Model of Fibril Growth

This notebook aims to model the curli fibril growth. We will base the foundation on the following rate equations:

$$\frac{dm}{dt} = pA - k_e P(t) m(t)$$

$$\frac{dP}{dt} = pB$$

$$\frac{dM}{dt} = k_e P(t) m(t)$$

Where m is the CsgA monomer concentration, P is the molar concentration of Curli fibrils, M is the mass-concentration of Curli fibrils (counted in CsgA monomer masses). pA is the bacteria CsgA production rate, k_e is the elongation rate and pB is the bacteria CsgB production rate. The rate equations have an analytical solution if one assumes a homogenous concentration of all substances in the environment around the bacteria. This is unfortunately not the case. As we move further from the cell membrane, the concentration decreases. Also, elongation occurs at the endpoints of the Curli fibrils, far away from the cell membrane. We will therefore begin by deriving a function for computing the concentration gradient.

Simulating CsgA Concentration Gradient

Fiffusion of a spherical particle in a sphericly-symmetrical environment can be descried according to Flicks law:

$$\frac{\partial U(t,r)}{\partial t} = -D\left(\frac{\partial^2 U(t,r)}{\partial r^2} + \frac{2}{r}\frac{\partial U(t,r)}{\partial r}\right)$$

Here, D is the diffision constant and r is the distace from the origin. The numerical solution to this equation is inspired by this report and implemented using Brian E. Rose's approach.

In [1]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import scipy.constants as constants
from IPython.display import display, Math, Latex
```

$R0 = 380nm$ is the bacteria radius. $pA \approx 10^{-10}$, $k_e = pA * MToNperCell$ is the secretion rate of the protein in absolute numbers, $D = 82114\frac{nm^2}{s}$ is the diffusion constant of the protein. $dist(/nm)$ is the distance from the membrane we want to cover in our simulation, $T(/s)$ is the time we want to simulate, $J1$ is the spacial resolution, $timesteps$ is the number of timesteps we want to simulate, The code has a time complexity of $O(timesteps * J1^2)$. We will measure concentration in $\frac{mol}{nm^3}$.

In [2]:
```
dist = 500
MToNperCell = 6.022e23*1e-12 #NOT FOUND VALUE, ONLY COPIED FROM OTHER IGEM TEAM
R0 = 380
```

```
ke = 10**-10*MToNperCell
T = 5*3600
D = 82114
J1 = 50
timesteps = int(1e4)
J = J1
deltat = T/timesteps
deltax = dist/J
display(Math(r'J = %i' %J))
display(Math(r'D = %i \frac{s}{nm^2}' %D))
display(Math(r'\Delta x = %0.3f nm' %deltax))
display(Math(r'\Delta t = %0.3f s' %deltat))
dm3tonm3 = 1e24 #Converter from moles/nm3 to moles/dm3
```

$J = 50$

$D = 82114 \frac{s}{nm^2}$

$\Delta x = 10.000 nm$

$\Delta t = 1.800 s$

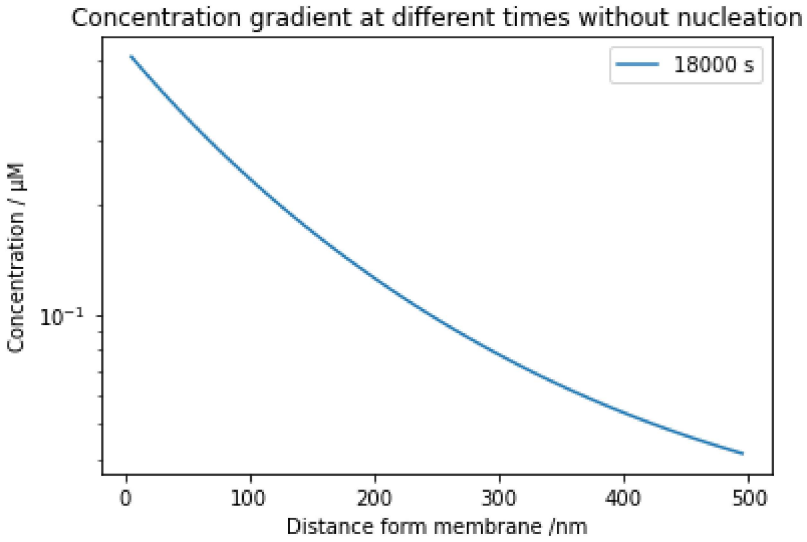
In [3]:
```
def getMatrix():
    """
    Generates the equation matrix A.
    """
    K = D*deltat/deltax**2
    r = lambda i : i*deltax+R0

    A = np.diag(([1 + 2*K/r(i)**2*(r(i)**2 + deltax**2) for i in range(J)]), 0)
    A += np.diag([-(r(i) + deltax)**2*K/r(i)**2 for i in range(J-1)], 1)
    A += np.diag([-(r(i) - deltax)**2*K/r(i)**2 for i in range(J-1)], -1)

