Scientific Computation

Spring, 2019

Lecture 18

Notes

- Corrections for HW3 posted yesterday

- Test data for Q2.2 will be provided this evening (blackboard announcement)

Linear data analysis

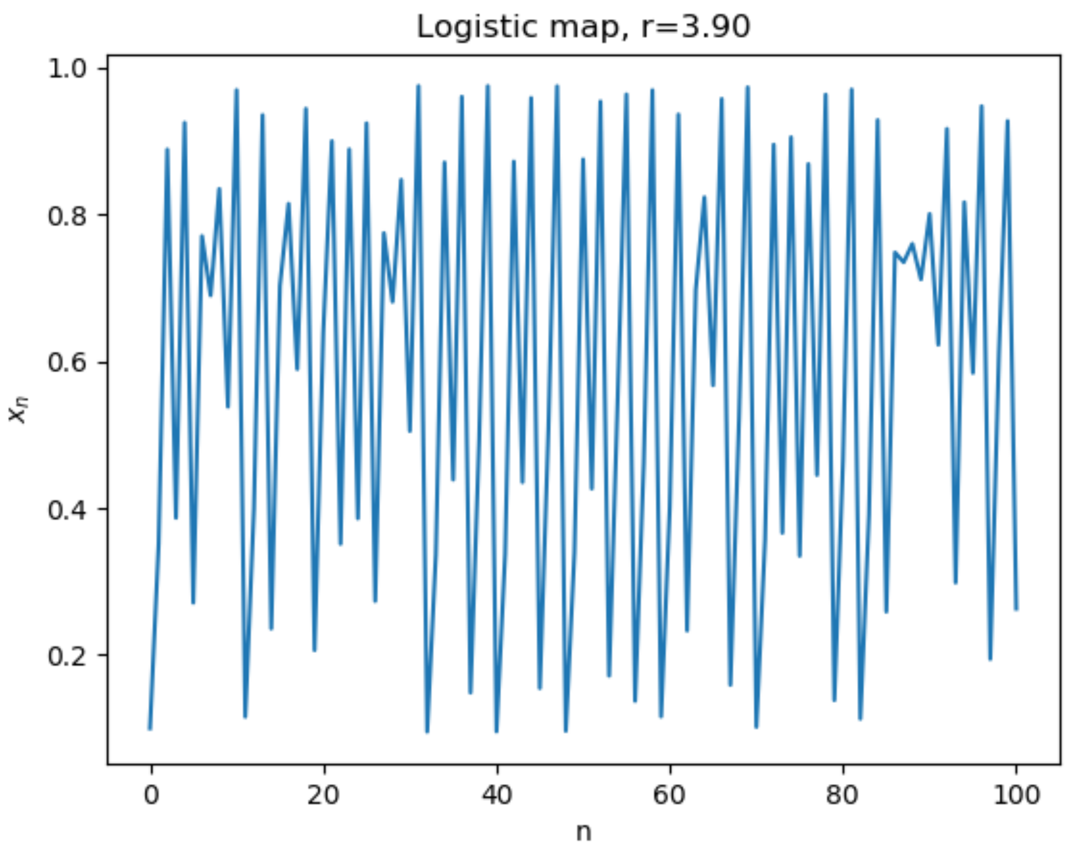
We have looked at a few linear methods:

- **DFT: superposition of sinusoidal oscillations**
- **PCA: linear transformation**
- **Many other decompositions and "tricks" from linear algebra can be useful**
  - **E.g. any matrix can be decomposed into a sum of symmetric and skew-symmetric matrices:** $A = \frac{1}{2}\left(A + A^T\right) + \frac{1}{2}\left(A - A^T\right)$

- **However, given that "complex systems" are typically nonlinear, we need to think about customized methods**
  - **Nonlinear PCA is one example**
  - **We will focus on computing the fractal dimension**

