Lecture 7

Order Out of Chaos

Lyapunov Exponents: Recall from Last Time

The Lyapunov exponent for maps is:

$$\lambda = \ln|f'(x^*)| = \begin{cases} < 0 \text{ if } |f'(x^*)| < 1 \\ > 0 \text{ if } |f'(x^*)| > 1 \end{cases}$$

Lyapunov Exponents: Numerical Estimation

Lyapunov exponent: $\lambda = \ln|f'(x^*)|$

Rate of divergence over time:

$$f'(x) = \frac{dx_{t+1}}{dx_t}$$

Lyapunov Exponents: Numerical Estimation

Lyapunov exponent: $\lambda = \ln |f'(x^*)|$

Rate of divergence over time:

$$f'(x) = \frac{dx_{t+1}}{dx_t}$$

Assumption: Lyapunov exponent is the same everywhere in the basin of attraction.

Lyapunov Exponents: Maximum

$$\lambda_{\max} = \lim_{t \gg 1} \frac{1}{t} \ln \left| \frac{f_t(x)}{dx} \right|$$

This is really the averaged sum of the Lyapunov exponents, λ.

Lyapunov Exponents: Maximum

$$\lambda\text{: Short term behaviour}$$

$$\lambda_{\text{max}}\text{: Long term behaviour}$$

Lyapunov Exponents: Numerical Estimation

Since maps depend on the previous step, we can show that:

Mean rate of divergence:

$$\lambda_{\max} = \lim_{t \gg 1} \frac{1}{t} \ln \left| \frac{f_t(x)}{dx} \right| = \frac{1}{t_{\max}} \sum_{t=t_0}^{t_{\max}} \ln \left| \frac{dx_{t+1}}{dx_t} \right|$$

Lyapunov Exponents: Numerical Estimation

This is our approximation of $\lambda_{\max}$

$$\frac{1}{t_{\max}} \sum_{t=t_0}^{t_{\max}} \ln \left| \frac{dx_{t+1}}{dx_t} \right|$$

Lyapunov Exponents: Numerical Estimation

We want to look at the behaviour near x^*.

How can we know x_t is near x^*?

Lyapunov Exponents: Numerical Estimation

We want to look at the behaviour near x^*.

How can we know x_t is near x^*?

We iterate the map long enough that x_t crosses the separatrix that defines the long term behaviour.

Lyapunov Exponents: Numerical Estimation

This is our approximation of $\lambda_{\max}$

$$\lambda_{\max} = \frac{1}{t_{\max} - t_{\text{start}}} \sum_{t=t_{\text{start}}}^{t_{\max}} \ln \left| \frac{dx_{t+1}}{dx_t} \right|$$

