# Introduction to computational programming

## Chapter 5 Exercise

### Bifurcations and second-order systems

### Solutions

David M. Rosenberg

University of Chicago

Committee on Neurobiology

Version control information:
Last changed date: 2009-11-17 13:51:18 -0600 (Tue, 17 Nov 2009)
Last changes revision:244
Version: Revision 244
Last changed by: David M. Rosenberg

November 17, 2009

## Part I

# Solutions

1. Explore the dynamics of the discrete logistic model $X_{t+1} = rX_t(1 - X_t)$

   (a) Use your code from the chapter 1 lab to solve the logistic model for 500 iterations, given any initial value and any value of the parameter $r$.

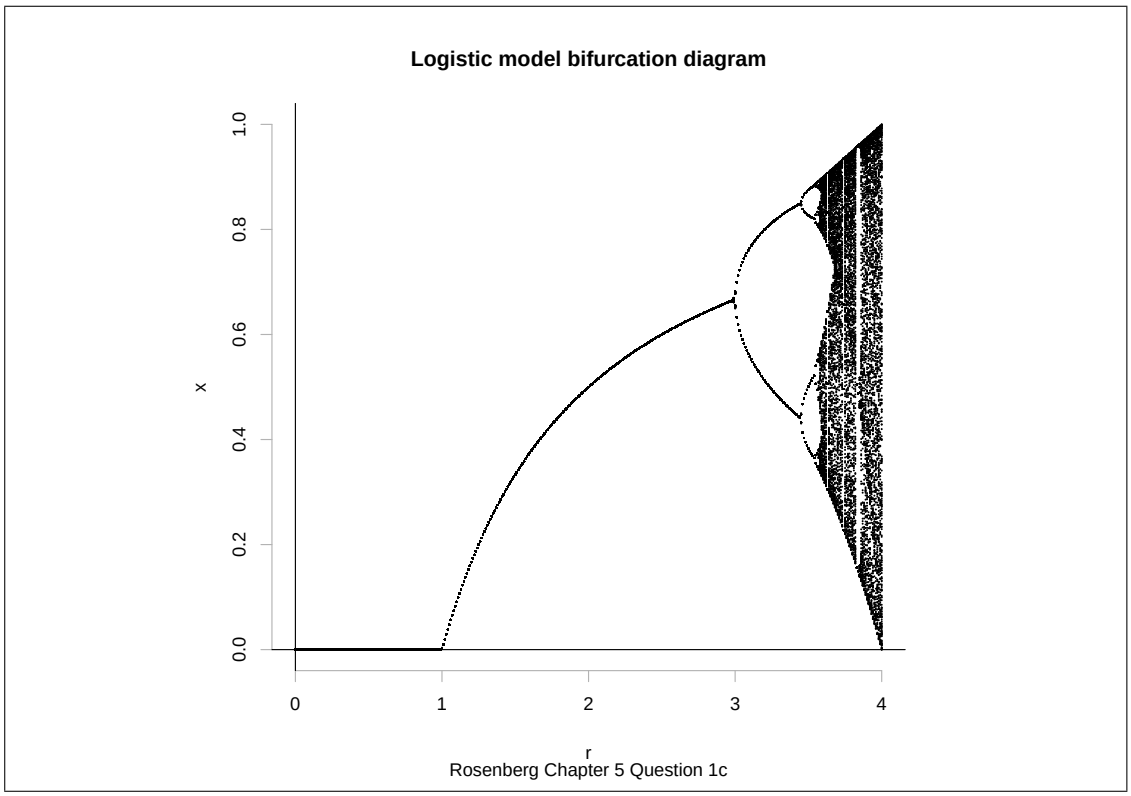
```
> iterLogistic1 <- function(r, x0, nIter=500) {
+   xVals <- numeric(length=nIter+1);
+   xVals[1] <- x0;
+   for (ii in 1:nIter) {
+     xVals[ii + 1] <- r * xVals[ii] * (1 - xVals[ii]);
+   }
+   return(xVals);
+ }
```

   (b) Write a "wrapper" code that will call your code to solve the logistic model for a range of values of $r$ between $r = 0$ and $r = 4$ (with a reasonable step size around 0.01)

```
> multIterLogistic1 <- function(rRange, x0, rStep=0.01, nIter=500) {
+   result <- list();
+   rVals <- seq(rRange[1], rRange[2], rStep)
+   for (jj in 1:length(rVals)) {
+     result[[jj]] <- list(r=rVals[jj], x=iterLogistic1(rVals[jj], x0, nIter));
+   }
+   return(result);
+ }
```

(b) For each value of $r$, discard the first 200 iterations, and plot the rest on a bifurcation diagram.