    A[0,0] = 1 + K/r(0)**2*(r(0)**2 + deltax**2)
    A[-1,-1] = 1 + K/r(J)**2*(r(J)**2 + deltax**2)
    return np.linalg.inv(A)
def stepNoNucleation(u, a):
    """
    Performs one timestep. There is more CsgA produced every timestep, and then diff
    """
    u[0,0] += ke*deltat/constants.N_A*3/4/np.pi/(R0**2*deltax)
    u = np.matmul(a,u)
    return u
Ai = getMatrix()
```

Running this for the defined time yields the following graph:

In [4]:
```
U = np.zeros((J,1))
xstag = np.linspace(0., dist, J+1)
x = xstag[:-1] + deltax/2
for i in range(int(timesteps)):
    U = stepNoNucleation(U,Ai)
plt.plot(x,U[:,0]*dm3tonm3*10**6, label=str(T) + " s")
plt.title("Concentration gradient at different times without nucleation")
plt.xlabel("Distance form membrane /nm")
plt.ylabel('Concentration / \u03BCM')
plt.yscale('log')
plt.legend()
plt.savefig("concentration gradient.png")
plt.show()
```



Simulation of Fibril Growth

We use the CsgB production rate pB to estimate the number of new fibrils per unit of time. For every fibril i we will keep track of its endpoint distance from the cell membrane r_i and its direction in relation to the r-axis α_i. At every timestep, we will do the following:

- For every fibril j
  - Draw the a random number of elongations from the poisson distribution with a size computed from the rate equations and the concentration at r_j
  - Elongate the fibril accordinly and adjust the concentration gradient according to the reduction of CsgA monomers at r_j
- Draw a random number of new fibrils that will form from the poisson distribution of size pB .
- Create that many new fibrils and compute the new shape of the concentration gradient. When elongating, we compute the new location of the endpoint using the length of the CsgA monomer, $l = 4nm$, and the standard angular deviation of the fibril per elongation $\sigma = 3.47^\circ$ [igem Delf 2014]

  The timecomplexcity is $O(timesteps * (0.1T + J1^2))$

  Increased resolution mainly comes with increasing timesteps and not with increasing $J1$

In [5]:
```
pB = 1.3e-13*MToNperCell
kplus = 1.4*10**6*dm3tonm3
```

In [6]:
```
class Fibril:
    sigma = 3.47/180*np.pi
    unitL = 4.0
    def __init__(self):
        self.pos = self.unitL
        self.alpha = 0.
        self.size = 0
        self.index = int(self.pos // deltax)

    def __lt__(self,other):
        if self.pos < other.pos:
            return True
        return False
```

```
    def elongate(self):
        self.alpha += np.random.normal(0,self.sigma)
        self.pos += self.unitL*np.cos(self.alpha)
        self.size += 1
        self.index = int(self.pos // deltax)
```

In [7]:
```
def stepNucl(cGradient, endpoints, mProfile):
    newFibrils = np.random.poisson(pB*deltat)
    mProfile[0,0] += newFibrils
    for f in endpoints:
        i = f.index
        xpos = i*deltax
        dV = 4/3*np.pi*(xpos+R0)**2*deltax
        nMonomers = cGradient[i,0]*constants.N_A*dV
        if nMonomers > 1:
            nNucleations = np.random.poisson(kplus*cGradient[i,0]*deltat)

            if nNucleations > nMonomers:
                nNucleations = np.floor(nMonomers)
            for _ in range(int(nNucleations)):
                p0 = f.pos
                i0 = f.index
                f.elongate()
                p1 = f.pos
                if f.index == i0:
                    mProfile[f.index,0] += 1
                else:
                    mProfile[f.index,0] += (p1 % deltax)/(abs(p1-p0))
                    mProfile[i0,0] += (deltax - (p0 % deltax))/abs(p1-p0)

            nMonomers -= nNucleations
            cGradient[i,0] = nMonomers/constants.N_A/dV

