바이오시스템해석

2022011038 기계공학과

장대윤

**1. Hill’s muscle model**

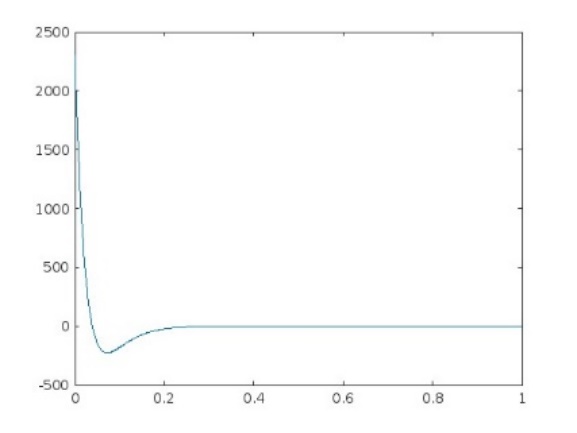
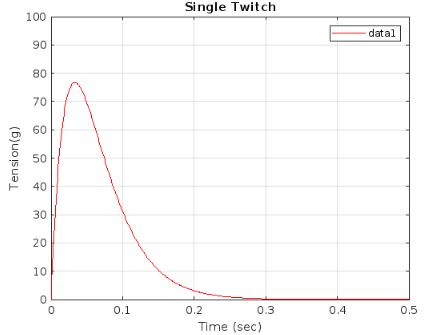
**a) Calculate the muscle force T(t) by numerically solving O.D.E. given above, by using matlab**  
  
Matlab의 ode45 function을 사용하여 0.5 sec 까지 그 값을 그려, Single Twitch의 형상을 확인해 볼 수 있었다. Action Potiential의 경우 하강하면서 음의 값을 가지는데, AP의 적분값으로 구성되는 T(t)의 양상이 이를 잘 표현해줌을 확인할 수 있다.  
  
 

Figure 1. Action Potential Figure 2. Tension Graph (g / sec)

% Problem 1 - a

clear all; clc; close all;

[t,y] = ode45(@(t,y) hill\_muscle\_model\_ODE(t,y), 0:0.001:1, 0);

A = 48144 \* exp(-t/0.0326) - 45845 \* exp(-t/0.034);

A = A';

plot(t,A')

plot(t,y,'r')

title('Single Twitch')

xlabel('Time (sec)')

ylabel("Force(N, tension)")

legend("show")

xlim([0 0.5])

ylim([0 100])

grid on

function dydt = hill\_muscle\_model\_ODE(t,y)

Kse=136; Kpe=75; b=50;

A = 48144 \* exp(-t/0.0326) - 45845 \* exp(-t/0.034);

delta\_x=0;

dydt = (Kse/b) \* ( Kpe \* delta\_x + b\* delta\_x - (1+Kpe/Kse)\*y + A );

end

**a) Calculate the muscle force T(t) by numerically solving O.D.E. given above, by using matlab**  
Using the acquired muscle force T(t) from a) and b), discuss at what frequency you can find the fused tentanus Action Potential을 주파수를 변경하며 주었을 때의 그래프 변화는 아래 그래프와 같다.

> 주파수 별 그래프

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Frequency = 1 Hz | Frequency = 3 Hz | Frequency = 10 Hz |
|  |  |  |
| Frequency = 30 Hz | Frequency = 100 Hz | Frequency = 150 Hz |
|  |  |  |
| Frequency = 170 Hz | Frequency = 200 Hz |  |

주파수에 따라 AP가 겹칠 경우 증폭되면서 처음엔 single twitch였다가 중첩이되는 과정동안 unfused tentanus, 최종적으로는 fused tentanus가 되는 과정을 볼 수 있었다. 또한 모든 과정에서 1초 이후 AP가 없어졌을 때는 같은 모양으로 감소함도 확인할 수 있었다.  
  
아래 그림은 muscle force를 자세히 들여다 본 것이다. 150 Hz에서는 아직 진동형 그래프가 그려졌으며, 170 Hz와 200 Hz에서는 진동이 비슷하게 작음을 볼 수 있었다. 따라서 tentanus의 주파수는 약 170 Hz 라고 판단되어진다.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Frequency = 150 Hz | Frequency = 170 Hz | Frequency = 200 Hz |

> 소스 코드

% Problem 1 – b)

clear all; clc; close all;

[t,y] = ode45(@(t,y) hill\_muscle\_model\_ODE(t,y), 0:0.001:1, 0);

A = 48144 \* exp(-t/0.0326) - 45845 \* exp(-t/0.034);

A = A';

grid on

max\_freq = 300; % Hz

T = 1000/max\_freq; % ms

x\_n = 0:0.001:3;

input = y';

for i=1:1:max\_freq

new\_total = zeros(1, 3001);

r = round(1000/i);

for j=0:i-1

new\_total(1+j\*r : 1001+j\*r) = new\_total(1+j\*r : 1001+j\*r) + input(1:end);

end

if mod(i,10)==0

fprintf("# %d th", i)

figure(i)

plot(x\_n, new\_total(1:3001), 'r')

xlabel('Time (sec)')

ylabel("Force(N, tension)")

xlim([0 1.5])

ylim([0 max(new\_total)\*1.1])

end

grid on

end

function dydt = hill\_muscle\_model\_ODE(t,y)

Kse=136; Kpe=75; b=50;