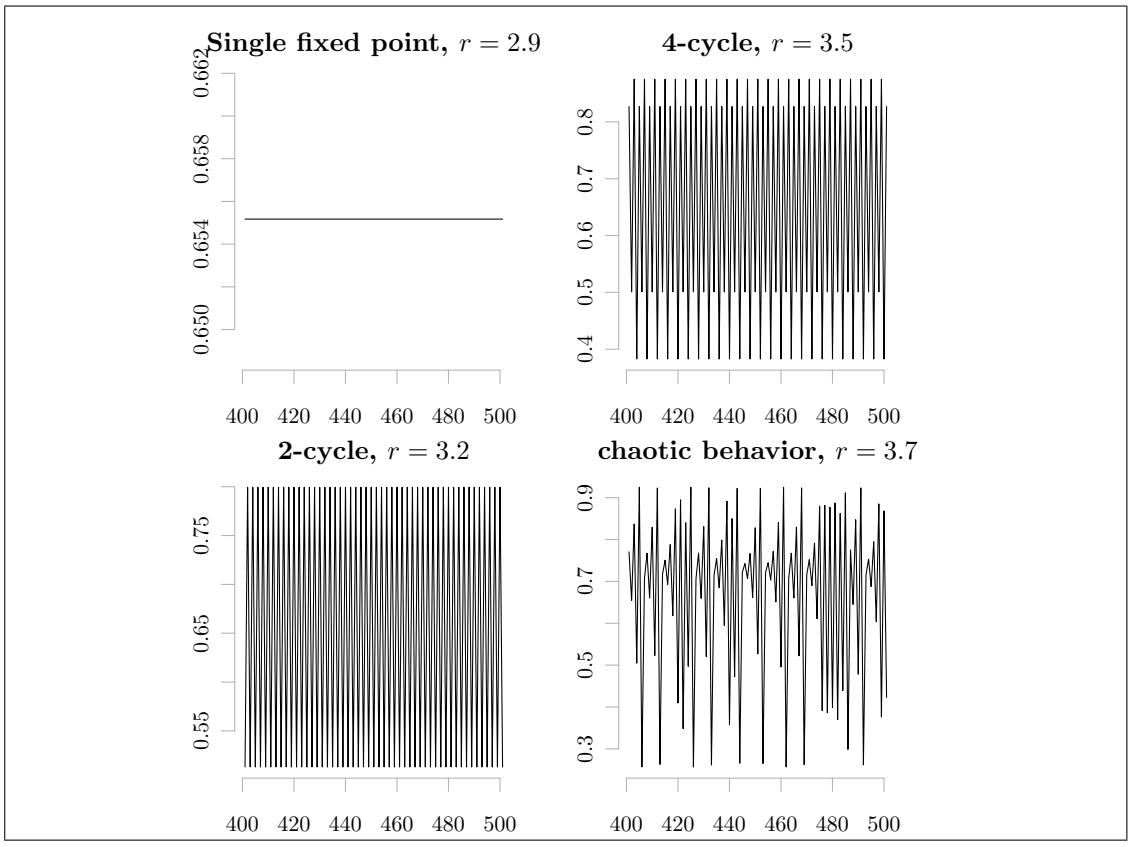
```
> xrVals <- multIterLogistic1(c(0, 4), x0=0.01);
> plot(c(0,4), c(0, 1), type='n', bty='n', fg=grey(0.7),
+      xlab=expression(r), ylab=expression(x),
+      sub='Rosenberg Chapter 5 Question 1c',
+      main='Logistic model bifurcation diagram');
> abline(h=0, v=0);
> for(ii in 1:length(xrVals)) {
+   soln <- xrVals[[ii]];
+   points(rep(soln$r, 301), soln$x[201:501], pch=19, cex=0.1);
+ }
```




(c) Pick a few representative values of $r$ and produce plots of the solutions over time (after discarding

the first 200 iterations) for qualitatively different types of behavior (single fixed point, two cycle, n-cycle, chaos).

```
> rVals <- c(2.9, 3.2, 3.5, 3.7);
> mains=paste(c('Single fixed point, $r =', '2-cycle, $r =',
+               '4-cycle, $r =', 'chaotic behavior, $ r ='),
+       as.character(rVals), c('$', '$', '$', '$'));
> x0 <- 0.1;
> layout(matrix(c(1:4), nrow=2));
> for (ii in 1:4) {
+   r <- iterLogistic1(rVals[ii], x0);
+   plot(401:501, r[401:501], type='l', main=mains[ii], xlab='$t$',
+        ylab='$x$', fg=grey(0.7), bty='n');
+   abline(h=0, v=0);
+ }
```




2. Consider the following Leslie population model:

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

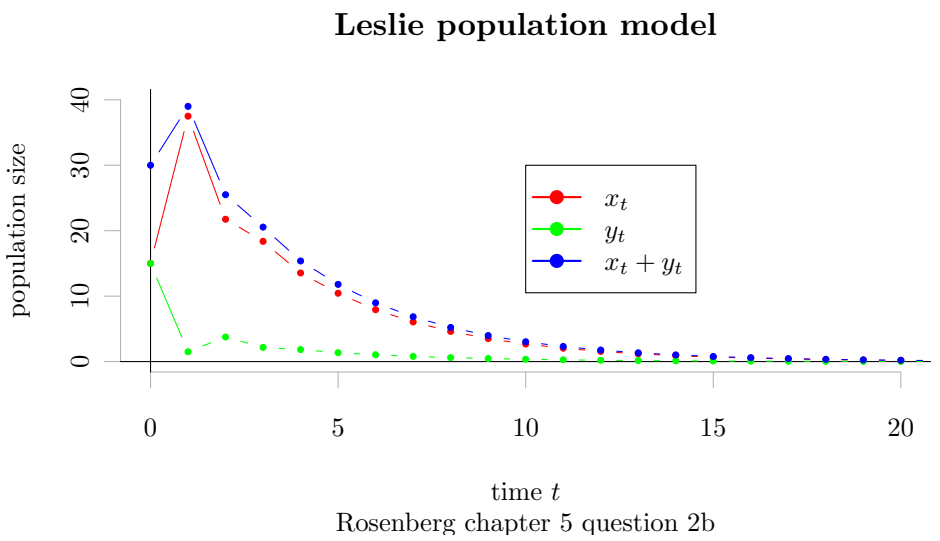
(a) Explain the meaning of the variables $x$ and $y$, and what the numbers in the matrix represent.

The variables $x_t, y_t, x_{t+1}$ and $y_{t+1}$ in the expression represent the number of individuals present in a population of age class 1 ($x$'s) and age class 2 ($y$'s) at time points $t$ and $t+1$. The first row on the $2 \times 2$ matrix represents the fecundity of members of these two age classes and the bottom row represents the probability of individuals of each class advancing to the next (as opposed to dying).

(b) Write code to propagate the population vector, starting with some initial value $x_0$ and $y_0$.

```
> runPopSim <- function(modelMatrix, x0, y0, nSteps=200) {
+   xyVals <- matrix(numeric((nSteps + 1) * 2), nrow=2);
+   xyVals[c(1, 2), 1] <- c(x0, y0);
+   for (ii in 1:nSteps) {
+     xyVals[c(1, 2), ii + 1] <- modelMatrix %*% xyVals[c(1, 2), ii]
+   }
+   return(xyVals);
+ }
> lesMat1 <- matrix(c(0.5, 2, 0.1, 0), nrow=2, byrow=2);
> res <- runPopSim(lesMat1, 15, 15, nSteps=30);
> res;

     [,1] [,2]  [,3]   [,4]    [,5]     [,6]     [,7]      [,8]      [,9]
[1,]   15 37.5 21.75 18.375 13.5375 10.44375 7.929375 6.0534375 4.6125938
[2,]   15  1.5  3.75  2.175  1.8375  1.35375 1.044375 0.7929375 0.6053438
         [,10]     [,11]     [,12]     [,13]     [,14]     [,15]      [,16]
[1,] 3.5169844 2.6810109 2.0439023 1.5581534 1.1878571 0.9055592 0.69035105
[2,] 0.4612594 0.3516984 0.2681011 0.2043902 0.1558153 0.1187857 0.09055592
         [,17]      [,18]      [,19]      [,20]      [,21]      [,22]      [,23]
[1,] 0.5262874 0.40121390 0.30586442 0.23317499 0.1777604 0.13551519 0.10330967
[2,] 0.0690351 0.05262874 0.04012139 0.03058644 0.0233175 0.01777604 0.01355152
         [,24]       [,25]       [,26]       [,27]       [,28]       [,29]
[1,] 0.07875787 0.060040871 0.045772010 0.034894179 0.026601492 0.020279582
[2,] 0.01033097 0.007875787 0.006004087 0.004577201 0.003489418 0.002660149
         [,30]       [,31]
[1,] 0.015460089 0.011785961
[2,] 0.002027958 0.001546009
```

```
> y <- list(res[1, ], res[2, ], res[1, ] + res[2, ]);
> x <- 0:30;
> plot(c(0, 20), c(0, 40), type='n', bty='n', fg=grey(0.7),
+      xlab='time $t$', ylab='population size',
+      sub='Rosenberg chapter 5 question 2b',
+      main='\\textbf{Leslie population model}');
> cols <- rainbow(n=3);
> abline(h=0, v=0);
> for (ii in 1:3) {
+   lines(x, y[[ii]], col=cols[ii], type='b', pch=19, cex=0.5);
+ }
> legend(10, 30, lwd=2, col=cols, legend=c('$x_t$', '$y_t$', '$x_t + y_t$'),
+        pch=19);
```




(c) Starting at different initial conditions, describe what population distribution the model converges to.

```
> for (x0 in c(0, 2^(c(0:1) * 2))) {
+   for (y0 in c(0, 2^(c(0:1) * 2))) {
+     res <- runPopSim(lesMat1, x0, y0);
+     total <- sum(res[, 200]);
+     ratio <- sum(res[, 200]) / sum(res[, 199]);
+     if ( total == 0 ) {
+       cat(sprintf("For an initial population of x0=%d and y0=%d, the population
     is extinct.\n", x0, y0));
+     } else {
+       cat(sprintf("For an Initial population of x0=%d and y0=%d the total
    population after 200 iterations is %s and the growth rate is %s.\n", x0, y0,
    as.character(total), as.character(ratio)));
+     }
+   }
+ }

For an initial population of x0=0 and y0=0, the population is extinct.
For an Initial population of x0=0 and y0=1 the total population after 200
    iterations is 7.80570475103043e-24 and the growth rate is 0.76234753829798.
For an Initial population of x0=0 and y0=4 the total population after 200
    iterations is 3.12228190041217e-23 and the growth rate is 0.76234753829798.
For an Initial population of x0=1 and y0=0 the total population after 200
    iterations is 2.97532990081445e-24 and the growth rate is 0.76234753829798.
For an Initial population of x0=1 and y0=1 the total population after 200
    iterations is 1.07810346518449e-23 and the growth rate is 0.76234753829798.
For an Initial population of x0=1 and y0=4 the total population after 200
    iterations is 3.41981489049362e-23 and the growth rate is 0.76234753829798.
For an Initial population of x0=4 and y0=0 the total population after 200
    iterations is 1.19013196032578e-23 and the growth rate is 0.76234753829798.
For an Initial population of x0=4 and y0=1 the total population after 200
    iterations is 1.97070243542882e-23 and the growth rate is 0.76234753829798.
For an Initial population of x0=4 and y0=4 the total population after 200
    iterations is 4.31241386073795e-23 and the growth rate is 0.76234753829798.
```

(d) Estimate the largest eigenvalue $\lambda$ from the convergence behavior.

From this behavior, I estimate that $\lambda_1 = 0.7623475$.

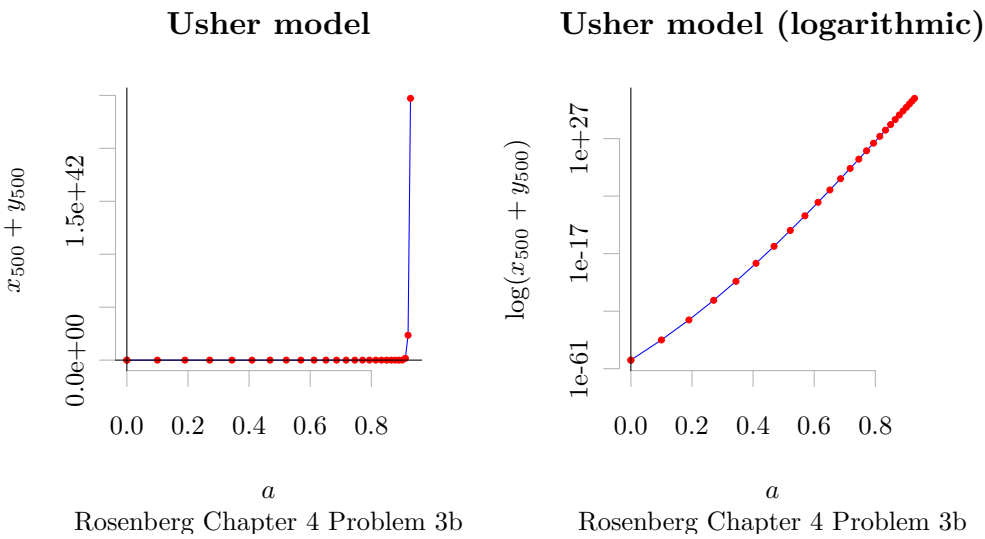
3. Modify the population model in the previous problem by adding a parameter $a$ instead of the zero in the matrix

   (a) Explain the meaning of the parameter $a$.

Adding a non-zero $a$ to the matrix transforms the model into a *Usher model*. Here the $a$ would represent the proportion of individuals in age class 2 who survive each iteration.

   (b) Write code to explore the convergence behavior of the population model, as you vary $a$ from near 0 to progressively larger values, and report the equilibrium population distribution and the largest eigenvalue $\lambda$.

```
> aVals <- c(0, 1 - (0.90 ^ (0:25)))
> uMat <- lesMat1;
> totals <- matrix(numeric(length(aVals) * 9), ncol=9)
> xyVals <- list(c(1, 1), c(1, 2), c(1, 4),
+                c(2, 1), c(2, 2), c(2, 4),
+                c(4, 1), c(4, 2), c(4, 4))
> for (ii in 1:length(aVals)) {
+   uMat[2, 2] <- aVals[ii];
+   for (jj in 1:length(xyVals)) {
+     res <- runPopSim(uMat, xyVals[[jj]][1], xyVals[[jj]][2],
+                                nSteps=500);
+     totals[ii, jj] <- sum(res[, 500]);
+   }
+ }
> res <- data.frame(aVals, totals);
> colnames(res) <- c('a', "x0=1, y0=1", "x0=1, y0=2", "x0=1, y0=4",
+             "x0=2, y0=1", "x0=2, y0=2", "x0=2, y0=4",
+             "x0=4, y0=1", "x0=4, y0=2", "x0=4, y0=4")
> fullTable <- res;       ## Complete table on last page
> layout(matrix(c(1, 2), nrow=1));
> plot(aVals, res[, 10], type='l', bty='n', fg=grey(0.7),
+      xlab='$a$', ylab='$\\lambda_1$',
+      main='Usher model',
+      sub='Rosenberg Chapter 4 Problem 3b', col='blue');
> points(aVals, res[, 10], pch=19, col='red', cex=0.5);
> abline(h=0, v=0);
> plot(aVals, res[, 10], type='l', bty='n', fg=grey(0.7),
+      xlab='$a$', ylab='$\\lambda_1$', log='y',
+      main='Usher model ($\\log \\lambda_1$)',
+      sub='Rosenberg Chapter 4 Problem 3b', col='blue');
> points(aVals, res[, 10], pch=19, col='red', cex=0.5);
> abline(h=0, v=0);
```

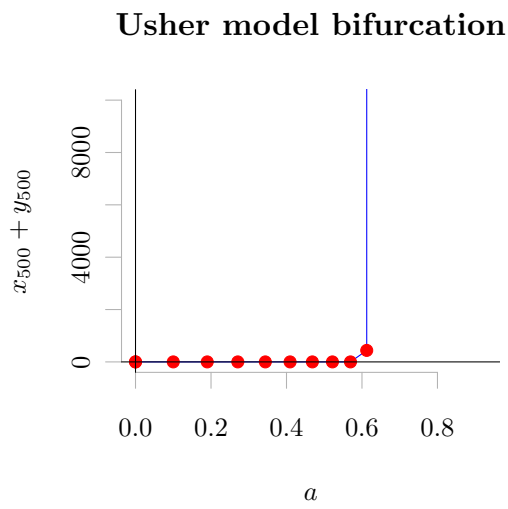



Rosenberg Chapter 4 Problem 3b

Rosenberg Chapter 4 Problem 3b

(c) Report the approximate value of $a$ at which a qualitative change in the solution occurs (bifurcation point). Produce plots of representative solutions on both sides of the bifurcation value of $a$. What is the value of the largest eigenvalue $\lambda$ at the bifurcation point?

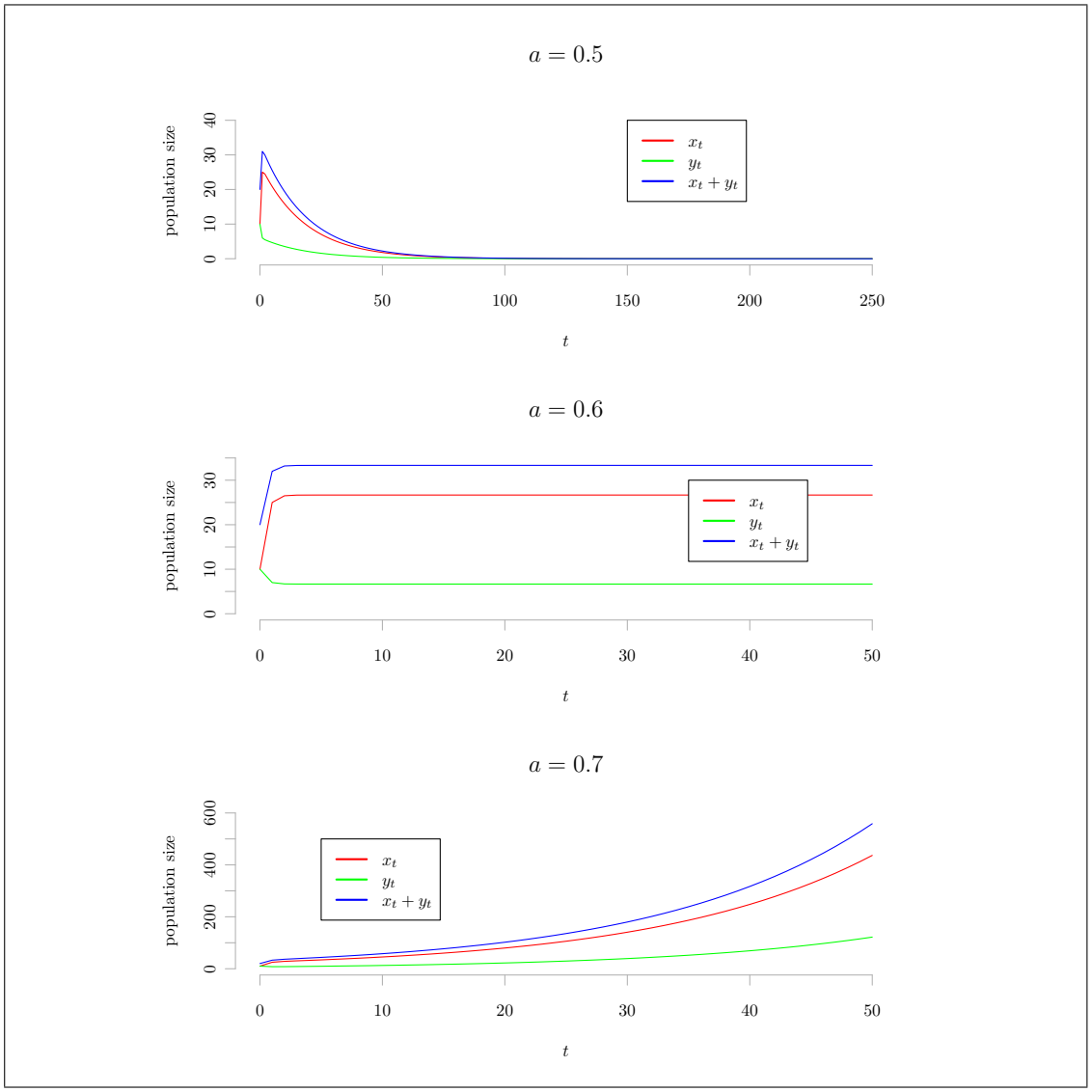
```
> uMat[2, 2] <- 0.6;
> eigen(uMat);

$values
[1] 1.0 0.1

$vectors
           [,1]       [,2]
[1,] -0.9701425 -0.9805807
[2,] -0.2425356  0.1961161

> plot(aVals, res[, 10], type='l', bty='n', fg=grey(0.7),
+      xlab='$a$', ylab='$x_{500} + y_{500}$', ylim=c(0, 10000),
+      main='Usher model bifurcation',
+      sub='Rosenberg Chapter 4 Problem 3c', col='blue');
> points(aVals, res[, 10], pch=19, col='red');
> abline(h=0, v=0);
```




