Random Walks

Random Walks

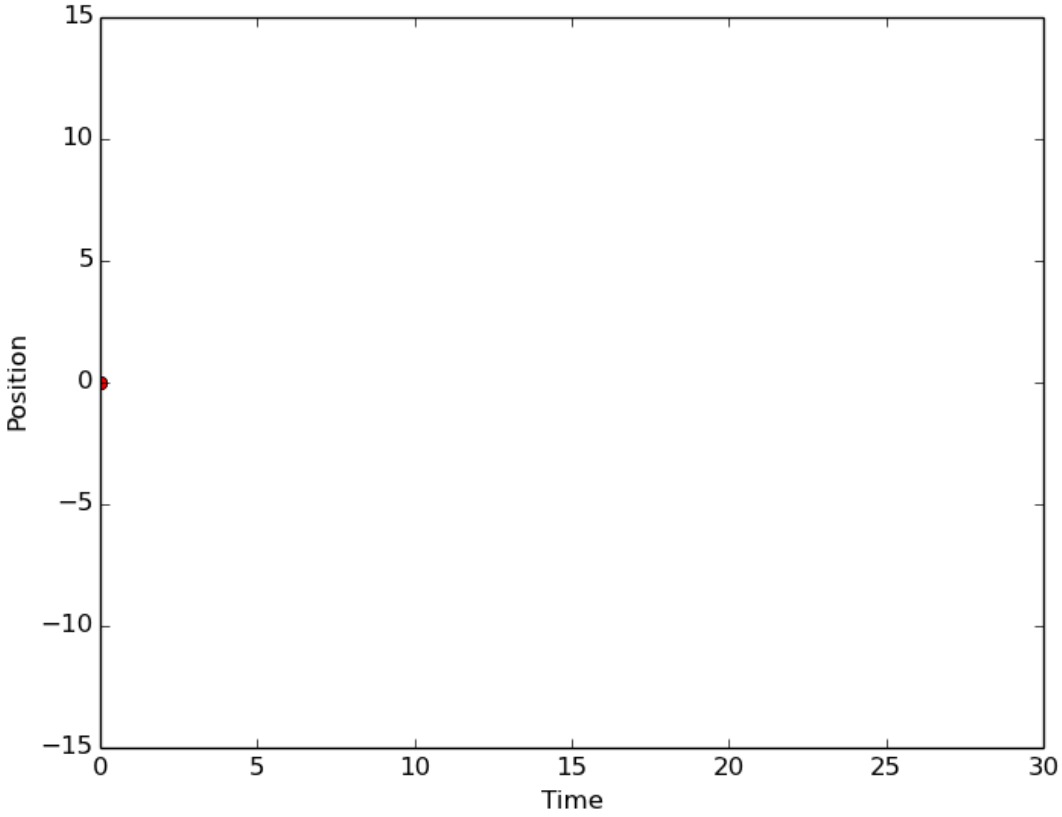
- A random walk is a trajectory made by taking successive random steps
  - Random means direction of steps is uncorrelated
- These processes occur in many systems with:
  - History / memory (integration of change)
  - Randomness
- Biology, Physics, Mathematics, Finance

1D Random Walker

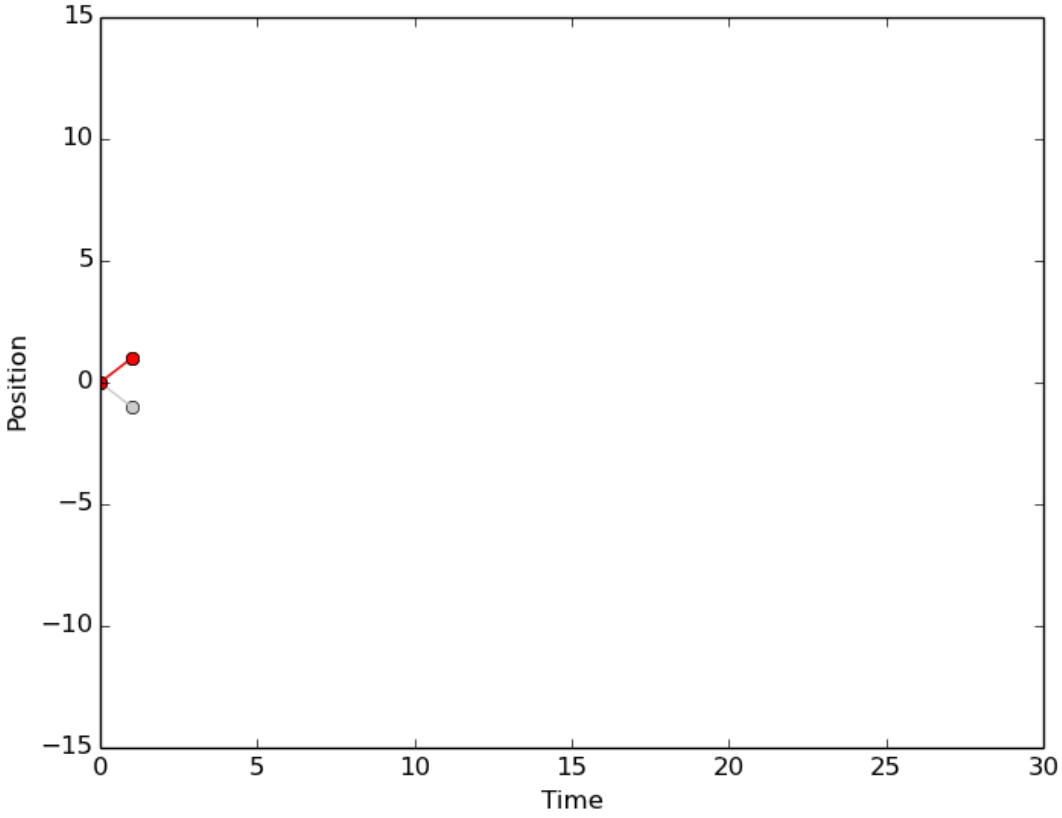
- As simple as it gets

- At fixed intervals we move either up or down by 1 unit with a 50/50 probability