    endpoints += [Fibril() for _ in range(newFibrils)]
    cGradient = stepNoNucleation(cGradient, Ai)
    return cGradient, endpoints, mProfile
```

We can now finally simulate how the fibrils will have grown.

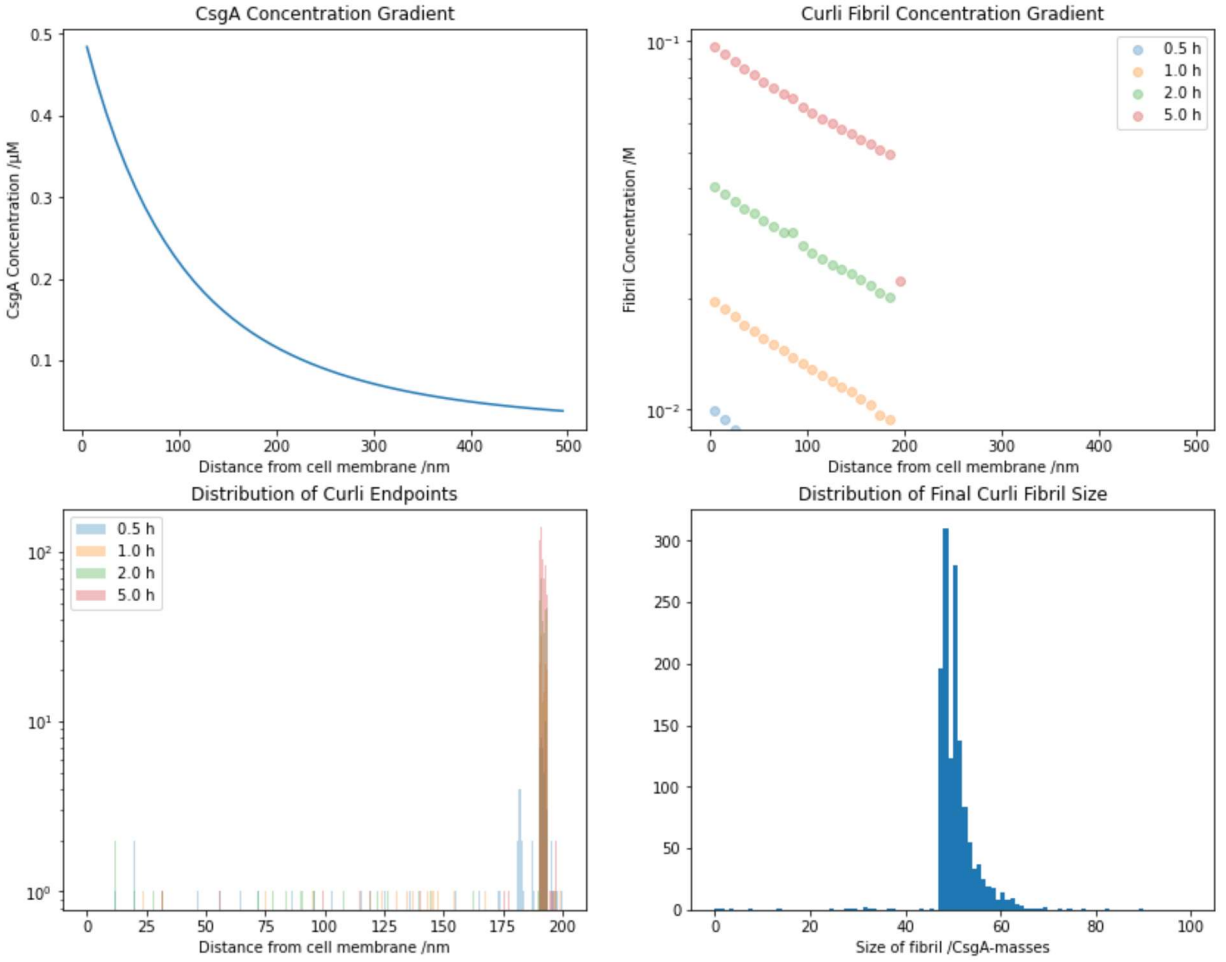
In [8]:
```
cGradient = np.zeros((J,1))
endpoints = []
mProfile = np.zeros((J,1))
xstag = np.linspace(0., dist, J+1)
x = xstag[:-1] + deltax/2

fig, axs = plt.subplots(2,2, figsize=(14,11))

for i in range(timesteps):
    cGradient, enpoints, mProfile = stepNucl(cGradient, endpoints, mProfile)
    if i*deltat in [60*30, 3600, 2*3600]:
        axs[1,0].hist([j.pos for j in endpoints], bins=500, range=(0,200), label = s
                      alpha = 0.3)
        axs[0,1].scatter(x, [mProfile[j,0]*10**24/constants.N_A*3/4/np.pi/(R0+deltax
                         label = str(i*deltat / 3600) + " h", alpha = 0.3)


axs[0,0].plot(x,cGradient*10**30)
axs[0,0].set_xlabel("Distance from cell membrane /nm")
axs[0,0].set_ylabel("CsgA Concentration /\u03BCM")
axs[0,0].set_title("CsgA Concentration Gradient")
axs[0,1].scatter(x, [mProfile[i,0]*10**24/constants.N_A*3/4/np.pi/(R0+deltax*i)**2*d
                 label= str(T / 3600) + " h", alpha=0.3)
axs[0,1].set_xlabel("Distance from cell membrane /nm")
axs[0,1].set_ylabel("Fibril Concentration /M")
```

```
axs[0,1].set_title("Curli Fibril Concentration Gradient")
axs[0,1].legend()
axs[0,1].set_yscale('log')
axs[1,0].hist([i.pos for i in endpoints], bins=500, range=(0,200), label= str(T / 36
axs[1,0].set_xlabel("Distance from cell membrane /nm")
axs[1,0].set_title("Distribution of Curli Endpoints")
axs[1,0].legend()
axs[1,0].set_yscale('log')
axs[1,1].hist([i.size for i in endpoints], bins=100, range=(0,100))
axs[1,1].set_xlabel("Size of fibril /CsgA-masses")
axs[1,1].set_title("Distribution of Final Curli Fibril Size")
plt.show()
```



In [ ]: