Abbott & Nolting (2016) Ecological Complexity / Shoemaker et al. (2020) Ecology

T.J. Clark

2020-08-13

Trying to recreate the figures

```
library(tidyverse)
library(deSolve)
library(QPot)
library(phaseR)
library(rootSolve)
library(rgl)
library(plot3D)
```

Derived from Rosenzweig & MacArthur (1963) model

From Shoemaker et al. (2020) Ecology. Shows sustained transient cycles that keep bumping the trajectory away from the stable state. In essence, stochasticity provokes the short-term transient behavior to become long-term. Therefore, stochasticity reveals equilibrial behaviors.

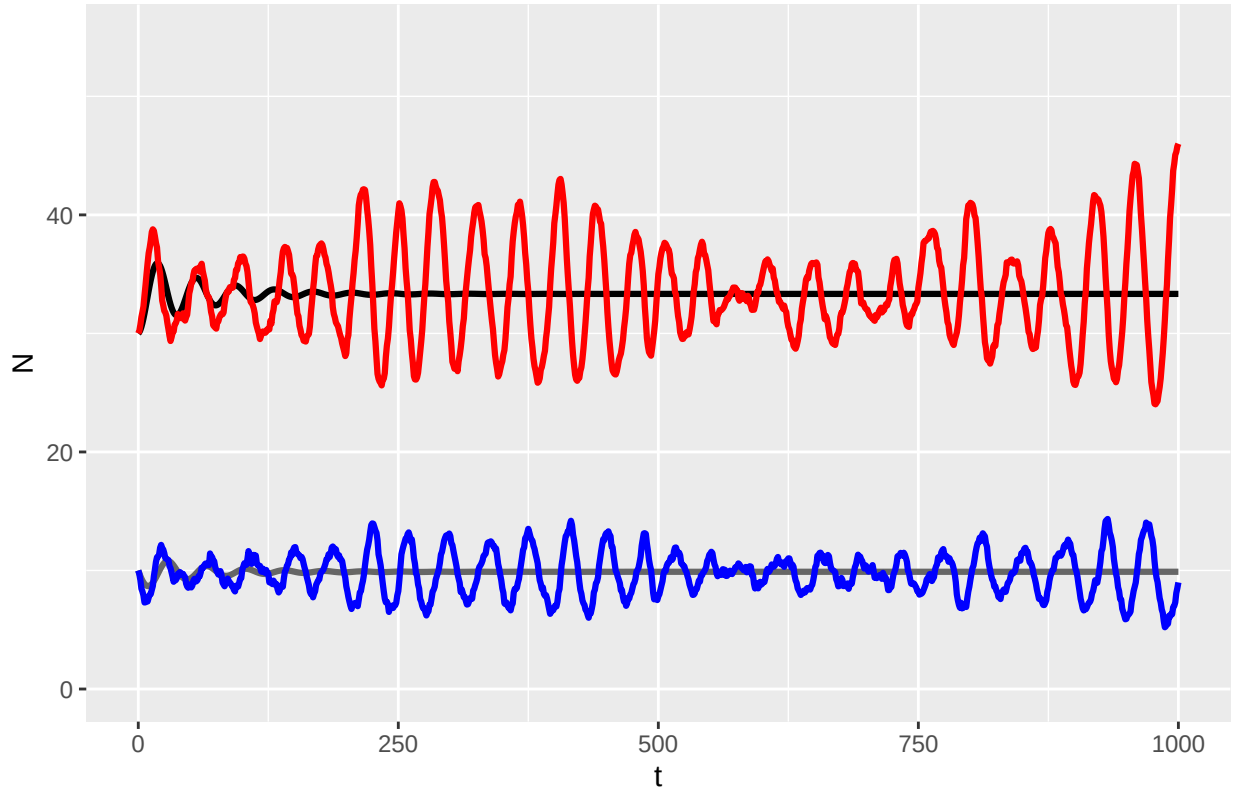
$$\frac{dN}{dt} = rN(1 - \frac{N}{K}) - \frac{mNP}{1 + mhN} + N\zeta_n\sigma_{t,n}$$

