PREFACE

First, we would like to thank the HCMC University of Technology, specially Assoc. Prof. Dr. HO PHAM HUY ANH, M.s CAO VAN KIEN and TRAN THIEN HUAN for providing such prestigious opportunity. Through the thesis, we student have a clearer looks and tremendous experience in the world of robot design, and this is the most memorable subject that we ever did in our university life.

At the time of free market economy, the application and exploration of new technology is emerging to be one of the most urgent priorities in our country, especially the robotic industry. The robot can assist humanity in different ways in every aspect. As a result, this field is being researched intensely every day. To prove the passionate to the robot, this thesis focused on the Biped robot research and development. Biped robot is a form of a walking robot using several control techniques. With the main purpose is to figure out the control algorithm, the biped robot can be applied in different ways to develop many field of scientific research and technology. Besides, this work can prove the precision of the control system.

This thesis will use 2 algorithms: Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) and geometric method to solve problems of biped. The geometric method is applied on calculating angle rotations of motors such that the harmony between the spins generating the the shape of the robot while it moving. After that, a comparison between GA and PSO is conducted with different parameters to make commentary and notation on the balancing of the biped robot. Finally, the thesis concludes the optimized results and suggests new directions in the future.

Table of Contents

HCM University of Technology SOCIALIST REPUBLIC OF VIETNAM

Deps of Electronics & Electrical Engineering Independence – Freedom – Happiness

----------*---------- ----------*----------

No: ……/HCMUT

THESIS'S WORKS

Students: KHƯU BẠCH THY STUDENT ID: ILI12110

NGUYỄN HỒ HIẾU TRUNG STUDENT ID: ILI12053

Major: AUTOMATION AND CONTROL Class: CT13COA1

1. Thesis Topic: **WALKING GAIT GENERATION FOR BIPED ROBOT**
2. Thesis' works:

a) Study on the fundamental concepts of Trajectory.

b) Study on the principle operation of **Biped**.

c) Programming hardware with Arduino.

d) Simulate a mathematical algorithm in MATLAB.

e) Conclusion on thesis's works.

3. Starting date: 04/2016

4. End date: 01/2017

5. Supervisor: ASC.PROF HỒ PHẠM HUY ÁNH

The contents and requirements of thesis are approved by Head of Division.

Ho Chi Minh City, …/…/……

HEAD OF DIVISION SUPERVISOR

(Signature) *(Signature)*

PART FOR DEPARTMENT, SUBJECT:

Censor: .....

Institution: .....

Protection date: .....

Total scores: .....

Dissertation repository: .....

CHAP 1: OVERVIEW

1.1. Abstract

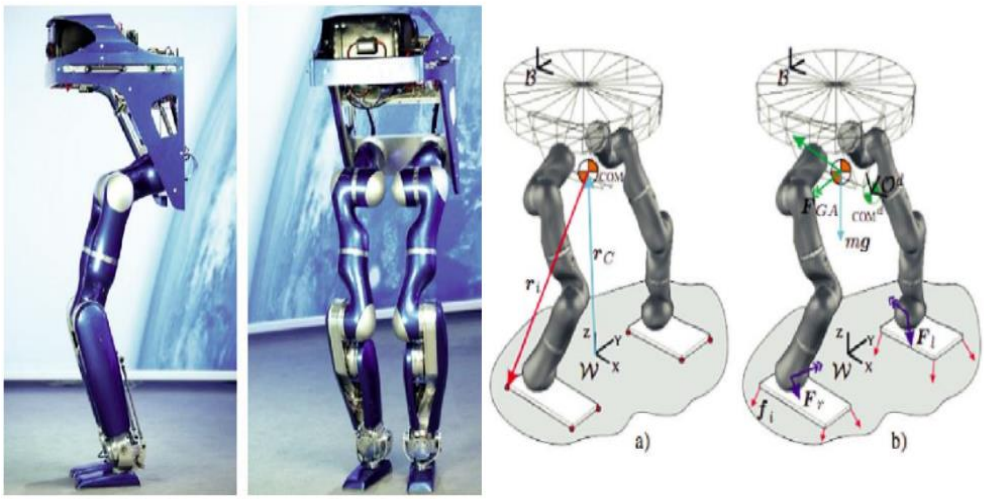
Bipedal walk as an activity requires an excellent sensorial and movement integration to coordinate the motions of different joints, getting as a result an efficient navigation system for a changing environment. Main applications of the study of biped walking

are in the field of medical technology, to diagnose gait pathologies, to take surgical decisions, to adequate prosthesis and orthoses design to supply natural deficiencies in people and for planning rehabilitation strategies for specific pathologies. The same principles can also be applied to develop biped machines. In daily situations, a biped robot would be the best configuration to interact with humans and to get through an environment difficult for navigation. If the biped robot is designed with human proportions, the robot could manage his way through spaces designed for humans, like stairs and elevators, and hopefully the interaction with the robot would be similar to interaction with a human being. In this project, we agree that mathematical algorithm and programming is very important. First, we abstract some specific requirements of this problem, including control technique, real-time monitoring, power duration of transmitters, Arduino robot high accuracy controlling. Based on these requirements and the characteristics of MCU, several research challenges in terms of new knowledge in balancing robot as well as hardware and software support are examined. Finally, we conclude that wireless sensor network is very powerful and suitable tool to be applied in this application. Finally, we describe the system architecture in this project, including the schematic of Biped Robot and base station circuit. Finally, we take out the advantages and things we should improve in this system in the future.

1.2. Researches in the world and nationality

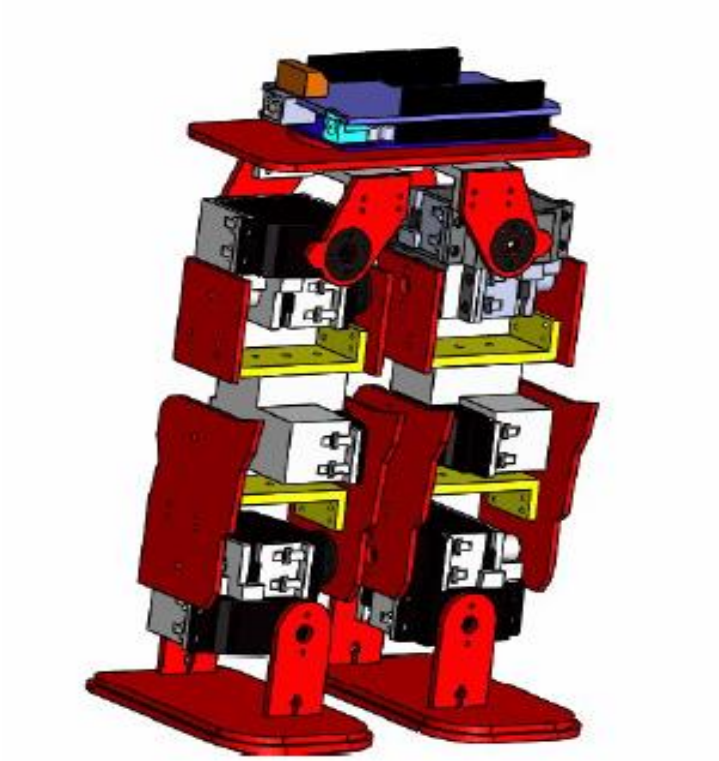
In the World, the development of the Biped Robot has become a global trend as more and more scientists in developing countries are finding the best ways to optimize their robots, avoid falling, also doing number of foot step by forward moving. The serious shortage of human resources has led to the continuous development of robotic services to serve people. Actually, Japan is the original country which is the one of top three nationals in the world researching about robot fields. Their robots take a responsibility either more humanly or more intelligently. An example of this is the ASIMO robot [1], which is one of the most famous pioneer robot. With the introduction of the I-Walk system, flexible and intelligent stepping-motives, in real time, ASIMO can be able

walked directly and continuously by control. It is also perfective about the reflection and stability while taking an encountering situations of sudden movement. Their robots even walk in more difficult conditions such as on stairs, rugged roads, etc.



In order to get that pointless, optimization is a tool of calculation which can be applied in a wide variety of areas, such as smoother operation and robot movement speed. However, ensuring the smoothness of the robot moves while making sure that the robot does not fall is the most urgent problem [2].

In Viet Nam, ASC.PROF HO PHAM HUY ANH and his colleagues applied the research proposal by using Genetic Algorithm (GA) to optimize the parameters affecting the gait of small-size Biped Robot with copper satisfying constraints. The speed and stability of the gait for biped robots are the same as human beings.



1.3. Topic, purpose and thesis detail

In 2016, Assoc. Prof. Dr. HO PHAM HUY ANH, M.S. TRAN THIEN HUAN and their college students continue to research about creating robot gaits. By applying the theory of opened-loop system control and newly Particle Swarm Optimization (PSO) to optimize the parameters that influence to the trajectory, we can generate some stable gaits for biped in small size model. Moreover, the robot is simultaneously constrained in both velocity and acceleration. This topic focuses on how to create a variety of stably walking gaits such as high lift, low legs, long legs and short legs.

1.4. Work assignments

The target of work is to create some walking gaits for biped robot that usually stable while moving on the flat surface. The thesis was planned and practiced during half years. The time to do works could be divided into 2 main parts such as simulation and experiment. The work for simulation had been making for 5 months and the experiment for 1 month. The two students in group were clearly taking an individual responsibility

- THY's works: Research about simulation, including practical model
- TRUNG's works: Study about how to apply GA and PSO to create or optimized working-gaits for biped

CHAP II: WALKING TRAJECTORY

2.1. Biped model

The biped robot consists of two legs. The waist-link connects the two legs. Each leg has three links: foot-link, shank-link, and thigh-link. The joint between the foot-link and shank-link is the ankle, the joint between shank-link and thigh-link is the knee while the one between thigh-link and waist-link is the hip. The biped has 10 DOFs, including 2 DOFs at ankle, 1 DOF at knee, and 3 DOFs at hip. It has total 5 DOFs in each leg [3].

The biped model can be simply described by geometry view as figure (2.1). Each DOF corresponds to independent servo motor.

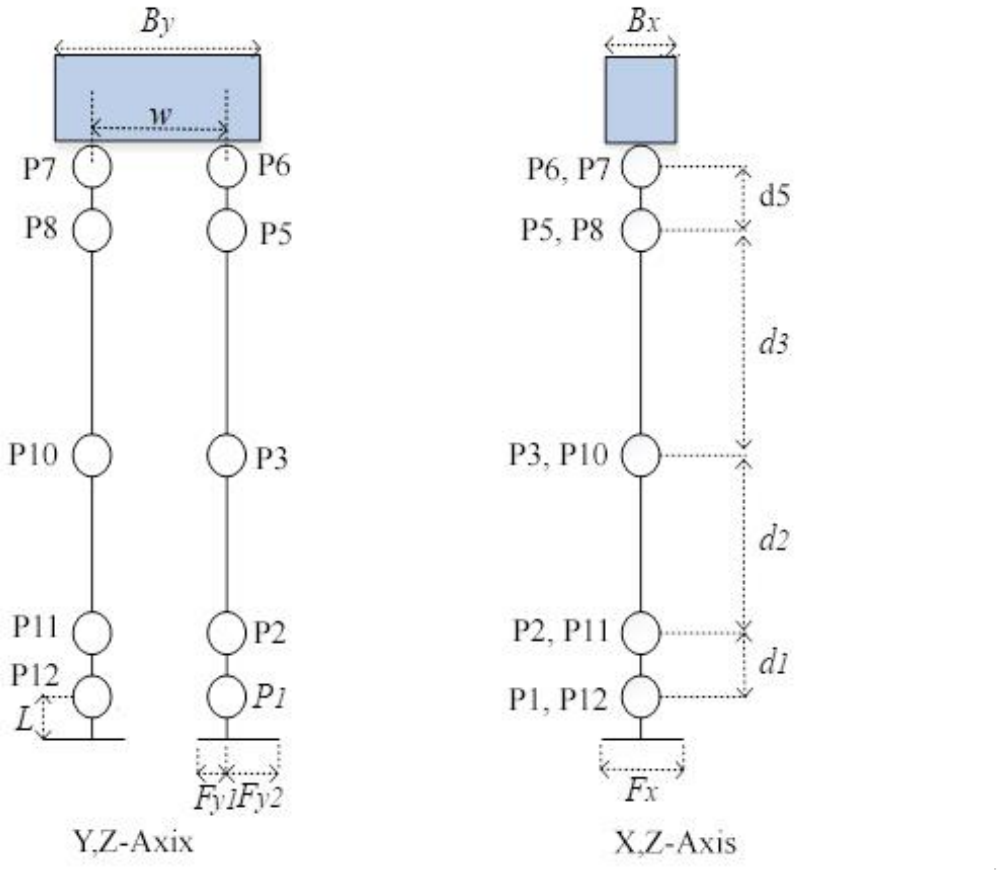


Fig.2.1: The geometry model in 2D space

The physical parameters are distributed, shown in table (2.1). ‘‘d1” indicates the thigh-link length and “d2” the shank-link length. “Fy” and “Fx” are the foot width and length. “By” and “Bx” are the hip-link width and length. The distance between two hip-joints is “w”, and the distance between the ankle-joint and center of the feet is “L”

Table 2.1: The physical parameters of biped robot

Parame-ters	**Values**	**Parame-ters**	**Values**
d_1 d_{11}	40mm	F_{y1}	20mm

d_2 d_{10}	45mm	F_{y2}	43mm
d_3 d_8	62mm	B_x	55mm
d_5 d_7	62 mm	F_x	85mm
L	10 mm	w	66mm

2.2. Walking gait generation

Biped robot has the 4 important parameters to create a structure of walking gait such as step-length (S), bending-height (h), maximum lifting-height (H) and maximum frontal-shift (n) [4]. This can be described as following figure (2.2.1)

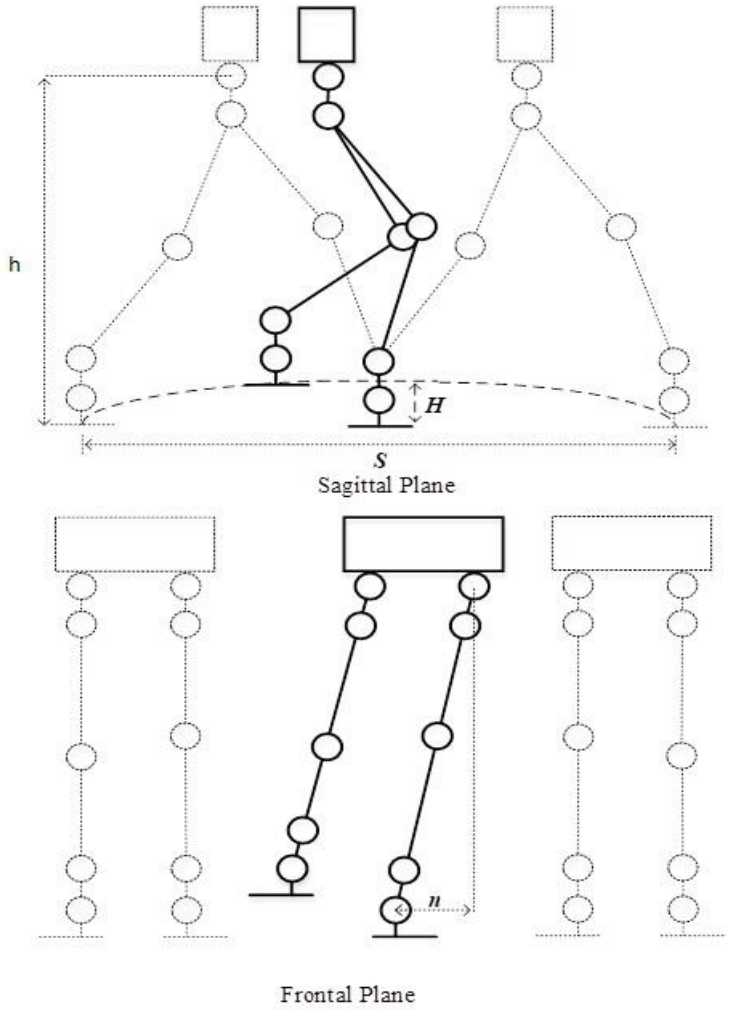


Fig.2.2.1: The biped's walking gait in 2D space

The biped walking motion is generated by choosing an appropriate time function for the three reference points: $P_1(P_{1x}, P_{1y}, P_{1z})$, $P_6(P_{6x}, P_{6y}, P_{6z})$, and $P_{12}(P_{12x}, P_{12y}, P_{12z})$, which will be introduced in the next part 2.3. Inside that, P_1 is stationary for $0< t<T$ and acts as a reference for P_6 and P_{12}. For $T<t <2T$, the positions of P_1 and P_{12} are inter-changed period of a step is indicated in figure (2.2.2).

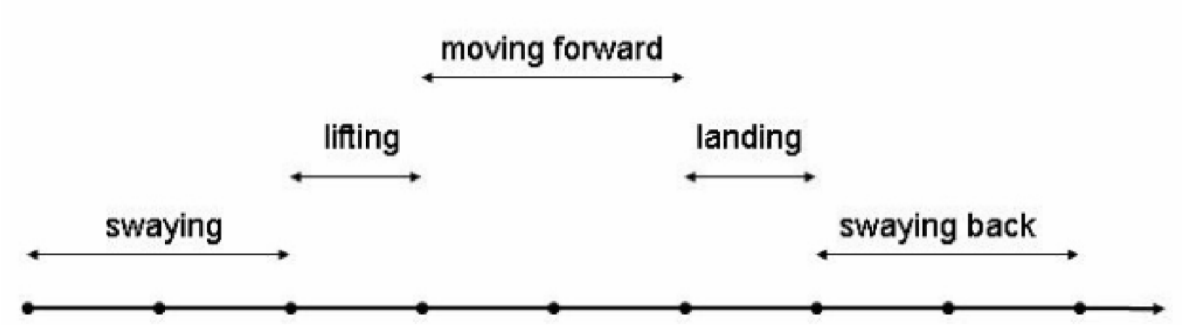


Fig.2.2.2: The cycle of forward walking process

2.3. Trajectory generation of supporting-foot, swing-foot and hip

Base on theory of DIP GOSWAMI and QIANG Huang [5] [8], gait-waking of a bi-ped robot was simply defined by main four elements such as step-length (**S),** lifting-height (**H),** bending-height (**h)** and frontal-shift **(n)**. The supporting-foot, swing-foot and hip are affected by the value of these four parameters.