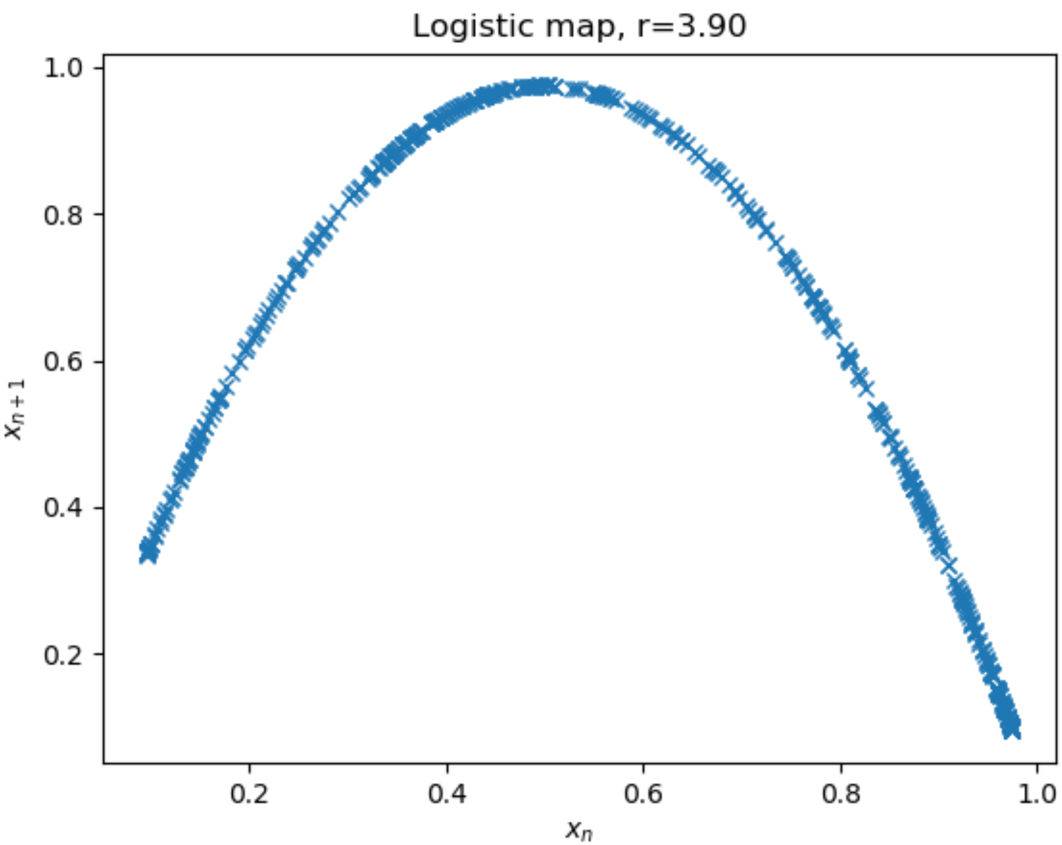
Last time

- Fourier (energy or power) spectra are a good place to start with stationary data
  - If the length of the data is sufficiently large
  - And the sampling rate sufficiently large (Δx or Δt sufficiently small)



- What can we do with this data?
- FFT won't work – data is not smooth
- Three basic options:
  - Compare peaks to peaks
  - Or troughs to troughs
  - Or peaks to troughs
- Let's try the 3rd option

Logistic map



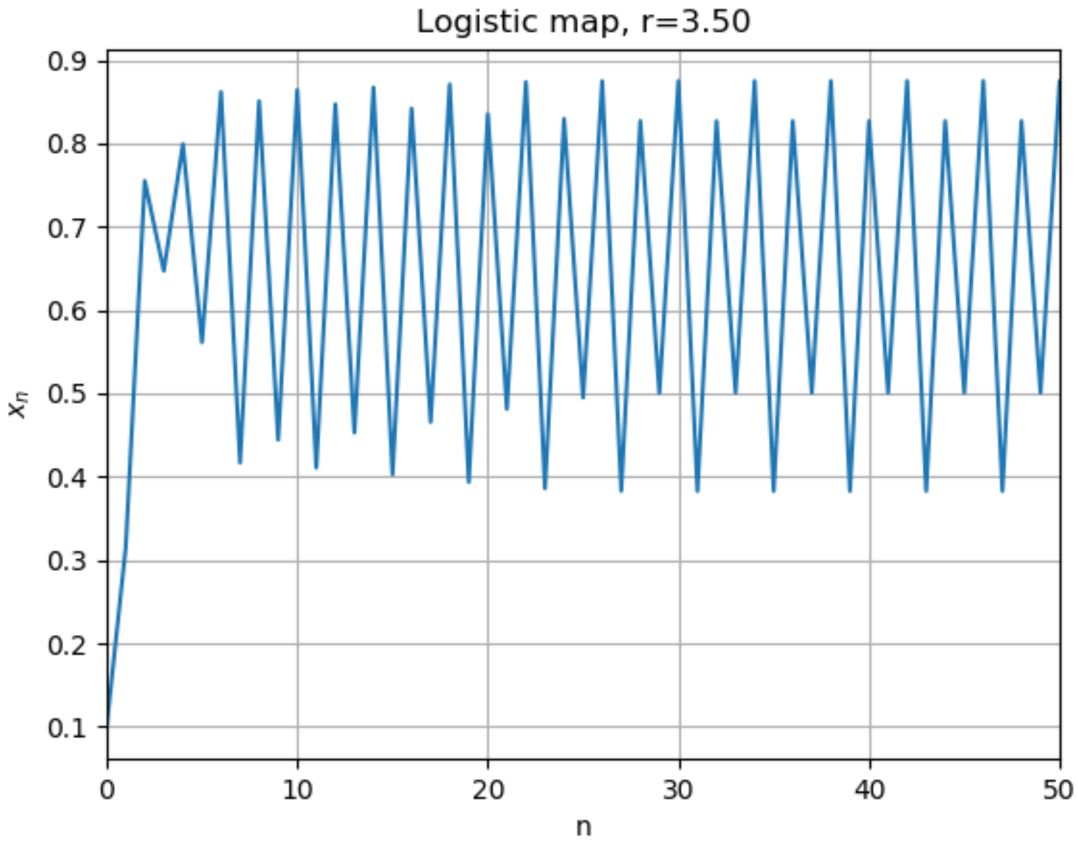

```
#Main calculation
for i in range(N):
    x[i+1] = r*x[i]*(1-x[i])
```