$$\frac{dP}{dt} = \frac{cmNP}{1 + mhN} - dP + P\zeta_p\sigma_{t,p}$$

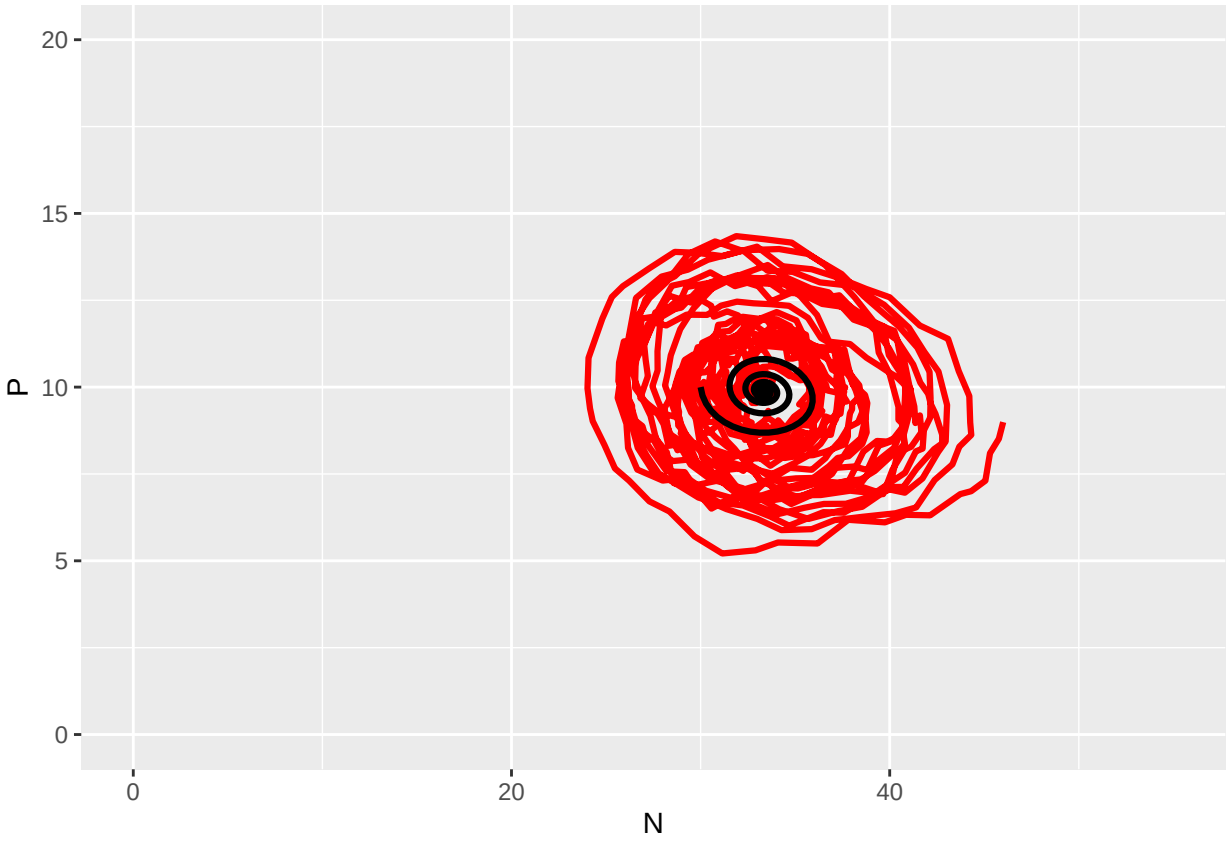
```
#### non-stochastic code
predprey <- function(Time, State, Pars){
  with(as.list(c(State, Pars)), {
    dN <- r*N*(1-N/K) - m*N*P/(1+m*h*N)
    dP <- c*m*N*P/(1+m*h*N) - d*P
    return(list(c(dN,dP)))
  })
}

# params
p <- c(r = 0.2, K = 82, m = 0.02, h = 1, c = 1, d = 0.4)
y <- c(N = 30, P = 10)
t <- seq(0,1000,by=1)

out <- as_tibble(ode(y, t, predprey, p)[,-1])

#### stochastic code
```

```
var.eqn.x <- "r*N*(1-N/K) - m*N*P/(1+m*h*N)"
var.eqn.y <- "c*m*N*P/(1+m*h*N) - d*P"
model.parms <- c(r = 0.2, K = 82, m = 0.02, h = 1, c = 1, d = 0.4)
parms.eqn.x <- Model2String(var.eqn.x, parms = model.parms, supress.print = T)
parms.eqn.y <- Model2String(var.eqn.y, parms = model.parms, supress.print = T)
model.state <- c(N = 30, P = 10)
model.sigma <- 0.25
model.time <- 1001
model.deltat <- 1 # frequency of stochastic perturbation

out2 <- as_tibble(TSTraj(y0 = model.state, time = model.time, deltat = model.deltat,
               x.rhs = parms.eqn.x, y.rhs = parms.eqn.y, sigma = model.sigma))
```

```
# time-series
out %>%
  ggplot(aes(t)) +
  geom_line(aes(y = N), color = "black", lwd = 1.1) +
  geom_line(aes(y = P), color = "grey40", lwd = 1.1) +
  geom_line(aes(y = out2$N), lwd = 1.1, color = "red") +
  geom_line(aes(y = out2$P), lwd = 1.1, color = "blue")+
  ylim(c(0,55)) +
  ggtitle("")
```



```r
# state-space
out %>%
  ggplot(aes(N,P)) +
  geom_path(aes(x = out2$N, y = out2$P), lwd = 1.1, color = "red") +
  geom_path(lwd=1.1, color = "black") +
  xlim(c(0,55))+
  ylim(c(0,20))
```



Derived from Freedman & Wolkowicz (1986) model.