The value of them could be found by understanding GA and PSO, also how to use the applied tool, will be introduced in the chapter 3. An example of this case is chosen as:

- ✓ Step-length (**S**=11)
- ✓ Lifting-height (**H**=2)
- ✓ Bending-height (**h**=3)
- ✓ Frontal-shift (**n**=5**)**

2.3.1. Reference trajectory of supporting-foot $P_1 = [P_{1x}, P_{1y}, P_{1z}]$

Suppose that $P_1(0,0,0)$ is set up on the center of supporting-feet. Making a survey on one walking period from $0 < T < 1$ (s), we can see P_1 stands still during that time as simulated picture (2.3.1.1). Thus, the trajectory of supporting-foot P_1 is also found by following method (2.3.1.1):

$$P_1 = \begin{cases} 0 \\ 0 \quad 0 < T < 1 \quad (s) \\ 0 \end{cases} \quad (2.3.1.1)$$

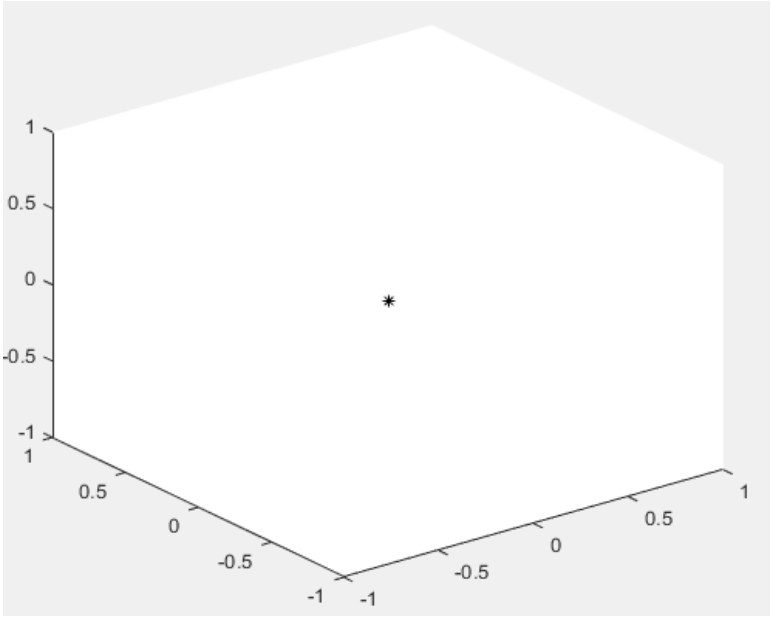


Fig.2.3.1.1: The trajectory of supporting-foot P_1

2.3.2. Reference trajectory of swing-foot $P_{12} = [P_{12x}, P_{12y}, P_{1z}]$

Suppose that P_{12} is set up on the center of swing-feet. Making a survey on one walking period from $0 < T < 1$ (s), we divide its time into 2 separate stages T_1 and T_2. The time of first stage $0 < T_1 < 0.2$ (s) while robot is performing knee binding, the swing-feet P_{12} is standing still. After that process, the time of second stage $0.2 < T_2 < 1$

(s) happens while robot is lifting his swing-feet P_{12}. From these two surveyed stages, we make a conclusion for P_{12x}, P_{12y}, P_{12z} as alternative simulated picture (2.3.2.1), method (2.3.2.1), simulated picture (2.3.2.2), method (2.3.2.2) and simulated picture (2.3.2.3), method (2.3.2.3):

$$P_{12x} = \begin{cases} \frac{S}{2} & for \quad 0 < T < 0.2(s) \\ \frac{S}{2}\sin\left(\frac{2\pi}{1.58}\mathrm{T} + 3.87\right) & for \quad 0.2 < T < 1(s) \end{cases} \quad (2.3.2.1)$$

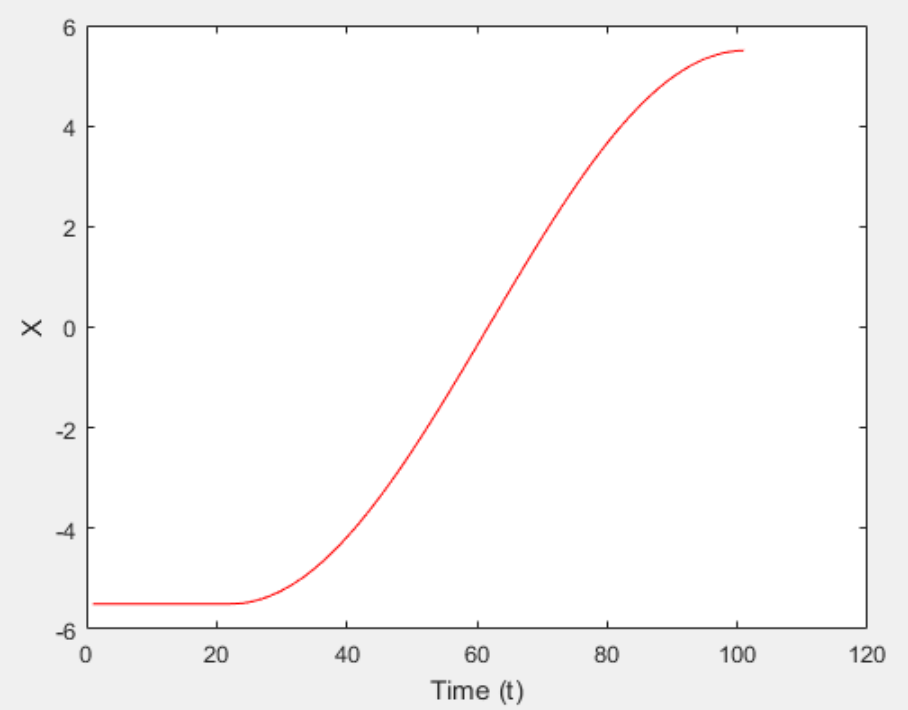


Fig.2.3.2.1: The trajectory of P_{12x} during one period 1s

$$P_{12y} = -w \quad for \quad 0 < T < 1(s) \quad (2.3.2.2)$$

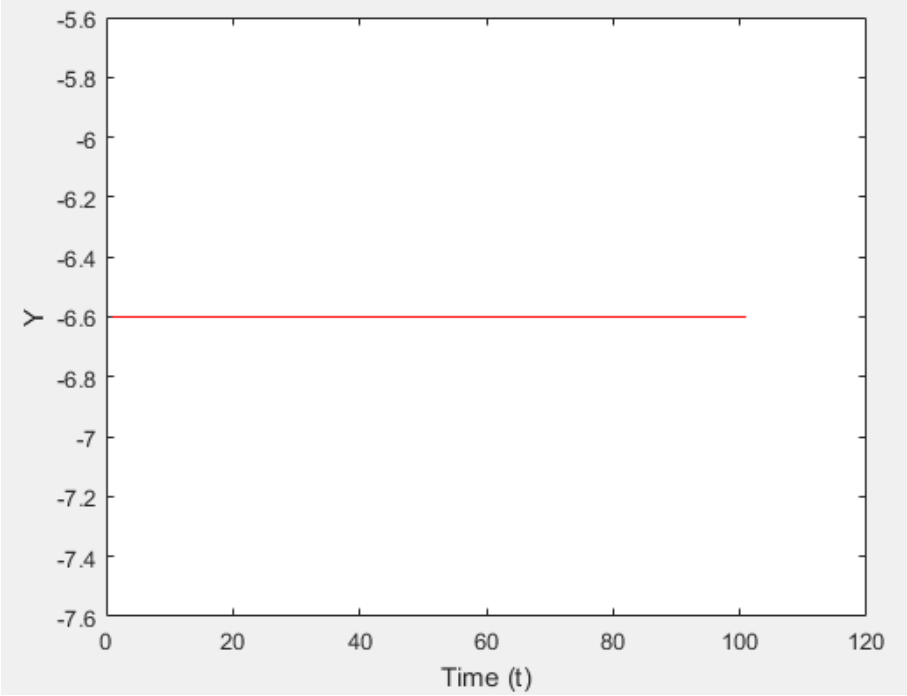


Fig.2.3.2.2: The trajectory of P_{12y} during one period 1s

$$P_{12z} = \begin{cases} 0 \quad for \quad 0 < T < 0.2(s) \\ \text{Hsin}\left(\dfrac{2\pi}{1.58}\text{T} - 0.832921\right) \quad for \quad 0.2 < T < 1(s) \end{cases} \quad (2.3.2.3)$$

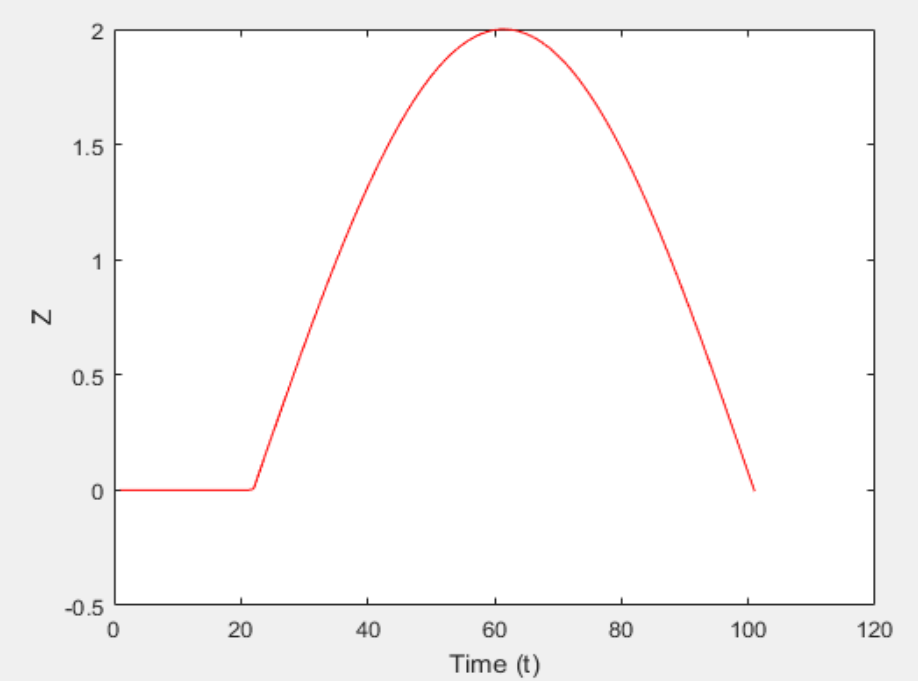


Fig.2.3.2.3: The trajectory of P_{12z} during one period 1s

In conclusion, the trajectory P_{12} (x, y, z) during 1s one walking period could be simulated as picture (2.3.2):

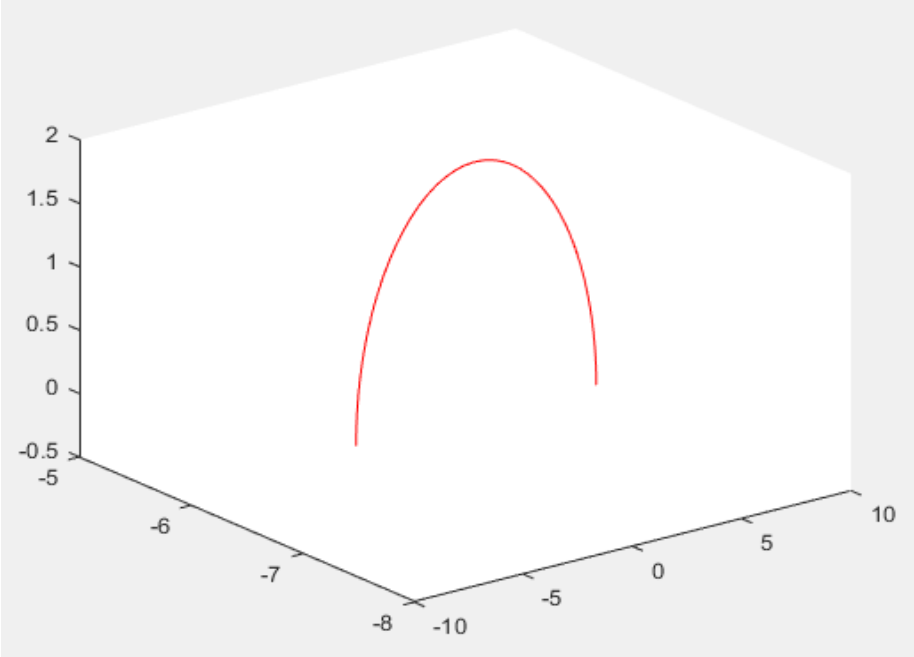


Fig.2.3.2: The trajectory of P_{12} during one period 1s

2.3.3. Reference trajectory of hip $P_6 = [P_{6x}, P_{6y}, P_{6z}]$

Suppose that P_6 is symbolled for the location of hip. Making a survey on one period of walking from $0 < T < 1$ (s), we divide its time into 3 separate stages T_1, T_2 and T_3. The time of first stage $0 < T_1 < 0.2$ (s) while robot is making knee binding. Then, the time of second stage $0.2 < T_2 < 0.8$ (s) when robot starts to hip swinging. Finally, the time of stage $0.8 < T_2 < 1$ (s) when robot is swinging hip back to initial. From mentioned describe, we make a conclusion for P_{6x}, P_{6y}, P_{6z} as alternative simulated picture (2.3.3.1), method (2.3.3.1), simulated picture (2.3.3.2), method (2.3.3.2) and simulated picture (2.3.3.3), method (2.3.3.3):

$$P_{6x} = \frac{S}{4}\sin(-\pi T - \frac{\pi}{2}) \quad for \quad 0 < T < 1(s) \quad (2.3.3.1)$$

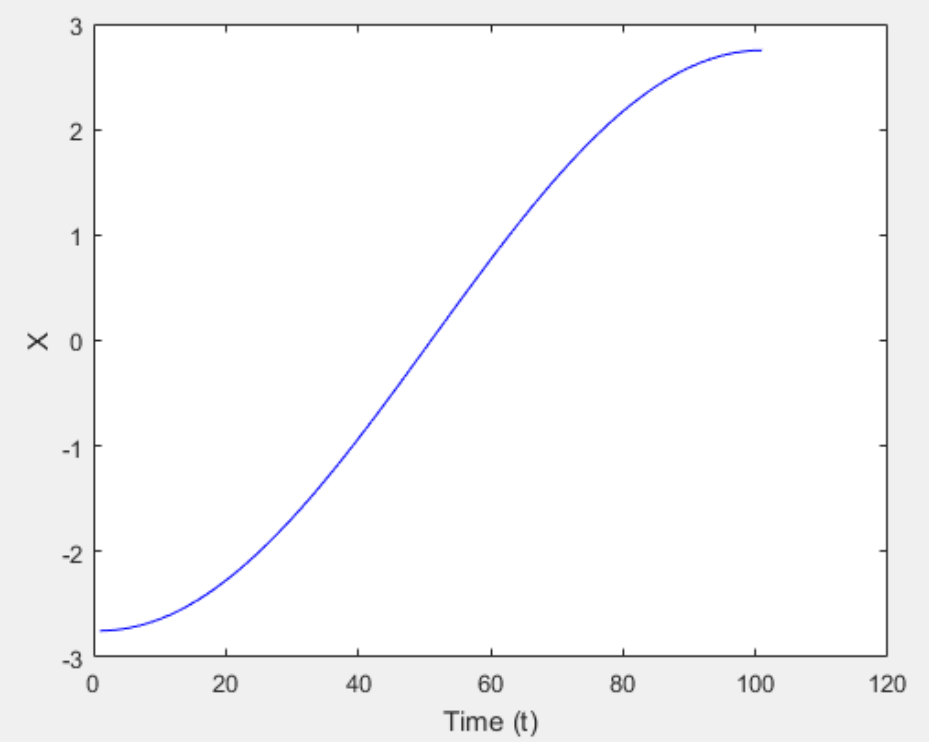


Fig.2.3.3.1: The trajectory of P_{6x} during one period 1s

$$P_{6y} = \begin{cases} \text{nsin}\left(\frac{2\pi}{0.8}T\right) & for \quad 0 < T < 0.2(s) \\ n \quad for \quad 0.2 < T < 0.8(\text{s}) \\ \text{nsin}\left(\frac{2\pi}{0.8}\text{T} + \frac{\pi}{2}\right) & for \quad 0.8 < T < 1(s) \end{cases} \quad (2.3.3.2)$$

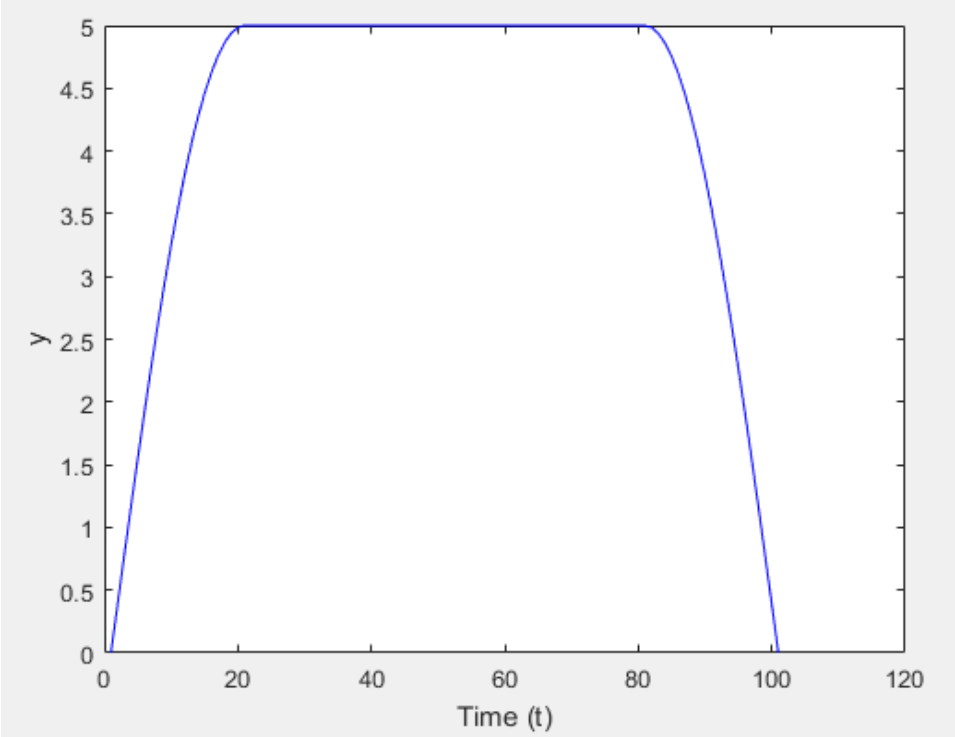


Fig.2.3.3.2: The trajectory of P_{6y} during one period 1s

$$P_{6z} = (d_1 + d_2 + d_3 + d_5) - h \quad for \quad 0 < T < 1(s) \quad (2.3.3.3)$$

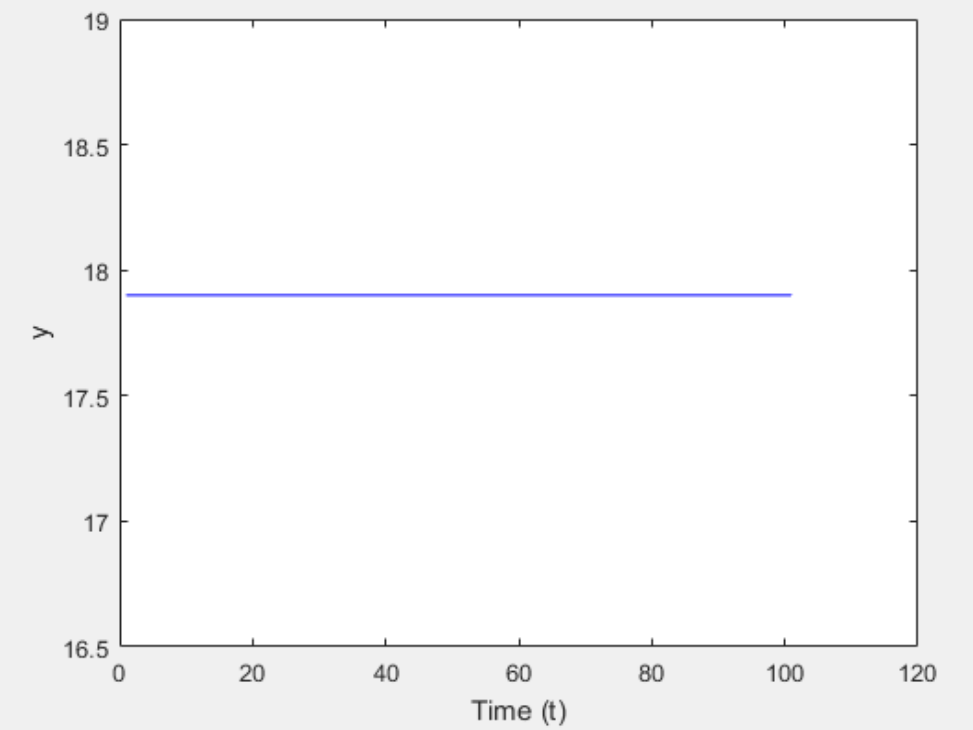


Fig.2.3.3.3: The trajectory of P_{6z} during one period 1s

In conclusion, the trajectory P_6 (x, y, z) during 1s one walking period could be simulated as picture (2.3.3):

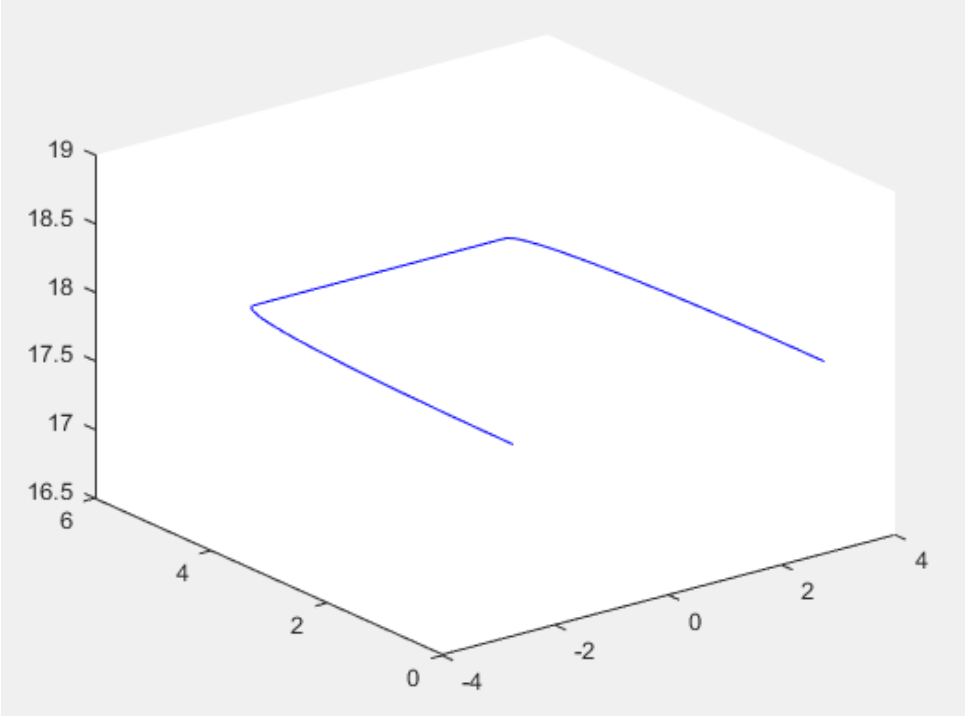


Fig.2.3.3: The trajectory of P_6 during one period 1s

The code of trajectory generation be shown in the APPENDIX 1 (p.62)

2.4. Forward Kinematics

In robotics literature, the meaning of forward kinematics is commonly known as the task in which the position and orientation of the end-effector is to be determined by giving the configurations for the active joints of the robot [23]. Based on biped model design, there consists of 5 joints on each foot namely position of knee, knee and ankle. In actually, the main purpose of this section is to find the position of these joints. By using the basic geometry method and we suppose the rotated direction for 10 servo motors as sequence (θ_1, θ_2, θ_3, θ_5, θ_6, θ_7, θ_8, θ_{10}, θ_{11}, θ_{12}) where they were installed on the robot framework, we can find out the 10 positions of those corresponding angle rotations (P_1, P_2, P_3, P_5, P_6, P_7, P_8, P_{10}, P_{11}, P_{12}). This work is simply modeled by the figure (2.4.1)

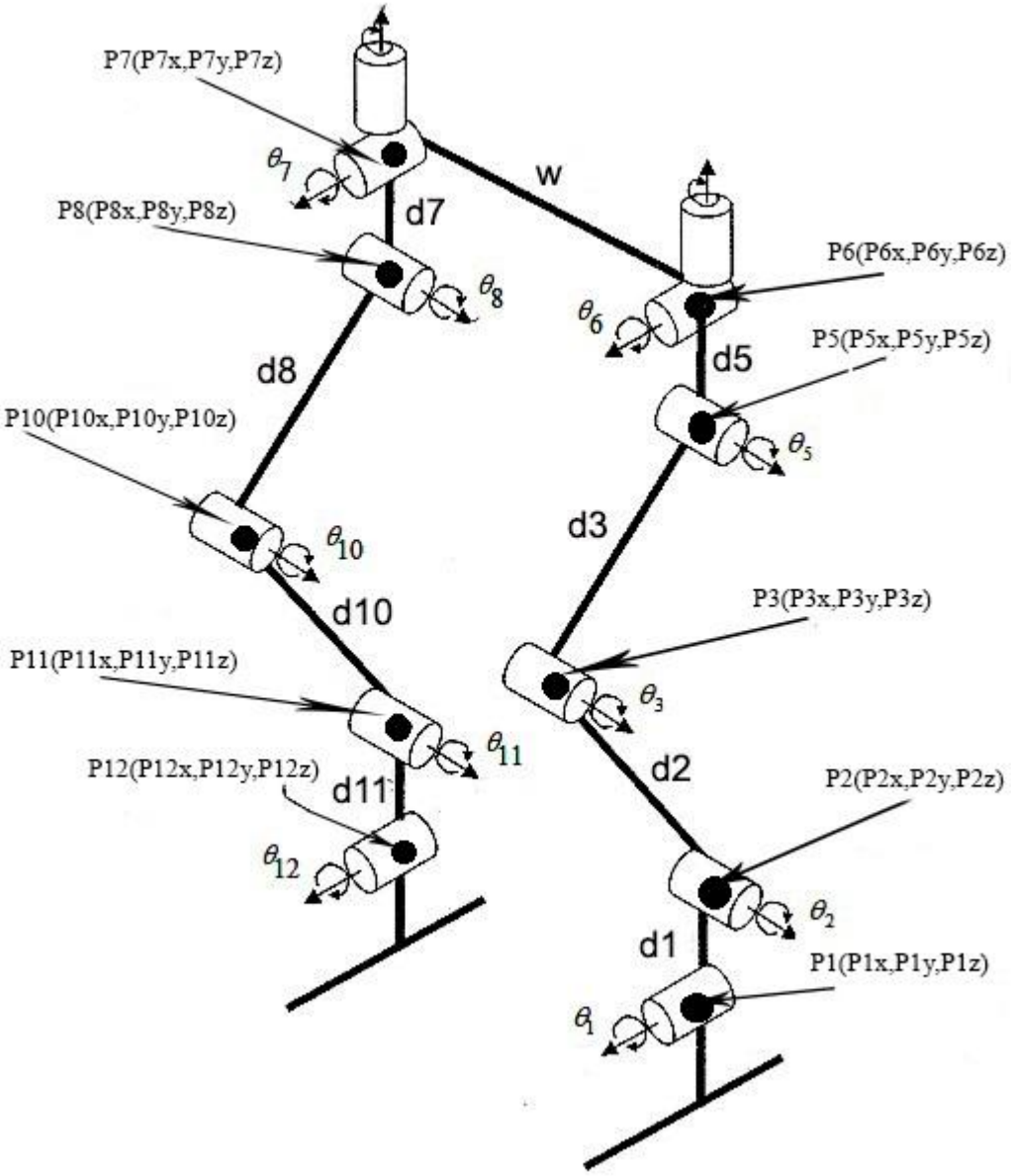


Fig.2.4.1: Joints and rotations distribution on the biped model

The first half work of kinematic problem, we apply geometry method to find the position of the first 5 joints including P_1, P_2, P_3, P_5 and P_6. We select $P_1 = [P_{1x}, P_{1y}, P_{1z}]$ is the origin where it is at the center of the left-feet. Thus, the position of P_1 is defined as the formula (2.4.1)

$$\begin{cases} P_{1x} = 0 \\ P_{1y} = 0 \\ P_{1z} = 0 \end{cases} \quad (2.4.1)$$

The picture (2.4.2) describes the relation between P_1 and P_2. The position of $P_2(P_{2x}, P_{2y}, P_{2z})$ is moved under the control of angle rotation θ_1 at P_1 and d_1 is the distance between P_1 and P_2, so P_2 does not change by direction x-axis. Then, we have the position of P_2 as the formula (2.4.2)

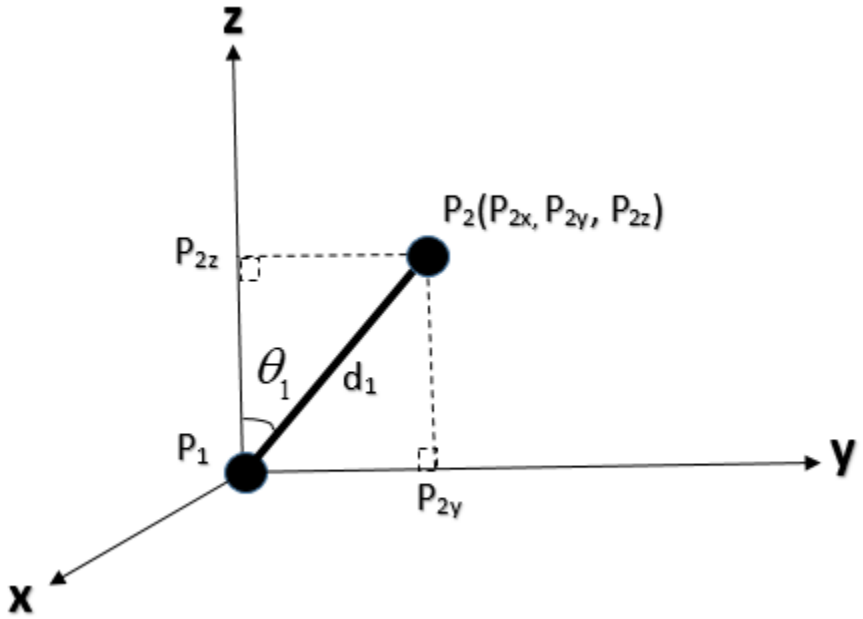


Fig.2.4.2: The relation between P_1 and P_2 by angle rotation θ_1

$$\begin{cases} P_{2x} = P_{1x} \\ P_{2z} = d_1 \cos(\theta_1) \\ P_{2y} = P_{2z} \sin(\theta_1) \end{cases} \quad (2.4.2)$$

The picture 2.4.3 show the relation between P_1, P_2 and P_3. The position of $P_3(P_{3x}, P_{3y}, P_{3z})$ is moved under the control of angle rotations (θ_1 , θ_2) at P_1 and P_2 and d_2 is the distance between P_2 and P_3. Thus, we can find out the formula (2.4.3) which describes the position of P_3

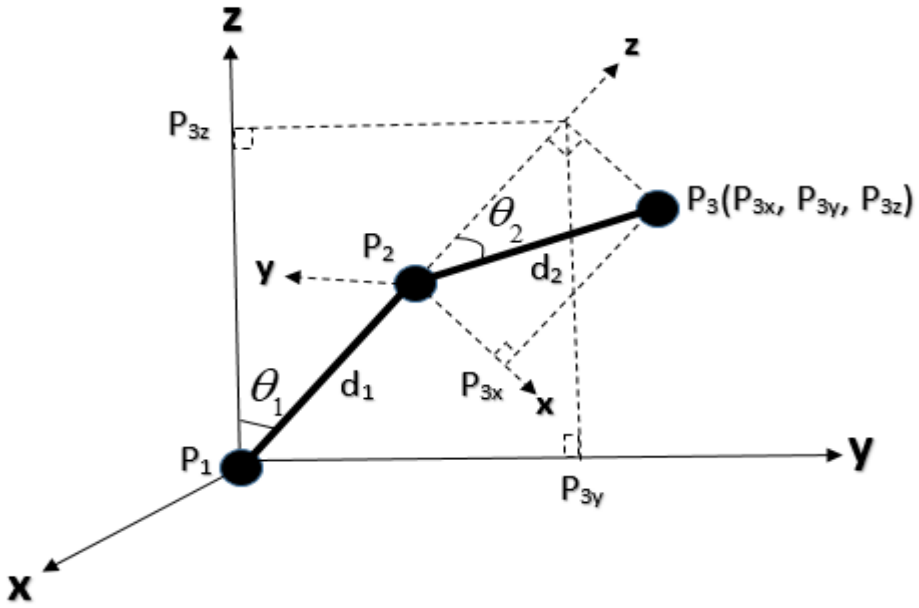


Fig.2.4.3: The relation between P_1, P_2 and P_3 by angle rotations θ_1, θ_2

$$\begin{cases} P_{3x} = d_2 \sin(\theta_2) \\ P_{3z} = P_{2z} + d_2 \cos(\theta_2) \times \cos(\theta_1) \\ P_{3y} = P_{3z} \sin(\theta_1) \end{cases} \quad (2.4.3)$$

The relation between P_1, P_2, P_3 and P_5 is illustrated by the picture 2.4.3. The position of $P_5(P_{5x}, P_{5y}, P_{5z})$ is moved under the control of angle rotations ($\theta_1, \theta_2, \theta_3$) at P_1, P_2 and P_3 and d_3 is the distance between P_3 and P_5, so P_5 is defined as following formula (2.4.3)

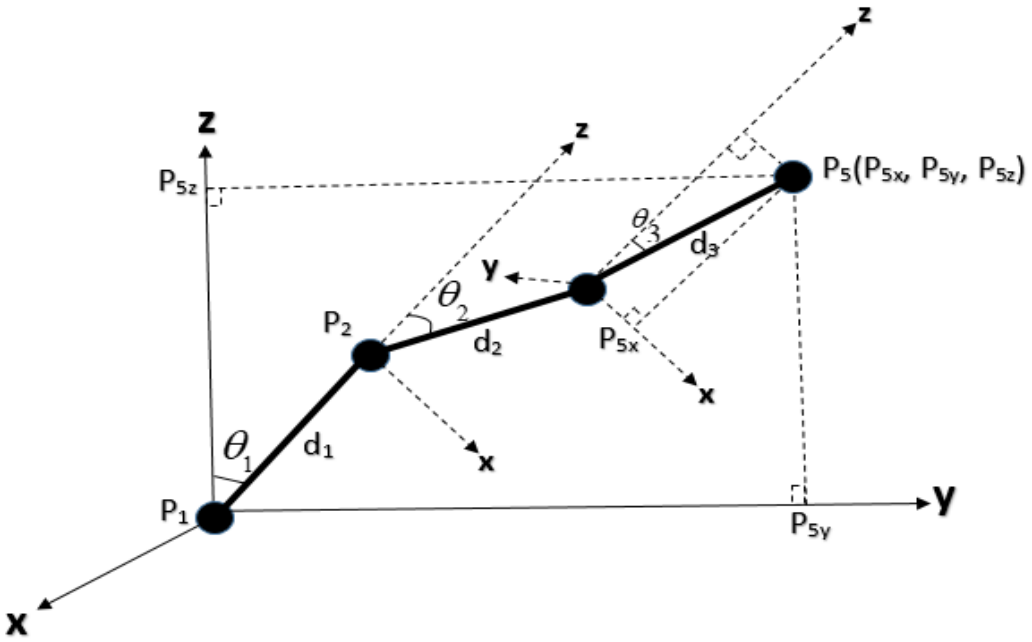


Fig.2.4.3: The relation between P_1, P_2,P_3 and P_5 by angle rotations $\theta_1, \theta_2, \theta_3$

$$\begin{cases} P_{5x} = P_{3x} + d_3 \sin\left(\theta_2 - \theta_3\right) \\ P_{5z} = P_{3z} + d_3 \cos\left(\theta_2 - \theta_3\right) \times \cos\left(\theta_1\right) \\ P_{5y} = P_{5z} \sin\left(\theta_1\right) \end{cases} \quad (2.4.3)$$

The relation between P_1, P_2, P_3, P_5 and P_6 is showed by the picture (2.4.3). The position of $P_6(P_{6x}, P_{6y}, P_{6z})$ is moved under the control of angle rotations ($\theta_1, \theta_2, \theta_3, \theta_5$) at P_1, P_2, P_3 and P_5. d_5 is the distance between P_5 and P_6, so P_6 is defined as the following formula (2.4.3)

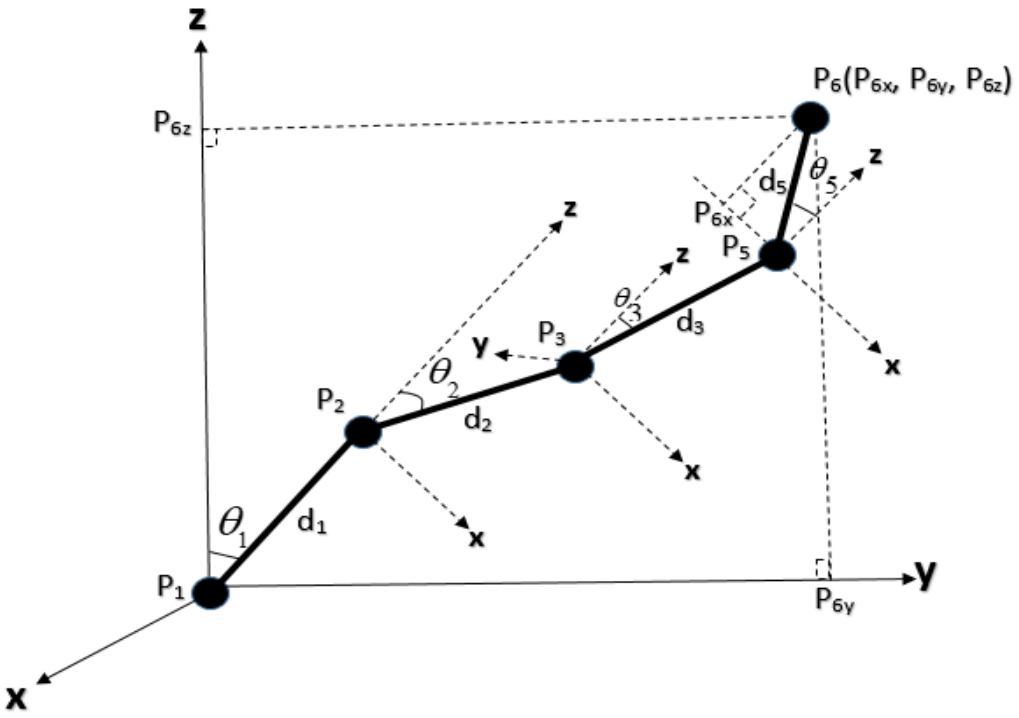


Fig.2.4.4: The relation between P_1,P_2,P_3,P_5 and P_6 by angle rotations $\theta_1, \theta_2, \theta_3, \theta_5$

$$\begin{cases} P_{6x} = P_{5x} + d5\sin\left(\theta_2 - \theta_3 + \theta_5\right) \\ P_{6z} = P_{5z} + d5\cos\left(\theta_2 - \theta_3 + \theta_5\right) \times \cos\left(\theta_1\right) \\ P_{6y} = P_{6z}\sin\left(\theta_1\right) \end{cases} \quad (2.4.4)$$

Here we complete a first half of kinematic problem. Next one we suppose that "w" is the distance between two legs and continue to solve how to find the other 5 positions as P_7, P_8, P_{10}, P_{11} and P_{12}. The P_7 is described by the picture (2.4.5) and also be defined by the formula (2.4.5)

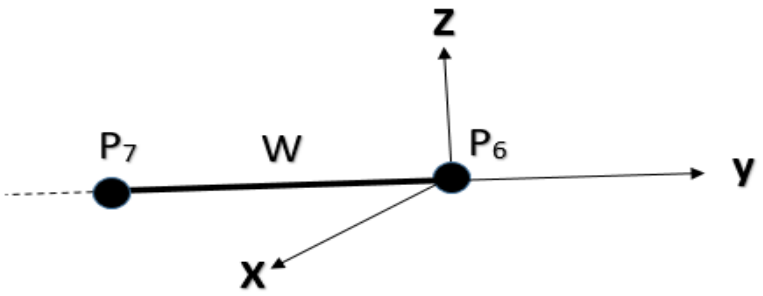


Fig.2.4.5: The relation between P_6 and P_7

$$\begin{cases} P_{7x} = P_{6x} \\ P_{7y} = P_{6y} - w \\ P_{7z} = P_{6z} \end{cases} \quad (2.4.5)$$

Suppose that d_7, d_8, d_{10} and d_{11} are alternately the distance between P_7 and P_8, P_8 and P_{10}, P_{10} and P_{11} then finally P_{11} and P_{12}. By the way above to find the position of P_1, P_2, P_3, P_5 and P_6, we could practice similarly on P_7, P_8, P_{10}, P_{11} and P_{12}, so they could be defined alternately by formulas (2.4.6), (2.4.7), (2.4.8) and (2.4.9)

$$\begin{cases} P_{8x} = P_{7x} \\ P_{8z} = P_{7z} - d_7 \times \cos(\theta_7) \\ P_{8y} = P_{7y} - (P_{7z} - P_{8z}) \times \sin(\theta_7) \end{cases} \quad (2.4.6)$$

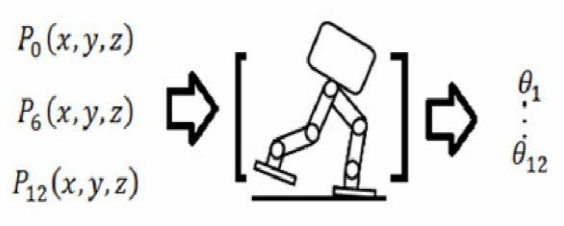
$$\begin{cases} P_{10x} = P_{8x} + d_8 \times \sin(\theta_8) \\ P_{10z} = P_{8z} - d_8 \times \cos(\theta_8) \times \cos(\theta_7) \\ P_{10y} = P_{7y} - (P_{7z} - P_{10z}) \times \sin(\theta_7) \end{cases} \quad (2.4.7)$$

$$\begin{cases} P_{11x} = P_{10x} + d_{10} \times \sin(\theta_8 - \theta_{10}) \\ P_{11z} = P_{10z} - d_{10} \times \cos(\theta_8 - \theta_{10}) \times \cos(\theta_7) \\ P_{11y} = P_{7y} - (P_{7z} - P_{11z}) \times \sin(\theta_7) \end{cases} \quad (2.4.8)$$

$$\begin{cases} P_{12x} = P_{11x} + d_{11} \times \sin(\theta_8 - \theta_{10} + \theta_{11}) \\ P_{12z} = P_{11z} - d_{10} \times \cos(\theta_8 - \theta_{10} + \theta_{11}) \times \cos(\theta_7) \\ P_{12y} = P_{11y} - (P_{7z} - P_{12z}) \times \sin(\theta_7) \end{cases} \quad (2.4.9)$$

The code of forward kinematics be shown in the APPENDIX 2 (p.63-64)

2.5. Inverse Kinematics



In this section, we discuss how to calculate the rotation angles of the biped robot ($\theta_1, \theta_2, \theta_3, \theta_5, \theta_6, \theta_7, \theta_8, \theta_{10}, \theta_{11}, \theta_{12}$) after we known the position of two legs [10] (P_1=[P_{1x}, P_{1y}, P_{1z}] and P_{12}=[$P_{12x}, P_{12y}, P_{12z}$]) and hip (P_6=[P_{6x}, P_{6y}, P_{6z}]) as the description in the following picture 2.5.1:

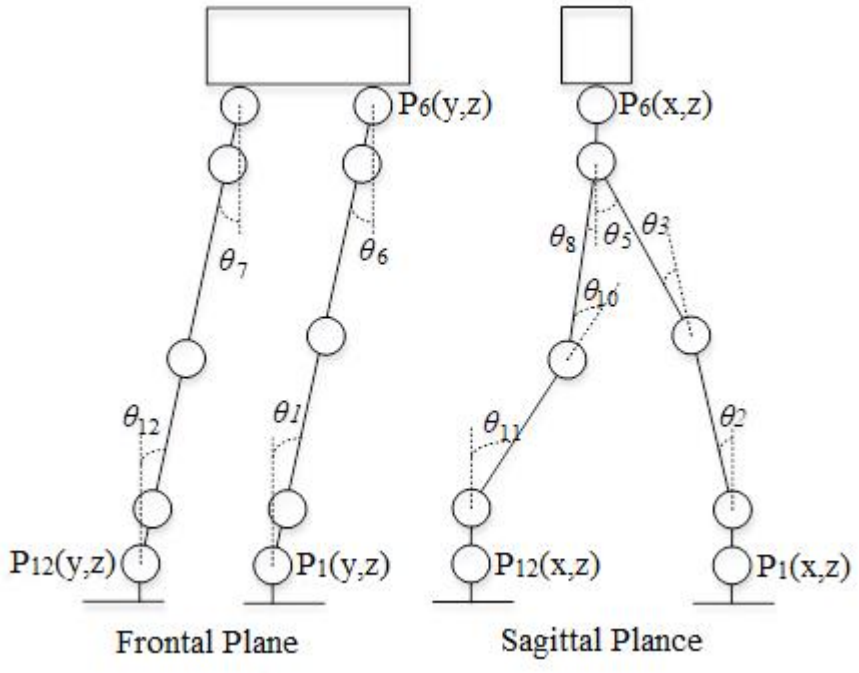


Fig.2.5.1: The relation between angle rotations and P_1, P_6, P_{12}

Biped perform walking in two periods:

- First period: The supporting leg $P_1(P_{1x},P_{1y},P_{1z})=P_1(0,0,0)$ and the moving leg $P_{12}(P_{12x},P_{12y},P_{12z})$

- Second period: The supporting leg $P_{12}=(P_{12x},P_{12y},P_{12z})=P_{12}(0,0,0)$ and the moving leg $P_1(P_{1x},P_{1y},P_{1z})$

The reverse dynamics problem can be solved by analytical or numerical methods. However, this section presents geometric methods [11]

Parameters of inverse kinematic model $\left(x_l, y_l, z_l, x_r, y_r, z_r, l_l, l_r, \theta_A, \theta_B, \theta_C, \theta_D\right)$ are defined in the picture (2.5.2) and method (2.5.2). However, we note that:

- l_l is the distance between P_2 and P_5
- l_r is the distance between P_8 and P_{11}

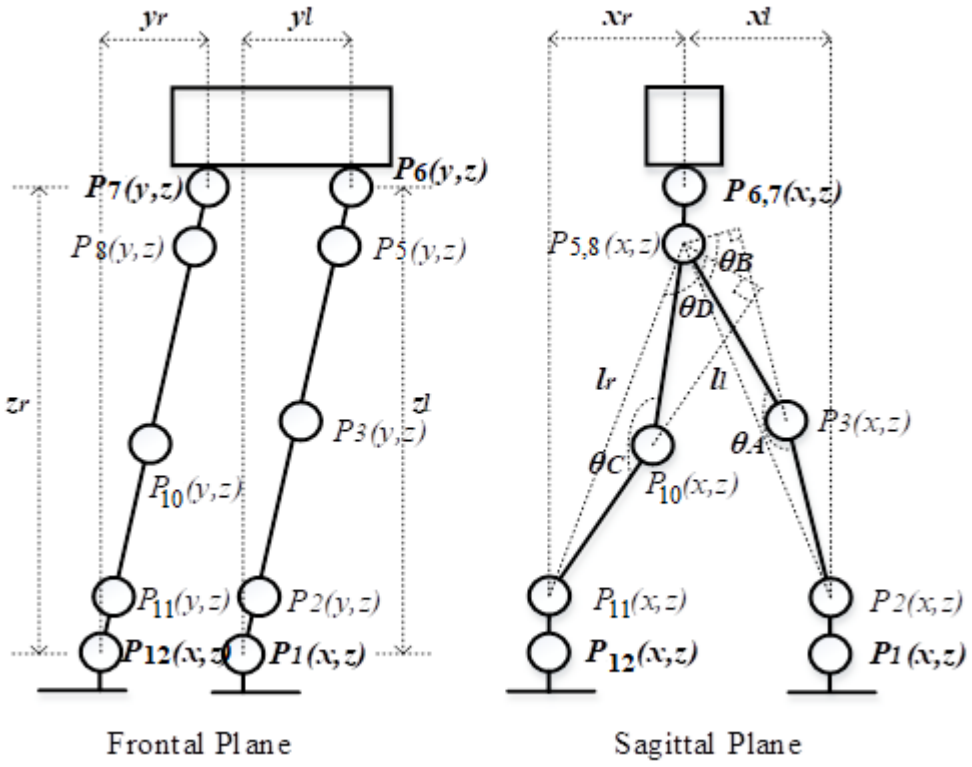


Fig.2.5.2: The parameters for inverse kinematics problem

Transition variables

$$\begin{cases} x_l = P_{5x} - P_{1x}; y_l = P_{5y} - P_{1y}; z_l = P_{5z} - P_{1z} \\ x_r = P_{6x} - P_{10x}; y_r = P_{6y} - P_{10y}; z_r = P_{6z} - P_{10z} \end{cases} \quad (2.5.2)$$

Sub-angles

$$\begin{cases} l_l = \sqrt{\left(P_{4x} - P_{2x}\right)^2 + \left(P_{4y} - P_{2y}\right)^2 + \left(P_{4z} - P_{2z}\right)^2} \\ l_r = \sqrt{\left(P_{7x} - P_{9x}\right)^2 + \left(P_{7y} - P_{9y}\right)^2 + \left(P_{7z} - P_{9z}\right)^2} \\ \theta_A = \arccos\left(\dfrac{d_2^2 + d_3^2 - l_l^2}{2d_2d_3}\right) \\ \theta_B = \arccos\left(\dfrac{d_2 \sin\left(\theta_A\right)}{l_l}\right) \\ \theta_C = \arccos\left(\dfrac{d_2^2 + d_3^2 - l_r^2}{2d_2d_3}\right) \\ \theta_D = \arccos\left(\dfrac{d_2 \sin\left(\theta_C\right)}{l_r}\right) \end{cases} \quad (2.5.2)$$

The angle of rotation of the two legs of biped is defined as formula (2.5.3)

$$\begin{cases} \theta_1 = \arctan\left(\dfrac{y_l}{z_l}\right); \theta_5 = -\theta_1 \\ \theta_{10} = \arctan\left(\dfrac{y_r}{z_r}\right); \theta_6 = -\theta_{10} \\ \theta_3 = \pi - \theta_A \\ \theta_8 = \pi - \theta_C \\ \theta_4 = \dfrac{\pi}{2} - \theta_A + \theta_B - \arcsin\left(\dfrac{x_l}{l_l}\right) \\ \theta_7 = \dfrac{\pi}{2} - \theta_C + \theta_D - \arcsin\left(\dfrac{x_r}{l_r}\right) \\ \theta_2 = \theta_3 - \theta_4 \\ \theta_{11} = \theta_9 - \theta_7 \end{cases} \quad (2.5.3)$$

The code of inverse kinematics be shown in the APPENDIX 3 (p.66)

2.6. ZMP trajectory

When study human's ZMP, ZMP does not stand in one place in supporting leg (Fig.2.6.1). In this fig, we see that the ZMP is moving from the heel to the hip

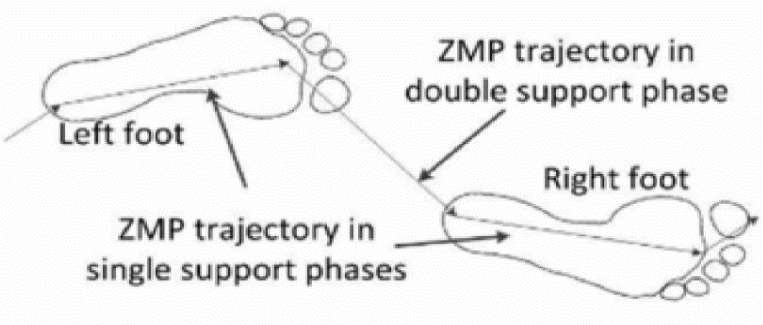


Fig.2.6.1: Human ZMP trajectory

Because of moving ZMP, human can walk with smooth velocity and continious acceleration. In order to get stable that robot not fall while moving forward, the ZMP usually obligates to be inside the supporting leg [7].

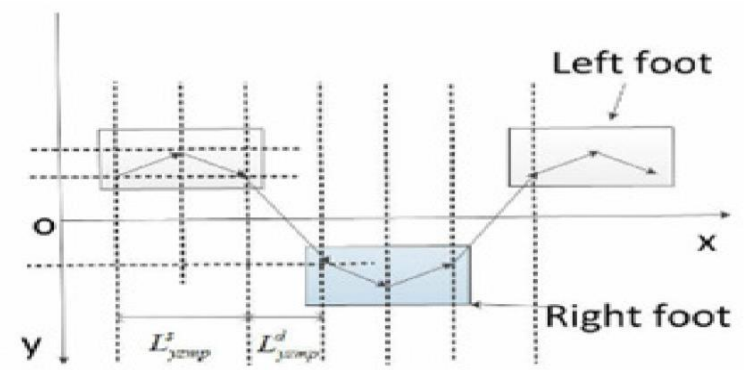


Fig.2.6.2: Planned ZMP trajectory

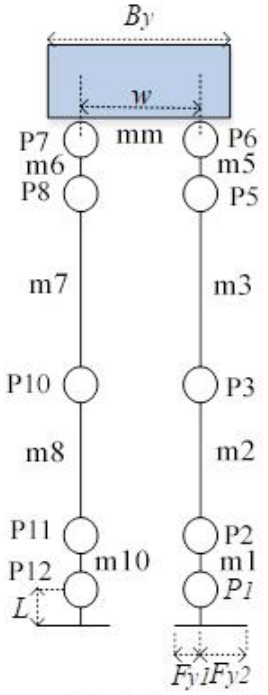
To calculate Q in each MOEA loop in the next chapter, the position of ZMP(x_{zmp},y_{zmp}) need to be found. First one, we have to define the position of center of mass (COM) that (x_{com}, y_{com} and z_{com}) by using the common formula (2.6.1)

$$\begin{cases} x_{com} = \dfrac{\sum_i m_i P_{ix}}{\sum_i m_i} \\ y_{com} = \dfrac{\sum_i m_i P_{iy}}{\sum_i m_i} \\ z_{com} = \dfrac{\sum_i m_i P_{iz}}{\sum_i m_i} \end{cases} \quad (2.6.1)$$

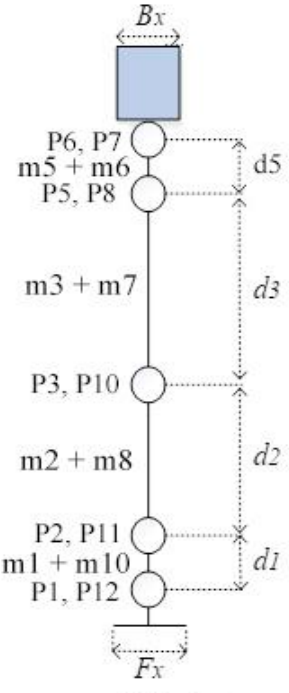
Here we can identify the position (x_{zmp},y_{zmp}) by using the result (2.6.2)

$$\begin{cases} x_{zmp} = x_{com} + \dfrac{\sum_i m_i P_{ix} \ddot{P}_{iz} - \sum_i m_i P_{iz} \ddot{P}_{ix}}{g \sum_i m_i} + \dfrac{\sum_i M_{iy}}{g \sum_i m_i} \\ y_{zmp} = y_{com} + \dfrac{\sum_i m_i P_{iy} \ddot{P}_{iz} - \sum_i m_i P_{iz} \ddot{P}_{iy}}{g \sum_i m_i} + \dfrac{\sum_i M_{ix}}{g \sum_i m_i} \end{cases} \quad (2.6.2)$$

The parameters "m_i" are the masses used to calculate COM and ZMP. Note that "m1, m2, m3, …, mm" are not the mass of each servo motor. They are the mass at the center point between two motors [9]. Their values and installed locations are also distributed on model as alternatively both following picture and table:



Y,Z-Axix



X,Z-Axis

Parame-ters	**Values**
m1 = m10	55g
m2 = m8	65g
m3 = m7	90g
m5 = m6	55g
mm	400g

Based on mass distribution model of biped, we can have final formula to calculate (x_{zmp},y_{zmp}) (2.6.3)

$$\begin{cases} x_{zmp} = x_{com} + \dfrac{\sum_i m_i P_{ix} \ddot{P}_{iz} - \sum_i m_i P_{iz} \ddot{P}_{ix}}{g \sum_i m_i} \\ y_{zmp} = y_{com} + \dfrac{\sum_i m_i P_{iy} \ddot{P}_{iz} - \sum_i m_i P_{iz} \ddot{P}_{iy}}{g \sum_i m_i} \end{cases} \quad (2.6.3)$$

The code of COM and ZMP be shown in the APPENDIX 4 (p.67-70)

CHAPTER 3: GENETIC ALGORITHM and PARTICLE SWARM OPTIMIZATION (PSO)

3.1 THEORY CONCEPT

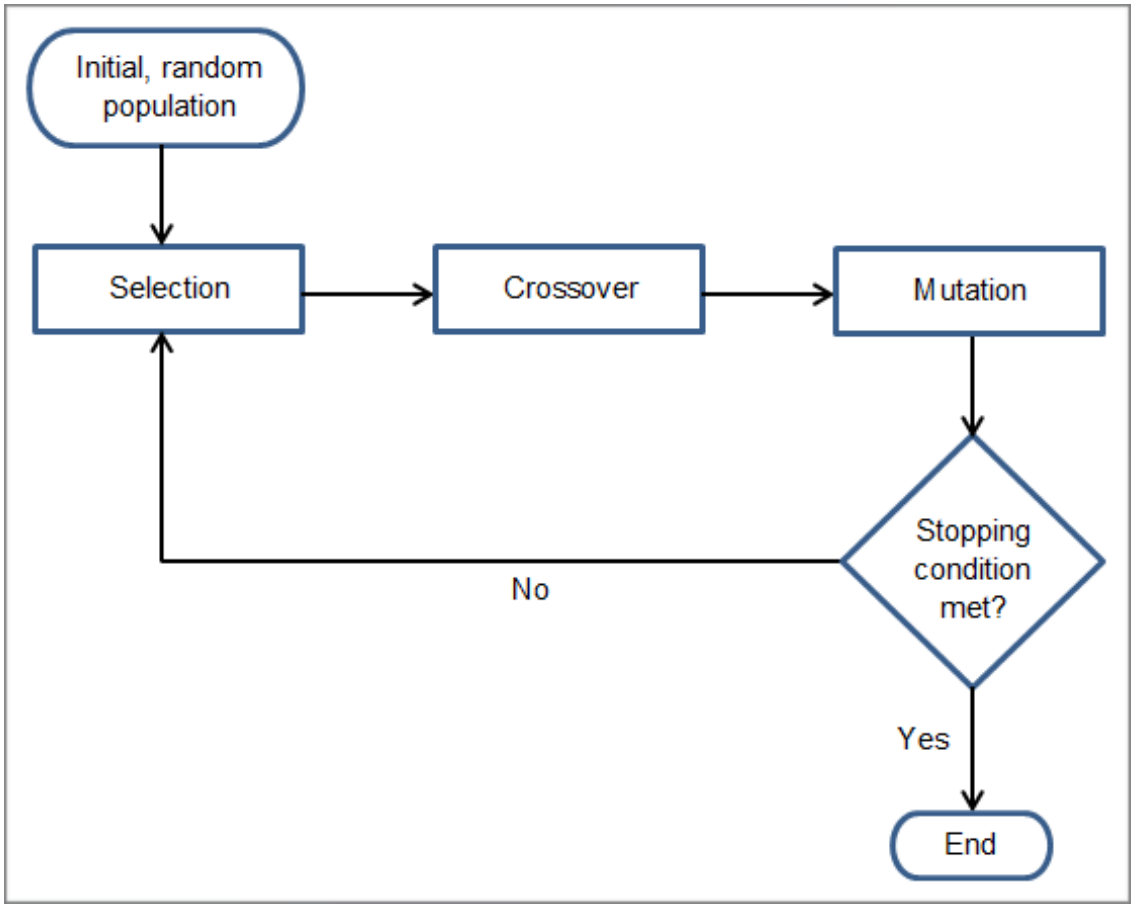
3.1.1. GENETIC ALGORITHM

3.1.1 Principle

The genetic algorithm based on natural evolution principle and over the years has become quite useful in order to optimize complex problem. In other words, it seem to be that the GA is probabilistic optimization tool, based on principle of natural evolution.

The genetic algorithm used to generate high-quality solutions to optimization and search problems by relying on bio-inspired operators such as mutation, crossover and selection [6].

The genetic algorithm consists of many step, be repeated over again until a set of solu-



tions has been found and the stopping criterion are satisfied. These steps are:

Create the initial population

Selection

Crossover

Mutation

Check fitness value

The figure above illustrates a simple GA working principle. The encoding and decoding operator will being used in coding to translate the created population in form of a matrix of decimal, binary, real parameters … value.

Natural Evolution	Genetic Algorithm
genotype	coded string
phenotype	uncoded point
chromosome	string
gene	string position
allele	value at a certain position
fitness	objective function value

In reality the terms of biological evolution such as genotype, phenotype, chromosome … will be understand by the algorithm as the coded string, un-coded point, string,… like in the figure above.

Compared with traditional methods, GA has the following advantages:

- The algorithm works with a set of coded parameters instead of the real parameters themselves. Each of the solutions has an accorded fitness value, which is how well it fits to the problem. These solutions can be throughout as being in a solution space, and after the solution found, it had be translated back into whatever parameter as desired.
- Genetic algorithm supports multi-objective optimization.

- Concept is not hard to comprehend, and once the implementation completed, it had be applied on a lot of problem, regardless of sector (technical, economics…) or complexity.

Creation of the initial population

The population is a set of solutions in the current generation, it plays the role of the pool from which the genetic material will be selected for selection, crossover or mutation operators.

Additionally, the population size must be chose carefully. If the size is small, the algorithm may lead to premature convergence before a solution had be found. In the contrary, if the population size is too big, it will take a very long time to find an optimal value (convergence) and may lead to resource waste.

In coding, the population can be represented as a two dimensional array (matrix) of population size and chromosome size or string length.

Selection

During each successive generation, a portion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected.

Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as the former process may be very time-consuming.

The genetic algorithm represents each solution in form of strings of number, treated them as a chromosome. These chromosomes will undergo the selection process based on their fitness value. Then the solutions with suitable fitness value selected to perform crossover and mutation.

There are several selection techniques, and the most popular ones are Roulette Wheel (Proportional), Tournament, Universal sampling selection.

The selection operation often performs following tasks:

- Each individual assessed by the fitness function. Then normalized by dividing that value by the sum of fitness value.
- Then the individual will be sorted according to their fitness value
- The accumulated value calculated by summing the fitness value of the last individual with all the previous fitness value. Noted that the accumulated fitness value will eventually equal 1.
- A random value between 0 and 1 will be selected.
- The chosen individual will be the first one, which accumulated value equal 1.

Roulette wheel: this process will be repeated over and over again until suffice individuals. There will be a single pointer spun multiple time, the individual with better fitness value will have higher chance of selection.

Tournament selection: The mating pool will divide into subsets, and the best individual in that subset will proceed into further operators.

Truncation selection: The best portion will be chosen (30%, 50% ….)

Crossover:

Crossover plays an important role in the creation of the new solutions, which expected to have better fitness value than the parent generation. Next, the operator will be applied to the strings of the mating pool.

There are many crossover methods used widely in GA optimization. However, the main concept is two parent strings are picked from the mating pool, and then a portion of the strings will be swapped with each other. Thus, the new children solution is found.

The single point crossover is one of the simple technique in GA optimization. It chooses a crossing site in the string of the parent and swaps all the bits in the right side of the crossing site.

Or in the N-point crossover, N points are chosen randomly. Two children will be created by combining these parts at the crossing point. For example:

Parent 1:	1 0 | 1 0 | 1 0 0 | 1 0
Parent 2:	1 1 | 0 0 | 1 0 1 | 1 0
Offspring 1:	1 0 | 0 0 | 1 0 0 | 1 0
Offspring 2:	1 1 | 1 0 | 1 0 1 | 1 0

Parent 1:	1 0 1 0 | 1 0 0 1 0
Parent 2:	1 0 1 1 | 1 0 1 1 0
Offspring 1:	1 0 1 0 | 1 0 1 1 0
Offspring 2:	1 0 1 1 | 1 0 0 1 0

Uniform crossover is also one of the simple crossover methods. It swapping bits in the parent string with a uniform rate. The rate is chosen from 0 to 1.

Here is an example of two-point crossovers MATLAB code, which has the following function:

1. Check the individual, if the individual is already a elitism or has best chromosome, the child will be the same as the parent.
2. Check the crossover probability, if a random number is less than the Pc, the crossover operator will take place.
3. Choose randomly an individual, which is different from the parent.
4. Initialize the crossover point 1 and 2, and make sure that the point 1 < point 2 by switching the position.
5. The children will inherit the characteristics of the parent.

Mutation

Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next. It is analogous to biological mutation. Mutation alters one or more gene values in a chromosome from its initial state. In mutation, the solution may change entirely from the previous solution. Hence, GA can come to a better solution by using mutation.

Mutation occurs during evolution according to a user-definable mutation probability. This probability should be set low. If it is set too high, the search will turn into a primitive random search.

The children generation created through the natural selection and crossover will have the "good" characteristics of the parent. However, sometime the initial population has not enough diversity, the individuals are not placed thorough the search space, thus the optimal solution may be impossible to found.

The main points of the mutation operator are diversity and exploration. For that reason, the mutation probability is often small and problem specific, in the MATLAB toolbox the probability has only a 0.01 default rate of mutation. When the GA converge to a local optimum, if the mutation rate is too big, that will prevent the convergence too happen. It is seem that the exploration of individual will never reach a local optimum. The suggested mutation rate is that the string length should be inverse proportional with the mutation probability. Another suggestion is that, to conserve the diversity in the population, in the beginning the probability should be set high to ensure that GA have a big search space, and then decrease it so the solution can be reached in the local optimum.

Mutation operator alter certain portions in the string of a chromosome. The mutation point will be chosen randomly and the bit in that position is replaced according to any value in the present gene. Similar to one point mutation, the number of bits would changed as desired.

The mutation operator functionality is described generally as follow:

1. The operator will take places only if the chromosome is not the best or elitism.

2. Randomly choose a mutation point and alter the bits.

Remarks

In a genetic algorithm, a population of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered; traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible.

The evolution usually starts from a population of randomly generated individuals, and is an iterative process, with the population in each iteration called a generation. In each generation, the fitness of every individual in the population being evaluated. the fitness is usually the value of the objective function in the optimization problem being solved. The more fit individuals are stochastically selected from the current population, and each individual's genome is modified (recombined and possibly randomly mutated) to form a new generation. Then, the new generation of candidate solutions is applied in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced or a satisfactory fitness level has been reached for the population.

This generational process repeated until a termination condition reached. Common terminating conditions are:

- A solution is found that satisfies minimum criteria
- Fixed number of generations reached
- Allocated budget (computation time/money) reached
- The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results
- Manual inspection
- Combinations of the above

3.1.2 Particle Swarm Optimization

Principle

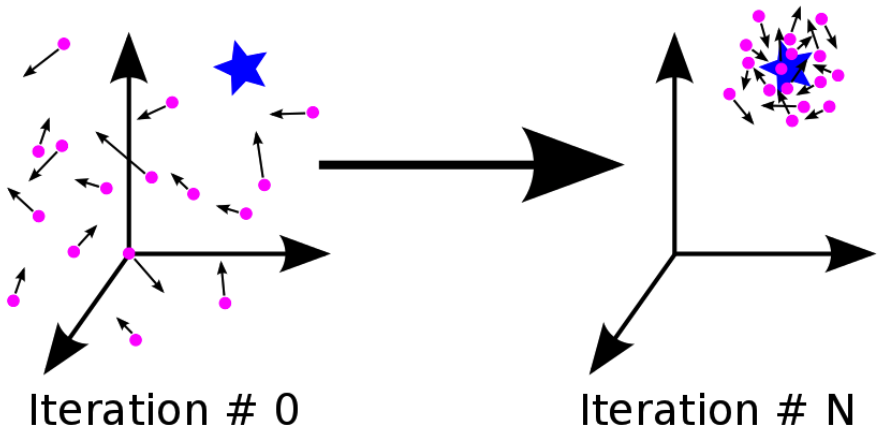
Particle swarm optimization has roots in two main component methodologies. Perhaps more obvious are its ties to artificial life (A-life) in general, and to bird flocking, fish schooling, and swarming theory in particular. It is also related. However, to evolutionary computation, and has ties to both genetic algorithms and evolutionary programming.

PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. The detailed information will be given in following sections.

Compared to GA, the advantages of PSO are that PSO is easy to implement and there are few parameters to adjust. The PSO is applied successfully in many areas: function optimization, artificial neural network training, fuzzy system control, and other areas where GA can be applied.

PSO simulates the behaviors of bird flocking. Suppose the following scenario: group of birds are randomly searching food in an area. There is only one piece of food in the area, be searched. The birds do not know where the food is, but they know how far the food is in each iteration. So what is the best strategy to find the food? The effective one is to follow the bird, which is nearest to the food.

PSO learned from the scenario and used it to solve the optimization problems. In PSO, each single solution is a "bird" in the search space. We call it "particle". All of particles have fitness values, evaluated by the fitness function, which will be optimized, and have the velocities, which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.



Initialization

PSO initialized with a group of random particles (solutions) and then searches for optima by updating generations. In every iteration, each particle updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. The fitness value is also stored, called “pbest”. Another "best" value is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called “gbest”. When a particle takes part of the population as its topological neighbors, the best value is a local best, named “ibest”.

After finding the two best values, the particle updates its velocity and positions with following equation (a) and (b):

- v[] = v[] + c1 * rand() * (pbest[] - present[]) + c2 * rand() * (gbest[] - present[]) (a)
- present[] = persent[] + v[] (b)

v[] is the **particle velocity**, persent[] is the **current particle (solution)**. pbest[] and gbest[] are defined as stated before. rand () is a random number between (0,1). c1, c2 are learning factors. usually c1 = c2 = 2.

The pseudo code of the procedure is as follows

For each particle

Initialize particle

END

Do

For each particle

Calculate fitness value

If the fitness value is better than the best fitness value (pBest) in history

set current value as the new pBest

End

Choose the particle with the best fitness value of all the particles as the gBest

For each particle

Calculate particle velocity according equation (a)

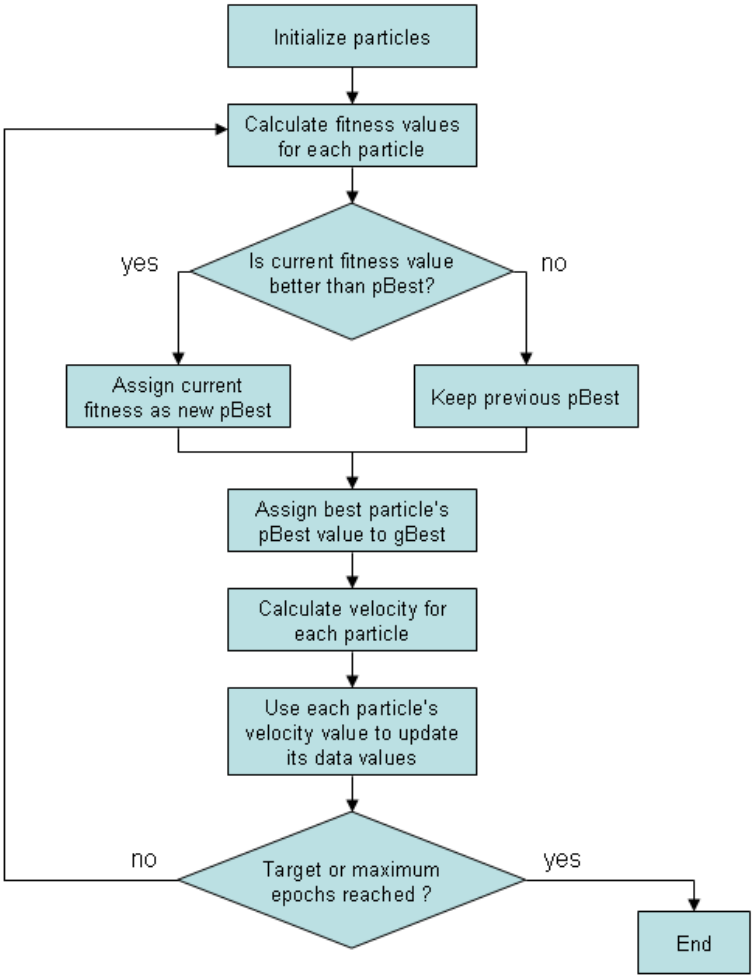
Update particle position according equation (b)

End

While maximum iterations or minimum error criteria is not attained

The particle's velocity on each dimension clamped to a maximum velocity Vmax. If the sum of accelerations would cause the velocity on that dimension to exceed Vmax, which is a parameter specified by the user. Then the velocity on that dimension is limited to Vmax.

Flowchart of the working process:



Parameters

There are not many parameters need to be tuned in PSO. Here is a list of the parameters and their typical values.

- The number of particles: the typical range is 20 - 40. Actually for most of the problems 10 particles is large enough to get good results. For some difficult or special problems, one can try 100 or 200 particles as well.
- Dimension of particles: It is determined by the problem to be optimized,

- Range of particles: they are identified by the problem, also be optimized, you can specify different ranges for different dimension of particles.
- Vmax: it determines the maximum change one particle can take during one iteration. Usually we set the range of the particle as the Vmax for example, the particle (x1, x2, x3)
- X1 belongs [-10, 10], then Vmax = 20
- Learning factors: c1 and c2 usually equal to "2". However, the other settings be used in different papers. But usually c1 equals to c2 and ranges from [0, 4]
- The stop condition: the maximum number of iterations the PSO execute and the minimum error requirement. For example, for ANN training in previous section, we can set the minimum error requirement is one misclassified pattern. The maximum number of iterations is set to 2000. This stop condition depends on the problem to be optimal.
- Global version vs. local version: we introduced two versions of PSO: global and local version. The global version is faster. However, this might converge to local optimum for some problems. Local version is a little bit slower but not easy to be trapped into local optimum. One can use global version to get quick result and use local version to refine the search.
- Another factor is inertia weight, introduced by SHI and EBERHART. Inertia Weight plays a key role in the process of providing balance between exploration and exploitation process. The Inertia Weight determines the contribution rate of a particle's previous velocity to its velocity at the current time step.

3.2 Practical Simulation and Experiment

Robot gait optimization

Compared to wheeled robots, legged robots usually possess superior mobility in uneven and unstructured environments. This is because they can use discrete footholds to over-

come obstacles, climb stairs, and so forth, instead of relying on a continuous support surface.

A gait is a cyclic, periodic motion of the joints of a legged robot, requiring the sequencing or coordination of the legs to obtain reliable locomotion. In other words, gait is the temporal and spatial relationship between all the moving parts of a legged robot

Gait optimization is very important for legged robots, because it determines the optimal position, velocity and acceleration for each Degree of Freedom (DOF) at any moment in time, and the gait pattern will directly affect the robot's dynamic stabilization, harmony, energy dissipation and so on. Gait optimization determines a legged robot's quality of movement.

Gait generation is very important as it directly affects the quality of locomotion of legged robots. As this is an optimization problem with constraints, it readily lends itself to Evolutionary Computation methods and solutions.

Objective function

To build the objective function, first we must define the inputs and outputs of the function.

The inputs are 4 variables that defined the robot walking gait (S, H, h, n), and the output is the accumulated differences of the reference model and the optimized realistic COM.

In order to make the objective function feasible, all of the related physical parameters of the robot must be included. The trajectory of the feet depends heavily on the set of variables (Step-length : S Bending-height: h Maximum lifting-height: H - Maximum frontal-shift: n). These 4 variables will affect the robot's walking gait, therefore, when we change these variables, the output gaits will be different. Here the Genetic Algorithm and Particle Swarm Optimization work to create the optimized parameters.

The goal here is to minimize the accumulated difference between the user-defined COM and the realistic COM.

The objective function works with following procedures:

1. Define the input and output: j = ofungenetic(x) in which j is the output function that we want to minimize, and x is the set of variables.

2. The related function to the main program be named respectively and the output of each process will become the input of the next. This procedure concluded by the ZMP-COM function.

3. After the COM was calculated, the difference between the Referenced COM and the Realistic COM is ready to be the input of the final objective function, which is the sum of square exponential of the differences between two models.

The objective function will be show in APPENDIX 5.

3.2.1 Practical Result And Algorithm Comparison

Since the two approaches are supposed to find a solution to a given objective function but employ different strategies and computational effort, it is appropriate to compare their performance.

The 2 algorithms will be apply with the same certain parameters to make the comparison as fair as possible.

The population size, dimension, iteration are the same for both algorithm, which are 100, 4, 1000 respectively.

The comparison takes place after each algorithm have run 10 times, and a plot of convergence statistics are drawn afterwards.

The real parameter is depended on the physical designation of the robot, and the COM referenced model is modified to find out different gait of the biped.

Attempt 1 : Long Feet Distance, Low Feet Lifting

The results of GA and PSO shown in MATLAB:

	GA	PSO
Best variables (S,H,h,n)	[12.4121 0.7595 3.9998 3.0121]	[12.4200 1.0000 4.0000 3.0190]
Time (seconds)	103.194893	23.782331
Fitness Value	8.1241	8.1243

Referenced COM coordination, with x, y are the axis names and the number “1, 2, 3, 4” are four moving stages of the COM trajectory.

- x1=-2.2; x2=-1.5; x3=2.5; x4=3.3;
- y1=-3.3; y2=-1.2; y3=-1.2; y4=-3.3;

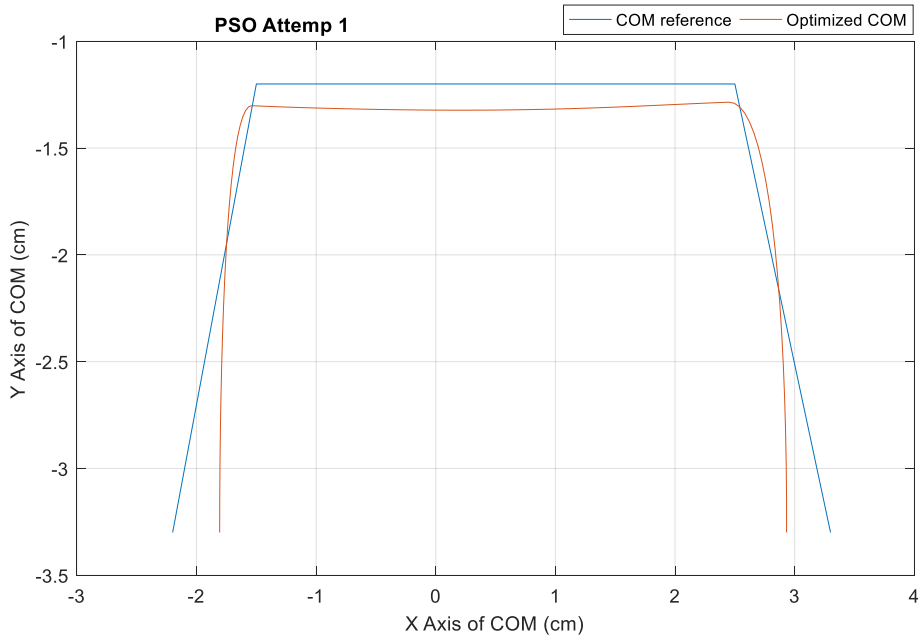


Figure 3.1: PSO optimization of COM

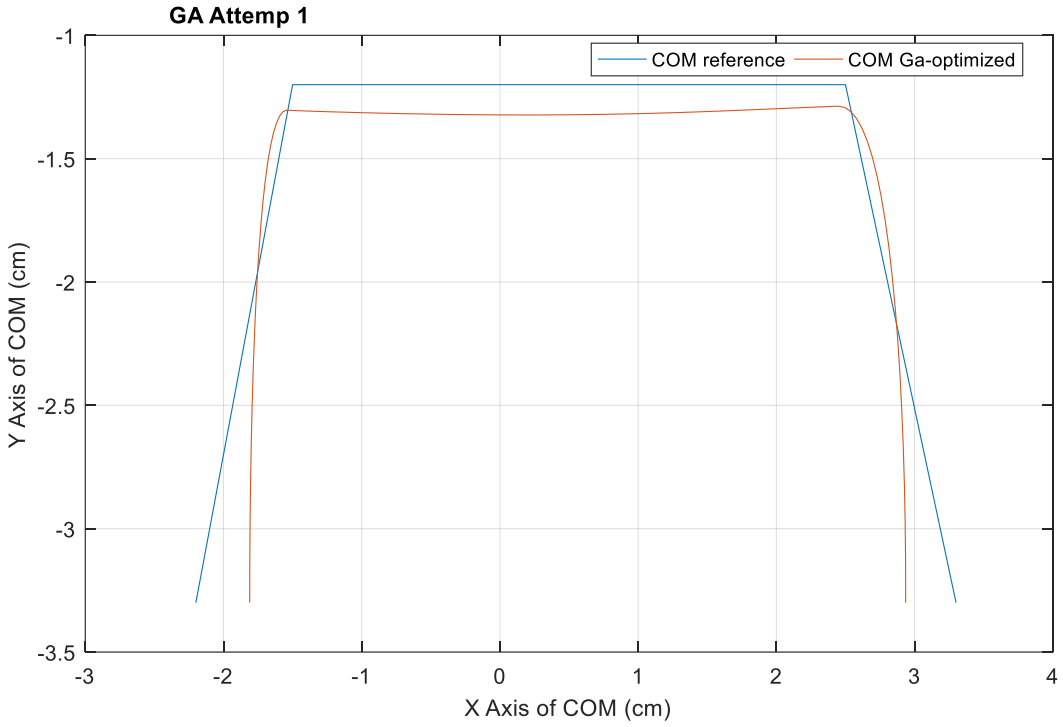


Figure 3.2: GA optimization of COM

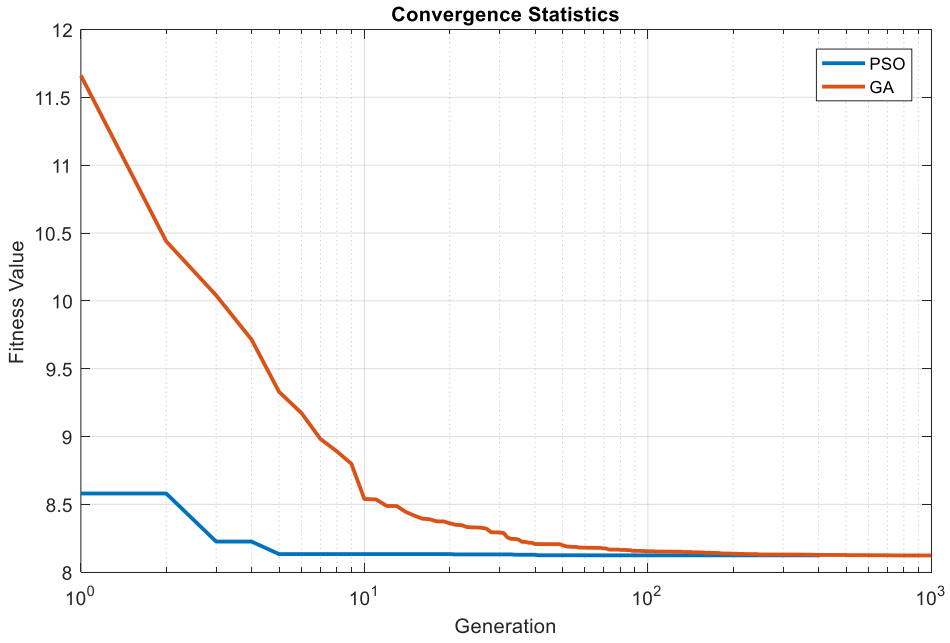


Figure 3.3: Convergence Statistics

- Figure 1.1 and 1.2 show the Optimized COM figure compare with the referenced Com created by the users. The closer the orange line to the blue line, the better the result.
- Figure 1.3 show the convergence characteristics of the 2 algorithms, by observation both have a convergence generation of ~ 100.

Attemp 2 : Waist Movement (n) Increased, Long Feet Distance, Low Feet Lifting

	GA	PSO
Best variables (S, H, h, n)	[12.4095 0.9229 3.9969 5.6988]	[12.3982 1.0000 4.0000 5.6995]
Time (seconds)	88.023012	26.278685
Fitness Value	18.5542	18.5538

Referenced COM coordination, with x, y are the axis names and the number “1, 2, 3, 4” are the four moving stages of the COM trajectory.

- x1=-2.2; x2=-1.5; x3=2.5; x4=3.3;
- y1=-3.3; y2=0.5; y3=0.5; y4=-3.3;

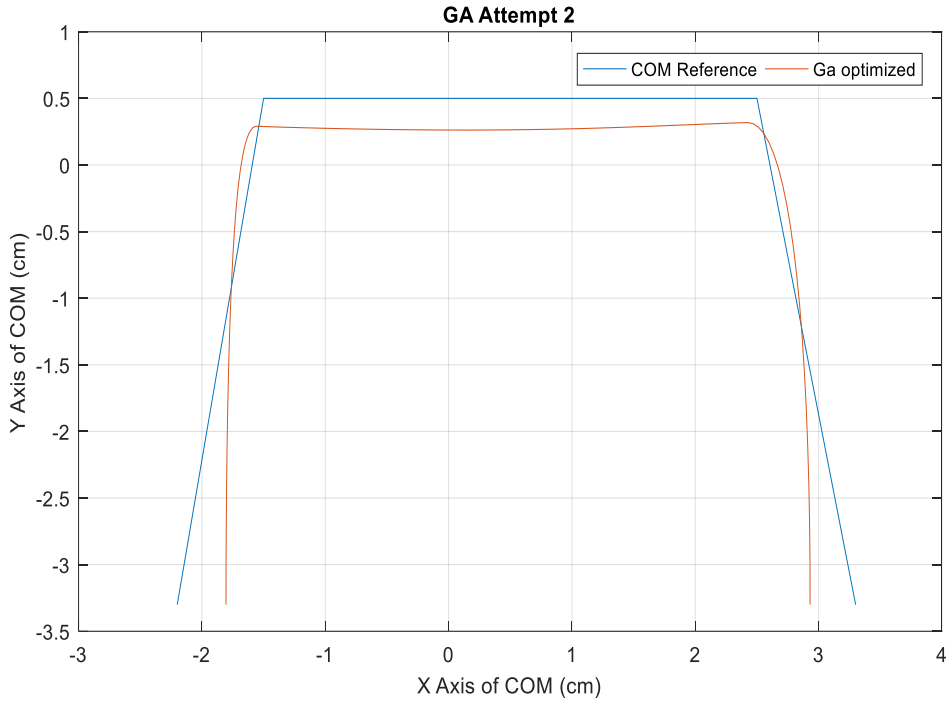


Figure 3.4: GA optimization of COM

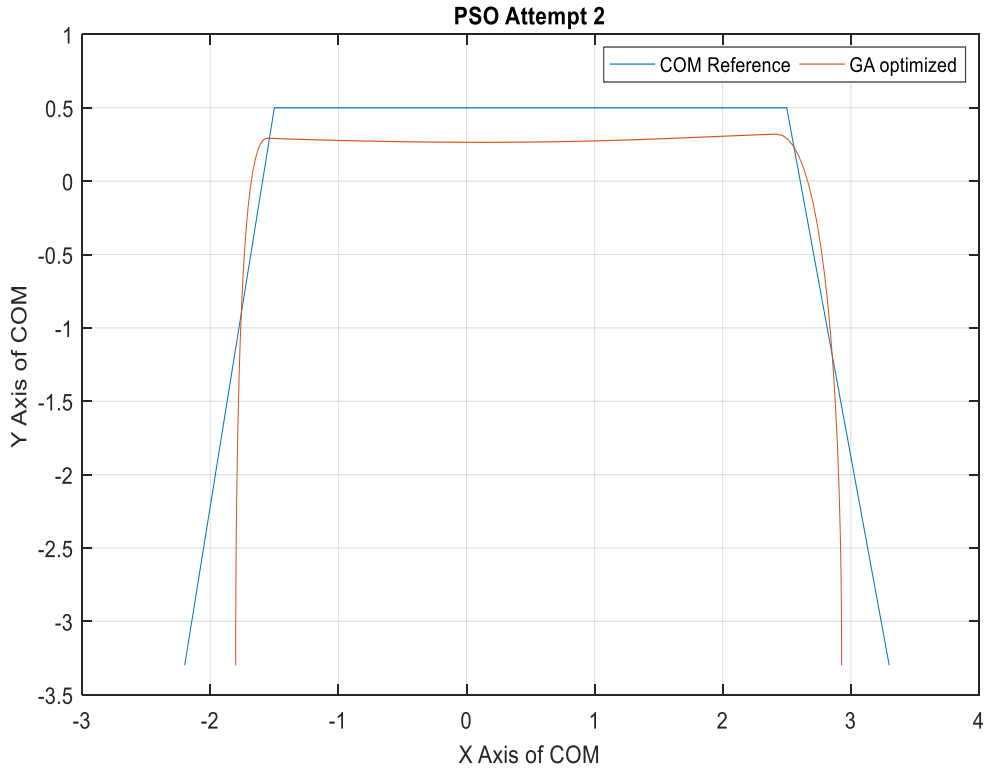


Figure 3.5: PSO optimization of COM

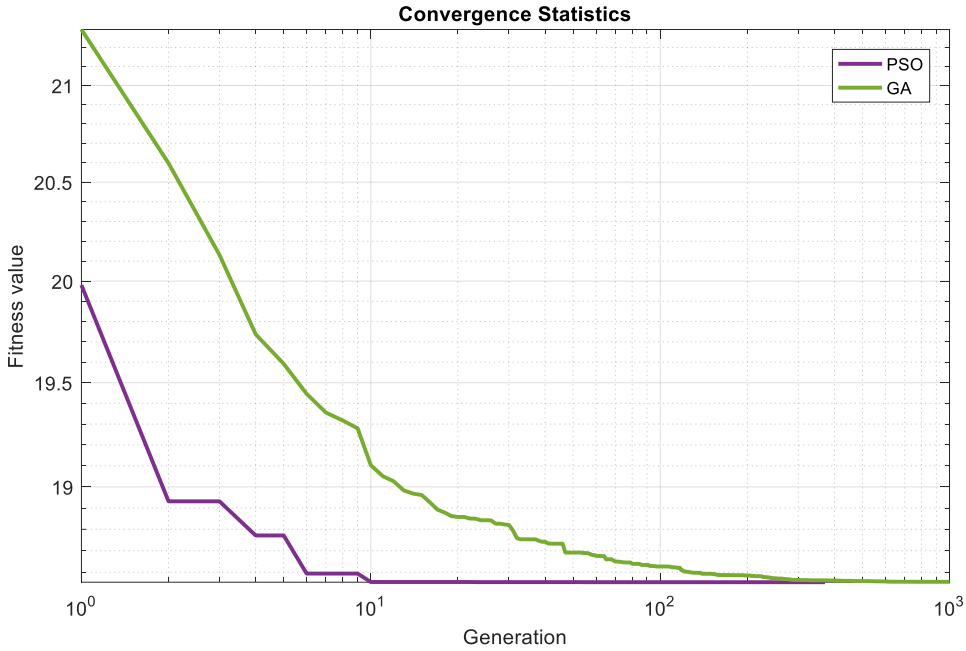


Figure 3.6: Convergence Statistics

Attempt 3 : Short walking distance, low feet lifting high waist movement

	GA	PSO
Best variables (S, H ,h, n)	[8.0896 1.0000 3.9570 5.6866]	[8.0466 1.0000 4.0000 5.7044]
Time (seconds)	83.074813	22.995260
Fitness Value	14.6899	14.7479

Referenced COM coordination, with x, y are the axis names and the number “1, 2, 3, 4” are the four moving stages of the COM trajectory.

