Design & Optimization of Animal's Locomotion Mechanism

"Grasshopper"

By

Nattavee Thiengkaew ID: 5930192021

Nuthasith Gerdpratoom ID: 5930194221

Ditsakorn Wanichratanagul ID: 5930201021

2103322 Mechanics of Machinery

Department of Mechanical Engineering

Faculty of Engineering

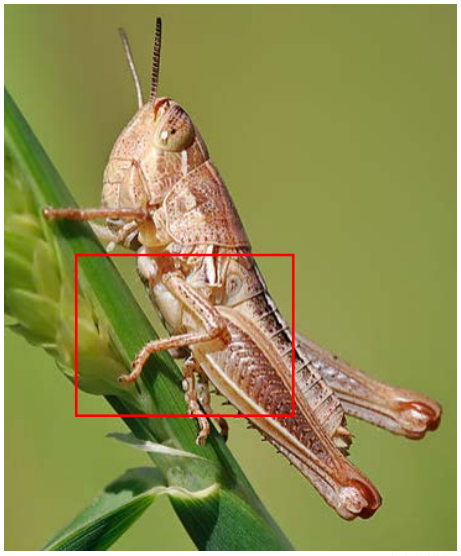
Chulalongkorn University

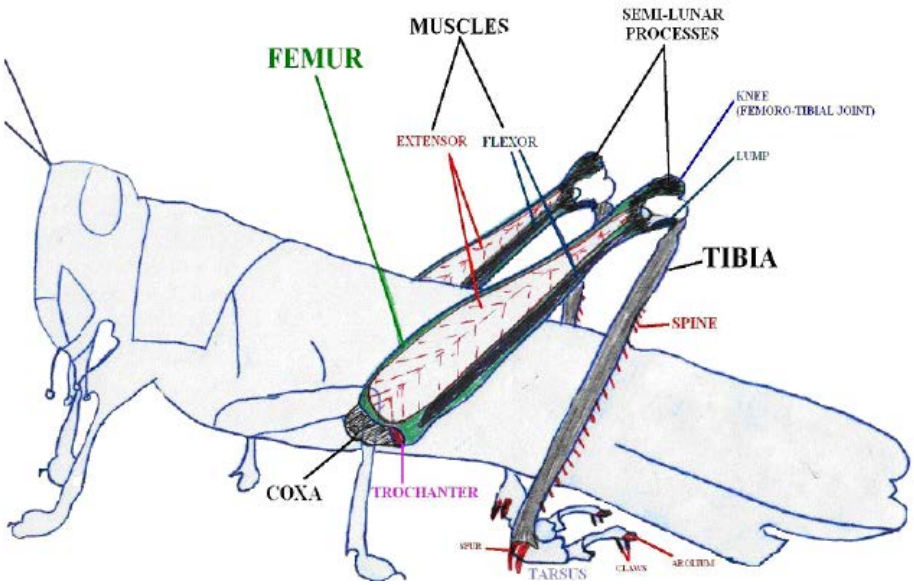
Contents

Part I

Grasshopper's Leg Mechanism

กลไกที่เลือกคือขาตั๊กแตน ตั๊กแตนเป็นแมลงชนิดหนึ่ง ซึ่งจัดเป็นสัตว์ไม่มีกระดูกสันหลังในไฟลัมอาร์โทรโพรดา (Arthropoda) ขาของตั๊กแตนประกอบด้วยกระดูก และข้อต่อต่าง ๆ เพื่อใช้ในการเคลื่อนที่ของมัน เช่น การกระโดด โดยมีกล้ามเนื้อ (Muscle) ที่อยู่ตามขาเพื่อใช้เป็นตัวกระตุ้น (Actuator) ให้เกิดการเคลื่อนที่ และระบบประสาทเป็นตัวควบคุม

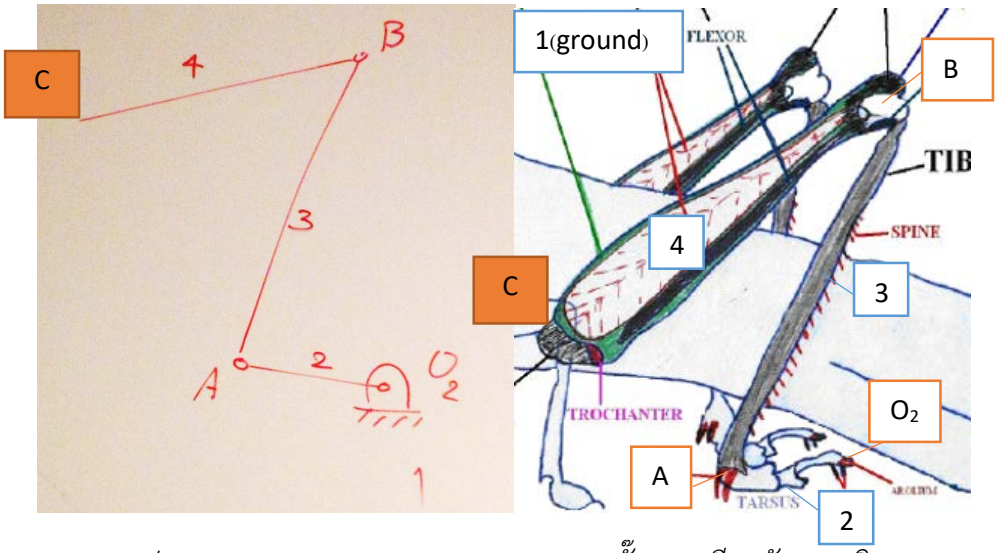




รูป 1.1 ตั๊กแตน และองค์ประกอบของขามัน (ที่มา: Wikipedia)