- **There is clear "structure" within the data!**
- **The *logistic map* is:**

  $$x_{n+1} = rx_n\left(1 - x_n\right)$$

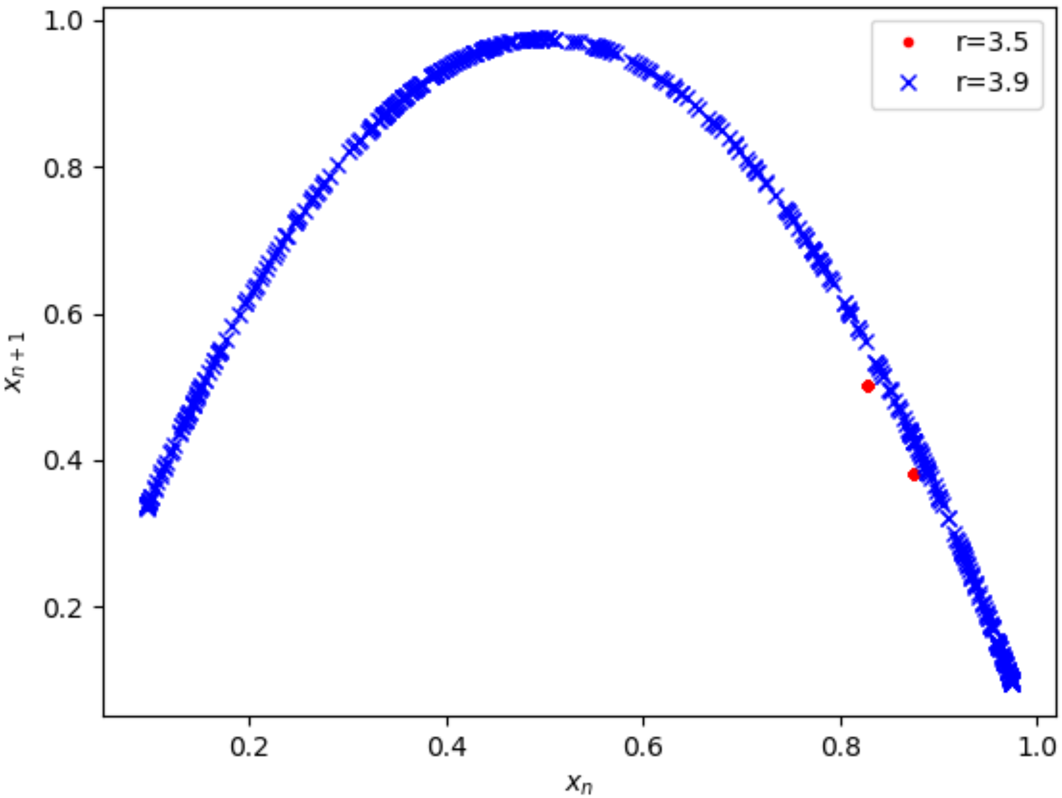
- **The parameter, r, controls the dynamics.**
  - **For smaller r, simple periodic behavior**
  - **For r=3.9, chaotic behavior**
  - **As time increases, more and more points on the parabola will be visited in an irregular order**

Logistic map



- At r=3.5, there are *period-2* oscillations
- $x_{n+4} = x_n$
- When characterizing a solution, we need to think about the set of points visited (after discarding the initial transient)
- And in particular, the *dimension* of this set

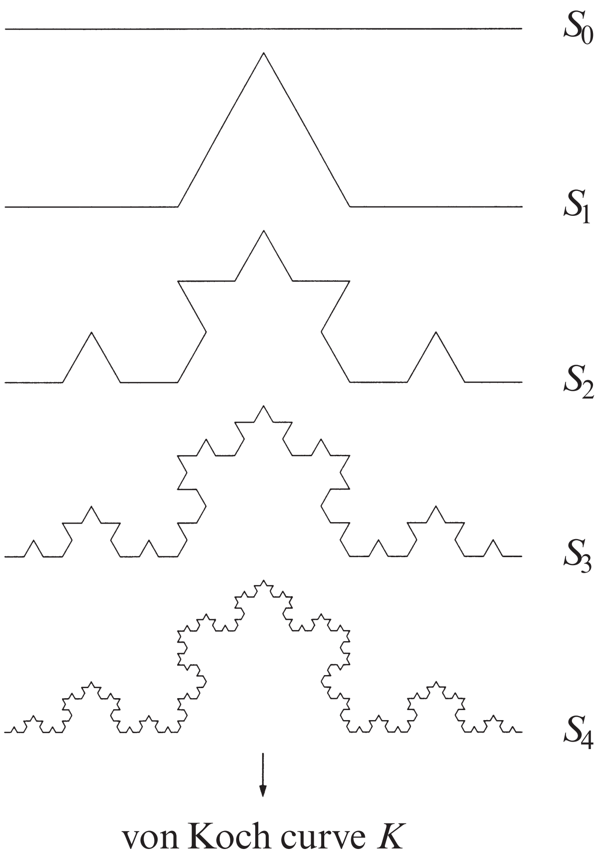
Logistic map



- **At** r=3.5**, there are *period-2* oscillations**
- $\mathbf{x_{n+4} = x_n}$
- **When characterizing a solution, we need to think about the set of points visited (after discarding the initial transient)**
- **And in particular, the *dimension* of this set**

Fractal dimension

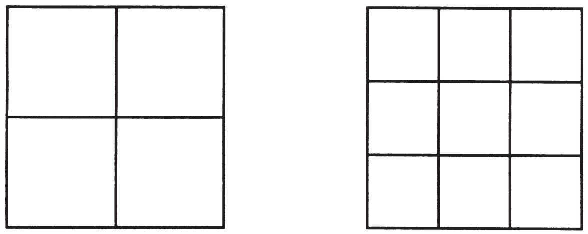
- We borrow from the study of fractals where there is a ***similarity*** **or** ***fractal*** **dimension**



- **An example: the Koch snowflake**
  - **Each "iteration", the "curve" is broken up into 4 smaller copies of itself. The length of each copy is 1/3 of the original**
  - **What is the dimension of this curve?**

Fractal dimension

- We borrow from the study of fractals where there is a *similarity* or *fractal* dimension



$m = 4$
$r = 2$

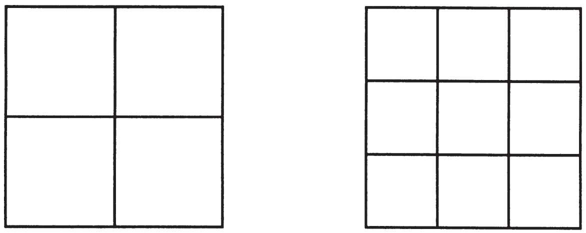
$m = 9$
$r = 3$

$m =$ number of copies
$r =$ scale factor

- A simpler example: A square.
  - Break a square into 4 smaller copies (m=4) – the sides of each copy have been scaled down by a factor of 2 (r=2)
  - With m=9, we have r=3
  - And the dimension is d = log(m)/log(r)=2
  - For the Koch snowflake, m=4, r=3, d=log(4)/log(3)=1.261…

Fractal dimension

- We borrow from the study of fractals where there is a *similarity* or *fractal* dimension

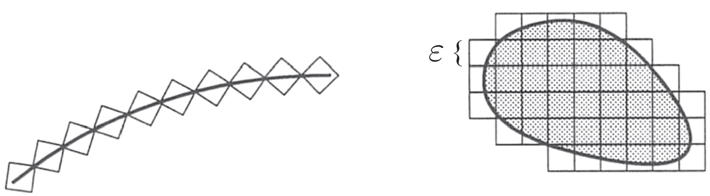


$m =$ number of copies
$r =$ scale factor

$m = 4$, $r = 2$ $\qquad$ $m = 9$, $r = 3$

How do we compute the fractal dimension given data?