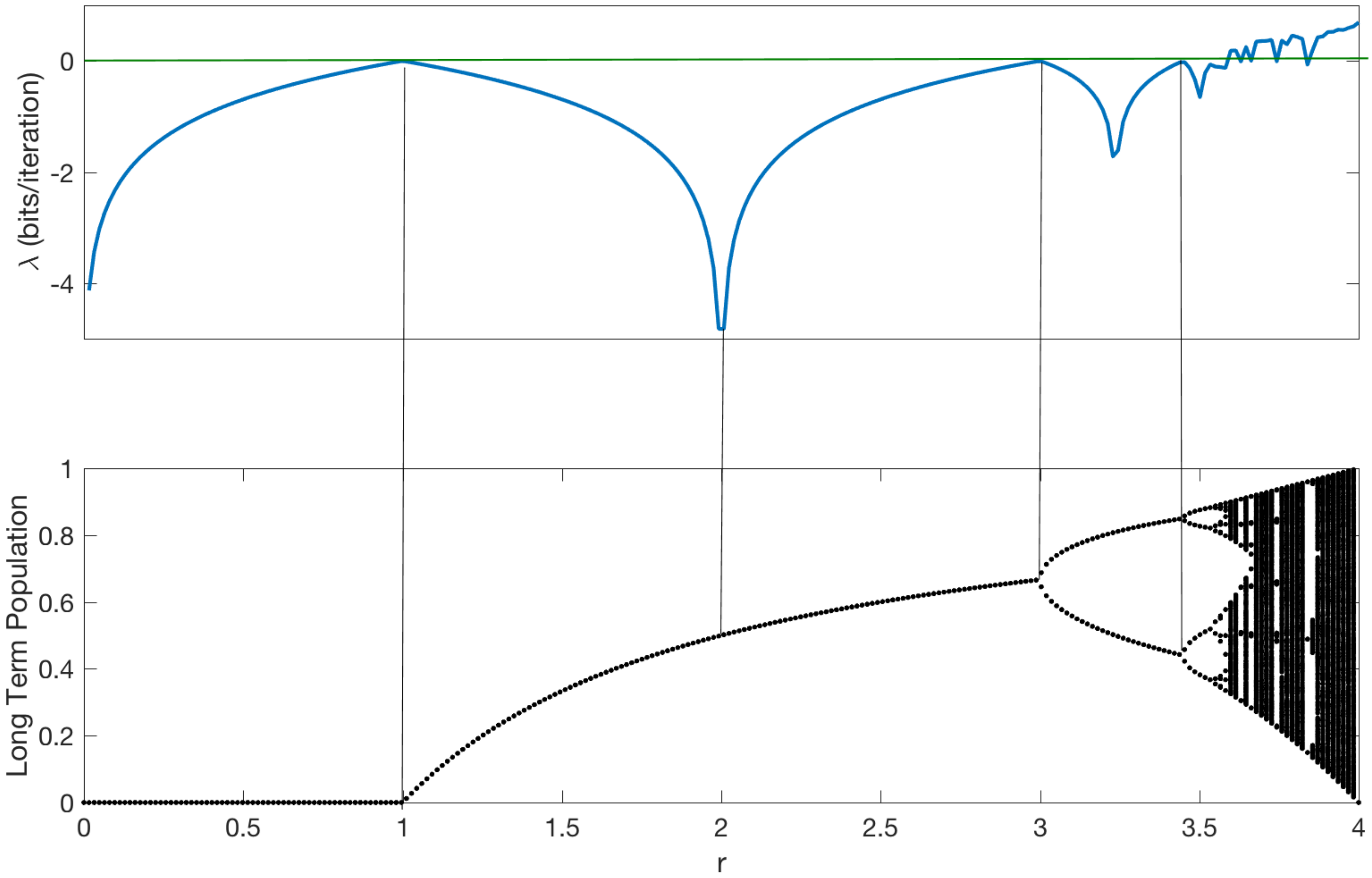
Where t_{start} exceeds the transient period.