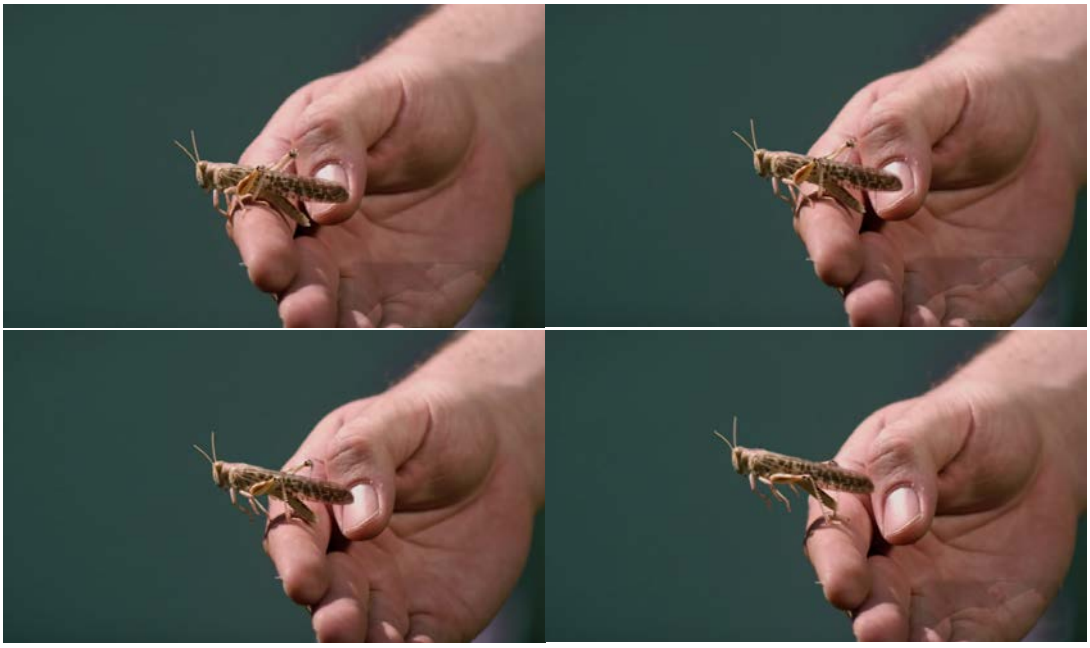
Summary of the Dataset

ขาตั๊กแตนเป็นกลไกชนิดหนึ่ง ประกอบด้วยข้อต่อ (Link) ซึ่งก็คือกระดูกขาส่วนต่าง ๆ โดยจะสมมติว่าเป็นวัตถุแข็งเกร็ง (Rigid body) และจุดต่อ (Joint) มารวมกัน ในที่นี้จะพิจารณาช่วงที่ตั๊กแตนกำลังจะกระโดด ก็คือช่วงเวลาที่ปลายเท้าของตั๊กแตนติดอยู่กับพื้นตลอด ทำให้สามารถมองว่าปลายเท้าของตั๊กแตนเป็น fixed pivot ติดกับ ground ได้ และเป็นการเคลื่อนที่ในระนาบ (2-D) จะได้เป็น open kinematic chain ดังแสดงใน Kinematic Diagram (รูป 1.2) โดยทุก ๆ joint จะเป็น Revolute Joint

