Problem set 2, Task 3
Synchronisation
Computational Biology
FFR110/FIM740

Mattias Berg, Konstantinos Zakkas

February 26, 2021

a.)

From lecture note 9 (CompBioLecture9.pdf), we have the following formula on page 52,

$$\gamma = K\gamma \int_{-\pi/2}^{\pi/2} cos^2(\Theta) g(Krsin\Theta)\, d\Theta$$

$$1 = K \int_{-\pi/2}^{\pi/2} cos^2(\Theta) g(Krsin\Theta)\, d\Theta \tag{1}$$

we start with finding K_c by letting r go towards zero in equation 1,

$$K_c = \frac{1}{\int_{-\pi/2}^{\pi/2} cos^2(\Theta) g(0)\, d\Theta}$$

as $\int_{-\pi/2}^{\pi/2} cos^2(\Theta)\, d\Theta = \pi/2$ and $g(0)$ dont depend on Θ we get the following,

$$K_c = \frac{2}{\pi g(0)}$$

$$g(0) = \frac{\gamma}{\pi(0^2 + \gamma^2)} = \frac{1}{\pi\gamma}$$

$$K_c = \frac{2}{\pi \frac{1}{\pi\gamma}} = 2\gamma$$

Now that we have K_c we want to solve the equation for r. We do this by using the following expansion (expansion in small r) of the previous equation (also found in lecture note 9 (CompBioLecture9.pdf)).

$$1 = \frac{K}{K_c} + \frac{K(Kr)^2}{2} g''(0) \int_{-\pi/2}^{\pi/2} cos^2(\Theta) sin^2(\Theta)\, d\Theta \tag{2}$$

where $\int_{-\pi/2}^{\pi/2} cos^2(\Theta) sin^2(\Theta)\, d\Theta$ is equal to $\pi/8$. Before solving it we need to find what $g''(0)$ is,

$$\begin{aligned}
g &= \frac{\gamma}{\pi(w^2+\gamma^2)} = \frac{1}{\pi\gamma} \\
\frac{dg}{dw} &= -\frac{2w\gamma}{\pi(w^2+\gamma^2)^2} \\
\frac{d^2g}{dw^2} &= -\frac{2\gamma(\gamma^2-rw^2)}{\pi(w^2+\gamma^2)^3} \\
g''(0) &= -\frac{2\gamma(\gamma^2-r0^2)}{\pi(0^2+\gamma^2)^3} = -\frac{2\gamma^3}{\pi\gamma^6} = -\frac{2}{\pi\gamma^3}
\end{aligned}$$

we now get the following,

$$\begin{aligned}
1 &= \frac{K}{K_c} + \frac{K(Kr)^2}{2}\frac{-2}{\pi\gamma^3}\frac{\pi}{8} \\
1 &= \frac{K}{K_c} - \frac{K^3r^2}{8\gamma^3} \Rightarrow (\text{ we set } \gamma \text{ to } \frac{K_c}{2}\ ) \Rightarrow \frac{K}{K_c} - \frac{K^3r^2}{8\frac{K_c^3}{2^3}} \\
1 &= \frac{K}{K_c} - \frac{K^3r^2}{K_c^3} \\
\frac{K^3r^2}{K_c^3} &= -1 + \frac{K}{K_c} \\
K^3r^2 &= \frac{(K-K_c)K_c^3}{K_c} \\
r^2 &= \frac{(K-K_c)K_c^2}{K^3} \\
r &= \sqrt{\frac{(K-K_c)K_c^2}{K^3}} \qquad (3)
\end{aligned}$$

we now use equation 3 combined with the equations gain in the question, $r = C\sqrt{\mu}$ and $0 < \mu = (K-K_c)/K_c << 1$ and gain the following

$$\begin{aligned}
C\sqrt{(K-K_c)/K_c} &= \sqrt{\frac{(K-K_c)K_c^2}{K^3}} \\
C &= \sqrt{\frac{K_c^3}{K^3}} \qquad (4)
\end{aligned}$$

we assume $\gamma = 1$ and therefor $K_c = 2\gamma = 2$, and we gain the following answer for C,

$$C = 2\sqrt{\frac{2}{K^3}}$$

b.)

Three cases were chosen to be tested, $K_1 = K_c/2 = 1$ (below K_c), $K_2 = K_c * 1.01 = 2.02$(close to K_c) and $K_3 = K_c * 2 = 2$ (above K_c). If we use the mean-field theory, we gain the 3 following estimates of r,

$$r_1 = \sqrt{\frac{(K_1 - K_c)K_c^2}{K_1^3}} = \sqrt{\frac{(1-2)2^2}{1^3}} = \sqrt{-4} \Rightarrow \ r_1 \text{ has no real part}$$

$$r_2 = \sqrt{\frac{(K_2 - K_c)K_c^2}{K_2^3}} = \sqrt{\frac{(2.02-2)2^2}{2.02^3}} = \sqrt{\frac{0.02*4}{8.242408}} \approx 0.0097$$

$$r_3 = \sqrt{\frac{(K_3 - K_c)K_c^2}{K_3^3}} = \sqrt{\frac{(4-2)2^2}{4^3}} = \sqrt{\frac{8}{64}} = 0.125$$

When comparing it to the simulated version in Figure 1, it seems like the mean-field approximation is lower than the simulated value (for higher N), where in the simulation K=2.02, the r hovers around 0.2 vs the approximations 0.0097, and for K=4, the simulation hovers around 0.7 vs the approximations 0.125.

It is a big difference in the noise for low values of N versus large values of N, where the larger N values give much more stable output. But it also generates more computational time needed.

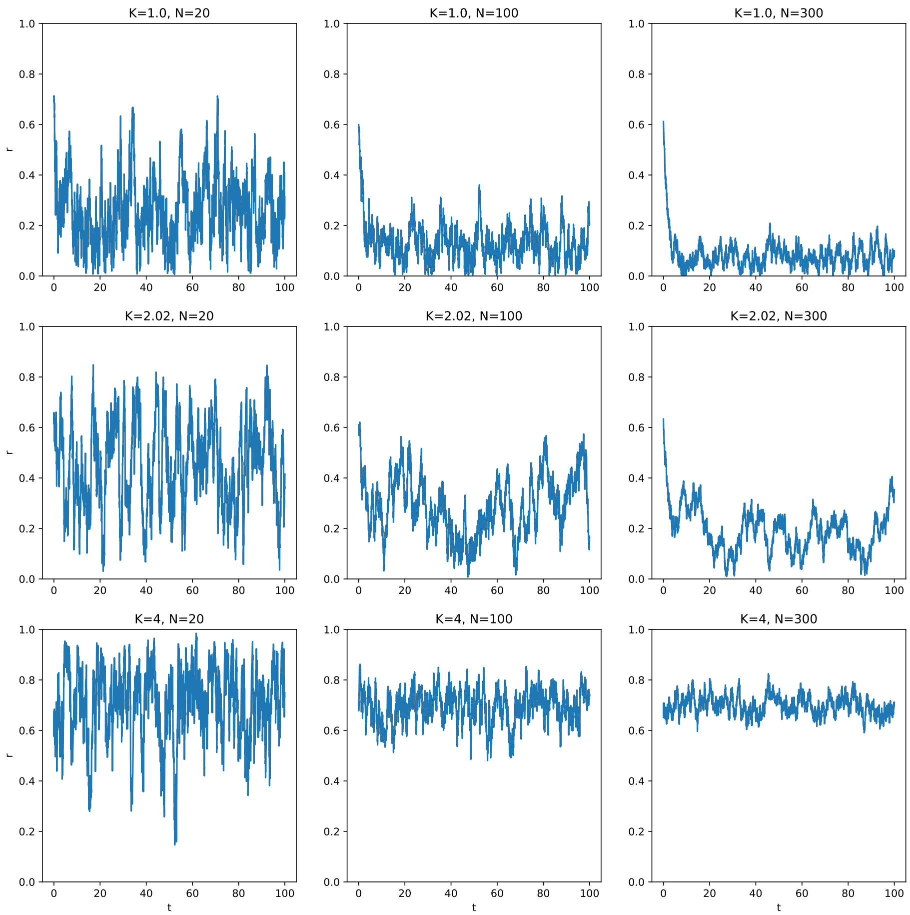

Figure 1: The plots shows all the different cases simulated. The time steps are set to 0.01 for all the simulations.

Appendix

```
#Course : FFR110/FIM740 Computational Biology
#Problem : Problem set 2, Task 3. Synchronisation
#Code : Python 3.8.5

import numpy as np
import matplotlib.pyplot as plt
import math

def Calc_Kuramoto(K, N, Time, dTime):

    Theta_prev = (np.random.rand(N)*2-1)*np.pi/2
    Theta_current = np.zeros(N)

    r=np.zeros(int(Time/dTime))

    for t in range(int(Time/dTime)):

        w_i=np.random.standard_cauchy(N)

        for i in range(N):
            Theta_current[i]=np.sum(np.sin(Theta_prev-Theta_prev[i]))

        Theta_current[:]=Theta_prev[:]+dTime*(w_i[:]+Theta_current[:]*K/N)

        r[t]=1/N*np.sqrt((np.sum(np.cos(Theta_current)))**2 + (np.sum(np.sin(Theta_current)))**2)

        Theta_prev[:]=Theta_current

    return r


Kc=2
Time=100
dTime=0.01

KList=[Kc/2, Kc*1.01, Kc*2]
NList=[20, 100, 300]

t = np.arange(0, Time, dTime)


fig, axs = plt.subplots(3, 3, figsize=(15,15))

counter=0
for K in range(len(KList)):
    axs[K, 0].set_ylabel('r')
    for N in range(len(NList)):
        r = Calc_Kuramoto(KList[K], NList[N], Time, dTime)

        rMeanCalc=(KList[K]-Kc)*(Kc**2)/(KList[K]**3)
        if rMeanCalc > 0:
            rMean=np.sqrt(rMeanCalc)
        else:
            rMean=0
        axs[K, N].set_ylim([0, 1])
        titleString = "K=%s, N=%s" % (KList[K] ,NList[N])
        axs[K, N].title.set_text(titleString)
        axs[K, N].plot(t, r)

        if K==len(KList)-1:
            axs[K, N].set_xlabel('t')


plt.savefig("3_Synchronisation_output_test.pdf", bbox_inches = 'tight')
plt.show()
```