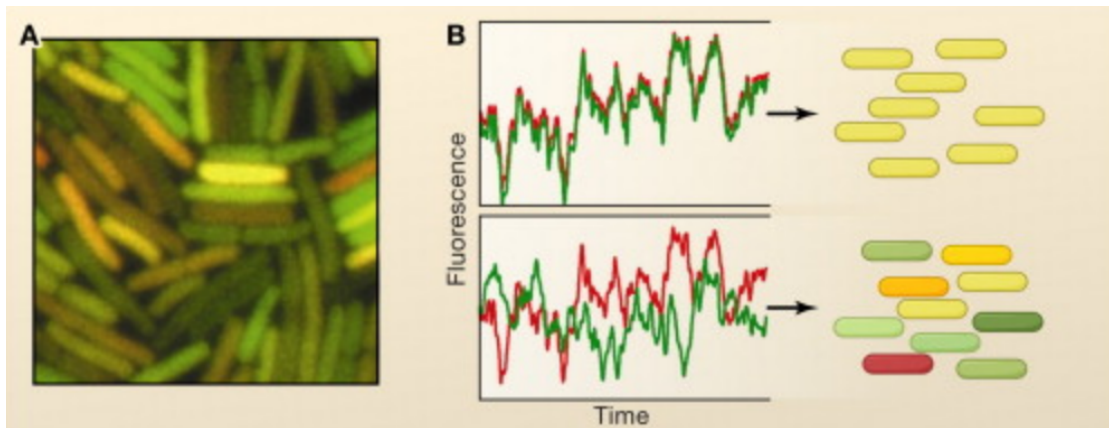


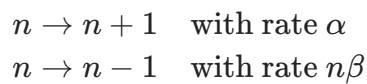
Low concentrations and stochasticity

We previously discussed that nano-molar concentrations correspond to 1 molecule per bacterium, micro-molar hence to 1000 molecules per bacterium. Signaling molecules in particular are often at low copy numbers and low numbers come with substantial stochasticity (or noise). Such noise due to low numbers is very common at the cellular levels. In these situations, it is important to think about numbers rather than concentrations!



Raj et al, Cell

Let's consider a simple model where molecules of a certain type are produced and decay:



This is a molecule driven version of the equation

$$\frac{dn}{dt} = \alpha - \beta n$$

When $\alpha = n\beta$ production and decay cancel each other out on average. We therefore expect that the mean number of molecules is

$$\bar{n} = \frac{\alpha}{\beta}$$

This is the fixpoint we discussed previously.

Side-note: Random number generation

Computers can't generate truly random numbers, but there are many libraries that generate pseudo-random numbers which are good enough for us

In [1]: `import numpy as np`

```
# numpy has a submodule called 'random'  
# the following generates a random number between 0 and 1  
print("One random number in [0-1):", np.random.random())  
  
print("5 random numbers in [0-):", np.random.random(size=5))
```

One random number in [0-1): 0.8119682650391323

5 random numbers in [0-): [0.49758914 0.46161446 0.53698254 0.03650752 0.27624581]