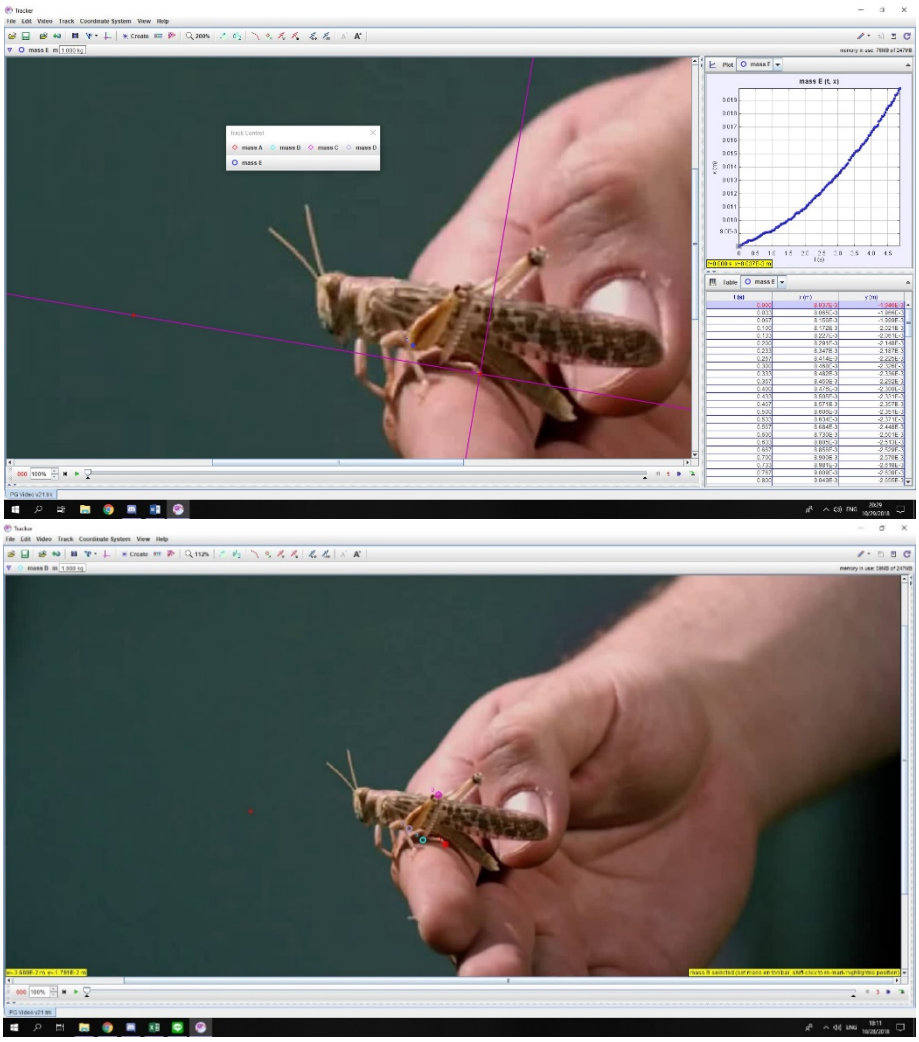


รูป 1.2 แสดง Kinematic Diagram ของขาตั๊กแตนเทียบกับของจริง

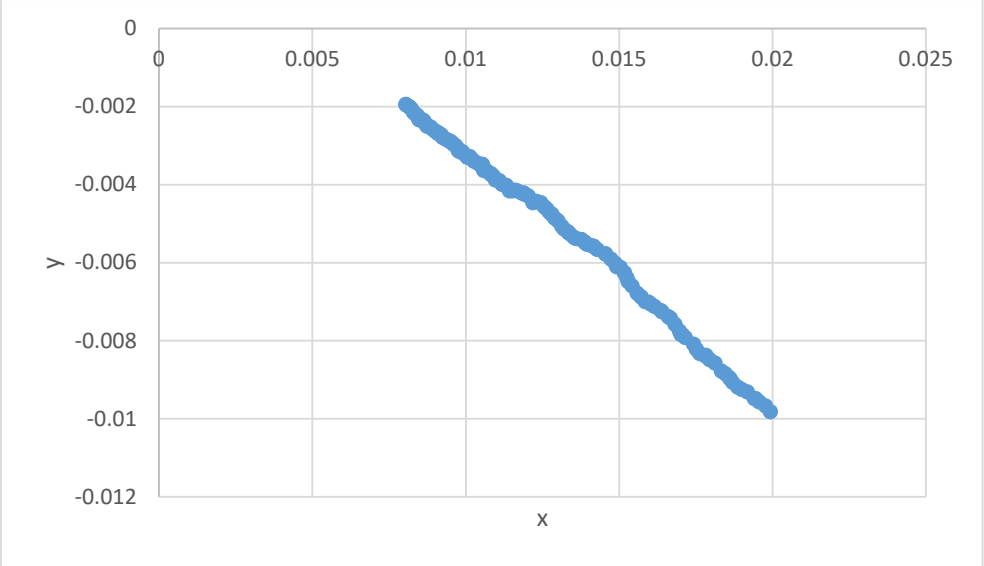
โดย Trajectory ของการเคลื่อนที่ของขาตั๊กแตนหาได้จากนำ Video การเคลื่อนที่ในระนาบของตั๊กแตนจาก YouTube และหาตำแหน่งของจุดที่สนใจที่เวลาต่าง ๆ ด้วยโปรแกรม Tracker โดยตัวโปรแกรมจะ set up frame และถ่ายภาพ Snapshot ที่เวลาต่าง ๆ ดังรูป 1.3 จากนั้นให้เรากำหนดจุดที่สนใจบน Snapshot ของเวลานั้น ๆ โปรแกรมจะให้ค่าตำแหน่งของจุดที่เรากำหนดเป็น coordinate x,y ดังรูป 1.4 จากนั้นนำข้อมูลตำแหน่งที่ได้มาบันทึกค่าลงในโปรแกรม Excel แล้ว plot x,y ออกมาเป็น Trajectory ของการเคลื่อนที่ของขาตั๊กแตน โดยจุดที่สนใจในการเคลื่อนที่คือจุด C บน Link 4 (อยู่ตรงข้ามกับ Joint B) เทียบกับ Joint O_2 (ปลายเท้า) โดยพิจารณาช่วงที่ปลายเท้ายังไม่หลุดจากพื้น ดังนั้น Trajectory ของจุด C จึงเทียบกับโลกดังรูป 1.5



รูป 1.3 แสดง snapshot การกระโดดของตั๊กแตน



รูป 1.4 แสดงการใช้โปรแกรม Tracker



รูป 1.5 แสดง Trajectory ของจุด C บน Link 4 ของขาตั๊กแตน เทียบกับโลก

ค่า parameter ของกลไกนี้คือความยาว และมวลของขาตั๊กแตน แต่ในที่นี้จะไม่สนใจมวลของมัน โดยการวัดความยาวของขาจะวัดด้วยการใช้โปรแกรม Tracker เช่นเดียวกับตอนหา Trajectory แล้วหาความยาวจุดต่อจุดของขามัน จะได้ความยาว Link 2 (O_2A) คือ 4.592 mm, Link 3 (AB) คือ 9.694 mm, Link 4 (BC) คือ 9.242 mm

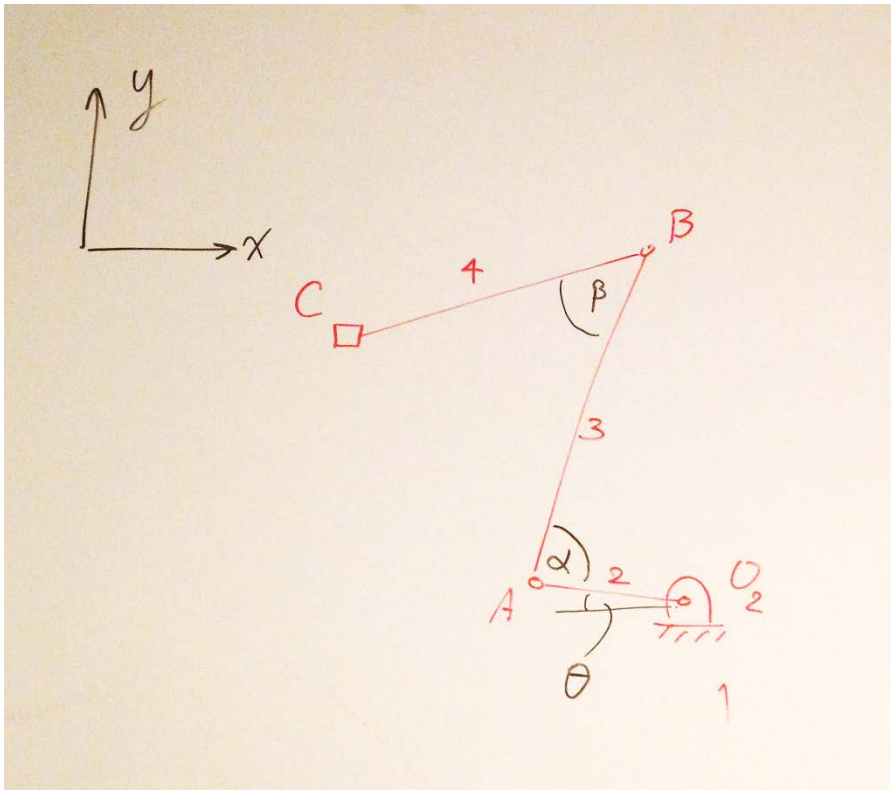
Design parameters for optimization

ค่า parameter ที่จะทำการ optimization คือความยาวของข้อต่อ (Link) ของกลไก หรือความยาวของท่อนขาของตั๊กแตน เพื่อให้ได้การเคลื่อนที่ตาม Trajectory ที่ต้องการ โดยให้ค่า Angular Velocity ของแต่ละ joint เป็นฟังก์ชัน ramp ที่มีความชันคงที่ค่าหนึ่ง เนื่องจากการกระโดดของตั๊กแตนจะมี Torque เกิดขึ้น (สมมติว่า Torque มีค่าคงที่ค่าหนึ่ง) จากกฎข้อที่สองของนิวตัน ทำให้ได้ว่า Angular Acceleration มีค่าคงที่ และ Angular Velocity เป็น ramp

Reference on Dataset

- Mesmerizing Video: Flying Insects in Super Slow Motion
  YouTube: Slow Motion Junkies
  Link: https://youtu.be/ED9jwEkuDtc

Hand calculation



จากความสัมพันธ์ $\vec{V}_C = \vec{V}_B + \vec{V}_{C/B}$

$$\vec{V}_C = \vec{V}_A + \vec{V}_{B/A} + \vec{V}_{C/B}$$

$$\vec{V}_C = \vec{V}_{A/O_2} + \vec{V}_{B/A} + \vec{V}_{C/B}$$

$$\vec{V}_C = \vec{\omega}_2 \times \vec{r}_{O_2 \to A} + \vec{\omega}_3 \times \vec{r}_{A \to B} + \vec{\omega}_4 \times \vec{r}_{B \to C} \quad (1)$$

เมื่อ $\omega_2 = \dot{\theta}$ และมีทิศตามเข็มนาฬิกา (-z)

$\omega_3 = \dot{\alpha}$ และมีทิศทวนเข็มนาฬิกา (+z)

$\omega_4 = \dot{\beta}$ และมีทิศตามเข็มนาฬิกา (-z)

Newton's 2nd Law; $T_i = I_i \dot{\omega}_i$ (2)

เมื่อ T_i คือ Torque ที่ joint O_2, A, B

I_i คือ Moment of inertia of link 2, 3, 4 about joint O_2, A, B

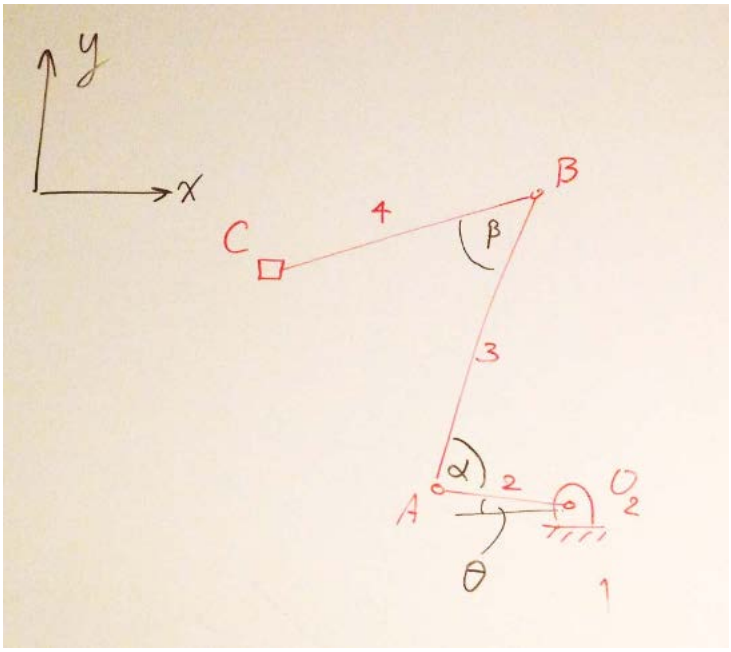
$\dot{\omega}_i$ คือ Angular velocity of link 2, 3, 4

จาก (2) เมื่อให้ T_i เป็นค่าคงที่ค่าหนึ่ง และ I_i คงที่เสมอ จะพบว่า $\dot{\omega}_i$ เป็นค่าคงที่ด้วย ดังนั้น ω_i จึงเป็นฟังก์ชัน ramp ซึ่งมีความชัน (slope) คือ $\frac{T_i}{I_i}$

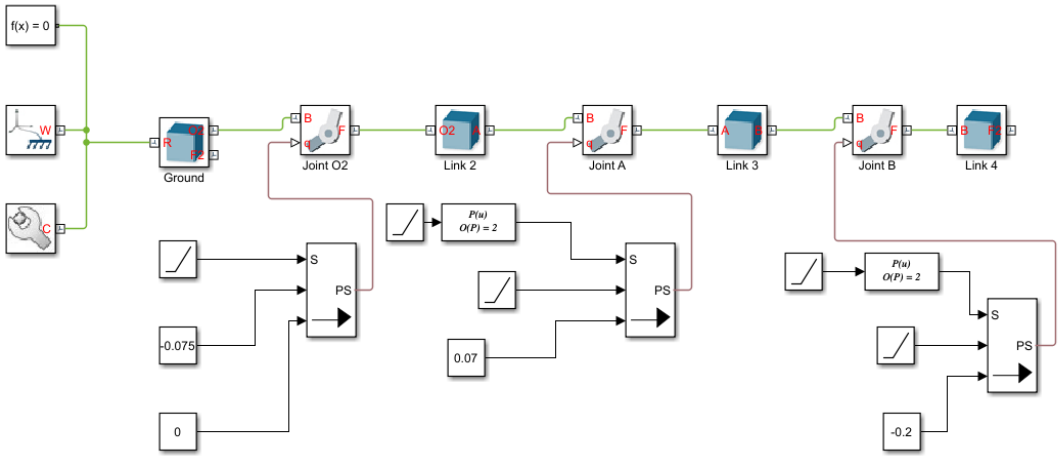
Part II

Matlab Script and Simscape Multibody Model: First run, given all parameters, and then simulate the motion trajectory.

สร้างข้อต่อของขาตั๊กแตนด้วยโปรแกรม Autodesk Fusion 360 จากนั้นเปิด Simscape Multibody สร้าง Revolute Joint และข้อต่อจากโปรแกรม Fusion นำมาต่อกันเป็น open chain kinematic ดังรูป 2.2 ตั้งค่า Actuation Torque ของแต่ละ joint เป็น automatically compute และ Actuation Motion เป็น provided by input

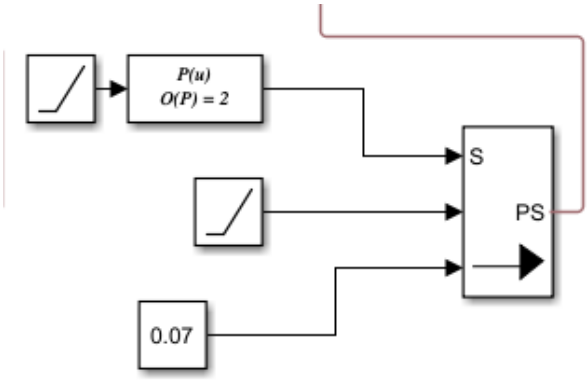


รูป 2.1 Kinematic Diagram แสดง Link และ Joint ต่าง ๆ



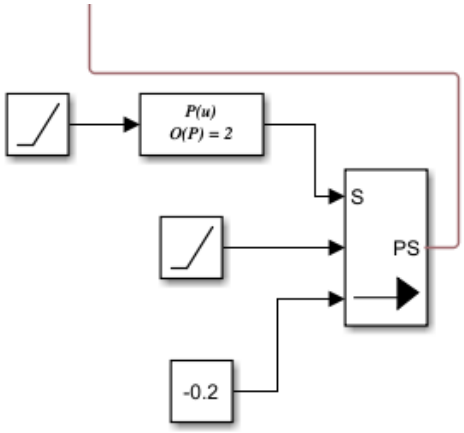
รูป 2.2 แสดงการต่อ kinematic chain ของขาตั๊กแตน

จาก part I ได้กำหนดไว้ว่า Torque ที่กระทำกับข้อต่อที่ joint A และ B มีค่าคงที่ค่าหนึ่ง เมื่อเราไม่คิดผลของแรงเฉื่อยที่กระทำระหว่างข้อต่อ จากกฎอนุรักษ์โมเมนตัมจะได้ว่าความเร่งเชิงมุม ($\dot{\omega}$) มีค่าคงที่ ค่าที่จะใส่ให้ joint A และ B คือค่ามุมสัมพัทธ์ระหว่างข้อต่อกับอนุพันธ์ของมันอีก 2 ค่า คือความเร็วเชิงมุม ความเร่งเชิงมุม ในกรณีนี้กำหนดให้ความเร่งเชิงมุมมีค่าคงที่ค่าหนึ่ง ดังนั้นเมื่อทำการหาปริพันธ์ของความเร่งเชิงมุมและให้เงื่อนไขเริ่มต้นเป็นศูนย์ จะได้ความเร็วเชิงมุมจะเป็นฟังก์ชัน ramp ที่มี slope เท่ากับความเร่ง และตำแหน่งเชิงมุมจะเป็นฟังก์ชัน polynomial ที่มีสัมประสิทธิ์หน้า t^2 เป็นครึ่งหนึ่งของความเร่งบวกกับมุมเริ่มต้น ตามรูป 2.3 – 2.4 และให้ความเร็วเชิงมุมของ joint O_2 คงที่ ซึ่งก็คือไม่มี input torque ที่ joint O_2 ตามรูป 2.5



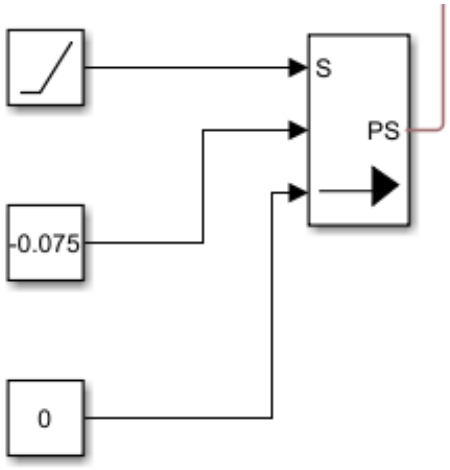
รูป 2.3 แสดง input motion ของ joint A

จากรูป 2.3 กำหนดให้ความเร่งเชิงมุมคงที่เท่ากับ 0.07 rad/s^2, ความเร็วเชิงมุมเท่ากับ 0.07t rad/s และตำแหน่งเชิงมุมเท่ากับ $0.035t^2$+4.4146 rad



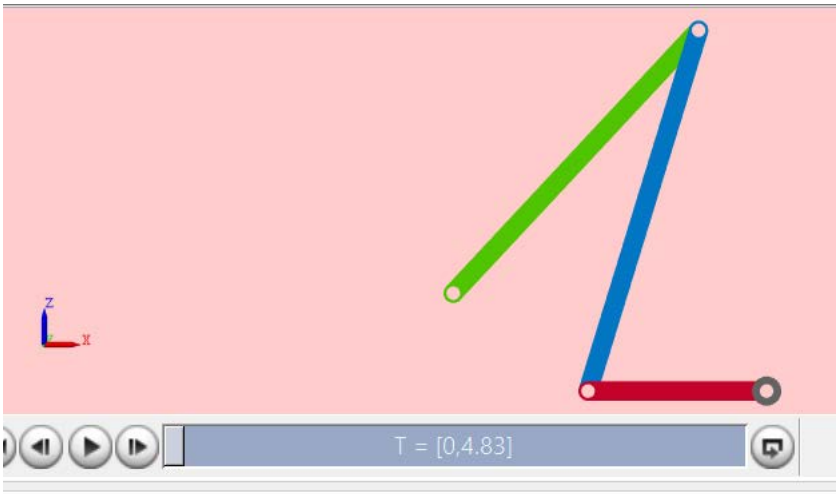
รูปที่ 2.4 แสดง input motion ของ joint B

จากรูป 2.4 กำหนดให้ความเร่งเชิงมุมคงที่เท่ากับ -0.2 rad/s^2, ความเร็วเชิงมุมเท่ากับ -0.2t rad/s และตำแหน่งเชิงมุมเท่ากับ $-0.1t^2$+2.6881 rad



รูป 2.5 แสดง input motion ของ joint O_2

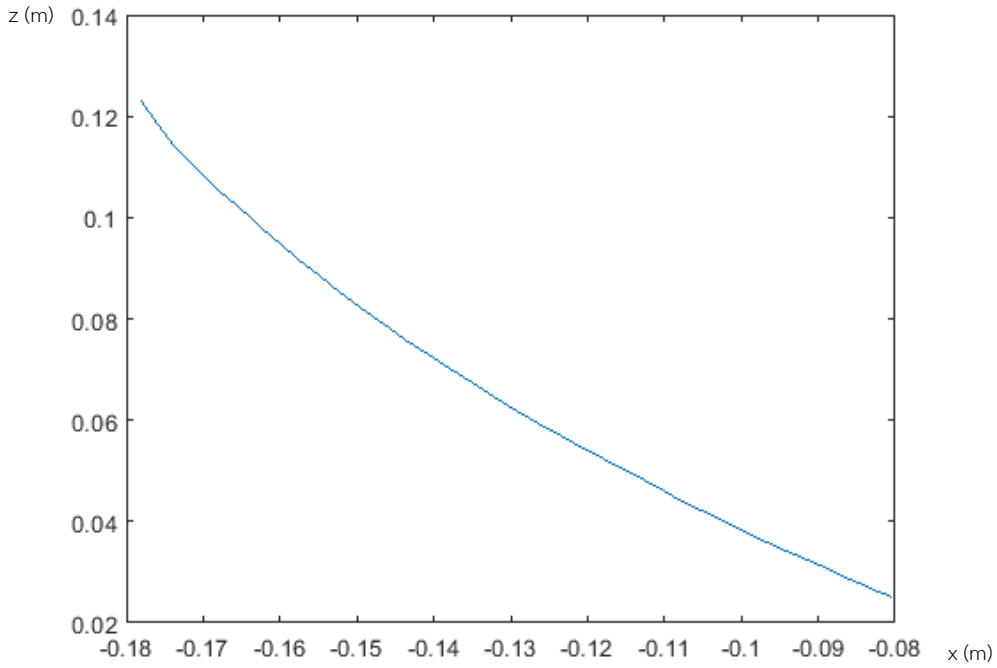
จากนั้นทำการ simulate โดยใช้ time interval 0-4.83 sec คือช่วงเวลาที่ตั๊กแตนเริ่มจะกระโดดจนถึงเวลาที่ขาไม่แตะพื้น จะได้ kinematic chain simulation ดังรูป 2.6



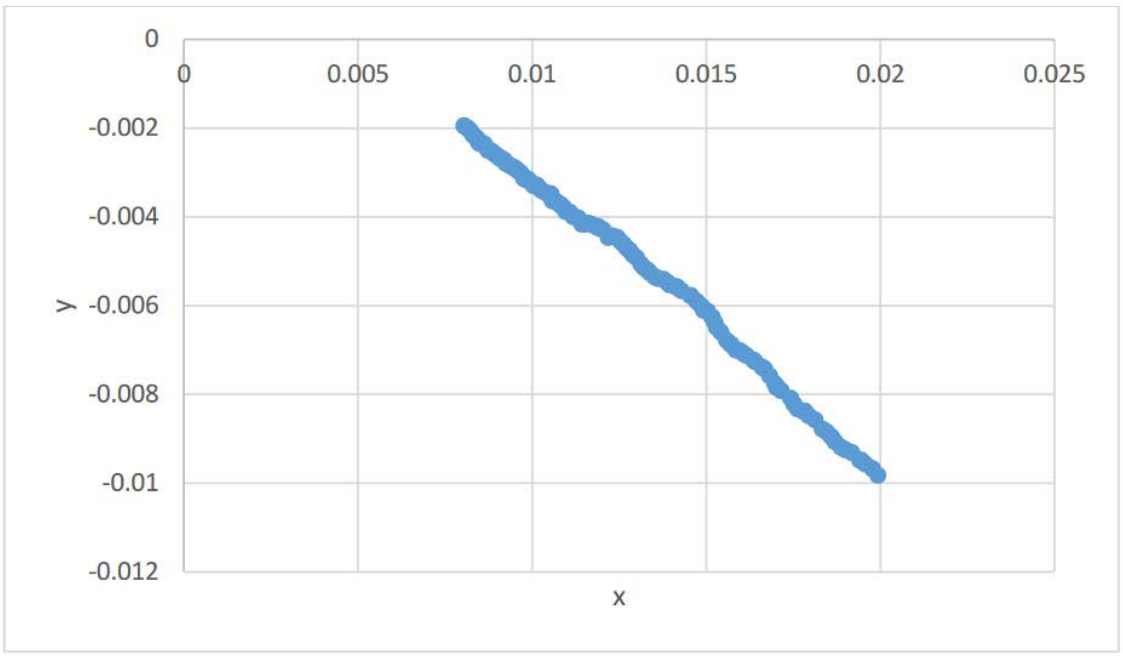
รูป 2.6 kinematic chain simulation

Plot of the important state variables, and trajectory profile plot.

x-z plot

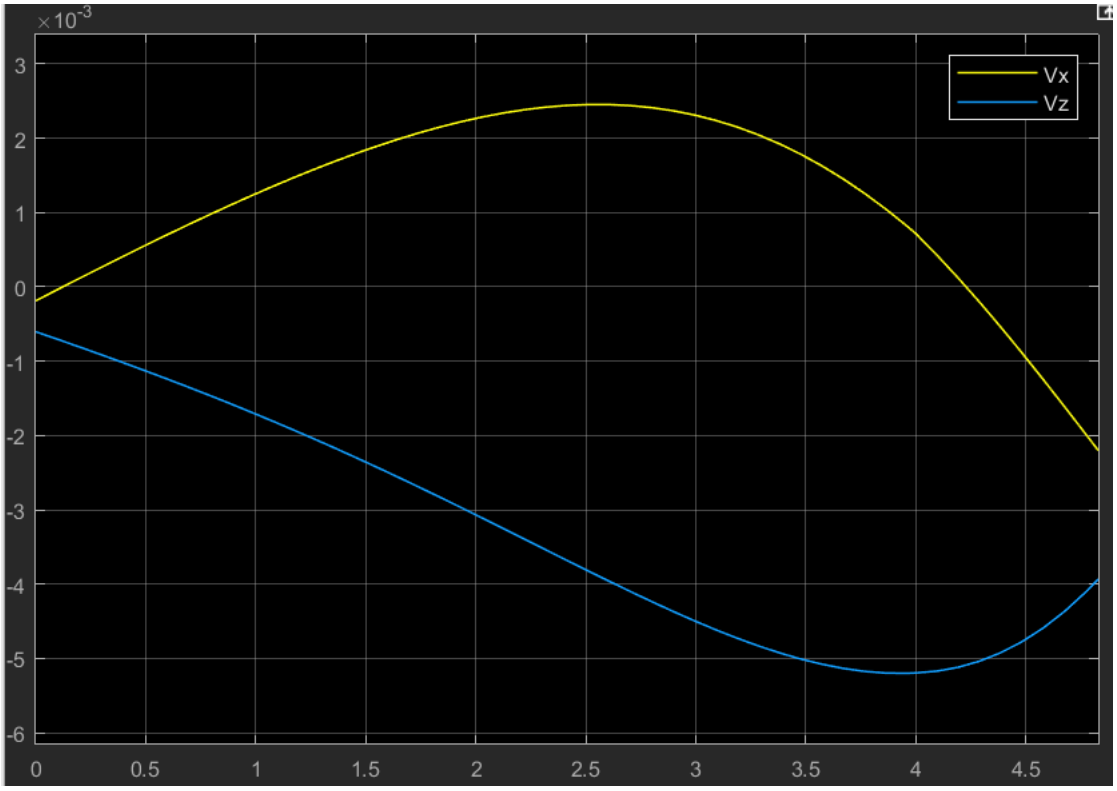


x-y plot จาก part I



v_x and v_z plot

v (m/s)



หมายเหตุ Video ของ simulation ขาตั๊กแตนได้แนบไปใน Assignment Part II ของ mycourseville

Part III

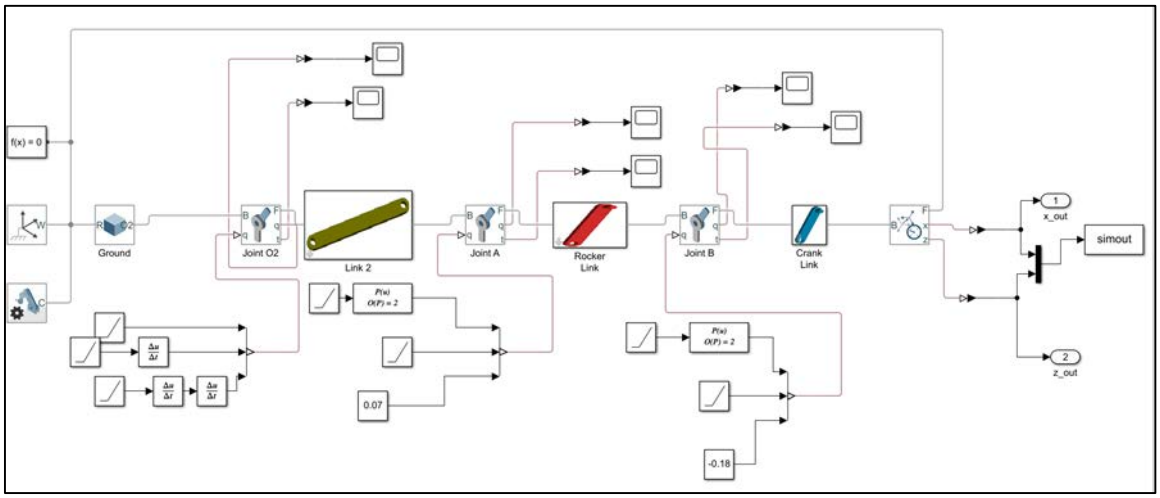
Optimization Script: Define cost function, optimization setting, and the values of optimal design parameters.

หลังจากสร้าง Simulink model จาก Part II ได้นำค่าตำแหน่งต่าง ๆ ที่ได้จาก Part I มาทำ regression ด้วยฟังก์ชัน fit แล้วเตรียมเป็น Initial Parameter ในชื่อไฟล์ workspace.mat พร้อมทั้งลดขนาด Path trajectory โดยการคูณ 0.8 เพื่อใช้เป็น Path สำหรับ Optimization ต่อไป

Workspace.mat

```
close all;
clear all;
clc;

L1 = 4.592*1;  % [cm]
L2 = 9.694*1;
L3 = 9.242*1;
x1=[0.0805602592779716;0.0805602592779716;0.0806251857217589;0.0808198834352290;0.
y1=[-0.0251279469440408;-0.0251279469440408;-0.0251696166190371;-0.025294672985944
for i=1:55;
    y(i)=-0.9439*x1(i)+0.05481;
    x(i)=x1(i);
end
x=-0.8*x+0.03;
y=-0.8*y+0.02;
```

MM.slx



ในส่วนของ Simulink ให้ Input เป็นความเร็วเชิงมุมของแต่ละ Link แล้วนำไป Optimize หาระยะ Link ของ Path trajectory ที่ต้องการโดยใช้ Cost function ดังนี้

Cost.m

```
function J = cost(X, XZ_ref)

    % Given L123;
    load('workspace.mat')
    Ts= 4.86666666666666/52;
    L1 = 1*X(1);
    L2 = 1*X(2);
    L3 = 1*X(3);


    simopt = simset('SrcWorkspace','Current');
    set_param(gcs,'SimulationCommand','Update'); % Update Model

    [~, ~, Yout]= sim('MM', [0:1:52]'*Ts, simopt);

    x = -1*Yout(:,1);
    z = -1*Yout(:,2);

    x_ref = -1*XZ_ref(:,1);
    z_ref = -1*XZ_ref(:,2);

    J = sum( (x - x_ref).^2 + (z - z_ref).^2 );

    % Plot
    figure(101);

    plot(x, z, 'LineWidth', 4, 'LineStyle', '--');


end
```

และในการ Optimization ได้ทำการหาค่า Initial Length ที่ทำให้ค่า J (หรือ F-function เมื่อ run fmincon) มีค่าน้อยที่สุด โดยพบว่าเมื่อ Lower bound คือ [1 1 1] ทุก Initial Length วิ่งเข้าสู่ Local minima ทั้งหมดจึงใช้ Initial Length เท่ากับ [4 9 9] โดย Optimization มากระทำดังนี้

Project_Machine1.m

Table of Contents


```
close all;
clear all;
clc;
```

Open Simulink Model

```
load('workspace.mat');
open('MM.slx');
```

Update Model

```
set_param(gcs,'SimulationCommand','Update'); % Update Model
```

Simulate the Simulink Model

```
sim('MM.slx');
```

Optimization

```
x_ref = x;
z_ref = y;
```

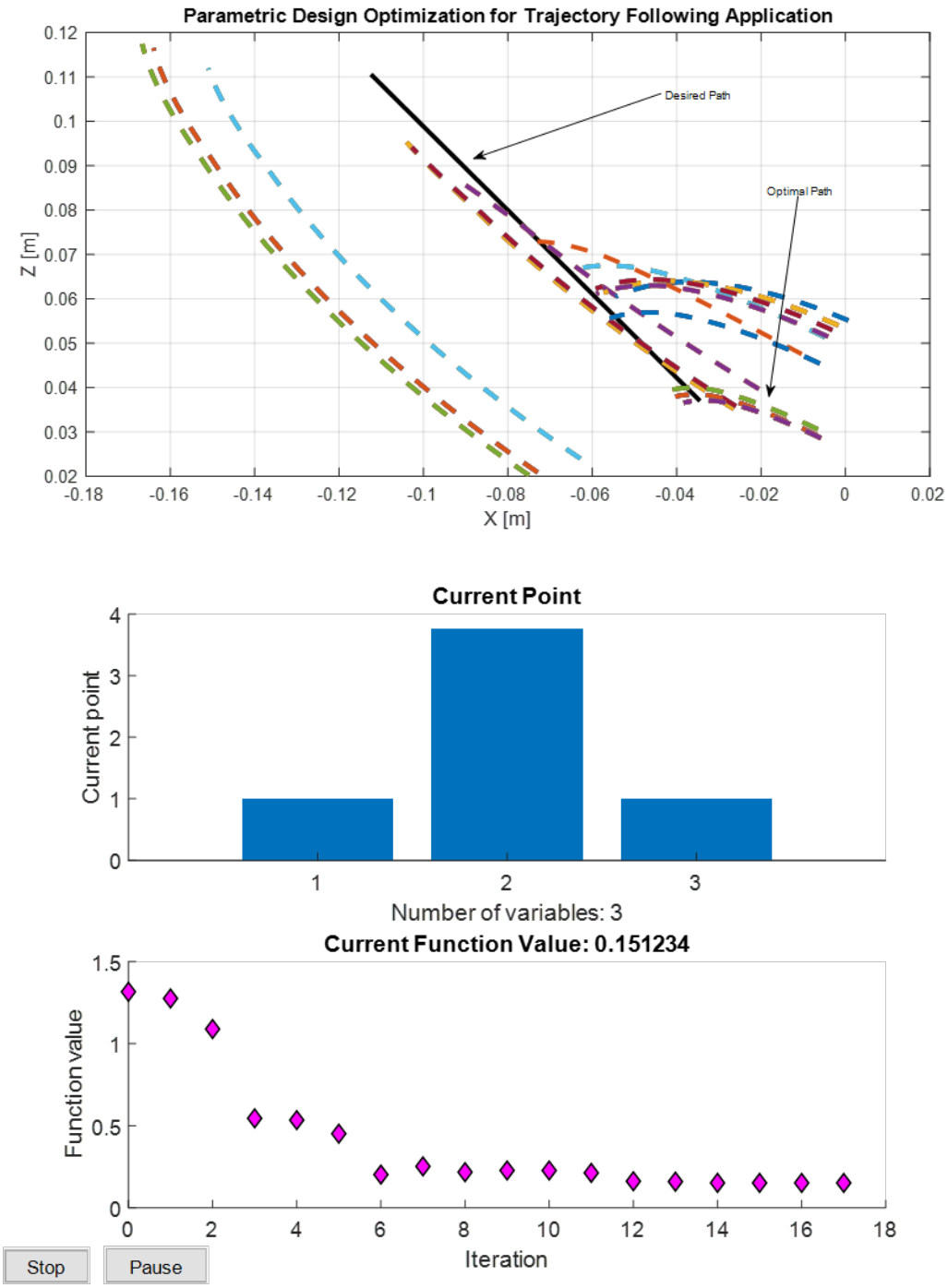
Optimization Part

```
clc;
close all;

X0 = [4 9 9];

lower_bound = [1,1,1];
upper_bound = [19, 13 ,20];


objective = @(X)cost(X, [x_ref z_ref]);
options = optimset('Display','iter','TolX',1e-8, 'MaxIter',
 50, 'PlotFcns', {'optimplotx', 'optimplotfval' }); % optimplotfval
```

```
figure(101);
set(gcf, 'Position', [1000 200 2560 1280]/2);
plot(x_ref, z_ref, 'LineWidth', 4, 'Color', 'k');
hold on;
xlabel('X [m]');
ylabel('Z [m]');
title('Parametric Design Optimization for Trajectory Following
 Application');
%legend('Reference Trajectory');
grid on;
set(gca, 'FontSize', 16);

% Solve Optimization Problem
[X_opt, fval] = fmincon(objective, X0, [], [], [], [], lower_bound,
 upper_bound,[], options);
L1_opt = X_opt(1);
L2_opt = X_opt(2);
L3_opt = X_opt(3);

% Display Result
disp(sprintf('Optimal Design Parameters\nL1 = %.4f [cm]\nL2 = %.4f
 [cm]\nL3 = %.4f [cm]', L1_opt, L2_opt,L3_opt));
```

```
 Iter F-count            f(x)  Feasibility   optimality         step
    0       4    1.315858e+00    0.000e+00    1.428e-01
    1       8    1.275420e+00    0.000e+00    1.346e-01    1.905e-01
    2      12    1.089025e+00    0.000e+00    1.227e-01    9.228e-01
    3      16    5.446923e-01    0.000e+00    7.410e-02    3.336e+00
    4      20    5.339678e-01    0.000e+00    7.301e-02    1.643e-01
    5      24    4.514506e-01    0.000e+00    6.434e-02    9.538e-01
    6      28    2.020327e-01    0.000e+00    3.529e-02    3.877e+00
    7      33    2.519940e-01    0.000e+00    3.550e-02    9.044e-01
    8      37    2.168333e-01    0.000e+00    2.847e-02    7.456e-01
    9      41    2.271050e-01    0.000e+00    2.806e-02    1.730e-01
   10      45    2.268489e-01    0.000e+00    2.633e-02    1.510e-01
   11      49    2.123087e-01    0.000e+00    1.999e-02    6.390e-01
   12      53    1.614917e-01    0.000e+00    5.402e-03    1.801e+00
   13      57    1.590501e-01    0.000e+00    4.002e-03    1.598e-01
   14      61    1.515581e-01    0.000e+00    7.335e-04    1.589e-01
   15      65    1.513114e-01    0.000e+00    4.004e-05    3.735e-03
   16      69    1.512340e-01    0.000e+00    2.396e-06    1.063e-03
   17      73    1.512340e-01    0.000e+00    4.000e-07    1.490e-05

Local minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the default value of the optimality tolerance,
and constraints are satisfied to within the default value of the constraint tolerance.

<stopping criteria details>

Optimal Design Parameters
L1 = 1.0000 [cm]
L2 = 3.7575 [cm]
L3 = 1.0000 [cm]
```

Comparison plot between desired vs. optimal trajectory based on the optimal design parameters.



พบว่า Optimal Length มีค่า [1 3.7575 1] โดยค่าความผิดพลาด J มีค่า 0.151234 แต่จากภาพ พบว่า Optimal Path ไม่ใกล้เคียงกับ Desired Path ทั้งนี้เป็นผลมาจากค่าความเร็วเชิงมุมที่หาได้จาก Part I มีค่าไม่เท่ากับความจริง และหากจะใช้ค่านี้ต่อไป Optimal Length จะต่ำกว่า Lower bound ซึ่ง [1 1 1] คือข้อจำกัดทางกายภาพของ Simulink

Part IV

Problems and Problem-Solving

- ในช่วงแรก เมื่อรันโค้ดใน MATLAB พบว่าค่าลู่เข้าสู่ค่า ต่ำสุด (lower bound) ที่ตั้งเอาไว้ จึงได้ลดค่าต่ำสุดลง

- เมื่อลดค่าต่ำสุดลงกระทั่งต่ำกว่า [1 1 1] พบว่าเกิด Error ทาง Geometry จึงใช้ค่าต่ำสุดที่ [1 1 1]

- เมื่อรันโค้ดพบว่าคำตอบที่ได้มีความผิดพลาดมากกว่า 0 มาก กล่าวคือค่าที่ได้มาจากจุดต่ำสุดสัมพัทธ์ (local minima) แก้โดยการสุ่มค่าความยาวเริ่มต้นหลายๆค่า แต่พบว่าไม่มีค่าที่ทำให้ความผิดพลาดเท่ากับ 0 จึงใช้ความยาวเริ่มต้นที่ทำให้ความผิดพลาดต่ำสุด

- เมื่อรันโค้ดด้วยค่าความยาวเริ่มต้นที่ทำให้ค่าความผิดพลาดต่ำสุด พบว่าได้ path ที่ไม่ตรงกับที่ต้องการ ทั้งนี้คาดว่าเป็นเพราะการขยับในแกนที่ 3 รวมถึงความผิดพลาดที่เกิดจากการใช้โปรแกรม Tracker ในหาร plot

Reference

1. Meriam, J.L., and Kraige L.G., **Engineering Mechanics Dynamics**, 7th ed. John & Wiley Sons, New York.