Rosenberg Chapter 4 Problem 3c

The value of the largest eigenvalue at the bifurcation point $a = 0.6$ is $\lambda_1 = 1.000$.

```
> uMatExtinct <- uMatInflate <- uMat;
> uMatInflate[2, 2] <- 0.7;
> uMatExtinct[2, 2] <- 0.5;
> layout(matrix(c(1, 2, 3), nrow=3));
> exPop <- runPopSim(uMatExtinct, 10, 10, 250);
> inPop <- runPopSim(uMatInflate, 10, 10, 50);
> stPop <- runPopSim(uMat, 10, 10, 50);
```




4. *Enrichment problem. Extra credit only.* A linked list is a ordered data structure consisting of an arbitrary number of elements called *nodes*. Each *node* contains a data part and a *pointer* to the next element in the list. Consider the following example.

```
> dataPart <- c('D', 'a', 'v', 'i', 'd');
> pointPart <- c( 5, 1, -1, 3, 4);
> firstElement <- 2
> lList <- list(dataPart=dataPart, pointPart=pointPart, firstElement=2);
```

This linked list structure is ordered alphabetically. The advantage of linked lists is the ease with which an element is added. For example, to add an element to the list while maintaining the alphabetical ordering:

```
> newElement <- 'b'
> lList$dataPart <- c(lList$dataPart, newElement);
> lList$pointPart[2] <- 6;
> lList$pointPart <- c(lList$pointPart, 6);
```

(a) Write a function which takes an (unsorted) vector of type numeric and returns a linked list containing those values.

```
> buildLinkedList <- function(rawData) {
+   lList <- list();
+   lList$dataPart <- sort(rawData);
+   lList$pointPart <- c(2:length(rawData), -1);
+   lList$firstElement <- 1;
+   return(lList);
+ }
> rawData <- sample(-1000:1000, 20);
> llInt <- buildLinkedList(rawData);
```

```
> llInt;

$dataPart
 [1] -996 -985 -919 -772 -754 -590 -497 -443 -390 -347 -179 -160   -1   17   66
[16]  168  193  268  719  874

$pointPart
 [1]  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 -1

$firstElement
[1] 1
```

(b) Write a function which takes a linked list and adds a single element to it (preserving ordering).

```
> addElement <- function(lList, element) {
+   pos <- lList$firstElement;
+   lList$dataPart <- c(lList$dataPart, element);
+   if (element <  lList$dataPart[pos]) {
+     lList$pointPart <- c(lList$pointPart, lList$firstElement);
+     lList$firstElement <- length(lList$dataPart);
+   } else {
+     while((lList$dataPart[pos] < element) && (lList$pointPart[pos] != -1)) {
+       pos <- lList$pointPart[pos];
+     }
+     if (lList$pointPart[pos] == -1) {
+       lList$pointPart[pos] <- length(lList$dataPart);
+       lList$pointPart <- c(lList$pointPart, -1);
+     } else {
+       lList$pointPart[lList$pointPart == pos] <- length(lList$dataPart);
+       lList$pointPart <- c(lList$pointPart, pos)
+     }
+   }
+   return(lList);
+ }
> printOrdered <- function(lList) {
+   p <- lList$firstElement;
+   v <- lList$dataPart[p];
+   while (lList$pointPart[p] != -1) {
+     p <- lList$pointPart[p];
+     v <- c(v, lList$dataPart[p]);
+   }
+   return(v);
+ }
```

```
> for (ii in 1:20) {
+   llInt <- addElement(llInt, sample(-1000:1000, 1));
+ }
> printOrdered(llInt);

 [1] -996 -985 -919 -794 -772 -762 -754 -590 -550 -507 -497 -480 -443 -428 -405
[16] -390 -362 -354 -347 -304 -290 -257 -236 -179 -160 -108   -1   17   42   66
[31]  163  168  193  268  305  559  595  719  874  938

> llInt;

$dataPart
 [1] -996 -985 -919 -772 -754 -590 -497 -443 -390 -347 -179 -160   -1   17   66
[16]  168  193  268  719  874   42  163 -428 -304 -354  938 -108 -362 -794  305
[31]  559 -507 -762 -405  595 -550 -480 -257 -290 -236

$pointPart
 [1]  2  3 29 33  6 36 37 23 28 24 12 27 14 21 22 17 18 30 20 26 15 16 34 39 10
[26] -1 13 25  4 31 35  7  5  9 19 32  8 40 38 11

$firstElement
[1] 1
```