In [2]: `# there are many functions within this module`

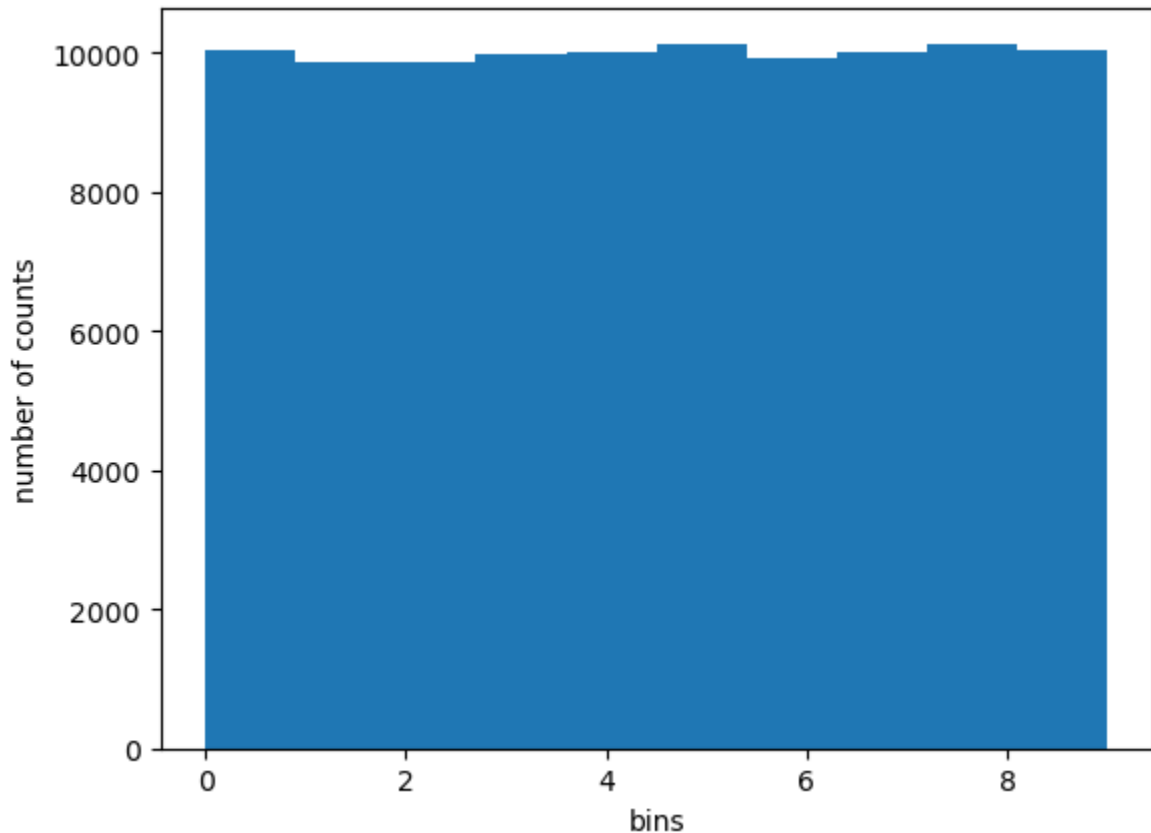
```
print("Random integer between 0,10:", np.random.randint(0,10))
```

Random integer between 0,10: 7

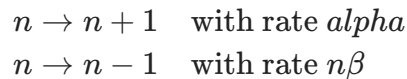
In [3]: `# a histogram of random numbers between 0 and 10 confirms this is uniform`
`import matplotlib.pyplot as plt`

```
n = 100000  
random_numbers = np.random.randint(0,10,size=n)  
y, x, lines = plt.hist(random_numbers, bins=10)  
plt.ylabel('number of counts')  
plt.xlabel('bins')  
print("counts in bins",y)
```

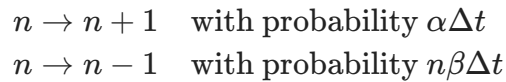
counts in bins [10038. 9875. 9868. 9975. 10016. 10120. 9939. 10003. 10135. 10031.]



