Logistics

- Current Reading:
  - Mitchell, *Complexity: A Guided Tour*, Chapter 2 (Today)
  - Flake, The Computational Beauty of Nature, Chapter 10 (Monday)
  - May, *Simple Mathematical Models with Very Complicated Dynamics* (Today)

- Homework: Project 1: Dynamical Systems v1.1 is now up.
- Choose a paper and two backups by 1:00pm Today. Email your choices to Bianca. You will be pair with whoever chooses or is assigned the same paper. The first presentation will be next Wednesday (Ben is looking for a partner).

George Kelbley Inbox - Exchange 23 January 2017 at 15:27

cs523 list

To: mfricke@cs.unm.edu

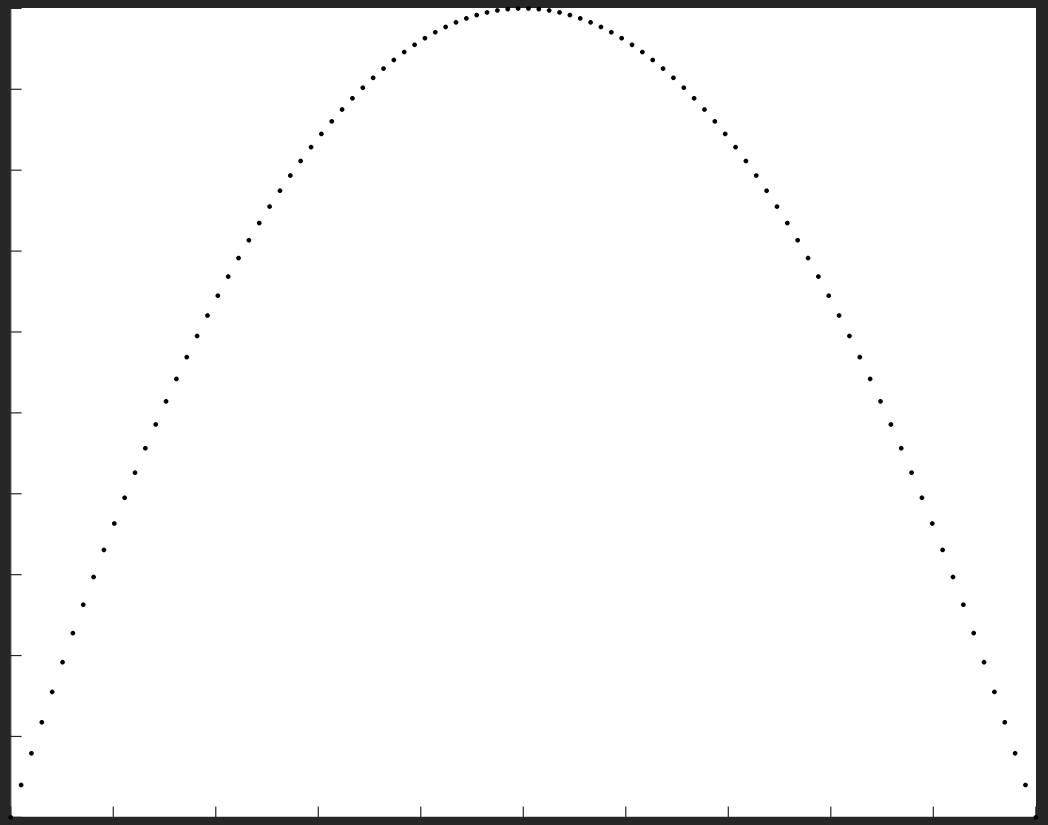
Hi, we have to delete and re-create the cs523 mailing list, its messed up.

Please resubscribe to the mailing list…

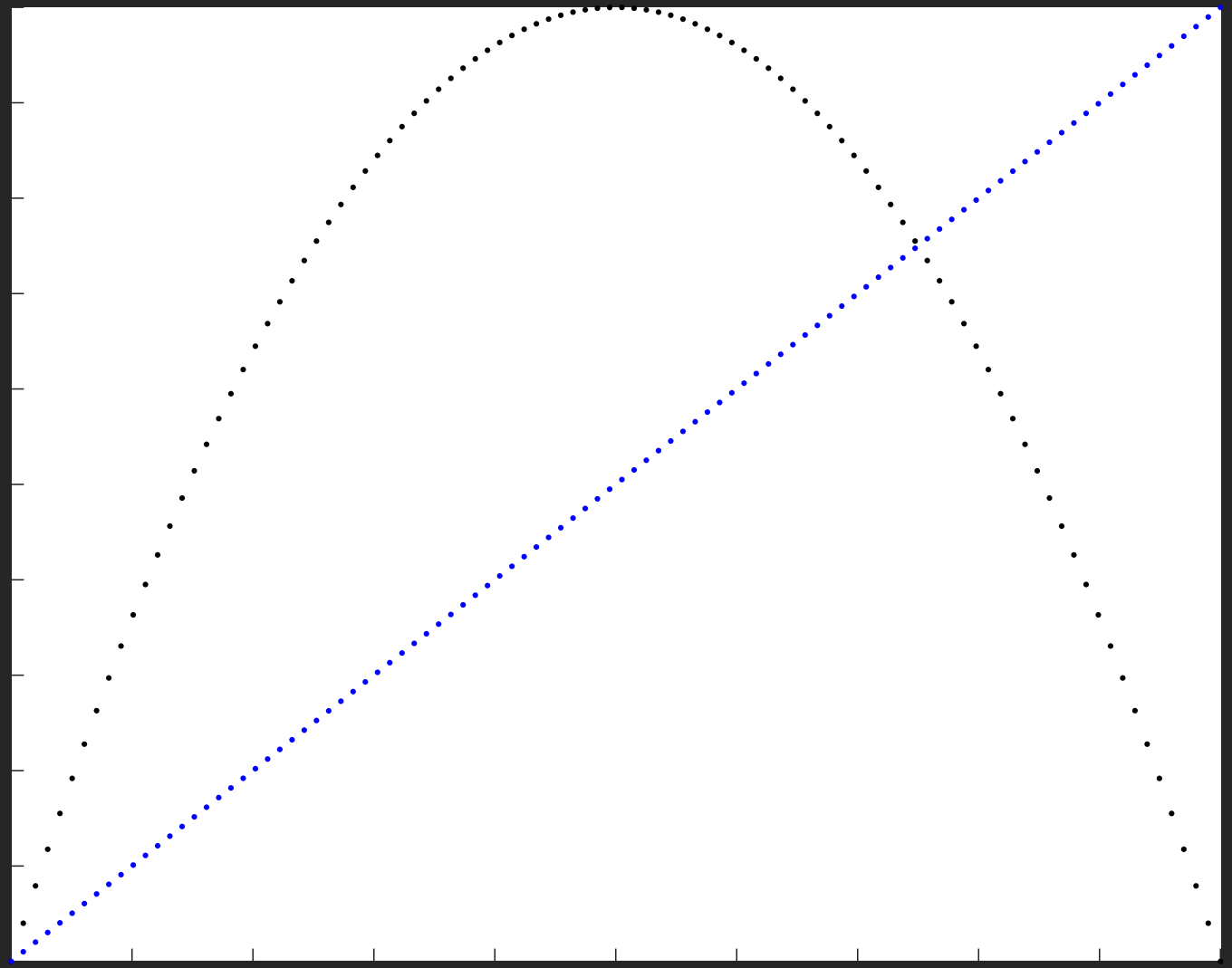
Dynamical Systems and Fixed Point Analysis

Lecture 4

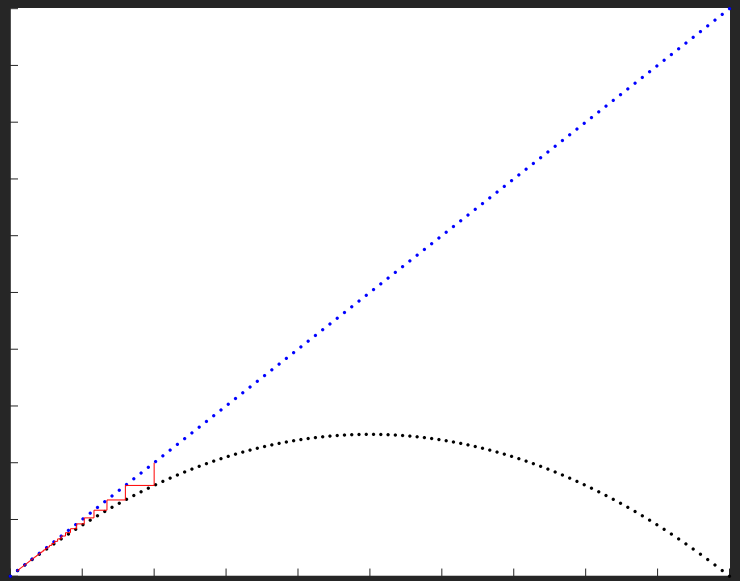
```
function cobweb(F,a,b,traj_x0,num_iterations)

% generate N linearly space values on (a,b)
x=linspace(a,b,num_iterations);

% turn hold on to gather up all plots in one
hold on;

% plot the function F
for t = 1:num_iterations
plot(x(t),F(x(t)),'k.');
end
```

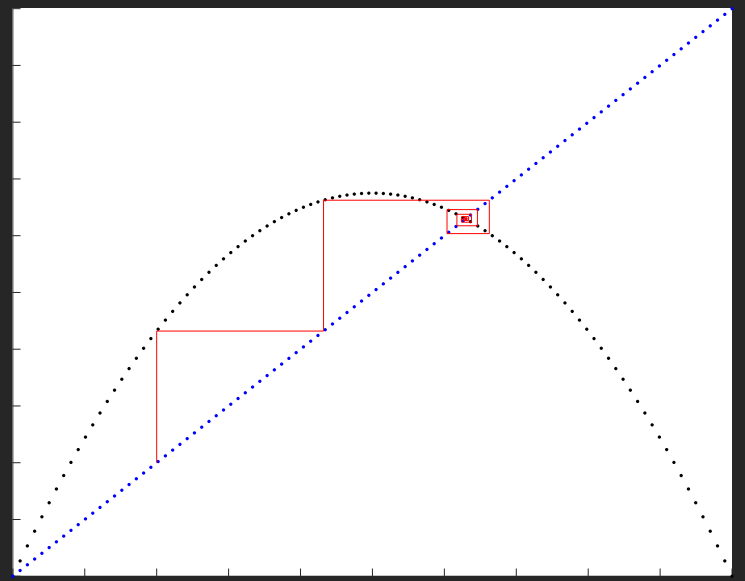


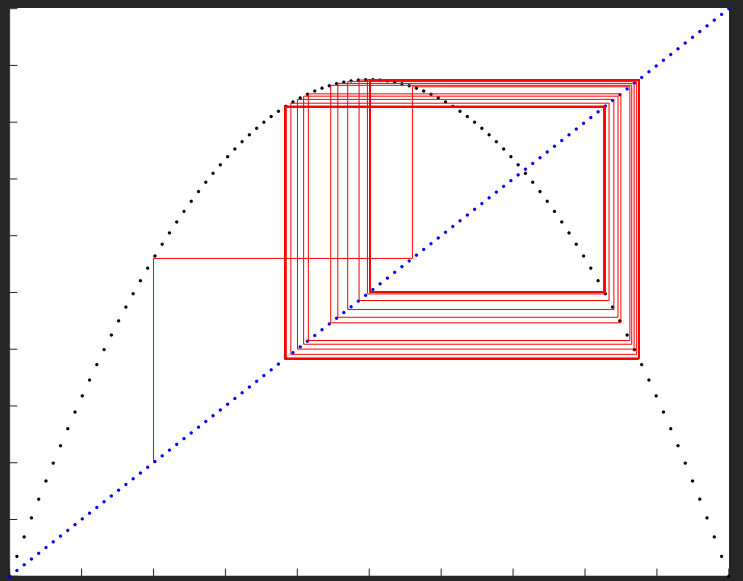
```
function cobweb(F,a,b,traj_x0,num_iterations). . .
% plot the diagonal (if x = F(x))
for t = 1:num_iterations
plot(x(t),x(t),'b.');
end
```



```
function cobweb(F,a,b,traj_x0,num_iterations) . . .

traj_x(1)=traj_x0; % plot trajectory starting at x0
 for t=1:num_iterations
     % Get the next point in the trajectory
     traj_x(t+1)=F(traj_x(t));

     % Draw a line between the diagonal at time t and F at time t
     line([traj_x(t),traj_x(t)],[traj_x(t),traj_x(t+1)],'Color', 'r');

     % Draw a line between F at t+1 and the point on the diagonal a t+1
     line([traj_x(t),traj_x(t+1)],[traj_x(t+1),traj_x(t+1)],'Color', 'r');
 end
```

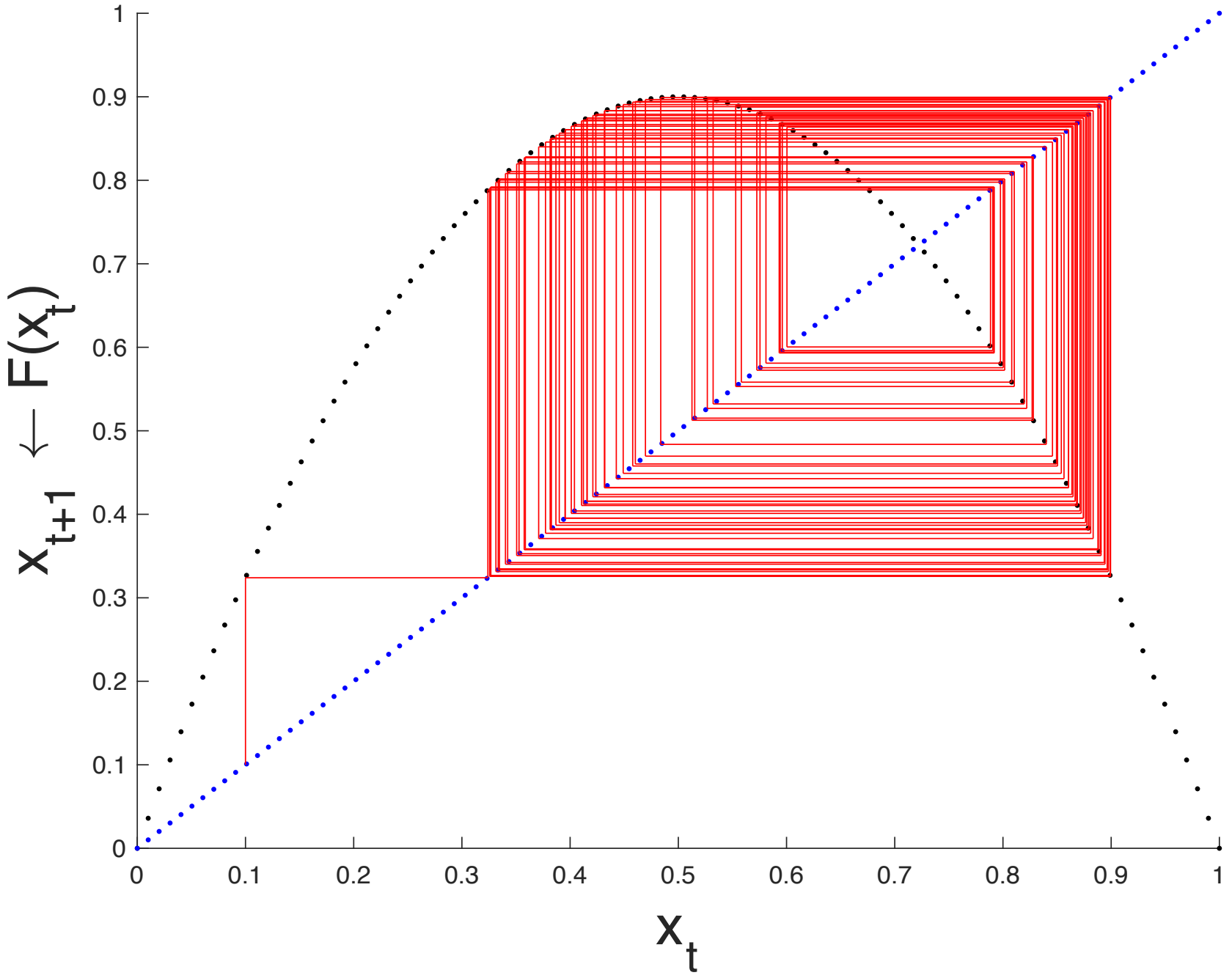




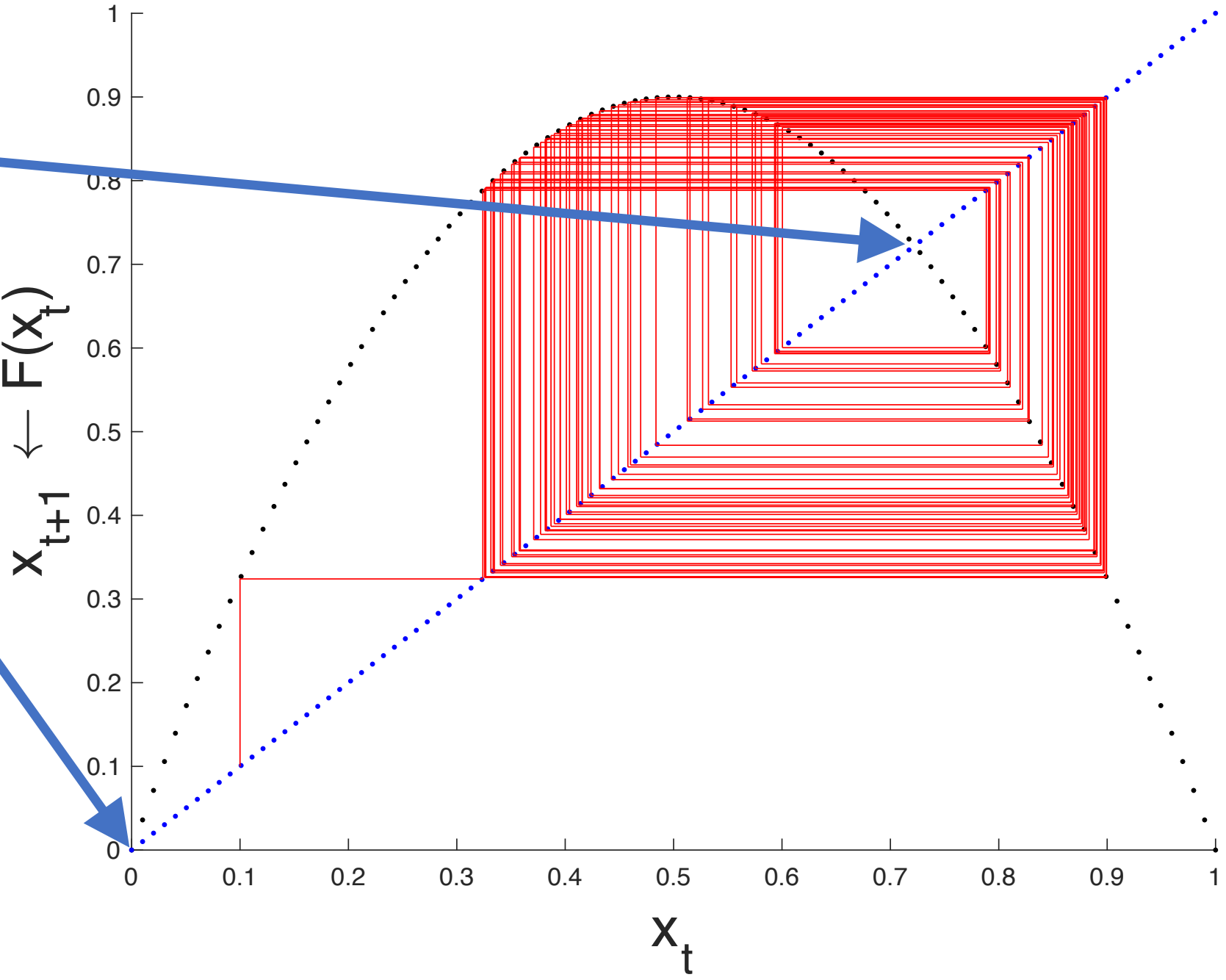


```
function logistic_cobweb(r,a,b,traj_x0,num_iterations)

F = @(x) r*x*(1-x);

% generate N linearly space values on
(a,b)
x=linspace(a,b,num_iterations);

% turn hold on to gather up all plots in
one
hold on;

% plot the function F
for t = 1:num_iterations
plot(x(t),F(x(t)),'k.');
end

% plot the diagonal (if x = F(x))
for t = 1:num_iterations
plot(x(t),x(t),'b.');
end
```

```
 traj_x(1)=traj_x0; % plot trajectory starting at x0
 for t=1:num_iterations
     % Get the next point in the trajectory
     traj_x(t+1)=F(traj_x(t));

     % Draw a line between the diagonal at time t and F at time t
     line([traj_x(t),traj_x(t)],[traj_x(t),traj_x(t+1)],'Color', 'r');

     % Draw a line between F at t+1 and the point on the diagonal a t+1
     line([traj_x(t),traj_x(t+1)],[traj_x(t+1),traj_x(t+1)],'Color', 'r');
 end

 xlabel('x_t', 'Fontsize', 22)
 ylabel('x_{t+1} \leftarrow F(x_{t})', 'Fontsize', 22)

hold off;
```


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
$x_{t+1} \leftarrow F(x_t)$
x_t

Fixed points



State Spaces: A Geometric Approach (Abraham and Shaw, 1984)

- History of system is represented graphically
- Trajectory: A curve in the state space representation, connecting subsequent observations
- Time series: A graph of the trajectory w.r.t. to time
- Phase space (why phase not state?)

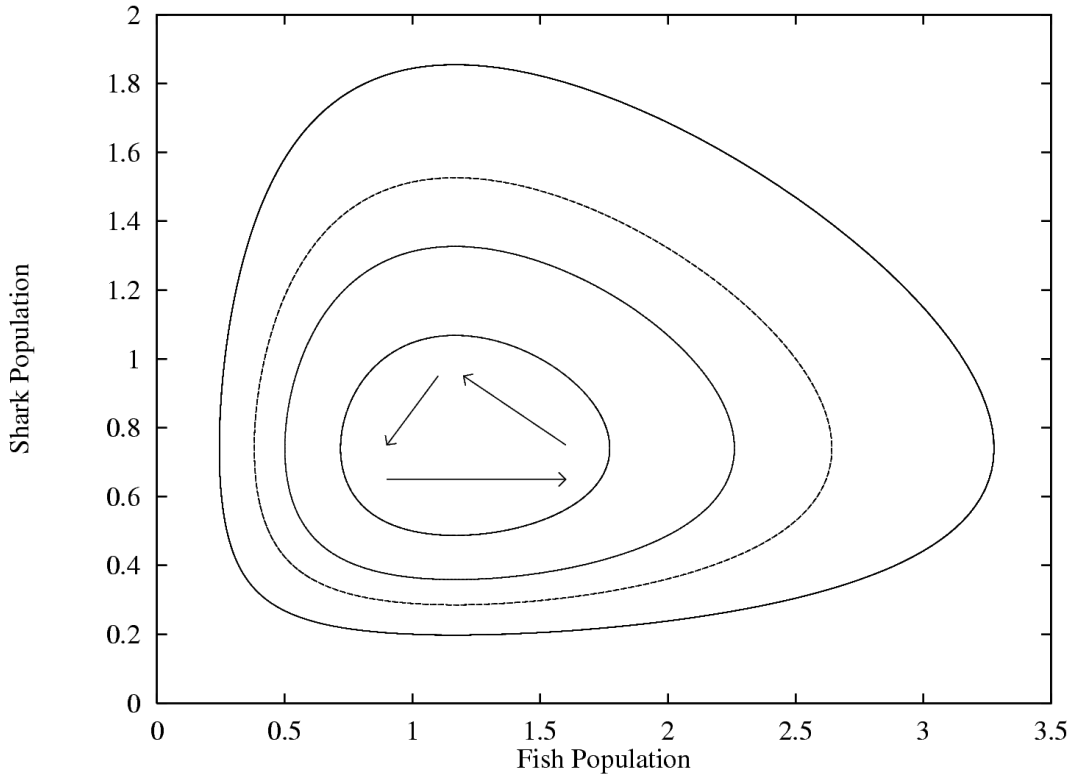
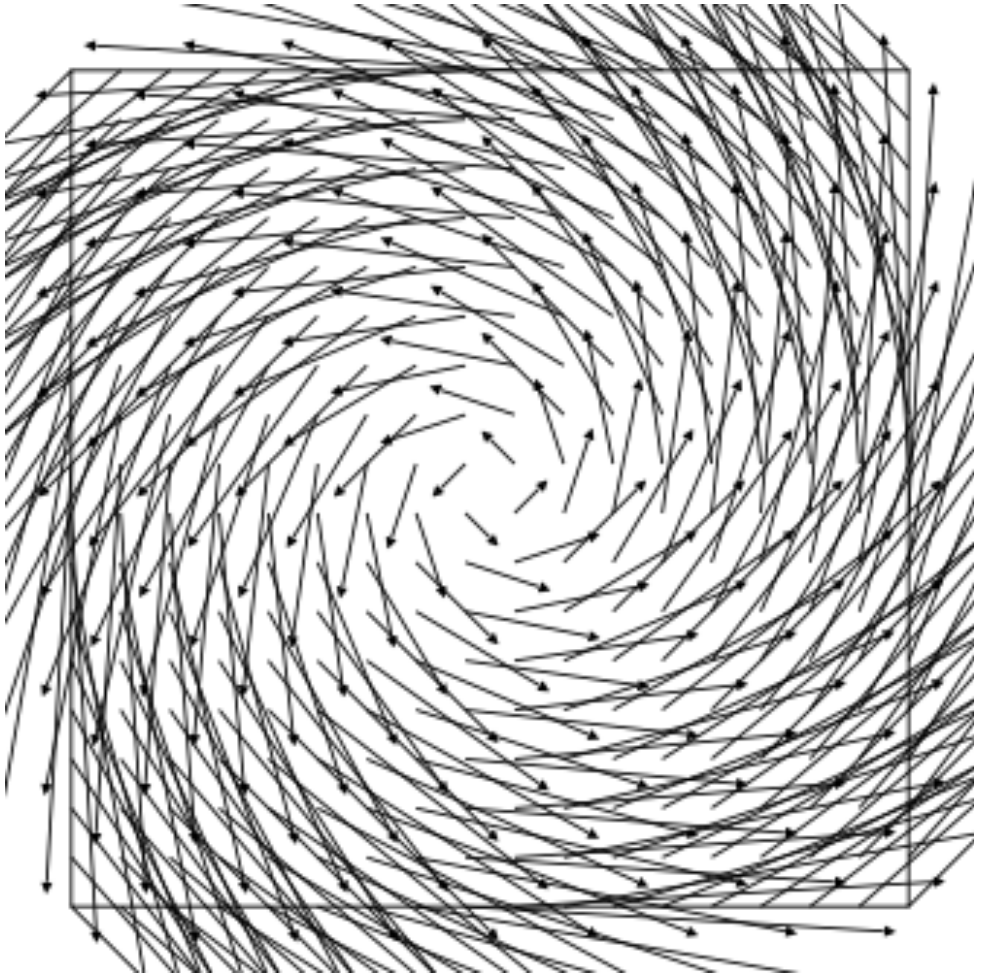


Figure 12.1 A simple Lotka-Volterra attractor which shows four (out of an infinite number of possible) limit cycles. The value of the four parameters are equal to 3.029850, 4.094132, 1.967217, and 2.295942, which yields a fixed point at 1.1671, 0.740047.

Example: Lotka-Volterra Equations

Phase Portraits

- All possible trajectories through the phase space (flows) or state space (maps)
- At any point on any of these curves, a velocity vector may be derived
- Can view a dynamical system as a vector field
- Velocity vector field can be derived from phase portrait by differentiation

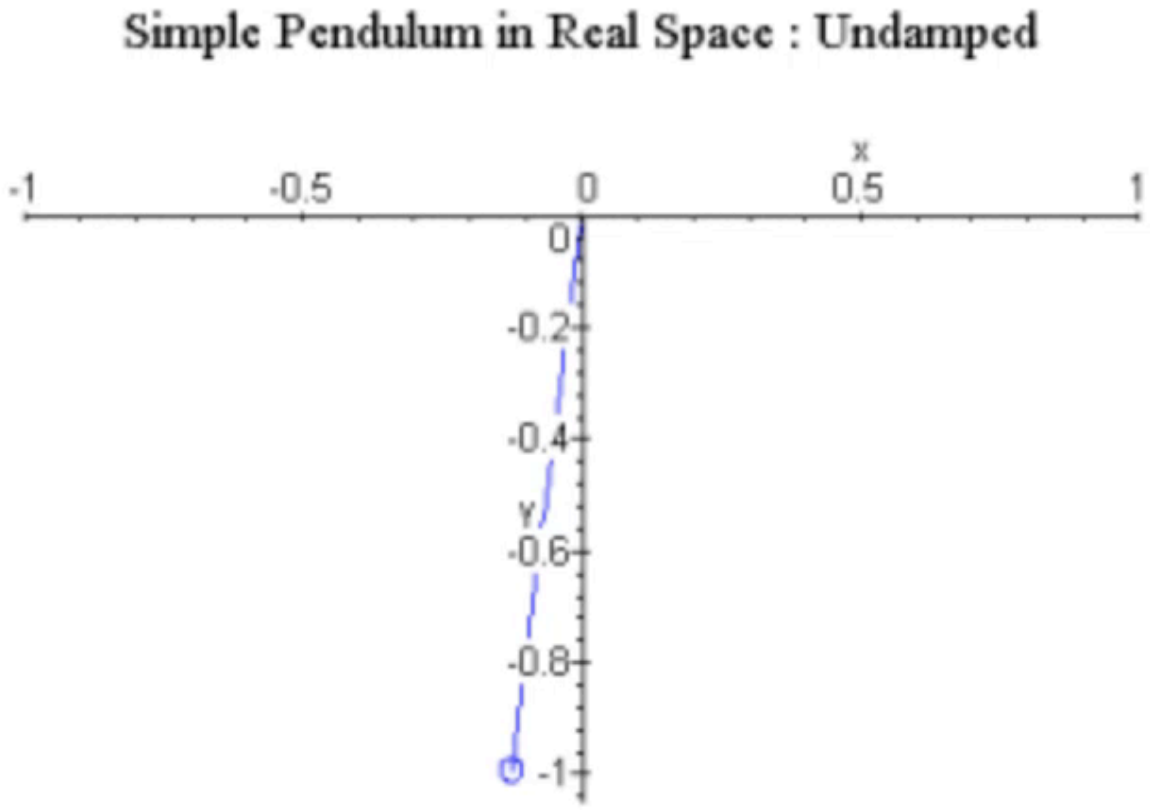


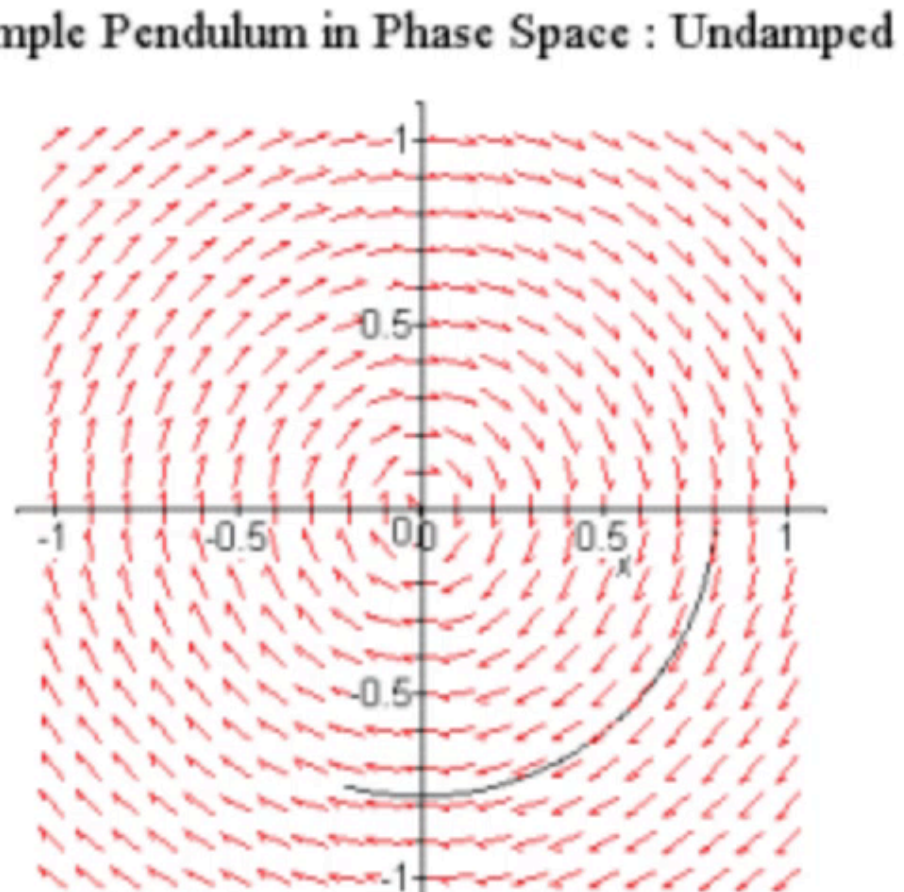
Example Vector Field; Wikipedia (2007)

System of equations for an undamped pendulum:

$$y_1' = y_2$$

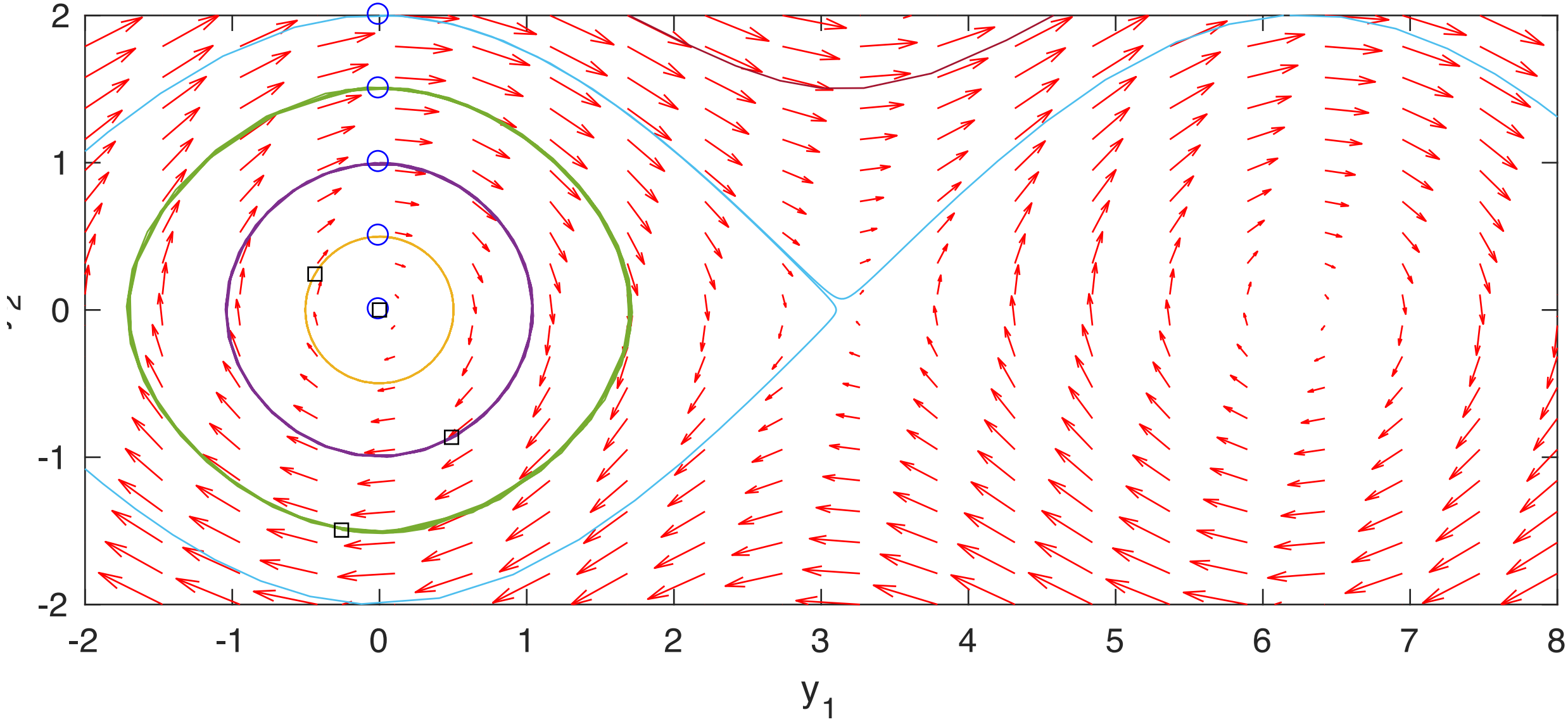
$$y_2' = -\sin(y_1)$$





```
function phase_plot()

F = @(t,y) [y(2); -sin(y(1))];

y1 = linspace(-2,8,20);
y2 = linspace(-2,2,20);

% Creates two matrices one for all the x-values on the grid, and one for
% all the y-values on the grid. [x,y] = meshgrid(y1,y2);
u = zeros(size(x));
v = zeros(size(x));

% we can use a single loop over each element to compute the derivatives at
% each point (y1, y2)
t=0; % we want the derivatives at each point at t=0, i.e. the starting time
for i = 1:numel(x)
    y_deriv = F(t,[x(i); y(i)]);
    u(i) = y_deriv(1);
    v(i) = y_deriv(2);
end
```

```
% Setup the plot
figure(gcf)
xlabel('y_1')
ylabel('y_2')
axis tight equal;
xlim([-2,8])
ylim([-2,2])

% Draw the vector field
quiver(x,y,u,v,'r');

% Plot some sample trajectories
hold on
max_time = 50;
for trajectory = [0 0.5 1 1.5 2 2.5, 3, 3.5]
    [ts,ys] =
ode45(F,[0,max_time],[0;trajectory]);
    plot(ys(:,1),ys(:,2))
    plot(ys(1,1),ys(1,2),'bo') % starting point
    plot(ys(end,1),ys(end,2),'ks') % ending
point
end
hold off
```


2
1
0
-1
-2
-2
-1
0
1
2
3
4
5
6
7
8
y1

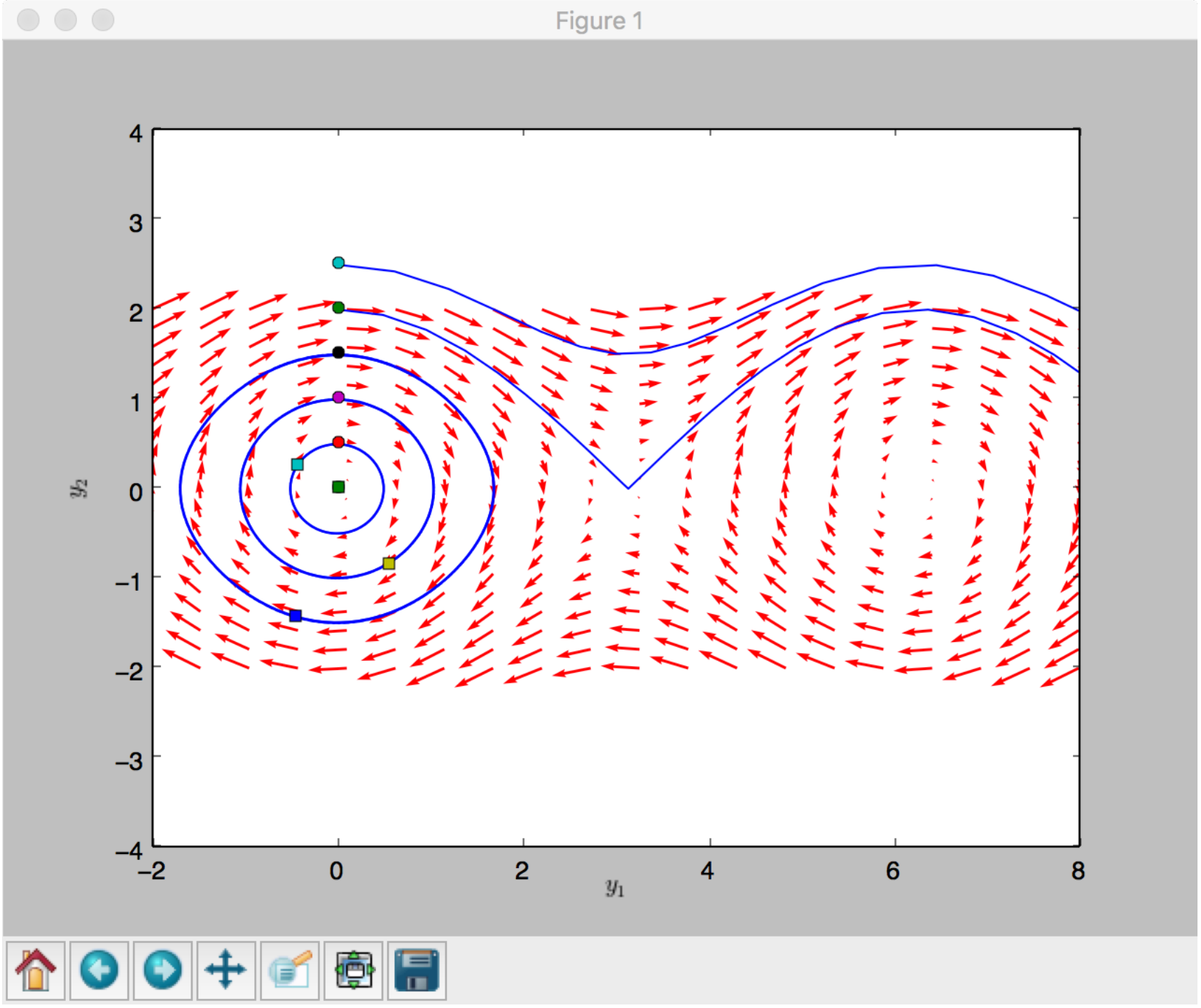
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

def F(y, t):
    y1, y2 = y
    return [y2, -np.sin(y1)]
y1 = np.linspace(-2.0, 8.0, 20)
y2 = np.linspace(-2.0, 2.0, 20)
y1, y2 = np.meshgrid(y1, y2)
t = 0
u, v = np.zeros(y1.shape), np.zeros(y2.shape)
NI, NJ = Y1.shape

for i in range(NI):
    for j in range(NJ):
        x = y1[i, j]
        y = y2[i, j]
        y_deriv = f([x, y], t)
        u[i,j] = y_deriv[0]
        v[i,j] = y_deriv[1]
```

```
Q = plt.quiver(y1, y2, u, v, color='r')


for y20 in [0, 0.5, 1, 1.5, 2, 2.5]:
    tspan = np.linspace(0, 50, 200)
    y0 = [0.0, y20]
    ys = odeint(F, y0, tspan)
    plt.plot(ys[:,0], ys[:,1], 'b-') #
path
    plt.plot([ys[0,0]], [ys[0,1]], 'o')
# start
    plt.plot([ys[-1,0]], [ys[-1,1]],
's') # end


plt.xlim([-2, 8])
plt.show()
```

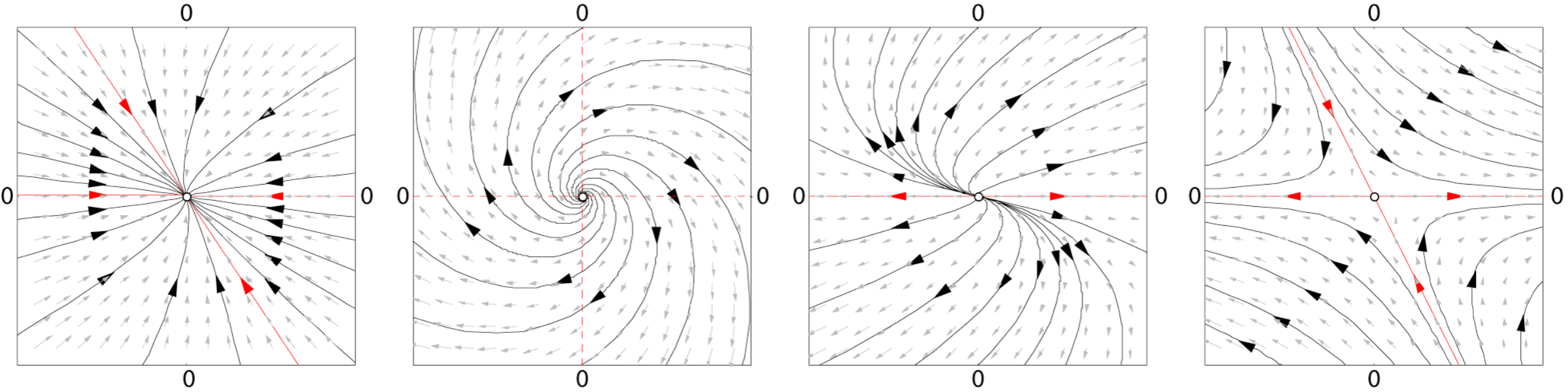
```
#Setup the plot
plt.xlabel('$y_1$')
plt.ylabel('$y_2$')
plt.xlim([-2, 8])
plt.ylim([-4, 4])
```

For python:

Numpy
odeint
matplotlib



Example Trajectories
Linear Vector Fields



Attractor (stable) ***Repellor (unstable)*** ***Repellor (unstable)*** ***Saddle Point (unstable for some init conditions)***

Define the ultimate fate of the system over long time periods.

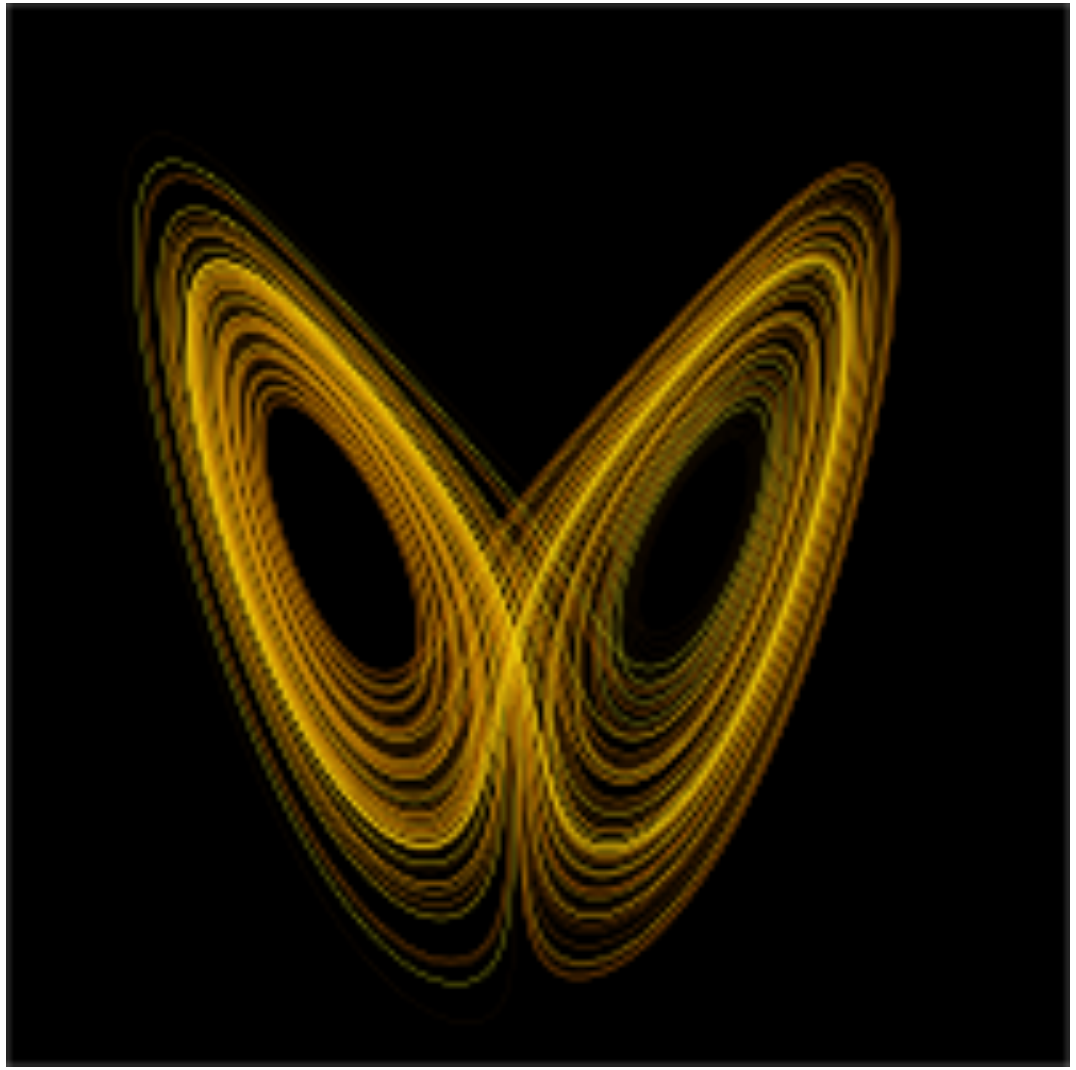
Wikipedia, 2007

```
% Plot the Lorenz attractor. A model of a weather system.
function [x,y,z] = lorenz(rho, sigma, beta, …
initial_values, max_time, eps, rate, line_style)

% Options for the ODE solver
options = odeset('RelTol',eps,'AbsTol',[eps eps eps/10]);

for t = 1:max_time
    [T,X] = ode45( @(T,X) F(T, X, sigma, rho, beta), [0, t],…
                   initial_values, options);

x = X(:,1);
y = X(:,2);
z = X(:,3);

plot3(x, y, z, line_style)
pause(rate)
drawnow
end

end
```

```
% Lorenz's System of Differential Equations

function dx = F(T, X, sigma, rho, beta)

    dx = zeros(3,1);
    dx(1) = sigma*(X(2) - X(1));
    dx(2) = X(1)*(rho - X(3)) - X(2);
    dx(3) = X(1)*X(2) - beta*X(3);
end
```

Attractors

- Initially, a trajectory through a dynamical system may be erratic. This is known as the initial transient, or start-up transient.
  - Asymptopia
- The asymptotic behavior of the system is known as equilibrium, steady state, or dynamic equilibrium.
  - This does not necessarily imply a static equilibrium or static state.
  - The only equilibrium states which can be observed experimentally are those modeled by limit sets which receive most of the trajectories.
- These are called attractors.



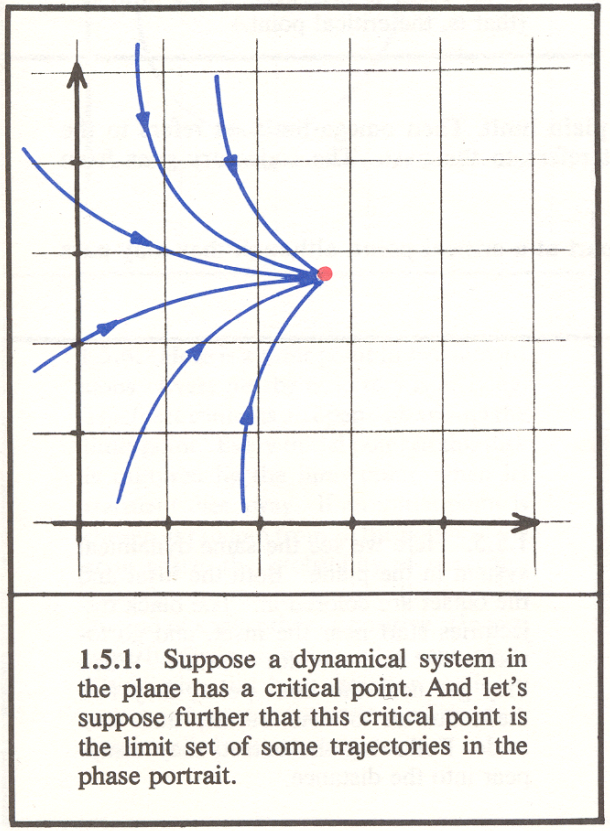
The Lorenz Attractor Ruele and Takens (1971)

Strange Attractors

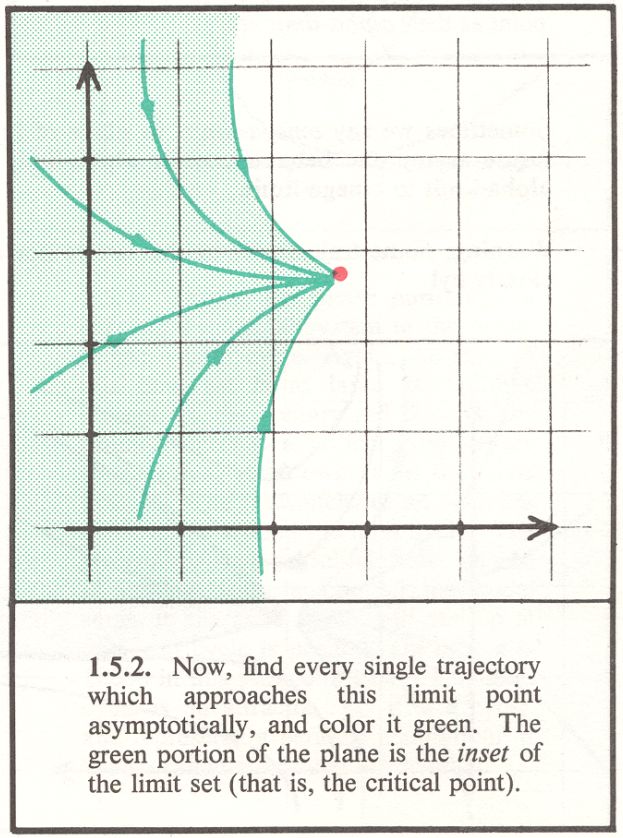
- The separatrices are FRACTAL.
- Think about that for a minute…

Attractors
(from Abraham and Shaw, 1984)

- Basin of attraction: The points of all trajectories that converge to a given attractor
- The dividing boundaries (or regions) between different attractor regions (basins) are called separatrices.



1.5.1. Suppose a dynamical system in the plane has a critical point. And let's suppose further that this critical point is the limit set of some trajectories in the phase portrait.



1.5.2. Now, find every single trajectory which approaches this limit point asymptotically, and color it green. The green portion of the plane is the *inset* of the limit set (that is, the critical point).

Next Time

- Some analytical methods for finding and describing fixed points.
- Lyapunov Exponents