <= 0) {
+             cat("No numeric values in the vector.")
+             return()
+         }
+         else if (i == 1) {
+             dataPart <- x
+             pointPart <- -1
+         }
+         else {
+             dataPart <- sort(x)
+             for (ii in 1:(i - 1)) {
+                 index = which(x == dataPart[ii])
+                 pointPart[index] = which(x == dataPart[ii + 1])
+             }
+             index = which(x == dataPart[i])
+             pointPart[index] <- -1
+             dataPart <- x
+         }
+         lList <- list(dataPart = dataPart, pointPart = pointPart)
+         return(lList)
+     }
+ }
```

4b.

```
> lList.add <- function(lList, element) {
+     if (is.numeric(lList$dataPart) & is.numeric(element)) {
+         if (length(which(lList$dataPart == element)) == 0) {
+             i <- length(lList$dataPart)
+             minValue <- min(lList$dataPart)
+             currentValue <- minValue
+             currentPoint <- lList$pointPart[which(lList$dataPart ==
+                 currentValue)]
+             while (currentValue <= element) {
+                 previousValue = currentValue
+                 previousPoint = currentPoint
+                 currentValue = lList$dataPart[currentPoint]
+                 currentPoint = lList$pointPart[currentPoint]
+             }
+             tmp <- lList$pointPart[lList$dataPart == previousValue]
+             lList$pointPart[lList$dataPart == previousValue] <- i +
+                 1
+             lList$pointPart <- c(lList$pointPart, tmp)
+             lList$dataPart <- c(lList$dataPart, element)
+             return(lList)
+         }
+         else {
+             cat("The element already exists in the linked list!")
+         }
+     }
+     else {
+         cat("The element or the linked list is not numeric! ")
+     }
+ }
```

4c.

```
> lList.delete <- function(lList, element) {
+     if (is.numeric(lList$dataPart) & is.numeric(element)) {
+         minValue <- min(lList$dataPart)
+         maxValue <- max(lList$dataPart)
+         if (length(which(lList$dataPart == element)) == 1) {
+             if (element == minValue) {
+                 index <- which(lList$dataPart == minValue)
+                 for (ii in 1:length(lList$pointPart)) {
+                   if (lList$pointPart[ii] > index) {
+                     lList$pointPart[ii] <- lList$pointPart[ii] -
+                       1
+                   }
+                 }
+             }
+             else if (element == maxValue) {
+                 index <- which(lList$dataPart == maxValue)
+                 lList$pointPart[lList$pointPart == index] <- -1
+             }
+             else {
+                 index <- which(lList$dataPart == element)
+                 currentPoint <- lList$pointPart[lList$dataPart ==
+                   element]
+                 lList$pointPart[lList$pointPart == index] <- currentPoint
+                 for (ii in 1:length(lList$pointPart)) {
+                   if (lList$pointPart[ii] > index) {
+                     lList$pointPart[ii] <- lList$pointPart[ii] -
+                       1
+                   }
+                 }
+             }
+             lList$dataPart <- lList$dataPart[-1 * index]
+             lList$pointPart <- lList$pointPart[-1 * index]
+             return(lList)
+         }
+         else if (length(which(lList$dataPart == element)) ==
+             0) {
+             cat("The element is not in the linked list! ")
+         }
+         else {
+             cat("There are more than one element having the value provided!")
+         }
+     }
+     else {
+         cat("The element or the linked list is not numeric! ")
+     }
+ }
```