Quasi-potentials: A framework for analyzing stochastic dynamical systems in ecology

Quasi-potential Tutorial

Dr. Chris Stieha, Case Western Reserve University

Dr. Ben Nolting, California State University, Chico
Case Western Reserve University

CASE WESTERN RESERVE UNIVERSITY EST. 1826

think beyond the possible℠

QPot: Quasi-Potential Analysis for Stochastic Differential Equations

A R package that introduces a suite of functions to assist in analysis of SDEs

Key functions

Function	Main arguments	Description
`TSTraj()`	Deterministic skeleton, σ, T, Δt	Creates a realization (time series) of the stochastic differential equations.
`TSPlot()`	`TSTraj()` output	Plots a realization of the stochastic differential equations, with an optional histogram side-plot. Plots can additionally be two-dimensional, which show realizations in (X, Y)-space.
`TSDensity()`	`TSTraj()` output	Creates a density plot of a trajectory in (X, Y)-space in one or two dimensions.
`QPotential()`	Deterministic skeleton, stable equilibria, bounds, mesh (number of divisions along each axis)	Creates a matrix corresponding to a discretized version of the local quasi-potential function for each equilibrium.
`QPGlobal()`	Local quasi-potential matrices, unstable equilibria	Creates a global quasi-potential surface.
`QPInterp()`	Global quasi-potential, $(x\,y)$-coordinates	Evaluates the global quasi-potential at (x, y).
`QPContour()`	Global quasi-potential	Creates a contour plot of the quasi-potential.
`VecDecomAll()`	Global quasi-potential, deterministic skeleton, bounds	Creates three vector fields: the deterministic skeleton, the negative gradient of the quasi-potential, and the remainder vector field. To find each field individually, the functions `VecDecomVec()`, `VecDecomGrad()`, or `VecDecomRem()` can be used.
`VecDecomPlot()`	Deterministic skeleton, gradient, or remainder field	Creates a vector field plot for the vector, gradient, or remainder field.

Moore et al. *in press*

Steps for the analysis of Stochastic Differential Equations

1) Analyze the deterministic model
   Find equilibria, determine stability

2) Simulate the equations, include stochasticity

3) Calculate the local quasi-potentials
   For each stable equillibrium

4) Calculate the global quasi-potential
   (if only one stable equilibrium, then local = global)

5) Visualize the global quasi-potential

6) Visualize the vector field decompositions

Moore et al. *in press*

Our Equations: Lotka-Volterra Competition

Species X

$$\frac{dx}{dt} = r_x x \frac{K_x - x - \alpha_{xy} y}{K_x}$$

Species Y

$$\frac{dy}{dt} = r_y y \frac{K_y - y - \alpha_{yx} x}{K_y}$$

x = state variable, density of species x

y = state variable, density of species y

Our Equations: Lotka-Volterra Competition

Species X

$$\frac{dx}{dt} = r_x x \frac{K_x - x - \alpha_{xy} y}{K_x}$$

Species Y

$$\frac{dy}{dt} = r_y y \frac{K_y - y - \alpha_{yx} x}{K_y}$$

x = state variable, density of species x

y = state variable, density of species y

r_x = growth rate of species x

r_y = growth rate of species y

Our Equations: Lotka-Volterra Competition

Species X

$$\frac{dx}{dt} = r_x x \frac{K_x - x - \alpha_{xy} y}{K_x}$$

Species Y

$$\frac{dy}{dt} = r_y y \frac{K_y - y - \alpha_{yx} x}{K_y}$$

x = state variable, density of species x

y = state variable, density of species y

r_x = growth rate of species x

r_y = growth rate of species y

K_x = carrying capacity of species x

K_y = carrying capacity of species y

Our Equations: Lotka-Volterra Competition

Species X

$$\frac{dx}{dt} = r_x x \frac{K_x - x - \alpha_{xy} y}{K_x}$$

Species Y

$$\frac{dy}{dt} = r_y y \frac{K_y - y - \alpha_{yx} x}{K_y}$$

x = state variable, density of species x

y = state variable, density of species y

r_x = growth rate of species x

r_y = growth rate of species y

K_x = carrying capacity of species x

K_y = carrying capacity of species y

α_{xy} = competition coefficient,
converts individuals of species y into species x equivalents

α_{yx} = competition coefficient,
converts individuals of species x into species y equivalents

Our Equations: Lotka-Volterra Competition

Species X

$$\frac{dx}{dt} = r_x x \frac{K_x - x - \alpha_{xy} y}{K_x}$$

Species Y

$$\frac{dy}{dt} = r_y y \frac{K_y - y - \alpha_{yx} x}{K_y}$$

Four different outcomes based on parameter values

Our Equations: Lotka-Volterra Competition

Species X

$$\frac{dx}{dt} = r_x x \frac{K_x - x - \alpha_{xy} y}{K_x}$$

Species Y

$$\frac{dy}{dt} = r_y y \frac{K_y - y - \alpha_{yx} x}{K_y}$$

Four different outcomes based on parameter values

X wins $K_x > \frac{K_y}{\alpha_{yx}}$ $K_y < \frac{K_x}{\alpha_{xy}}$

Our Equations: Lotka-Volterra Competition

Species X

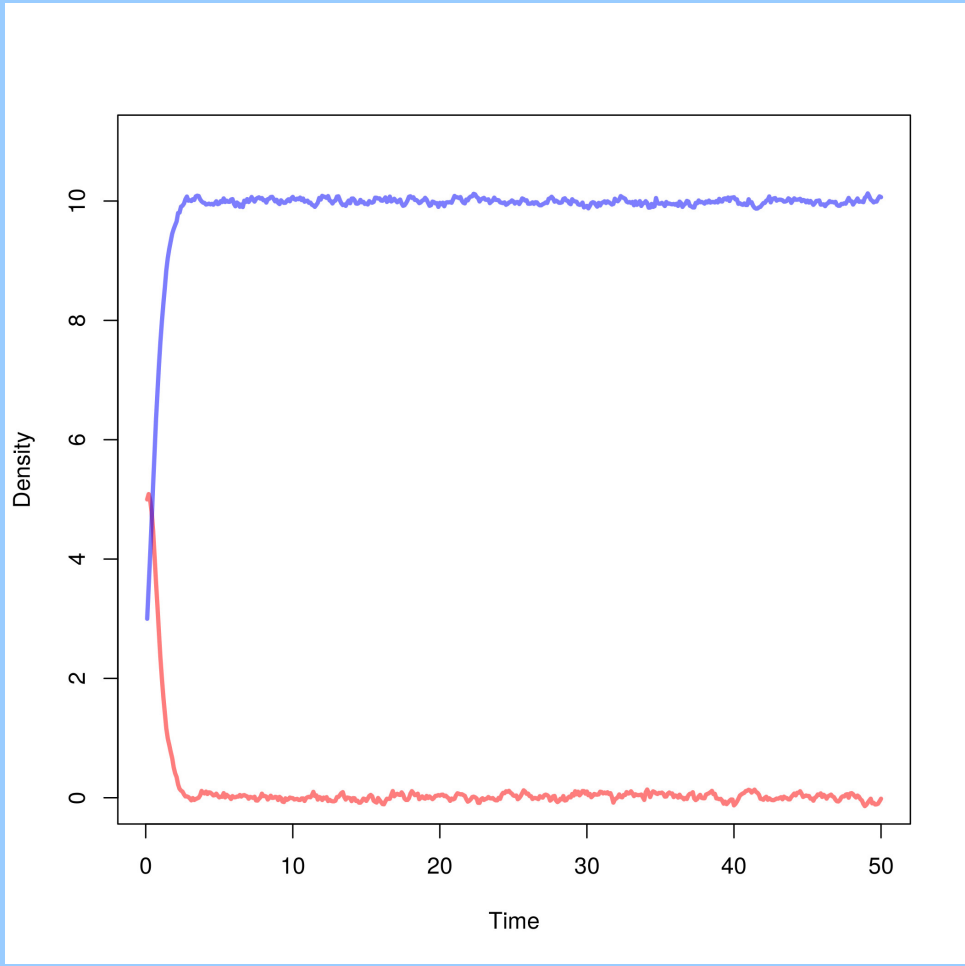
$$\frac{dx}{dt} = r_x x \frac{K_x - x - \alpha_{xy} y}{K_x}$$

Species Y

$$\frac{dy}{dt} = r_y y \frac{K_y - y - \alpha_{yx} x}{K_y}$$

Four different outcomes based on parameter values

X wins $K_x > \frac{K_y}{\alpha_{yx}}$ $K_y < \frac{K_x}{\alpha_{xy}}$



Our Equations: Lotka-Volterra Competition

Species X

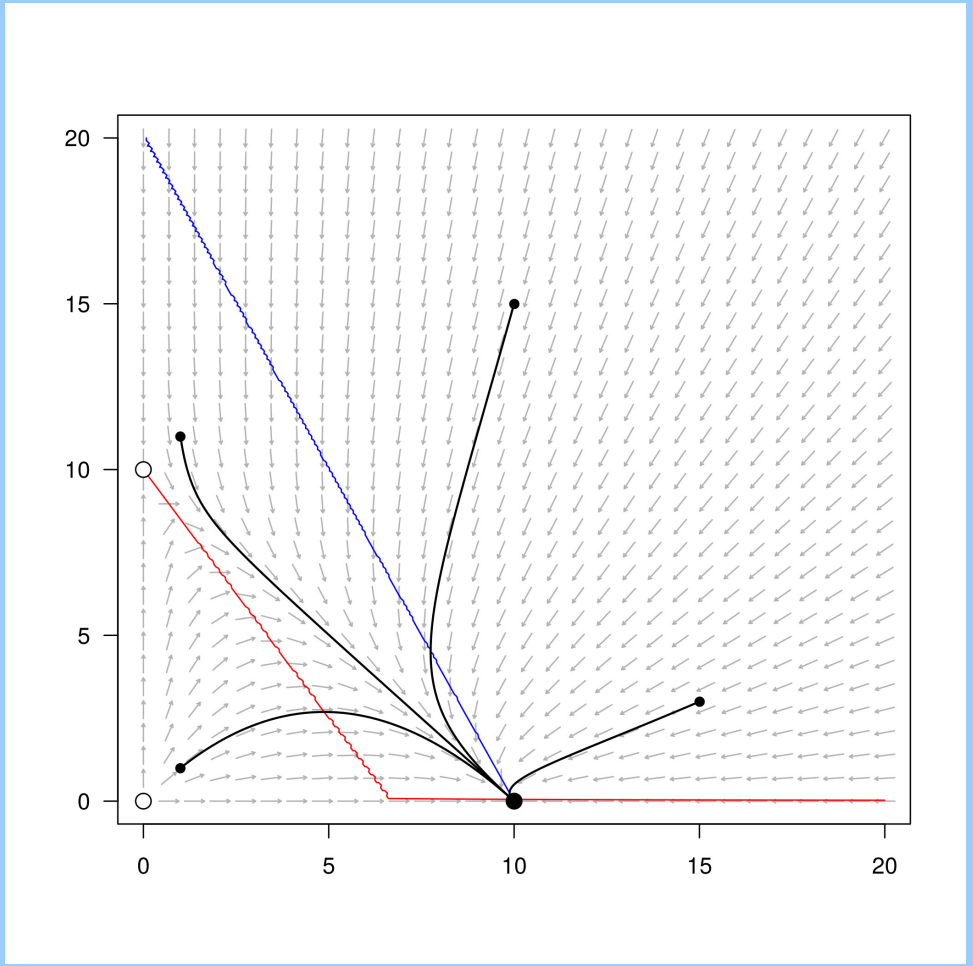
$$\frac{dx}{dt} = r_x x \frac{K_x - x - \alpha_{xy} y}{K_x}$$

Species Y

$$\frac{dy}{dt} = r_y y \frac{K_y - y - \alpha_{yx} x}{K_y}$$

Four different outcomes based on parameter values

X wins $K_x > \frac{K_y}{\alpha_{yx}}$ $K_y < \frac{K_x}{\alpha_{xy}}$



Our Equations: Lotka-Volterra Competition

Species X

$$\frac{dx}{dt} = r_x x \frac{K_x - x - \alpha_{xy} y}{K_x}$$

Species Y

$$\frac{dy}{dt} = r_y y \frac{K_y - y - \alpha_{yx} x}{K_y}$$

Four different outcomes based on parameter values

X wins $K_x > \frac{K_y}{\alpha_{yx}}$ $K_y < \frac{K_x}{\alpha_{xy}}$

Y wins $K_x < \frac{K_y}{\alpha_{yx}}$ $K_y > \frac{K_x}{\alpha_{xy}}$

Our Equations: Lotka-Volterra Competition

Species X

$$\frac{dx}{dt} = r_x x \frac{K_x - x - \alpha_{xy} y}{K_x}$$

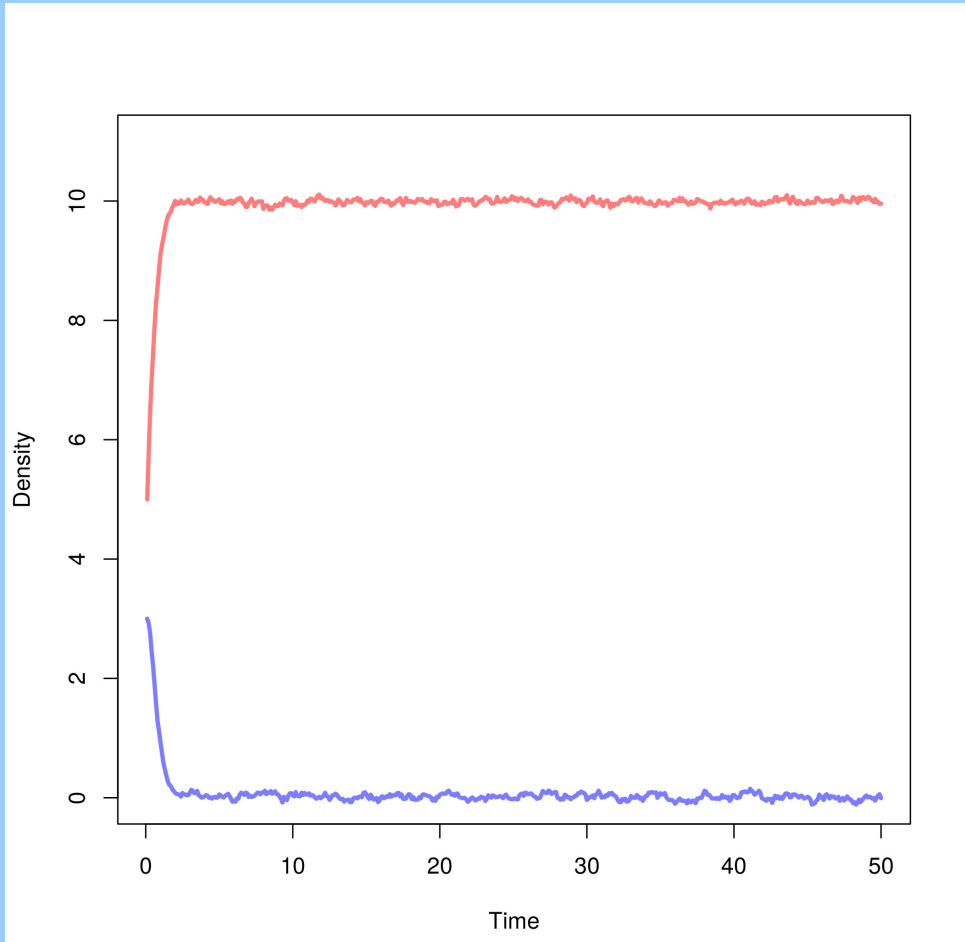
Species Y

$$\frac{dy}{dt} = r_y y \frac{K_y - y - \alpha_{yx} x}{K_y}$$

Four different outcomes based on parameter values

X wins $K_x > \frac{K_y}{\alpha_{yx}}$ $K_y < \frac{K_x}{\alpha_{xy}}$

Y wins $K_x < \frac{K_y}{\alpha_{yx}}$ $K_y > \frac{K_x}{\alpha_{xy}}$



Our Equations: Lotka-Volterra Competition

Species X

$$\frac{dx}{dt} = r_x x \frac{K_x - x - \alpha_{xy} y}{K_x}$$

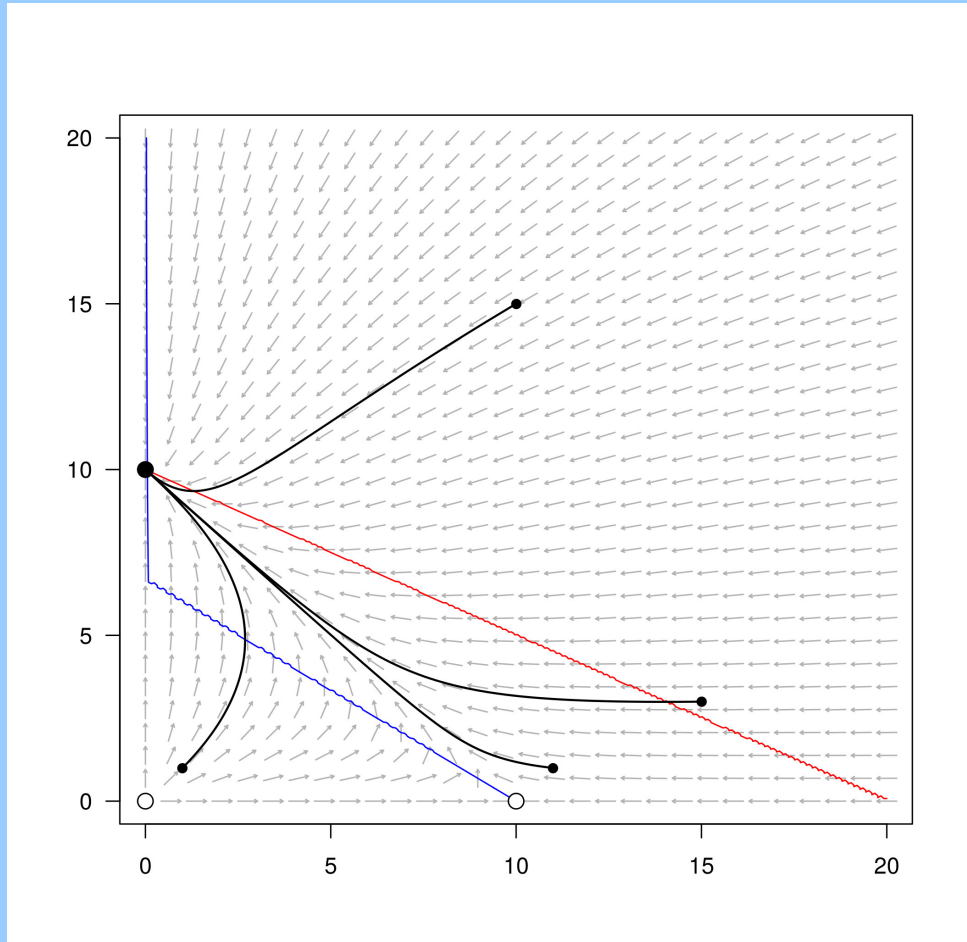
Species Y

$$\frac{dy}{dt} = r_y y \frac{K_y - y - \alpha_{yx} x}{K_y}$$

Four different outcomes based on parameter values

X wins $K_x > \frac{K_y}{\alpha_{yx}}$ $K_y < \frac{K_x}{\alpha_{xy}}$

Y wins $K_x < \frac{K_y}{\alpha_{yx}}$ $K_y > \frac{K_x}{\alpha_{xy}}$



Our Equations: Lotka-Volterra Competition

Species X

$$\frac{dx}{dt} = r_x x \frac{K_x - x - \alpha_{xy} y}{K_x}$$

Species Y

$$\frac{dy}{dt} = r_y y \frac{K_y - y - \alpha_{yx} x}{K_y}$$

Four different outcomes based on parameter values

X wins $K_x > \frac{K_y}{\alpha_{yx}}$ $K_y < \frac{K_x}{\alpha_{xy}}$

Y wins $K_x < \frac{K_y}{\alpha_{yx}}$ $K_y > \frac{K_x}{\alpha_{xy}}$

Initial condition dependence $K_x > \frac{K_y}{\alpha_{yx}}$ $K_y > \frac{K_x}{\alpha_{xy}}$

Our Equations: Lotka-Volterra Competition

Species X

$$\frac{dx}{dt} = r_x x \frac{K_x - x - \alpha_{xy} y}{K_x}$$

Species Y

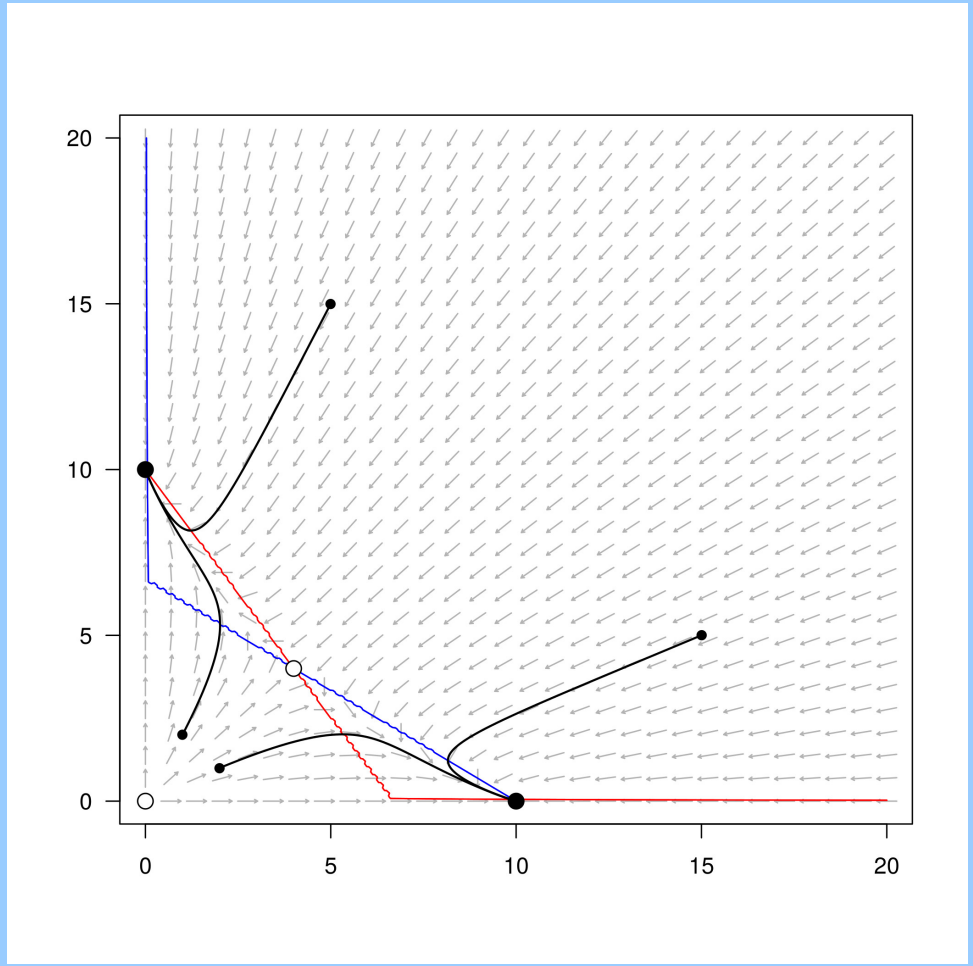
$$\frac{dy}{dt} = r_y y \frac{K_y - y - \alpha_{yx} x}{K_y}$$

Four different outcomes based on parameter values

X wins $K_x > \frac{K_y}{\alpha_{yx}}$ $K_y < \frac{K_x}{\alpha_{xy}}$

Y wins $K_x < \frac{K_y}{\alpha_{yx}}$ $K_y > \frac{K_x}{\alpha_{xy}}$

Initial condition dependence $K_x > \frac{K_y}{\alpha_{yx}}$ $K_y > \frac{K_x}{\alpha_{xy}}$



Our Equations: Lotka-Volterra Competition

Species X

$$\frac{dx}{dt} = r_x x \frac{K_x - x - \alpha_{xy} y}{K_x}$$

Species Y

$$\frac{dy}{dt} = r_y y \frac{K_y - y - \alpha_{yx} x}{K_y}$$

Four different outcomes based on parameter values

X wins $K_x > \frac{K_y}{\alpha_{yx}}$ $K_y < \frac{K_x}{\alpha_{xy}}$

Y wins $K_x < \frac{K_y}{\alpha_{yx}}$ $K_y > \frac{K_x}{\alpha_{xy}}$

Initial condition dependence $K_x > \frac{K_y}{\alpha_{yx}}$ $K_y > \frac{K_x}{\alpha_{xy}}$

Coexist $K_x < \frac{K_y}{\alpha_{yx}}$ $K_y < \frac{K_x}{\alpha_{xy}}$

Our Equations: Lotka-Volterra Competition

Species X

$$\frac{dx}{dt} = r_x x \frac{K_x - x - \alpha_{xy} y}{K_x}$$

Species Y

$$\frac{dy}{dt} = r_y y \frac{K_y - y - \alpha_{yx} x}{K_y}$$

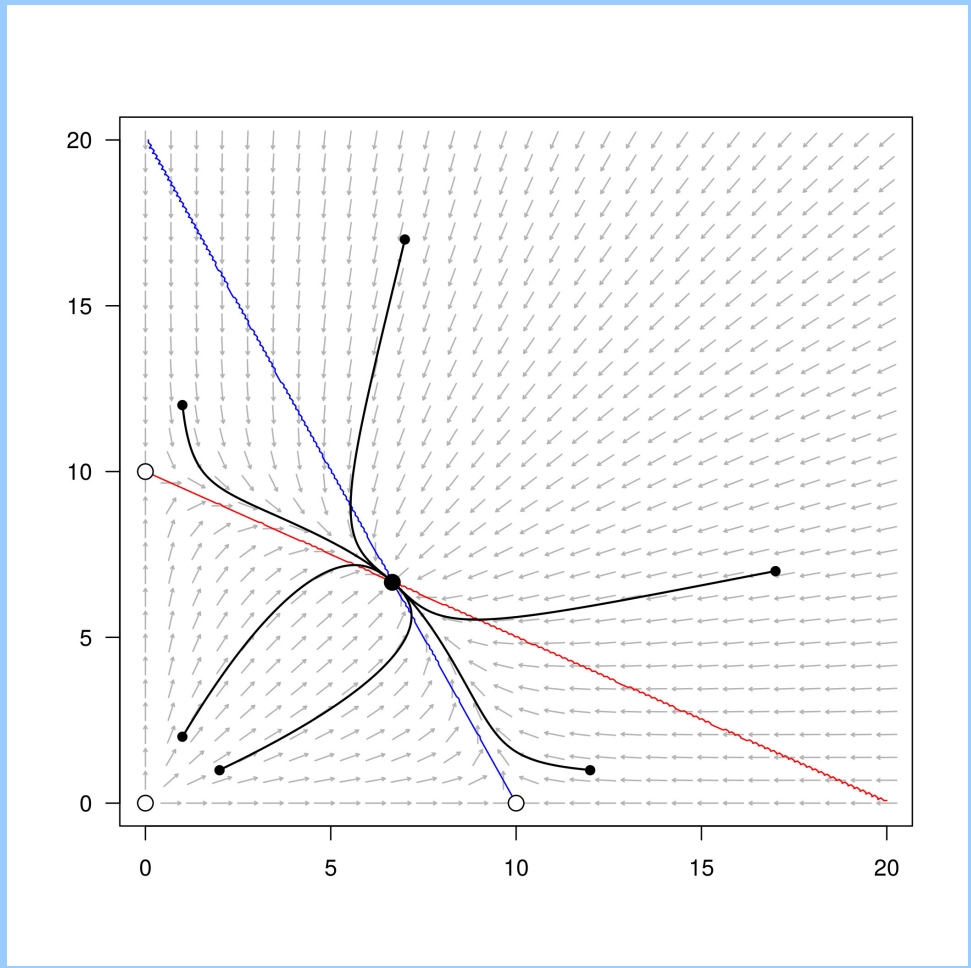
Four different outcomes based on parameter values

X wins $K_x > \frac{K_y}{\alpha_{yx}}$ $K_y < \frac{K_x}{\alpha_{xy}}$

Y wins $K_x < \frac{K_y}{\alpha_{yx}}$ $K_y > \frac{K_x}{\alpha_{xy}}$

Initial condition dependence $K_x > \frac{K_y}{\alpha_{yx}}$ $K_y > \frac{K_x}{\alpha_{xy}}$

Coexist $K_x < \frac{K_y}{\alpha_{yx}}$ $K_y < \frac{K_x}{\alpha_{xy}}$



Our Equations: Lotka-Volterra Competition

Species X

$$\frac{dx}{dt} = r_x x \frac{K_x - x - \alpha_{xy} y}{K_x}$$

Species Y

$$\frac{dy}{dt} = r_y y \frac{K_y - y - \alpha_{yx} x}{K_y}$$

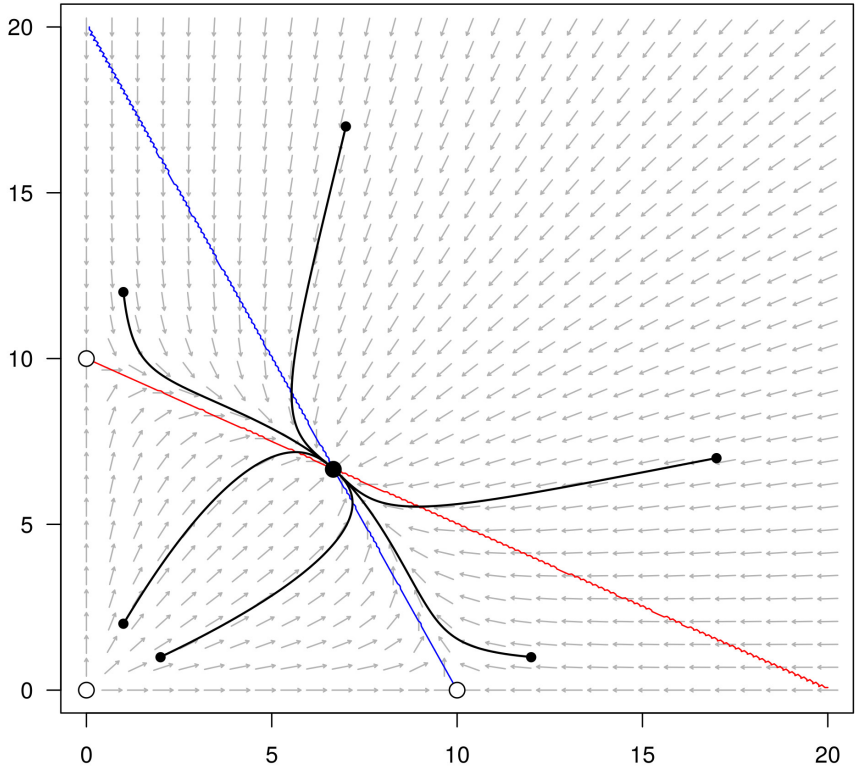
Four different outcomes based on parameter values

X wins $K_x > \frac{K_y}{\alpha_{yx}}$ $K_y < \frac{K_x}{\alpha_{xy}}$

Y wins $K_x < \frac{K_y}{\alpha_{yx}}$ $K_y > \frac{K_x}{\alpha_{xy}}$

Initial condition dependence $K_x > \frac{K_y}{\alpha_{yx}}$ $K_y > \frac{K_x}{\alpha_{xy}}$

Coexist $K_x < \frac{K_y}{\alpha_{yx}}$ $K_y < \frac{K_x}{\alpha_{xy}}$



Lotka-Volterra Competition: Coexistence

Coexistence $K_x < \frac{K_y}{\alpha_{yx}}$ $K_y < \frac{K_x}{\alpha_{xy}}$

1) Analyze the deterministic model
Find equilibria, determine stability

$$0 = r_x x \frac{K_x - x - \alpha_{xy} y}{K_x}$$

$$0 = r_y y \frac{K_y - y - \alpha_{yx} x}{K_y}$$

(0,0) unstable equilibrium

$(K_x,0)$ unstable equilibrium

$(0,K_y)$ unstable equilibrium

$$\left(\frac{K_x - \alpha_{xy} K_y}{1 - \alpha_{xy}\alpha_{yx}}, \frac{K_y - \alpha_{yx} K_x}{1 - \alpha_{xy}\alpha_{yx}} \right)$$

stable equilibrium

Lotka-Volterra Competition: Coexistence

Install these packages: (order is wrong in handout)

```
install.packages('Qpot')
install.packages('plot3D')
install.packages('deSolve')
install.packages('phaseR')
```

in R, anything after the # is ignored

Lotka-Volterra Competition: Coexistence

Install these packages: (order is wrong in handout)

```
install.packages('Qpot')
install.packages('plot3D')
install.packages('deSolve')
install.packages('phaseR')

library(QPot)
library(plot3D)
library(deSolve)
library(phaseR)
```

in R, anything after the # is ignored

Lotka-Volterra Competition: Coexistence

1) Analyze the deterministic model

Find equilibria, determine stability

```
# parameters that lead to coexistence
model.state.coexist <- c(rx = 4, ry = 4, kx = 10, ky = 10, alphaxy = 0.5, alphayx = 0.5)

require(phaseR)
competition.model <- function(t, y, parameters){
      x <- y[1]                                  # initial state of species x
      y <- y[2]                                  # initial state of species y
      rx <- parameters["rx"]                     # growth rate of species x
      ry <- parameters["ry"]                     # growth rate of species y
      alphaxy <- parameters["alphaxy"]   # competition coefficient
      alphayx <- parameters["alphayx"]   # competition coefficient
      kx <- parameters["kx"]                     # carrying capacity species x
      ky <- parameters["ky"]                     # carrying capacity species y
      dx <- rx*x*(kx - x - alphaxy*y)*(1/kx) # equation for species x
      dy <- ry*y*(ky - y - alphayx*x)*(1/ky) # equation for species y
      list(c(dx,dy))
}
```

Lotka-Volterra Competition: Coexistence

1) Analyze the deterministic model
Find equilibria, determine stability

```
# this function will draw the vector field that describes how the densities will change
flowField(deriv = competition.model, x.lim = c(0, 20), y.lim = c(0, 20),
          parameters = model.state.coexist, add = F, points = 30)

# Draw the null clines, where dx/dt = 0 or dy/dt = 0
supp.print <- nullclines(deriv = competition.model, x.lim = c(0, 20),
                         y.lim = c(0, 20), parameters = model.state.coexist,
                         col = c("blue", "red"), points = 250)

#plot points on graph for stable equilbrium (black) and unstable (white)
points(x = 0, y = 10 , pch = 21 , bg = "white" , cex = 1.5)
points(x = 10, y = 0 , pch = 21 , bg = "white" , cex = 1.5)
points(x = 6.667, y = 6.667 , pch = 21 , bg = "black" , cex = 1.5)
points(x = 0, y = 0 , pch = 21 , bg = "white" , cex = 1.5)

xstart = runif(1,0,20)    #1 random point between 0 and 20 from uniform dist
ystart = runif(1,0,20)    # change this to a value to set start point
traj <- trajectory(competition.model,y0=c(xstart,ystart),
                         parameters = model.state.coexist , t.end = 250)
```

Lotka-Volterra Competition: Coexistence

1) Analyze the deterministic model

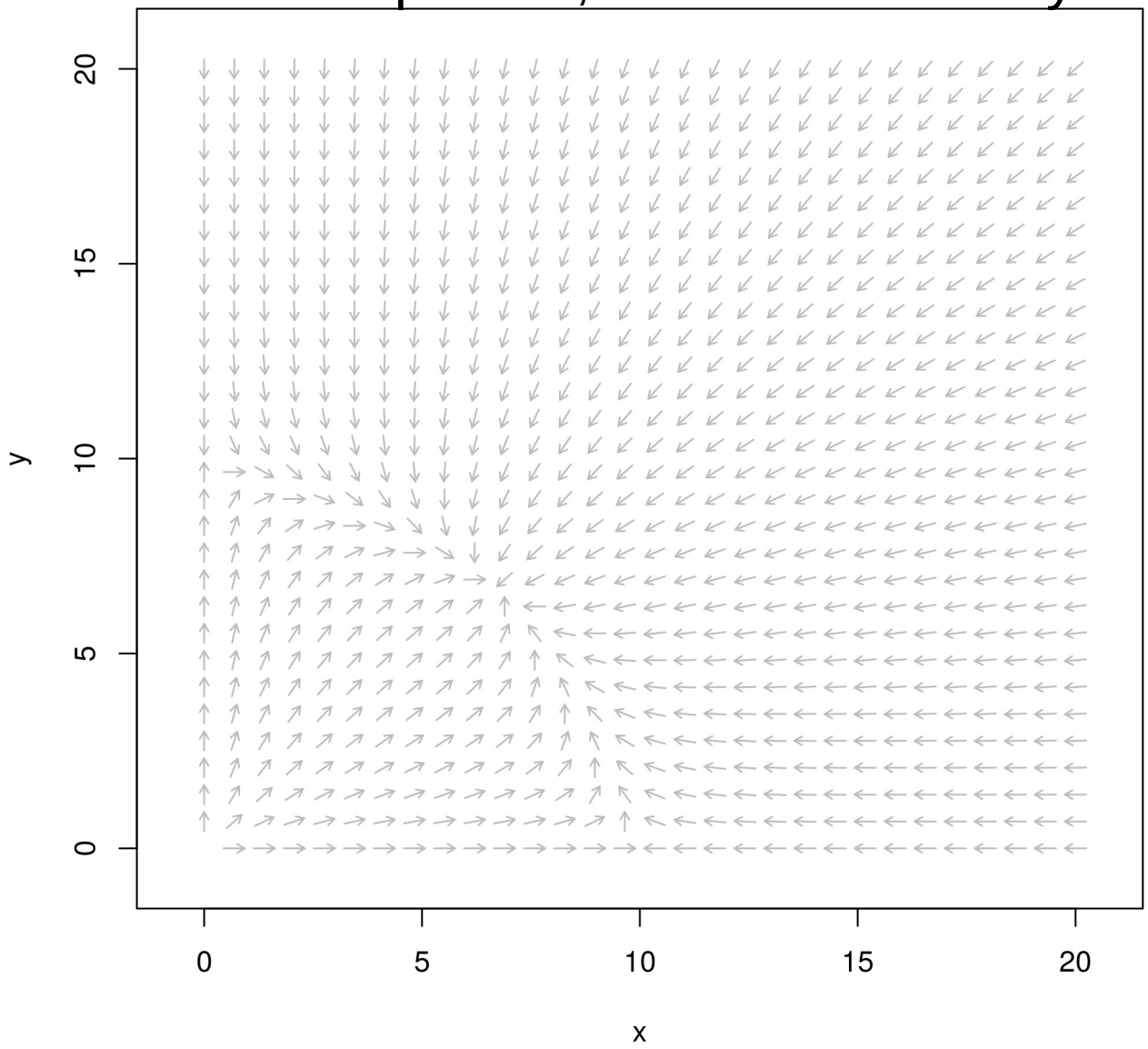
Find equilibria, determine stability

```
flowField(deriv = competition.model, x.lim = c(0, 20), y.lim = c(0, 20),
          parameters = model.state.coexist, add = F, points = 30)
```

Lotka-Volterra Competition: Coexistence

1) Analyze the deterministic model

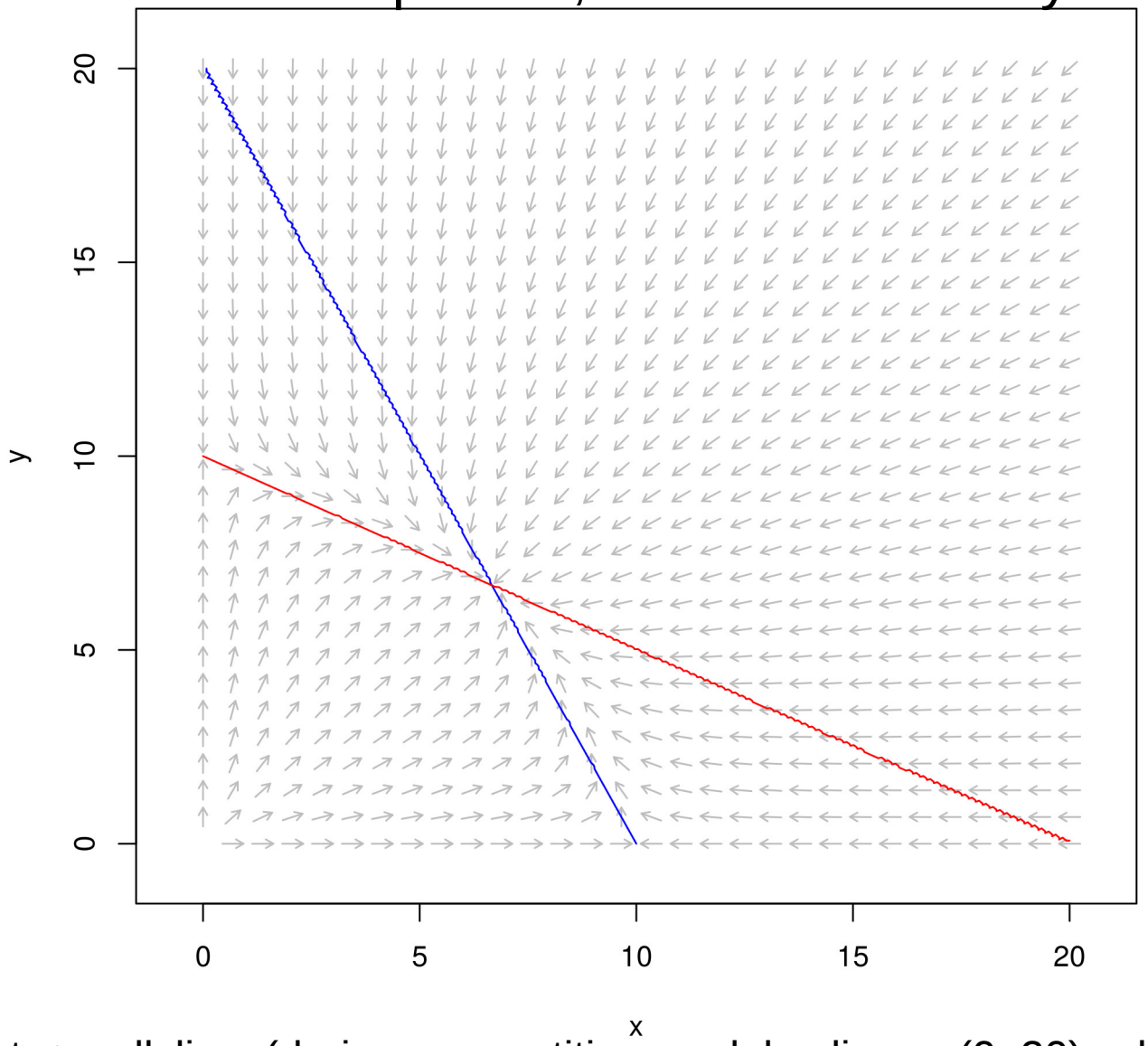
Find equilibria, determine stability



flowField(deriv = competition.model, x.lim = c(0, 20), y.lim = c(0, 20),
parameters = model.state.coexist, add = F, points = 30)

Lotka-Volterra Competition: Coexistence

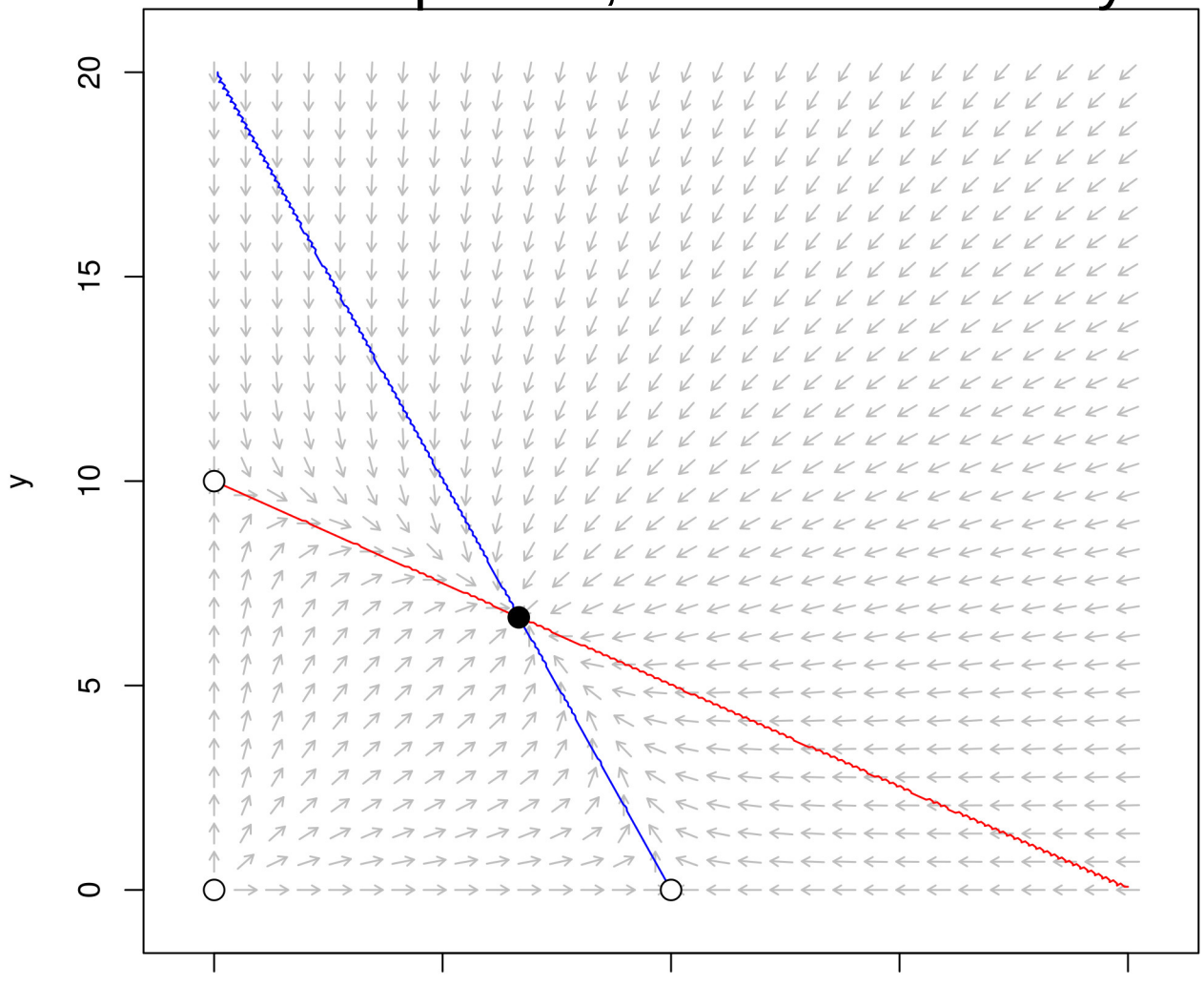
1) Analyze the deterministic model
Find equilibria, determine stability




```
supp.print <- nullclines(deriv = competition.model, x.lim = c(0, 20), y.lim = c(0, 20),
parameters = model.state.coexist, col = c("blue", "red"), points = 250)
```

Lotka-Volterra Competition: Coexistence

1) Analyze the deterministic model
Find equilibria, determine stability

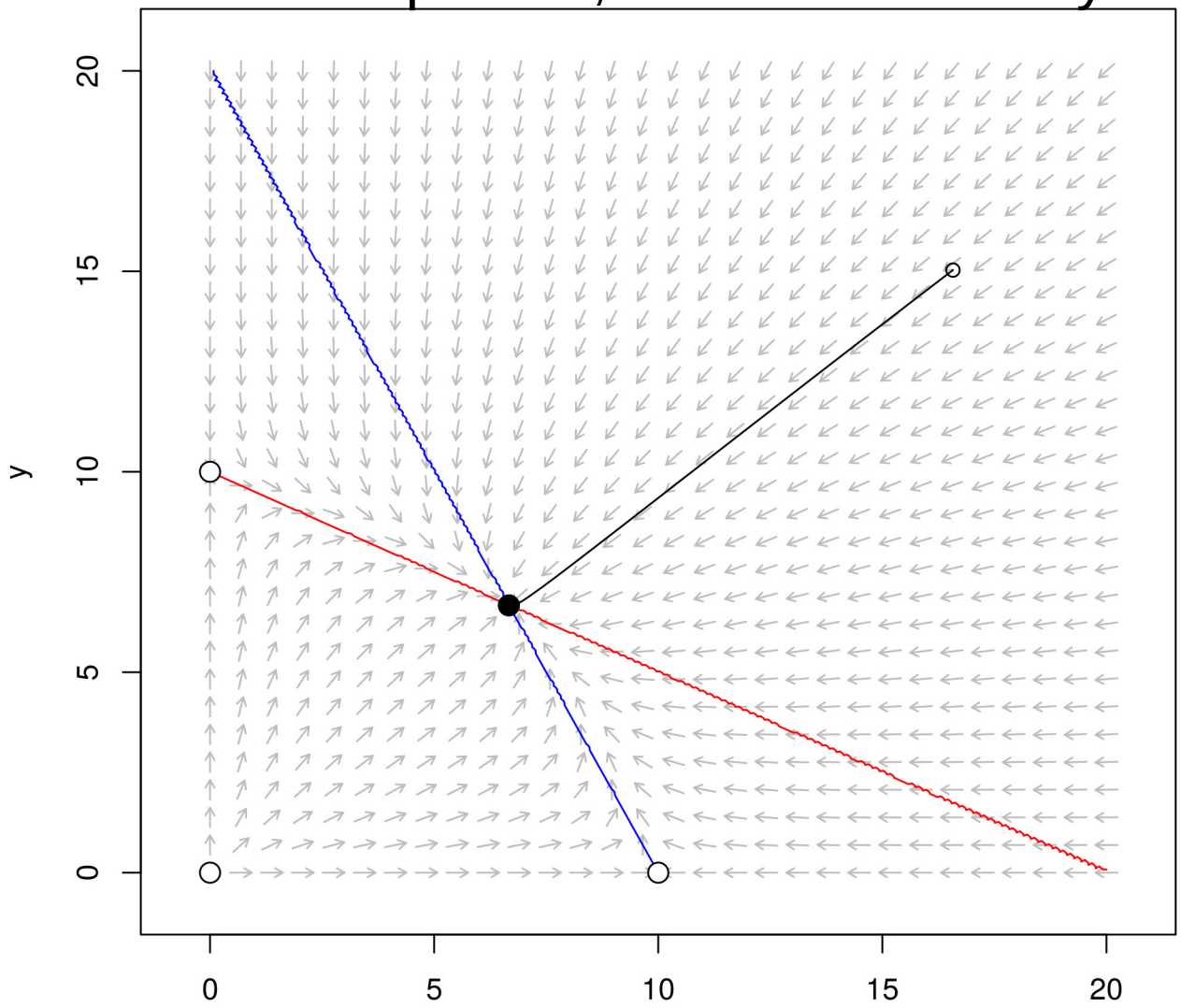


points(0,10 , pch = 21 , col = "black" , bg = "white" , cex = 1.5)
points(10,0 , pch = 21 , col = "black" , bg = "white" , cex = 1.5)
points(6.667,6.667 , pch = 21 , col = "black" , bg = "black" , cex = 1.5)
points(0,0 , pch = 21 , col = "black" , bg = "white" , cex = 1.5)

Lotka-Volterra Competition: Coexistence

1) Analyze the deterministic model

Find equilibria, determine stability



xstart = runif(1,0,20); ystart = runif(1,0,20)

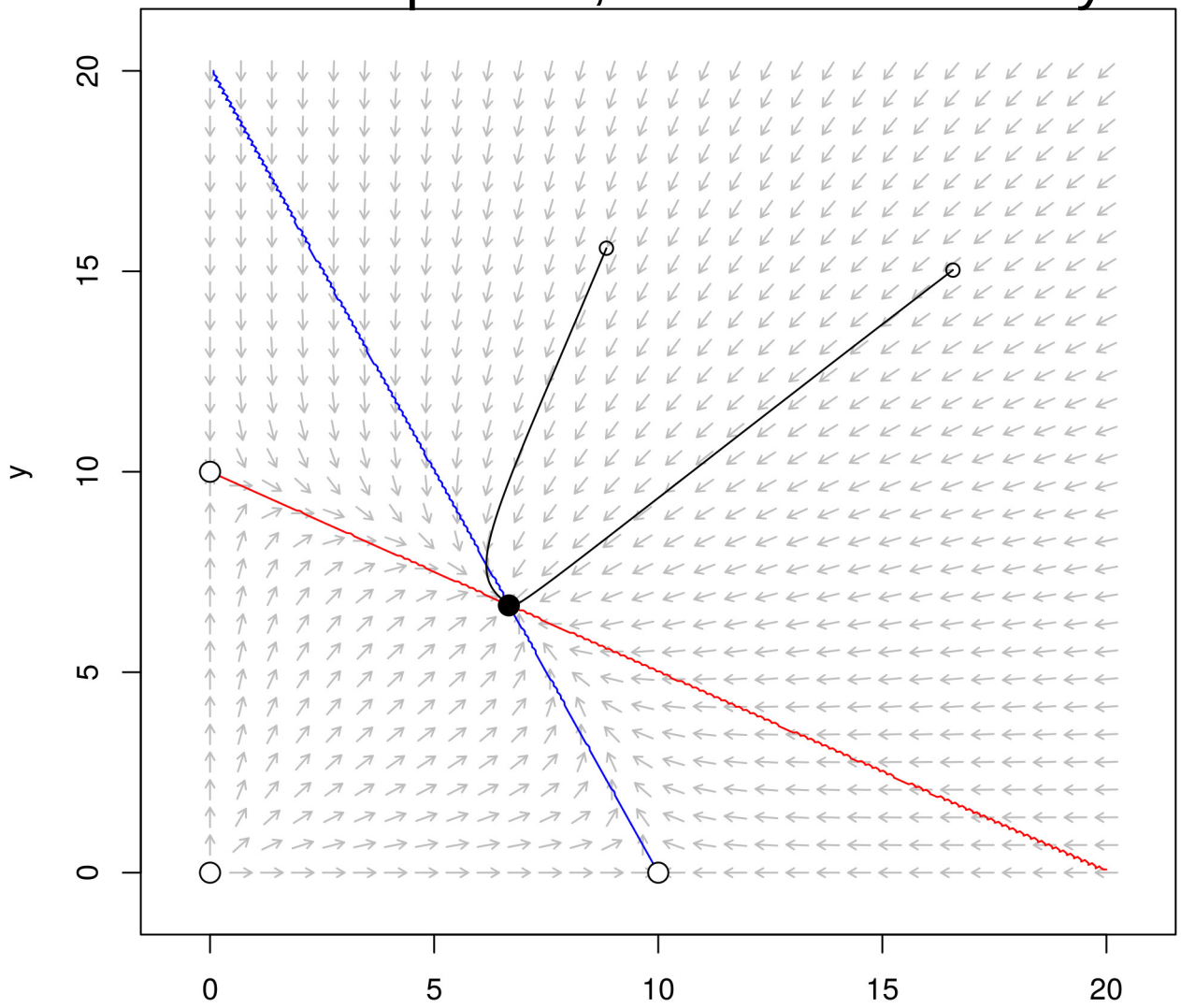
traj <- trajectory(competition.model,y0=c(xstart,ystart),

parameters = model.state.coexist , t.end = 250)

Lotka-Volterra Competition: Coexistence

1) Analyze the deterministic model

Find equilibria, determine stability



xstart = runif(1,0,20); ystart = runif(1,0,20)

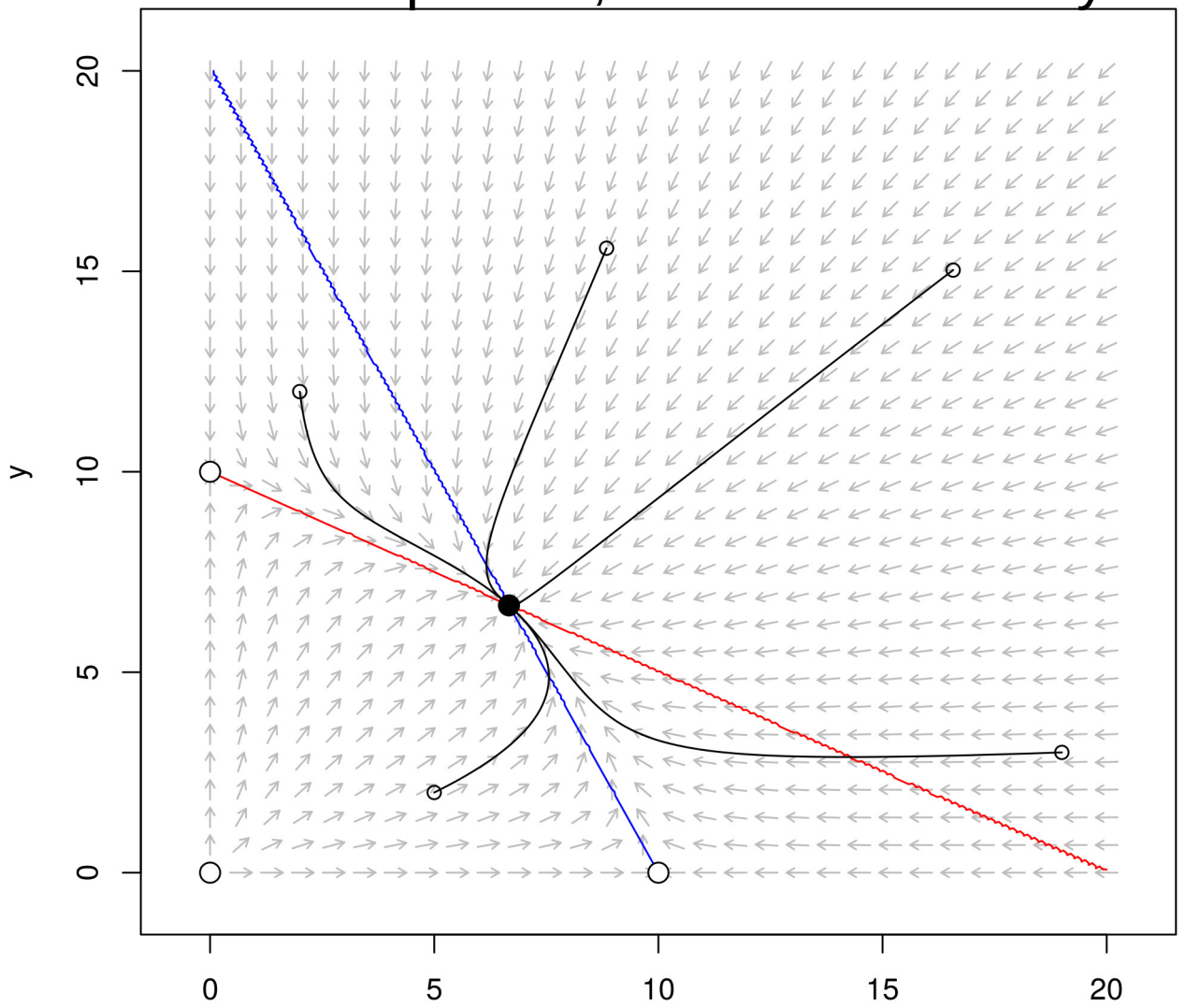
traj <- trajectory(competition.model,y0=c(xstart,ystart),

parameters = model.state.coexist , t.end = 250)

Lotka-Volterra Competition: Coexistence

1) Analyze the deterministic model

Find equilibria, determine stability



xstart = runif(1,0,20); ystart = runif(1,0,20)

traj <- trajectory(competition.model,y0=c(xstart,ystart),

parameters = model.state.coexist , t.end = 250)

Lotka-Volterra Competition: Coexistence

Coexistence $K_x < \frac{K_y}{\alpha_{yx}}$ $K_y < \frac{K_x}{\alpha_{xy}}$

2) Simulate the equations, include stochasticity

Lotka-Volterra Competition: Coexistence

Coexistence $K_x < \frac{K_y}{\alpha_{yx}}$ $K_y < \frac{K_x}{\alpha_{xy}}$

2) Simulate the equations, include stochasticity

```
dxstring <- 'rx*x*(kx - x - alphaxy*y)*(1/kx)'     # L-V competition
dystring <- 'ry*y*(ky - y - alphayx*x)*(1/ky)'     # from earlier

# Parameters included in step one, but in case you are skipping around
model.state.coexist <- c(rx = 4, ry = 4, kx = 10, ky = 10, alphaxy = 0.5, alphayx = 0.5)

dx <- Model2String(model = dxstring, parms = model.state.coexist)
dy <- Model2String(model = dystring, parms = model.state.coexist)
```

Lotka-Volterra Competition: Coexistence

Coexistence $K_x < \frac{K_y}{\alpha_{yx}}$ $K_y < \frac{K_x}{\alpha_{xy}}$

2) Simulate the equations, include stochasticity

```
dxstring <- 'rx*x*(kx - x - alphaxy*y)*(1/kx)'     # L-V competition
dystring <- 'ry*y*(ky - y - alphayx*x)*(1/ky)'     # from earlier

# Parameters included in step one, but in case you are skipping around
model.state.coexist <- c(rx = 4, ry = 4, kx = 10, ky = 10, alphaxy = 0.5, alphayx = 0.5)

dx <- Model2String(model = dxstring, parms = model.state.coexist)
dy <- Model2String(model = dystring, parms = model.state.coexist)

# We need to define some values for our simulation
init.cond <- c(x = 3, y = 10)# initial state of simulation
sigma = 0.1          # level of stochasticity, sigma = 0 is deterministic model
timesteps = 1000                    # number of time steps
deltat = 0.1                        # size of time step

# Run a stochastic simulation, but TSTraj does not return a plot
timesimulation <- TSTraj(y0 = init.cond, time = timesteps,
                         deltat = deltat, x.rhs = dx, y.rhs = dy, sigma = sigma)
```

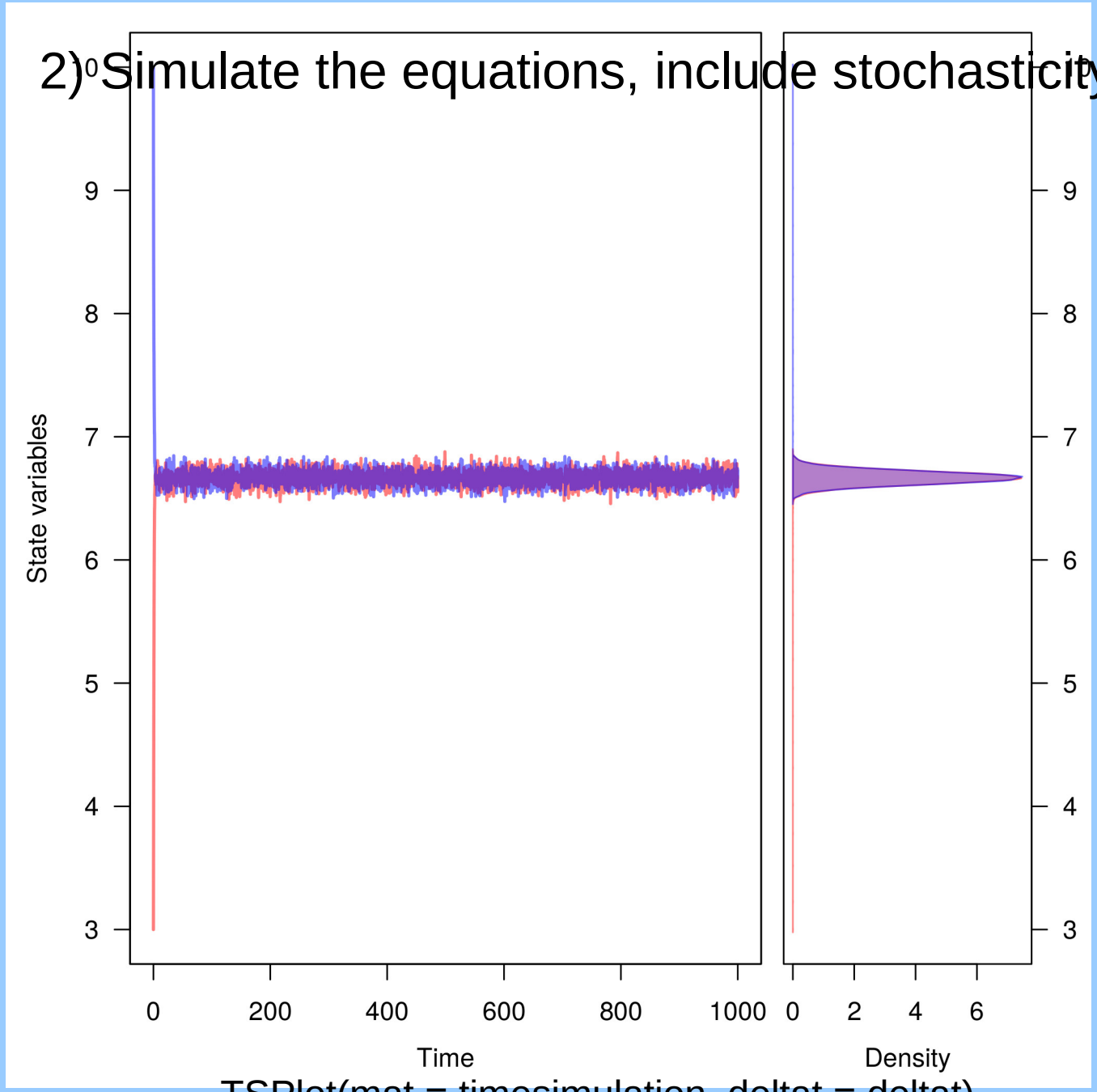
Lotka-Volterra Competition: Coexistence

2) Simulate the equations, include stochasticity

TSPlot(mat = timesimulation, deltat = deltat)

Lotka-Volterra Competition: Coexistence

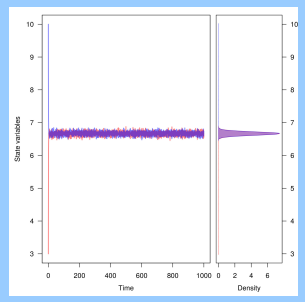
2) Simulate the equations, include stochasticity



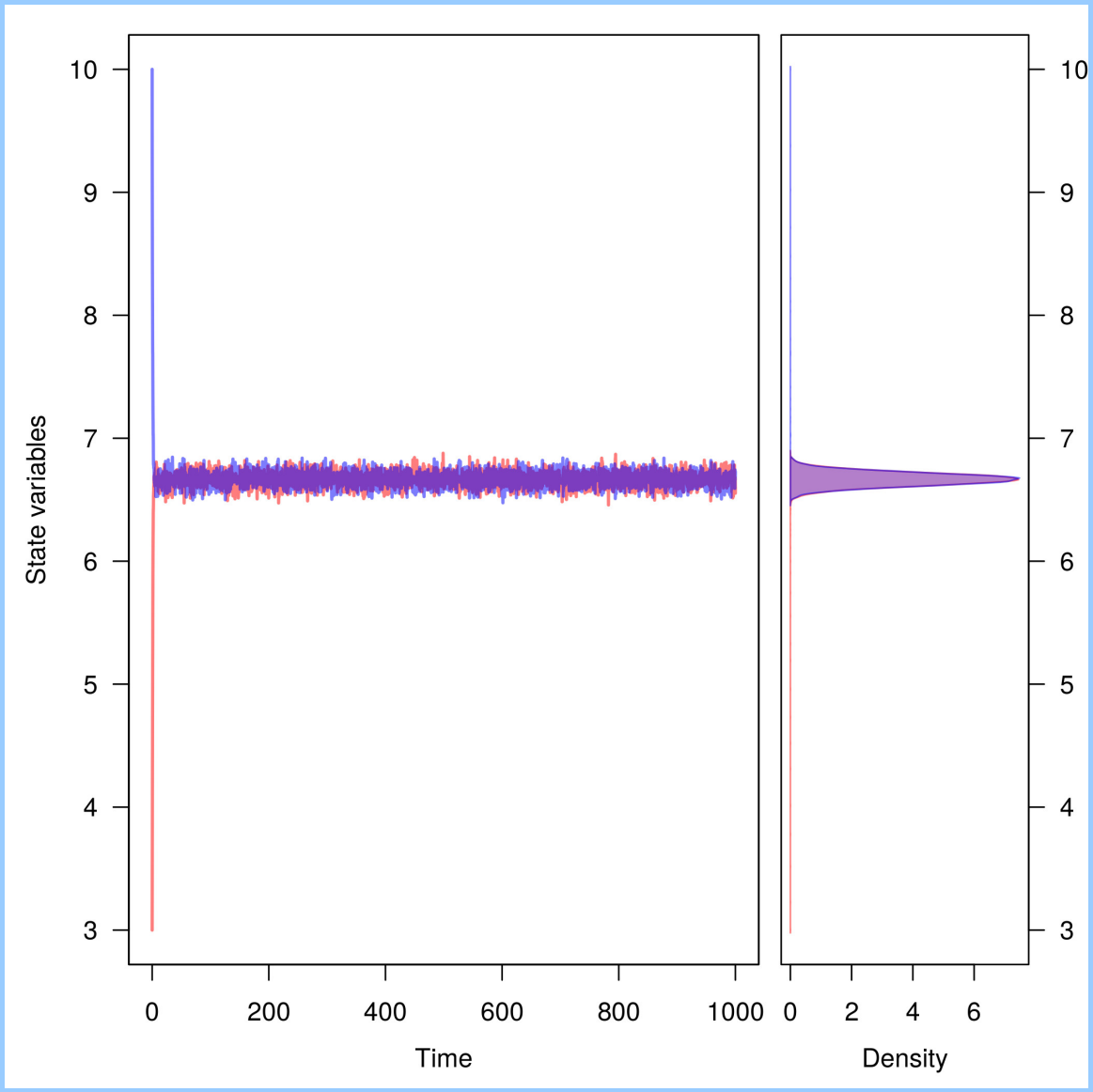
TSPlot(mat = timesimulation, deltat = deltat)

Lotka-Volterra Competition: Coexistence

2) Simulate the equations, include stochasticity

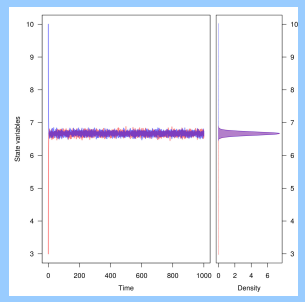


TSPlot(timesimulation, deltat = deltat)

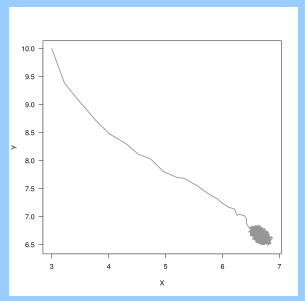


Lotka-Volterra Competition: Coexistence

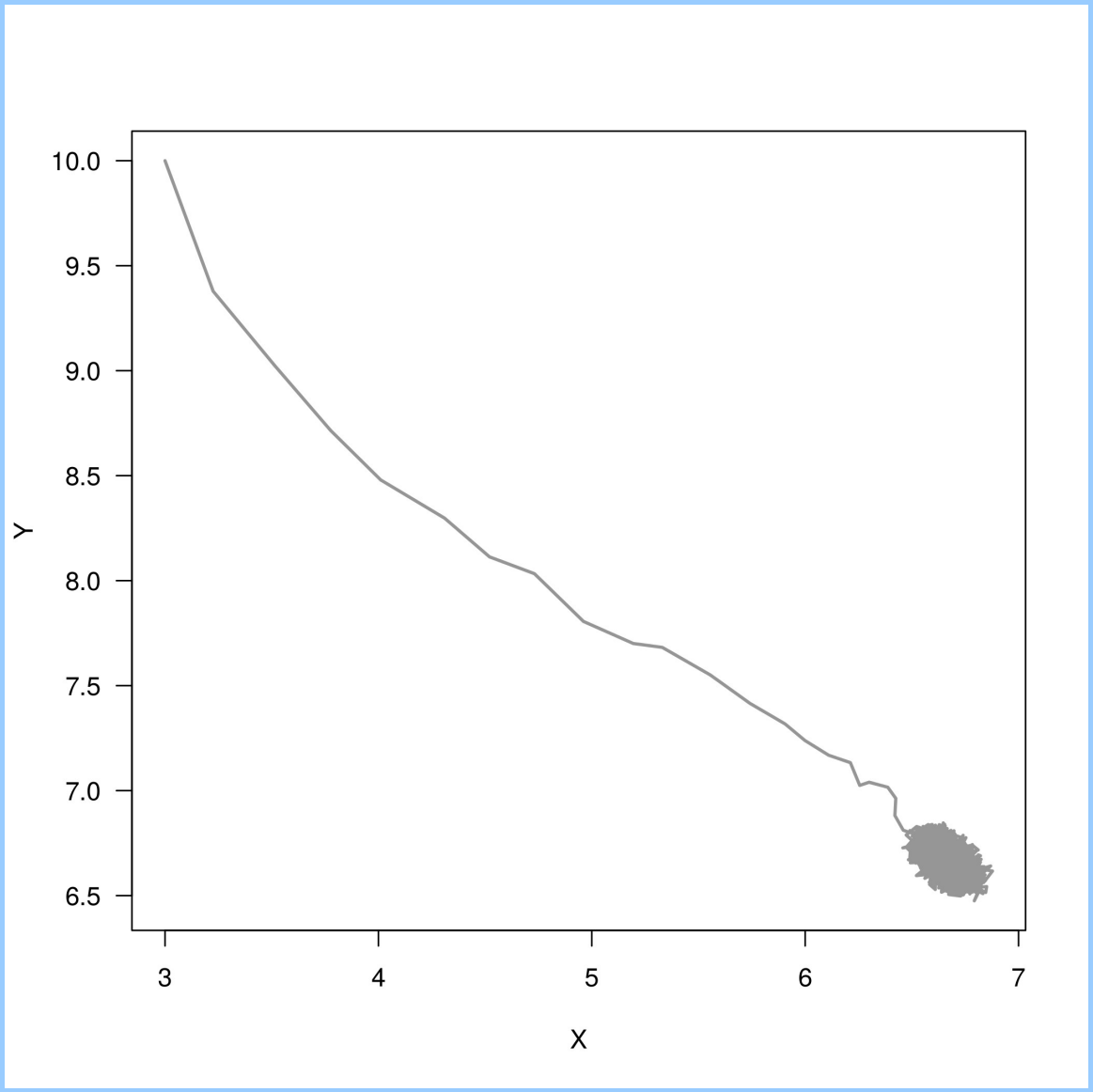
2) Simulate the equations, include stochasticity



TSPlot(timesimulation,
deltat = deltat)

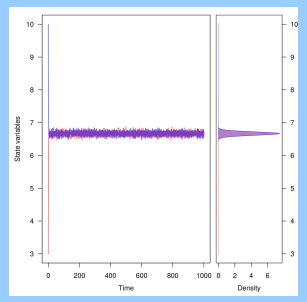


TSPlot(timesimulation,
deltat = model.deltat,
dim = 2)

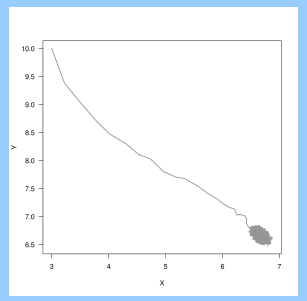


Lotka-Volterra Competition: Coexistence

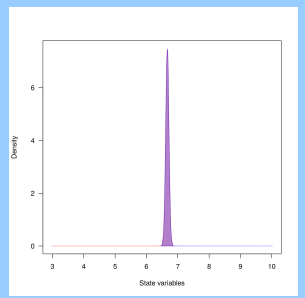
2) Simulate the equations, include stochasticity



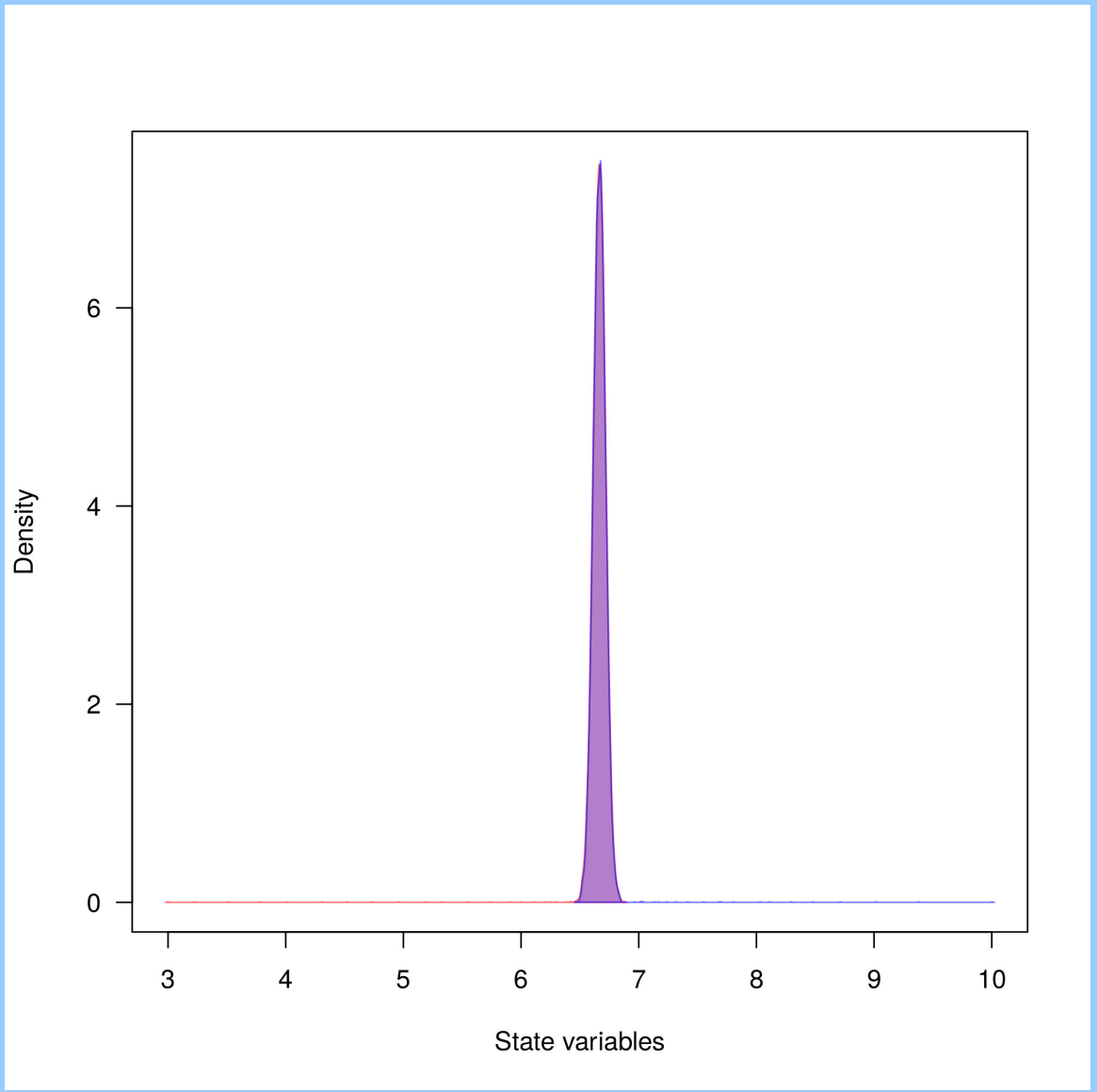
TSPlot(timesimulation, deltat = deltat)



TSPlot(timesimulation, deltat = model.deltat, dim = 2)

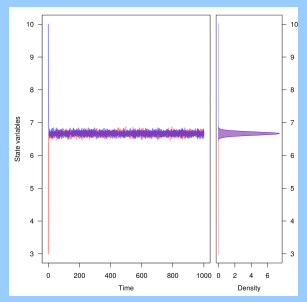


TSDensity(timesimulation, dim = 1)

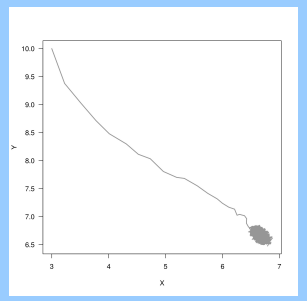


Lotka-Volterra Competition: Coexistence

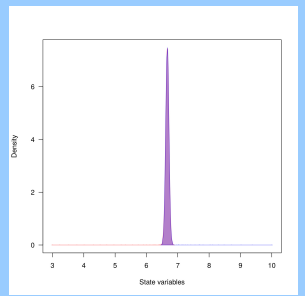
2) Simulate the equations, include stochasticity



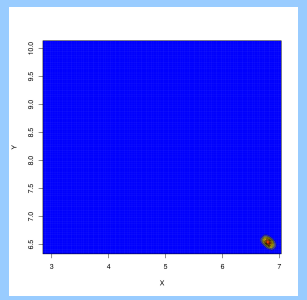
TSPlot(timesimulation, deltat = deltat)



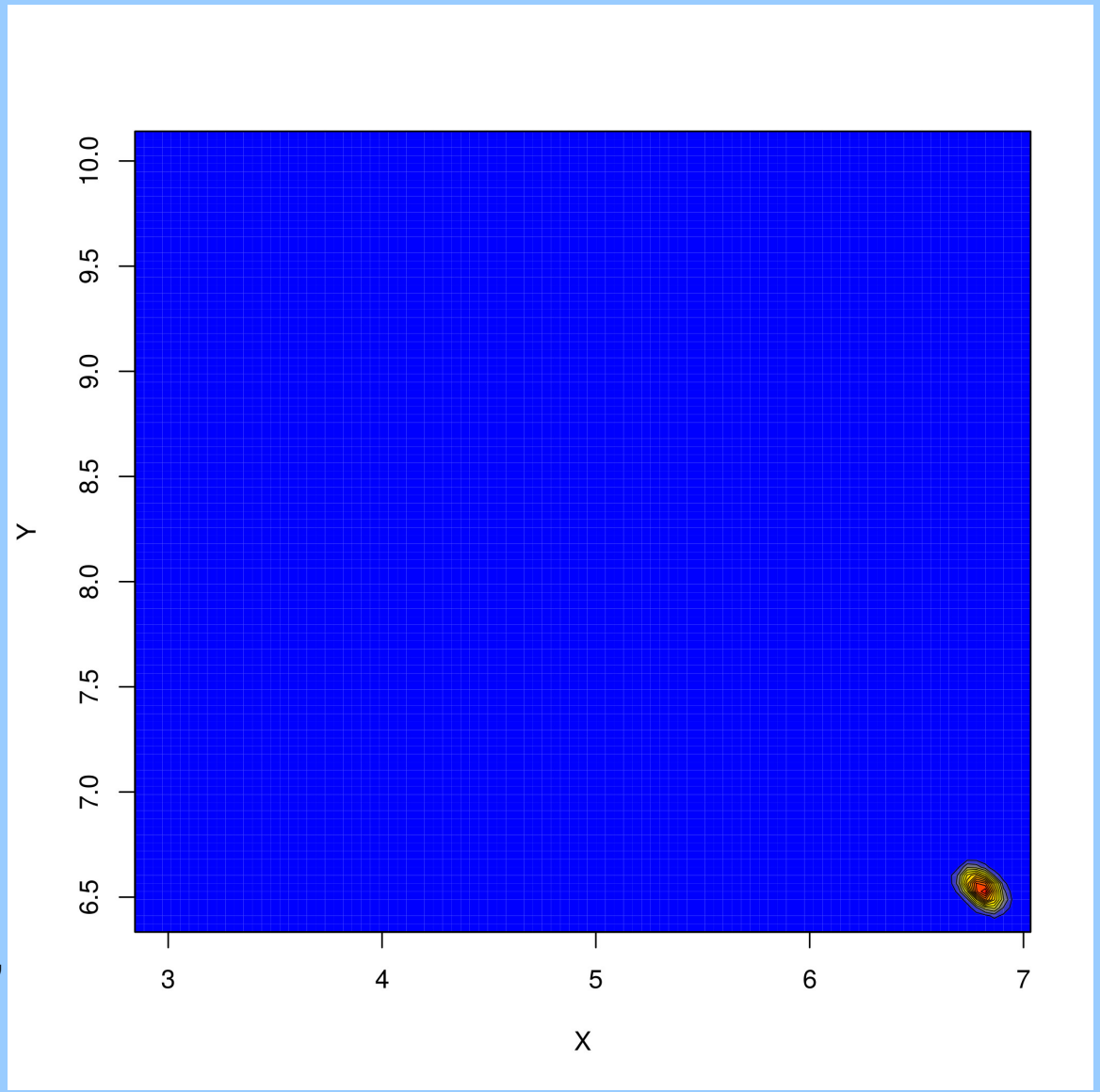
TSPlot(timesimulation, deltat = model.deltat, dim = 2)



TSDensity(timesimulation, dim = 1)



TSDensity(timesimulation, dim = 2)



Lotka-Volterra Competition: Coexistence

3) Calculate the local quasi-potentials

```
bounds.x = c(-5,15)     # upper and lower limit of x
bounds.y = c(-5,15)     # upper and lower limit of y

step.number.x = 2000    # number of divisions in bounds.x
step.number.y = 2000    # number of divisions in bounds.y

xinit = 6.667           # equilibrium x value
yinit = 6.667           # equilibrium y value

# Compute the quasi-potential using the ordered upwind method
coexist <- QPotential(x.rhs = dx, x.start = xinit, x.bound = bounds.x,
                      x.num.steps = step.number.x, y.rhs = dy, y.start = yinit,
                      y.bound = bounds.y, y.num.steps = step.number.y)
```

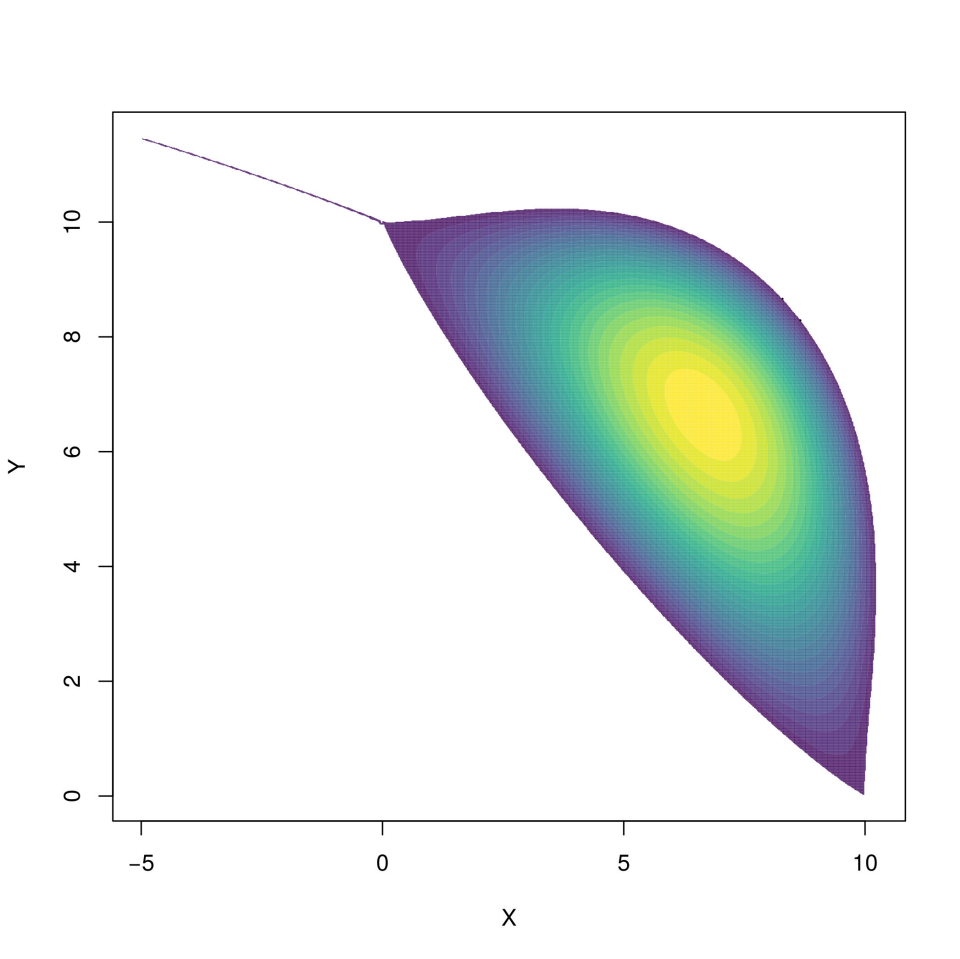
Lotka-Volterra Competition: Coexistence

3) Calculate the local quasi-potentials

```
QPContour(surface = coexist, dens = c(1000, 1000), x.bound = bounds.x,
    y.bound = bounds.y, c.parm = 0)
```

Lotka-Volterra Competition: Coexistence

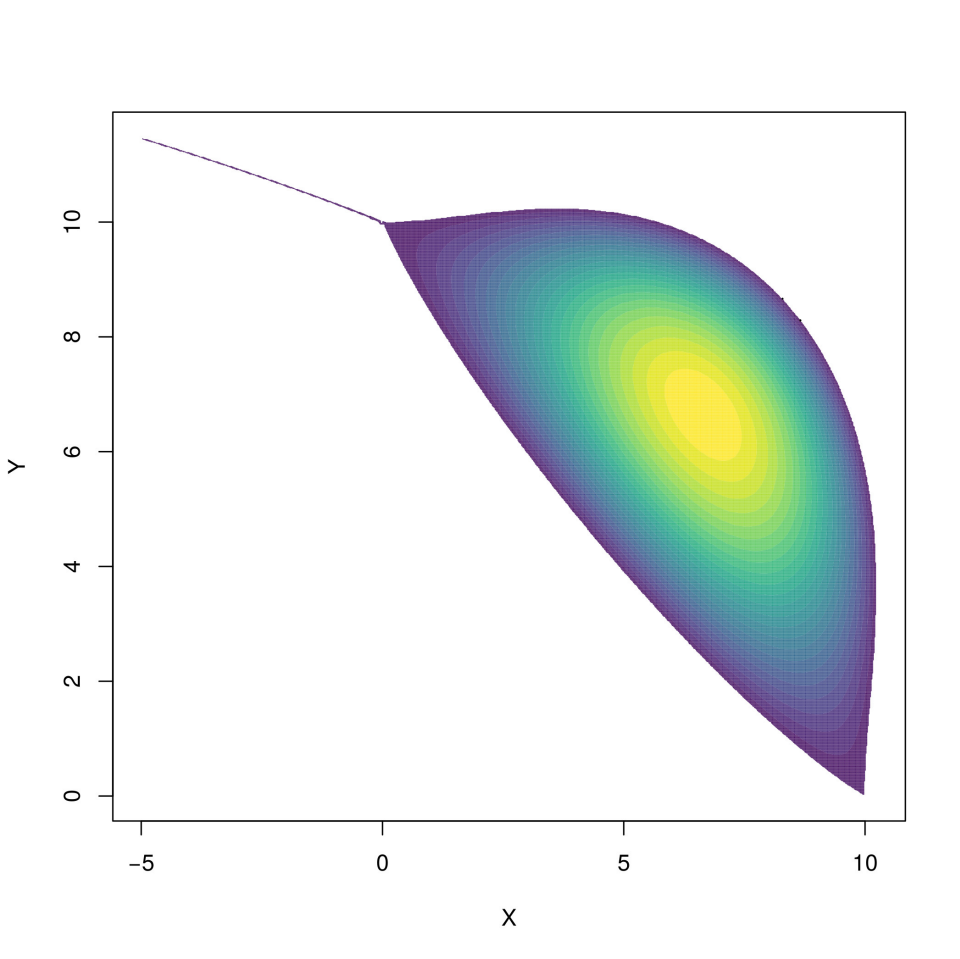
3) Calculate the local quasi-potentials



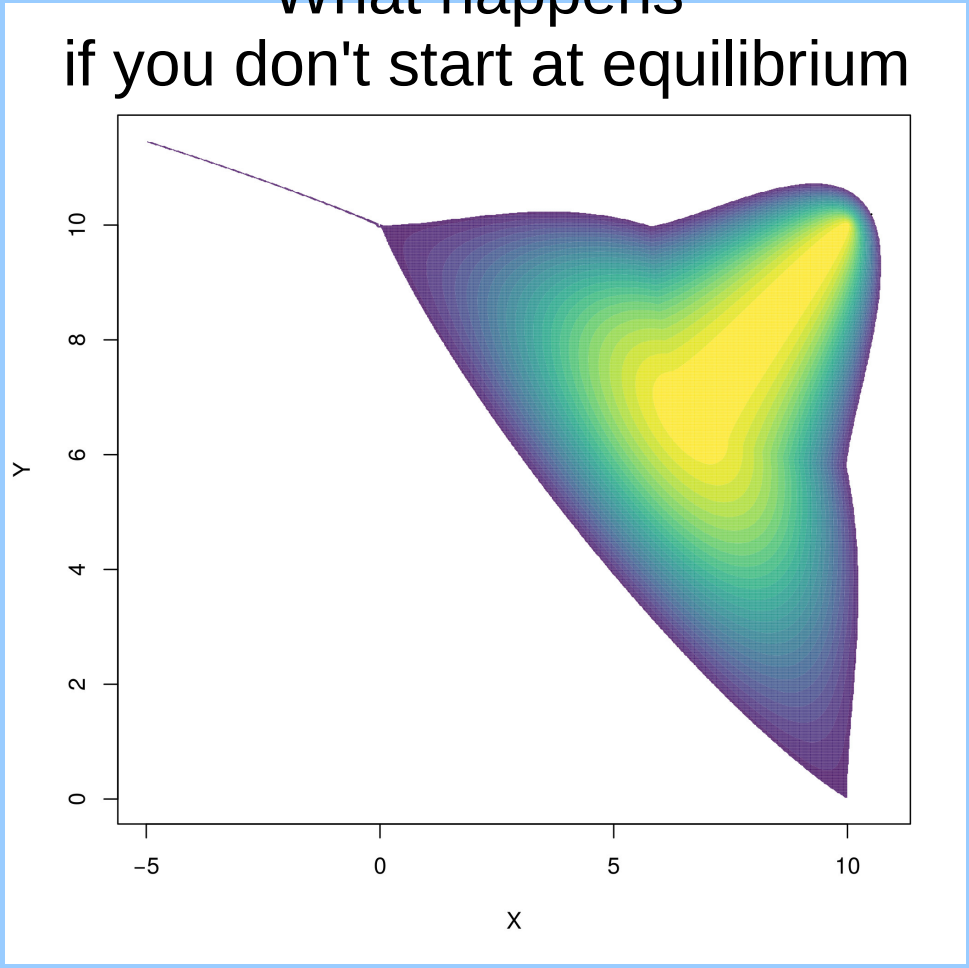

```
QPContour(surface = coexist, dens = c(1000, 1000), x.bound = bounds.x,
      y.bound = bounds.y, c.parm = 0)
```

Lotka-Volterra Competition: Coexistence

3) Calculate the local quasi-potentials



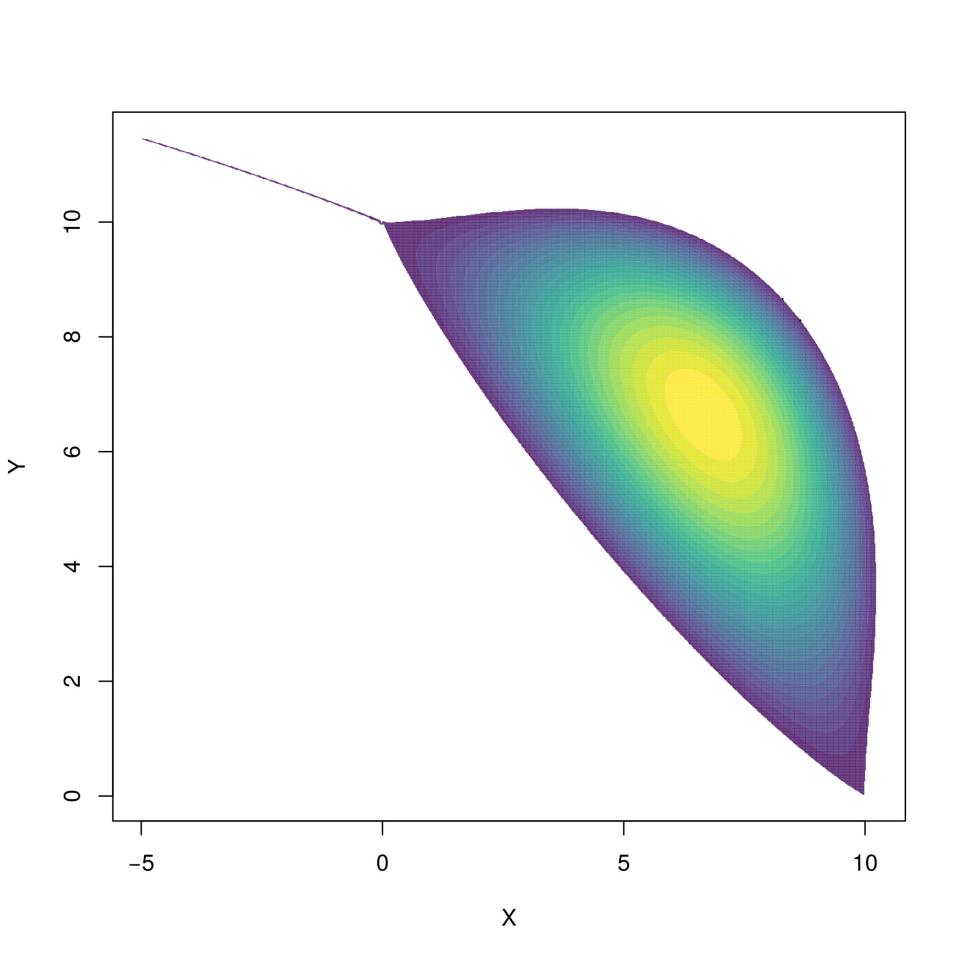
What happens
if you don't start at equilibrium

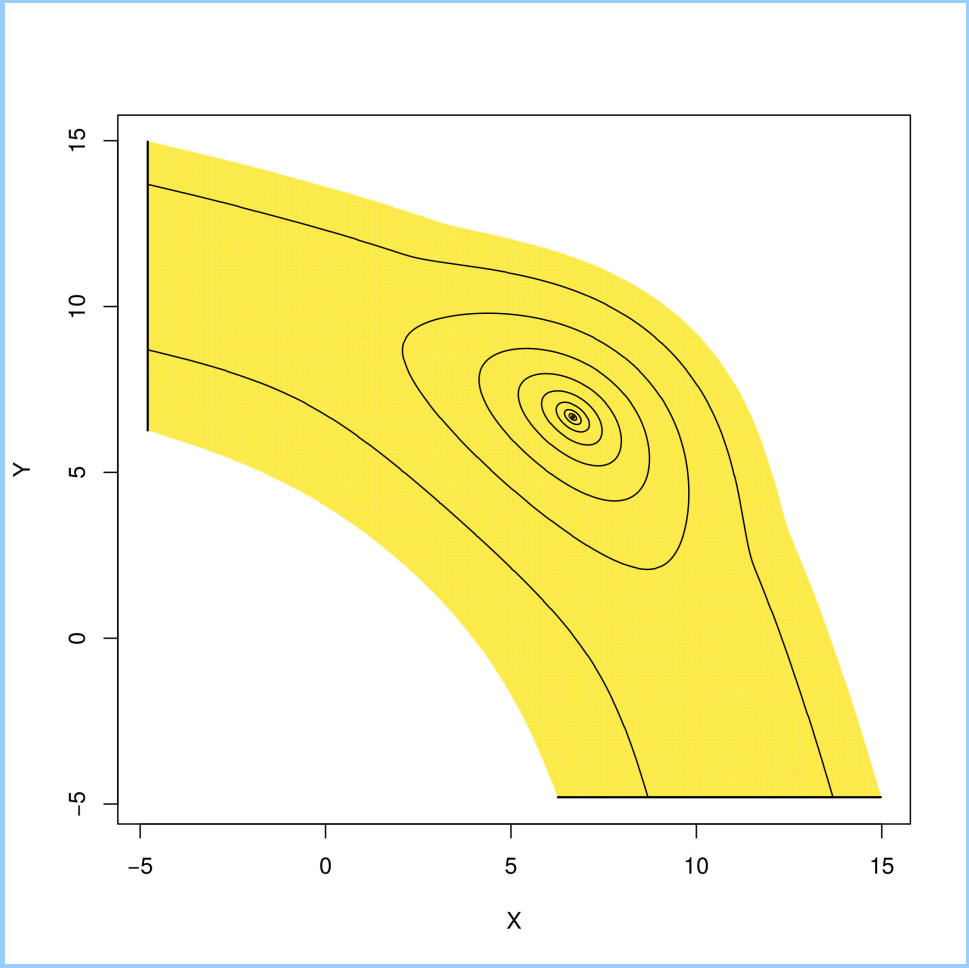



```
QPContour(surface = coexist, dens = c(1000, 1000), x.bound = bounds.x,
      y.bound = bounds.y, c.parm = 0)
```

Lotka-Volterra Competition: Coexistence

3) Calculate the local quasi-potentials

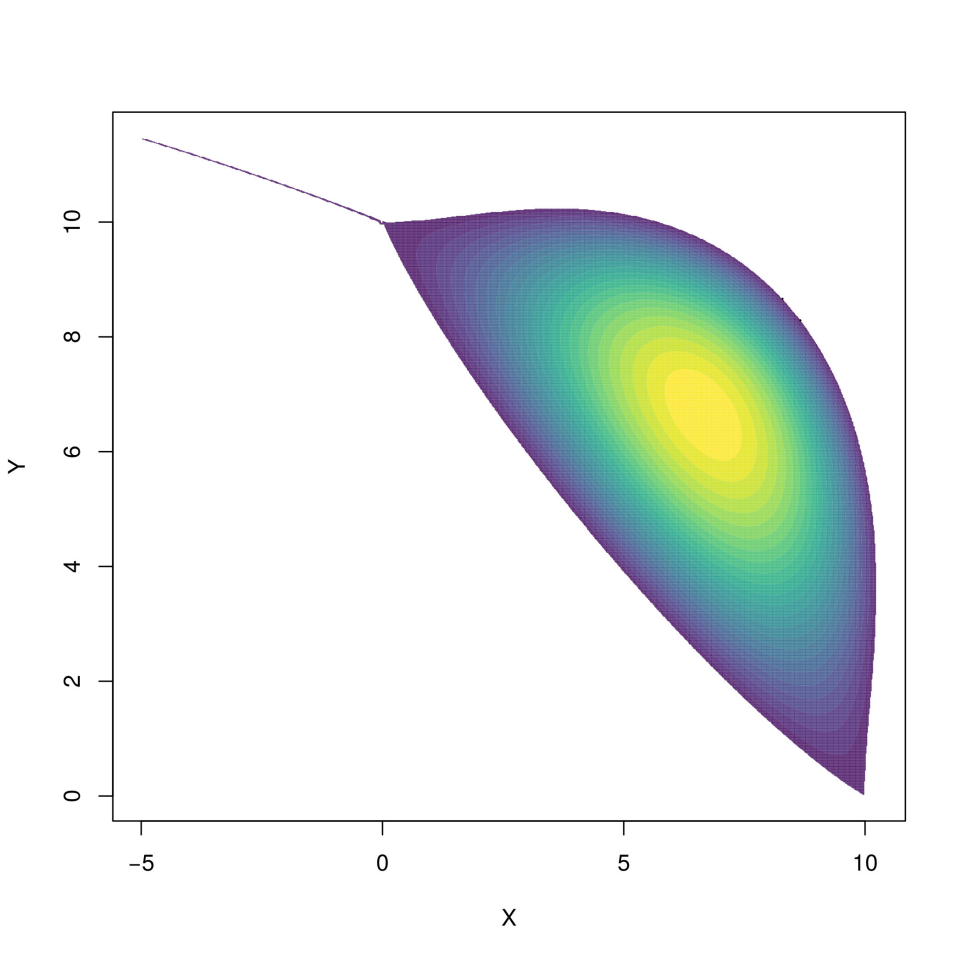


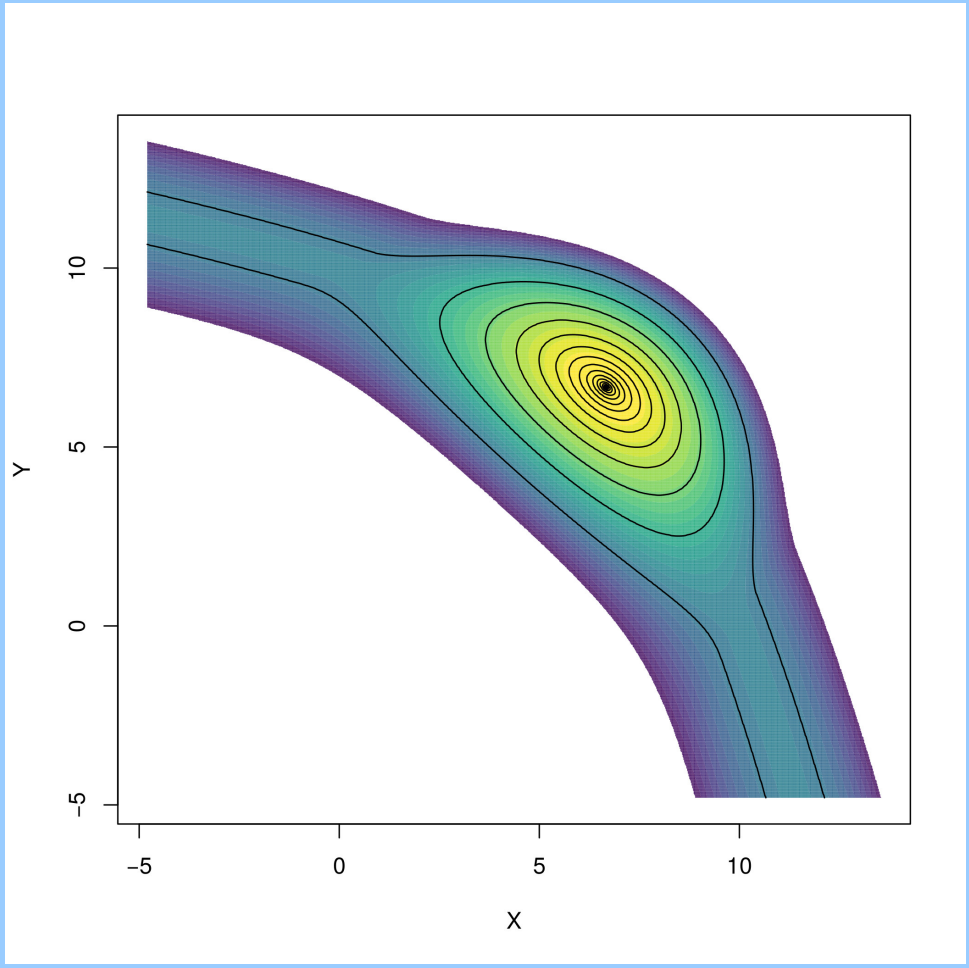



```
QPContour(surface = coexist.BOUNCE, dens = c(1000, 1000), x.bound = bounds.x,
    y.bound = bounds.y, c.parm = 10)
```

Lotka-Volterra Competition: Coexistence

3) Calculate the local quasi-potentials

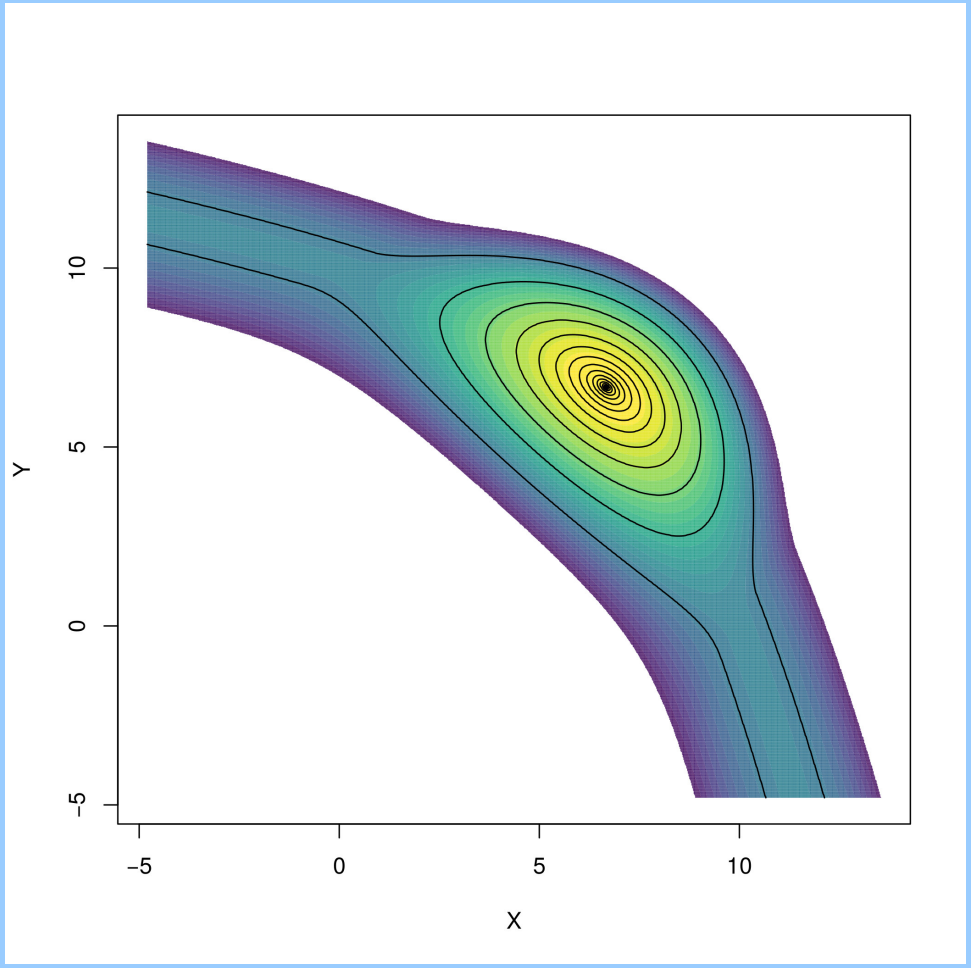





```
QPContour(surface = coexist.BOUNCE.REMOVED, dens = c(1000, 1000), x.bound = bounds.x,
      y.bound = bounds.y, c.parm = 10)
```

Lotka-Volterra Competition: Coexistence

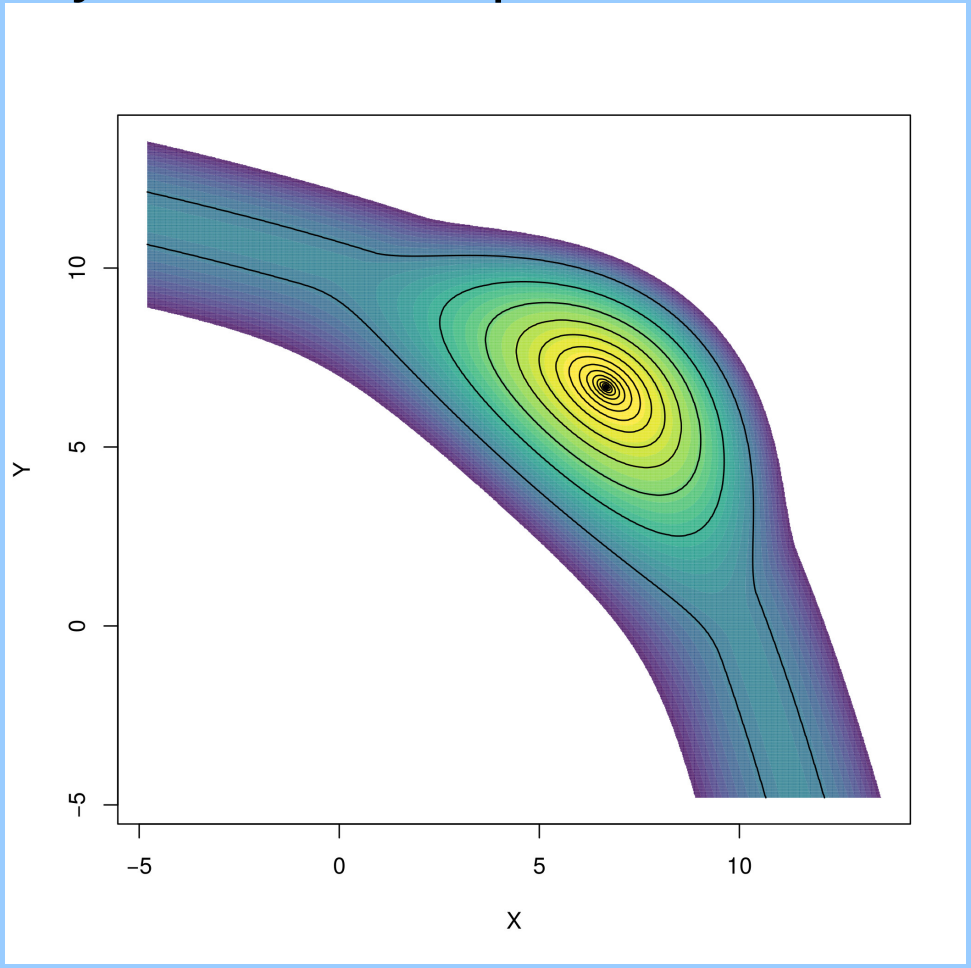
3) Calculate the local quasi-potentials



QPContour(surface = coexist.BOUNCE.REMOVED, dens = c(1000, 1000), x.bound = bounds.x, y.bound = bounds.y, c.parm = 10)

Lotka-Volterra Competition: Coexistence

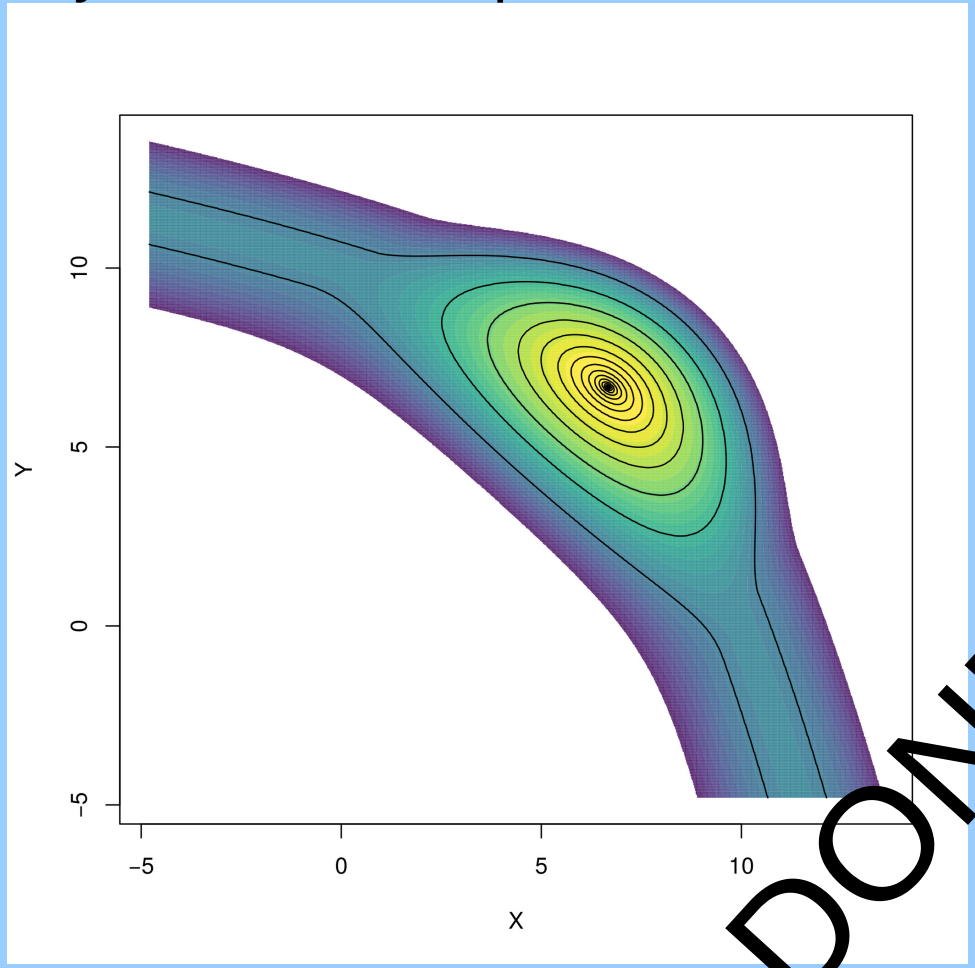
4) Calculate the global quasi-potential
(if only one stable equilibrium, then local = global)



QPContour(surface = coexist.BOUNCE.REMOVED, dens = c(1000, 1000), x.bound = bounds.x, y.bound = bounds.y, c.parm = 10)

Lotka-Volterra Competition: Coexistence

4) Calculate the global quasi-potential
(if only one stable equilibrium, then local = global)



DONE!

In later examples,
we will have to combine
local quasi-potentials
Into a global
quasi-potential

QPContour(surface = coexist.BOUNCE.REMOVED, dens = c(1000, 1000), x.bound = bounds.x,
y.bound = bounds.y, c.parm = 10)

Lotka-Volterra Competition: Coexistence

5) Visualize the global quasi-potential

```
require(plot3D)
subsample.x = 200 # 3D plotting all the points takes forever
subsample.y = 200 # so we subsample and only plot 200*200 points

my.theta = 110                    # rotation on 3D graph
my.phi = 0                        # altitude of 3D graph
my.zlim = c(-0.001, 25)      # range of z axis on plot, by making many
                                  # graphs, we decided 25 looks good
                                  # also notice that 25 removed the funk
my.zlab = intToUtf8(0x03A6) # this allows us to plot the letter phi
```

Lotka-Volterra Competition: Coexistence

5) Visualize the global quasi-potential

```
require(plot3D)
subsample.x = 200 # 3D plotting all the points takes forever
subsample.y = 200 # so we subsample and only plot 200*200 points

my.theta = 110                    # rotation on 3D graph
my.phi = 0                        # altitude of 3D graph
my.zlim = c(-0.001, 25)      # range of z axis on plot, by making many
                                  # graphs, we decided 25 looks good
                                  # also notice that 25 removed the funk
my.zlab = intToUtf8(0x03A6) # this allows us to plot the letter phi


# again, need to subsample from the bigger matrix to make graphing faster
coexist.BOUNCE.SUBSAMPLE <- coexist.BOUNCE.REMOVED[
                                  seq(1,step.number.x,step.number.x/subsample.x),
                                  seq(1,step.number.y,step.number.y/subsample.y) ]
```

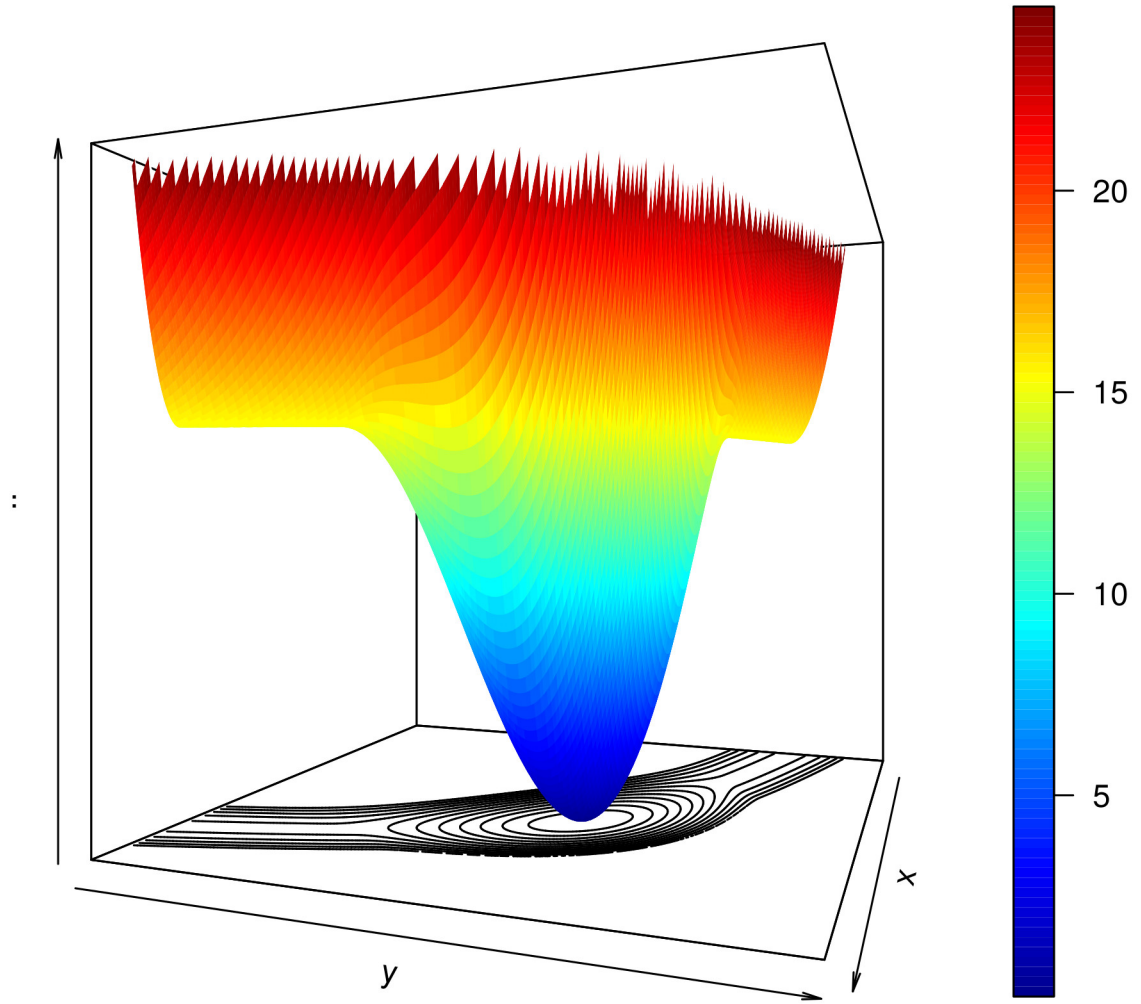
Lotka-Volterra Competition: Coexistence

5) Visualize the global quasi-potential

```
require(plot3D)
subsample.x = 200 # 3D plotting all the points takes forever
subsample.y = 200 # so we subsample and only plot 200*200 points

my.theta = 110                    # rotation on 3D graph
my.phi = 0                        # altitude of 3D graph
my.zlim = c(-0.001, 25)      # range of z axis on plot, by making many
                                  # graphs, we decided 25 looks good
                                  # also notice that 25 removed the funk
my.zlab = intToUtf8(0x03A6) # this allows us to plot the letter phi


# again, need to subsample from the bigger matrix to make graphing faster
coexist.BOUNCE.SUBSAMPLE <- coexist.BOUNCE.REMOVED[
                                  seq(1,step.number.x,step.number.x/subsample.x),
                                  seq(1,step.number.y,step.number.y/subsample.y) ]

# plot the quasipotential when we used bounce = 'b'
persp3D(z = coexist.BOUNCE.SUBSAMPLE, theta = my.theta, phi = my.phi,
          contour = T, zlim = my.zlim, zlab = my.zlab)
```

Lotka-Volterra Competition: Coexistence

5) Visualize the global quasi-potential



persp3D(z = coexist.BOUNCE.SUBSAMPLE, theta = my.theta, phi = my.phi, contour = T, zlim = my.zlim, zlab = my.zlab)

Lotka-Volterra Competition: Coexistence

6) Visualize the vector field decompositions

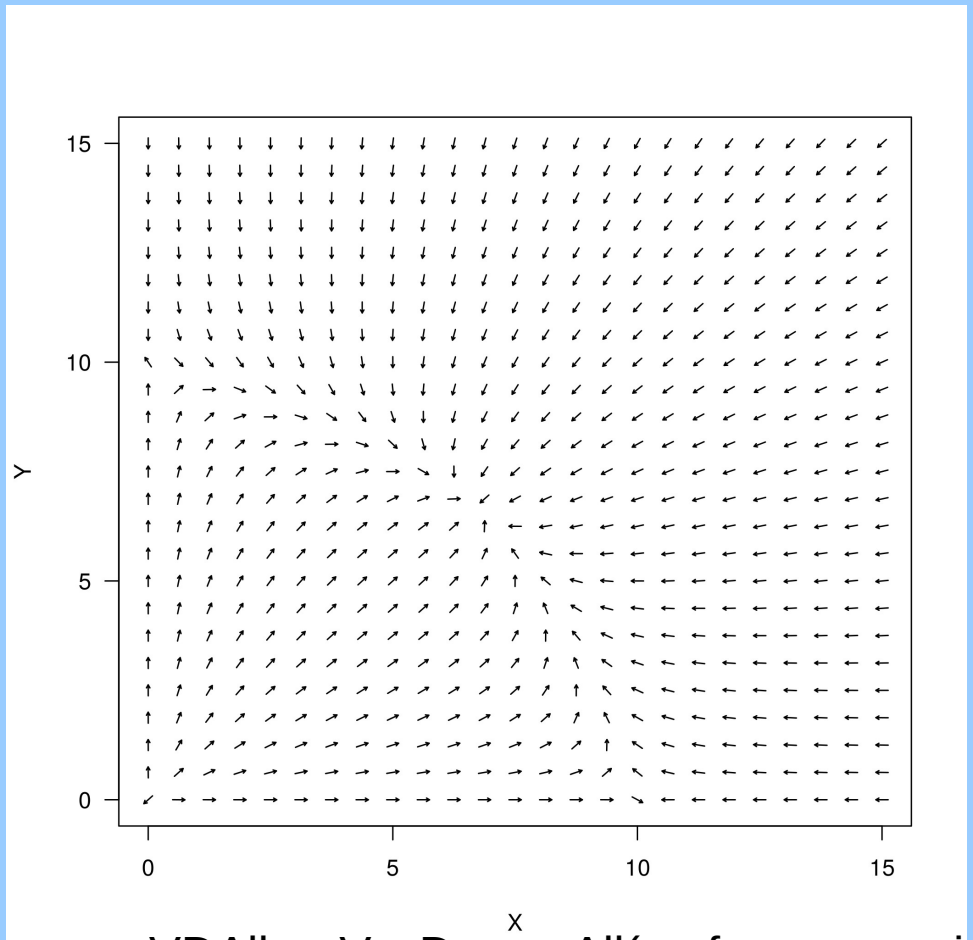
a) deterministic skeleton

```
VDAll <- VecDecomAll(surface = coexist.BOUNCE, x.rhs = dx,
                     y.rhs = dy, x.bound = bounds.x, y.bound = bounds.y)
x.skel = VDAll[,,1] ; y.skel = VDAll[,,2]
VecDecomPlot(x.field = x.skel, y.field = y.skel, dens = c(25, 25), x.bound = bounds.x,
      y.bound = bounds.y,  arrow.type = "equal", xlim = c(0, 15), ylim = c(0, 15),
      tail.length = 0.25, head.length = 0.025)
```

Lotka-Volterra Competition: Coexistence

6) Visualize the vector field decompositions

a) deterministic skeleton

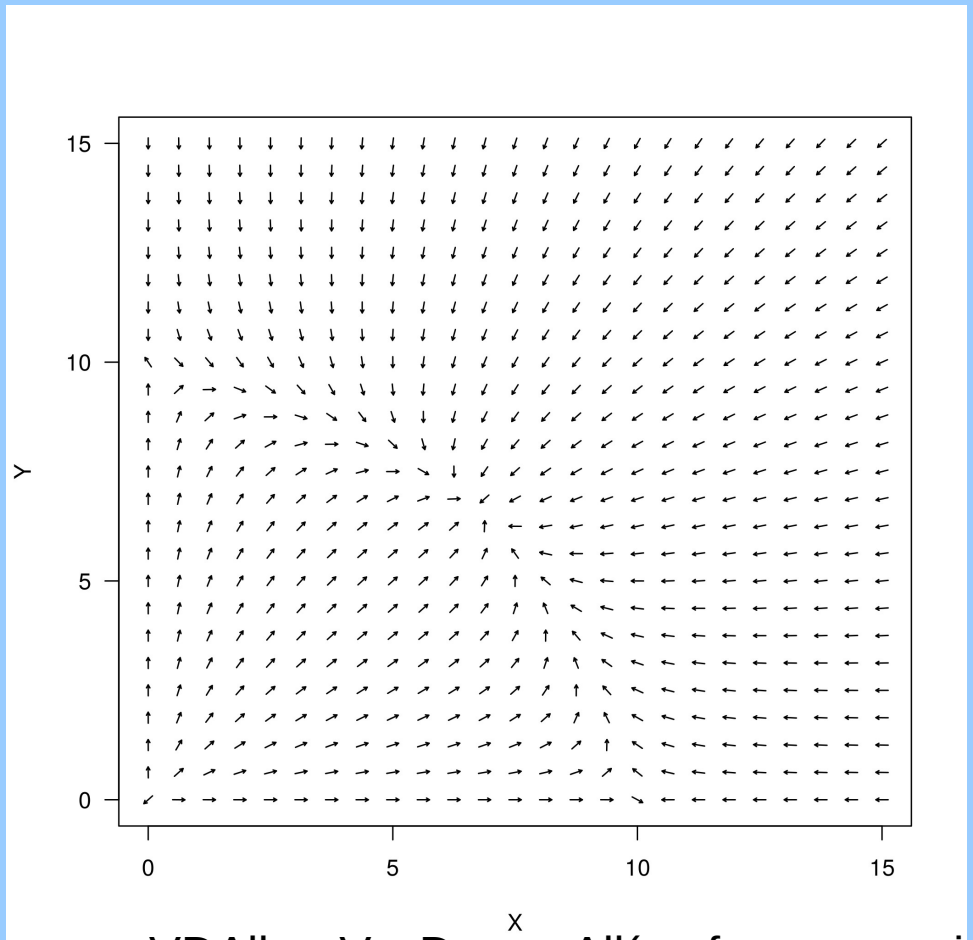



```
VDAll <- VecDecomAll(surface = coexist.BOUNCE, x.rhs = dx,
                     y.rhs = dy, x.bound = bounds.x, y.bound = bounds.y)
x.skel = VDAll[,,1] ; y.skel = VDAll[,,2]
VecDecomPlot(x.field = x.skel, y.field = y.skel, dens = c(25, 25), x.bound = bounds.x,
      y.bound = bounds.y,  arrow.type = "equal", xlim = c(0, 15), ylim = c(0, 15),
      tail.length = 0.25, head.length = 0.025)
```

Lotka-Volterra Competition: Coexistence

6) Visualize the vector field decompositions

a) deterministic skeleton



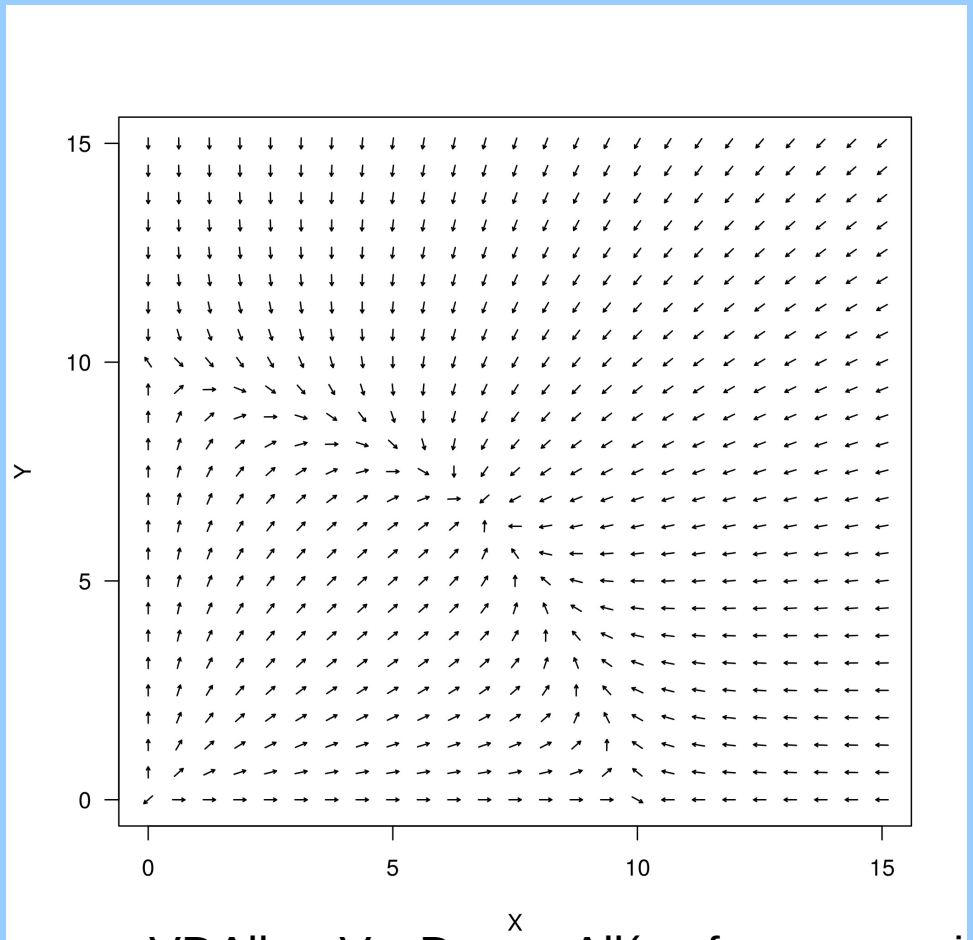
Should look familiar

```
VDAll <- VecDecomAll(surface = coexist.BOUNCE, x.rhs = dx,
                     y.rhs = dy, x.bound = bounds.x, y.bound = bounds.y)
x.skel = VDAll[,,1] ; y.skel = VDAll[,,2]
VecDecomPlot(x.field = x.skel, y.field = y.skel, dens = c(25, 25), x.bound = bounds.x,
      y.bound = bounds.y,  arrow.type = "equal", xlim = c(0, 15), ylim = c(0, 15),
      tail.length = 0.25, head.length = 0.025)
```

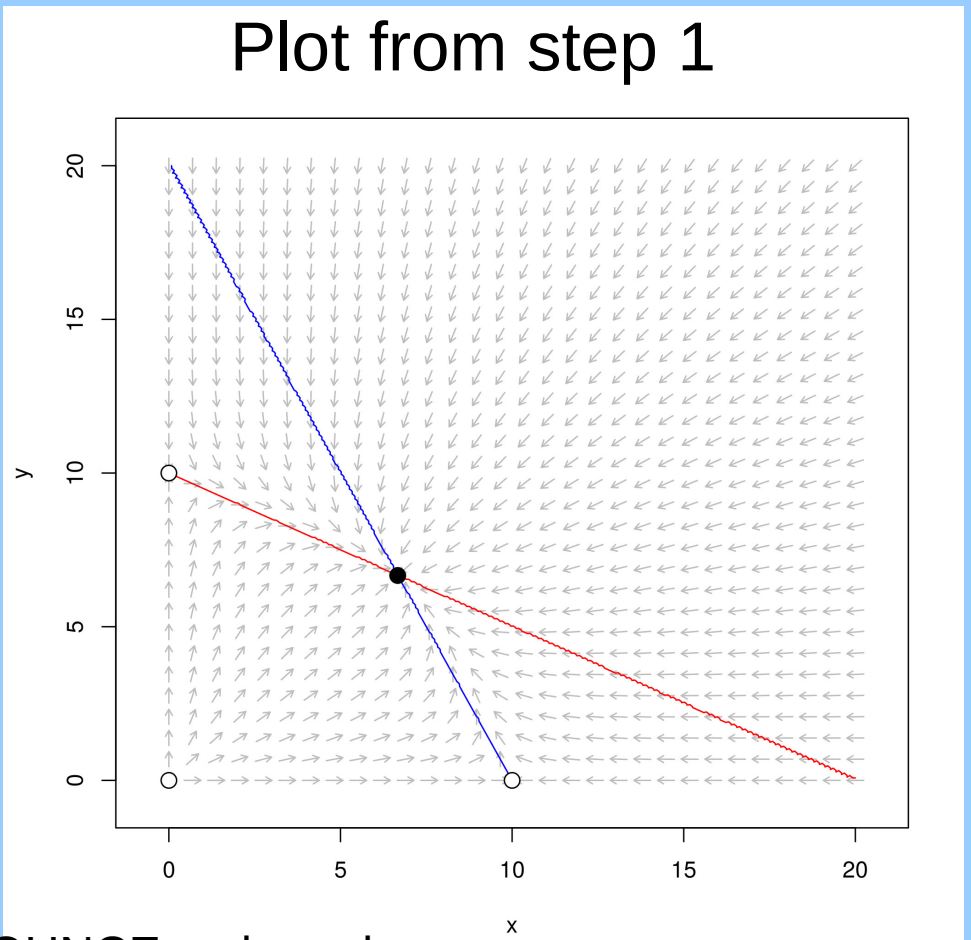
Lotka-Volterra Competition: Coexistence

6) Visualize the vector field decompositions

a) deterministic skeleton



Plot from step 1



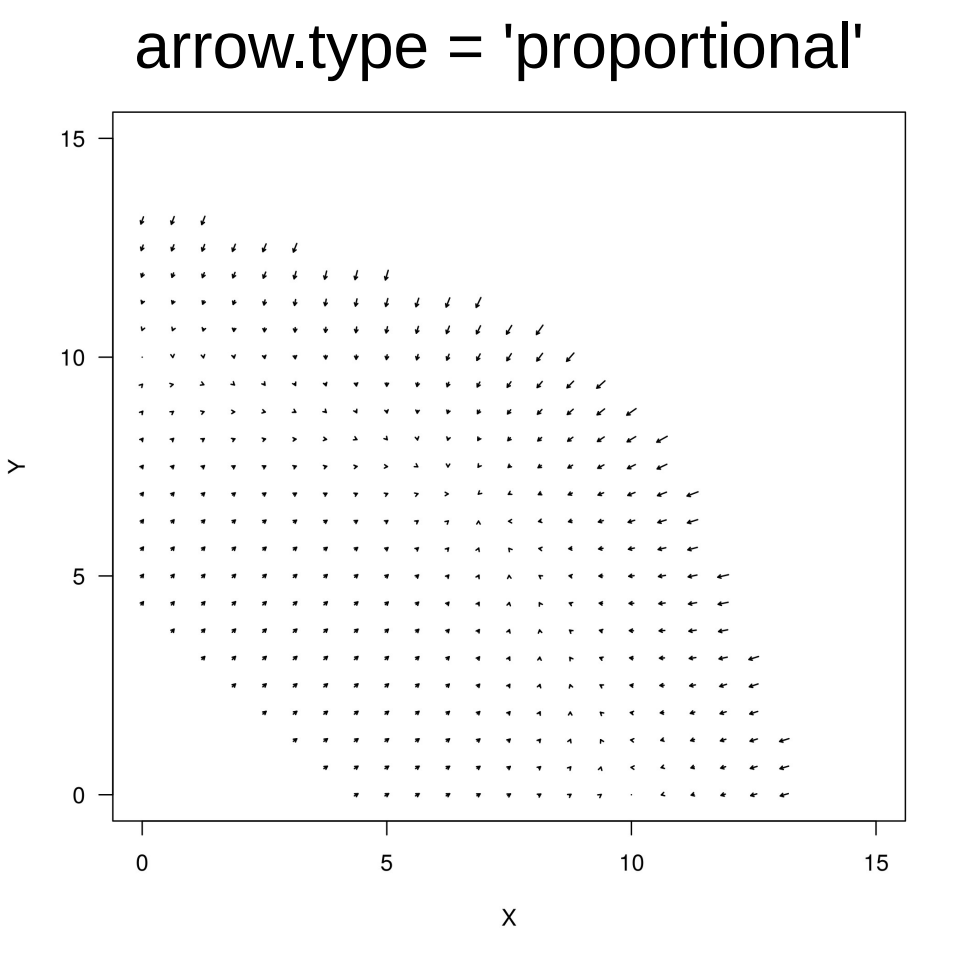

```
VDAll <- VecDecomAll(surface = coexist.BOUNCE, x.rhs = dx,
                     y.rhs = dy, x.bound = bounds.x, y.bound = bounds.y)
x.skel = VDAll[,,1] ; y.skel = VDAll[,,2]
VecDecomPlot(x.field = x.skel, y.field = y.skel, dens = c(25, 25), x.bound = bounds.x,
      y.bound = bounds.y,  arrow.type = "equal", xlim = c(0, 15), ylim = c(0, 15),
      tail.length = 0.25, head.length = 0.025)
```

Lotka-Volterra Competition: Coexistence

6) Visualize the vector field decompositions

b) visualize the gradient

arrow.type = 'proportional'



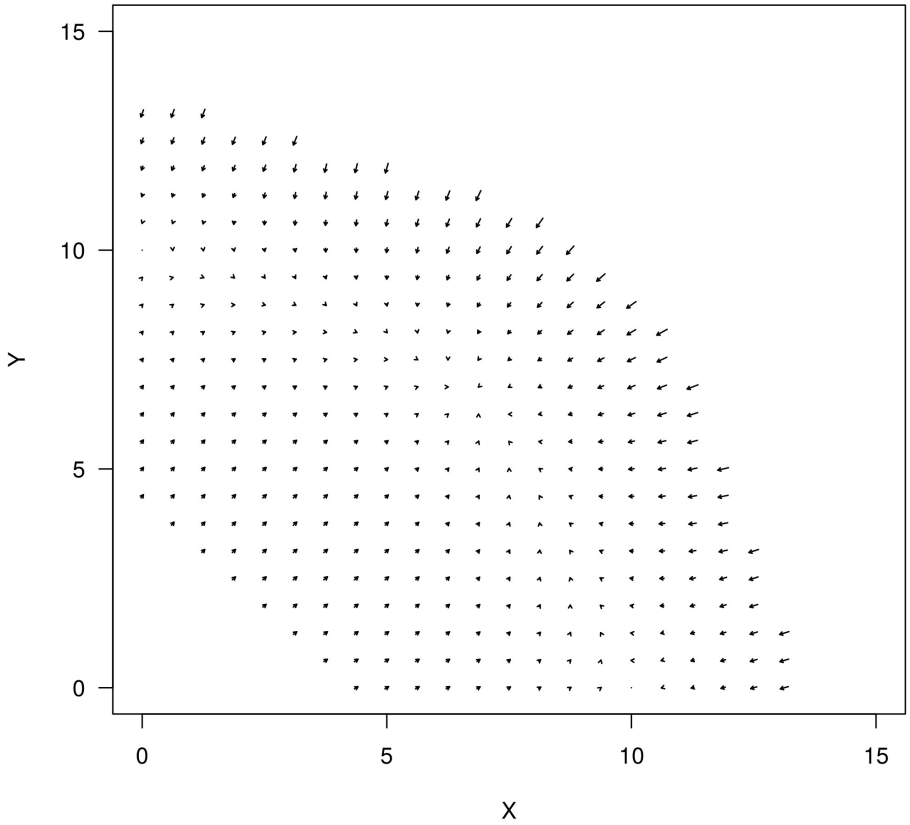

```
VecDecomPlot(x.field = VDAll[,,3], y.field = VDAll[,,4], dens = c(25, 25),
      x.bound = bounds.x, y.bound = bounds.y, arrow.type = "XXXXXXXXX",
      tail.length = 0.25, head.length = 0.025, xlim = c(0,15), ylim = c(0,15))
```

Lotka-Volterra Competition: Coexistence

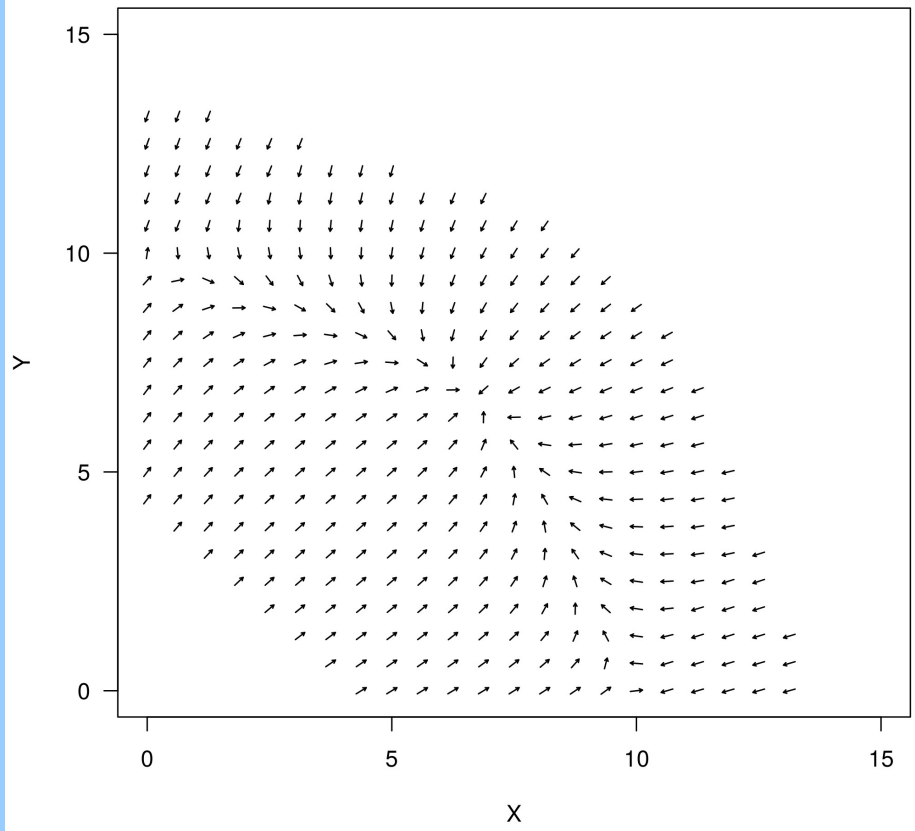
6) Visualize the vector field decompositions

b) visualize the gradient

arrow.type = 'proportional'



arrow.type = 'equal'

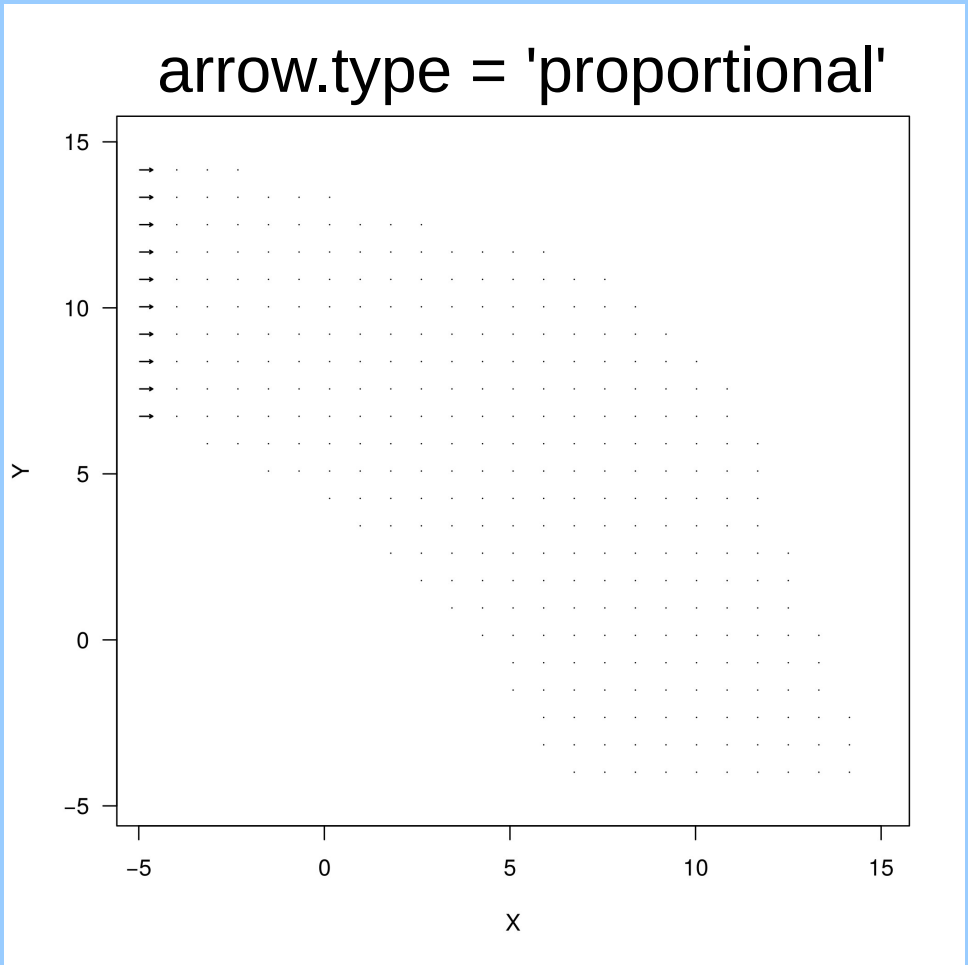



```
VecDecomPlot(x.field = VDAll[,,3], y.field = VDAll[,,4], dens = c(25, 25),
      x.bound = bounds.x, y.bound = bounds.y, arrow.type = "XXXXXXXXX",
      tail.length = 0.25, head.length = 0.025, xlim = c(0,15), ylim = c(0,15))
```

Lotka-Volterra Competition: Coexistence

6) Visualize the vector field decompositions

b) visualize the remainder



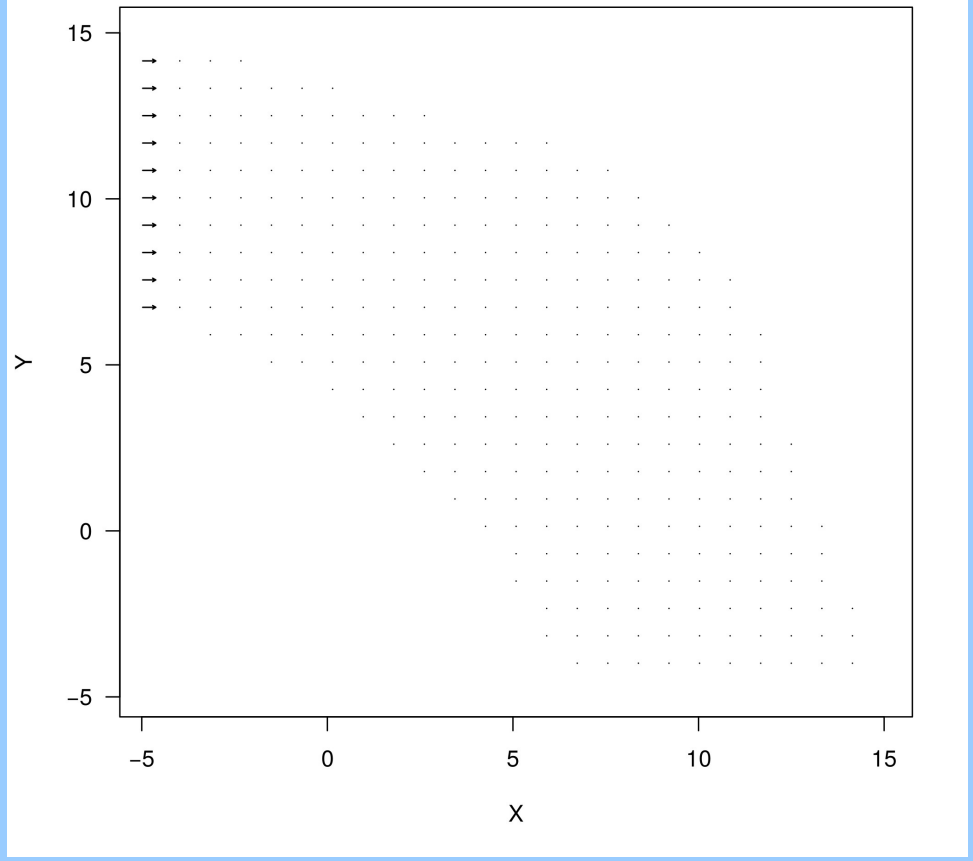

```
VecDecomPlot(x.field = VDAll[,,5], y.field = VDAll[,,6], dens = c(25, 25), x.bound = bounds.x,
                                y.bound = bounds.y, arrow.type = "XXXXXXXX",
                                tail.length = 0.35, head.length = 0.025)
```

Lotka-Volterra Competition: Coexistence

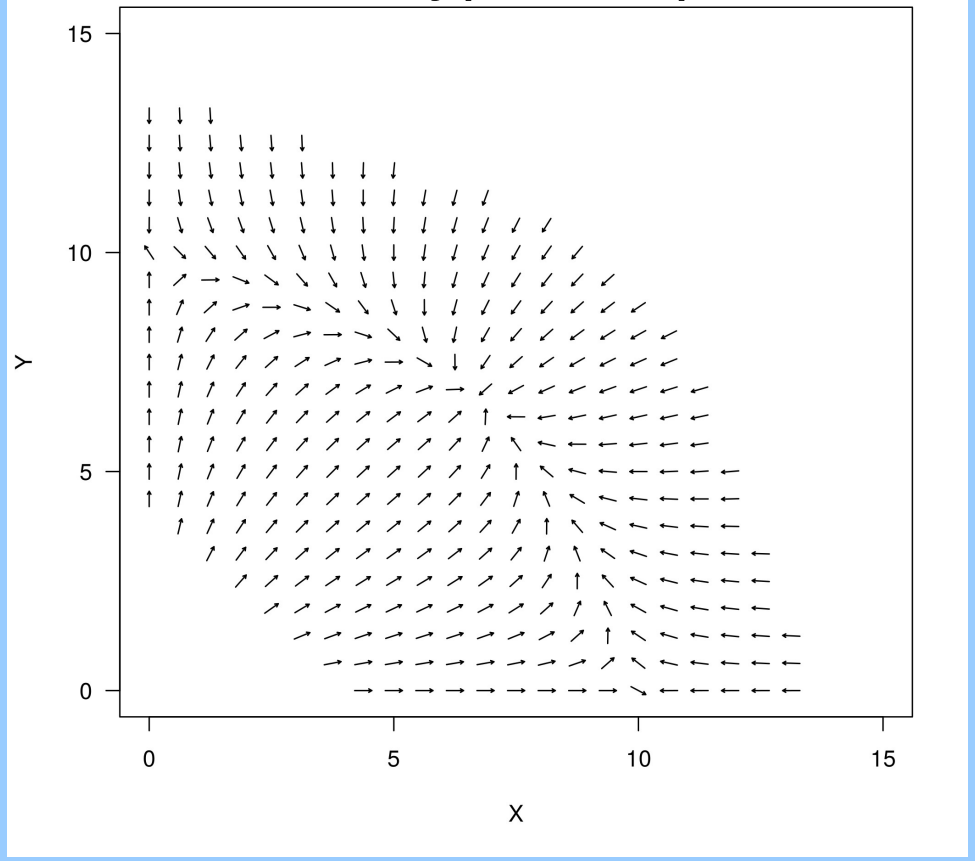
6) Visualize the vector field decompositions

b) visualize the remainder

arrow.type = 'proportional'



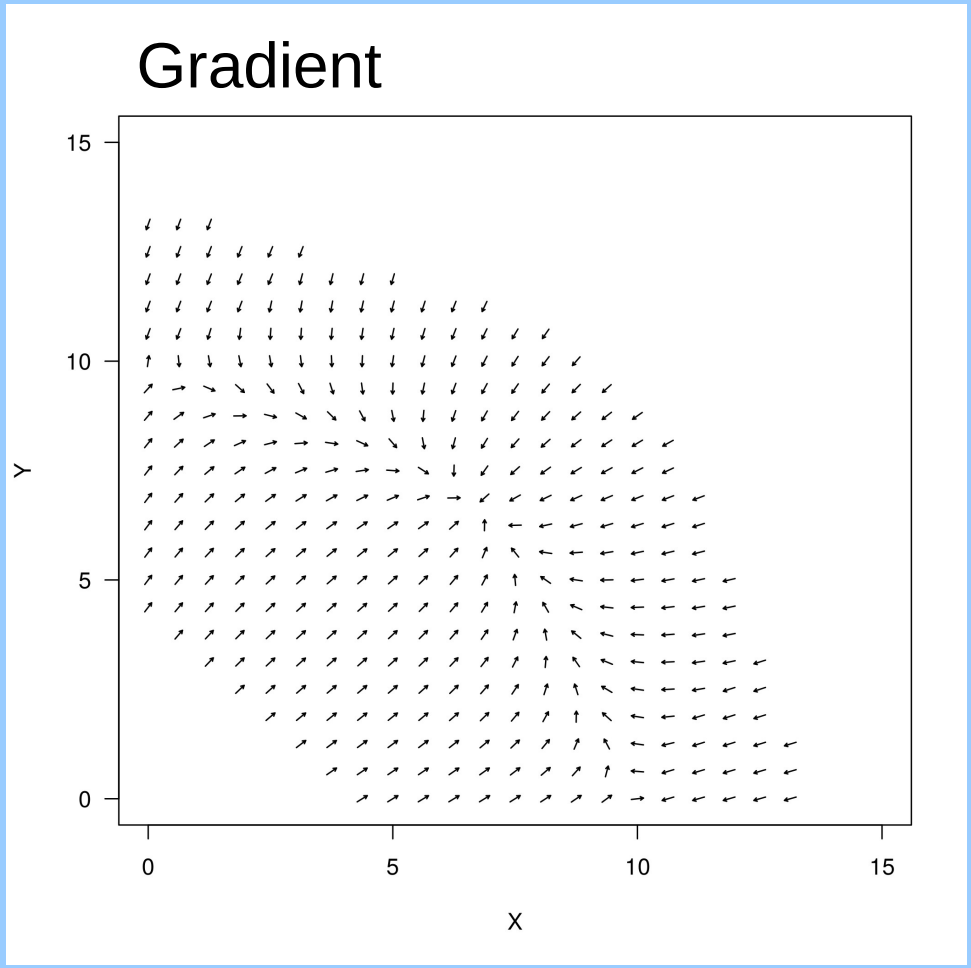
arrow.type = 'equal'