Back to our simple model



We will simulate this using random number generators! To do so, let's consider the changes to n when proceeding in small time steps Δt . If these steps are small, most of the time nothing happens, but



```
In [4]: n = 100
a = 7 # per second
b = 1 # per second and molecule
Delta_t = 0.01
t_max = 100
t = 0

trajectory = [n]

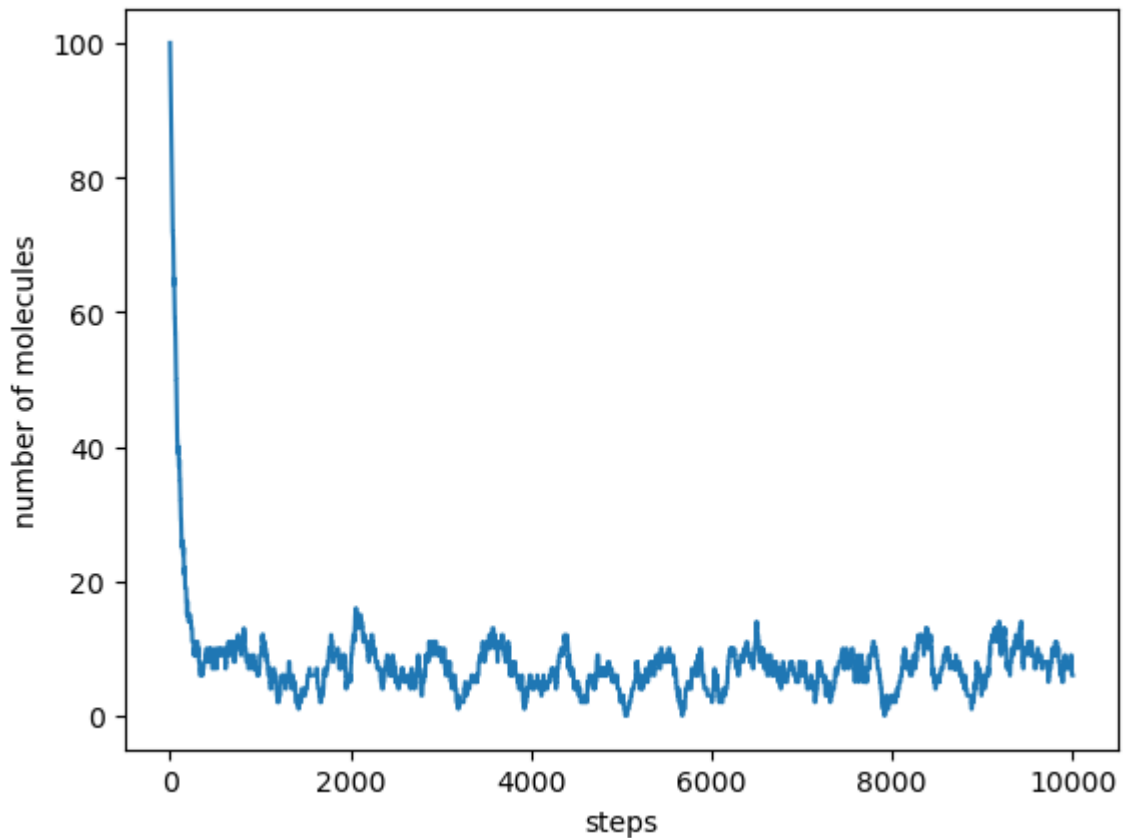
while (t < t_max):
    p = np.random.random() # random number between 0 and 1
    if (a*Delta_t > p):
        n += 1

    p = np.random.random() # random number between 0 and 1
    if (n*b*Delta_t > p):
        n -= 1

    t += Delta_t
    trajectory.append(n)

plt.plot(trajectory)
plt.ylabel('number of molecules')
plt.xlabel('steps')
print("Mean:", np.mean(trajectory))
```

Mean: 7.714228577142285



The simulation above makes a few approximations and for example assumes that in one time step Δt at most one molecule is produced or destroyed. This is not a big problem if the probability of these events is low, but something one has to keep in mind!

The problem is simple enough, that we can actually solve it exactly. What we would like to know is the probability $p(n)$ to observe n molecules at a randomly chosen time. At steady state, the probability of being at state $n - 1$ and moving to state n should be balanced by the reverse:

$$p(n - 1)\alpha = \beta n p(n) \quad \Rightarrow \quad p(n) = \frac{\alpha}{\beta} \frac{1}{n} p(n - 1)$$

This condition has a simple solution:

$$p(n) = \frac{(\alpha/\beta)^n}{n!} e^{-\alpha/\beta}$$

This distribution is known as the Poisson distribution, where $n! = n(n - 1)(n - 2) \dots 1$ is the factorial.

It has mean $\bar{n} = \alpha/\beta$ and variance $\sigma^2 = \bar{n}$

Dig deeper:

- In the histogram of random numbers between 0 and 10, what are the fluctuations you expect in each bin? Explore how they depend on n ! If you know the answer, verify it by simulation!
- Explore what happens if you change b and d in the simulation? At what point does the simulation become inaccurate?
- Calculate the mean and variance of n analytically!

In []:

In []: