Laboratory 6 Solutions
Building Two State Variable Models
for Consumer–Resource Dynamics

Introduction

Task 6.1: Two entity models with logistic space-control feedback

For this task, I constructed an general R script that contained the model function and all of the required run information. This script simulates the two-state variable model described in the lab manual (Appendix A). Again, the model equations were

$$\frac{dX_1}{dt} = C - \delta_1 X_1 - \tau_{12} X_2 f(X_2) f(X_1) \qquad (1)$$

$$\frac{dX_2}{dt} = \tau_{12} X_2 f(X_2) f(X_1) - \delta_2 X_2 \qquad (2)$$

Where

$$f(X_2) = \left(1 - \left(1 - \frac{\delta_2}{\tau_{12}}\right)\left(\frac{X_2}{K_2}\right)_+\right)_+, \qquad (3)$$

$$f(X_1) = \frac{X_1}{X_1 + 0.0001} \qquad (4)$$

and the variables and parameters are defined as in Table 1. We assumed 100% efficiency in the conversion of phosphate to phosphorus in phytoplankton.

Task 6.1.1: Forrester Diagram

Figure 1 shows a Forrester type diagram for this assignment. The critical component for this exercise is to recognize that the uptake of phosphorus by the phytoplankton is controlled by the phytoplankton density.

Task 6.1.2: Model Sensitivity to K

For this subtask, I modified my original model program to analyze the sensitivity of the model behavior to the carrying capacity ($K2$; Appendix A.1). I chose to use K values of 2, 5, 7, 9, 12, and 15. The results of these changes appear in Figure 2.

Table 1: Variable and parameter definitions and nominal values for Task 6.1

Name	Symbol	Nominal Value
State Variables		
Phosphorus	X_1	2
Phytoplankton	X_2	0.01
Constant Inputs		
Input of available phosphorus	C	0.5
Specific rate parameters		
Loss from available phosphorus pool	δ_1	0.001
Loss of phosphorus from phytoplankton	δ_2	0.08
Uptake (gross) of phosphorus by phytoplankton	τ_{12}	0.3
Control Parameters		
Maximum density of phytoplankton	K_2	10

Results The graphs in Figure 2 show that when the phytoplankton (consumer) carrying capacity is low, the phytoplankton quickly grow to their environmental or space limited density. In this case, the Phosphorus nutrient (resource) continues to grow without bound because there is a constant input and eventually a constant loss. However, as the phytoplankton carrying capacity increases, their biomass becomes great enough so that their phosphorus consumption (plus the environmental loss) exceeds the value of the rate of resource replenishment (C). Thus, as the carrying capacity increases the effective control on the phytoplankton biomass switches from intraspecific control (competition for space or some set of unspecified resources) modeled by the logistic to a resource limitation or exploitative control. Also, notice that even when the phosphorus concentration is zero, the algae persist. This is because they are immediately consuming all of the incoming phosphorus (C).

Task 6.1.3: Variation in Phytoplankton Initial Density

Results & Discussion Figures 3 suggests that increasing the initial phytoplankton density alters the system dynamics to a small degree. When the initial algal density is low, the phosphorus increases, which is follows by an increase in the algae concentration. The increased algae draw down the phosphorus concentration to zero, at which point the phosphorus input is limiting the phytoplankton growth. Notice that if $X_2(0)$ starts larger than the carrying capacity, then the phosphorus can initially increase because the phytoplankton uptake of the phosphorus is limited by their density dependent feedback. Ultimately, however, phosphorus becomes the limiting factor in all cases shown, and the two variables find the same equilibrium values.

Task 6.1.4: Enrichment Variation

Results & Discussion Figure 4 illustrates that increasing the phosphorus replenishment rate C pushes the system toward a regime in which the phytoplankton are controlled by their own density (intraspecific competition for an unspecified resource like space) and the increase in phosphorus is unchecked. As mentioned, this model broadly mimics what might happen when an ecosystem becomes eutrophic.

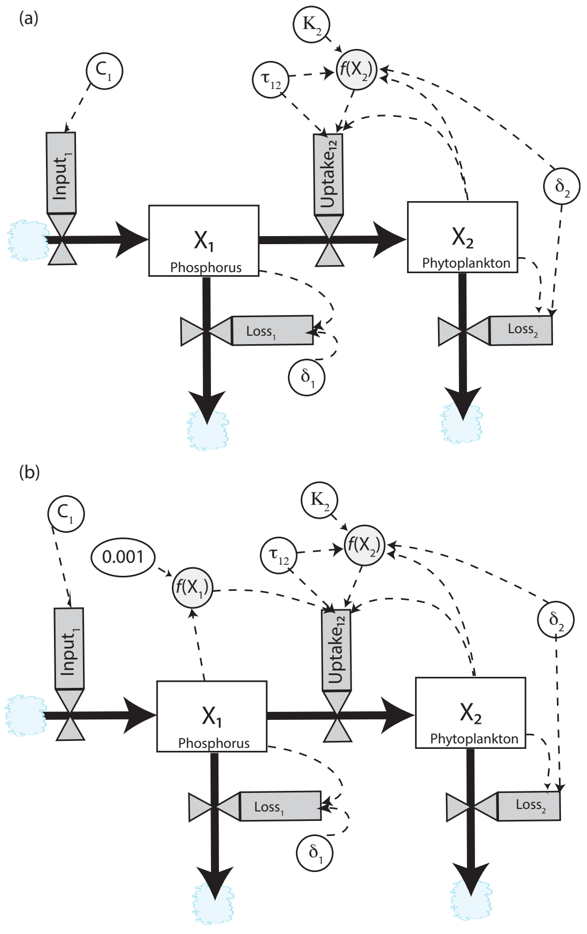

Figure 1: Forrester diagrams for the first task model. (a) shows the control arrangement as described where the model is only recipient controlled and (b) shows the addition of the donor control function ($f(X_1)$) that we added so that the phytoplankton could not consume more phosphorus than was physically available.

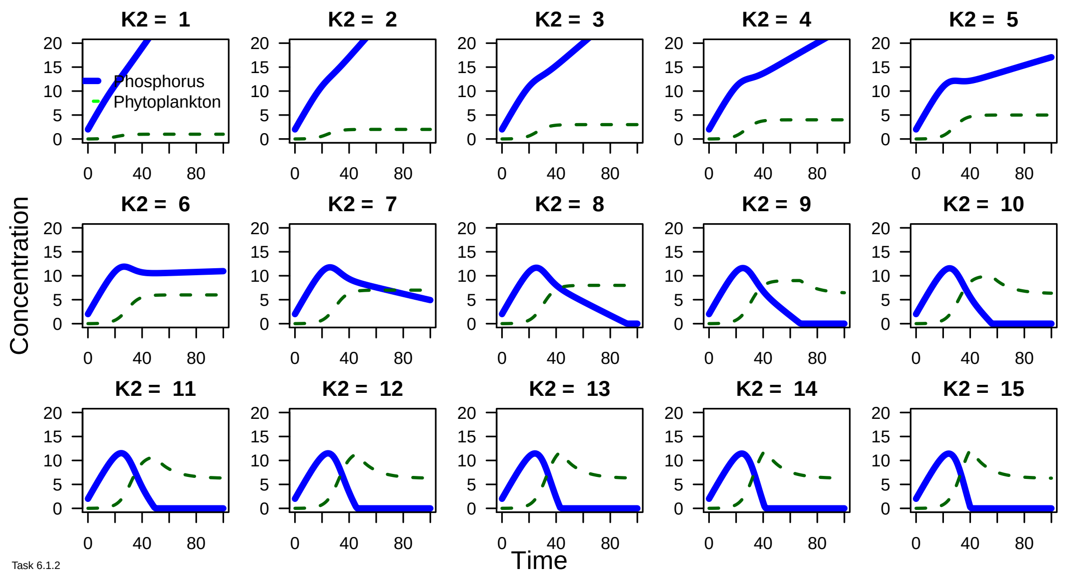

Figure 2: Sensitivity of two-entity model with logistic space-control feedback to the carrying capacity. Each panel shows the system dynamics under a different carrying capacity value ($K_2 = \{2, 5, 7, 9, 12, \text{ and } 15\}$).

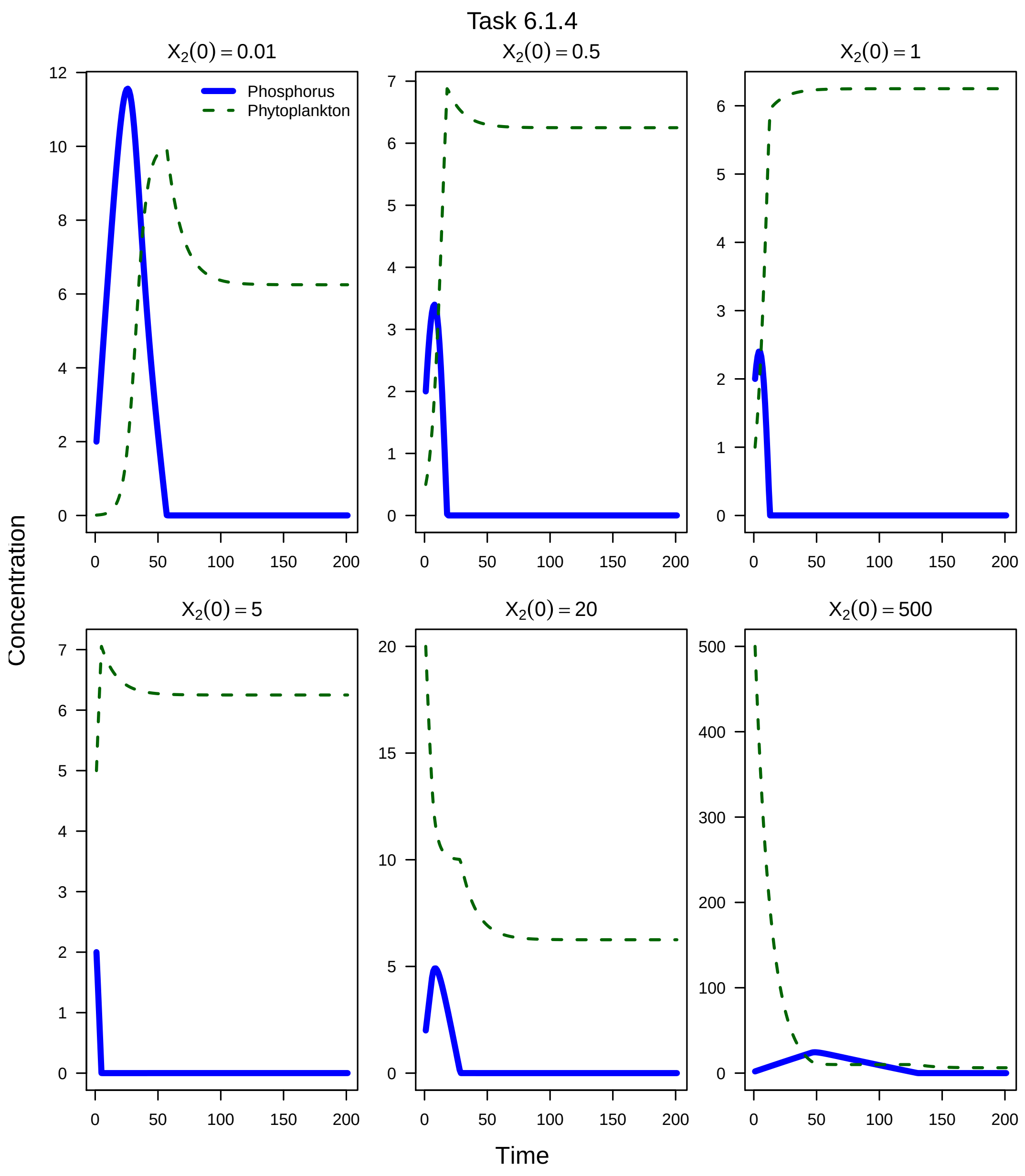

Figure 3: Sensitivity of two-entity model with logistic space-control feedback to the initial value of Phytoplankton (X_2). Each panel in the left column shows the system dynamics under a different initial Phytoplankton biomass ($X_2(0) = \{0.01, 0.5, 1, 5, 20, 500\}$). Panels on the right show the realized behavior of the control function $f(X_2)$.

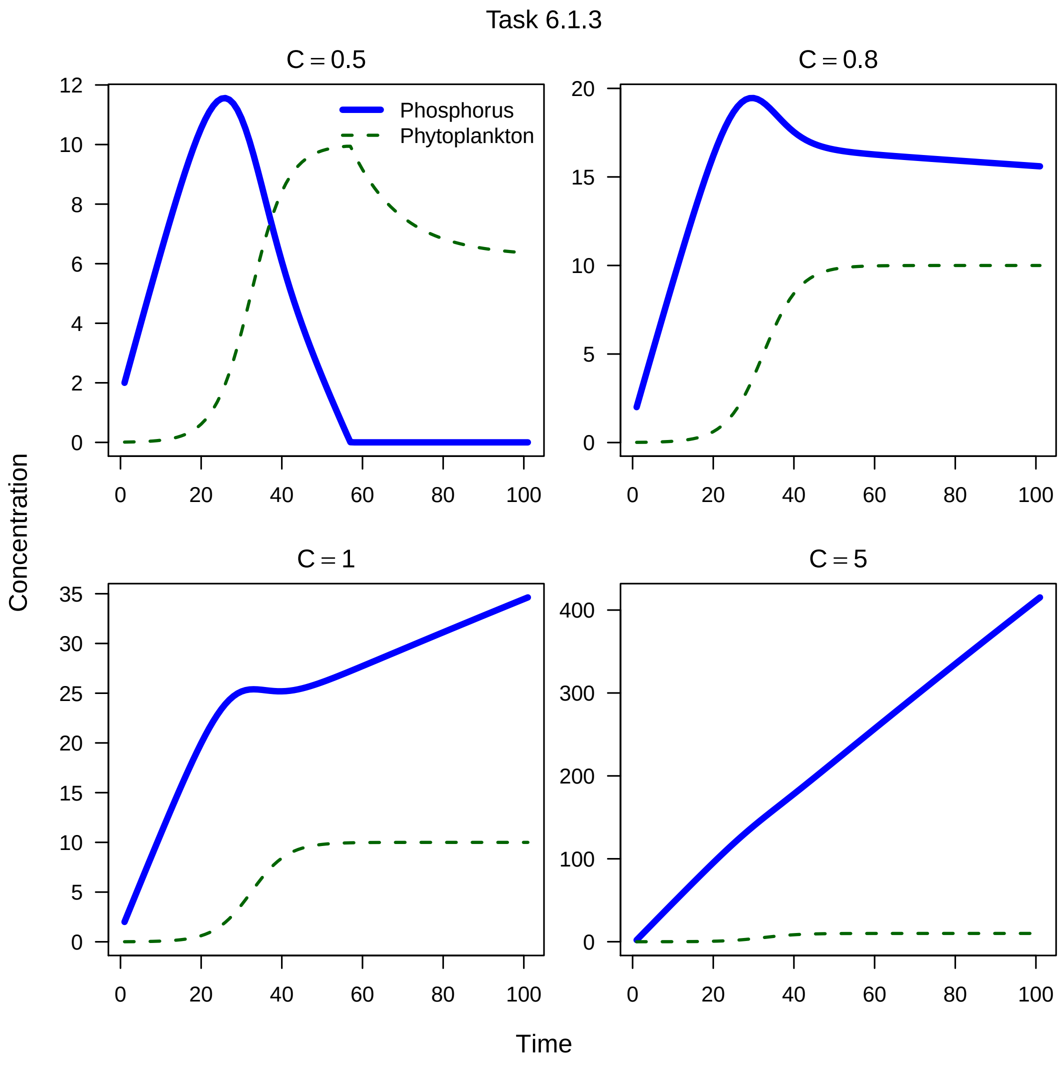

Figure 4: Sensitivity of two-entity model with logistic space-control feedback to the rate of nutrient replenishment. Each panel shows the system dynamics under a different phosphorus replenishment rate ($C = \{0.5, 0.8, 1, 5\}$).