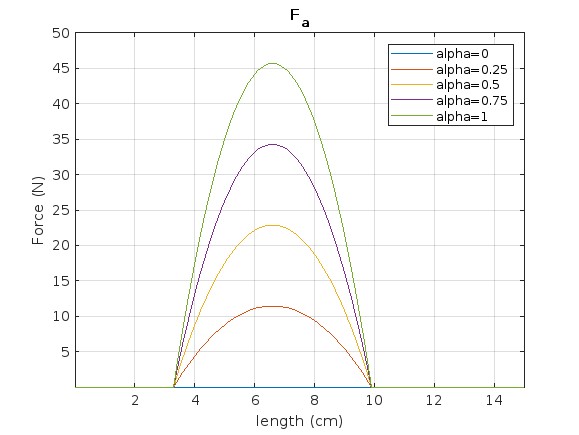
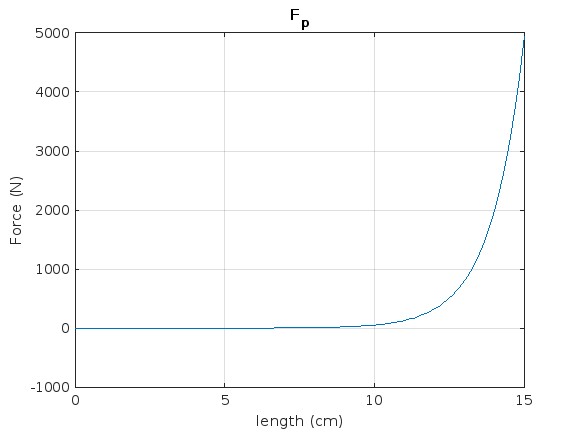
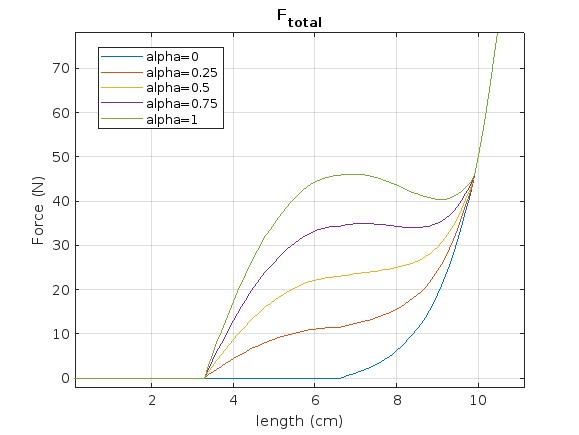
A = 48144 \* exp(-t/0.0326) - 45845 \* exp(-t/0.034);

delta\_x=0;

dydt = (Kse/b) \* ( Kpe \* delta\_x + b\* delta\_x - (1+Kpe/Kse)\*y + A );

end

**2. 3-element Windkessel model**

**a) Plot the force-length curve**    
α가 증가할수록 가 convex한 형태를 보이면서 증가한다. 이에 따라 의 값도 증가한다. 다만 가 약 length 10 cm 부근에서 증가함에 따라 우상향 형태의 곡선 그래프를 그리게 된다.

**b) Simulation the motion when input a(t)**

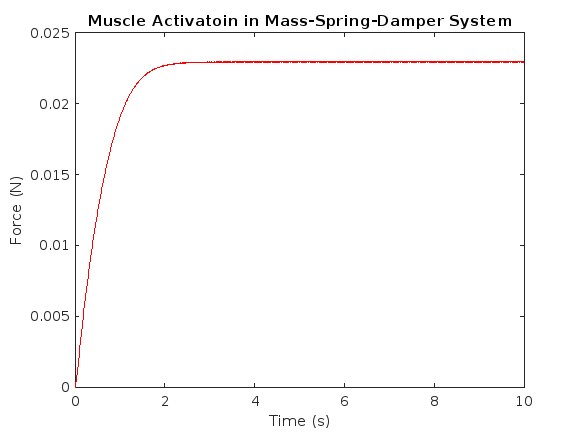
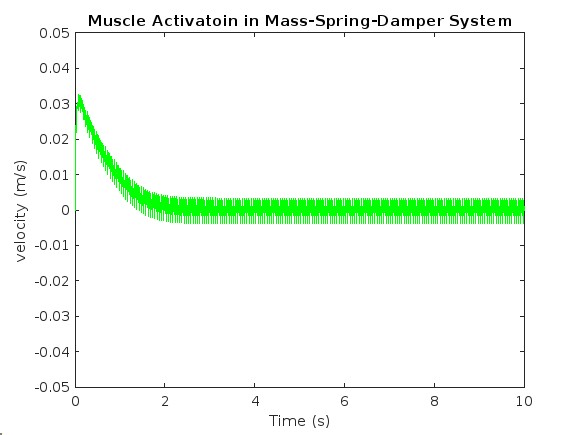
   
 Fig 1. Velocity of x Fig 2. Forceotime graph

Fig 1.은 time에 따른 x 위치의 속도 변화를 보여준다. 우측은 time에 따른 x의 가속도를 통해 구해낸 Force 결과이다. X의 속도는 0부근에서 진동을 하지만 우측의 Force 그래프 또한 직선처럼 나오는 것을 보아 약 2초 이후에는 Force가 Steady-State 상태(Stable)에 있음을 볼 수 있다.  
  