One approach is to compute the *box dimension*:



$$N(\varepsilon) \propto \frac{L}{\varepsilon} \qquad\qquad N(\varepsilon) \propto \frac{A}{\varepsilon^2}$$

Construct m-dimensional cubes with edge-length, ε, find the number of cubes needed to cover the set of points in the solution, N(ε)

We expect: $N(\varepsilon) \propto 1/\varepsilon^d$. where d is the box dimension

Fractal dimension

- **In practice, the box dimension is not used – it's computation is too expensive for large high-dimensional sets**
- **The *correlation dimension* is often used instead**
- **We collect** n m**-dimensional points "visited" during a process after discarding the effect of the initial condition:** $\{\mathbf{x}_i, i = 1, \ldots, n\}$
- **The *correlation sum*,** C(ε) **is: (total number of pairs of points within distance** ε**)/(Total number of distinct pairs)**

$$C(\epsilon) = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} H\left(\epsilon - ||\mathbf{x}_i - \mathbf{x}_j||\right)$$

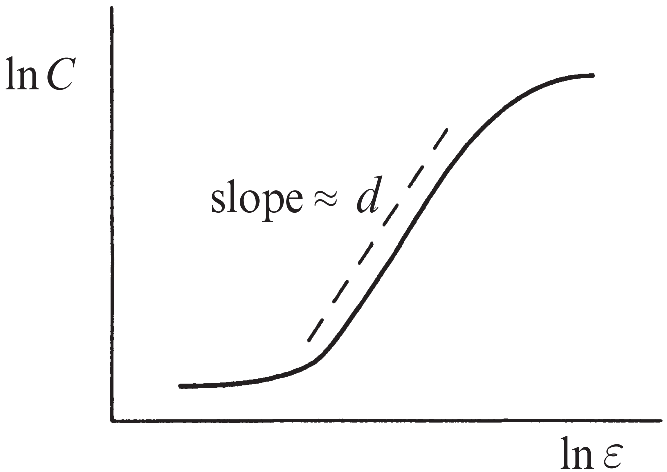
- **Here,** H, **is the Heaviside function,** H(z)=0 if z<=0, H(z)=1 if z>0
- $||\mathbf{x}_i - \mathbf{x}_j||$ **is the Euclidian distance (for m-dimensional x)**
- **For points falling on a fractal-like structure, we expect:** $C(\epsilon) \sim \epsilon^{D}$

Fractal dimension

- **The *correlation sum*, C(ε) is: (total number of pairs of points within distance ε)/(Total number of distinct pairs)**

$$C(\epsilon) = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} H\left(\epsilon - ||\mathbf{x}_i - \mathbf{x}_j||\right)$$

Generically, we expect:



- At small ε, few if any pairs within ε of each other

- At large ε, almost all pairs within ε

- Often some trial-and-error is needed to choose a good range of ε

- **Straightforward to compute for logistic map using** scipy.spatial.distance.pdist

Imperial College
London

Fractal dimension

1. **Compute solution:**

```
for i in range(N):
    x[i+1] = r*x[i]*(1-x[i])
```

2. **Discard influence of initial condition:**

```
y = x[N//2:]
n = y.size
```

3. **Split into two vectors** (m=2) **and collect in** n x m **matrix**

```
y1 = y[:-1:2]
y2 = y[1::2]
A = np.vstack([y1,y2]).T
```

4. pdist **will then compute all** n(n-1)/2 **distances:**

```
D = pdist(A)
```

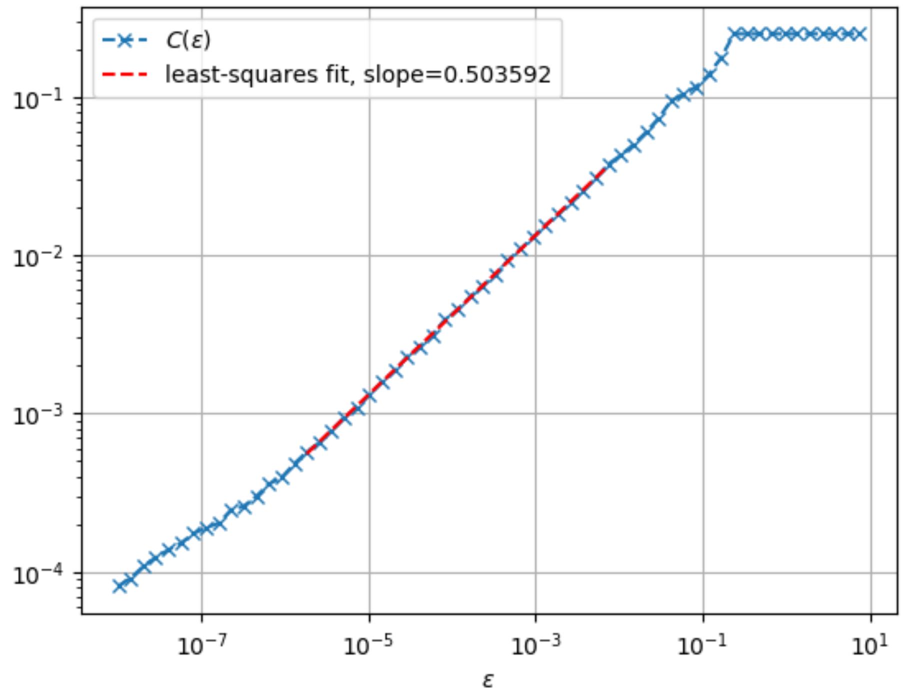
5. **Now just need to pass these distances through Heaviside function for a range of** ε…

Fractal dimension

5. Now just need to pass these distances through Heaviside function for a range of ε…

```
D = D[D<eps[i]] #Discard distances larger than eps. Assumes eps[i+1]<eps[i]

C[i] = D.size #Size of new D is Correlation sum (without m*(m-1)/2 scaling)
```



- **Results for r=3.5699456 are shown, m=8000**
- **1st 15 and last 20 points have been discarded for best fit calculation, fractal dimension is estimated as 0.5 (for this value of r)**
- **Estimate computed using** np.polyfit

Can construct a range of epsilons using np.logspace

Lorenz system

A (famous) example:

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = rx - y - xz$$

$$\frac{dz}{dt} = xy - bz$$

- σ, b**, and** r **are parameters. In certain parameter ranges, chaotic dynamics are generated**
- **Initially developed in 1960s as model for atmopheric flows**
- **Modern field of chaotic dynamics emerged from the study of these equations**
- **Can integrate these equations using** odeint…

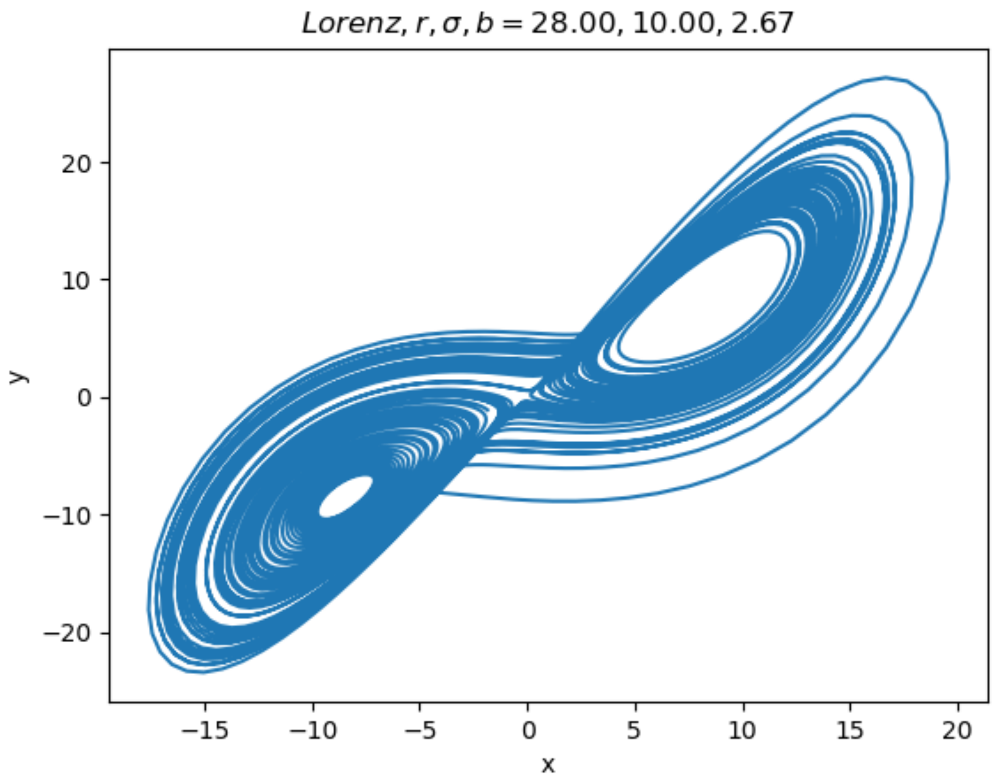
Lorenz system

A (famous) example:

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = rx - y - xz$$

$$\frac{dz}{dt} = xy - bz$$



- **These are trajectories in the x-y plane (a phase plane)**
- **It looks like trajectories are crossing (suggests periodic dynamics), but they are in fact avoiding each other…**