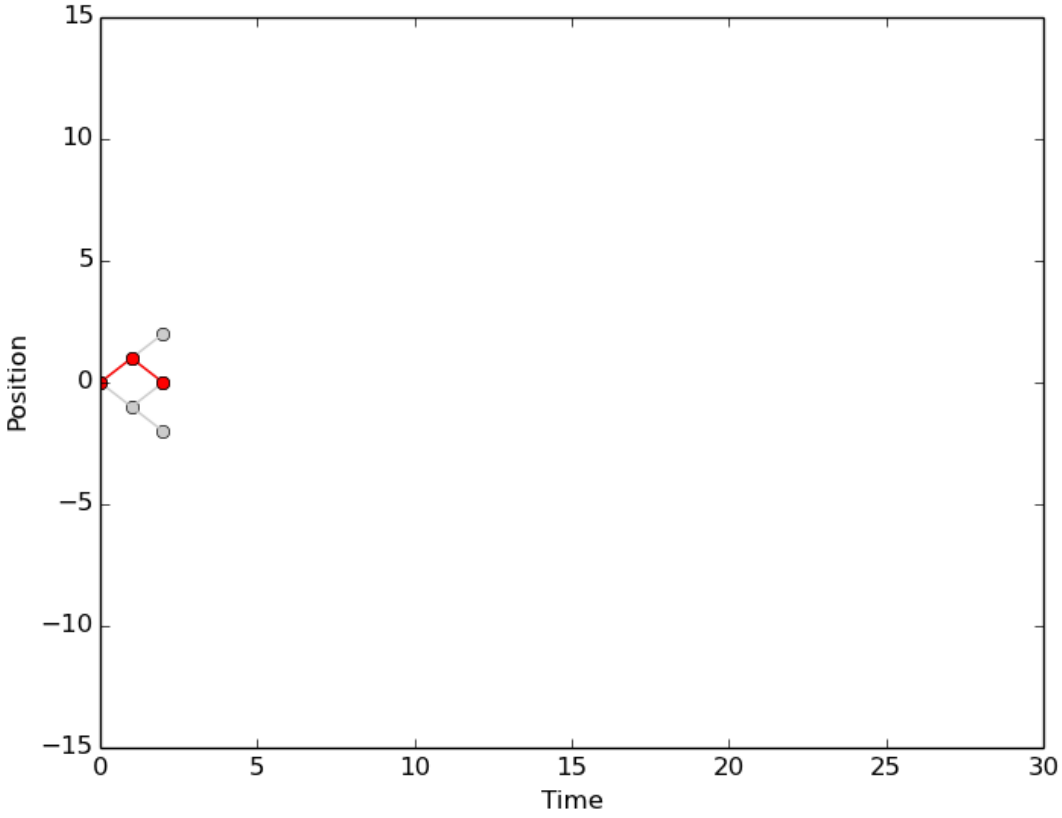
1D – example



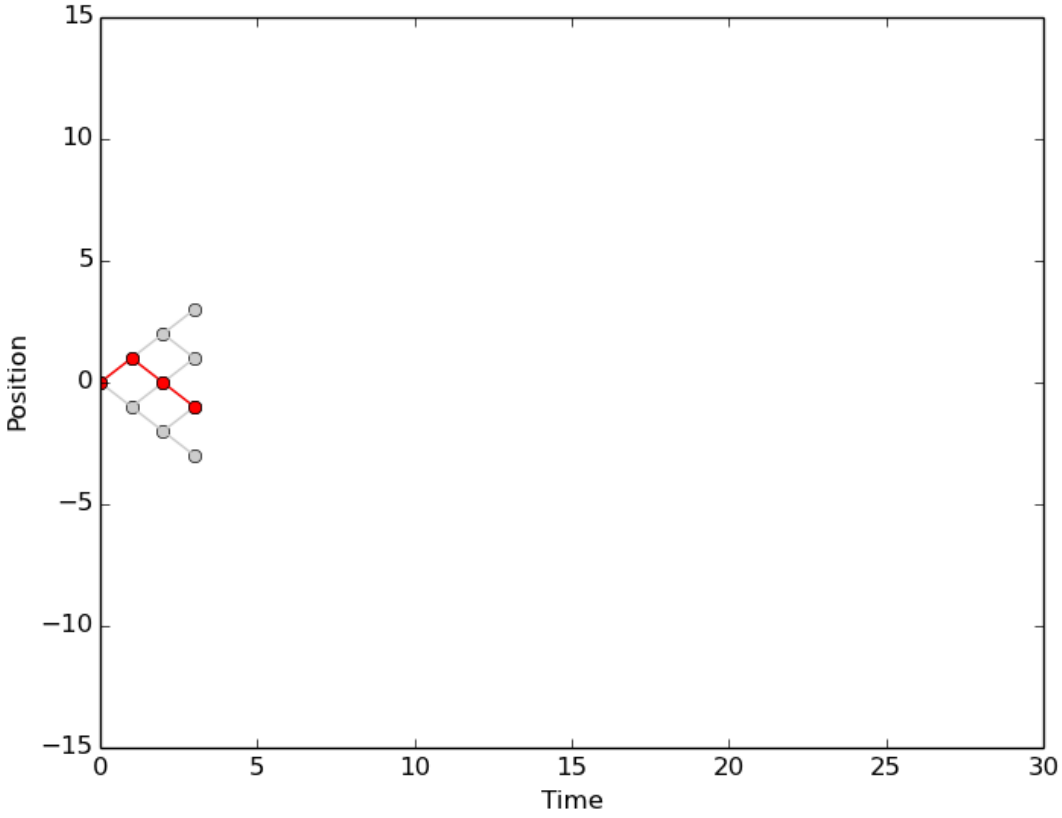
1D – example



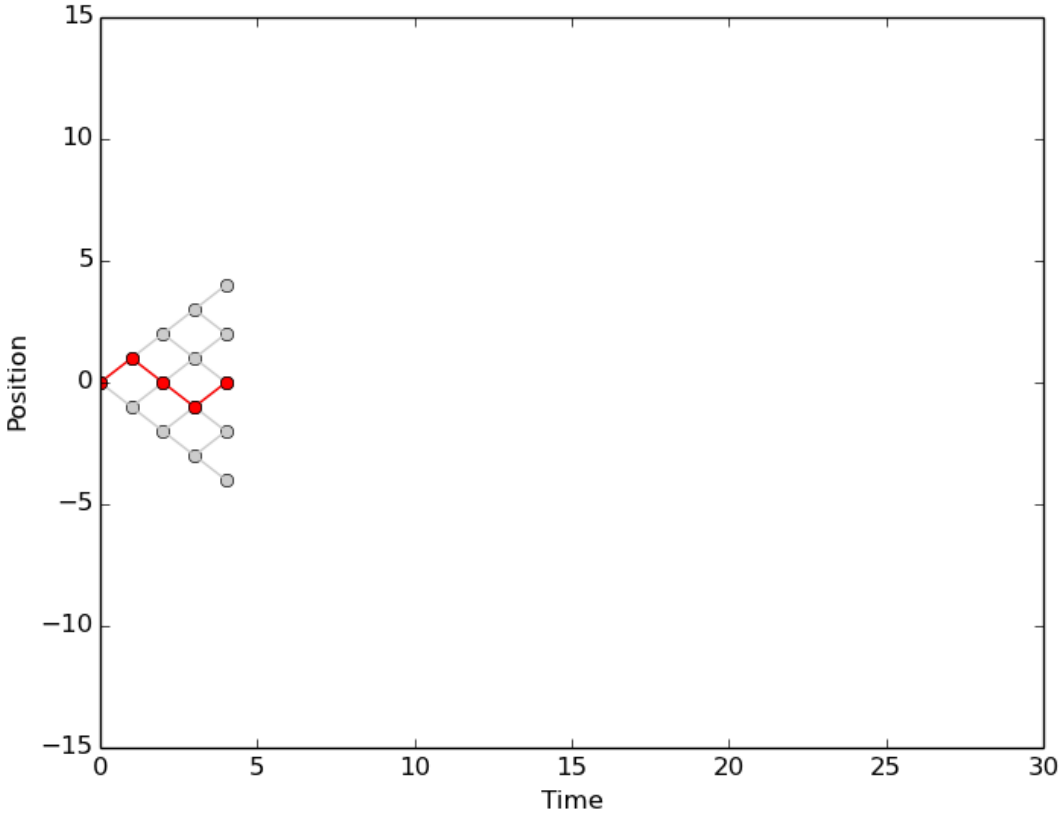
1D – example



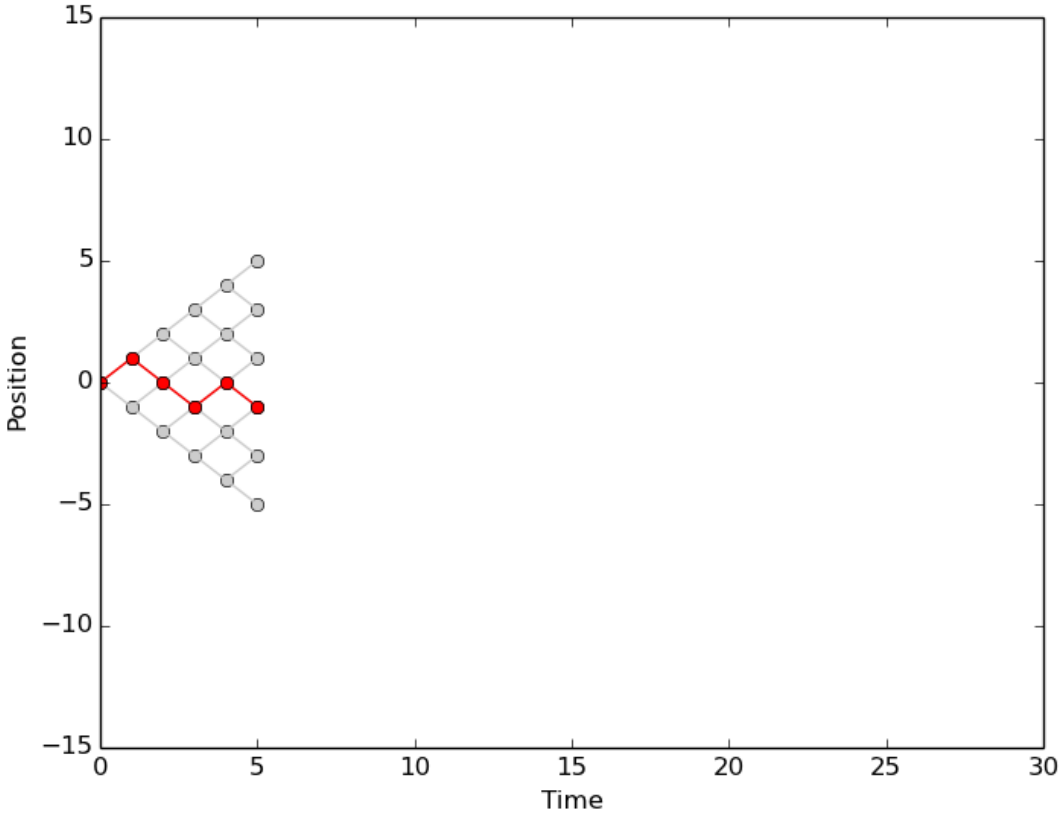
1D – example



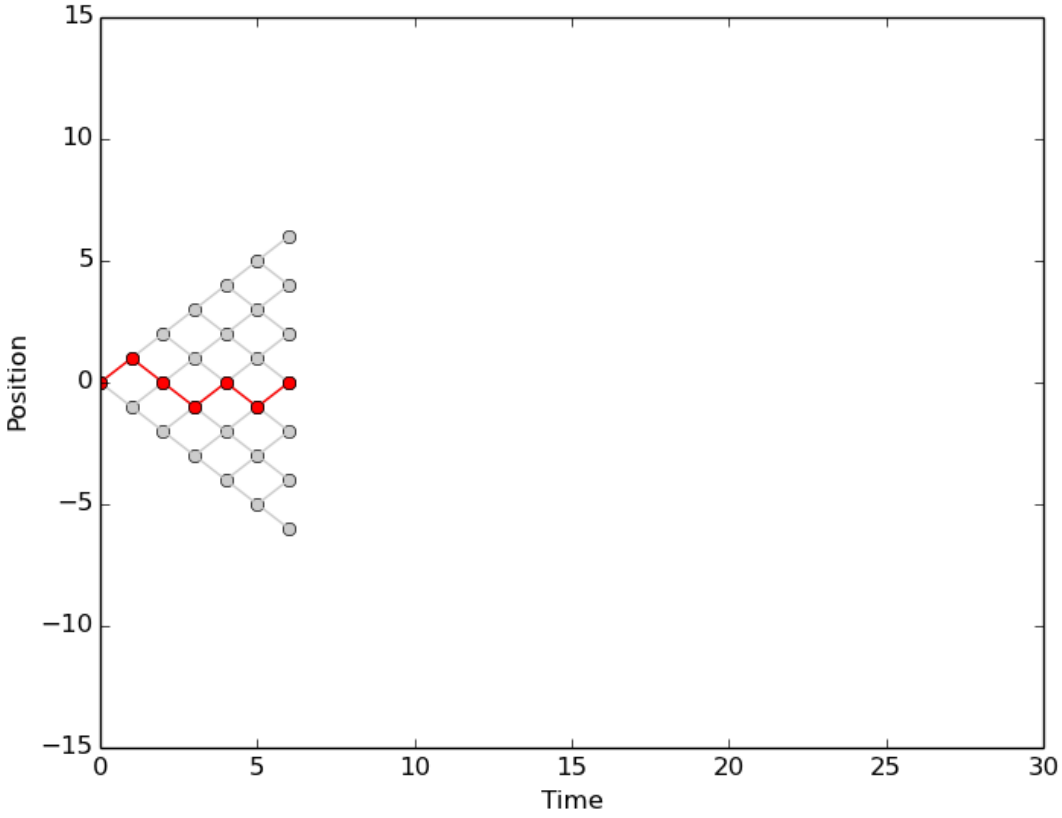
1D – example



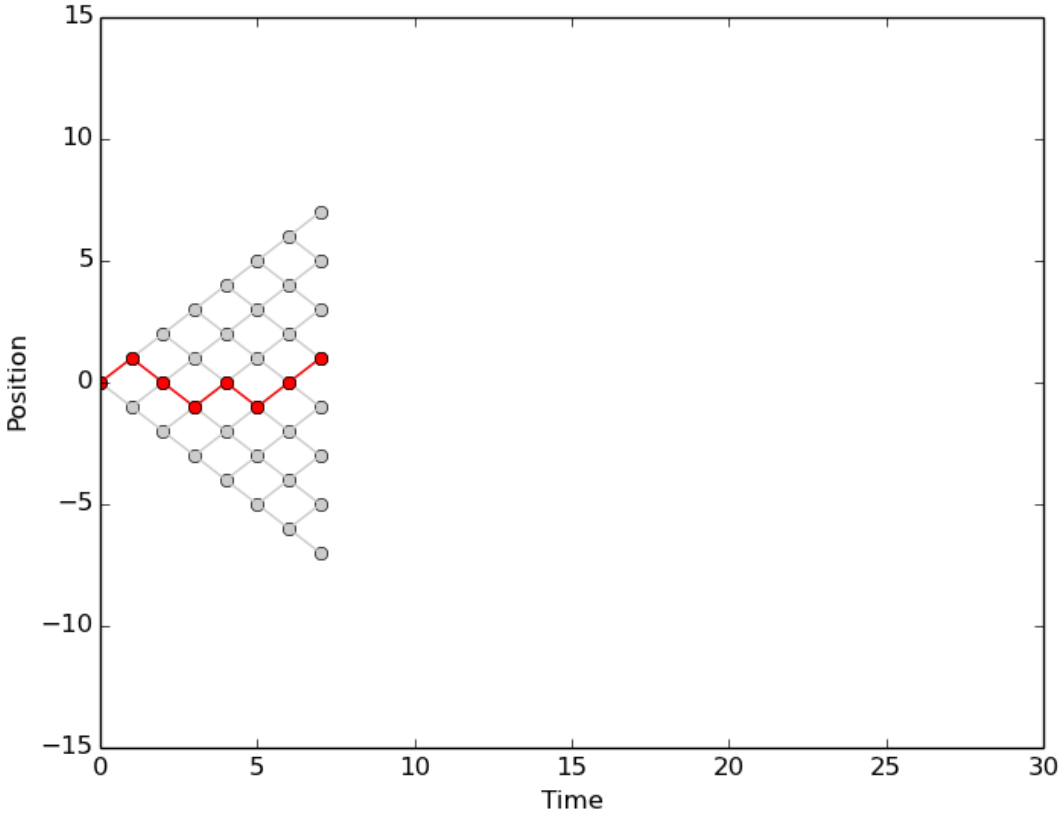
1D – example



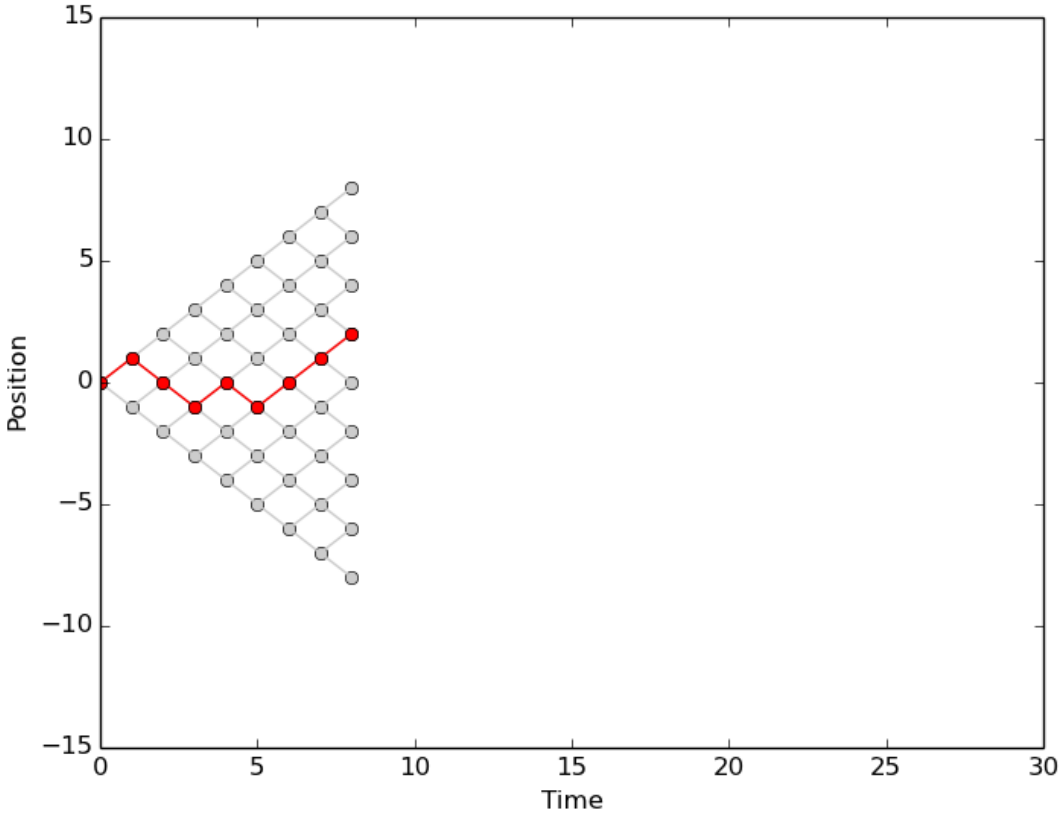
1D – example



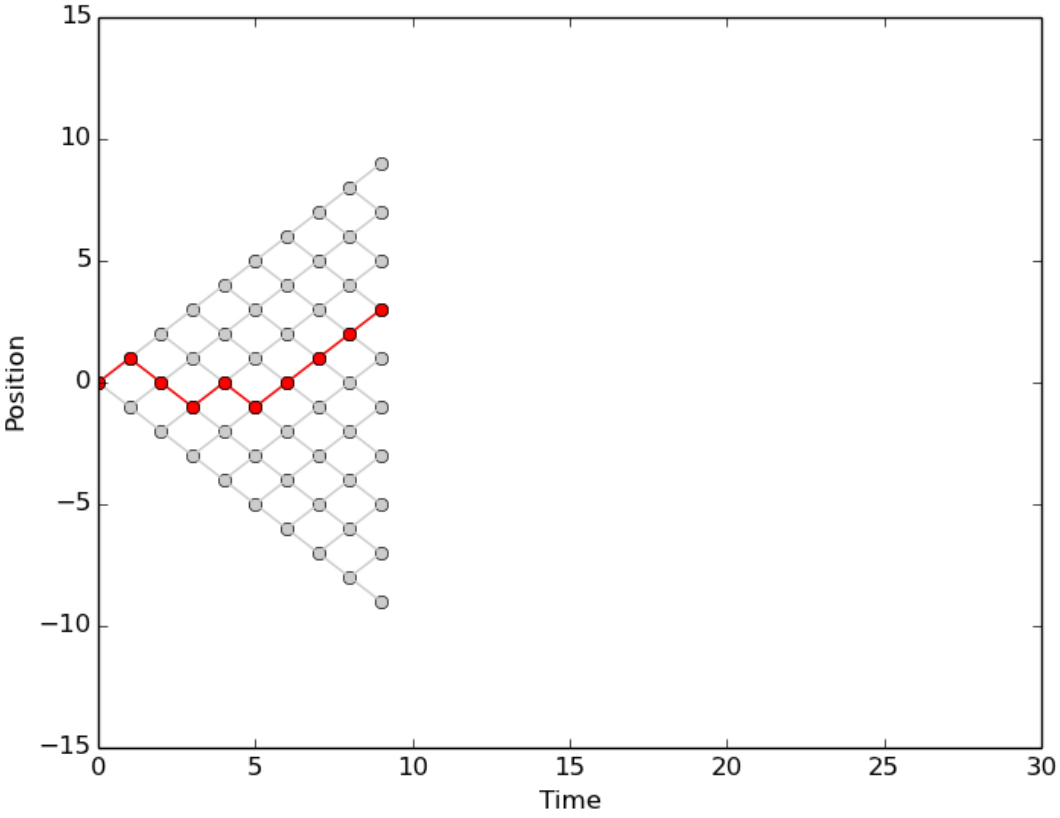
1D – example



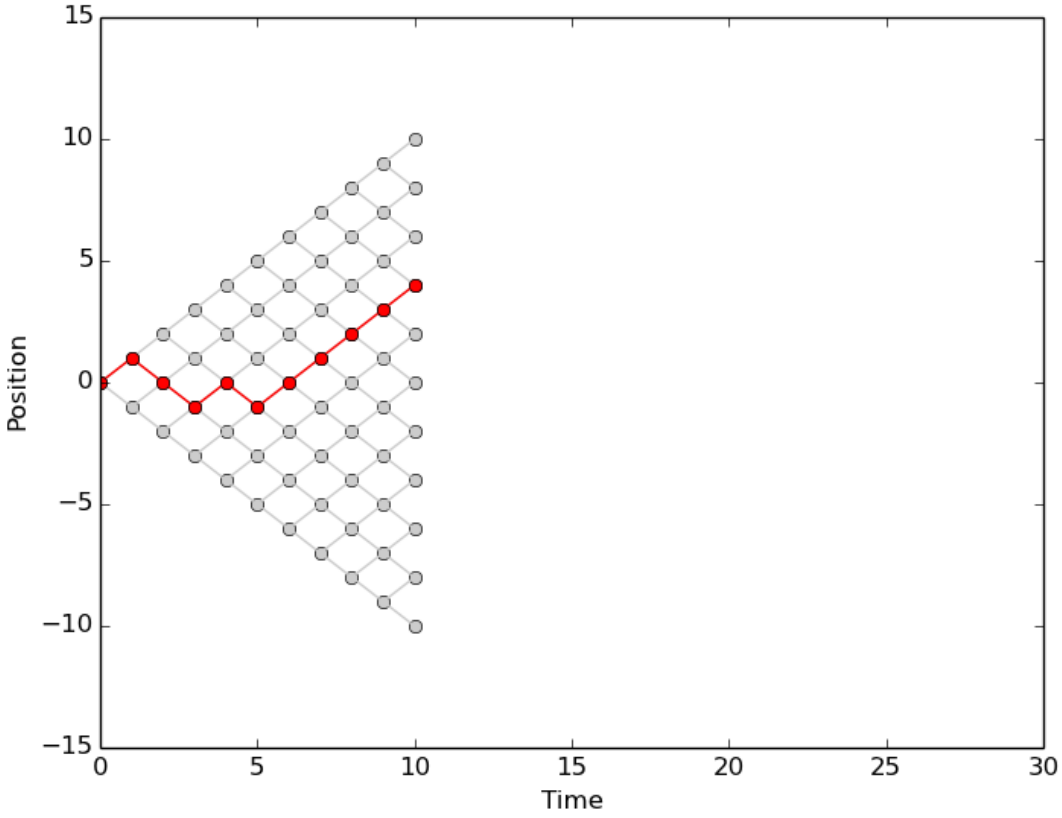
1D – example



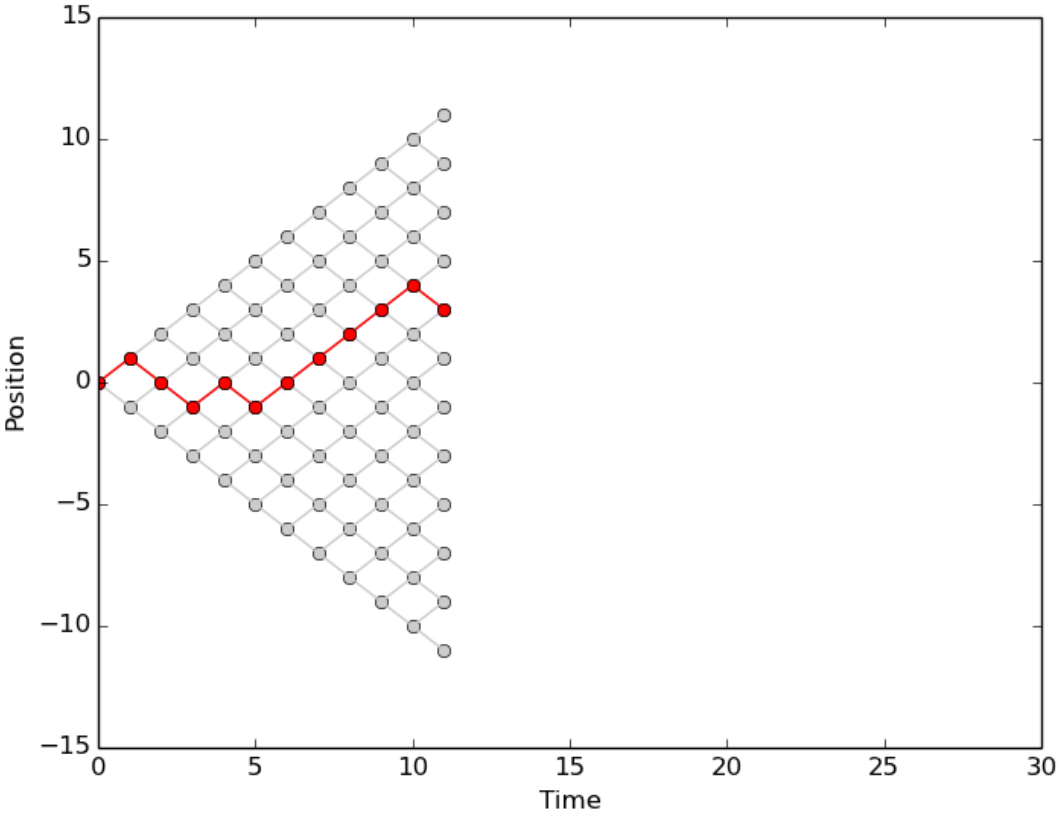
1D – example



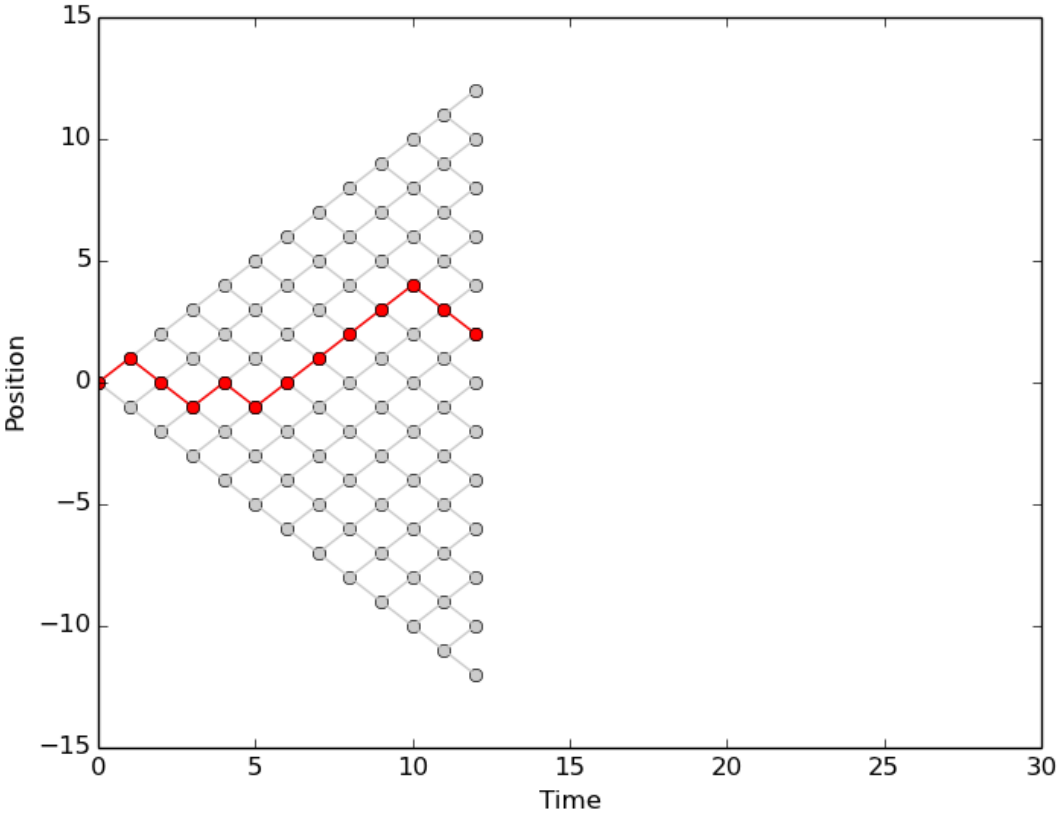
1D – example



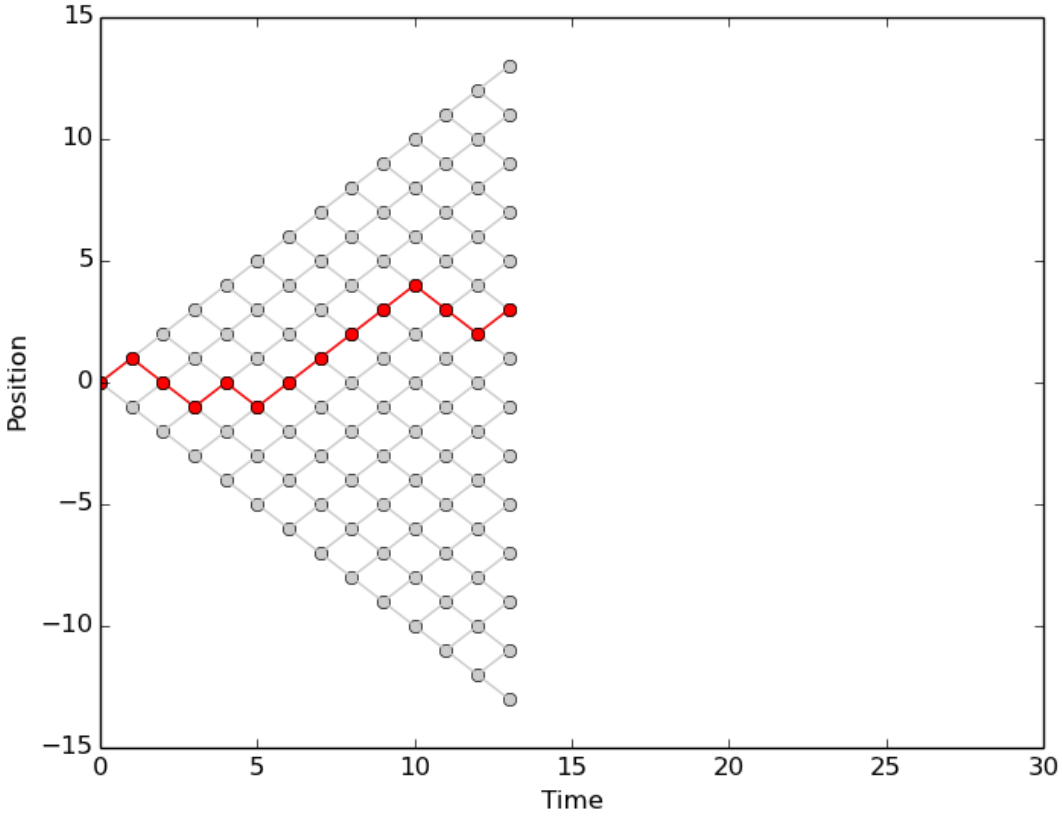
1D – example



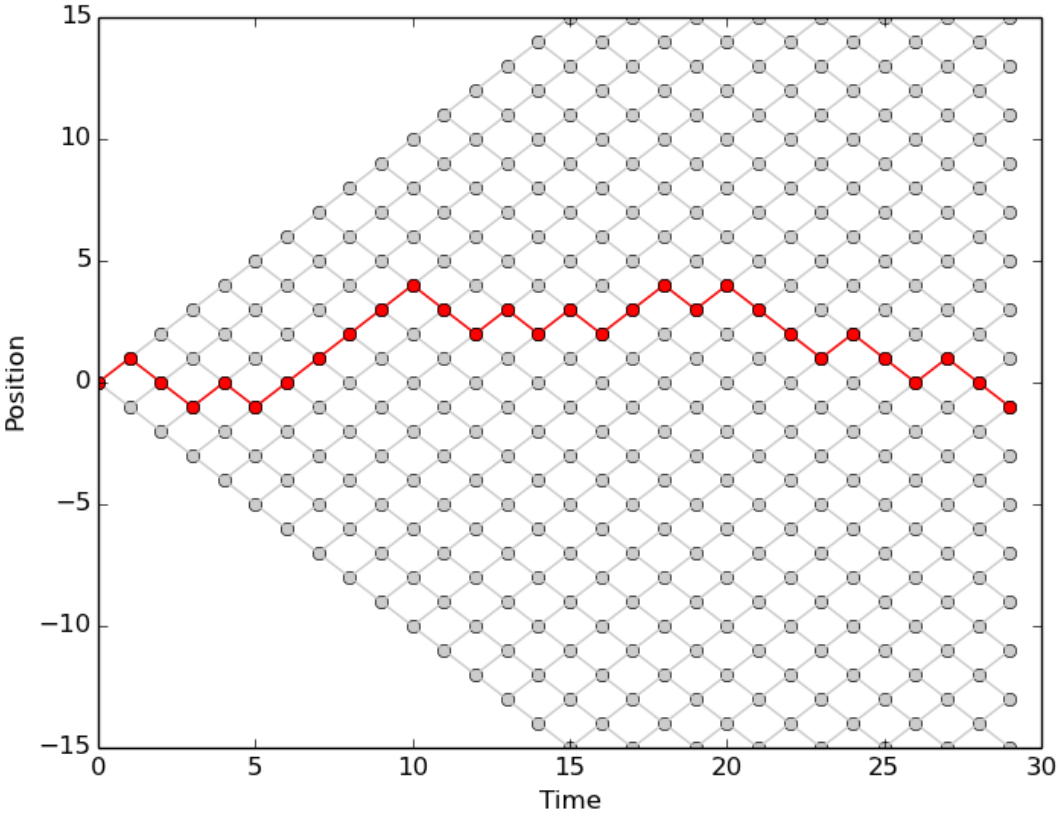
1D – example



1D – example



1D – example

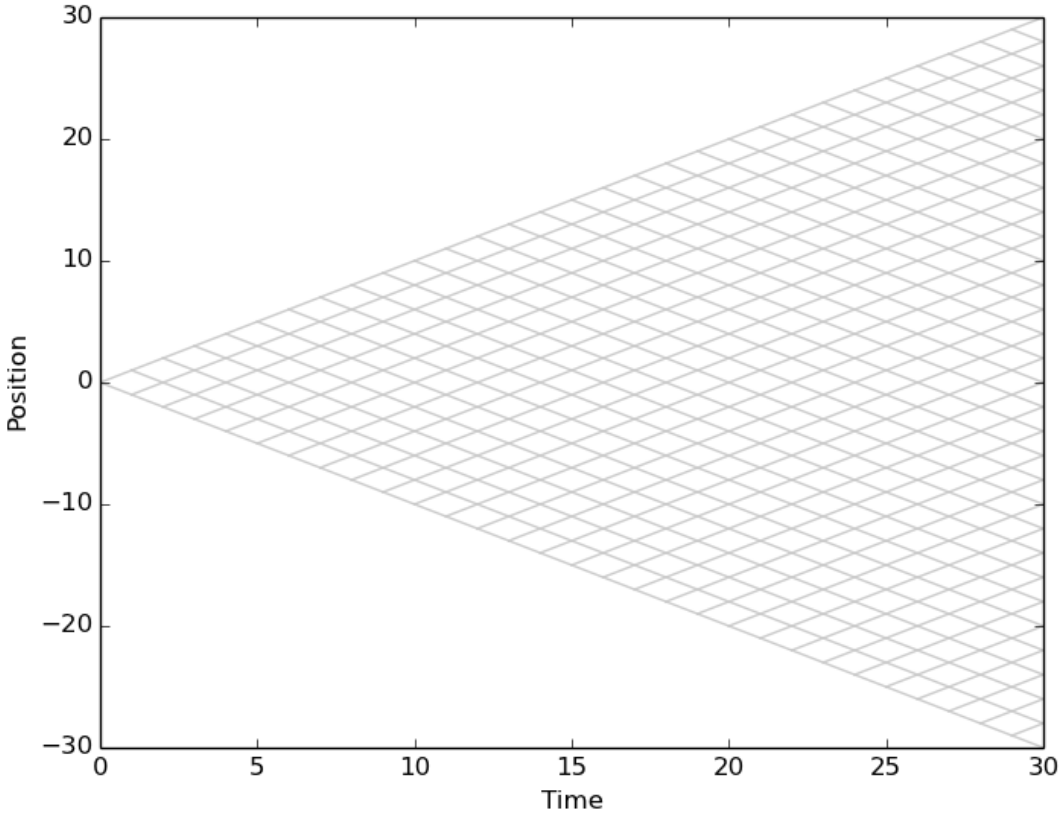


Bulk Behavior

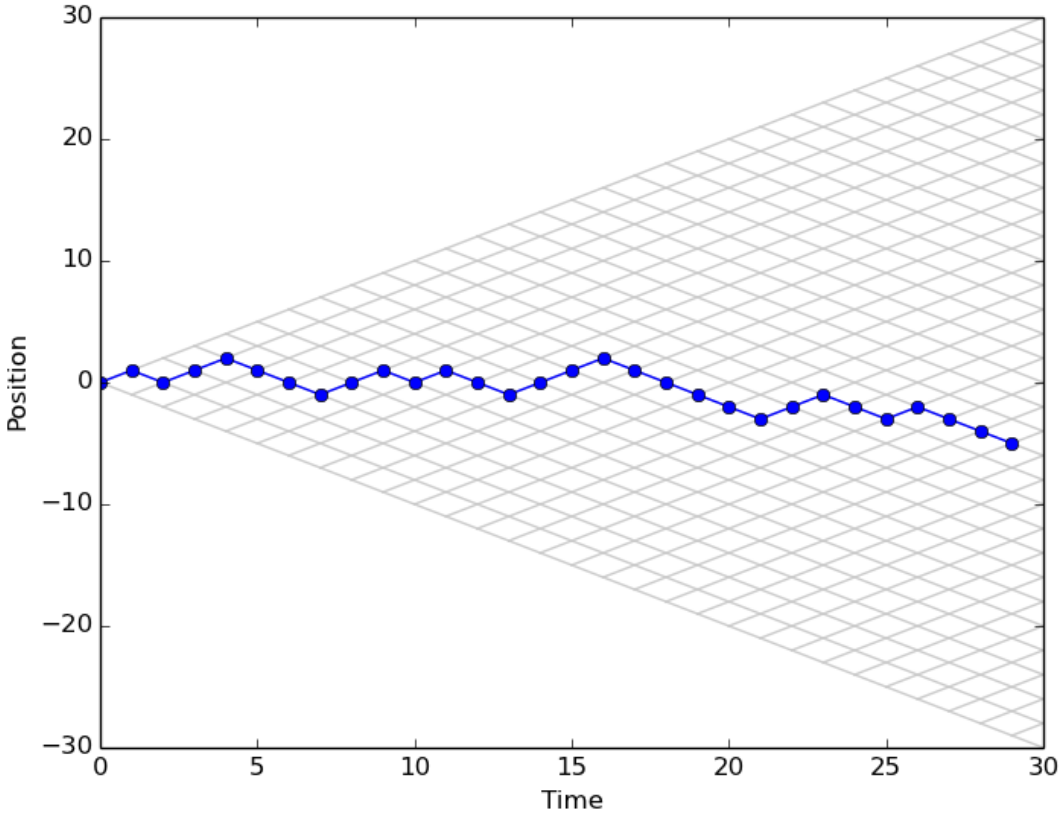
- An individual example follows a random course

- However, we still have a “continuum behavior”
  - Just like radioactive decay

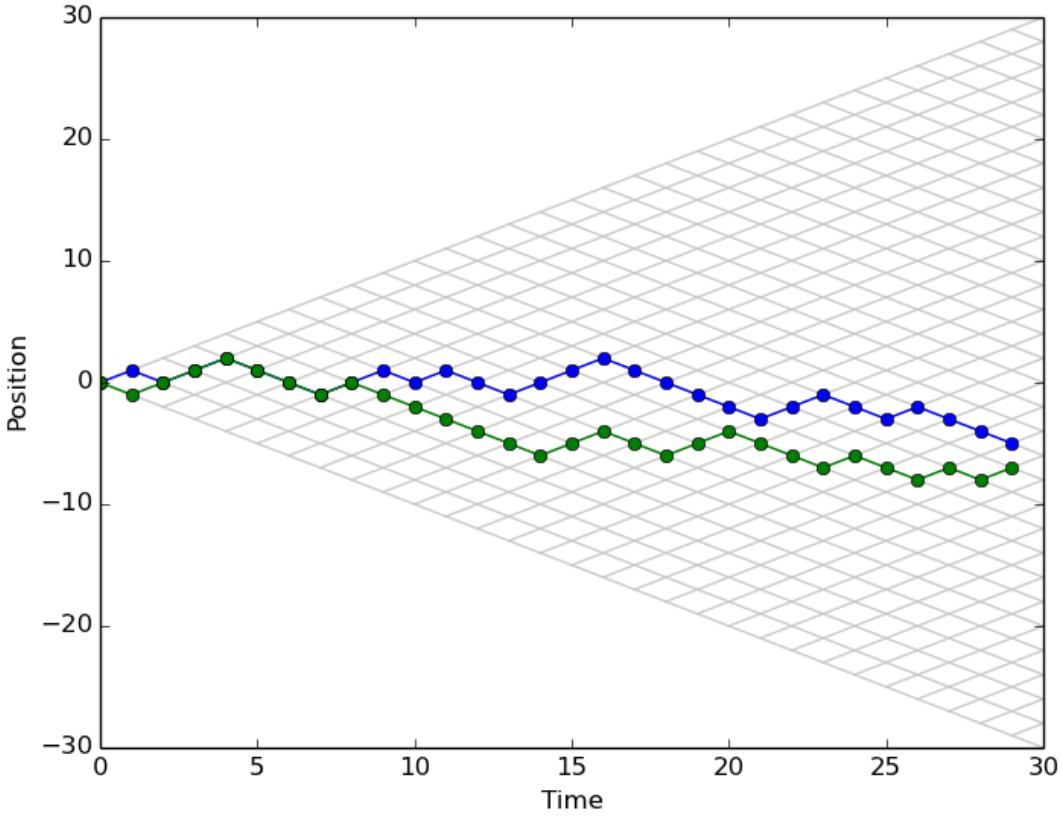
Bulk Behavior



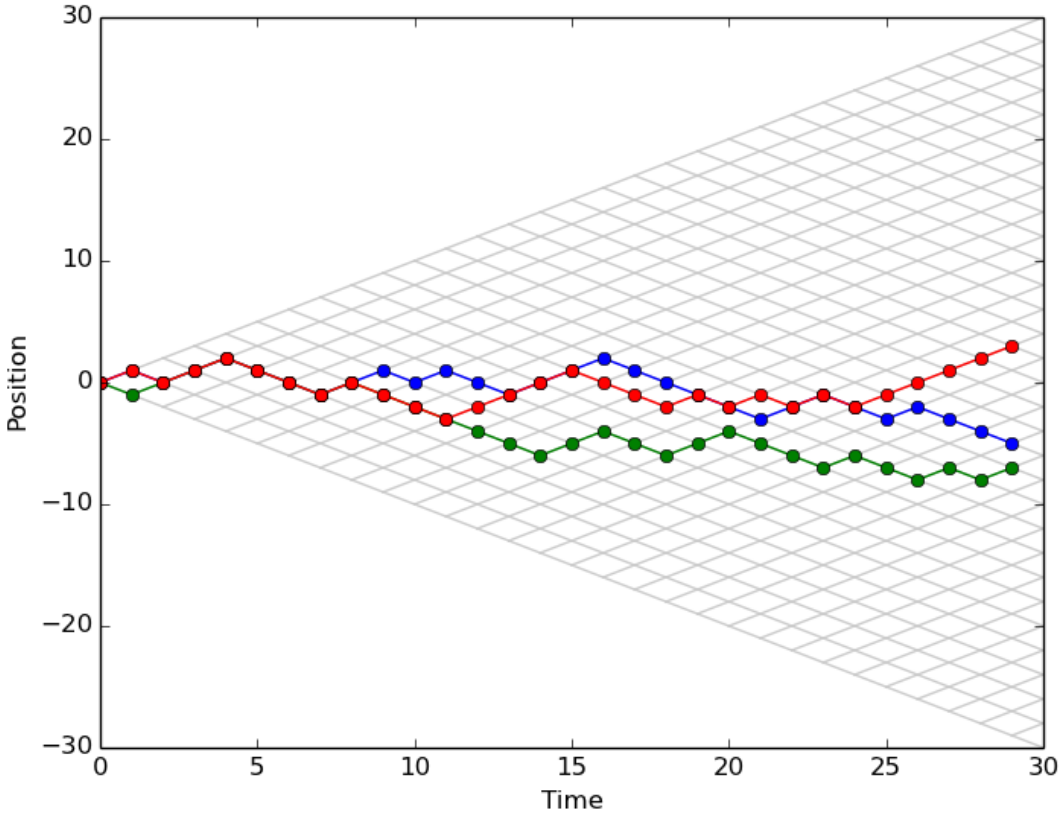
Bulk Behavior



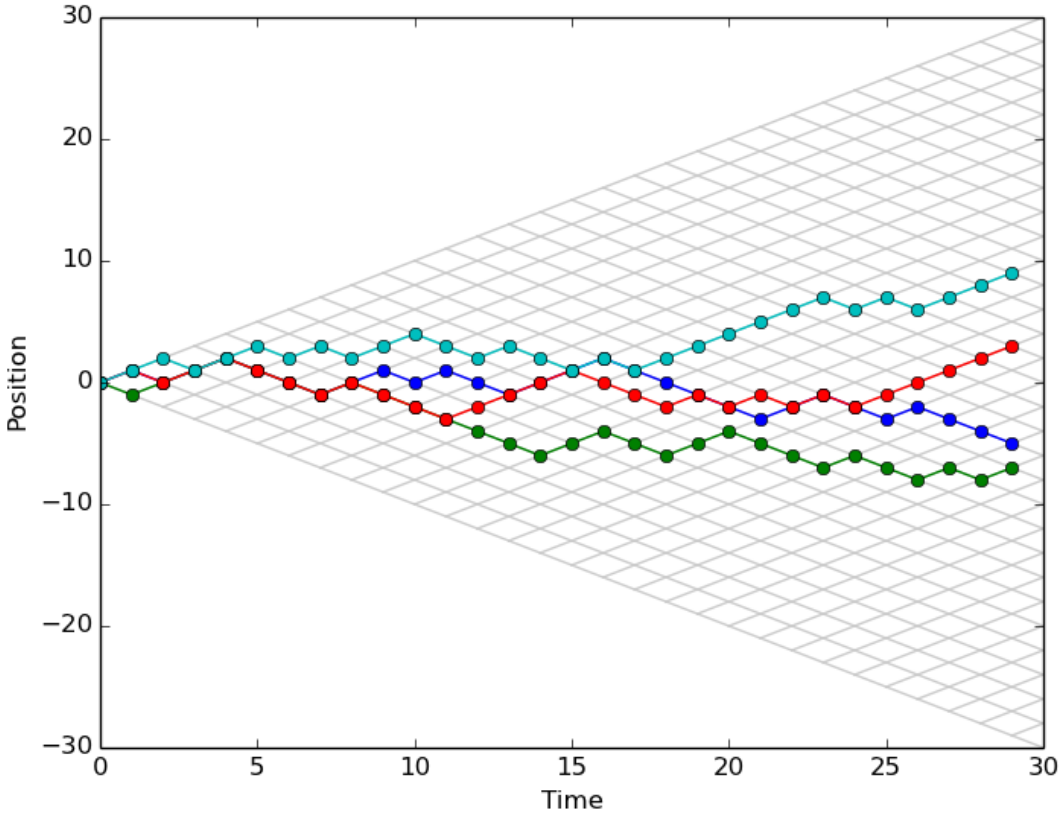
Bulk Behavior



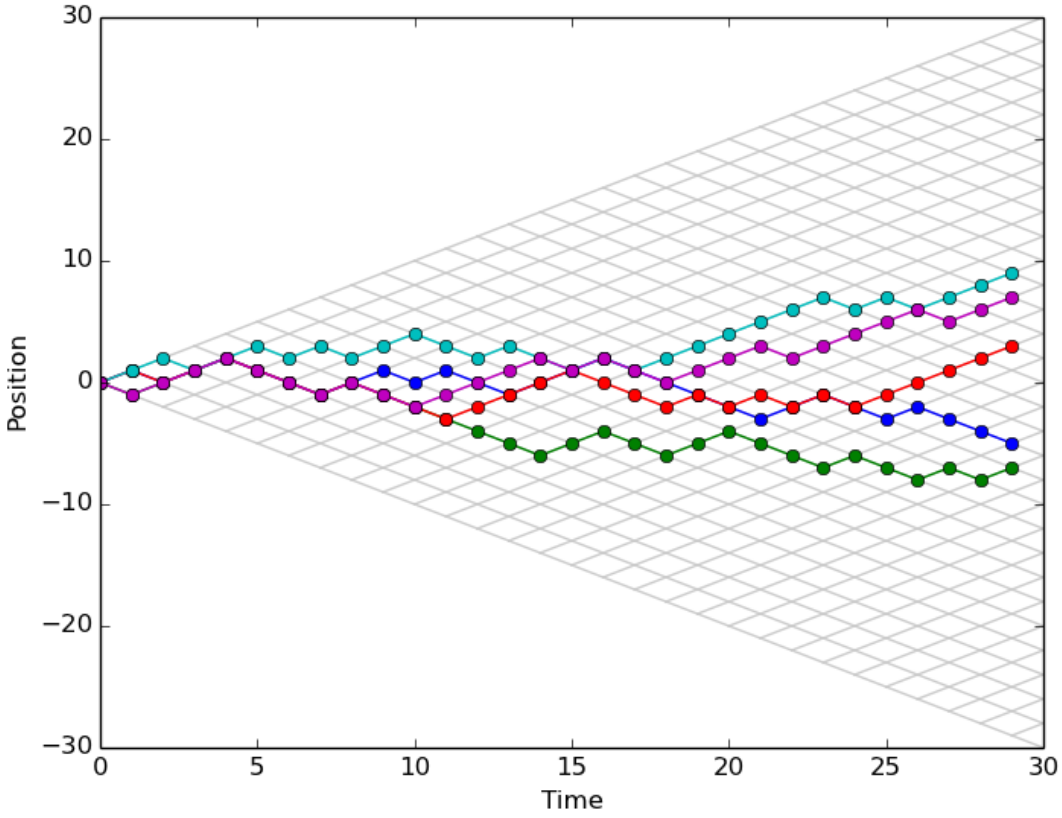
Bulk Behavior



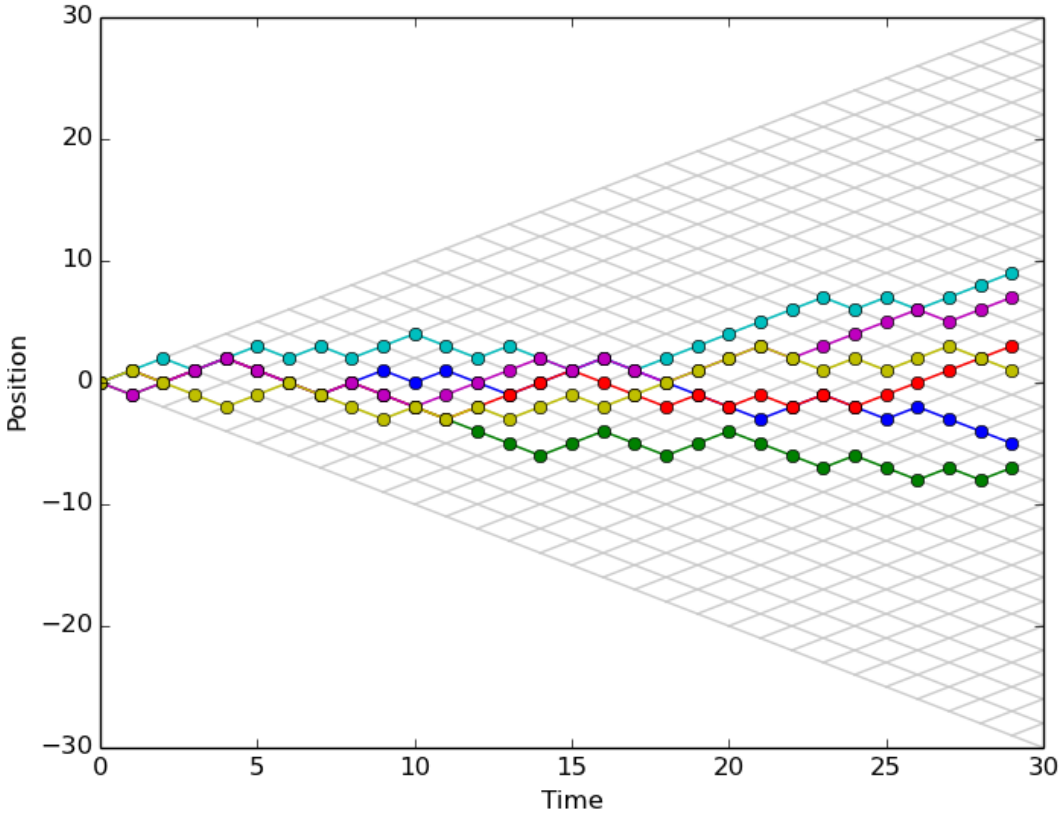
Bulk Behavior



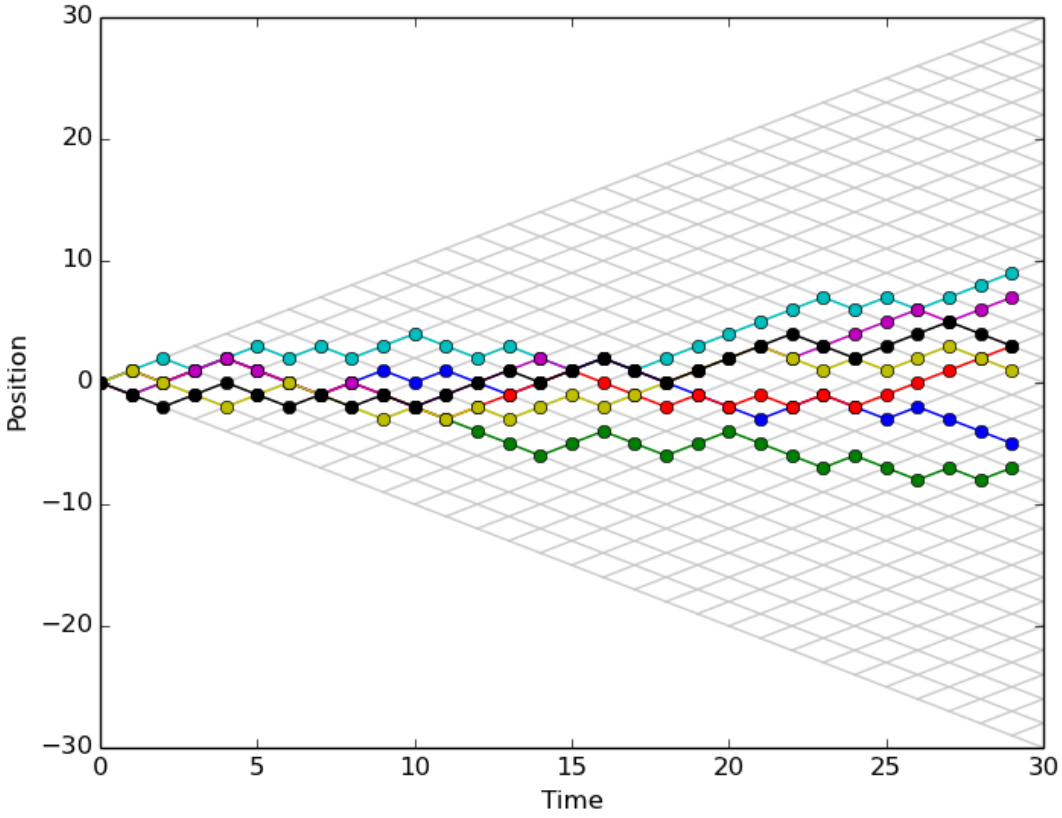
Bulk Behavior



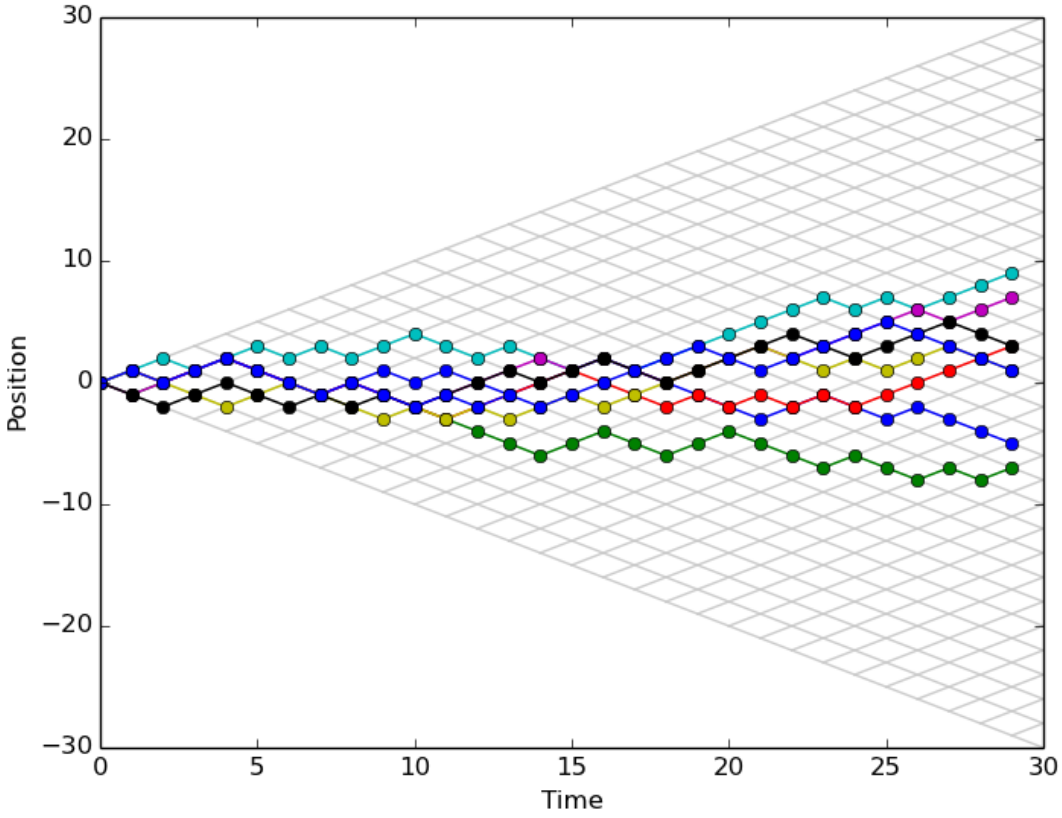
Bulk Behavior



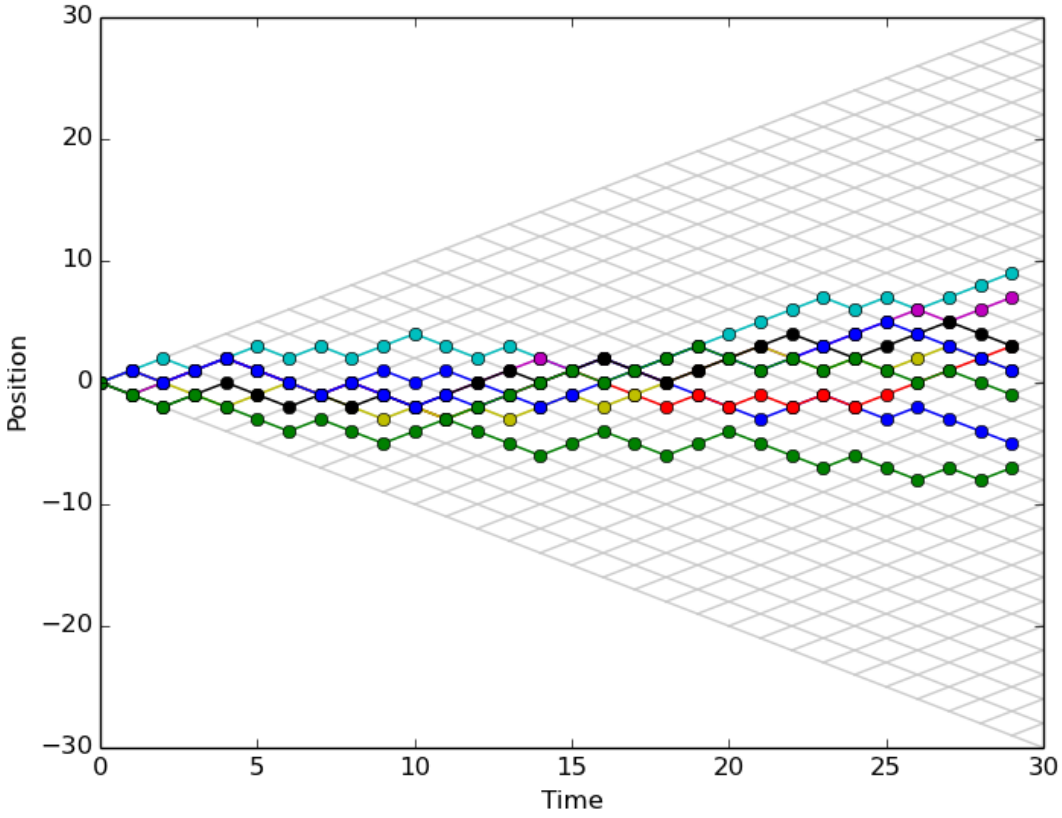
Bulk Behavior



Bulk Behavior



Bulk Behavior



Examining bulk properties

- $P(x,t)$ Probability of position x at time t
- $\langle x \rangle$ Average position
- $\langle (x-x_0)^2 \rangle$ Mean Square Displacement

P(x,t)

Monte-Carlo

- Simulation many thousands of particles over many timesteps
- See how many there are at each (x,t) and normalize to a probability
- Similar to your work last week

Analytical Solution

P(x,t)

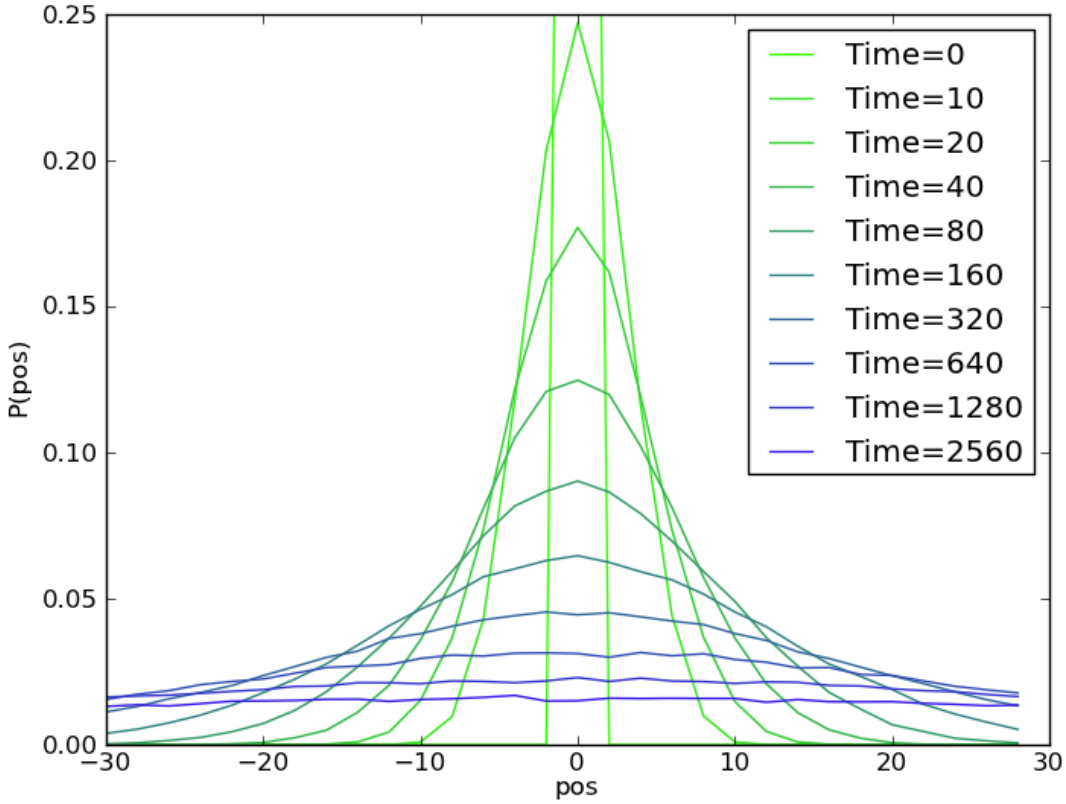
Monte-Carlo

- Simulation many thousands of particles over many timesteps
- See how many there are at each (x,t) and normalize to a probability

Analytical Solution

- Pascal's triangle

P(x,t)


0.25
0.20
0.15
0.10
0.05
0.00
P(pos)
−30
−20
−10
0
10
20
30
pos
Time=0
Time=10
Time=20
Time=40
Time=80
Time=160
Time=320
Time=640
Time=1280
Time=2560

Diffusion

- A bulk statistical behavior emerges from the random motion
- **Statistical Mechanics**
- Diffusion
  - The spread of particles
  - From an area of high concentration
  - To areas of low concentration
  - Through random motion
- No directional force acts on the particles
- Disorder just increases because that is statistically more likely than disorder decreasing
- The arrow of time!

Mean Square Displacement

- MSD is a very useful way of characterizing diffusing systems

$\vec{r}_i(0)$ position of particle i at time 0

$\vec{r}_i(t)$ position of particle i at time t

$$msd(t) = \left\langle \left(r_i(t) - r_i(0) \right)^2 \right\rangle$$ mean square displacement at time t

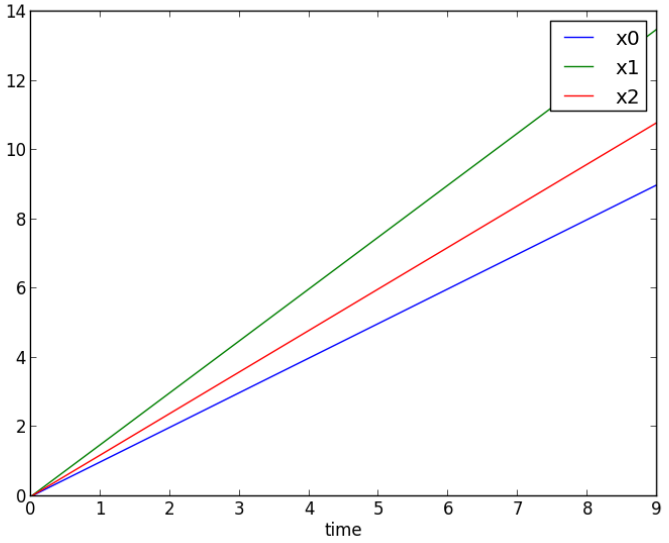
- The average of the square of the displacements many particles have moved
- A scalar quantity regardless of dimensionality

Aside – numpy.average

average0.py – /Users/cds/cds/Teaching/CompPhys/Week 7/a...

```
from __future__ import division
import matplotlib.pyplot as pyplot
import numpy

# Example of using numpy to average trajectories

# Let's make 3 trajectoris of 'x' at many timepoints
timebase = numpy.arange(10)
x0 = timebase
x1 = 1.5 * timebase # this one is faster
x2 = 1.2 * timebase

# Let's plot them
pyplot.figure()
pyplot.plot(timebase, x0, label='x0')
pyplot.plot(timebase, x1, label='x1')
pyplot.plot(timebase, x2, label='x2')
pyplot.xlabel('time')
pyplot.legend()
pyplot.savefig('average_0.png')
pyplot.show()

# make a list of trajectories
t_list = [x0, x1, x2]

# average them
print numpy.average(t_list)
```

Ln: 1 Col: 0

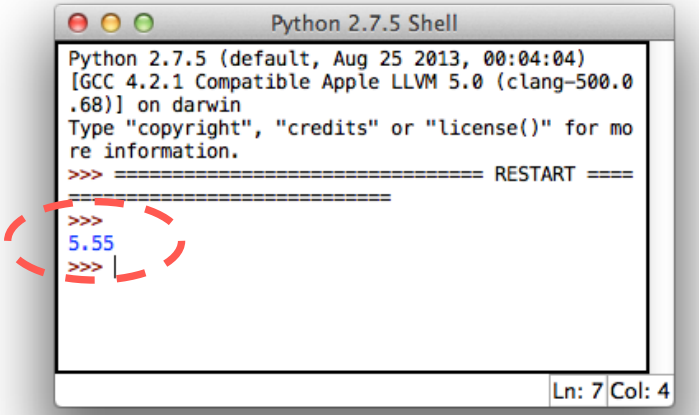


Aside – numpy.average

average0.py – /Users/cds/cds/Teaching/CompPhys/Week 7/a...

```
from __future__ import division
import matplotlib.pyplot as pyplot
import numpy

# Example of using numpy to average trajectories

# Let's make 3 trajectoris of 'x' at many timepoints
timebase = numpy.arange(10)
x0 = timebase
x1 = 1.5 * timebase # this one is faster
x2 = 1.2 * timebase

# Let's plot them
pyplot.figure()
pyplot.plot(timebase, x0, label='x0')
pyplot.plot(timebase, x1, label='x1')
pyplot.plot(timebase, x2, label='x2')
pyplot.xlabel('time')
pyplot.legend()
pyplot.savefig('average_0.png')
pyplot.show()

# make a list of trajectories
t_list = [x0, x1, x2]

# average them
print numpy.average(t_list)
```

Ln: 1 Col: 0



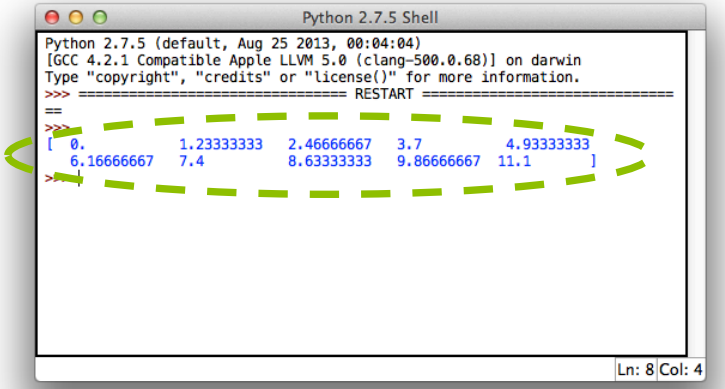
Oh dear, it's averaged all the numbers at all timepoints

Aside – numpy.average

average1.py – /Users/cds/cds/Teaching/CompPhys/Week 7/a...

```
from __future__ import division
import matplotlib.pyplot as pyplot
import numpy

# Example of using numpy to average trajectories

# Let's make 3 trajectoris of 'x' at many timepoints
timebase = numpy.arange(10)
x0 = timebase
x1 = 1.5 * timebase # this one is faster
x2 = 1.2 * timebase

# Let's plot them
pyplot.figure()
pyplot.plot(timebase, x0, label='x0')
pyplot.plot(timebase, x1, label='x1')
pyplot.plot(timebase, x2, label='x2')
pyplot.xlabel('time')
pyplot.legend()
pyplot.savefig('average_0.png')
pyplot.show()

# make a list of trajectories
t_list = [x0, x1, x2]

# average them
print numpy.average(t_list, axis=0)
```

Ln: 27 Col: 28



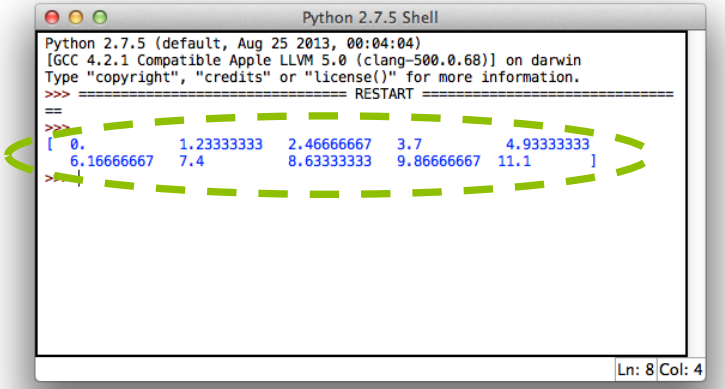
This is more like it – all values at a single timepoint are averaged

Aside – numpy.average

average1.py – /Users/cds/cds/Teaching/CompPhys/Week 7/a...

```
from __future__ import division
import matplotlib.pyplot as pyplot
import numpy

# Example of using numpy to average trajectories

# Let's make 3 trajectoris of 'x' at many timepoints
timebase = numpy.arange(10)
x0 = timebase
x1 = 1.5 * timebase # this one is faster
x2 = 1.2 * timebase

# Let's plot them
pyplot.figure()
pyplot.plot(timebase, x0, label='x0')
pyplot.plot(timebase, x1, label='x1')
pyplot.plot(timebase, x2, label='x2')
pyplot.xlabel('time')
pyplot.legend()
pyplot.savefig('average_0.png')
pyplot.show()

# make a list of trajectories
t_list = [x0, x1, x2]

# average them
print numpy.average(t_list, axis=0)
```

Ln: 27 Col: 28



This is more like it – all values at a single timepoint are averaged

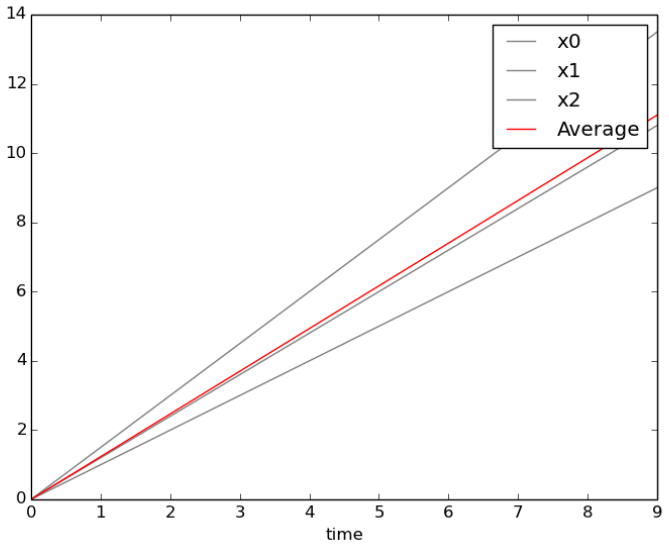
- Numpy *reduction* methods can take an axis= option which specifics to specify which axis/axes to reduce the data over
- e.g., numpy.sum, .min, .max, .std, .average
- Default is to reduce over all axes

Aside – numpy.average

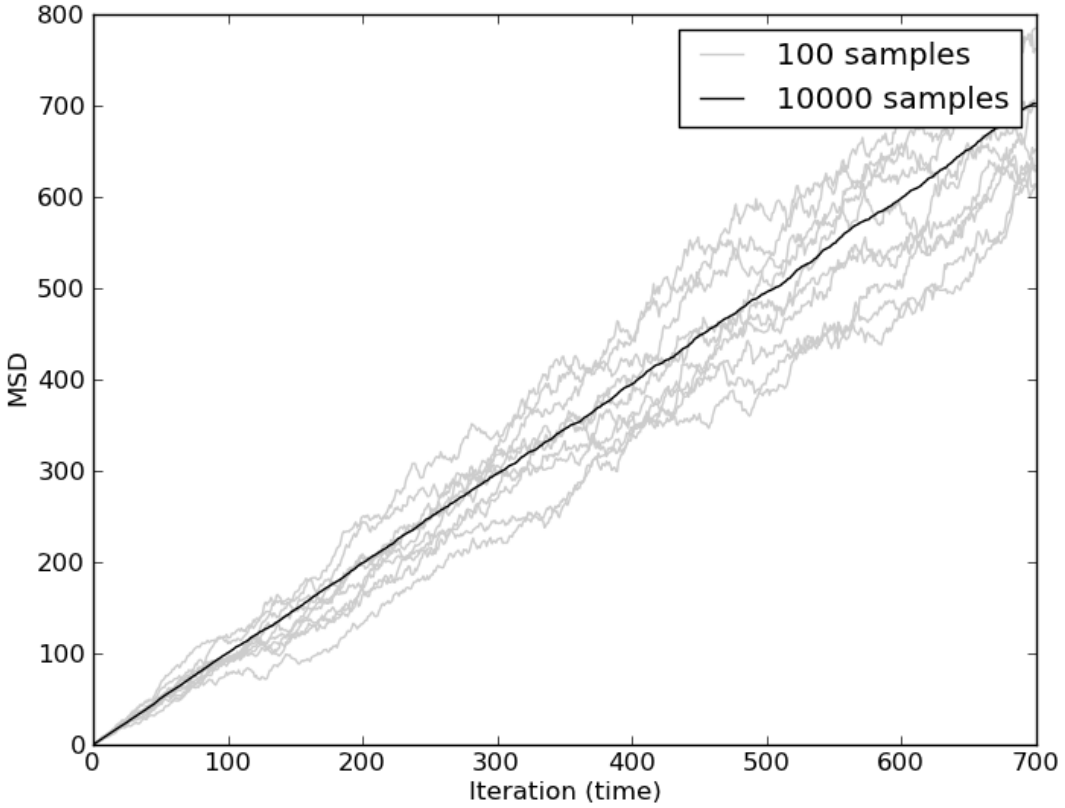
average2.py – /Users/cds/cds/Teaching/CompPhys/Week 7/a...

```
from __future__ import division
import matplotlib.pyplot as pyplot
import numpy

# Example of using numpy to average trajectories

# Let's make 3 trajectoris of 'x' at many timepoints
timebase = numpy.arange(10)
x0 = timebase
x1 = 1.5 * timebase # this one is faster
x2 = 1.2 * timebase

# Let's plot them
pyplot.figure()
pyplot.plot(timebase, x0, label='x0', color='grey')
pyplot.plot(timebase, x1, label='x1', color='grey')
pyplot.plot(timebase, x2, label='x2', color='grey')
pyplot.xlabel('time')

# make a list of trajectories
t_list = [x0, x1, x2]

# average them
avg = numpy.average(t_list, axis=0)

pyplot.plot(timebase, avg, label='Average', color='red')
pyplot.legend()
pyplot.savefig('average_2.png')
pyplot.show()
```

Ln: 28 Col: 25

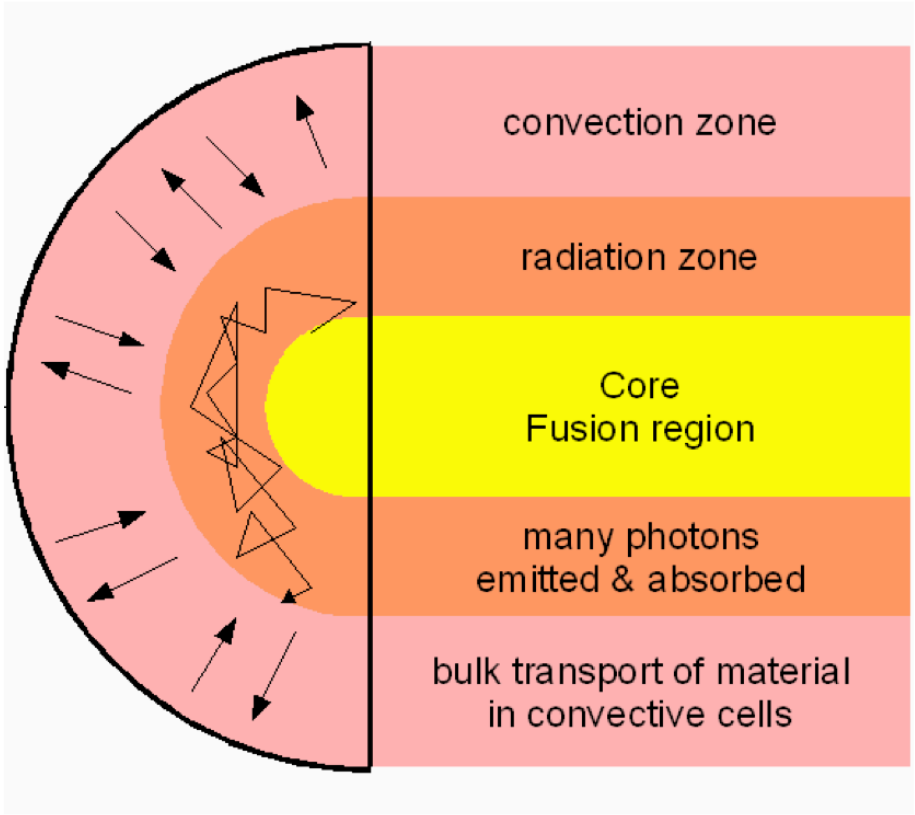


Mean Square Displacement



Random Walks in the Wild

Solar Radiation



Mean Free Path of a photon in the radiation zone is ~1cm

It takes an average of > 50,000 years for one photon to escape the radiation zone

Brownian Motion

- Discovered in 1827 boy Robert Brown
- Botanist studying pollen under a microscope
- He noticed they appeared to move randomly without a cause
- Actually, collisions with molecules in the medium

Brownian Motion

- Albert Einstein published a paper in 1905 describing the statistical mechanics behind Brownian Motion
- One of his great *annus mirabilis papers*
- “Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen”
- “On the movement of small particles suspended in a stationary liquid demanded by the molecular-kinetic theory of heat”
- http://users.physik.fu-berlin.de/~kleinert/files/eins_brownian.pdf
  - (English translation)

Brownian Motion

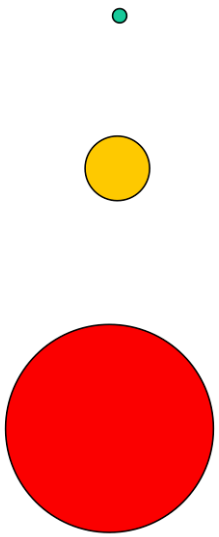
- There is a simple mathematical relationship between:
  - Size of a particle
  - The density and temperature of the fluid it is in
  - The strength and timescales of Brownian motion it experiences

  - This allows Brownian motion to be used to determine the size of particles from ~1nm to ~100μm through *dynamic lights scattering*, *optical tweezing* and *microrheology*

  - *http://www.youtube.com/watch?v=cDcprgWiQEY*

Bacterial mobility

Bacteria



Virus

Prokaryotes

(including Bacteria)

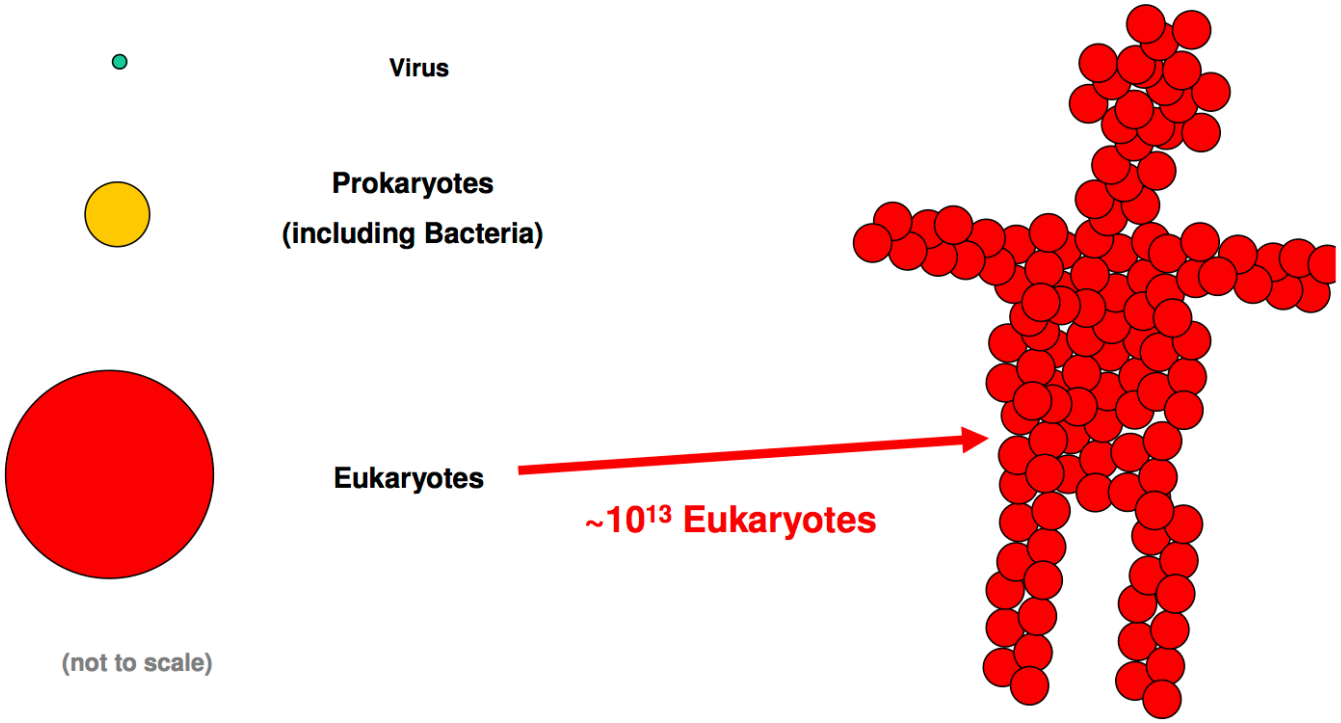
Single-celled living organism, no separate nucleus

Eukaryotes

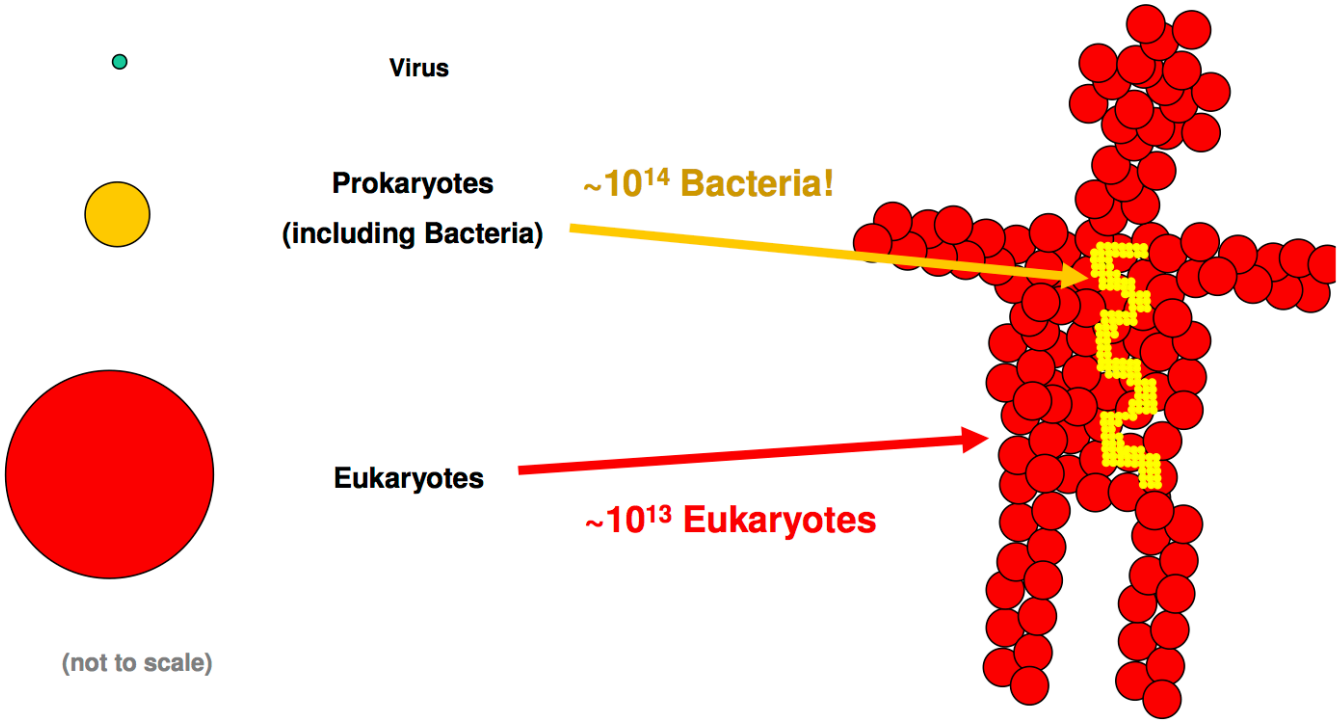
Single cell with a separate nucleus – forms the basis of multi-cellular life

(not to scale)

Are you human?



Are you human?



Bacterial Motility

- Bacteria need to move for various reasons
  - To find food (energy)
  - To escape toxins / poisons (including waste)

Flagella

- Flagella are long whip-like protrusions ~20nm in diameter
- The cell rotates them about their axis
  - One of only two genuine **rotary joints** found in biology
- Rotation of flagella has two states
  1. **Run** - CCW rotation aligns all flagella and propels bacteria in a straight line
  2. **Tumble** – CW rotation separates flagella causing the bacteria to rotate "on the spot" randomly

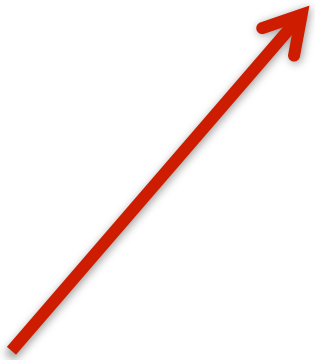
http://info.fujita-hu.ac.jp/~tsutsumi/photo/photo002-6.htm Yutaka Tsutsumi, M.D. Professor Department of Pathology Fujita Health University School of Medicine

Computational Biophysics

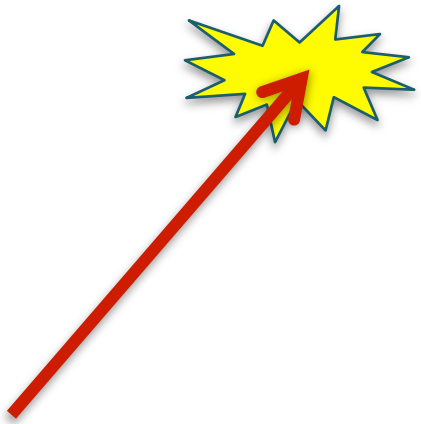
- Today we will be using CP to look at the motion of a bacterial cell

- CP is also a key tool to study the behavior of smaller parts such as the flagella's operation
  - Protein folding
  - Fluid dynamics
  - Physical Chemistry
  - Brownian Dynamics

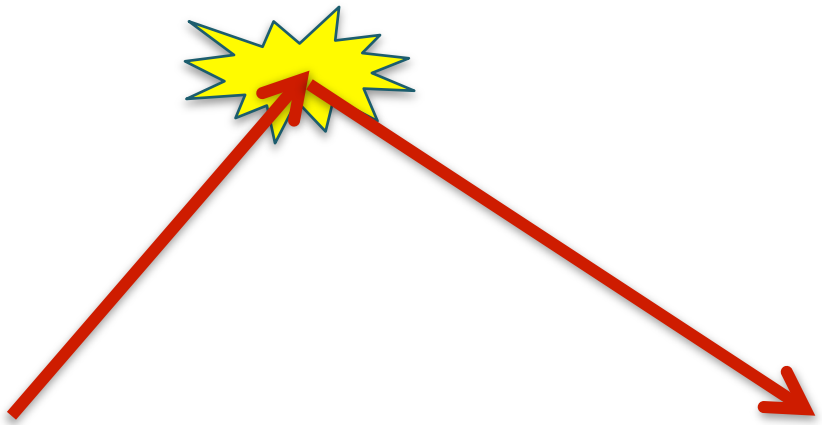
Run and Tumble



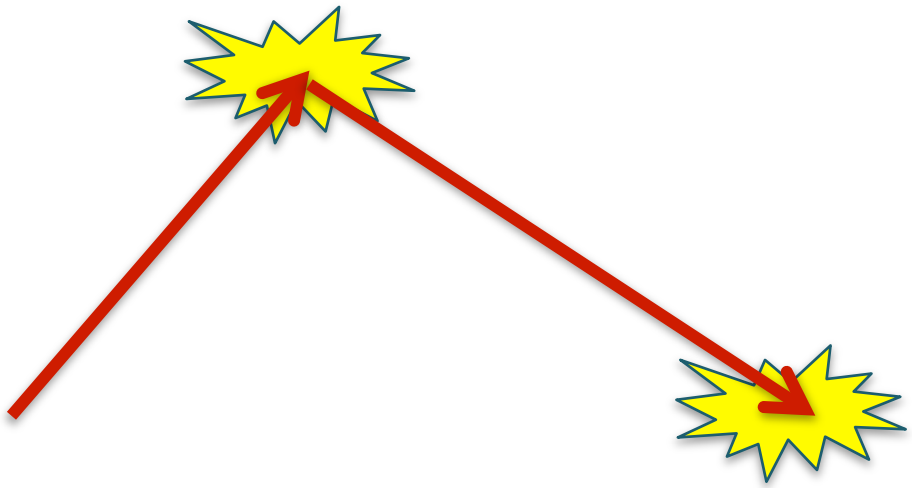
Run and Tumble



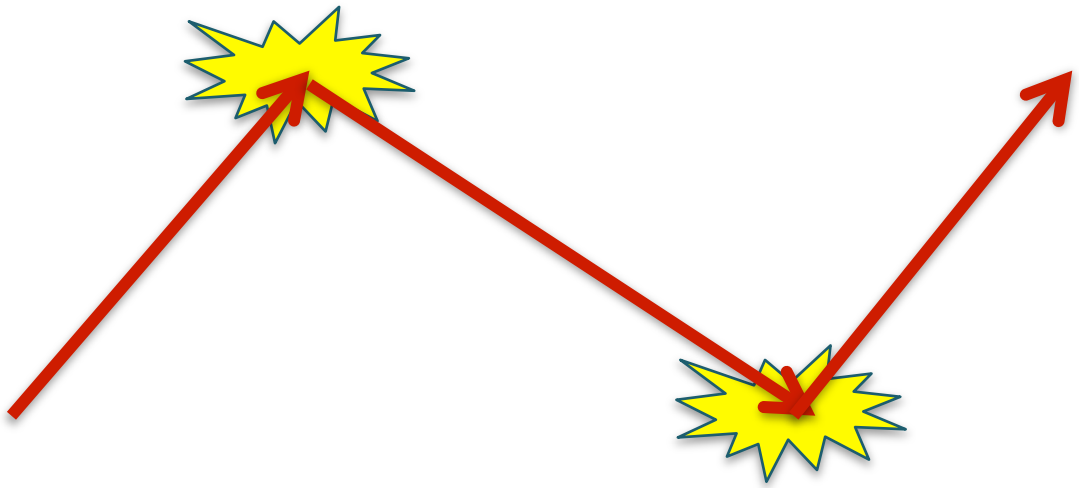
Run and Tumble



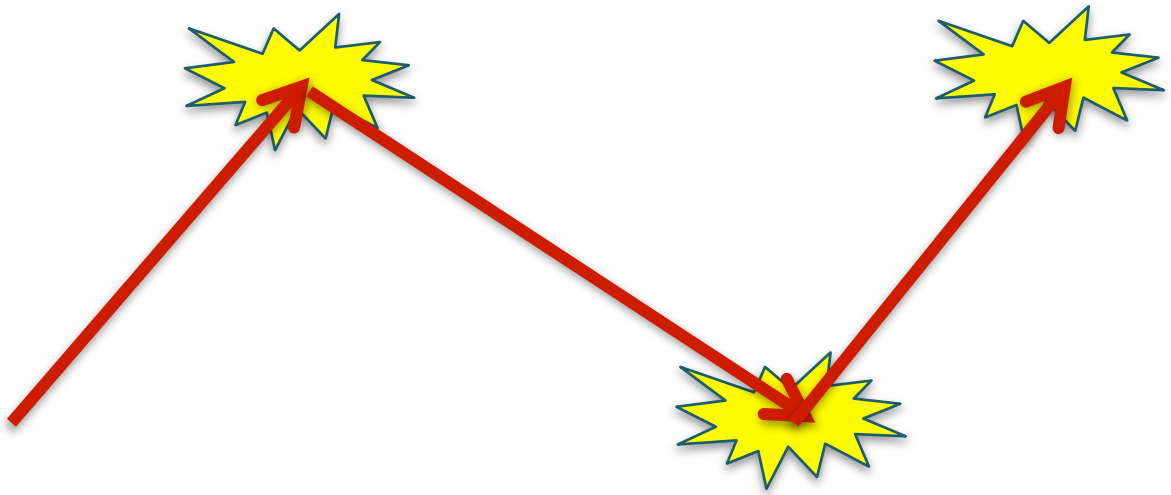
Run and Tumble



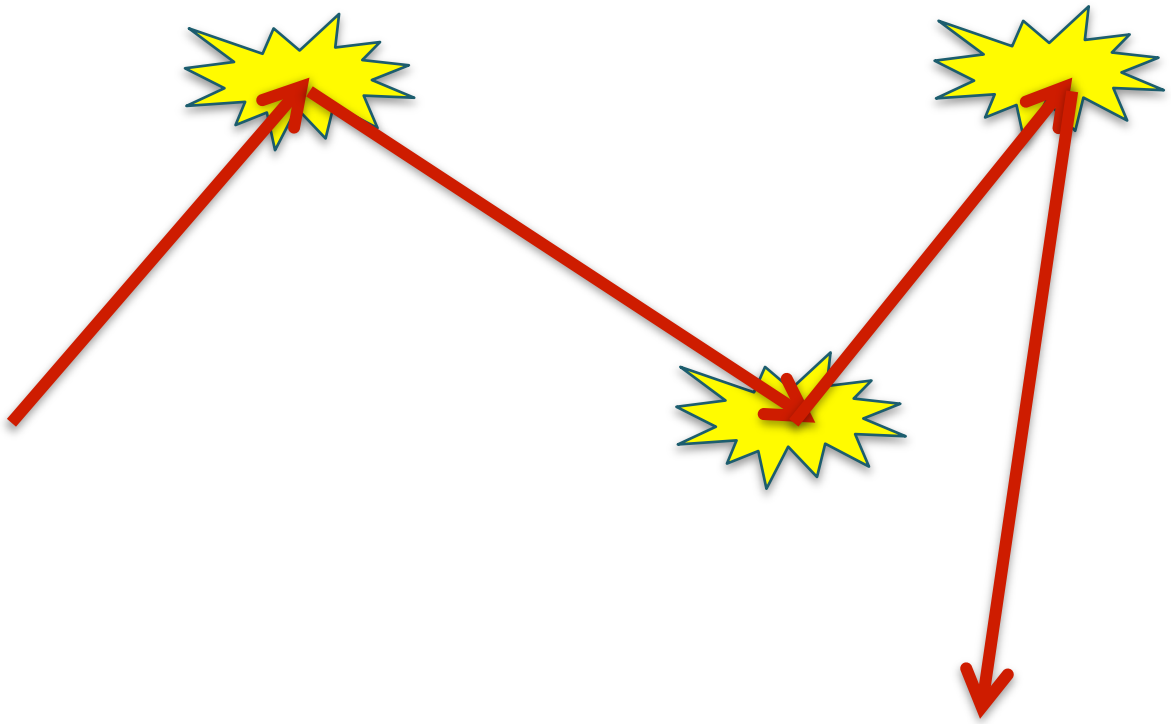
Run and Tumble



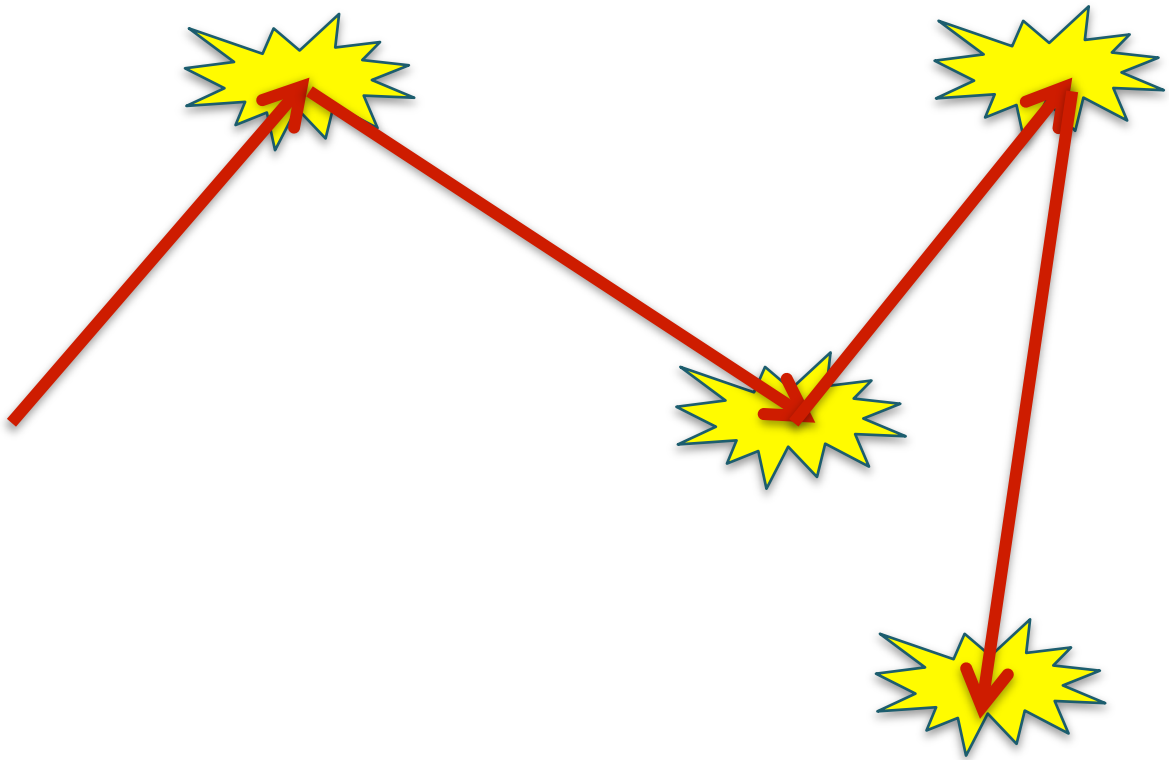
Run and Tumble



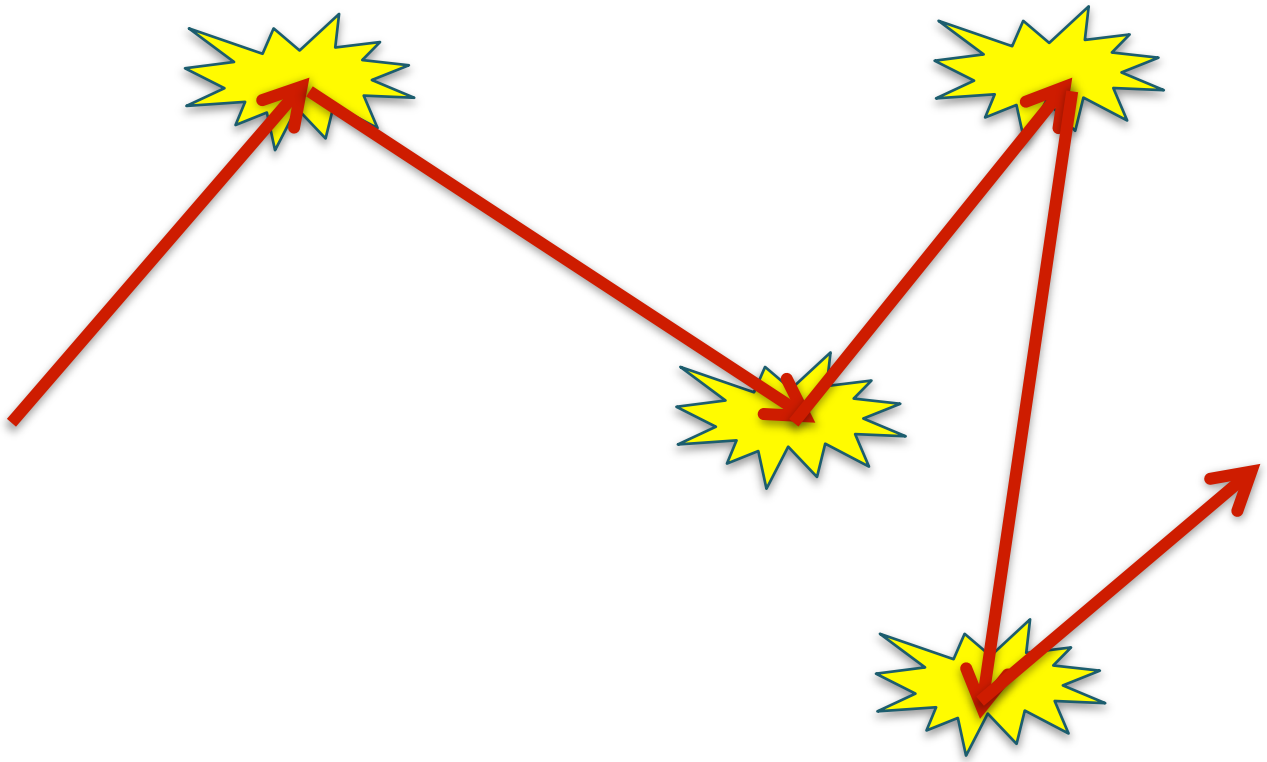
Run and Tumble



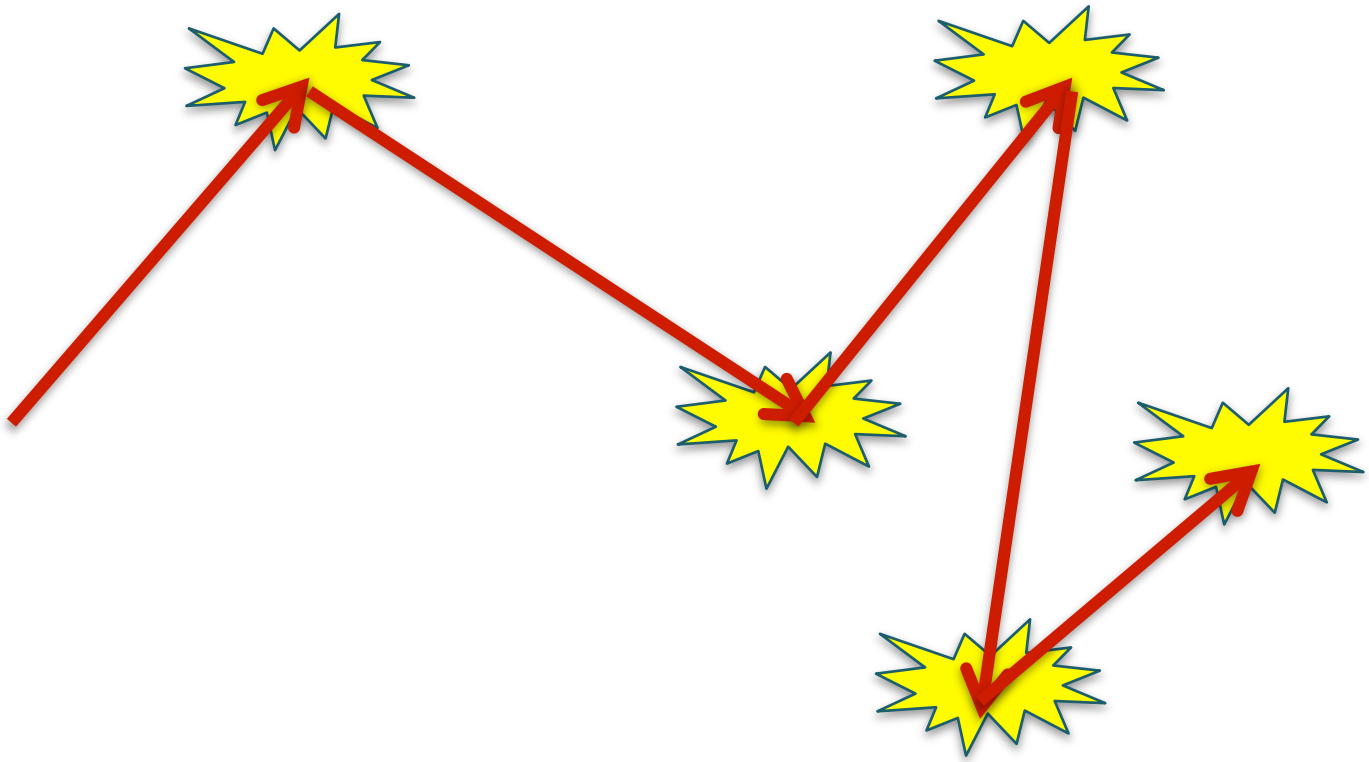
Run and Tumble



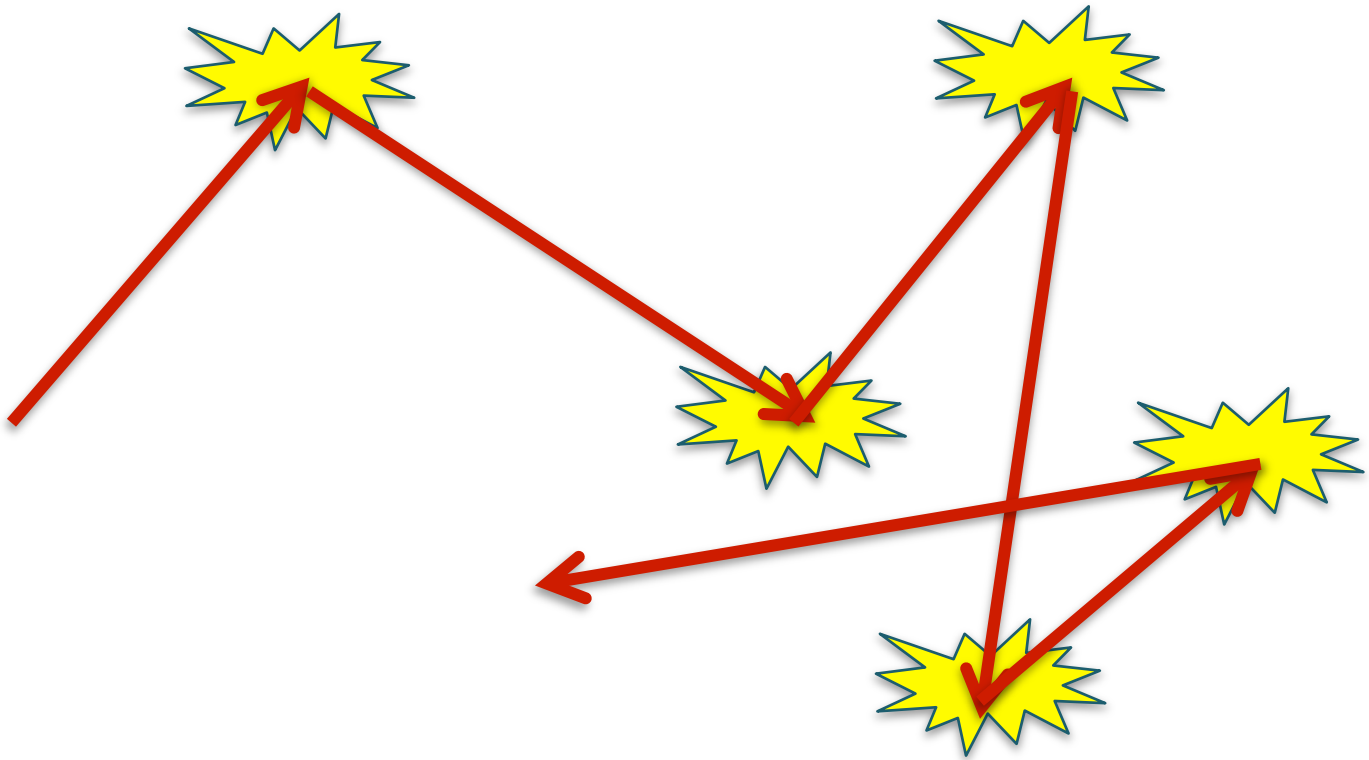
Run and Tumble



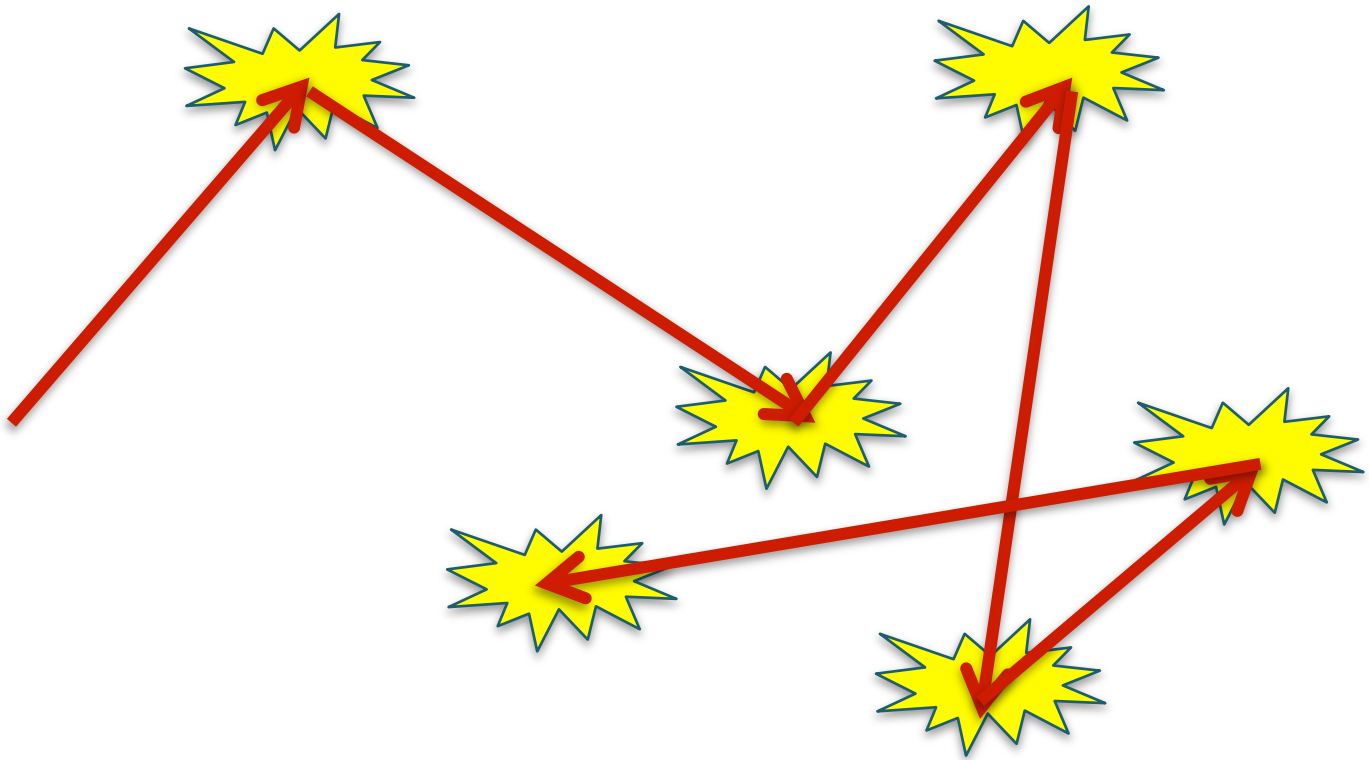
Run and Tumble



Run and Tumble



Run and Tumble

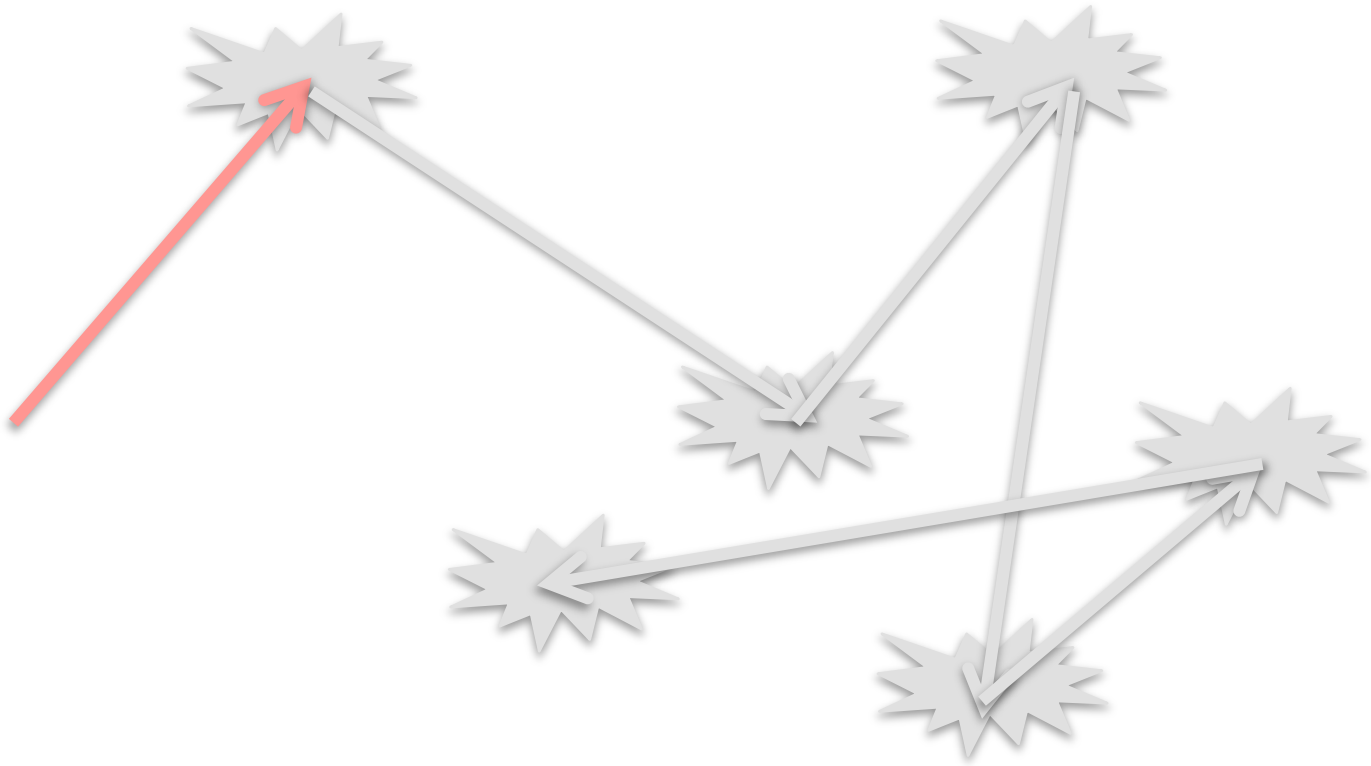


Modeling the motion

- **Before you write any code**
  - Diagram
  - Assumptions
  - Quantities
    - Constants
    - Initial conditions
    - State – how do we represent the state of the model at any one time?

- Formulate your problem before solving it
  - Applies far wider than Computational Physics!

Diagram



Assumptions

- Assumptions:
  - 2-dimensional space
    - Many bacteria live on a surface
    - Either running or tumbling
    - Velocity (speed and angle constant during a run)
    - Angle after a tumble is random
    - Speed after a tumble is the same as before
    - Probability of a tumble in some time interval, dt, is constant

Quantities

Constants

- V — Run speed
- P_tumble — Probability of tumbling in 1 sec

Initial conditions

- $\mathbf{R_0} = (x_0, y_0)$ — Initial position
- α_0 — Initial angle

State

- t_n — Time at the n^{th} iteration(timestep)
- $\mathbf{R_n} = (x_n, y_n)$ — Position at n^{th}
- α_n — Angle of travel at n^{th}

Simulation approach

Decide on a timestep, dt

Quantize time

Initialize speed, time, angle, position

Repeat many times

    If random() < p(tumbles in time dt)

        Pick a new, random angle

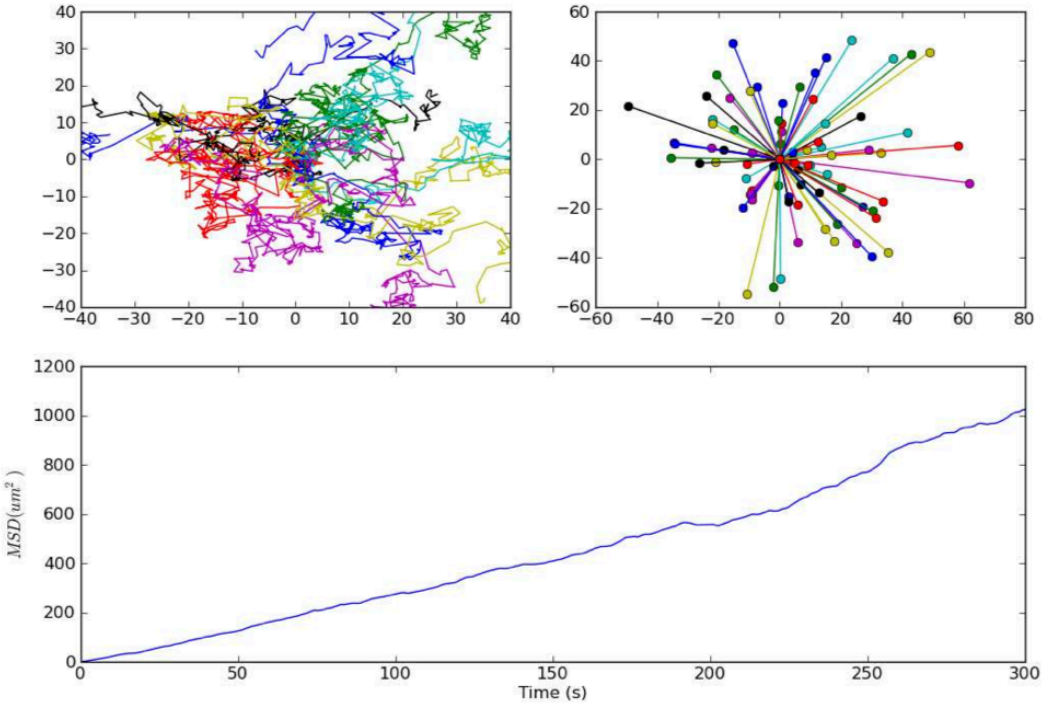
    Else

        move speed*dt in direction angle

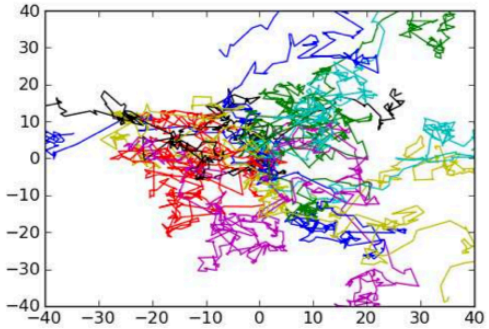
Simulation

- Launch 100 particles from (0,0) with initial, random angles
- Run event has a half life of 1 sec
  - p_tumble(1 sec) = 0.5
- Simulate for many timesteps over a 300 seconds

Simulation Results



Simulation Results



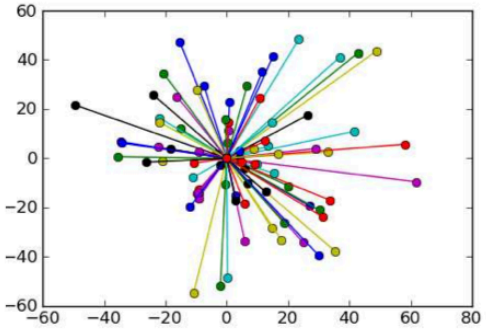
`pyplot.subplot(221)`

Multiple trajectories shown on one plot

Simulation Results

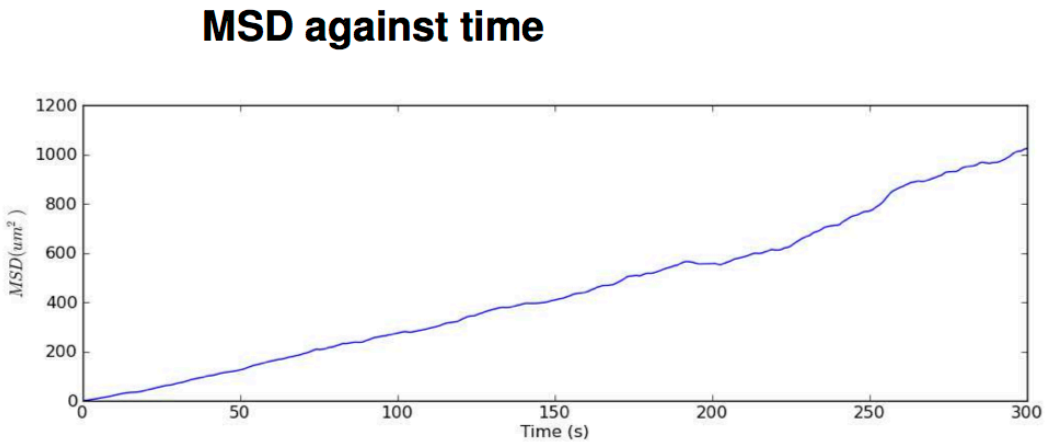
```
pyplot.subplot(222)
```

Simplified trajectories showing only initial and final positions



Simulation Results

```
pyplot.subplot(212)
```



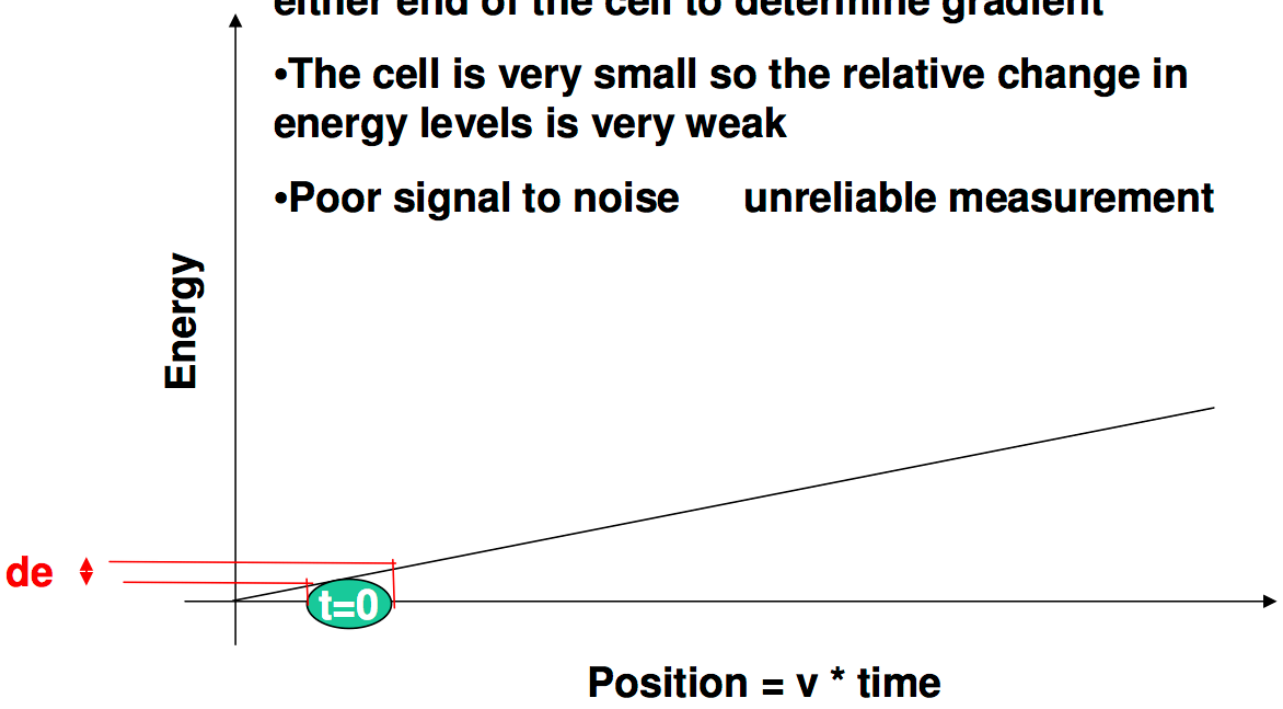
Chemotaxis

Hungry, Hungry bacteria

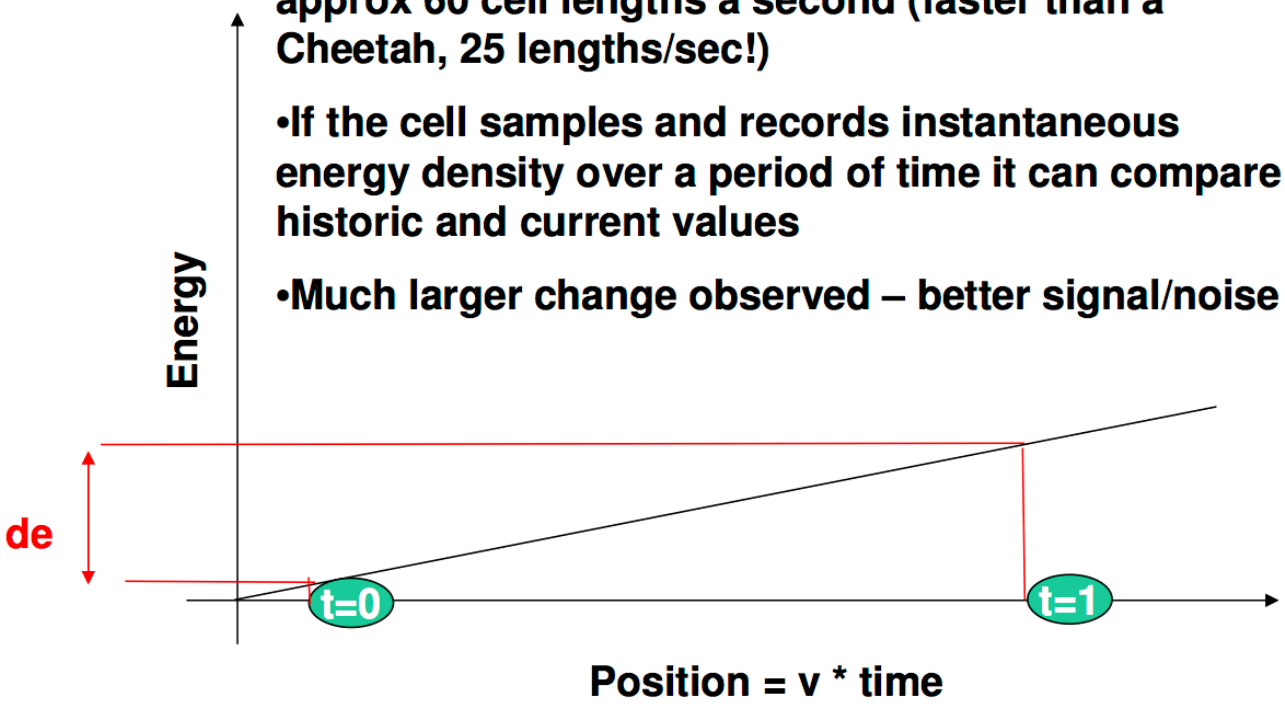
Hungry

- We have a simple model of the process of bacterial motion
- Bacteria need to consume external sources of energy to live and reproduce
- A random walk isn't a very efficient method of finding that food!

- The bacteria could (somehow) have sensors at either end of the cell to determine gradient
- The cell is very small so the relative change in energy levels is very weak
- Poor signal to noise unreliable measurement



- The bacteria moves very fast in 'scale speed' – approx 60 cell lengths a second (faster than a Cheetah, 25 lengths/sec!)
- If the cell samples and records instantaneous energy density over a period of time it can compare historic and current values
- Much larger change observed – better signal/noise



Chemotaxis

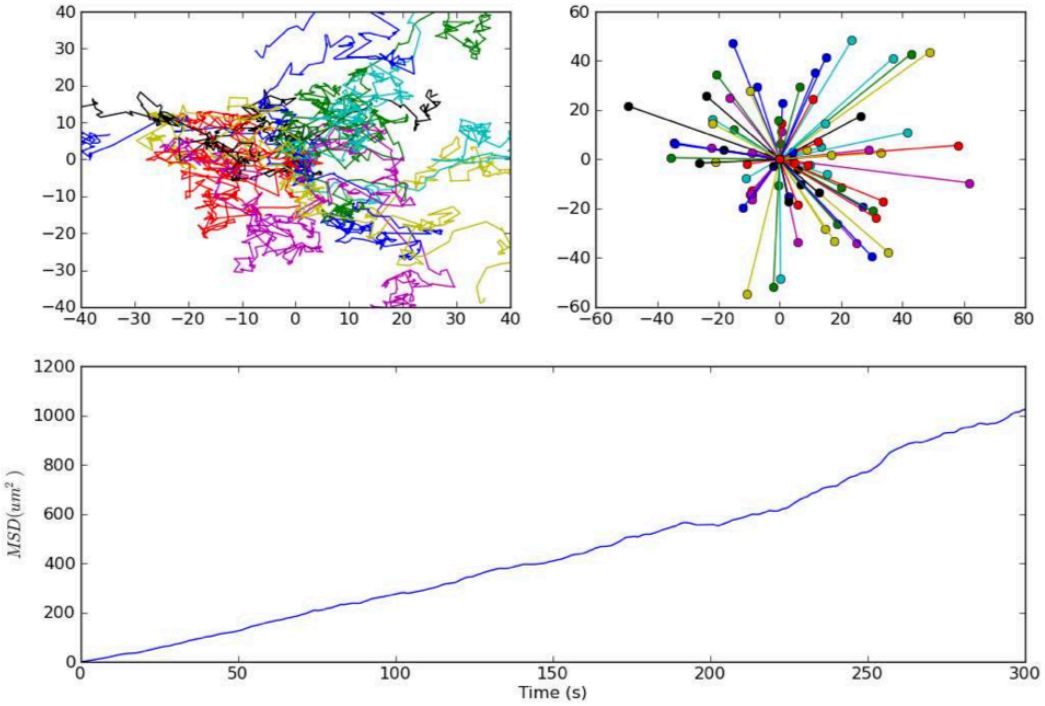
- Bacteria are to simple to develop a coordinated approach to hunting food
- Instead they modulate the behavior of their random walk to make it more likely that they walk towards food
- Probability of tumbling relates to de/dt – rate of change of energy with time
  - – Increasing energy: less likely to tumble
  - – Decreasing energy: more likely to tumble

Weekly Assessment Hints

- You need to keep track of historic energy levels to calculate the differential
- Use a Python list as a *shift register*
  - *See blackboard*
  - See live example

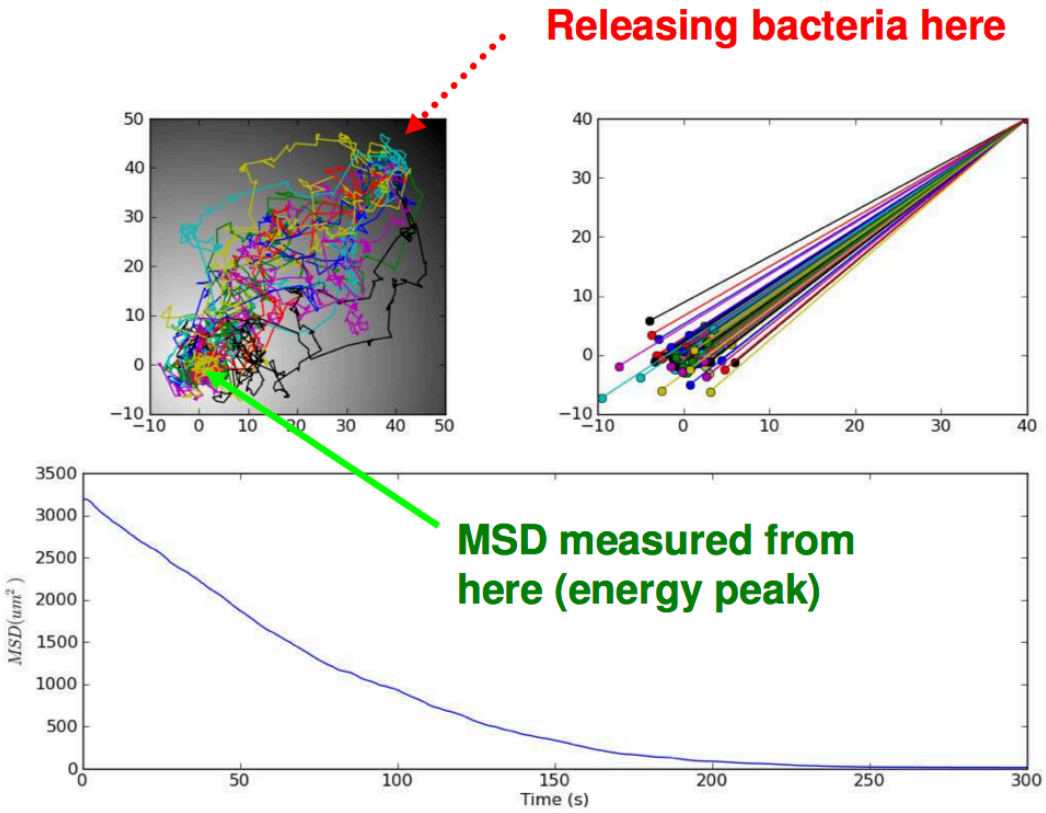
Chemotaxis in action

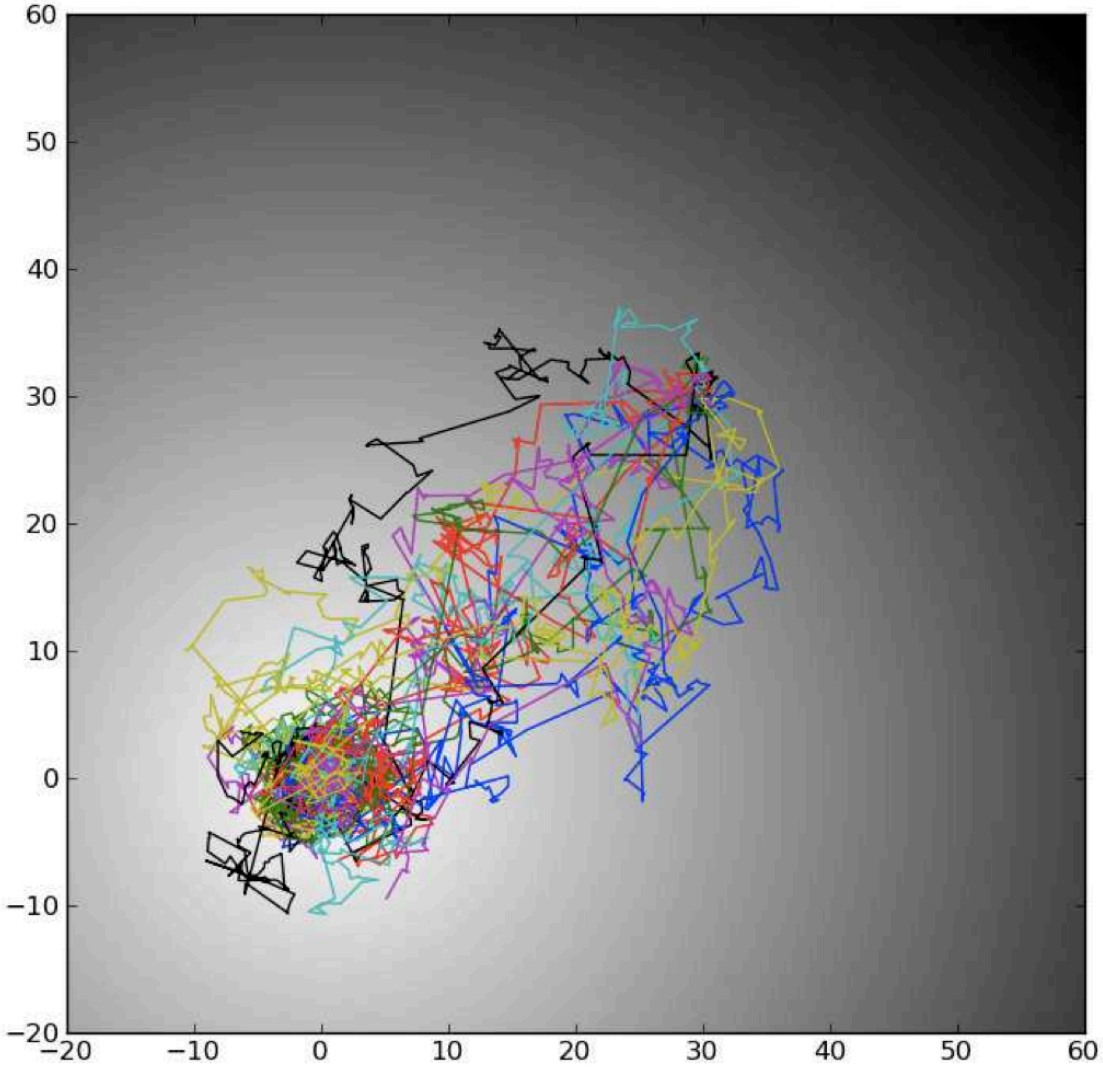
No energy field



** Field is different to the weekly assessment **

$$\text{Field} = 200-(x^2+y^2)^{.5}$$




60
50
40
30
20
10
0
−10
−20
−20
−10
0
10
20
30
40
50
60