Task 6.2: Two entity models with logistic space-control feedback

For the second task, I again wrote a series of R programs (Appendix B. The model is based on the following equations. Unless otherwise noted, I used the same variables, parameters, and nominal values as shown in Table 1.

Equations

$$\frac{dX_1}{dt} = C - \delta_1 X_1 - \tau_{12} X_2 f(X_1) \tag{5}$$

$$\frac{dX_2}{dt} = \tau_{12} X_2 f(X_1) - \delta_2 X_2 \tag{6}$$

Where $f(X_1)$ can be one of several functional forms

$$f(X_1) = \left(1 - \frac{k_1}{(k_1 + X_1)}\right)_+ \tag{7}$$

$$f(X_1) = \left(1 - \frac{\alpha_{12}}{X_1}\right)_+ \tag{8}$$

$$f(X_1) = \left(a \frac{X_1^b}{X_1^b + k_1^b}\right)_+ \tag{9}$$

Where equation (7) is the half-saturation form of the hyperbolic control function for resource uptake, equation (8) is the refuge form of hyperbolic resource control, and equation (9) is the modified Michaelis–Menten or Monod functional response.

Task 6.2.1: Forrester Diagram

Figure 5 shows a Forrester type diagram for this assignment. The critical component for this subtask is to recognize that the uptake of phosphorus by the phytoplankton is only controlled by the donor.

Task 6.2.2: Hyperbolic Form of the Michaelis–Menton Function

Results & Discussion With this formulation of resource control, a higher 1/2 saturation value k_1 implies that it takes a greater resource concentration for the uptake process to reach its maximal values (Figure 11). As Figure 6 illustrates, this translates into a faster initial accumulation of phosphorus in the system, a larger steady-state value, and a slower growth of the phytoplankton. Notice that when k_1 is 5 or 10, the steady state quantity of phytoplankton is larger than phosphorus, but this reverses when k_1 exceeds about 15. This is more clear in Fig. 7, which shows the equilibrium values of the system for a range of parameter values.

Task 6.2.3: Hyperbolic Form of the Michaelis–Menten Function with Refuge

Results & Discussion The refuge form of the Michaelis–Menten control function prevents the phytoplankton form consuming the phosphorus until a certain threshold concentration is achieved α_{12}. Like the increase in k_1 in the 1/2 saturation form, an increase in α_{12}, shown in Figure 8, delays the phosphorus and phytoplankton peak concentrations. The maximum and steady state

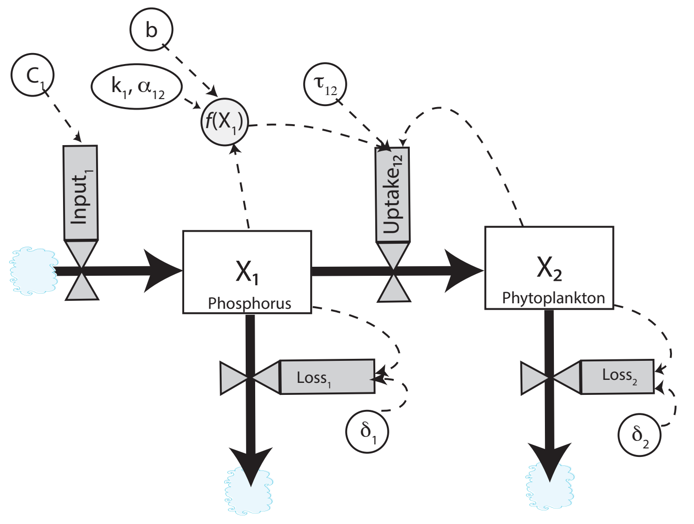

Figure 5: Forrester diagram for the model in the second task. Notice that Uptake$_{12}$ is now only donor controlled.

phosphorus quantities generally increase with α_{12}, and the steady-state value of the phytoplankton declines slowly.

When α_{12} was set to 0.2 or less (not shown), many of the model values became undefined. This is why some of your plots stop before the model time is completed. Does it make sense for α_{12} to be less than 1?

Task 6.2.4: Generalized Form of the Michaelis–Menten Function

Increasing the exponent b in the generalized form of the Michaelis–Menton control function caused the peak quantities to occur earlier (Figure 10). When all else was equal, the increase in b caused uptake to be smaller at lower resource concentration values, but the transition to reduced uptake occurred more rapidly as the 1/2 saturation constant was approached. This is evident in the realized control function plots in Figure 10 and in the potential control function behavior illustrated in Figure 11.

Task 6.2.5: Classifying the Controls

Figure 11 shows the *potential* behavior of the three control functions we investigated in the second task. This shows that the first two functions have a Holling Type II functional response, while the third function can be made to have a sigmoid or Holling Type III response when b is greater than unity.

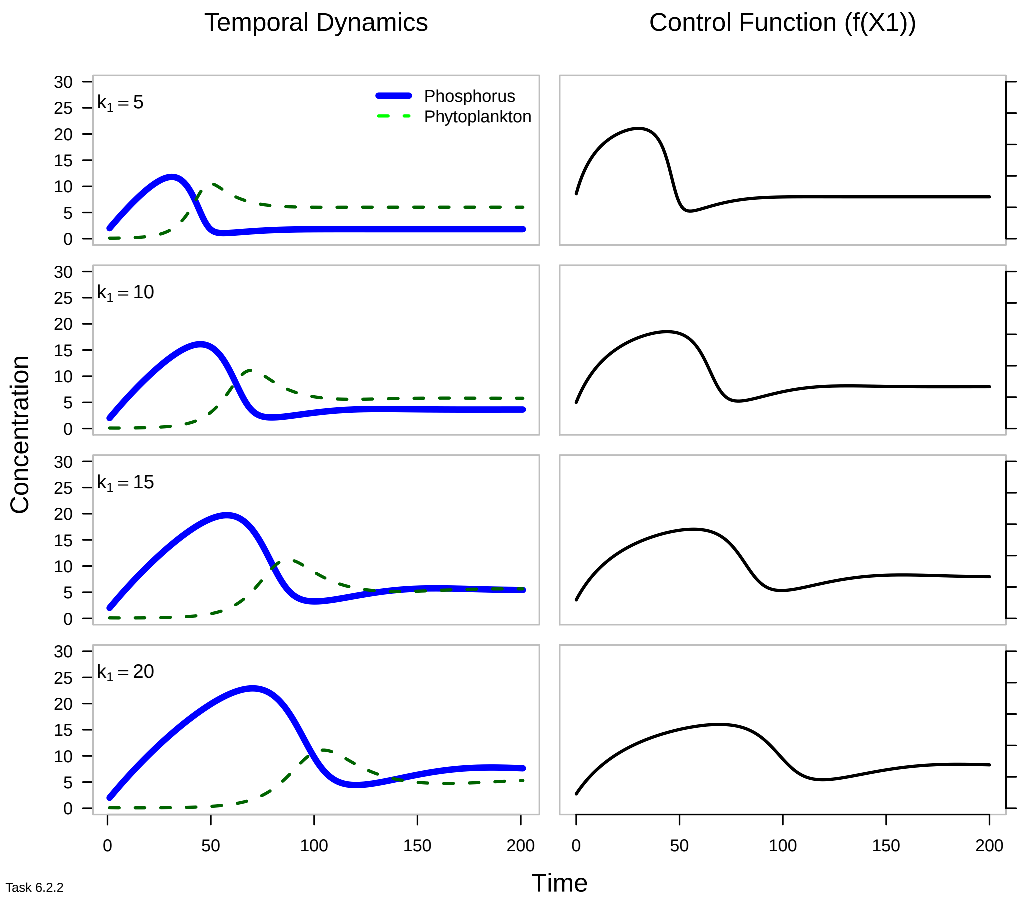

Figure 6: Hyperbolic form of the 1/2 saturation control used to control phytoplankton (X_2) uptake of phosphorus (X_1). This is an example of donor control of consumption (exploitative control). Each panel in the left column shows the system dynamics under a different 1/2 saturation constant ($k_1 = \{5, 10, 15, 20\}$). Panels on the right show the realized behavior of the control function $f(X_1)$

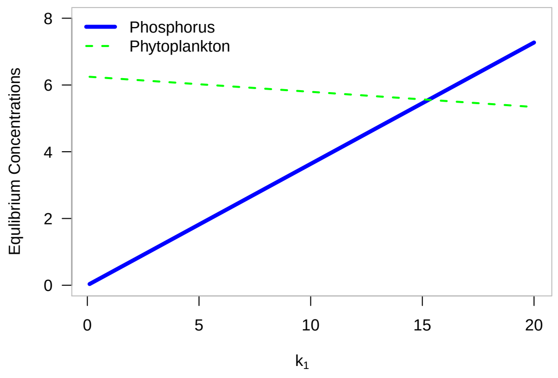

Figure 7: Change in equilibrium values of the two compartments as the parameter k_1 changes from 0 to 20. Notice that the dominant (most abundant) compartment switches as k_1 exceeds approximately 15.

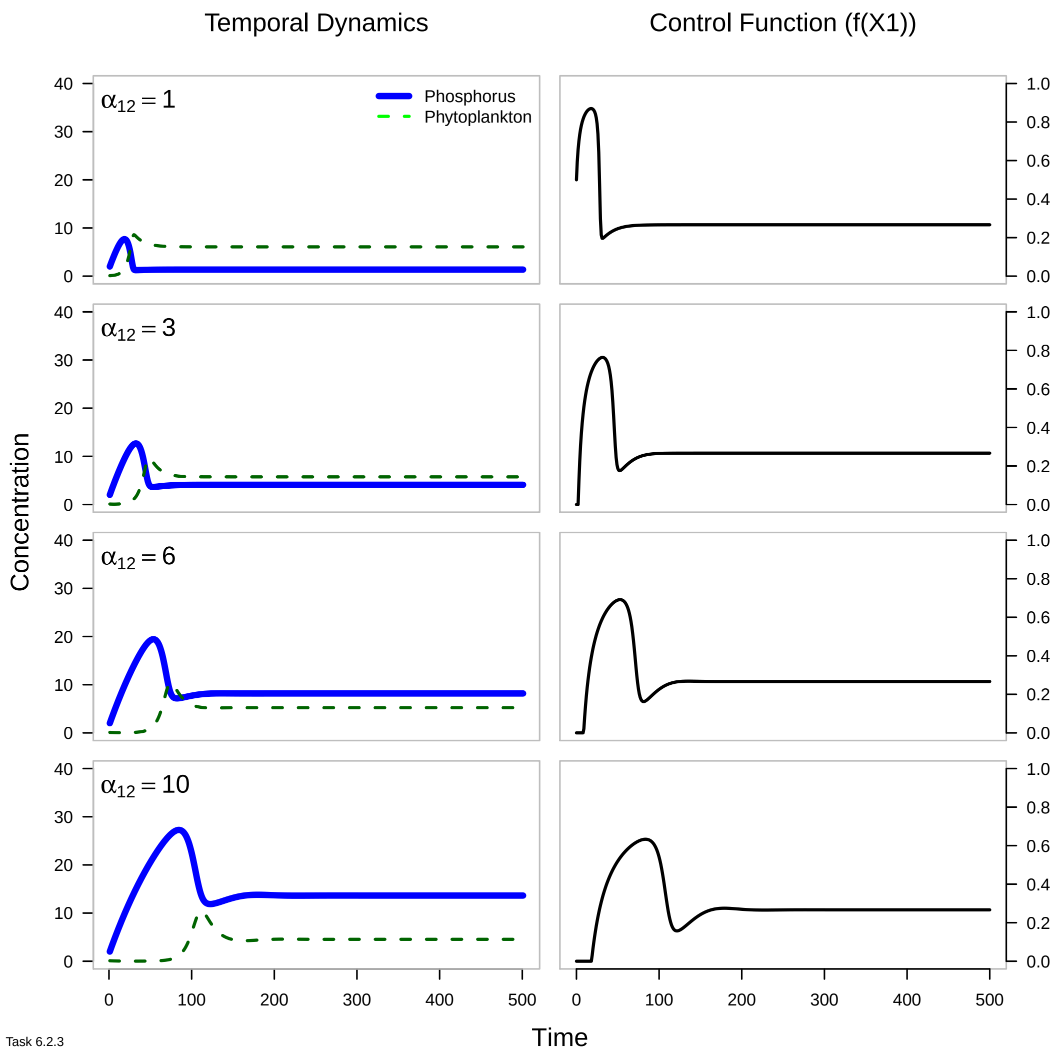

Figure 8: Hyperbolic form of the Michaelis-Menten control function with a refuge used to control phytoplankton (X_2) uptake of phosphorus (X_1). This is an example of donor control of consumption (exploitative control). Each panel in the left column shows the system dynamics under a different refuge value ($\alpha_{12} = \{1, 3, 6, 10\}$). Panels on the right show the realized behavior of the control function $f(X_1)$

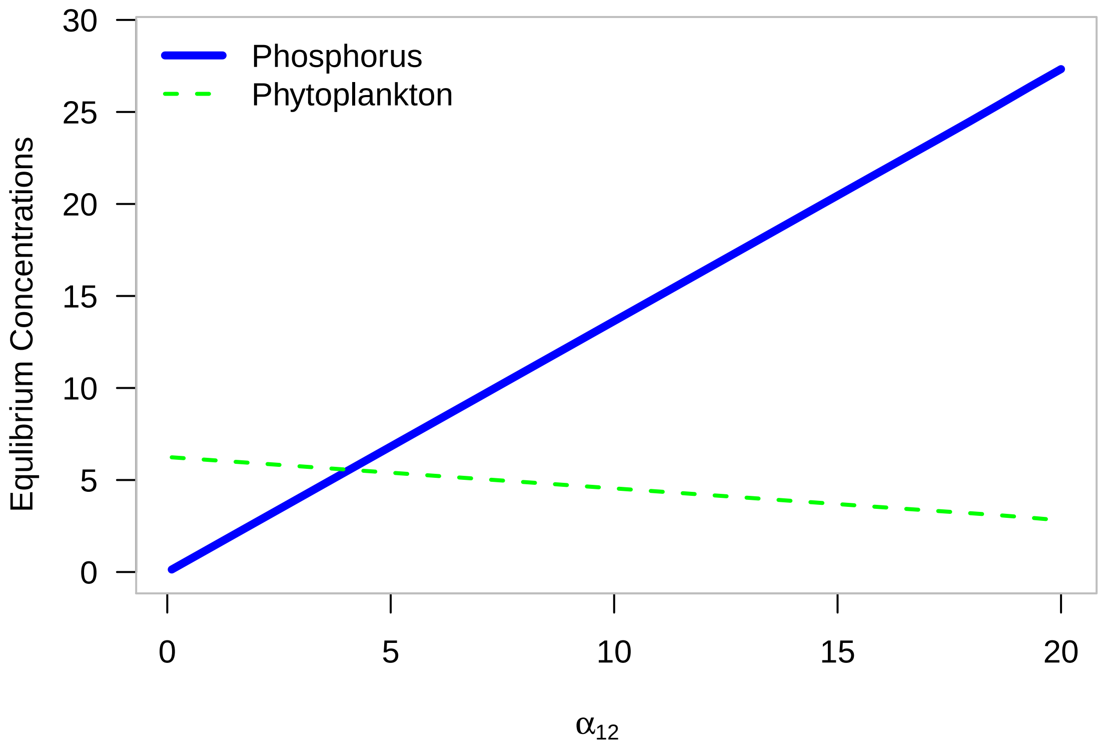

Figure 9: Equilibrium values of phosphorus and phytoplankton as the α_{12} values increase from 1 to 20 in Task 6.2.3.

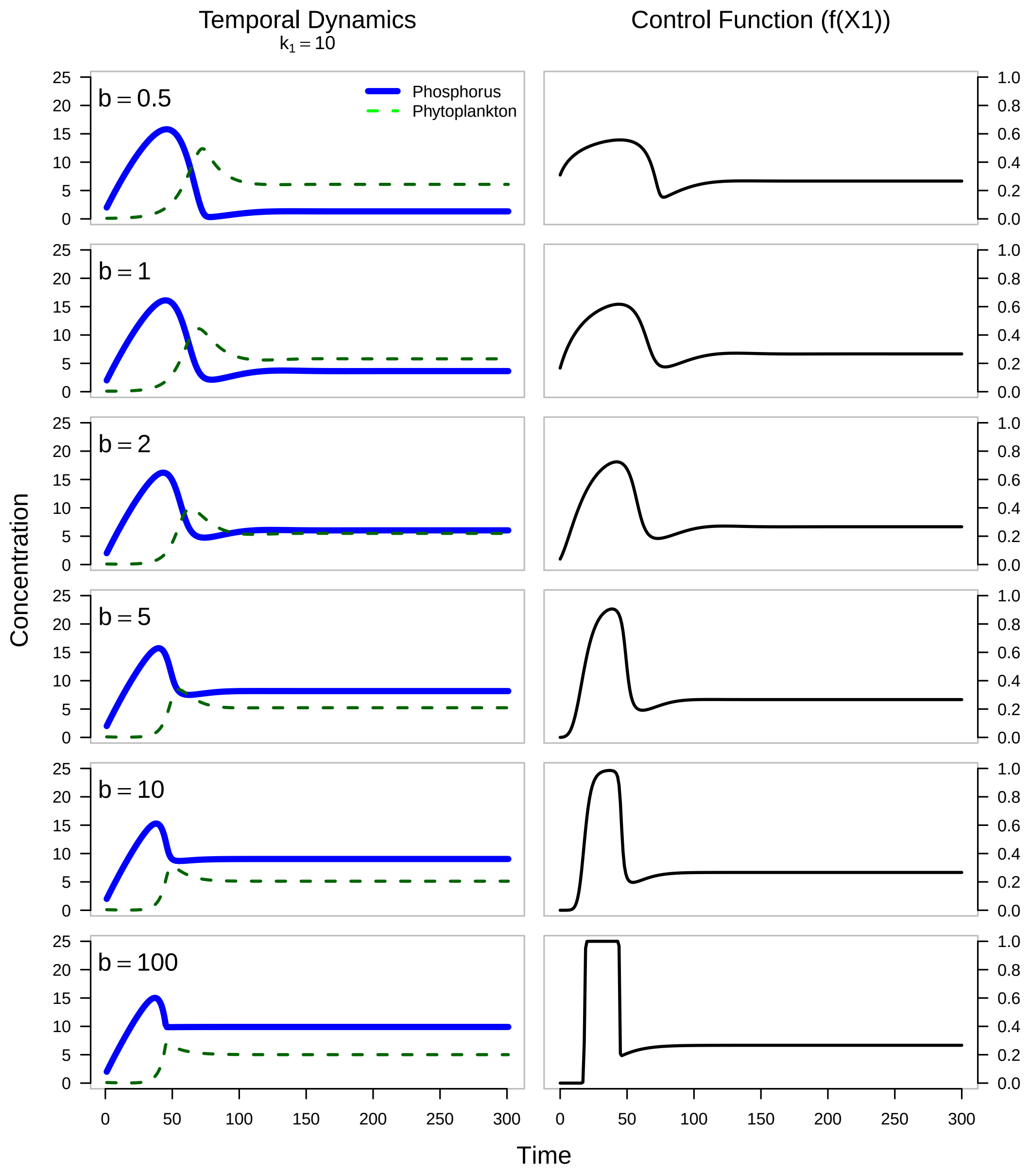

Figure 10: Generalized form of Michaelis–Menten 1/2 saturation control function used to control phytoplankton (X_2) uptake of phosphorus (X_1). This is an example of donor control of consumption (exploitative control). Each panel in the right column shows the system dynamics when $k_1 = 10$, $a = 1$, and $b = \{0.5, 1, 2, 5, 10, 100\}$). Panels on the right show the realized behavior of the control function $f(X_1)$

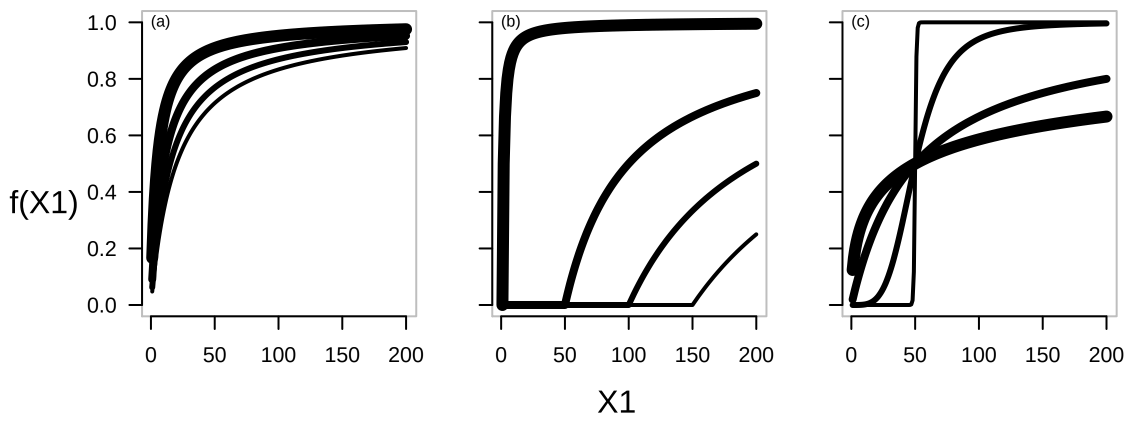

Figure 11: Potential values of the (a) hyperbolic form of the Michaelis-Menten 1/2 saturation control function, (b) Hyperbolic Form of the Michaelis–Menten Function with Refuge, and the (c) generalized form of Michaelis–Menten control function.

Appendices

A Programs for Task 6.1

A.1 Task 6.1.2

```
# 2 State Variables Laboratory
# Borrett
# October 2013
# -----------------
rm(list=ls())
library(deSolve)

# -- Model Function --
model=function(t,state,parameters){
  with(as.list(c(state,parameters)), {

    # CONTROL FUNCTIONS
    wc = 1 - d2/tau12

    cfx2 = pmax(0, (1 - wc * pmax(0, X2/K2) ) )

    cfx1 = X1/(X1 + k1)     # we are using this to approximate a step function

    # PROCESS EQUATION ELEMENTS
    input.resource = CIN
    loss.resource = d1 * X1
    uptake.resource = tau12 * X2 * cfx2 * cfx1
    uptake.consumer = uptake.resource
    loss.consumer = d2*X2

    # Diff Eqs.
    dX1 =  input.resource - loss.resource - uptake.resource
    dX2 =  uptake.consumer - loss.consumer
    return(list( c(dX1, dX2),  c(cfx1,cfx2) ) )
  })
}
# ----

# -- RUN Code
ti = 0
tf = 100
tspan=seq(ti,tf,by=1)

state=c(X1 = 2,
  X2 = 0.01)

parameters=c(CIN = 0.5,
  d1 = 0.001,
  d2 = 0.08,
  tau12 = 0.3,
  K2 = 10,
  k1 = 0.001)
```

```
out=ode(y=state,times=tspan,func=model,parms=parameters)

# -- PLOT ---

# SHOW IN CLASS
opar <- par(las=1, mfrow=c(1,3), cex.axis=1,
            cex.lab=2,
            mar=c(6,6,1,1),
            oma=c(1,1,1,1))
# temporal dynamics
matplot(out[,2:3],type="l",lwd=c(4,2),col=c("blue","darkgreen"),
        ylim=c(0,25),main="Temporal Dynamics",
        xlab="Time",ylab="Concentration")
legend("topright",legend=c("Phosphorus","Phytoplankton"),
       bty="n",col=c("blue","darkgreen"),lwd=c(4,2),lty=1:2,cex=1.2)
# statespace
plot(out[,2],out[,3],lwd=2,col="blue",type="l",main="State Space",
     ylim=c(0,max(out[,3])),xlim=c(0,max(out[,2])),
     xlab="X1",ylab="X2")
# cfx1
matplot(out[,4:5],ylim=c(0,1),main="control functions",
     lwd=2,xlab="Time",ylab="f(Xi)",type="l")
legend("bottomright", legend=c("f(X1)","f(X2)"),
       lty=1:2,col=1:2,bty="n",cex=2)
#cfx2
#plot(out[,1],out[,5],ylim=c(0,1),main="control function",
#          lwd=2,xlab="Time",ylab="f(X2)",type="l")
rm(opar)
```

A.2 Task 6.1.3

```
# 2 State Variables Laboratory
# Borrett
# October 2013
# -----------------
rm(list=ls())
library(deSolve)

# -- Model Function --
model=function(t,state,parameters){
  with(as.list(c(state,parameters)), {

    # CONTROL FUNCTIONS
    wc = 1-d2/tau12
    cfx2 = pmax(0,(1 - wc * pmax(0,X2/K2)))
    cfx1 = X1/(X1 + k1)     # we are using this to approximate a step function
    #if(X1 > 0){ cfx1 = 1} else {cfx1=0} # DOES NOT WORK


    # PROCESS EQUATION ELEMENTS
    input.resource = CIN
    loss.resource = d1 * X1
    uptake.resource = tau12 * X2 * cfx2 * cfx1
    uptake.consumer = uptake.resource
    loss.consumer = d2*X2

    # Diff Eqs.
    dX1 =  input.resource - loss.resource - uptake.resource
    dX2 =  uptake.consumer - loss.consumer
    return(list(c(dX1,dX2),c(cfx1,cfx2)))
  })
}
# ----

# -- RUN Code
ti = 0
tf = 100
tspan=seq(ti,tf,by=1)

state=c(X1 = 2,
  X2 = 0.01)

CIN.vec=c(0.5,0.8,1,5)
fn="../../results/2sv/task613.pdf"
pdf(fn,height=7,width=7)
opar <- par(las=1,mfrow=c(2,2),mar=c(3,3,2,0),oma=c(1,1,2,1))

for(i in 1:length(CIN.vec)){

parameters=c(CIN = CIN.vec[i],
  d1 = 0.001,
  d2 = 0.08,
  tau12 = 0.3,
  K2 = 10,
  k1 = 0.001)

out=ode(y=state,times=tspan,func=model,parms=parameters)

# temporal dynamics
matplot(out[,2:3],type="l",lwd=c(4,2),col=c("blue","darkgreen"),
        #ylim=c(0,35),
        main = bquote(C ==  .(CIN.vec[i])),
        xlab="Time",ylab="Population Abundance")

if(i == 1){
  legend("topright",legend=c("Phosphorus","Phytoplankton"),
         bty="n",col=c("blue","darkgreen"),lwd=c(4,2),lty=1:2)
}

}

mtext("Task 6.1.3",side=3,line=0,outer=TRUE)
mtext("Time",side=1,line=0,outer=TRUE)
mtext("Concentration",side=2,line=0,outer=TRUE,las=3)
dev.off()
rm(opar)
```

```
cmd <- paste("open",fn)
system(cmd)
```

A.3 Task 6.1.4

```
# 2 State Variables Laboratory
# Borrett
# October 2013
# -----------------
rm(list=ls())
library(deSolve)

# -- Model Function --
model=function(t,state,parameters){
  with(as.list(c(state,parameters)), {

    # CONTROL FUNCTIONS
    wc = 1-d2/tau12
    cfx2 = pmax(0,(1 - wc * pmax(0,X2/K2)))
    cfx1 = X1/(X1 + k1)     # we are using this to approximate a step function
    #if(X1 > 0){ cfx1 = 1} else {cfx1=0} # DOES NOT WORK


    # PROCESS EQUATION ELEMENTS
    input.resource = CIN
    loss.resource = d1 * X1
    uptake.resource = tau12 * X2 * cfx2 * cfx1
    uptake.consumer = uptake.resource
    loss.consumer = d2*X2

    # Diff Eqs.
    dX1 =  input.resource - loss.resource - uptake.resource
    dX2 =  uptake.consumer - loss.consumer
    return(list(c(dX1,dX2),c(cfx1,cfx2)))
  })
}
# ----

# -- RUN Code
ti = 0
tf = 200
tspan=seq(ti,tf,by=1)


X20=c(0.01,0.5,1,5,20,500)

fn="../../results/2sv/task614.pdf"
pdf(fn,height=8,width=7)
opar <- par(las=1,mfrow=c(2,3),mar=c(3,3,2,0),oma=c(1,1,2,1))

for(i in 1:length(X20)){

  state=c(X1 = 2,
    X2 = X20[i])


parameters=c(CIN = 0.5,
  d1 = 0.001,
  d2 = 0.08,
  tau12 = 0.3,
  K2 = 10,
  k1 = 0.001)

out=ode(y=state,times=tspan,func=model,parms=parameters)

# temporal dynamics
matplot(out[,2:3],type="l",lwd=c(4,2),col=c("blue","darkgreen"),
        #ylim=c(0,20),
        main = bquote(X[2](0) == .(X20[i])),
        xlab="Time",ylab="Population Abundance")

if(i == 1){
  legend("topright",legend=c("Phosphorus","Phytoplankton"),
         bty="n",col=c("blue","darkgreen"),lwd=c(4,2),lty=1:2)
}

}

mtext("Task 6.1.4",side=3,line=0,outer=TRUE)
mtext("Time",side=1,line=0,outer=TRUE)
mtext("Concentration",side=2,line=0,outer=TRUE,las=3)
dev.off()
rm(opar)

cmd <- paste("open",fn)
system(cmd)
```

B Programs for Task 6.2

B.1 Task 6.2.2

```
# 2 State Variables Laboratory
# Borrett
# October 2013
# modified to include parameter vs. equilibrium plot
                                        # -----------------
rm(list=ls())
library(deSolve)

# -- Model Function --
model=function(t,state,parameters){
  with(as.list(c(state,parameters)), {

    # CONTROL FUNCTIONS
    cfx2 = 1  # This effectively turns off this control fuction.

    # use sw parameter to select which version of cfx1 to use
    if(sw==1) cfx1 <- pmax(0,1-k1/(X1 + k1))            # Hyperbolic Form of Michaelis-Menton
    if(sw==2) cfx1 <- pmax(0,(1 - alpha12/X1))          # refuge form of the Michaelis-Menton
    if(sw==3) cfx1 <- pmax(0, X1^b/(X1^b + k1^b))       # generalized form of the Michaelis-Menton


    # PROCESS EQUATION ELEMENTS
    input.resource = CIN
    loss.resource = d1 * X1
    uptake.resource = tau12 * X2 * cfx2 * cfx1
    uptake.consumer = uptake.resource
    loss.consumer = d2*X2

    # Diff Eqs.
    dX1 =  input.resource - loss.resource - uptake.resource
    dX2 =  uptake.consumer - loss.consumer
    return(list(c(dX1,dX2),c(cfx1)))
  })
}
# ----

# ============================================================
# -- RUN Code --------------
ti = 0
tf = 500
tspan=seq(ti,tf,by=1)
state=c(X1 = 2,
  X2 = 0.1)
#my.vec=c(5,10,15,20)  # vector used for primary program loop
my.vec=seq(0.1,20,length=30)  # used for equilibirum analysis
my.kvec=c(1)  # this is useful for task 6.2.4

## === PLOTTING CONTROLS ======================
sw2=0 # turn on and off this type of plotting
if(sw2==1){
  fn <- "../../results/2sv/task622_dynamics.pdf"
  pdf(fn,height=6,width=7)
  opar <- par(las=1,mfrow=c(length(my.vec),2),
              mar=c(0,0,1,1),oma=c(5,5,3,2))
}

sw.3=1  # equilbrium plot
if(sw.3==1){
  fn <- "../../results/2sv/task622_equilibriums.pdf"
  pdf(fn,height=4,width=6)
  opar <- par(las=1,mfrow=c(1,1),mar=c(2,2,1,1),oma=c(2,2,0,0))
}
# ==============================================

## --------------- PROGRAM LOOP -----------------
for(j in 1:length(my.kvec)){  # this loop allows me to test the sensitivity of the results to a second p
  out=list()
  eq.vals=matrix(NA,nrow=length(my.vec),ncol=2)

for(i in 1:length(my.vec)){  # this is the MAIN PROGRAM loop

  parameters=c(CIN = 0.5,
    d1 = 0.01,
    d2 = 0.08,
    tau12 = 0.3,
                                      #  K2 = 10,
    k1 = my.vec[i],
    sw=1,
    alpha12=0,
    b=1)

  out[[i]]=ode(y=state,times=tspan,func=model,parms=parameters)
  # I am storing each result in a out, which has a list data structure

  eq.vals[i,]=out[[i]][tf,2:3]  # collect equlibrium values, assuming
                                # the model has reached equilibrium by
                                # t=tf

# -- PLOT ---
# SHOW IN CLASS
  if(sw2==1){
  # -- temporal dynamics
    matplot(out[[i]][,2:3],type="l",lwd=c(4,2),col=c("blue","darkgreen"),
            ylim=c(0,30),#main="Temporal Dynamics",
            xlab="Time",ylab="Concentraion",
            axes=FALSE)
    if(i == length(my.vec)){
      axis(1)
    }
    axis(2)
    box(col="grey")
              #  mtext(bquote(k[1] == .(my.vec[i])),side=3,line=-2,adj=0.02)
    mtext(bquote(k[1] == .(parameters[5])),side=3,outer=FALSE,line=-2,cex=0.75,adj=0.01)
    if(i ==1) {
```

```
    mtext("Temporal Dynamics",side=3,outer=FALSE,line=2)
        mtext("Task 6.2.2",line=3,side=1,adj=-0.1,outer=TRUE,cex=0.5)
    mtext("Time", side=1,outer=TRUE,line=3)
    mtext("Concentration", side=2,outer=TRUE,line=3,las=3)
    legend("topright",legend=c("Phosphorus","Phytoplankton"),
           bty="n",col=c("blue","green"),lwd=c(4,2),lty=1:2)
  }
    # -- cfx1 -- plot the observed resource control function dynamics
    plot(out[[i]][,1],out[[i]][,4],ylim=c(0,1),#main="Control Function",
         lwd=2,xlab="Time",ylab="f(X1)",type="l",axes=FALSE)
    box(col="grey")
    axis(4)
    if(i == length(my.vec)) axis(1)
    if(i ==1) mtext("Control Function (f(X1))",side=3,outer=FALSE,line=2)
  }
}

# new plot

if(sw.3==1){
  plot(my.vec,eq.vals[,1],type="l",col="blue",lwd=4,
       ylim=c(0,ceiling(1.05*max(eq.vals))),
       ylab = "equlibrium values",
       xlab = "parameter values",
       axes=FALSE
       )
  axis(1)
  axis(2)
  box(col="grey")
  points(my.vec,eq.vals[,2],type="l",col="green",lty=2,lwd=2)
#  mtext(bquote(k[1] == .(parameters[5])),side=3,outer=FALSE,line=0.75,cex=1)
  if(j ==1) {
    legend("topleft",legend=c("Phosphorus","Phytoplankton"),
           bty="n",col=c("blue","green"),lwd=c(4,2),lty=1:2)
    mtext("Equlibrium Concentrations",side=2,las=3,line=0.5,outer=TRUE)
    mtext(bquote(k[1]),side=1,line=1,outer=TRUE)
    mtext("Task 6.2.3",line=3,side=1,adj=-0.1,outer=TRUE,cex=0.5)
  }

#  if(i == length(my.kvec) || i == (length(my.kvec)-1)) {axis(1)}
#    if(j == length(my.kvec) || j == (length(my.kvec)-1)) {axis(1)}
}

}


dev.off()
cmd <- paste("open",fn)
system(cmd)
rm(opar)
```

B.2 Task 6.2.3

```
# 2 State Variables Laboratory
# Borrett
# October 2013
# modified to include parameter vs. equilibrium plot
                                        # -----------------
rm(list=ls())
library(deSolve)

# -- Model Function --
model=function(t,state,parameters){
  with(as.list(c(state,parameters)), {

    # CONTROL FUNCTIONS
    cfx2 = 1  # This effectively turns off this control fuction.

    # use sw parameter to select which version of cfx1 to use
    if(sw==1) cfx1 <- pmax(0,1-k1/(X1 + k1))            # Hyperbolic Form of Michaelis-Menton
    if(sw==2) cfx1 <- pmax(0,(1 - alpha12/X1))          # refuge form of the Michaelis-Menton
    if(sw==3) cfx1 <- pmax(0, X1^b/(X1^b + k1^b))       # generalized form of the Michaelis-Menton


    # PROCESS EQUATION ELEMENTS
    input.resource = CIN
    loss.resource = d1 * X1
    uptake.resource = tau12 * X2 * cfx2 * cfx1
    uptake.consumer = uptake.resource
    loss.consumer = d2*X2

    # Diff Eqs.
    dX1 =  input.resource - loss.resource - uptake.resource
    dX2 =  uptake.consumer - loss.consumer
    return(list(c(dX1,dX2),c(cfx1)))
  })
}
# ----

# ============================================================
# -- RUN Code --------------
ti = 0
tf = 500
tspan=seq(ti,tf,by=1)
state=c(X1 = 2,
  X2 = 0.1)
my.vec=c(1,3,6,10) #1:20 #20
#my.vec=seq(0.1,20,length=30)
my.kvec=c(1)  # this is useful for task 6.2.4

## === PLOTTING CONTROLS ======================
sw2=1 # turn on and off this type of plotting
if(sw2==1){
  fn <- "../../results/2sv/task623_dynamics.pdf"
  pdf(fn,height=7,width=7)
  opar <- par(las=1,mfrow=c(length(my.vec),2),
              mar=c(0,0,1,1),oma=c(5,5,3,2))
}
```

```
sw.3=0  # equilbrium plot
if(sw.3==1){
  fn <- "../../results/2sv/task623_equilibriums.pdf"
  pdf(fn,height=4,width=6)
  opar <- par(las=1,mfrow=c(1,1),mar=c(2,2,1,1),oma=c(2,2,0,0))
}
# ==============================================

## --------------- PROGRAM LOOP -----------------
for(j in 1:length(my.kvec)){  # this loop allows me to test the sensitivity of the results to a second p
  out=list()
  eq.vals=matrix(NA,nrow=length(my.vec),ncol=2)

for(i in 1:length(my.vec)){  # this is the MAIN PROGRAM loop

  parameters=c(CIN = 0.5,
    d1 = 0.01,
    d2 = 0.08,
    tau12 = 0.3,
                                       #  K2 = 10,
    k1 = my.kvec[j],
    sw=2,
    alpha12=my.vec[i],
    b=my.vec[i])

  out[[i]]=ode(y=state,times=tspan,func=model,parms=parameters)
  # I am storing each result in a out, which has a list data structure

  eq.vals[i,]=out[[i]][tf,2:3]  # collect equlibrium values, assuming
                                # the model has reached equilibrium by
                                # t=tf

# -- PLOT ---
# SHOW IN CLASS
  if(sw2==1){
  # -- temporal dynamics
    matplot(out[[i]][,2:3],type="l",lwd=c(4,2),col=c("blue","darkgreen"),
            ylim=c(0,40),#main="Temporal Dynamics",
            xlab="Time",ylab="Concentraion",
            axes=FALSE)
    if(i == length(my.vec)){
      axis(1)
    }
    axis(2)
    box(col="grey")
              #  mtext(bquote(k[1] == .(my.vec[i])),side=3,line=-2,adj=0.02)
    mtext(bquote(alpha[12] == .(my.vec[i])),side=3,line=-2,adj=0.02)
    if(i ==1) {
    mtext("Temporal Dynamics",side=3,outer=FALSE,line=2)
#   mtext(bquote(k[1] == .(parameters[5])),side=3,outer=FALSE,line=0.75,cex=0.75)
    mtext("Task 6.2.3",line=3,side=1,adj=-0.1,outer=TRUE,cex=0.5)
    mtext("Time", side=1,outer=TRUE,line=3)
    mtext("Concentration", side=2,outer=TRUE,line=3,las=3)
    legend("topright",legend=c("Phosphorus","Phytoplankton"),
           bty="n",col=c("blue","green"),lwd=c(4,2),lty=1:2)
  }
    # -- cfx1 -- plot the observed resource control function dynamics
    plot(out[[i]][,1],out[[i]][,4],ylim=c(0,1),#main="Control Function",
         lwd=2,xlab="Time",ylab="f(X1)",type="l",axes=FALSE)
    box(col="grey")
    axis(4)
    if(i == length(my.vec)) axis(1)
    if(i ==1) mtext("Control Function (f(X1))",side=3,outer=FALSE,line=2)
  }
}

# new plot

if(sw.3==1){
  plot(my.vec,eq.vals[,1],type="l",col="blue",lwd=4,
       ylim=c(0,ceiling(1.05*max(eq.vals))),
       ylab = "equlibrium values",
       xlab = "parameter values",
       axes=FALSE
       )
  axis(1)
  axis(2)
  box(col="grey")
  points(my.vec,eq.vals[,2],type="l",col="green",lty=2,lwd=2)
#  mtext(bquote(k[1] == .(parameters[5])),side=3,outer=FALSE,line=0.75,cex=1)
  if(j ==1) {
    legend("topleft",legend=c("Phosphorus","Phytoplankton"),
           bty="n",col=c("blue","green"),lwd=c(4,2),lty=1:2)
    mtext("Equlibrium Concentrations",side=2,las=3,line=0.5,outer=TRUE)
    mtext(bquote(alpha[12]),side=1,line=1,outer=TRUE)
    mtext("Task 6.2.3",line=3,side=1,adj=-0.1,outer=TRUE,cex=0.5)
  }

#  if(i == length(my.kvec) || i == (length(my.kvec)-1)) {axis(1)}
#    if(j == length(my.kvec) || j == (length(my.kvec)-1)) {axis(1)}
}

}


dev.off()
cmd <- paste("open",fn)
system(cmd)
rm(opar)
```

B.3 Task 6.2.4

```
# 2 State Variables Laboratory
# Borrett
# October 2015
# modified to include parameter vs. equilibrium plot
                                        # -----------------
```

```
rm(list=ls())
library(deSolve)

# -- Model Function --
model=function(t,state,parameters){
  with(as.list(c(state,parameters)), {

    # CONTROL FUNCTIONS
    cfx2 = 1  # This effectively turns off this control fuction.

    # use sw parameter to select which version of cfx1 to use
    if(sw==1) cfx1 <- pmax(0, 1 - k1 / (X1 + k1))            # Hyperbolic Form of Michaelis-Menton
    if(sw==2) cfx1 <- pmax(0,(1 - alpha12 / X1))           # refuge form of the Michaelis-Menton
    if(sw==3) cfx1 <- pmax(0, X1^b / (X1^b + k1^b))       # generalized form of the Michaelis-Menton


    # PROCESS EQUATION ELEMENTS
    input.resource = CIN
    loss.resource = d1 * X1
    uptake.resource = tau12 * X2 * cfx2 * cfx1
    uptake.consumer = uptake.resource
    loss.consumer = d2*X2

    # Diff Eqs.
    dX1 =  input.resource - loss.resource - uptake.resource
    dX2 =  uptake.consumer - loss.consumer
    return(list(c(dX1,dX2),c(cfx1)))
  })
}
# ----

# -- RUN Code --------------
ti = 0
tf = 2000
tspan=seq(ti, tf, by=1)
state=c(X1 = 2,
    X2 = 0.1)
sw = 2

if(sw ==1){
    my.vec=seq(1,100,length=100)
    my.kvec = 1
}

if(sw == 2) {
    my.vec=seq(1,100,length=100)
    my.kvec=1
}

if(sw == 3) {
    my.vec=c(0.5,1,2,5,10,100) # b
    my.kvec=c(4,6,10,20,30,50)
}
```

```
sw2=0 # turn on and off this type of plotting
if(sw2==1){
  fn <- "../../results/2sv/task2_4b_dynamics_c.pdf"
  pdf(fn,height=8,width=7)
  opar <- par(las=1,mfrow=c(length(my.vec),2),
              mar=c(0,0,1,1),oma=c(5,5,3,2))
}

sw.3=1  # equilbrium plot
if(sw.3==1){
  fn <- "../../results/2sv/task2_4b_equilibriums.pdf"
  pdf(fn,height=8,width=7)
  opar <- par(las=1,mfrow=c(3,2),mar=c(2,2,2,1),oma=c(2,2,1,1))
}

for(j in 1:length(my.kvec)){    ## - K loop
    out=list()

    eq.vals=matrix(NA,nrow=length(my.vec),ncol=2)
## -- PROGRAM LOOP ---
    for(i in 1:length(my.vec)){

      parameters=c(CIN = 0.5,
          d1 = 0.01,
          d2 = 0.08,
          tau12 = 0.3,
                                      #  K2 = 10,
          k1 = my.kvec[j],
          sw=2,
          alpha12=my.vec[i],
          b=my.vec[i])

      out[[i]]=ode(y=state,times=tspan,func=model,parms=parameters)

      eq.vals[i,]=out[[i]][tf,2:3]

# -- PLOT ---
# SHOW IN CLASS
      if(sw2==1){
  # -- temporal dynamics
          matplot(out[[i]][,2:3],type="l",lwd=c(4,2),col=c("blue","darkgreen"),
                  ylim=c(0,25),#main="Temporal Dynamics",
                  xlab="Time",ylab="Concentraion",
                  axes=FALSE)
          box(col="grey")
          if(i == length(my.vec)){
              axis(1)
          }
          axis(2)
                                      #  mtext(bquote(k[1] == .(my.vec[i])),side=3,line=-2,adj=0.02)
          mtext(bquote(b == .(my.vec[i])),side=3,line=-2,adj=0.02)
          if(i ==1) {
              mtext("Temporal Dynamics",side=3,outer=FALSE,line=2)
```

```
                mtext(bquote(k[1] == .(parameters[5])),side=3,outer=FALSE,line=0.75,cex=0.75)
                mtext("Time", side=1,outer=TRUE,line=3)
                mtext("Concentration", side=2,outer=TRUE,line=3,las=3)
                legend("topright",legend=c("Phosphorus","Phytoplankton"),
                       bty="n",col=c("blue","green"),lwd=c(4,2),lty=1:2)
          }
# cfx1
          plot(out[[i]][,1],out[[i]][,4],ylim=c(0,1),#main="Control Function",
               lwd=2,xlab="Time",ylab="f(X1)",type="l",axes=FALSE)
          box(col="grey")
          axis(4)
          if(i == length(my.vec)) axis(1)
          if(i ==1) mtext("Control Function (f(X1))",side=3,outer=FALSE,line=2)
      }
  }

# new plot

  if(sw.3==1){
      plot(my.vec,eq.vals[,1],type="l",col="blue",lwd=4,
           ylim=c(0,ceiling(1.05*max(eq.vals))),
           ylab = "equlibrium values",
           xlab = "parameter values",
           axes=FALSE
           )
      axis(2)
      box(col="grey")
      points(my.vec,eq.vals[,2],type="l",col="green",lty=2,lwd=2)
      mtext(bquote(k[1] == .(parameters[5])),side=3,outer=FALSE,line=0.75,cex=1)
      if(j ==1) {
          legend("bottomright",legend=c("Phosphorus","Phytoplankton"),
                 bty="n",col=c("blue","green"),lwd=c(4,2),lty=1:2)
          mtext("Equlibrium Concentrations",side=2,las=3,line=0.5,outer=TRUE)
          mtext("Parameter Values (b)",side=1,line=1,outer=TRUE)
      }

#  if(i == length(my.kvec) || i == (length(my.kvec)-1)) {axis(1)}
      if(j == length(my.kvec) || j == (length(my.kvec)-1)) {axis(1)}
  }

}


dev.off()
cmd <- paste("open",fn)
system(cmd)
rm(opar)
```