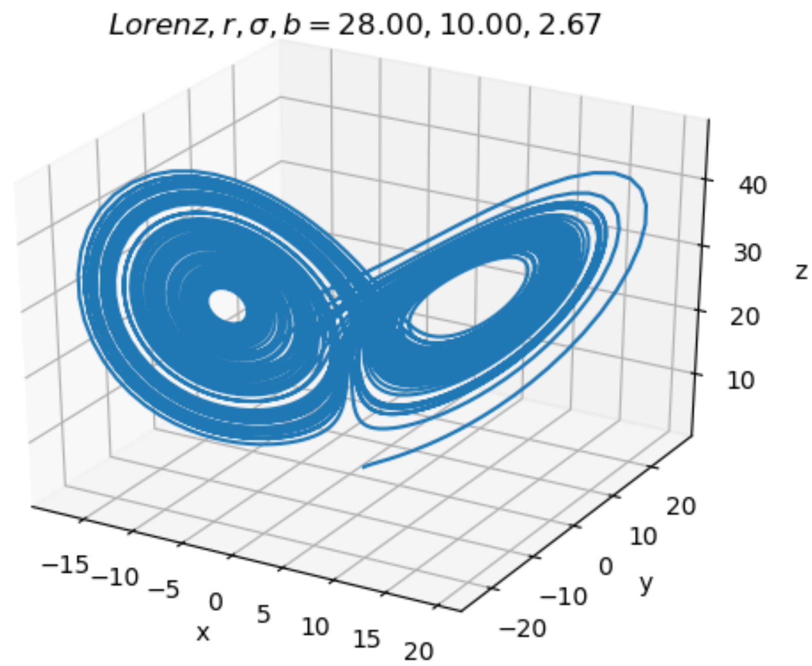
Lorenz system

A (famous) example:

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = rx - y - xz$$

$$\frac{dz}{dt} = xy - bz$$



- These are trajectories in the x-y plane (a phase plane)
- It looks like trajectories are crossing (suggests periodic dynamics), but they are in fact avoiding each other…

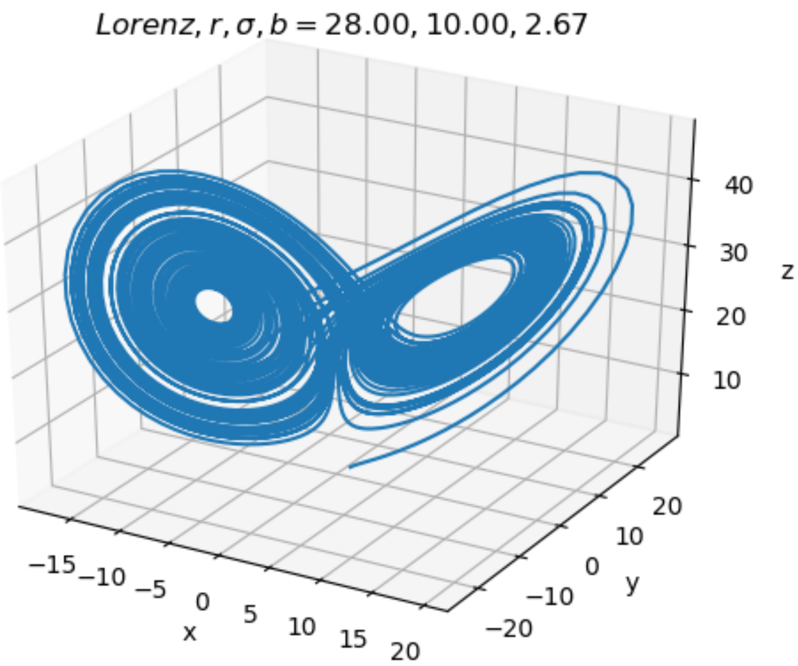
Lorenz system

A (famous) example:

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = rx - y - xz$$

$$\frac{dz}{dt} = xy - bz$$



- **We can again compute a correlation sum with n three-element vectors, [**$x(t_i),y(t_i),z(t_i)$**] (m=3) each of which corresponds to a point on the plot above**
- **Using** pdist **after discarding points for t<10…**

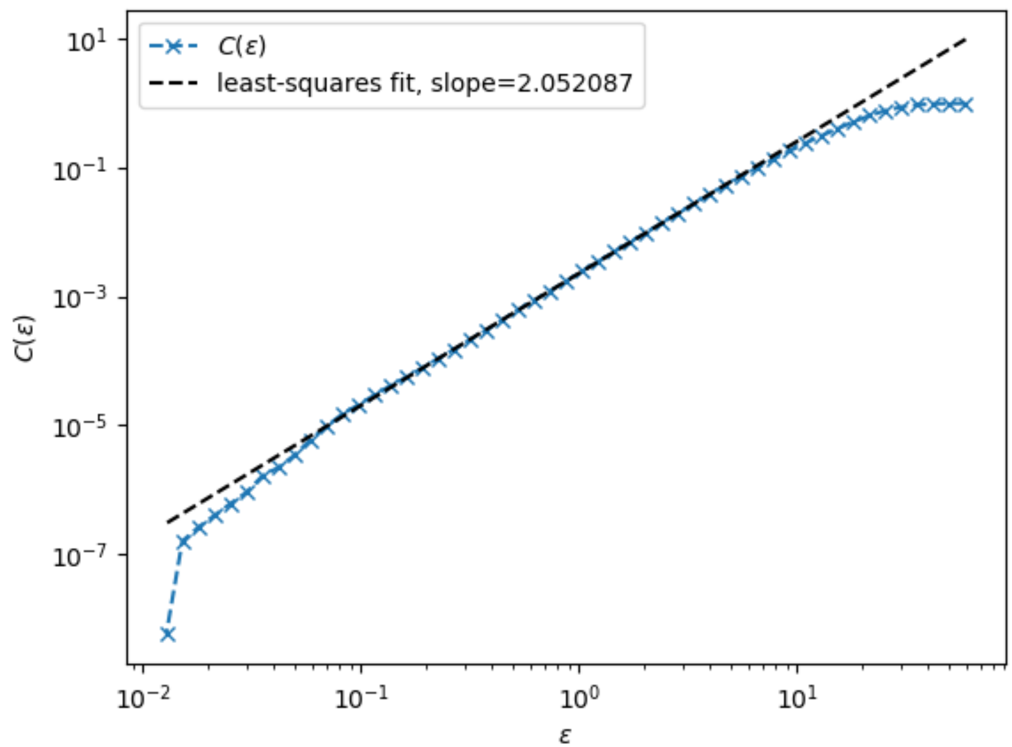
Lorenz system

A (famous) example:

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = rx - y - xz$$

$$\frac{dz}{dt} = xy - bz$$



- **The dimension is ~2.05 – this is greater than 2 as we expect and indicates that the dynamics are weakly chaotic**

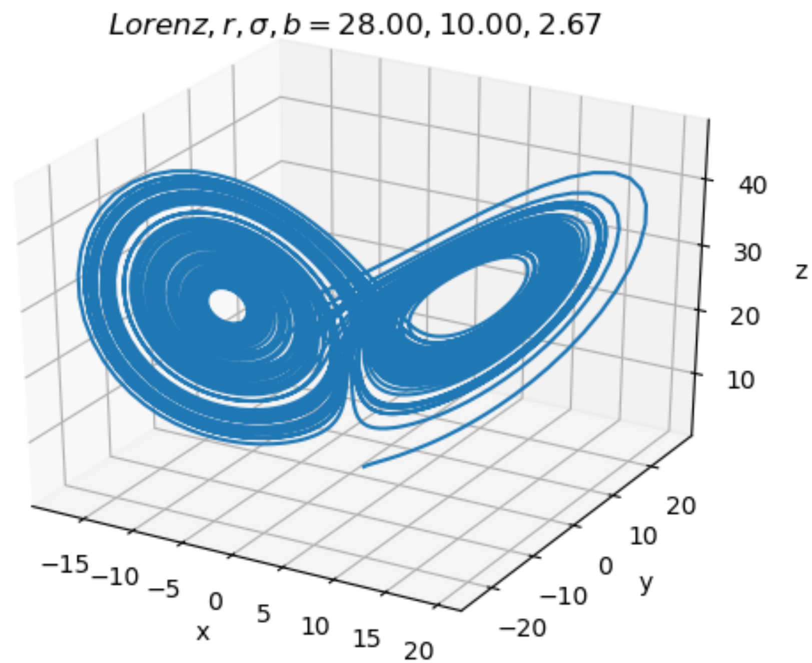
Attractor reconstruction

A (famous) example:

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = rx - y - xz$$

$$\frac{dz}{dt} = xy - bz$$



- **What do we do if we have a single (chaotic) time series?**
- **Or if we're working with a PDE?**

Attractor reconstruction

- **What do we do if we have a single (chaotic) time series?**

- **Construct an m-dimensional vector at time, t_i, using time delays:**

$$\mathbf{v}_i = [x(t_i), x(t_i - \tau), x(t_i - 2\tau), ..., x(t_i - (m-1)\tau)]$$

- **How do we choose m?**
  - **Should be larger than the fractal dimension (but not too much larger!)**
  - **Typically requires some manual adjustment/iteration**

Attractor reconstruction

- What do we do if we have a single (chaotic) time series?

- Construct an m-dimensional vector at time, t_i, using time delays:

$$\mathbf{v}_i = [x(t_i), x(t_i - \tau), x(t_i - 2\tau), ..., x(t_i - (m-1)\tau)]$$

- How do we choose m?
  - Should be larger than the fractal dimension (but not too much larger!)
  - Typically requires some manual adjustment/iteration

- How do we choose τ?
  - Again, typically requires some iteration
  - Often, a system has a dominant time scale (e.g. time to do a loop on the Lorenz attractor)
  - Something like 1/5 of this time scale is a good place to start
- Lab 9 considers these issues

- Time delays are often the best approach for results from PDEs