  
  
> 소스코드

% Problem 2-a)

clear; clc; close all;

alpha = 0:0.25:1;

F\_max = 45.7;

l\_0 = 6.6;

K\_sh = 3;

l = 0:0.1:15;

% there is no under Force(Voltage) in F\_a

F\_a = alpha' \* F\_max \* ( 1 - 4 \* ((l-l\_0)./l\_0).^2 );

F\_a = max(F\_a, 0);

% if l <= l\_0, F\_p = 0

% and in this code l = -y1 + l\_o, so it means y1 > 0

if l < l\_0

F\_p = 0;

else

F\_p = ( F\_max/(exp(K\_sh)-1) .\* (exp( K\_sh.\*(l-l\_0)./(0.5\*l\_0))-1) );

end

F\_total = F\_a + max(F\_p,0);

% plot

figure(1)

plot(l,F\_a)

title("F\_a")

xlabel('length (cm)')

ylabel('Force (N)')

legend('alpha=0', 'alpha=0.25', 'alpha=0.5', 'alpha=0.75', 'alpha=1')

grid on

figure(2)

plot(l,F\_p)

title("F\_p")

xlabel('length (cm)')

ylabel('Force (N)')

grid on

figure(3)

plot(l, F\_total)

title("F\_t\_o\_t\_a\_l")

xlabel('length (cm)')

ylabel('Force (N)')

legend('alpha=0', 'alpha=0.25', 'alpha=0.5', 'alpha=0.75', 'alpha=1')

xlim([0 11])

ylim([0 80])

grid on

% Problem 2 - b)

clear, clc, close all;

disp("solution of Problem 1 - b)")

x\_n = 0:0.01:10;

init\_y = [0 0];

[t, y] = ode45(@hill\_muscle\_model\_2ndODE, x\_n, init\_y);

% plot

figure(1)

plot(t, y(:, 1), 'r');

title('Muscle Activatoin in Mass-Spring-Damper System')

xlabel('Time (s)')

ylabel('Force (N)')

xlim([0 10])

ylim([0 0.025])

figure(2)

plot(t, y(:,2), 'g');

title('Muscle Activatoin in Mass-Spring-Damper System')

xlabel('Time (s)')

ylabel('velocity (m/s)')

xlim([0 10])

ylim([-0.050 0.05])

function dydt = hill\_muscle\_model\_2ndODE(t, y)

% initial parameters

m=12; k=510; b=210; F\_max=45.7; l\_0=0.066; K\_sh=3;

a1=0.8; a2=0.5; a3=0.43; a4=58;

% adjustable parameters

omega = 40;

y1 = y(1);

y2 = y(2);

factor\_function = a1 + a2\*atan(a3 + a4\*(-y2));

alpha = (sin(omega\*pi\*t))^2;

% there is no under Force(Voltage) in F\_a

F\_a = alpha \* F\_max \* (1-4 \* (y1/l\_0)^2);

F\_a = max(F\_a, 0);

% if l <= l\_0, F\_p = 0

% and in this code l = -y1 + l\_o, so it means y1 > 0

if y1 > 0

F\_p = 0;

else

F\_p = (F\_max/(exp(K\_sh)-1)) \* (exp( K\_sh.\*(-y1)/(0.5\*l\_0) )-1);

end

F\_total = ( factor\_function \* (F\_a + F\_p) - b \* y2 - k \* y1 ) / m;

dxdt = y2;

dxdt2 = F\_total;

dydt = [dxdt; dxdt2];

end

**4. Hodgkin-Huxley model**

**a) Find the threshold input I by trial-and-error**

Neuron에서 Resting potential은 약 60 ~ 70 mV의 값을 가지며, 우리는 60 mV의 값으로 시뮬레이션을 진행한다.

분석은 전류를 0.0000 mA부터 0.1 mA까지 0.0001 mA 간격으로 분석하였다. 이 때, membrane potential이 -55 mV가 넘어가는 순간을 A.P. 가 생성되는 것으로 보고 진행하였다.



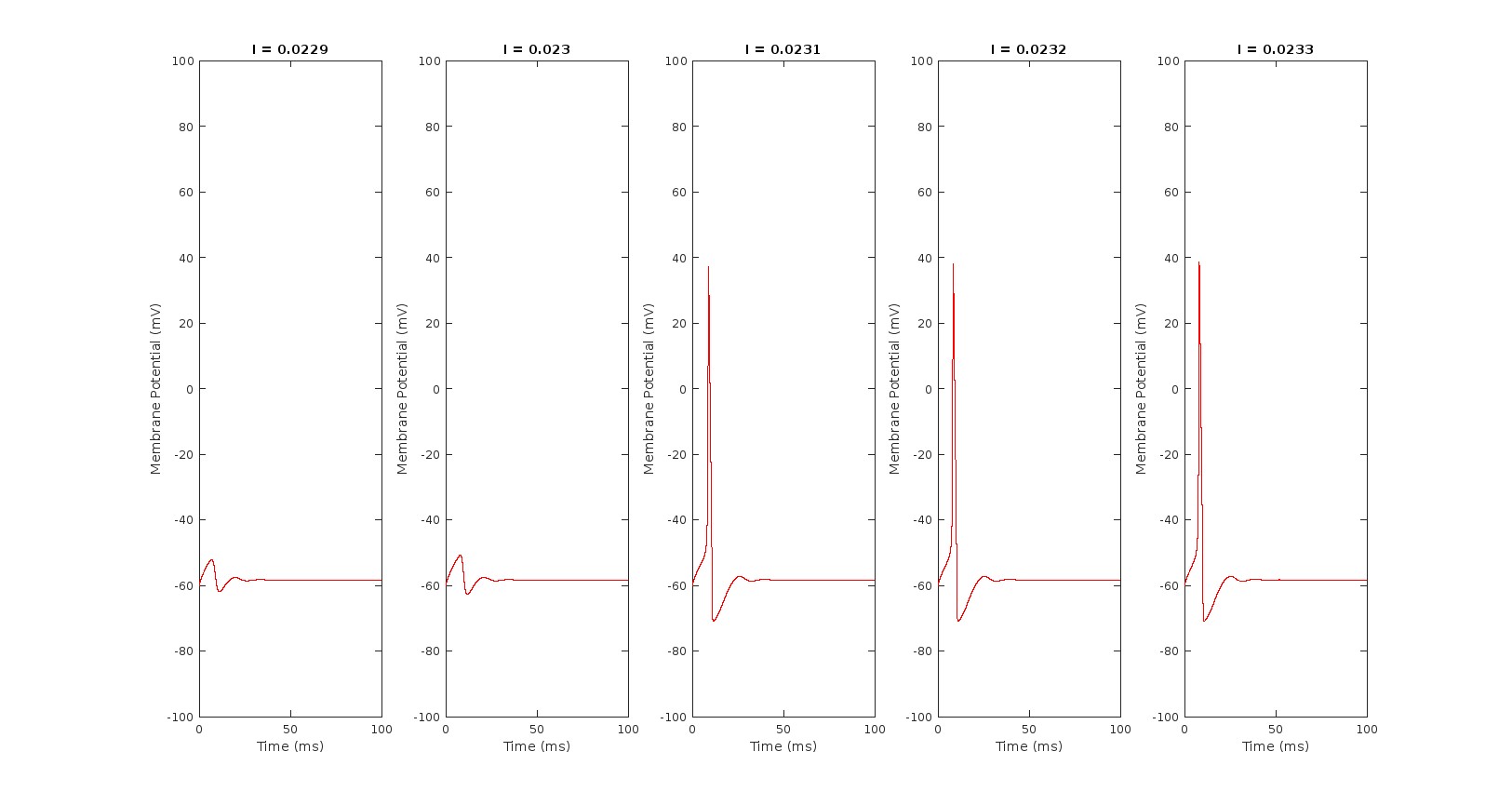
Fig 1. Threshold current(mA) and membrane potential voltage(mV)

A.P. 가 발생한 current는 약 0.0231 mA이며, 이 때 발생한 membrane potential의 전압은 약 37.1692 mV이다.

**b) Simulate the A.P**

a)에서 구한 전류 값(0.0231)을 기준으로 전 후 두 전류(0.0229, 0.0230, 0.0232, 0.0233) 에 대해서 membrane potential을 그래프로 그려보면 아래와 같다.

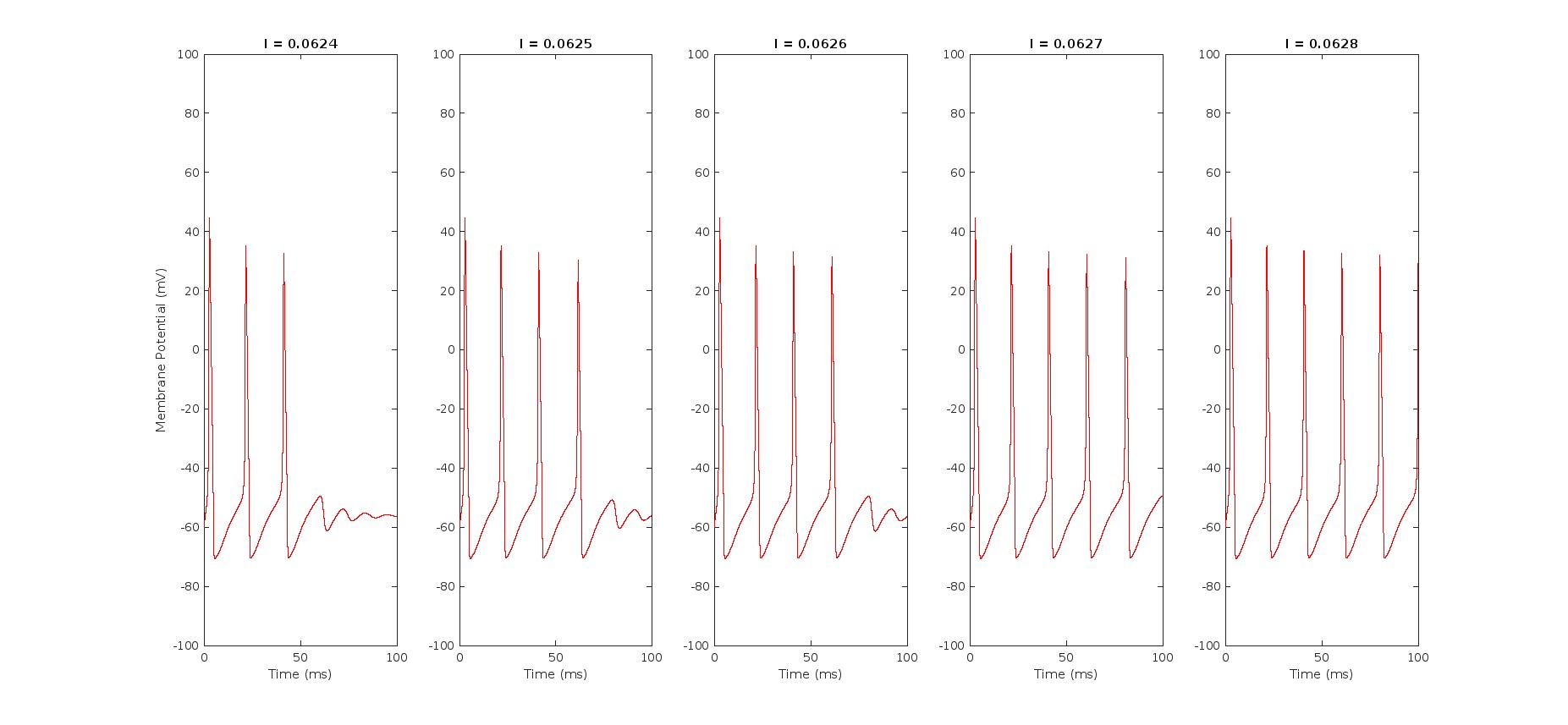
Threshold의 예상대로 A.P.가 생김을 육안으로 확인할 수 있었다.



> Stable State를 보여주는 영역

이는 최초 A.P.가 발생한 전류(0.0231 mA) 이후부터 동일한 간격으로 진행하여 A.P.가 5개 이상 나오는 순간의 전류값을 구한 것이다.





A.P.가 한 번 발생했다면, 이후 전류 값에서는 더 여러 번 A.P갈 발생하거나 상승 곡선을 그리다가 일정한 전압으로 출력이 될 것이라고 생각하고 시뮬레이션을 진행하였다.

위의 값과 그래프를 보면 0.0626 mA에서 A.P.가 네 번 생성 되었으며, 0.0627 mA에서 A.P.가 다섯 개 이상 만들어짐을 볼 수 있었다. 즉 0.063 mA 부근에서 주기적으로 A.P.가 생성됨을 알 수 있었다.

추가로 여러 번 A.P.가 발생했을 때의 그 전압은 약 45 mV 수준으로 그 이상 잘 올라가지 않는 현상을 보이고 있다.

> 소스코드

% Problem 4-a)

clear all; clc; close all;

% initial values

v\_rest = 60;

v\_0 = - v\_rest;

% when t=0, dn/dt = dm/dt = dh/dt = 0

n\_0 = alpha\_n(v\_0)/(alpha\_n(v\_0)+beta\_n(v\_0));

m\_0 = alpha\_m(v\_0)/(alpha\_m(v\_0)+beta\_m(v\_0));

h\_0 = alpha\_h(v\_0)/(alpha\_h(v\_0)+beta\_h(v\_0));

y\_0 = [v\_0; n\_0; m\_0; h\_0];

x\_n = 0:0.001:100;

current\_lim = 0.0000:0.0001:0.1000;

max\_voltage = 0;

max\_voltage\_plot = 0;

index\_top\_voltage = 0;

value\_top\_voltage = 0;

flag\_find\_top\_voltage = true;

% for monitoring the proceeding status

waitbar\_process = waitbar(0, 'Process ...');

disp("iteration " + string(length(current\_lim)))

prev\_top\_voltage = 0;

% current\_top\_voltage = 0;

for i=1:1:length(current\_lim)

disp(i)

current=current\_lim(i);

disp(current)

[t,y] = ode45( @(t,y) HodgkinHuxley\_ODE(t,y,current), x\_n, y\_0 );

max\_voltage = max(y(:,1));

disp(max\_voltage)

current\_top\_voltage = max\_voltage;

% for find the top voltage -> flag have the index of OUTPUT Volatage

if i==1

prev\_top\_voltage = max\_voltage;

else

if current\_top\_voltage > -50

if flag\_find\_top\_voltage == true

index\_top\_voltage = i;

max\_voltage\_plot = y(:,1);

prev\_max\_voltage\_plot = max\_voltage;

value\_top\_voltage = current\_top\_voltage;

flag\_find\_top\_voltage = false;

break

end

end

end

prev\_top\_voltage = current\_top\_voltage;

waitbar(i/length(current\_lim), waitbar\_process);

end

disp("Top Voltage value : " + string(value\_top\_voltage))

disp(" Current : " + string(current\_lim(index\_top\_voltage)))

disp("Index of Top Voltage : " + string(index\_top\_voltage))

% Compare before 2 plot and after 2 plot base on the threshold Current

[t1,y1] = ode45( @(t1,y1) HodgkinHuxley\_ODE(t1,y1,current\_lim(index\_top\_voltage-2)), x\_n, y\_0 );

[t2,y2] = ode45( @(t2,y2) HodgkinHuxley\_ODE(t2,y2,current\_lim(index\_top\_voltage-1)), x\_n, y\_0 );

[t3,y3] = ode45( @(t3,y3) HodgkinHuxley\_ODE(t3,y3,current\_lim(index\_top\_voltage )), x\_n, y\_0 );

[t4,y4] = ode45( @(t4,y4) HodgkinHuxley\_ODE(t4,y4,current\_lim(index\_top\_voltage+1)), x\_n, y\_0 );

[t5,y5] = ode45( @(t5,y5) HodgkinHuxley\_ODE(t5,y5,current\_lim(index\_top\_voltage+2)), x\_n, y\_0 );

figure(11)

subplot(1,5,1)

plot(t1, y1(:,1),'r')

title('I = ' + string(current\_lim(index\_top\_voltage-2)))

xlabel('Time (ms)')

ylabel('Membrane Potential (mV)')

axis([0 100 -100 100]);

subplot(1,5,2)

plot(t2, y2(:,1),'r')

title('I = ' + string(current\_lim(index\_top\_voltage-1)))

xlabel('Time (ms)')

ylabel('Membrane Potential (mV)')

axis([0 100 -100 100]);

subplot(1,5,3)

plot(t3, y3(:,1),'r')

title('I = ' + string(current\_lim(index\_top\_voltage)))

xlabel('Time (ms)')

ylabel('Membrane Potential (mV)')

axis([0 100 -100 100]);

subplot(1,5,4)

plot(t4, y4(:,1),'r')

title('I = ' + string(current\_lim(index\_top\_voltage+1)))

xlabel('Time (ms)')

ylabel('Membrane Potential (mV)')

axis([0 100 -100 100]);

subplot(1,5,5)

plot(t5, y5(:,1),'r')

title('I = ' + string(current\_lim(index\_top\_voltage+2)))

xlabel('Time (ms)')

ylabel('Membrane Potential (mV)')

axis([0 100 -100 100]);

close(waitbar\_process); clear waitbar\_process;

disp("Process finished")

% Problem 4-b)

clear; clc; close all;

% initial values

v\_rest = 60;

v\_0 = - v\_rest;

% when t=0, dn/dt = dm/dt = dh/dt = 0

n\_0 = alpha\_n(v\_0)/(alpha\_n(v\_0)+beta\_n(v\_0));

m\_0 = alpha\_m(v\_0)/(alpha\_m(v\_0)+beta\_m(v\_0));

h\_0 = alpha\_h(v\_0)/(alpha\_h(v\_0)+beta\_h(v\_0));

y\_0 = [v\_0; n\_0; m\_0; h\_0];

% x\_n = 0:0.001:50;

% I\_n = 0:0.001:0.4;

x\_n = 0:0.001:100;

% current\_lim = 0.0500:0.0010:0.1000;

current\_lim = 0.060:0.0001:0.065;

index\_stable\_voltage = 1;

value\_top\_voltage = 0;

flag\_find\_top\_voltage = true; % for counting the number of A.P.

flag\_voltage\_direction = true; % for counting the number of A.P.

% for monitoring the proceeding status

waitbar\_process = waitbar(0, 'Process ...');

disp("iteration " + string(length(current\_lim)))

current\_top\_voltage = 0;

for i=1:1:length(current\_lim)

% count the number of AP appear (-> and if count is more than 5, that

% can be a stable state...

number\_of\_AP = 0;

current=current\_lim(i);

% disp(current)

[t,y] = ode45( @(t,y) HodgkinHuxley\_ODE(t,y,current), x\_n, y\_0 );

max\_voltage = max(y(:,1));

current\_top\_voltage = max\_voltage;

% find the point of A.P. & count the number of A.P.

% find

current\_y = y(:,1);

for n=1:1:length(current\_y)

if current\_y(n) > -50

if flag\_voltage\_direction == true

number\_of\_AP = number\_of\_AP + 1;

flag\_voltage\_direction = false;

end

else

if flag\_voltage\_direction == false

number\_of\_AP = number\_of\_AP + 1;

flag\_voltage\_direction = true;

end

end

end

% count

if flag\_find\_top\_voltage == true

if number\_of\_AP/2 >= 5

index\_stable\_voltage = i;

disp(number\_of\_AP)

flag\_find\_top\_voltage = false;

break

end

end

waitbar(i/length(current\_lim), waitbar\_process);

end

disp("Stable Current : " + string(current\_lim(index\_stable\_voltage)) + " mV")

disp("Index of stable A.P. : " + string(index\_stable\_voltage))

% Compare before 2 plot and after 2 plot base on the threshold Current

[t1,y1] = ode45( @(t1,y1) HodgkinHuxley\_ODE(t1,y1,current\_lim(index\_stable\_voltage-2)), x\_n, y\_0 );

[t2,y2] = ode45( @(t2,y2) HodgkinHuxley\_ODE(t2,y2,current\_lim(index\_stable\_voltage-1)), x\_n, y\_0 );

[t3,y3] = ode45( @(t3,y3) HodgkinHuxley\_ODE(t3,y3,current\_lim(index\_stable\_voltage )), x\_n, y\_0 );

[t4,y4] = ode45( @(t4,y4) HodgkinHuxley\_ODE(t4,y4,current\_lim(index\_stable\_voltage+1)), x\_n, y\_0 );

[t5,y5] = ode45( @(t5,y5) HodgkinHuxley\_ODE(t5,y5,current\_lim(index\_stable\_voltage+2)), x\_n, y\_0 );

figure(12)

subplot(1,5,1)

plot(t1, y1(:,1),'r')

title('I = ' + string(current\_lim(index\_stable\_voltage-2)))

xlabel('Time (ms)')

ylabel('Membrane Potential (mV)')

axis([0 100 -100 100]);

subplot(1,5,2)

plot(t2, y2(:,1),'r')

title('I = ' + string(current\_lim(index\_stable\_voltage-1)))

xlabel('Time (ms)')

% ylabel('Membrane Potential (mV)')

axis([0 100 -100 100]);

subplot(1,5,3)

plot(t3, y3(:,1),'r')

title('I = ' + string(current\_lim(index\_stable\_voltage)))

xlabel('Time (ms)')

% ylabel('Membrane Potential (mV)')

axis([0 100 -100 100]);

subplot(1,5,4)

plot(t4, y4(:,1),'r')

title('I = ' + string(current\_lim(index\_stable\_voltage+1)))

xlabel('Time (ms)')

% ylabel('Membrane Potential (mV)')

axis([0 100 -100 100]);

subplot(1,5,5)

plot(t5, y5(:,1),'r')

title('I = ' + string(current\_lim(index\_stable\_voltage+2)))

xlabel('Time (ms)')

% ylabel('Membrane Potential (mV)')

axis([0 100 -100 100]);

close(waitbar\_process); clear waitbar\_process;

disp("Process finished")

function dydt = HodgkinHuxley\_ODE(t,y,current)

% constant

C\_m=0.01;

E\_na=55.17;

E\_k=-72.14;

E\_l=-49.42;

g\_na=1.2;

g\_k=0.36;

g\_l=0.003;

% variables

v = y(1);

n = y(2);

m = y(3);

h = y(4);

dvdt = (1/C\_m) \* ( current - g\_na\*m^3\*h\*(v-E\_na) - g\_k\*n^4\*(v-E\_k) - g\_l\*(v-E\_l) );

dndt = alpha\_n(v)\*(1-n) - beta\_n(v)\*n;

dmdt = alpha\_m(v)\*(1-m) - beta\_m(v)\*m;

dhdt = alpha\_h(v)\*(1-h) - beta\_h(v)\*h;

dydt = [dvdt; dndt; dmdt; dhdt];

% dydt = dvdt;

end

function output = alpha\_n(v)

output = 0.01\*(v+50) / (1 - exp(-(v+50)/10));

end

function output = alpha\_m(v)

output = 0.1\*(v+35) / (1 - exp(-(v+35)/10));

end

function output = alpha\_h(v)

output = 0.07 \* exp(-0.05 \* (v+60));

end

function output = beta\_n(v)

output = 0.125 \* exp(-(v+60)/80);

end

function output = beta\_m(v)

output = 4.0 \* exp(-0.0556 \* (v+60));

end

function output = beta\_h(v)

output = 1 / (1 + exp(-0.1 \* (v+30)));

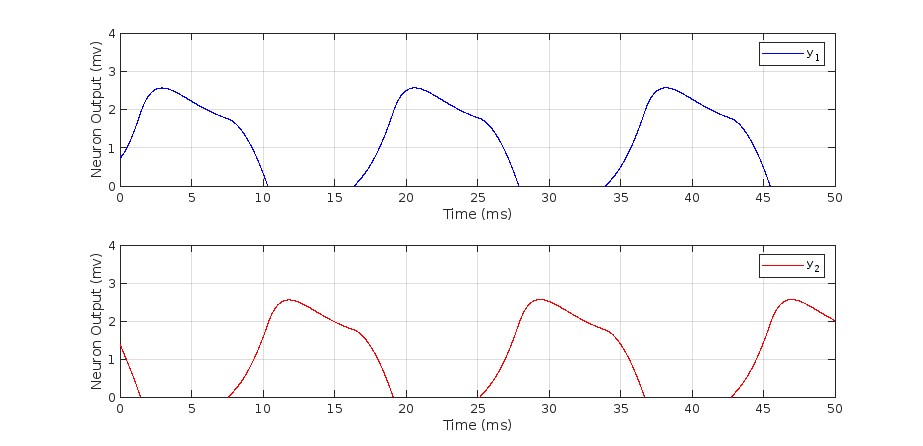
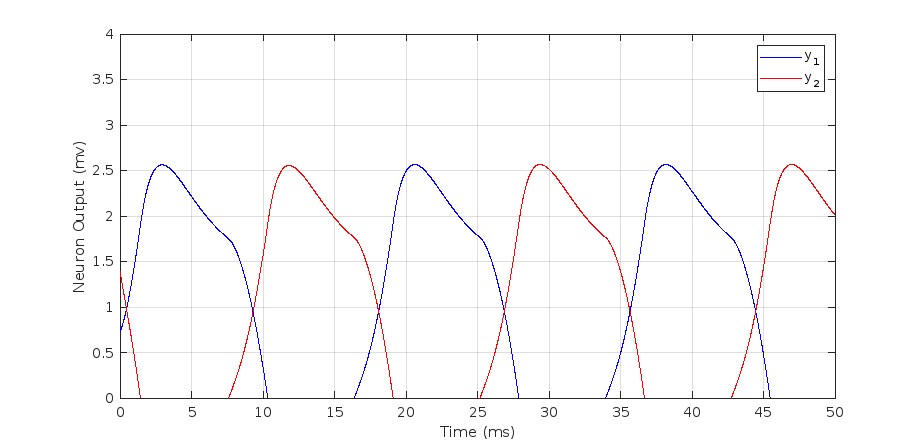
end

**5. neural oscillator with reciprocal inhibition**

**a) Find the initial conditions by trial-and-error**

과 의 값은 서로 다르며, 이 때 0이 아닌 실수를 부여할 경우 일정 위치부터 rhythmical 한 양상이 생기는 것으로 보인다. 특히 1 이상의 큰 정수로 넘어갈 경우 초반에 안정되지 않은 모습일 보였으며 peak point 가 안정적인 부분에 비해 높거나 낮았다. 따라서 여러 번 시뮬레이션을 진행해 가장 적합한 초기값을 부여했다.

b) plot y1 and y2 versus time



위의 그래프처럼 y1(파란색) y2(붉은색) 그래프가 서로 번갈아가면서 발생하는 현상을 확인할 수 있었다. 특히 서로 증감하는 양 끝 구간에서 일부 겹쳐져 보이는 양상 또한 보인다.

> 소스코드

% Problem 5-a) and b)

clear all; clc; close all;

% initial condition

x1\_0 = 0.73;

x2\_0 = 1.39;

z1\_0 = max(0,x1\_0);

z2\_0 = max(0,x2\_0);

y\_0 = [x1\_0 x2\_0 z1\_0 z2\_0];

x\_n = 0:0.01:50;

lim = 0.00:0.01:1.00;

current\_lim = 0:0.00001:0.0001;

[t,y] = ode45( @(t,y) NeuralOscillator\_ODE(t,y,current\_lim(1)), x\_n, y\_0);

% end

% plot

figure(51)

plot(t,y(:,1),'b', t,y(:,2), 'r')

xlabel('Time (ms)')

ylabel('Neuron Output (mv)')

ylim([0 4])

legend('y\_1','y\_2')

grid on

figure(52)

% hold on

subplot(2,1,1)

plot(t,y(:,1), 'b')

ylim([0 4])

xlabel('Time (ms)')

ylabel('Neuron Output (mv)')

legend('y\_1')

grid on

subplot(2,1,2)

plot(t,y(:,2), 'r')

ylim([0 4])

xlabel('Time (ms)')

ylabel('Neuron Output (mv)')

legend('y\_2')

grid on

% hold off

disp("Process finished")

function dydt = NeuralOscillator\_ODE(t,y,z)

T\_r=1;

T\_a=12;

b=2.5;

a12=1.5;

a21=1.5;

u1=5;

u2=5;

s1=0;

s2=0;

x1 = y(1);

x2 = y(2);

z1 = y(3);

z2 = y(4);

dx1dt = (1/T\_r) \* (-x1 - a12 \* (max(0,x2)) - b\*z1 + u1 + s1 ) ;

dx2dt = (1/T\_r) \* (-x2 - a21 \* (max(0,x1)) - b\*z2 + u2 + s2 ) ;

dz1dt = (1/T\_a) \* (-z1 + max(0,x1));

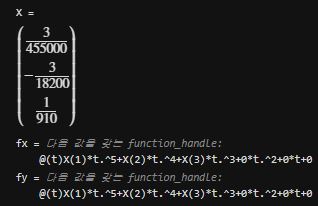
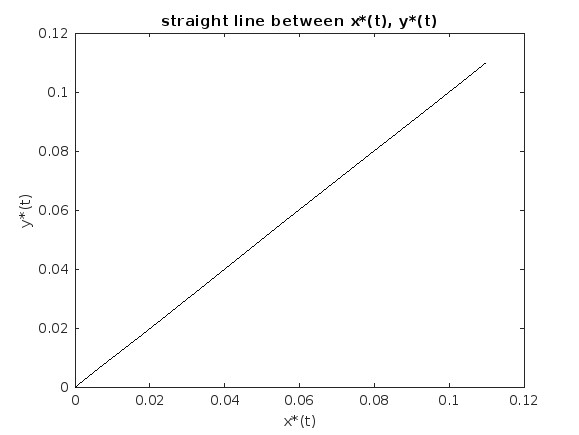
dz2dt = (1/T\_a) \* (-z2 + max(0,x2));

dydt = [dx1dt; dx2dt; dz1dt; dz2dt];

end

**6. minimum-jerk model with Euler-Lagrangian Equation**

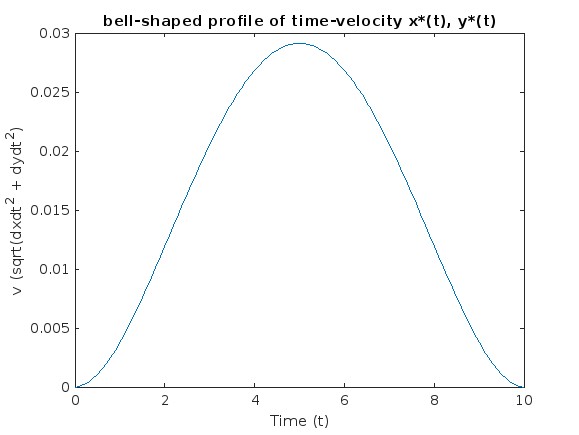
**b)**

****

5차 다항식으로 이루어진 x(t)와 y(t)의 5차 4차 3차항의 계수값을 연립방정식을 통해 구해낼 수 있었다.

또한 t=[0, 10] 구간에서의 x(t)-y(t) 그래프는 직선으로 나타났다. 당연히 x(t)와 y(t)는 동일한 계수를 가진 5차다항식이기 때문에 보이는 현상이다.

c) plot velocity of x(t), y(t)



속도는 주어진 정의대로 x(t)와 y(t)를 각각 제곱한 것을 더하고 그 값에 제곱근을 취해주는 것이다.

그에 따라 plot으로 표현했고, bell-shaped profile을 구해볼 수 있었다.

> 소스코드

% 6 - b)

clear, clc, close all;

syms a b c

eq1 = 1e4\*a + 1e3\*b + 10e2\*c == 1

eq2 = 5\*1e2\*a + 4\*10\*b + 3\*c ==0

eq3 = 20\*1e2\*a + 12\*10\*b + 6\*c == 0

[A,B] = equationsToMatrix([eq1, eq2, eq3], [a,b,c])

X = linsolve(A,B)

syms x y

fx = @(t) X(1)\*t.^5 + X(2)\*t.^4 + X(3)\*t.^3 + 0\*t.^2 + 0\*t + 0

fy = @(t) X(1)\*t.^5 + X(2)\*t.^4 + X(3)\*t.^3 + 0\*t.^2 + 0\*t + 0

lim = [0:1:10];

plot(fx(lim), fy(lim), 'k')

title('straight line between x\*(t), y\*(t)')

xlabel('x\*(t)')

ylabel('y\*(t)')

% 6 - c)

clear, clc, close all;

syms a b c

eq1 = 1e4\*a + 1e3\*b + 10e2\*c == 1

eq2 = 5\*1e2\*a + 4\*10\*b + 3\*c ==0

eq3 = 20\*1e2\*a + 12\*10\*b + 6\*c == 0

[A,B] = equationsToMatrix([eq1, eq2, eq3], [a,b,c])

X = linsolve(A,B)

syms x y t

fx = @(t) X(1)\*t.^5 + X(2)\*t.^4 + X(3)\*t.^3 + 0\*t.^2 + 0\*t + 0

fy = @(t) X(1)\*t.^5 + X(2)\*t.^4 + X(3)\*t.^3 + 0\*t.^2 + 0\*t + 0

fxdt = @(t) 5\*X(1)\*t.^4 + 4\*X(2)\*t.^3 + 3\*X(3)\*t.^2 + 2\*0\*t.^1 + 0

fydt = @(t) 5\*X(1)\*t.^4 + 4\*X(2)\*t.^3 + 3\*X(3)\*t.^2 + 2\*0\*t.^1 + 0

lim = [0:0.1:10];

xdot2 = fxdt(lim).^2;

ydot2 = fydt(lim).^2;

v = sqrt(xdot2+ydot2)

plot(lim, v)

title('bell-shaped profile of time-velocity x\*(t), y\*(t)')

xlabel('Time (t)')

ylabel('v (sqrt(dxdt^2 + dydt^2)')