Lyapunov Exponents: Numerical Estimation

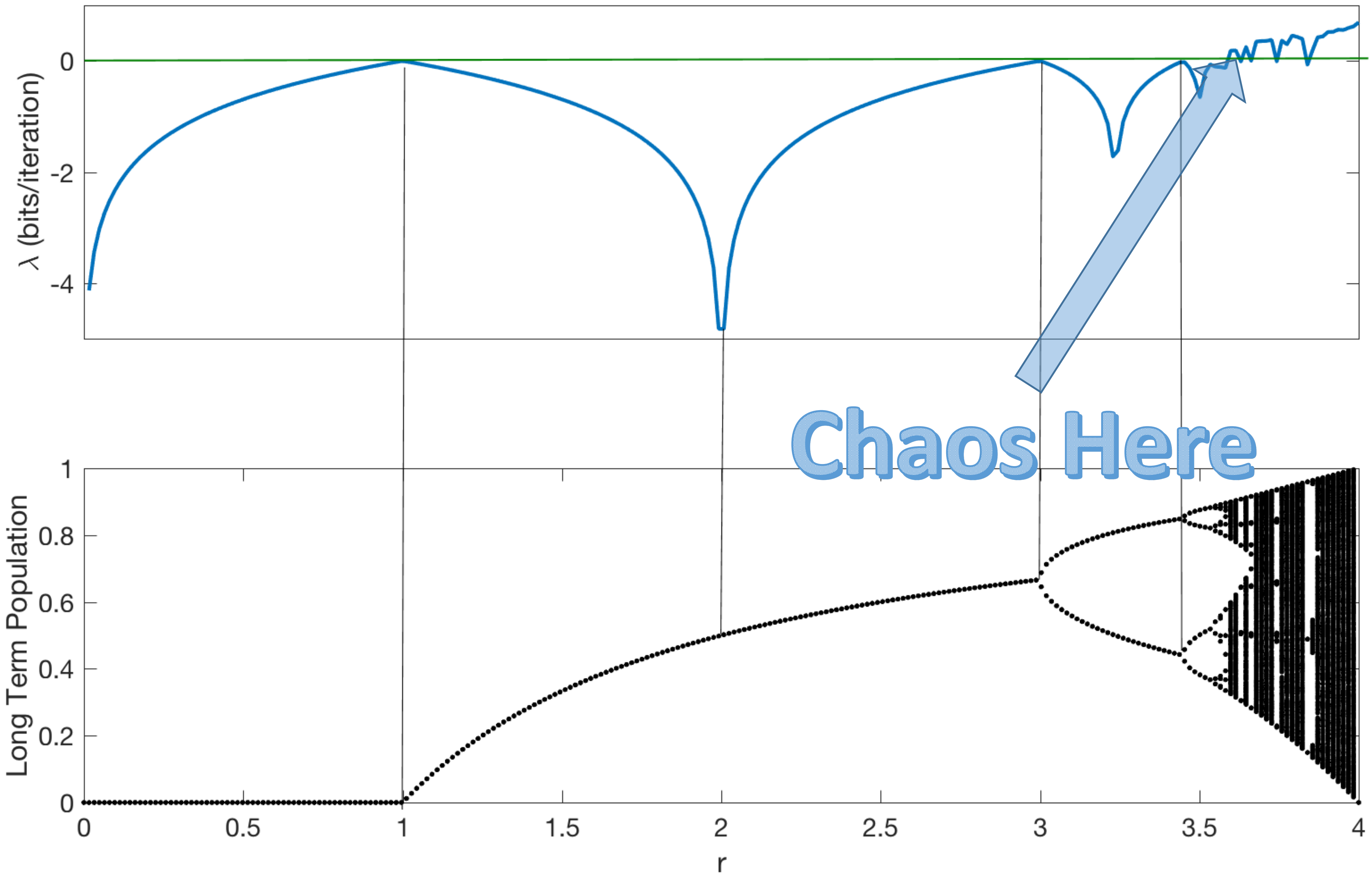
Code it up!

```
function ls = lyapunov(F, F_deriv, n_samples, param_range, transient_time, max_time)
% Takes a dynamical 1-parameter map and plots the Lyapunov exponents as a function of
% the parameter

% Create a vector of parameter values to evaluate
param_values = linspace(param_range(1), param_range(2), n_samples);

ls=[]; % This stores the Lyapunov exponents

for param=param_values
    x_t=rand(1); % Sample over many initial values
    lyp_exps = [];

    for(t = 1:max_time)

        % Evaluate the user defined map F
        x_t=F(x_t,param);

       % Wait until the transient period is over
       if(t > transient_time)
           % Evaluate the derivative at the current point
           lyp_exps = [lyp_exps, F_deriv(x_t, param)];
       end
    end

    % Calculate Lyapunov approximation for the vector of derivatives
    ls = [ls, mean(log((abs(lyp_exps))))];
end
```

```
% Show the Lyapunov exponents and bifurcation plot for the logistic map
F = @(x,r) r*x*(1-x) % Define the logistic map
F_deriv =@(x,r) r-2*r*x % Define the derivative of the logistic map
x0 = 0.5 % An initial value for the bifurcation plot
n_samples = 250 % Number of points to plot
param_range = [0,4] % Parameter range to plot, r for the logistic map
transient_time = 500 % Make sure we are in the fixed point's basin
max_time = 1000    % This minus the transient time is the number
                   % of Lyapunov samples to average over

% Calculate the Lyapunov exponents
ls = lyapunov(F, F_deriv, n_samples, param_range, transient_time, max_time);

% Calculate the long term populations
bs = bifurcation(F,x0,param_range(1),param_range(2),n_samples,max_time);

% Plot the results
subplot(2,1,1)
plot(linspace(param_range(1), param_range(2), n_samples), ls, 'b-');
subplot(2,1,2)
plot(linspace(param_range(1), param_range(2), n_samples), bs, 'k.');
```



Top Panel: Plot of Lyapunov Exponents, Bottom Panel: Bifurcation Plot for r=0.5



Top Panel: Plot of Lyapunov Exponents, Bottom Panel: Bifurcation Plot for r=0.5

Lyapunov Exponents for 2D Maps

The Duffing or Holmes Map:

$$x_{t+1} = y_t$$

$$y_{t+1} = -bx_t + ay_t - y_t^3$$

Chaotic at $a = 2.75$ and $b = 0.2$.

Lyapunov Exponents for 2D Maps

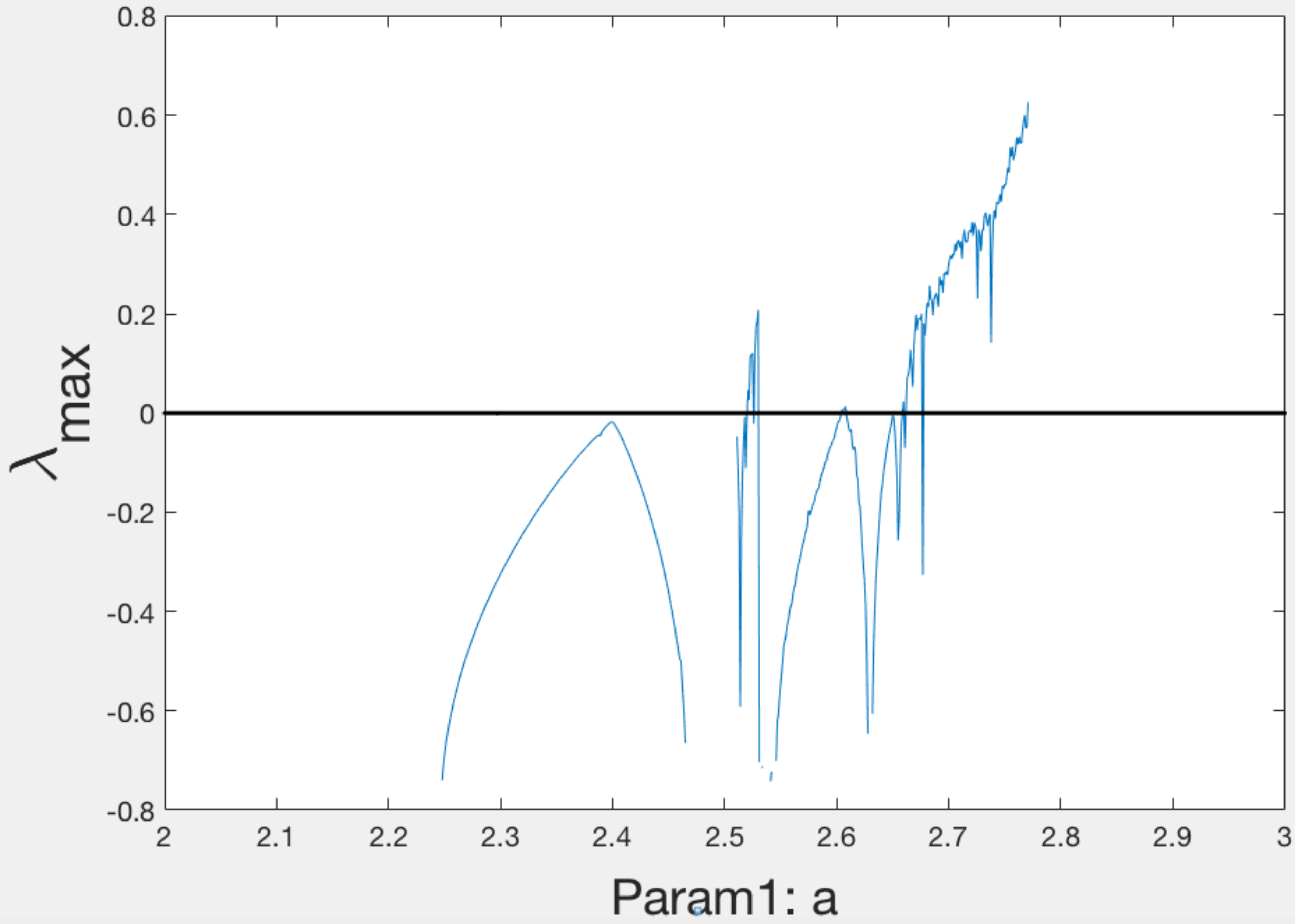
This time we have to deal with divergence in 2-dimensions.
To do that we use the Jacobian.

See Lecture 5 for our discussion of the Jacobian.

```
function max_lyapunovs = lyapunov2d(F, F_Jacobian, t_max, param1_range,…
                          param2, x0, y0)

current_l = 0;
for param1=param1_range
    current_l = current_l + 1;

    % Initialize variables
    xy = [x0; y0]; xy_lengths = [1;0];

    for i=1:t_max
        J = F_Jacobian(xy, param1, param2);
        xy=F(xy, param1, param2);

        % Calculate divergence rate in the direction defined by the Jacobian
        xy_lengths=J*xy_lengths;
        length=sqrt(sum(xy_lengths.^2)); % Distance formula
        max_lyapunovs(current_l) = log(length)/i; % Calculate the average
    end
end
end
```

```
F = @(xy,a, b) [xy(2); -b*xy(1)+a*xy(2)-xy(2)^3] % Duffling Map
F_Jacobian = @(xy,a, b) [0 1; -b a-3*(xy(2))^2] % Duffling Jacobian

max_time = 500; % How long to run (= number of samples to average)
parameter1_range = 2:0.001:3; % Range over parameter 1: (a for Duffling Map)
parameter2 = 0.2; % Fix parameter 2 (b for Duffling Map)

% Initial values for x and y
x0 = 0.5
y0 = 0.5

% Calculate the maximum Lyapunov exponents
max_lyapunovs = lyapunov2d(F, F_Jacobian, max_time, parameter1_range,…
                           parameter2, x0, y0);

% Make a plot of the maximum exponents with a line at 0
plot(parameter1_range,max_lyapunovs, parameter1_range, 0, 'k.')
xlabel('Param1: a', 'FontSize', 24);
ylabel('\lambda_{max}', 'FontSize', 24);
```