(c) Write a function which takes a linked list and removes a single element.

```
> removeElement <- function(lList, element) {
+   if(lList$dataPart[lList$firstElement] == element) {
+     lList$firstElement <- lList$pointPart[lList$firstElement];
+   } else {
+     idx <- (1:length(lList$dataPart))[lList$dataPart == element][1]
+     nextId <- lList$pointPart[idx];
+     prevId <- (1:length(lList$pointPart))[lList$pointPart == idx];
+     lList$pointPart[prevId] <- nextId
+   }
+   return(lList);
+ }
> removeUnusedData  <- function(lList) {
+   unusedIds <- setdiff(1:length(lList$dataPart), unique(c(lList$firstElement,
    lList$pointPart)));
+   while (length(unusedIds) > 0) {
+     idx <- unusedIds[1];
+     unusedIds <- unusedIds[-1];
+     unusedIds[ unusedIds > idx ] <- unusedIds[ unusedIds > idx ] - 1;
+     if (lList$firstElement > idx) {
+       lList$firstElement <- lList$firstElement - 1;
+     }
+     lList$pointPart[ lList$pointPart > idx ] <- lList$pointPart[ lList$
    pointPart > idx ] - 1;
+     lList$pointPart <- lList$pointPart[ (1:length(lList$pointPart)) != idx ];
+     lList$dataPart <- lList$dataPart[ (1:length(lList$dataPart)) != idx ];
+   }
+   return(lList);
+ }
```

```
> llInt <- removeElement(llInt, min(llInt$dataPart));
> for (ii in 1:50) {
+   llInt <- removeElement(llInt, sample(llInt$dataPart, 1));
+ }
> llInt;

$dataPart
 [1] -996 -985 -919 -772 -754 -590 -497 -443 -390 -347 -179 -160   -1   17   66
[16]  168  193  268  719  874   42  163 -428 -304 -354  938 -108 -362 -794  305
[31]  559 -507 -762 -405  595 -550 -480 -257 -290 -236

$pointPart
 [1]  2 23 23 23 23 23 23 23 24 24 21 21 21 21 35 35 35 35 20 -1 15 35 24 40 24
[26] -1 21 24 23 35 35 23 23 24 19 23 23 40 40 11

$firstElement
[1] 2

> printOrdered(llInt);

 [1] -985 -428 -304 -236 -179   42   66  595  719  874
```

```
> llInt <- removeUnusedData(llInt);
> printOrdered(llInt);

 [1] -985 -428 -304 -236 -179   42   66  595  719  874

> llInt;

$dataPart
 [1] -985 -179   66  719  874   42 -428 -304  595 -236

$pointPart
 [1]  7  6  9  5 -1  3  8 10  4  2

$firstElement
[1] 1
```

```
> fullTable;
```

| | a | x0=1, y0=1 | x0=1, y0=2 | x0=1, y0=4 | x0=2, y0=1 | x0=2, y0=2 |
|---|---|---|---|---|---|---|
| 1 | 0.0000000 | 4.770469e−59 | 8.224394e−59 | 1.513224e−58 | 6.087015e−59 | 9.540939e−59 |
| 2 | 0.0000000 | 4.770469e−59 | 8.224394e−59 | 1.513224e−58 | 6.087015e−59 | 9.540939e−59 |
| 3 | 0.1000000 | 2.428333e−51 | 4.233854e−51 | 7.844896e−51 | 3.051146e−51 | 4.856667e−51 |
| 4 | 0.1900000 | 1.133423e−43 | 1.995894e−43 | 3.720837e−43 | 1.404374e−43 | 2.266845e−43 |
| 5 | 0.2710000 | 3.668103e−36 | 6.515855e−36 | 1.221136e−35 | 4.488453e−36 | 7.336205e−36 |
| 6 | 0.3439000 | 6.468380e−29 | 1.157764e−28 | 2.179615e−28 | 7.827503e−29 | 1.293676e−28 |
| 7 | 0.4095100 | 5.146957e−22 | 9.273202e−22 | 1.752569e−21 | 6.167670e−22 | 1.029391e−21 |
| 8 | 0.4685590 | 1.619848e−15 | 2.935081e−15 | 5.565548e−15 | 1.924463e−15 | 3.239696e−15 |
| 9 | 0.5217031 | 1.864568e−09 | 3.395102e−09 | 6.456171e−09 | 2.198603e−09 | 3.729137e−09 |
| 10 | 0.5695328 | 7.597105e−04 | 1.389184e−03 | 2.648131e−03 | 8.899474e−04 | 1.519421e−03 |
| 11 | 0.6125795 | 1.098910e+02 | 2.016789e+02 | 3.852549e+02 | 1.279940e+02 | 2.197819e+02 |
| 12 | 0.6513216 | 5.808631e+06 | 1.069413e+07 | 2.046512e+07 | 6.731764e+06 | 1.161726e+07 |
| 13 | 0.6861894 | 1.175282e+11 | 2.169723e+11 | 4.158604e+11 | 1.356124e+11 | 2.350565e+11 |
| 14 | 0.7175705 | 9.637985e+14 | 1.783540e+15 | 3.423023e+15 | 1.107856e+15 | 1.927597e+15 |
| 15 | 0.7458134 | 3.410835e+18 | 6.324971e+18 | 1.215324e+19 | 3.907533e+18 | 6.821669e+18 |
| 16 | 0.7712321 | 5.557214e+21 | 1.032386e+22 | 1.985715e+22 | 6.347781e+21 | 1.111443e+22 |
| 17 | 0.7941089 | 4.444330e+24 | 8.269522e+24 | 1.591991e+25 | 5.063468e+24 | 8.888660e+24 |
| 18 | 0.8146980 | 1.855927e+27 | 3.458117e+27 | 6.662497e+27 | 2.109665e+27 | 3.711855e+27 |
| 19 | 0.8332282 | 4.290990e+29 | 8.005124e+29 | 1.543339e+30 | 4.867848e+29 | 8.581981e+29 |
| 20 | 0.8499054 | 5.801967e+31 | 1.083563e+32 | 2.090297e+32 | 6.570266e+31 | 1.160393e+32 |
| 21 | 0.8649148 | 4.826483e+33 | 9.022432e+33 | 1.741433e+34 | 5.457018e+33 | 9.652966e+33 |
| 22 | 0.8784233 | 2.588001e+35 | 4.841983e+35 | 9.349945e+35 | 2.922021e+35 | 5.176002e+35 |
| 23 | 0.8905810 | 9.334353e+36 | 1.747701e+37 | 3.376233e+37 | 1.052605e+37 | 1.866871e+37 |
| 24 | 0.9015229 | 2.354217e+38 | 4.410797e+38 | 8.523958e+38 | 2.651853e+38 | 4.708433e+38 |
| 25 | 0.9113706 | 4.300871e+39 | 8.062740e+39 | 1.558648e+40 | 4.839874e+39 | 8.601743e+39 |
| 26 | 0.9202336 | 5.875968e+40 | 1.102130e+41 | 2.131198e+41 | 6.606599e+40 | 1.175194e+41 |
| 27 | 0.9282102 | 6.179504e+41 | 1.159604e+42 | 2.242911e+42 | 6.942473e+41 | 1.235901e+42 |