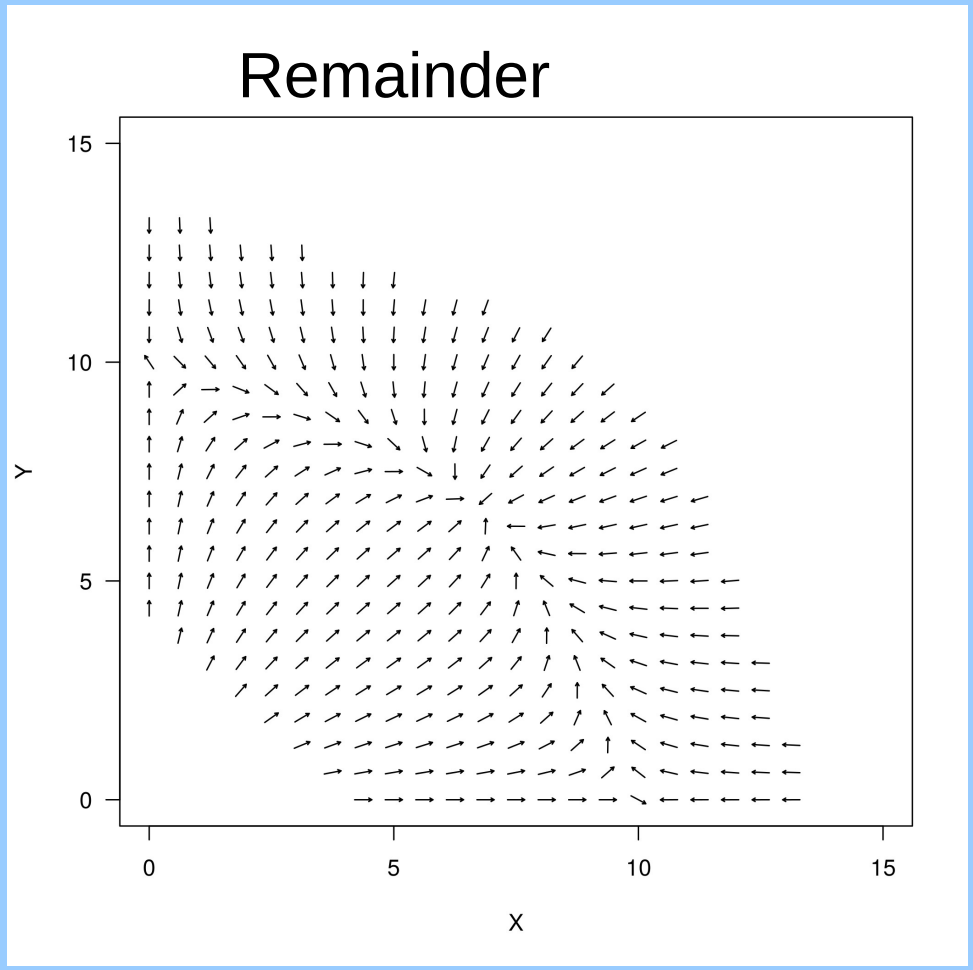



```
VecDecomPlot(x.field = VDAll[,,5], y.field = VDAll[,,6], dens = c(25, 25), x.bound = bounds.x,
                    y.bound = bounds.y, arrow.type = "XXXXXXXX",
                    tail.length = 0.35, head.length = 0.025)
```

Lotka-Volterra Competition: Coexistence

6) Visualize the vector field decompositions

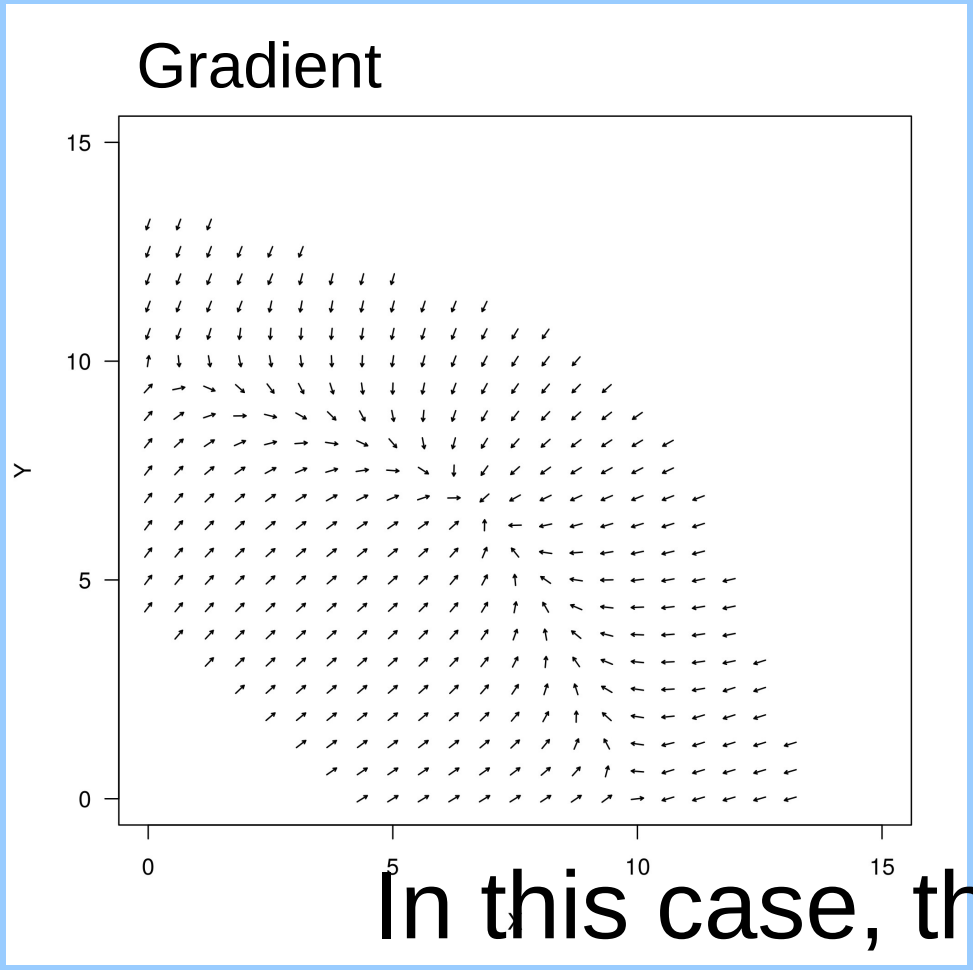


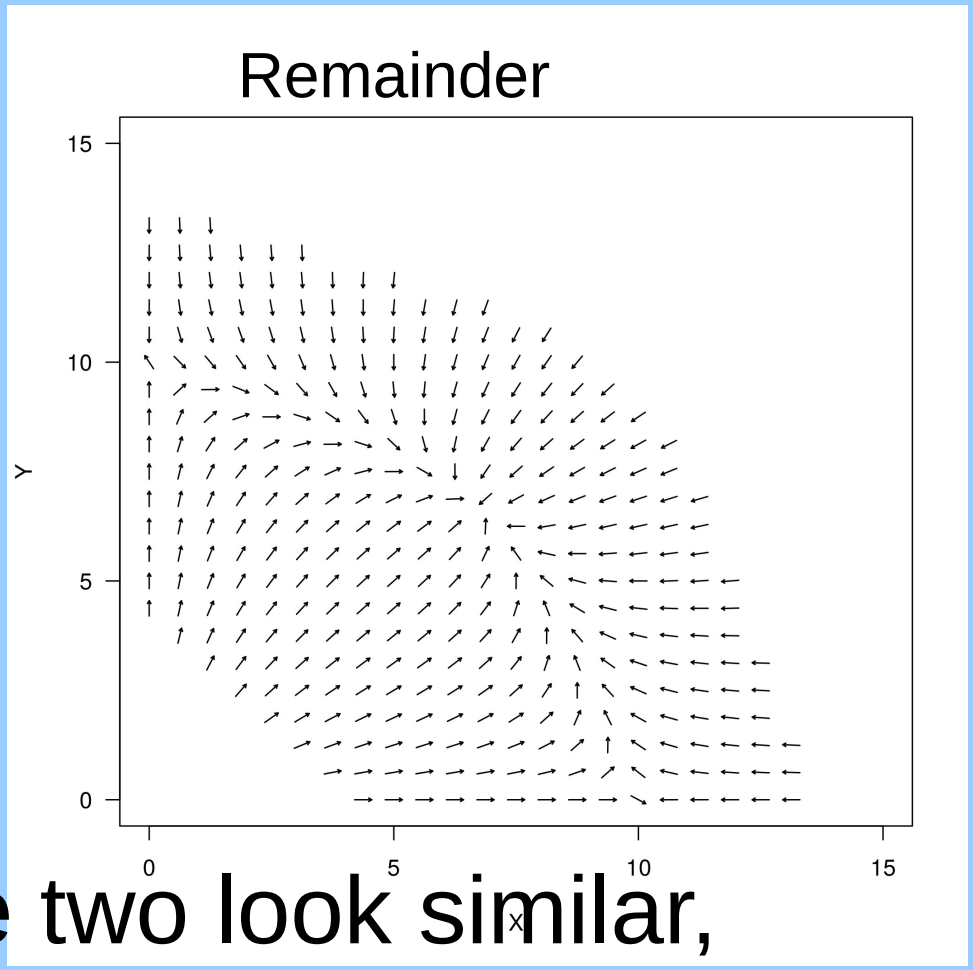


arrow.type = "equal"

Lotka-Volterra Competition: Coexistence

6) Visualize the vector field decompositions

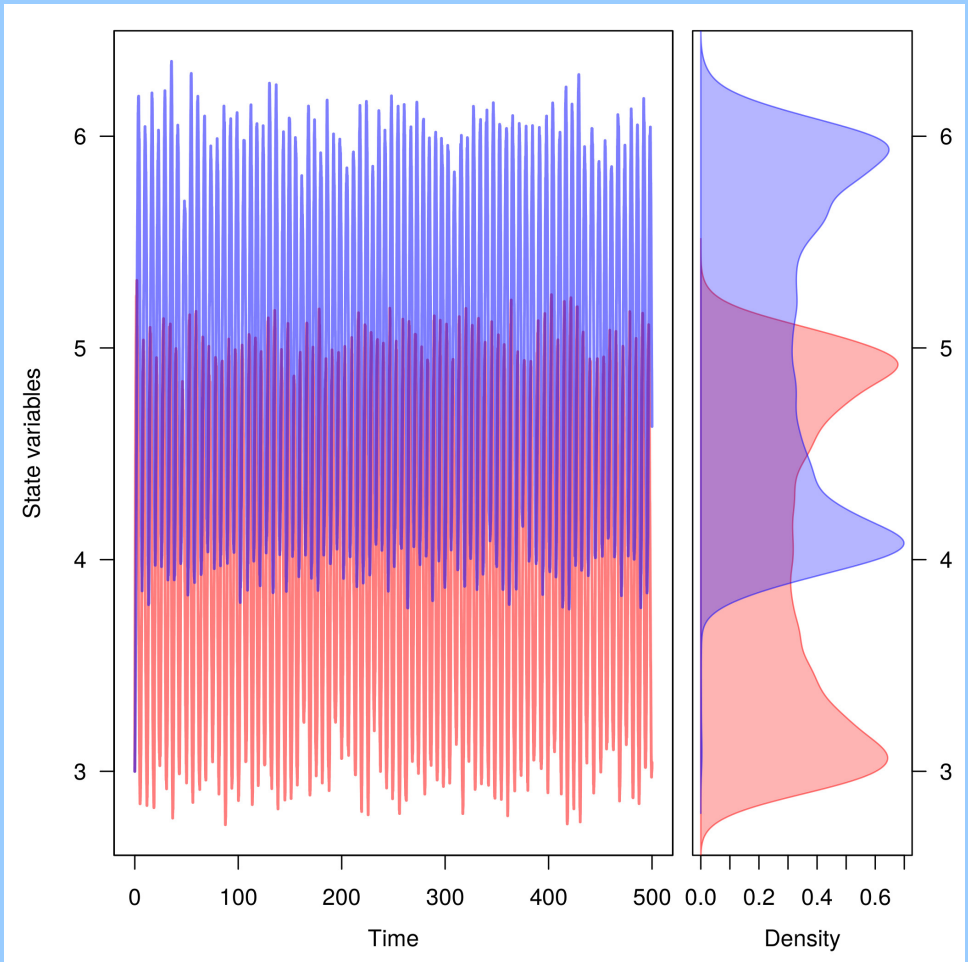


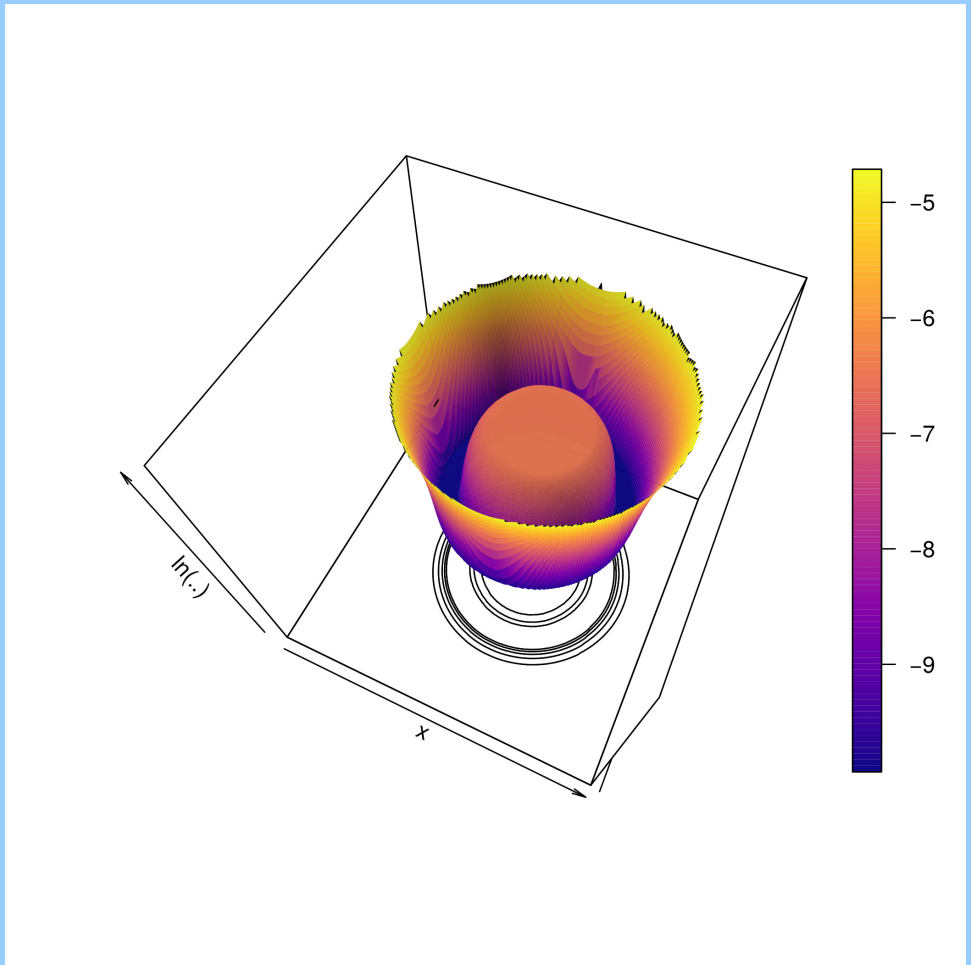


In this case, the two look similar, but could look different.

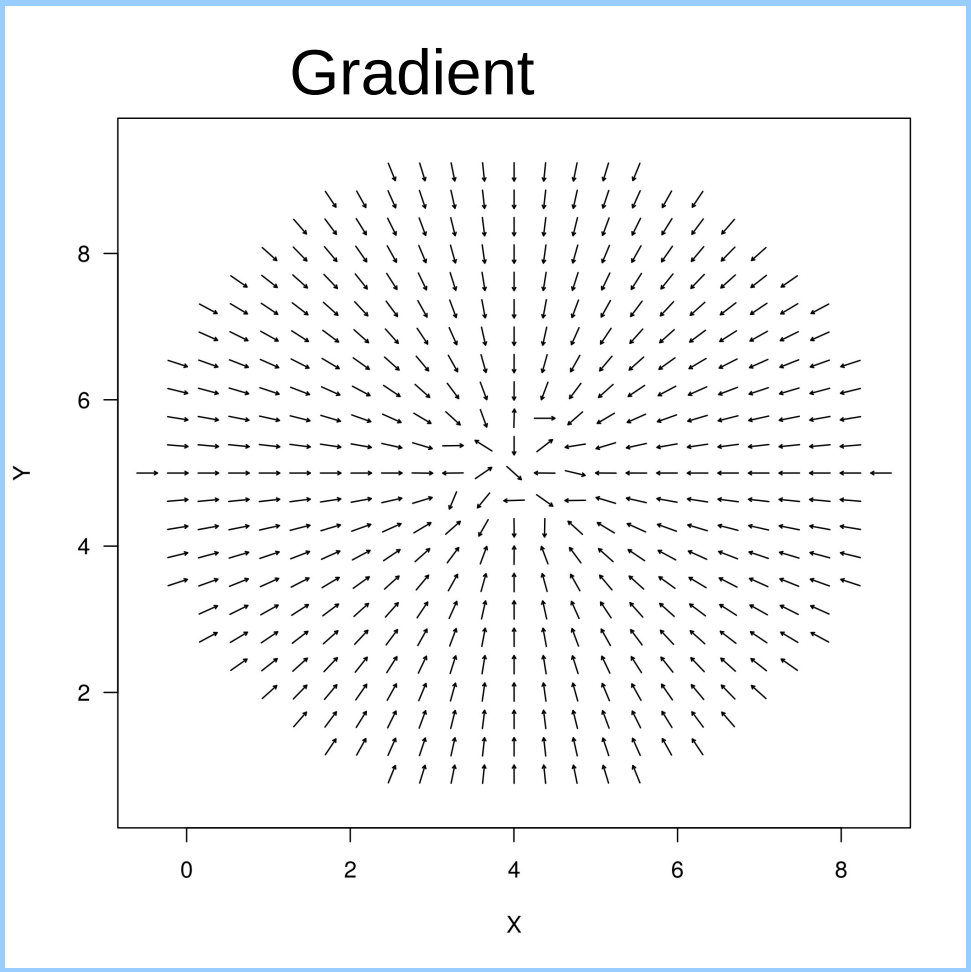
arrow.type = "equal"

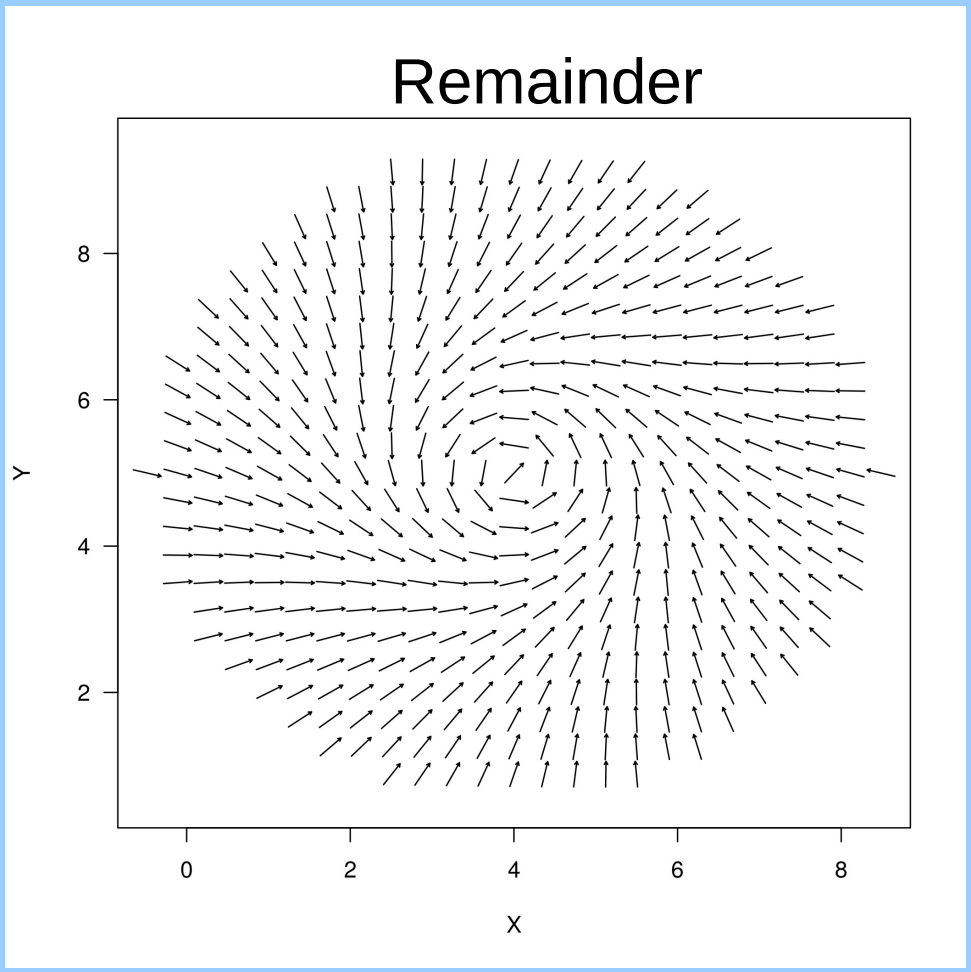
Limit-cycle


State variables
6
5
4
3
0
100
200
300
400
500
Time
0.0
0.2
0.4
0.6
Density


ln(.,)
x
−5
−6
−7
−8
−9

Limit-cycle





remainder vector field, which is the “force that causes trajectories to circulate around level sets of the quasi-potential.”

Other Examples

At stieha.com

stieha@hotmail.com
stieha@case.edu

1) Competition with initial condition dependence
    This shows an example of having multiple stable equilibria,
        Where you need to use QPGlobal() to merge
    Depending on the parameter values,
        may require changing boundary conditions to get good results

2) Consumer-resource model with two stable equilibria
    Appeared in talk, also appears in Moore et al. 2016 as Example 1
    Have multiple stable equilibria, so need to combine local quasi-potentials
        into a global quasi-potential

3) Consumer-resource model with limit cycle (no code yet)
    Appeared in talk,
    Same model as #2, but different parameter values

4) System of equations with a limit cycle
    Example 2 in Moore et al. 2016

5) Combining local quasi-potentials into a global quasi-potential
    In a difficult system
    Example 3 in Moore et al. 2016.
    Download article for code