Example: Holmes Map



R ≈ 3 (2 period attractor)

R ≈ 3.449 (4 period)

R ≈ 3.544 (8 period)

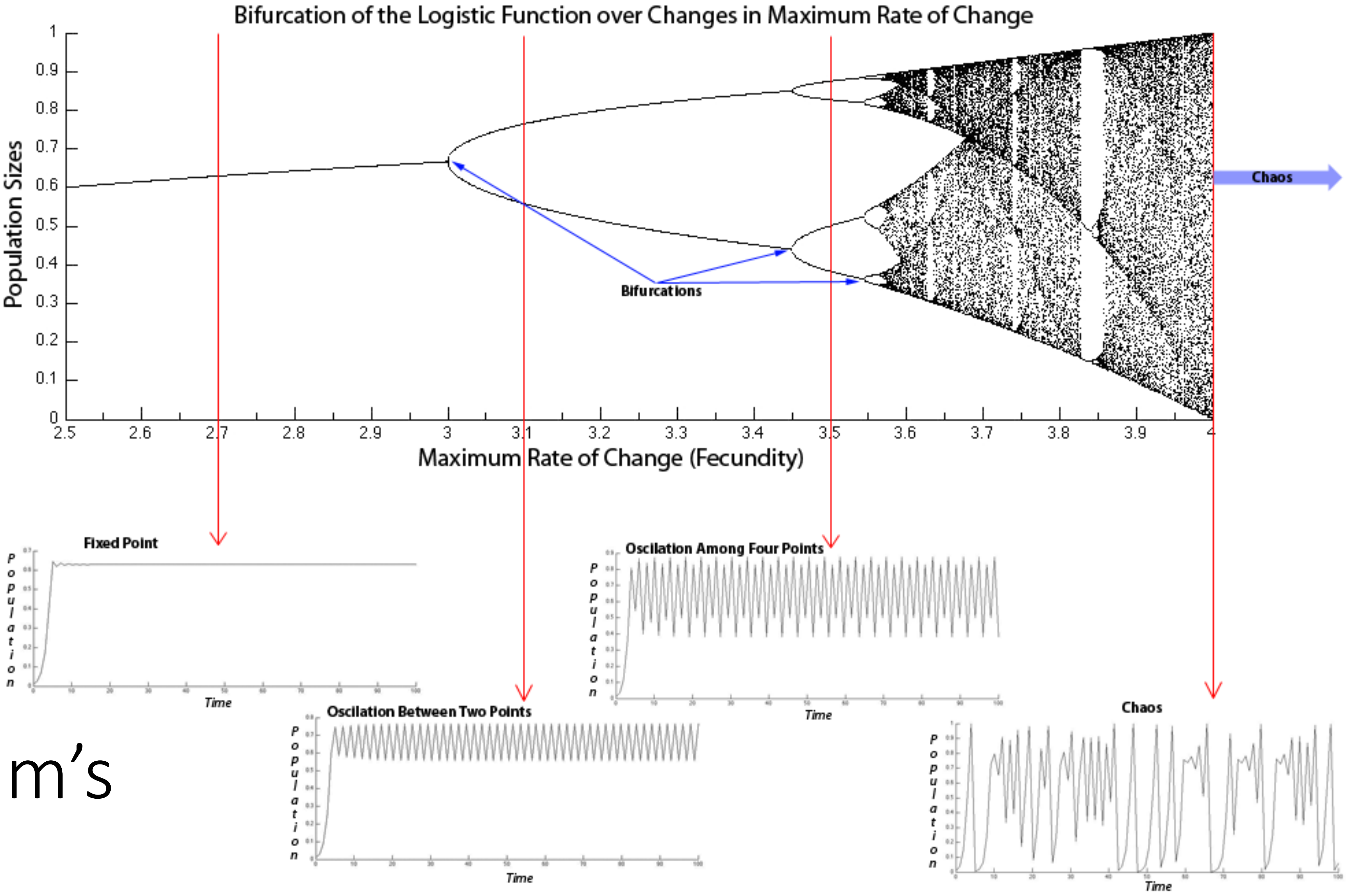
R ≈ 3.564 (16 period)

.

.

.

R ≈ 3.569 (∞ period)



Feigenbaum's Constant

Feigenbaum's Constant

$$\frac{R_1 - R_0}{R_2 - R_1} \approx 4.75$$

$$\frac{R_{n-1} - R_{n-2}}{R_n - R_{n-1}} \approx 4.669$$

Feigenbaum's Constant

True for ALL maps that approach chaos by bifurcation.

$$\frac{R_1 - R_0}{R_2 - R_1} \approx 4.75$$

$$\frac{R_{n-1} - R_{n-2}}{R_n - R_{n-1}} \approx 4.669$$

Phase Space Contraction

We can look at the evolution of

a small volume (in 3D) as it contracts near an attractor

The change in volume is measured by looking

at the trace of the Jacobian, $\vec{\nabla} \cdot \dot{\vec{x}}$

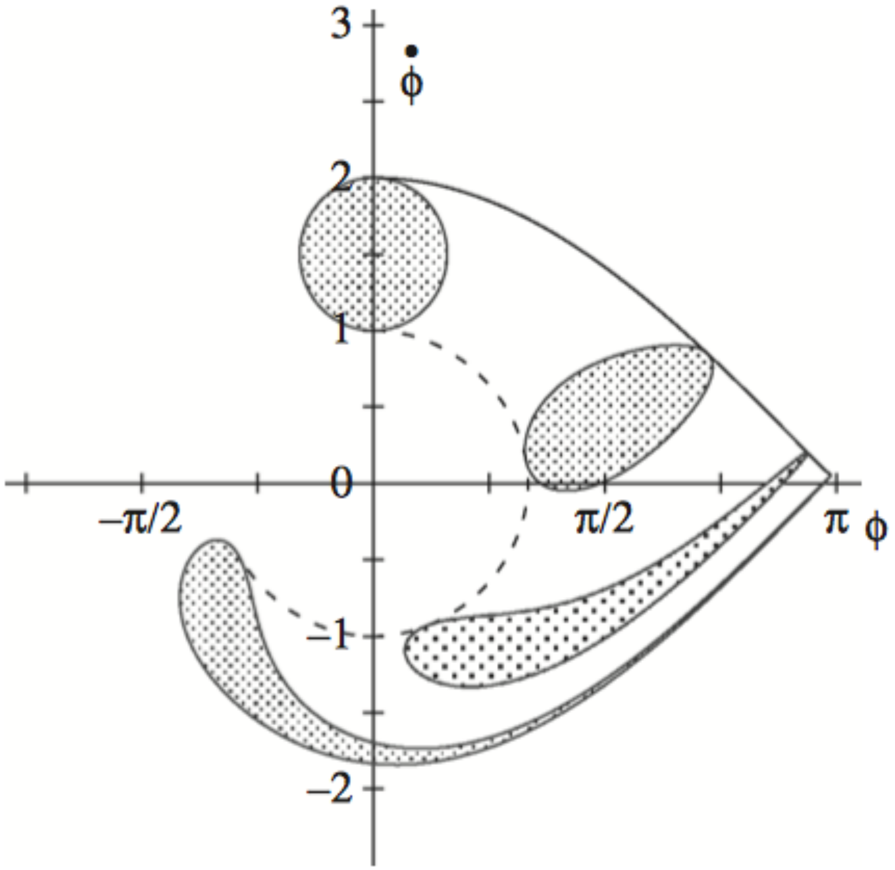
Dissipative vs Conserving Systems

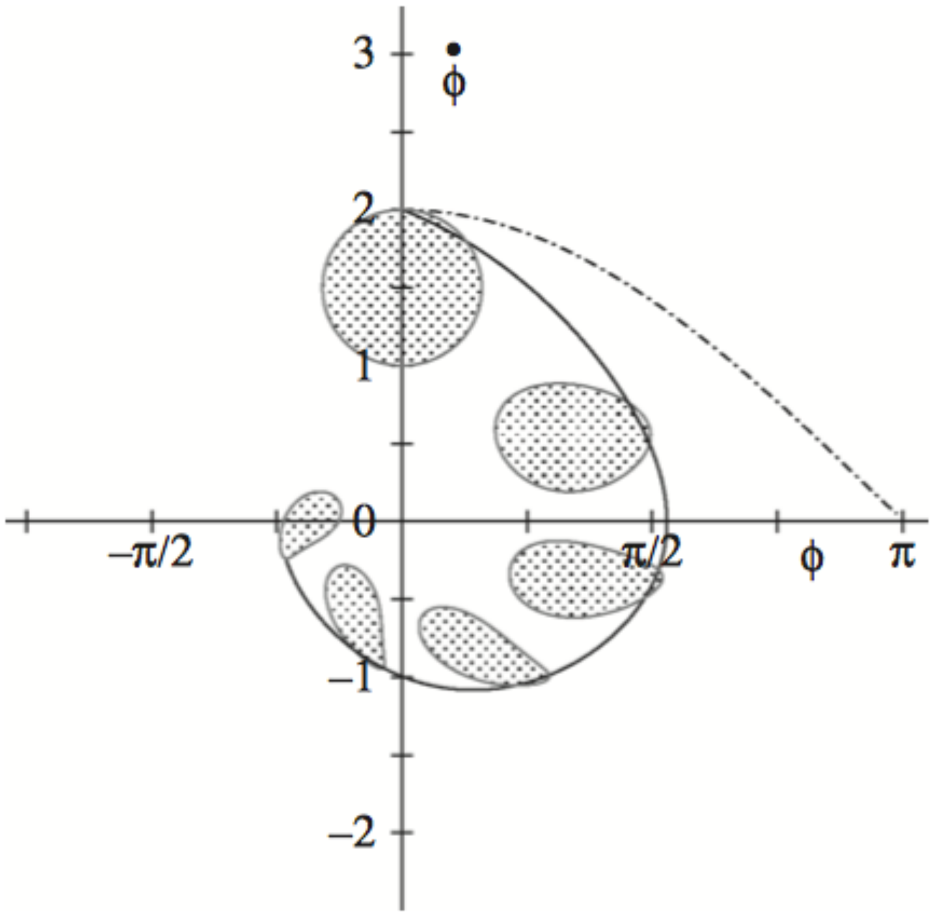
A dynamical system is dissipative,

if it's phase space volume contracts continuously,

$$\vec{\nabla} \cdot \dot{\vec{x}}(t) < 0, \forall t$$

Dissipative vs Conserving Systems

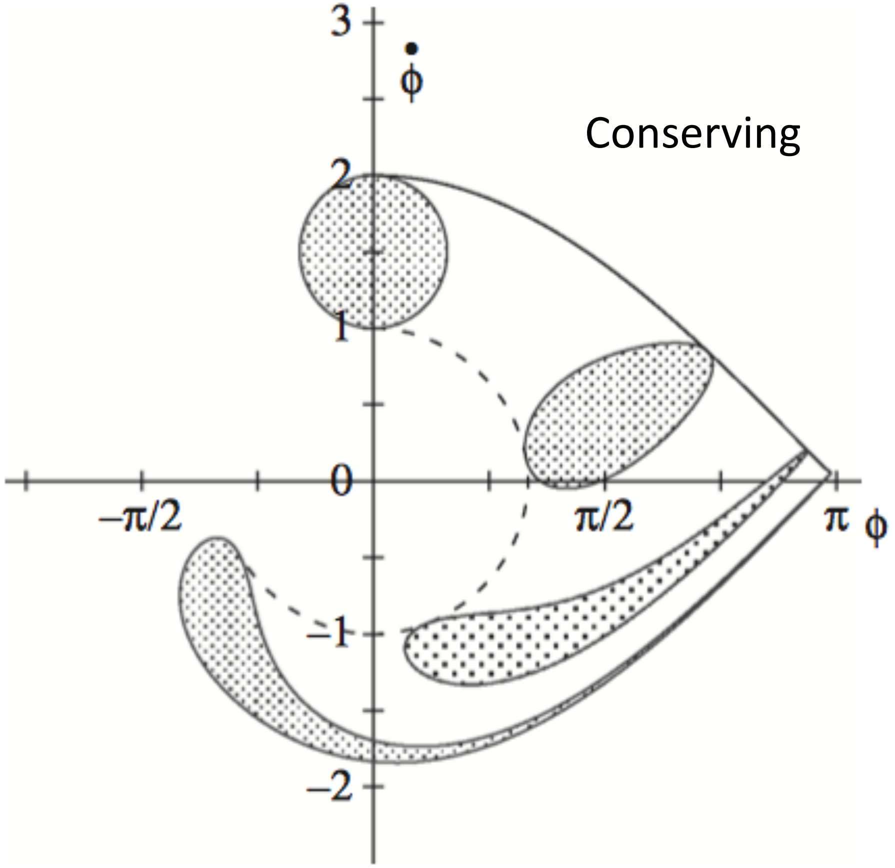


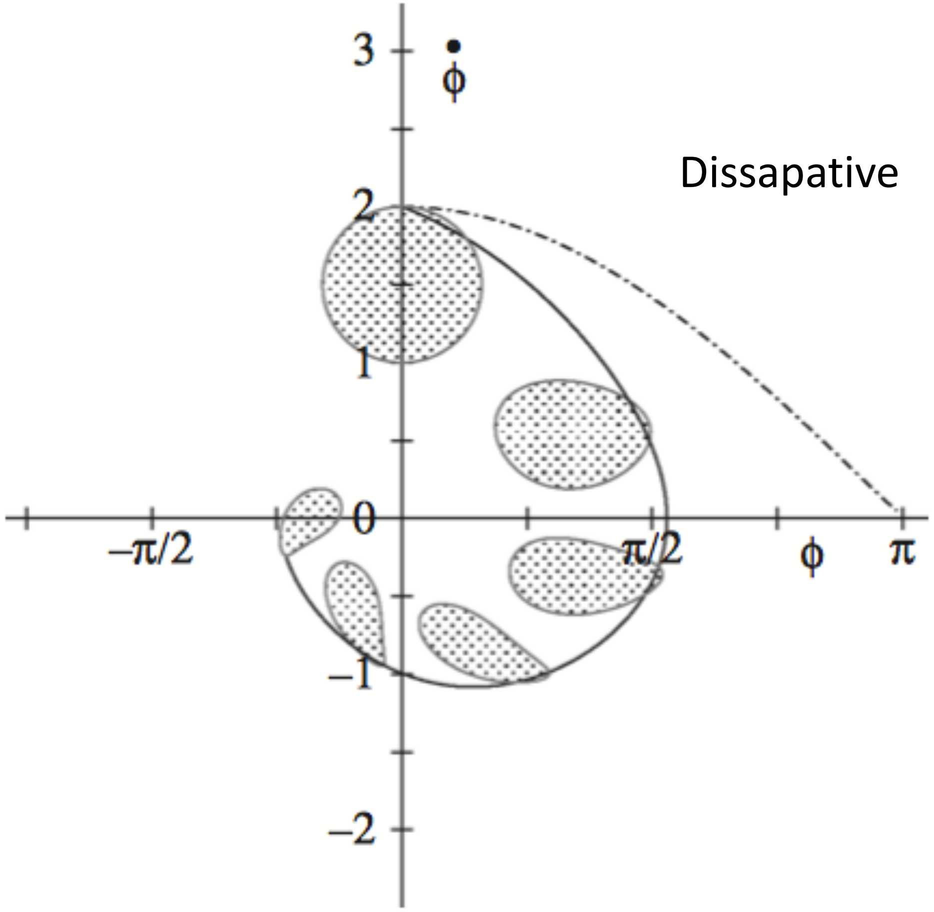


Dissapative or conservative?

Gros, C., Complex and Adaptive Dynamical Systems, 4th Edition

Dissipative vs Conserving Systems





Dissapative or conservative?

Gros, C., Complex and Adaptive Dynamical Systems, 4th Edition

Dissipative vs Conserving Systems

A dynamical system is conserving,
if it's phase space volume is constant,

$$\vec{\nabla} \cdot \dot{\vec{x}}(t) = 0, \forall t$$

Adaptive Systems

Adaptive systems are neither fully dissipative nor fully conserving.

In terms of energy they have periods of taking up energy and periods of expending it.

Technically, any system where $\vec{\nabla} \cdot \dot{\vec{x}}(t)$ can change sign is adaptive.

Outline

- Feigenbaum Constant (Order -> Chaos -> Deeper Order)
- Kuhns Revolutions in Science
  - Microscope
  - Telescope
  - Computer (George Luger)
  - May, Lorenz, Crutchfield, Mitchell and many others…
- Biological Complexity from Simple Rules. Hearts (fractal structure)

- Exploring Dynamical Systems has demonstrated that surprising complexity can arise from simple rules.
- This allows us to understand the diversity generated by things like capitalism, biological evolution, computer architectures, and computer algorithms.