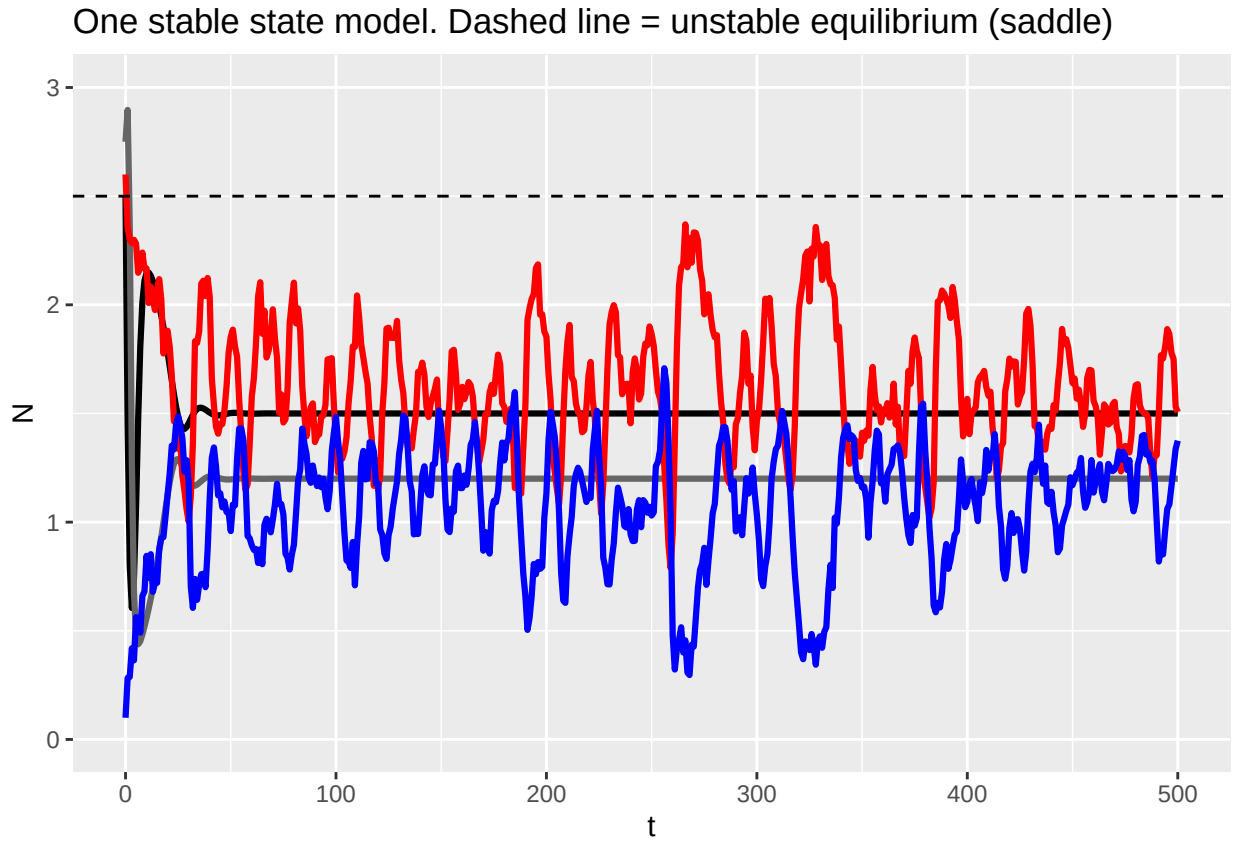
Includes "Type IV" functional response which accounts for prey grouping defense. From Shoemaker et al. and Abbott & Nolting (2016) Ecological Complexity. Stochasticity reveals unstable dynamics as well, by visiting both stable and unstable equilibria with stochasticity. AND dynamics may be mistaken for multiple stable states, even though there's only one!

$$\frac{dN}{dt} = rN(1 - \frac{N}{K}) - \frac{mNP}{N^2 + m(N+1)} + N\zeta_n\sigma_{t,n}$$

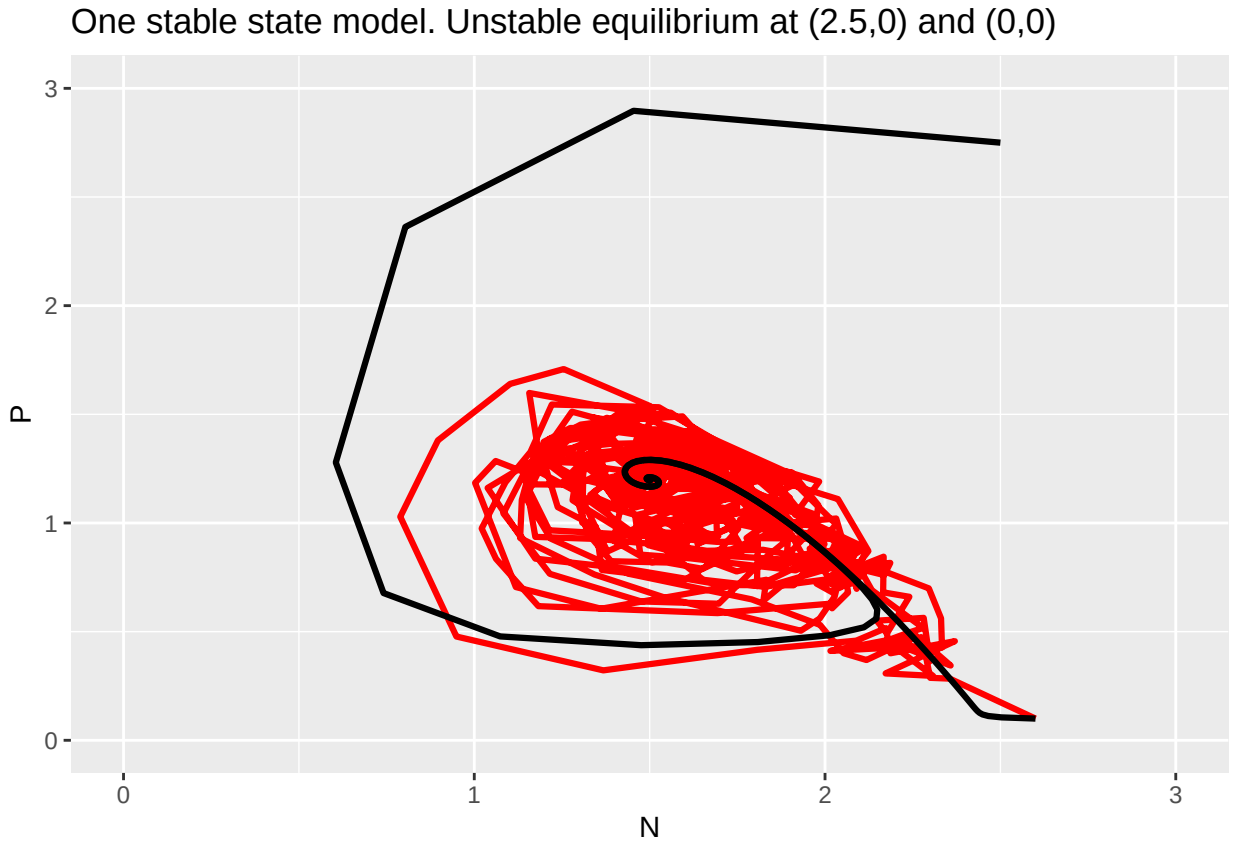
$$\frac{dP}{dt} = \frac{cmNP}{N^2 + m(N+1)} - dP + P\zeta_p\sigma_{t,p}$$

```
#### non-stochastic code
predprey2 <- function(Time, State, Pars){
  with(as.list(c(State, Pars)), {
    dN <- r*N*(1-N/K) - m*N*P/(N^2 + m*(N+1))
    dP <- c*m*N*P/(N^2 + m*(N+1)) - d*P
    return(list(c(dN,dP)))
  })
}

# params
p <- c(r = 1, K = 2.5, m = 4.5, c = 5, d = 2.5)
p2 <- c(r = 1, K = 2.5, m = 4.1, c = 5, d = 2.5)
y <- c(N = 2.5, P = 2.75)
y2 <- c(N = 2.6, P = 0.1) # 2nd approach, illustrating saddle dynamics
t <- seq(0,500,by=1)

out1 <- as_tibble(ode(y, t, predprey2, p)[,-1])
out1b <- as_tibble(ode(y2, t, predprey2, p)[,-1])
out1c <- as_tibble(ode(y, t, predprey2, p2)[,-1])
out1d <- as_tibble(ode(y2, t, predprey2, p2)[,-1])

#### stochastic code

var.eqn.x <- "r*N*(1-N/K) - m*N*P/(N^2 + m*(N+1))"
var.eqn.y <- "c*m*N*P/(N^2 + m*(N+1)) - d*P"
model.parms <- c(r = 1, K = 2.5, m = 4.5, c = 5, d = 2.5)
model.parms.2 <- c(r = 1, K = 2.5, m = 4.1, c = 5, d = 2.5)
parms.eqn.x <- Model2String(var.eqn.x, parms = model.parms, supress.print = T)
parms.eqn.y <- Model2String(var.eqn.y, parms = model.parms, supress.print = T)
parms.eqn.x.2 <- Model2String(var.eqn.x, parms = model.parms.2, supress.print = T)
parms.eqn.y.2 <- Model2String(var.eqn.y, parms = model.parms.2, supress.print = T)
model.state <- c(N = 2.6, P = 0.1)
model.sigma <- 0.08
model.time <- 501
model.deltat <- 1 # frequency of stochastic perturbation

out2 <- as_tibble(TSTraj(y0 = model.state, time = model.time, deltat = model.deltat,
               x.rhs = parms.eqn.x, y.rhs = parms.eqn.y, sigma = model.sigma))

out3 <- as_tibble(TSTraj(y0 = model.state, time = model.time, deltat = model.deltat,
               x.rhs = parms.eqn.x.2, y.rhs = parms.eqn.y.2, sigma = model.sigma))
```

```r
# time-series
out1 %>%
  ggplot(aes(t)) +
  geom_line(aes(y = N), color = "black", lwd = 1.1) +
  geom_line(aes(y = P), color = "grey40", lwd = 1.1) +
  geom_line(aes(y = out2$N), lwd = 1.1, color = "red") +
  geom_line(aes(y = out2$P), lwd = 1.1, color = "blue")+
  geom_hline(yintercept = 2.5, linetype = "dashed")+
  ylim(c(0,3)) +
  ggtitle("One stable state model. Dashed line = unstable equilibrium (saddle)")
```

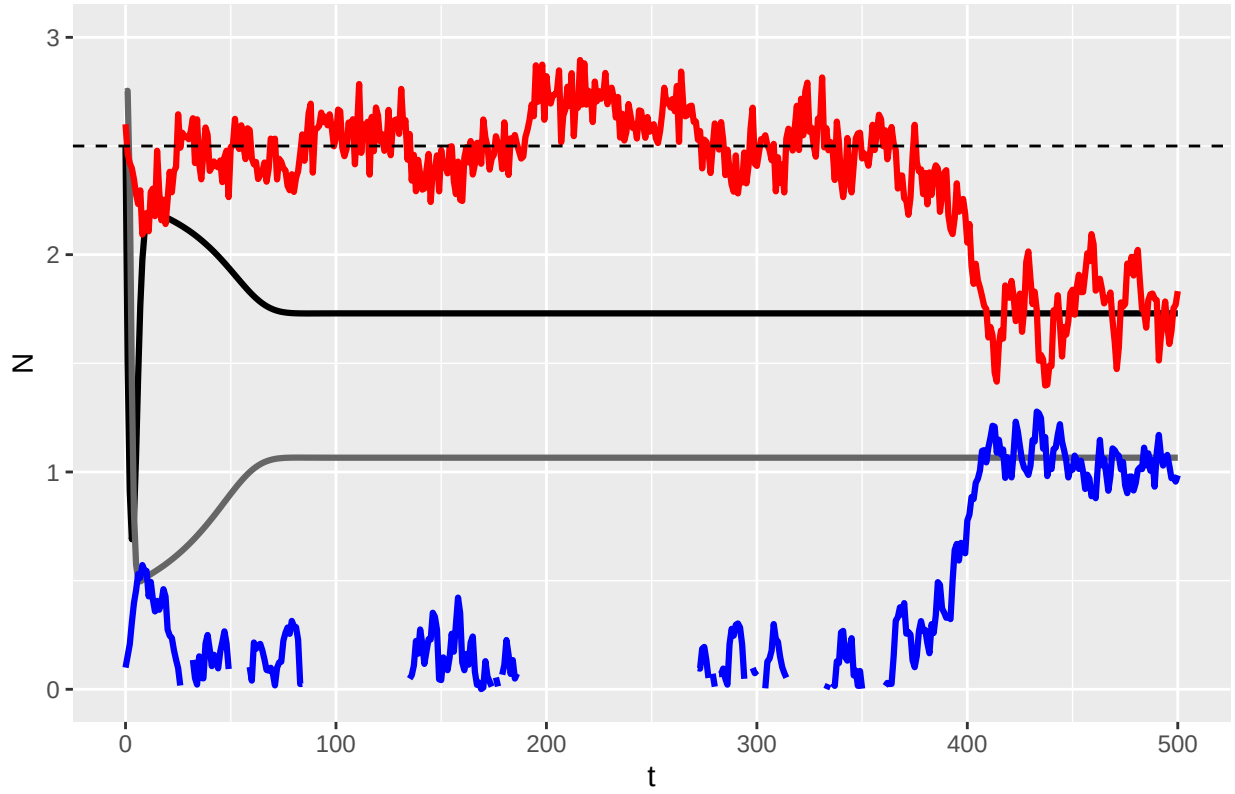



```r
# state-space
out1 %>%
  ggplot(aes(N,P)) +
  geom_path(aes(x = out2$N, y = out2$P), lwd = 1.1, color = "red") +
  geom_path(lwd=1.1, color = "black") +
  geom_path(aes(x = out1b$N, y = out1b$P), lwd = 1.1, color = "black") +
  xlim(c(0,3))+
  ylim(c(0,3))+
  ggtitle("One stable state model. Unstable equilibrium at (2.5,0) and (0,0)")
```

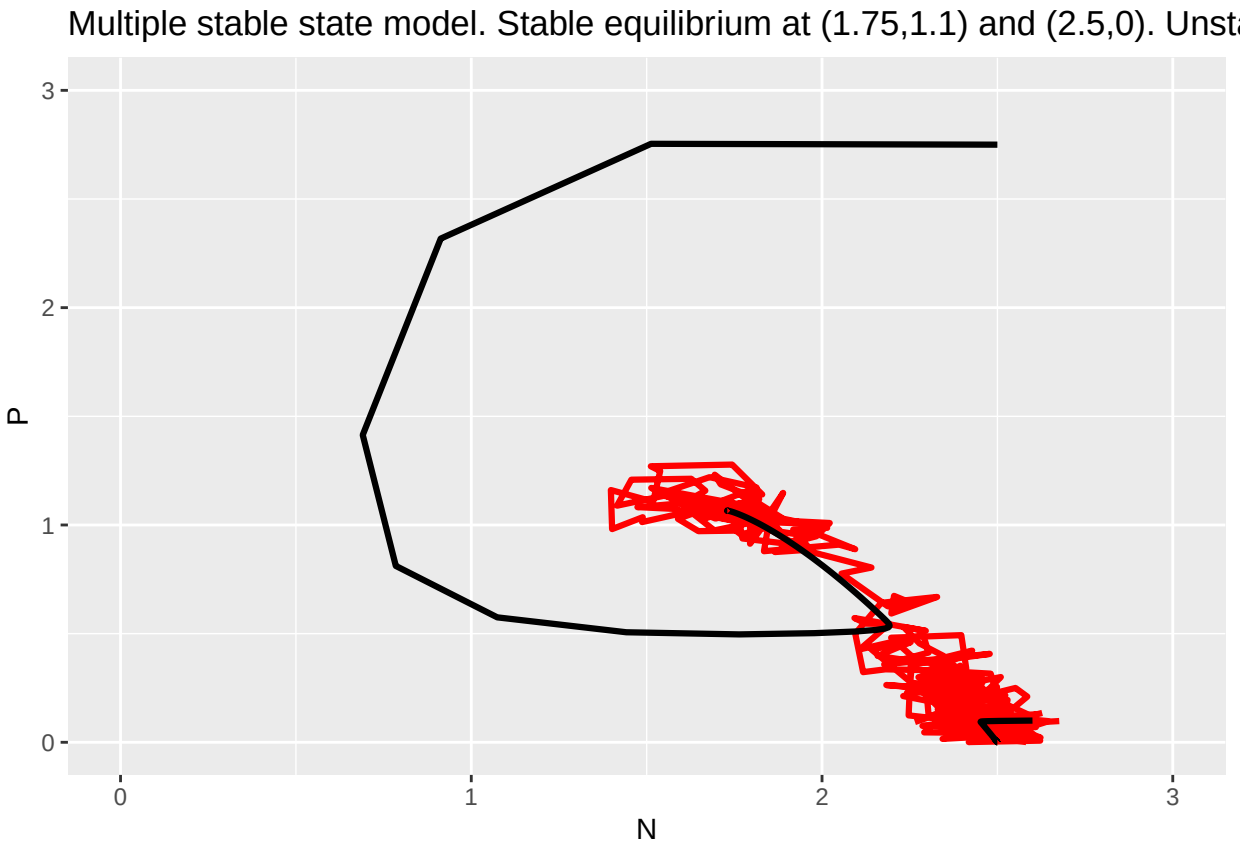



```
out1c %>%
  ggplot(aes(t)) +
  geom_line(aes(y = N), color = "black", lwd = 1.1) +
  geom_line(aes(y = P), color = "grey40", lwd = 1.1) +
  geom_line(aes(y = out3$N), lwd = 1.1, color = "red") +
  geom_line(aes(y = out3$P), lwd = 1.1, color = "blue")+
  geom_hline(yintercept = 2.5, linetype = "dashed")+
  ylim(c(0,3)) +
  ggtitle("Multiple stable state model. Lines = stable/unstable states.")
```

Multiple stable state model. Lines = stable/unstable states.




```
out1c %>%
  ggplot(aes(N,P)) +
  geom_path(aes(x = out3$N, y = out3$P), lwd = 1.1, color = "red") +
  geom_path(lwd=1.1, color = "black") +
  geom_path(aes(x = out1d$N, y = out1d$P), lwd = 1.1, color = "black") +
  xlim(c(0,3))+
  ylim(c(0,3))+
  ggtitle("Multiple stable state model. Stable equilibrium at (1.75,1.1) and (2.5,0). Unstable equilibr
```


Multiple stable state model. Stable equilibrium at (1.75,1.1) and (2.5,0). Unst
P
N
0
1
2
3

Compute the quasi-potentials for both models. From Abbott & Nolting (2016).

Compute the local quasi-potentials

```
# determine equilibrium points
# xspace <- seq(from = 1, to = 3, length.out = 10)
# yspace <- seq(from = 1, to = 3, length.out = 10)
# xspace_l <- length(x=xspace)
# yspace_l <- length(x=yspace)
# space.mat <- matrix(data=NA, nrow=xspace_l*yspace_l, ncol=2)
# # numerically find equilibria...
# for (i in 1:xspace_l){
#   for (j in 1:yspace_l){
#     y <- c(N=xspace[i], P=yspace[j])
#     STO <- stode(y = y, func = predprey2, parms = model.parms, positive = T)
#     space.mat[((i-1)*xspace_l)+j),] <- STO$y
#   }
# }
# eqs <- unique(x=round(x=space.mat,digits=3))

bounds.x <- c(-0.5,5)
bounds.y <- c(-0.5,5)
step.number.x <- 1000
step.number.y <- 1000
eq1.x <- 1.5
eq1.y <- 1.2

# maybe revise equations?...super dumb
# new ones...beta = 2, gamma = 2.6, delta = 2.5, alpha = 4.5

parms.eqn.x <- "1*x*(1-x/2.5) - 4.5*x*y/(x^2 + 4.5*(x+1))"
parms.eqn.y <- "5*4.5*x*y/(x^2 + 4.5*(x+1)) - 2.5*y"

# have to analyze at EACH stable state
# 1 stable state
eq1.local <- QPotential(x.rhs = parms.eqn.x, x.start = eq1.x, x.bound = bounds.x,
                        x.num.steps = step.number.x, y.rhs = parms.eqn.y, y.start = eq1.y,
                        y.bound = bounds.y, y.num.steps = step.number.y,
                        verboseC = F)
```

```
## hx = 0.00550551
## hy = 0.00550551
## Finished initializing a bunch of matrices in param() function
## cputime = 12.815
## Initial count = 4
## current count = 1000
## 49847    (585    2) is accepted, g=0.0606
## Final count = 1521
```

```
###################

eq2a.x <- 1.73
```

```
eq2a.y <- 1.066
eq2b.x <- 2.5
eq2b.y <- 0
# beta = 2, gamma = 2.6, delta = 2.5, alpha = 4.1
parms.eqn.x.2 <- "1*x*(1-x/2.5) - 4.1*x*y/(x^2 + 4.1*(x+1))"
parms.eqn.y.2 <- "5*4.1*x*y/(x^2 + 4.1*(x+1)) - 2.5*y"

# 2 stable states
eq2a.local <- QPotential(x.rhs = parms.eqn.x.2, x.start = eq2a.x, x.bound = bounds.x,
                         x.num.steps = step.number.x, y.rhs = parms.eqn.y.2, y.start = eq2a.y,
                         y.bound = bounds.y, y.num.steps = step.number.y,
                         verboseC = F)
```

```
## hx = 0.00550551
## hy = 0.00550551
## Finished initializing a bunch of matrices in param() function
## cputime = 13.183
## Initial count = 4
## 20254    (584    2) is accepted, g=0.0186
## Final count = 1219
```

```
eq2b.local <- QPotential(x.rhs = parms.eqn.x.2, x.start = eq2b.x, x.bound = bounds.x,
                         x.num.steps = step.number.x, y.rhs = parms.eqn.y.2, y.start = eq2b.y,
                         y.bound = bounds.y, y.num.steps = step.number.y,
                         verboseC = F)
```

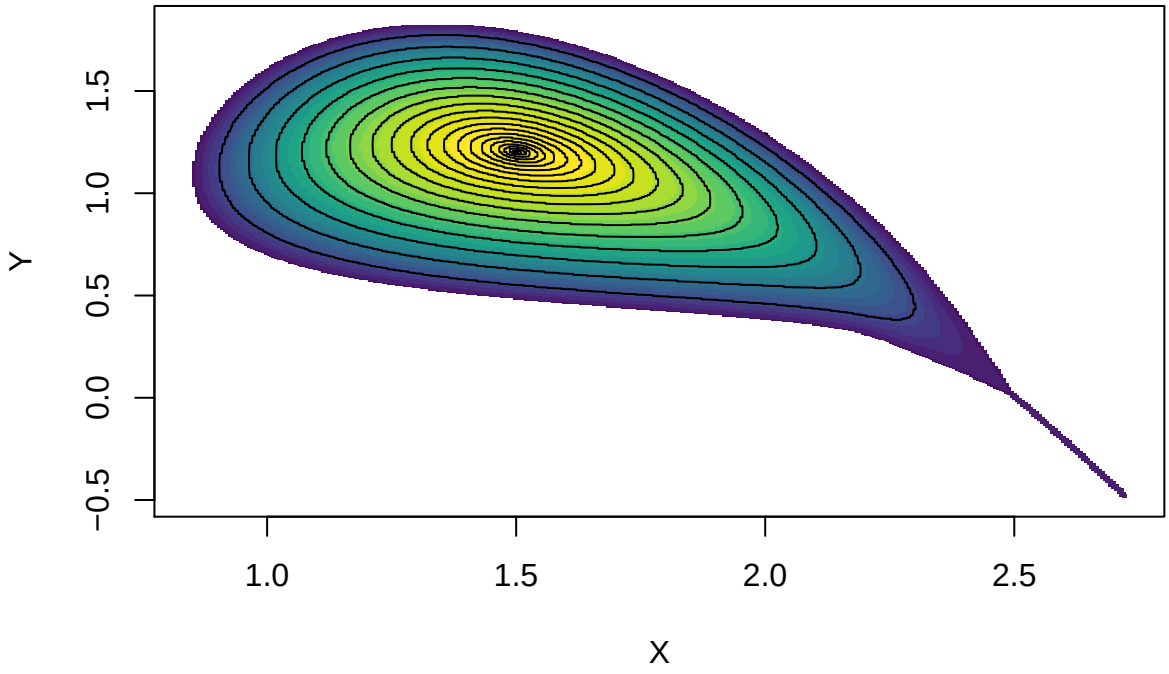
```
## hx = 0.00550551
## hy = 0.00550551
## Finished initializing a bunch of matrices in param() function
## cputime = 12.294
## Initial count = 4
## 13657    (584    2) is accepted, g=0.0073
## Final count = 1115
```

Global quasi-potential calculation

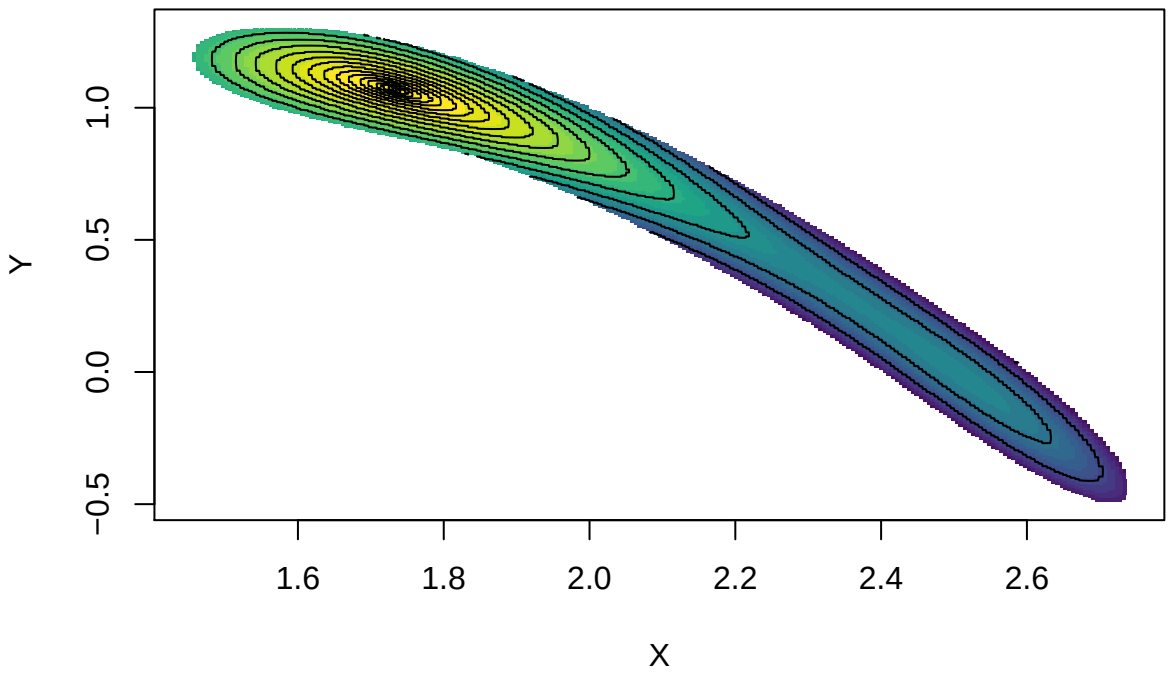
```
# one state..possibly redundant
unstable.x <- c(0,2.5)
unstable.y <- c(0,0)
ex1.global <- QPGlobal(local.surfaces = list(eq1.local),
                       unstable.eq.x = unstable.x, unstable.eq.y = unstable.y,
                       x.bound = bounds.x, y.bound = bounds.y)

# multi-stable state
unstable.x <- c(0, 2.37)
unstable.y <- c(0, 0.246)
ex2.global <- QPGlobal(local.surfaces = list(eq2a.local, eq2b.local),
                       unstable.eq.x = unstable.x, unstable.eq.y = unstable.y,
                       x.bound = bounds.x, y.bound = bounds.y)
```

Visualize surfaces

```
# look at local quasi-potential surfaces
#QPContour(surface = eq1.local, dens = c(1000,1000), x.bound = bounds.x, y.bound = bounds.y)
#QPContour(surface = eq2a.local, dens = c(1000,1000), x.bound = bounds.x, y.bound = bounds.y)
#QPContour(surface = eq2b.local, dens = c(1000,1000), x.bound = bounds.x, y.bound = bounds.y)

# look at global quasi-potential surface as surface plots
QPContour(surface = ex1.global, dens = c(1000,1000), x.bound = bounds.x, y.bound = bounds.y,c.parm=5)
```




```
QPContour(surface = ex2.global, dens = c(1000,1000), x.bound = bounds.x, y.bound = bounds.y,c.parm=5)
```




```
# 3D surface
#persp3d(x = ex2.global, col = "#FF5500")
#persp3D(z = ex2.global, contour = F)

dens.sub <- c(4000,4000)
global.sub <- ex2.global[round(seq(1,nrow(ex2.global),length.out=dens.sub[1])),
                         round(seq(1,ncol(ex2.global),length.out=dens.sub[2]))]

#persp3d(x = global.sub, col = "orange", expand = 1.1)
```