| | x0=2, y0=4 | x0=4, y0=1 | x0=4, y0=2 | x0=4, y0=4 |
|---|---|---|---|---|
| 1 | 1.644879e−58 | 8.720105e−59 | 1.217403e−58 | 1.908188e−58 |
| 2 | 1.644879e−58 | 8.720105e−59 | 1.217403e−58 | 1.908188e−58 |
| 3 | 8.467708e−51 | 4.296771e−51 | 6.102292e−51 | 9.713333e−51 |
| 4 | 3.991788e−43 | 1.946276e−43 | 2.808747e−43 | 4.533690e−43 |
| 5 | 1.303171e−35 | 6.129153e−36 | 8.976905e−36 | 1.467241e−35 |
| 6 | 2.315528e−28 | 1.054575e−28 | 1.565501e−28 | 2.587352e−28 |
| 7 | 1.854640e−21 | 8.209094e−22 | 1.233534e−21 | 2.058783e−21 |
| 8 | 5.870163e−15 | 2.533693e−15 | 3.848926e−15 | 6.479393e−15 |
| 9 | 6.790205e−09 | 2.866671e−09 | 4.397205e−09 | 7.458273e−09 |
| 10 | 2.778368e−03 | 1.150421e−03 | 1.779895e−03 | 3.038842e−03 |
| 11 | 4.033578e+02 | 1.641999e+02 | 2.559879e+02 | 4.395638e+02 |
| 12 | 2.138825e+07 | 8.578032e+06 | 1.346353e+07 | 2.323452e+07 |
| 13 | 4.339446e+11 | 1.717808e+11 | 2.712248e+11 | 4.701130e+11 |
| 14 | 3.567080e+15 | 1.395970e+15 | 2.215711e+15 | 3.855194e+15 |
| 15 | 1.264994e+19 | 4.900929e+18 | 7.815065e+18 | 1.364334e+19 |
| 16 | 2.064772e+22 | 7.928916e+21 | 1.269556e+22 | 2.222886e+22 |
| 17 | 1.653904e+25 | 6.301745e+24 | 1.012694e+25 | 1.777732e+25 |
| 18 | 6.916234e+27 | 2.617139e+27 | 4.219329e+27 | 7.423709e+27 |
| 19 | 1.601025e+30 | 6.021562e+29 | 9.735696e+29 | 1.716396e+30 |
| 20 | 2.167127e+32 | 8.106866e+31 | 1.314053e+32 | 2.320787e+32 |
| 21 | 1.804486e+34 | 6.718086e+33 | 1.091404e+34 | 1.930593e+34 |
| 22 | 9.683965e+35 | 3.590060e+35 | 5.844041e+35 | 1.035200e+36 |
| 23 | 3.495403e+37 | 1.290943e+37 | 2.105209e+37 | 3.733741e+37 |
| 24 | 8.821594e+38 | 3.247126e+38 | 5.303706e+38 | 9.416867e+38 |
| 25 | 1.612548e+40 | 5.917879e+39 | 9.679748e+39 | 1.720349e+40 |
| 26 | 2.204261e+41 | 8.067863e+40 | 1.321320e+41 | 2.350387e+41 |
| 27 | 2.319208e+42 | 8.468411e+41 | 1.388495e+42 | 2.471802e+42 |