- x1=-1.2; x2=-0.8; x3=1.8; x4=2.3;
- y1=-3.3; y2=0.5; y3=0.5; y4=-3.3;

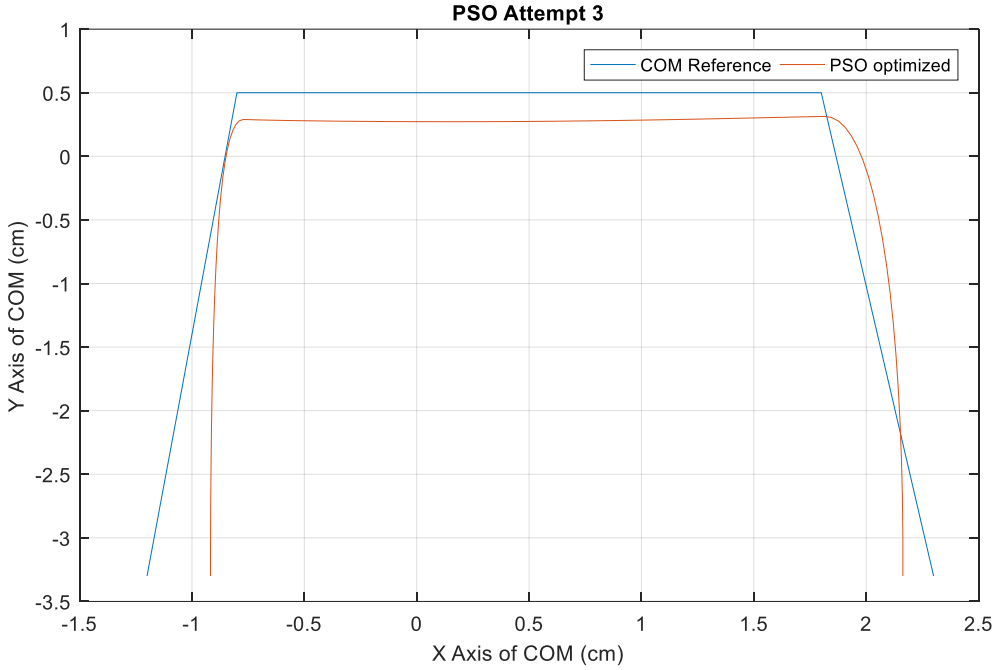


Figure 3.7: GA optimization of COM

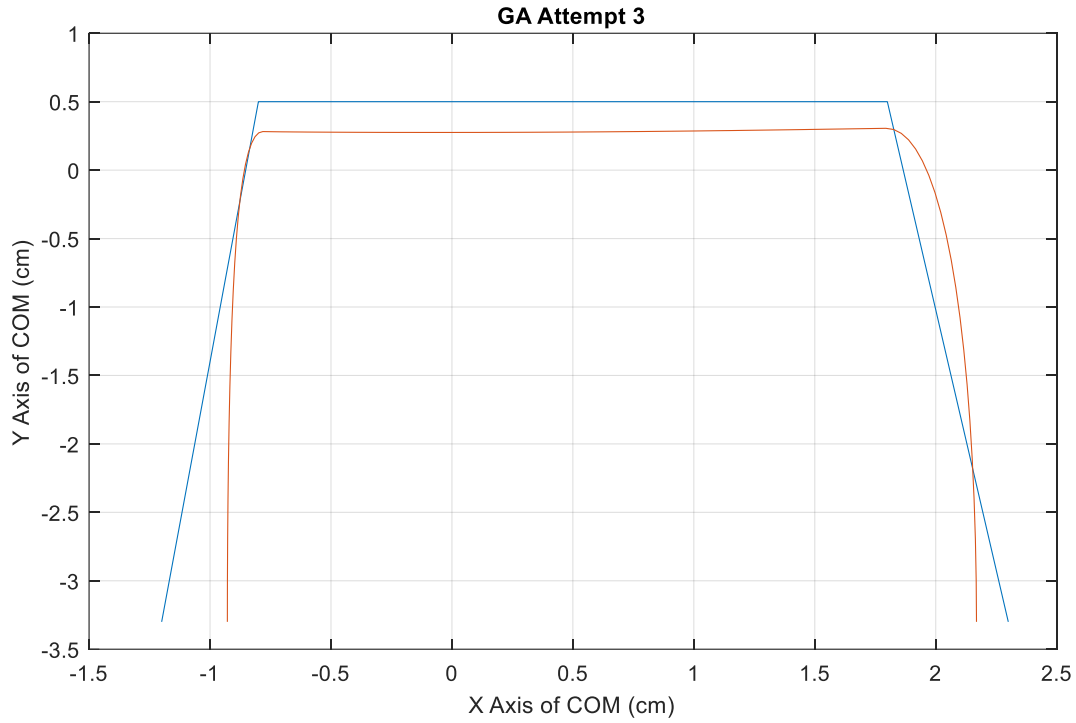


Figure 3.8: GA optimization of COM

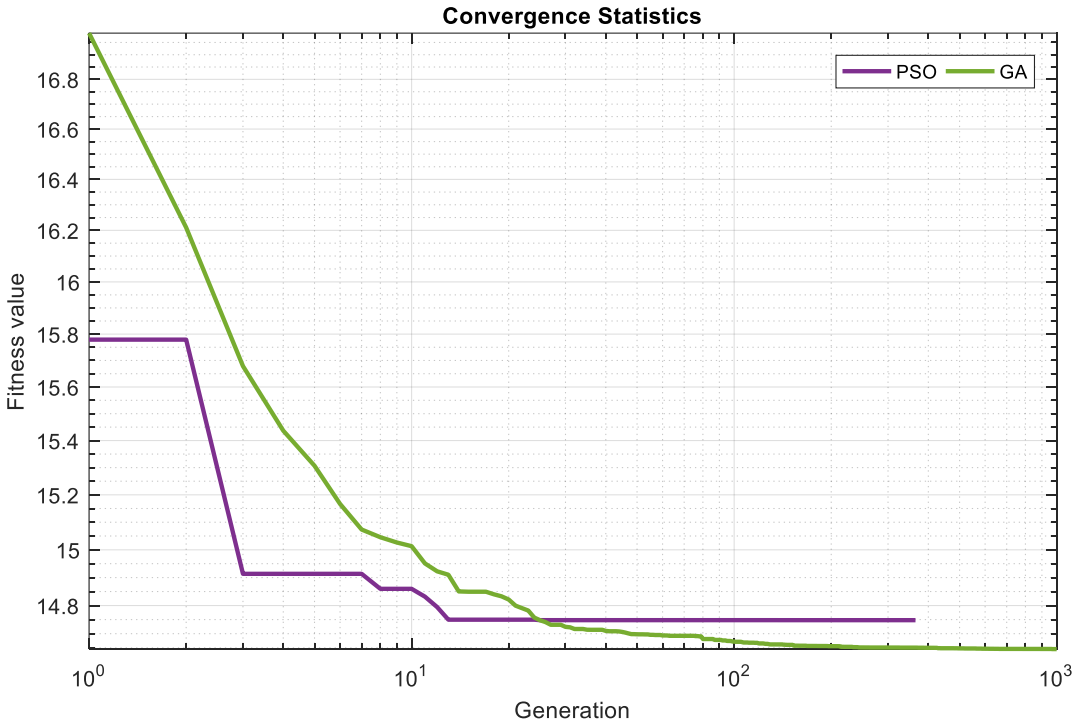


Figure 3.9: Convergence Statistics

Attempt 4 : Average Walking Distance – High Feet Lifting – Average Kneeling and Waist Movement

	GA	PSO
Best variables (S, H, h, n)	[12.4290 3.9566 3.9963 3.0167]	[12.2242 3.9857 3.3325 3.0230]
Time (seconds)	90.357987	48.121739
Fitness Value	8.1253	8.1255

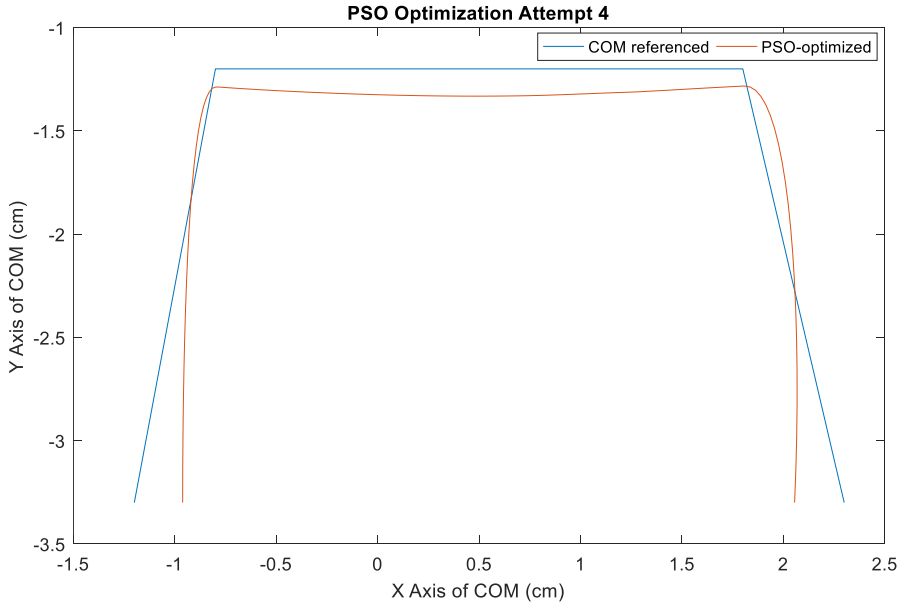


Figure 3.10: GA optimization of COM

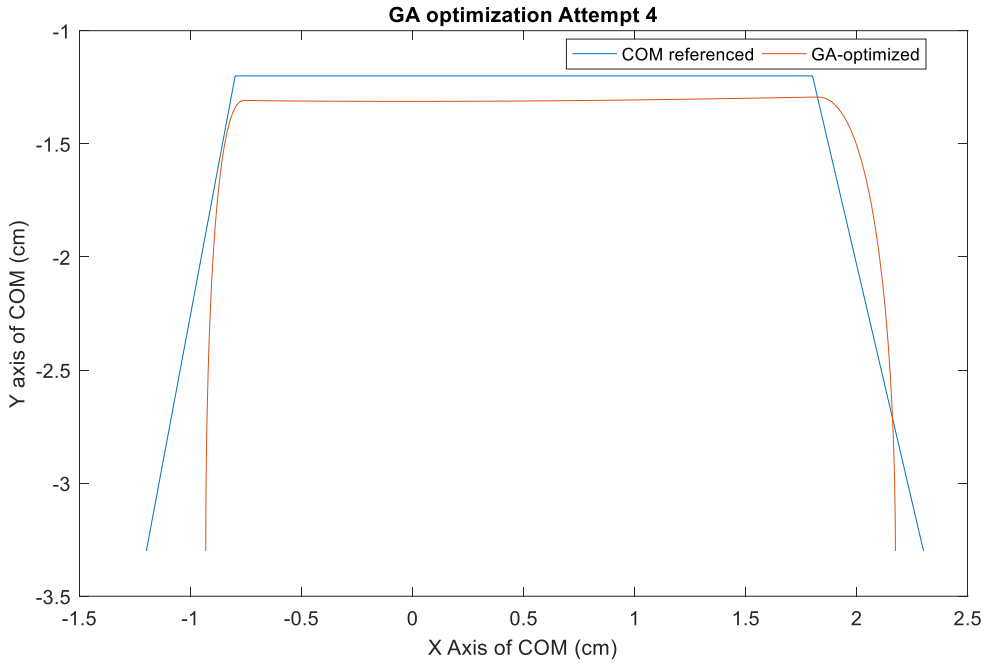


Figure 3.11: GA optimization of COM

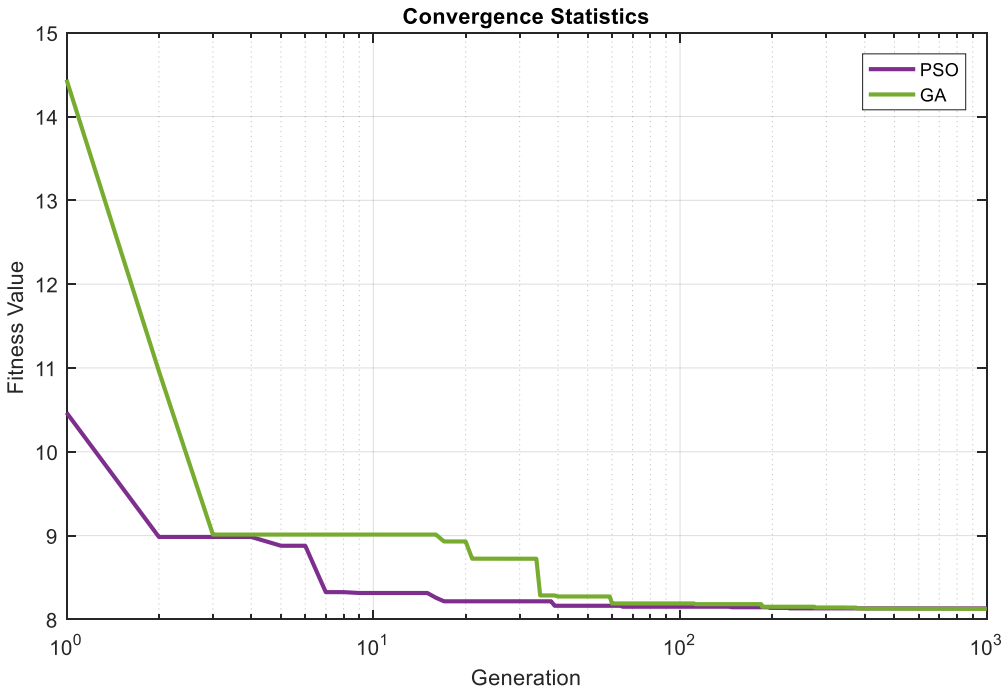


Figure 3.12: Convergence Statistics

Remarks

PSO is similar to the Genetic Algorithm (GA) that they are both population-based search approaches and that they both depend on information sharing among their population members to enhance their search processes using a combination probabilistic rules. PSO is more computationally efficient (uses less number of function evaluations) than the GA.

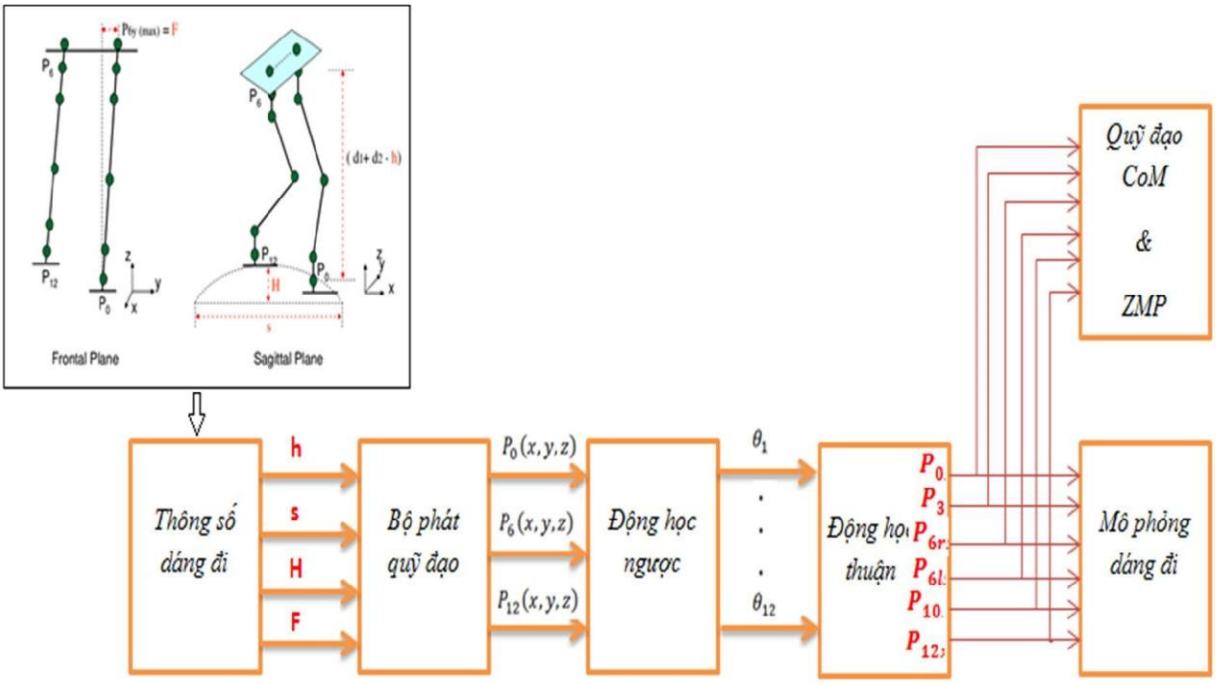
Conversely, the GA is a well established algorithm with many versions and many applications.

Observed from the result, when we change the referenced COM (the trapezoid), the 2 algorithms will output different value of parameters. One another way is to modify the constraint that bound the four parameters S, H, h, n to create the desired gait. However, the gait must be within the physical operational range of the biped.

Some of the observation noticed from the result:

- The waist movement of the robot will increase if we increase the height of the trapezoid (like in the Attempt 1 and 2).
- The walking distance initial on the width of the trapezoid (x1, x4 coordination of the trajectory).
- The feet lifting parameters is hard to experiment, it needs to both reducing the width and increase the height of the trapezoid a little.
- All of the results can obtain another way by modifying the constraint of the parameters in the code.

CHAPTER 4: MATHEMATICAL MODEL SIMULATION



Input and output parameters of the simulation program

The output of the model:

- Biped gait parameters (S, H, h, n) were optimized by using genetic algorithms (GA) and Particle Swarm Optimization (PSO), corresponding to the physical structure of the "model actually small biped".
- Corresponding to the gait parameters have been optimized, the program continues to compute:
  - The trajectory of foot-swing and hip
  - 2D simulation of biped gait
  - ZMP and COM trajectory during the movements of the biped

The interface of the simulation program could be simply coding. This code is used to simulate the gait-waking of biped, also checking the stability of ZMP during moving process. It be presented in the appendix 4.

4.1. Trajectory of feet and hip MATLAB simulation

In one period of walking forward, trajectory of foot-swing is described by blue color, the supporting-foot is showed by black point and red is the trajectory of hip. All of them are presented in the figure (4.1.1):

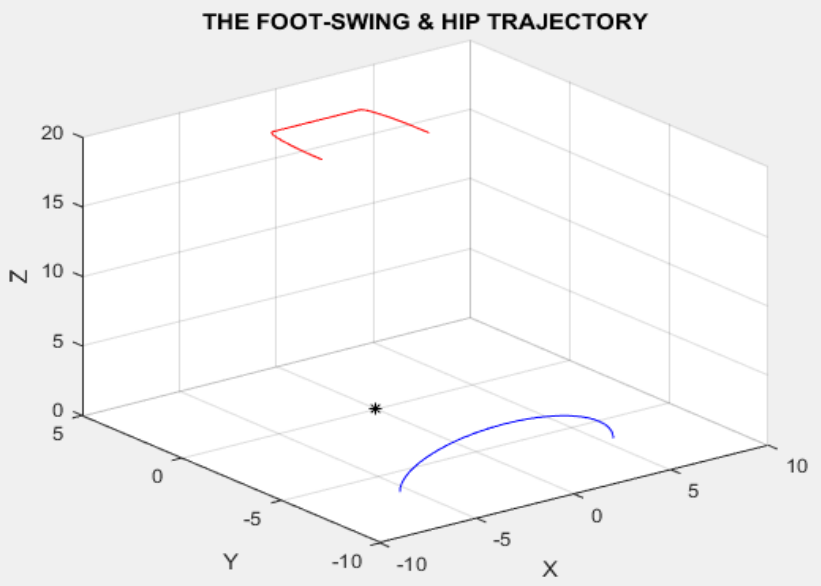


Fig.4.1.1: The trajectory of foot-swing and hip

4.2. Gait-walking MATLAB simulation

The gait-walking of biped robot during one period is simply modeled by connecting 12 links from P_1, P_2, P_3, …, P_{11}, P_{12} and under the control of 12 rotation angles. This can be painted by MATLAB tool and simulated as following figure (4.2.1). However, to make it moving one more period or over, we could create more trajectory for both foot-swing and hip. By an easy way, after complete one walking period the supporting-foot take a new responsibility as swing-foot, also otherwise the swing-foot changed role by supporting foot. More meaning, both of two points P_1 and P_{12} will be changed roles each

other after finishing one walking period. This process also suggests reversing 12 angle rotations corresponding with positions of joints.

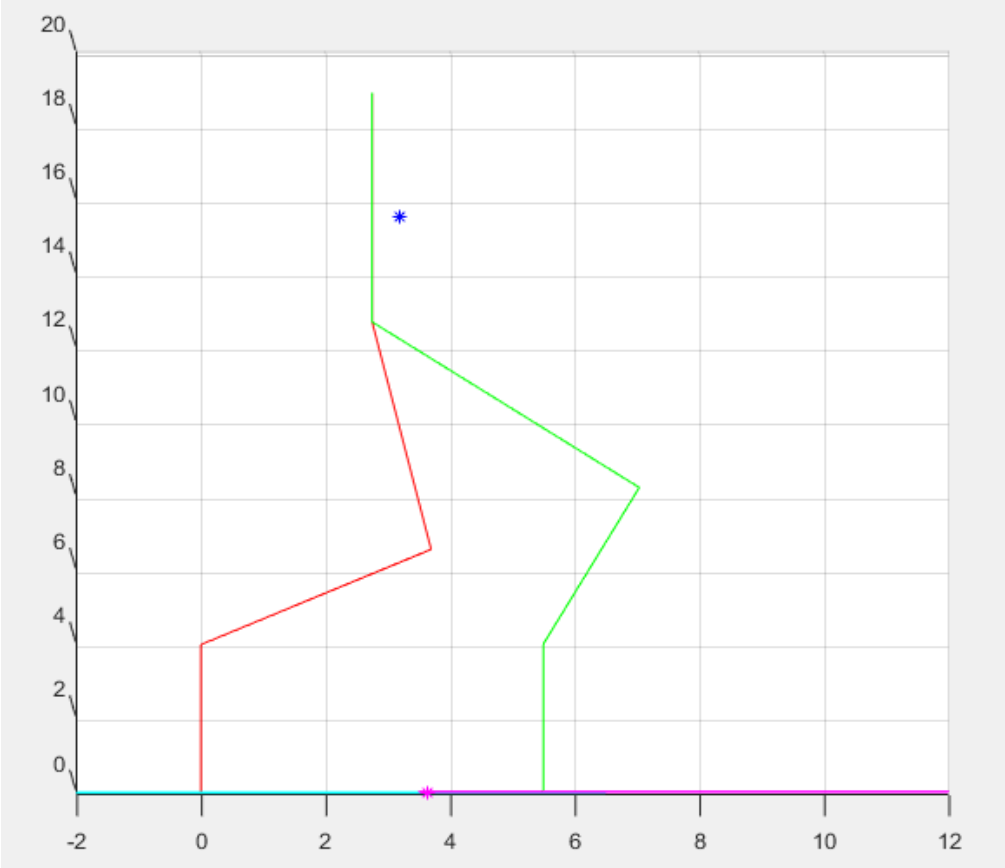


Fig.4.2.1: The 2D-space of biped gait walking

4.3. ZMP and COM trajectory MATLAB simulation

The figure (4.3.1) shows the result of ZMP during one period of moving forward. In order to get stable that biped not fall, the ZMP need to be inside the supporting-foot while the foot-swing was moving. The pink-point inside the supporting-foot is represented "ZMP" and blue-pink is symbolled COM.

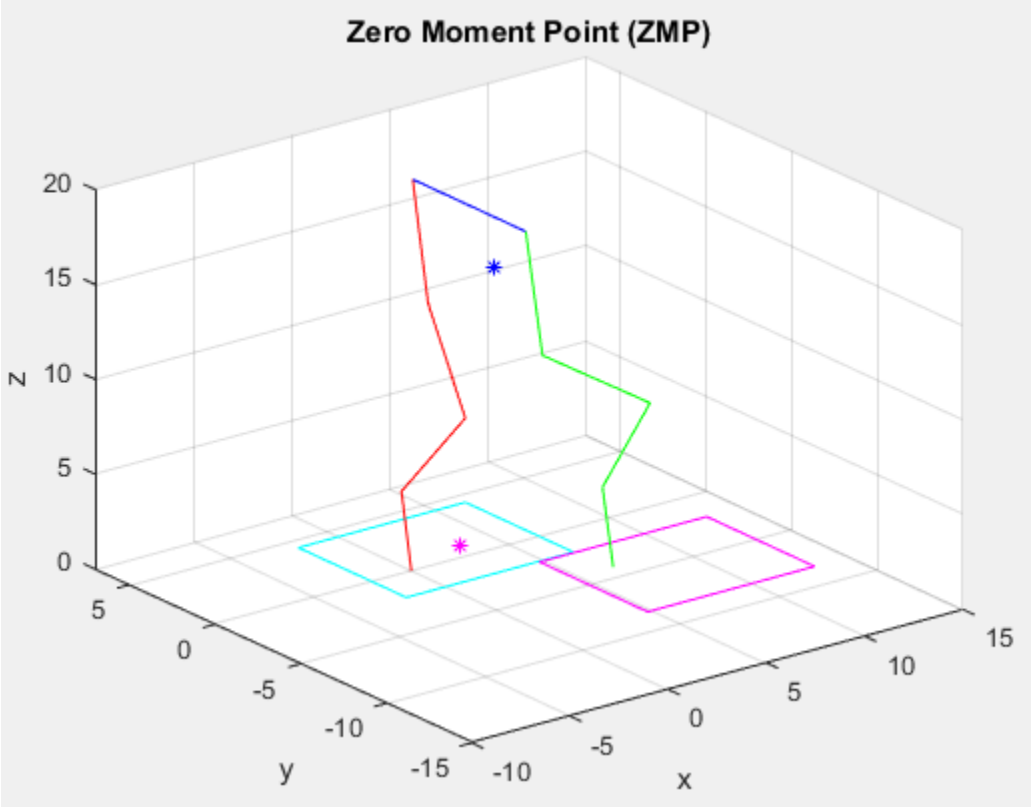


Fig.4.1.3: ZMP during one period of moving

CHAPTER 5: PRACTICAL MODEL OF BIPED

5.1. Model design

The Biped's parts are made from cutting CNC aluminum. Although the biped model bough from other robot designer company, we create the special point by editing the initial feet-size of biped. By the way, we change its feet by new one smaller size. After many times focusing to improve algorithm, we make sure that ZMP being always satisfied with stable condition that not fall while moving over again.

General drawing

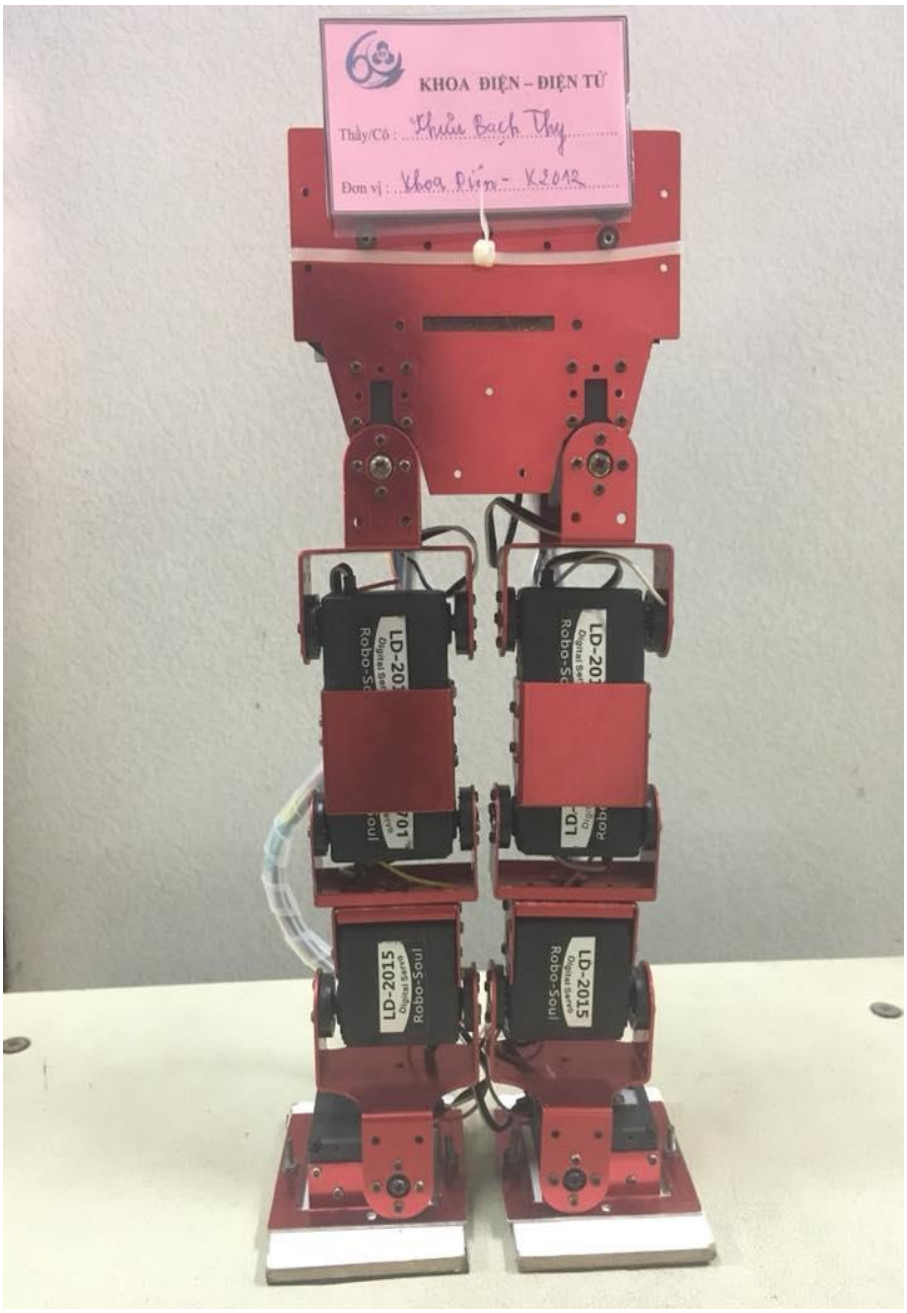


Fig.5.1.1: Front view of biped

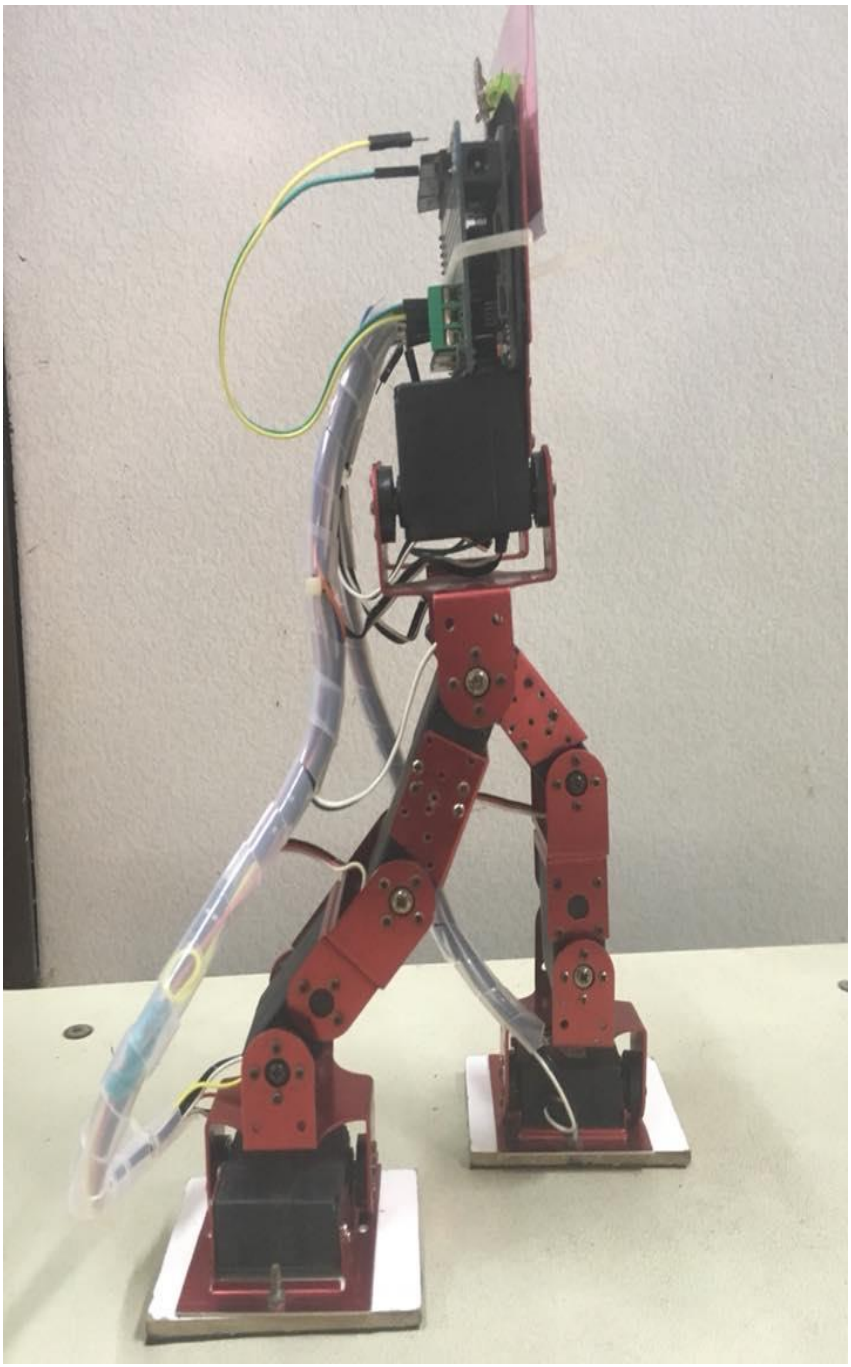


Fig.5.1.2: Side view of biped

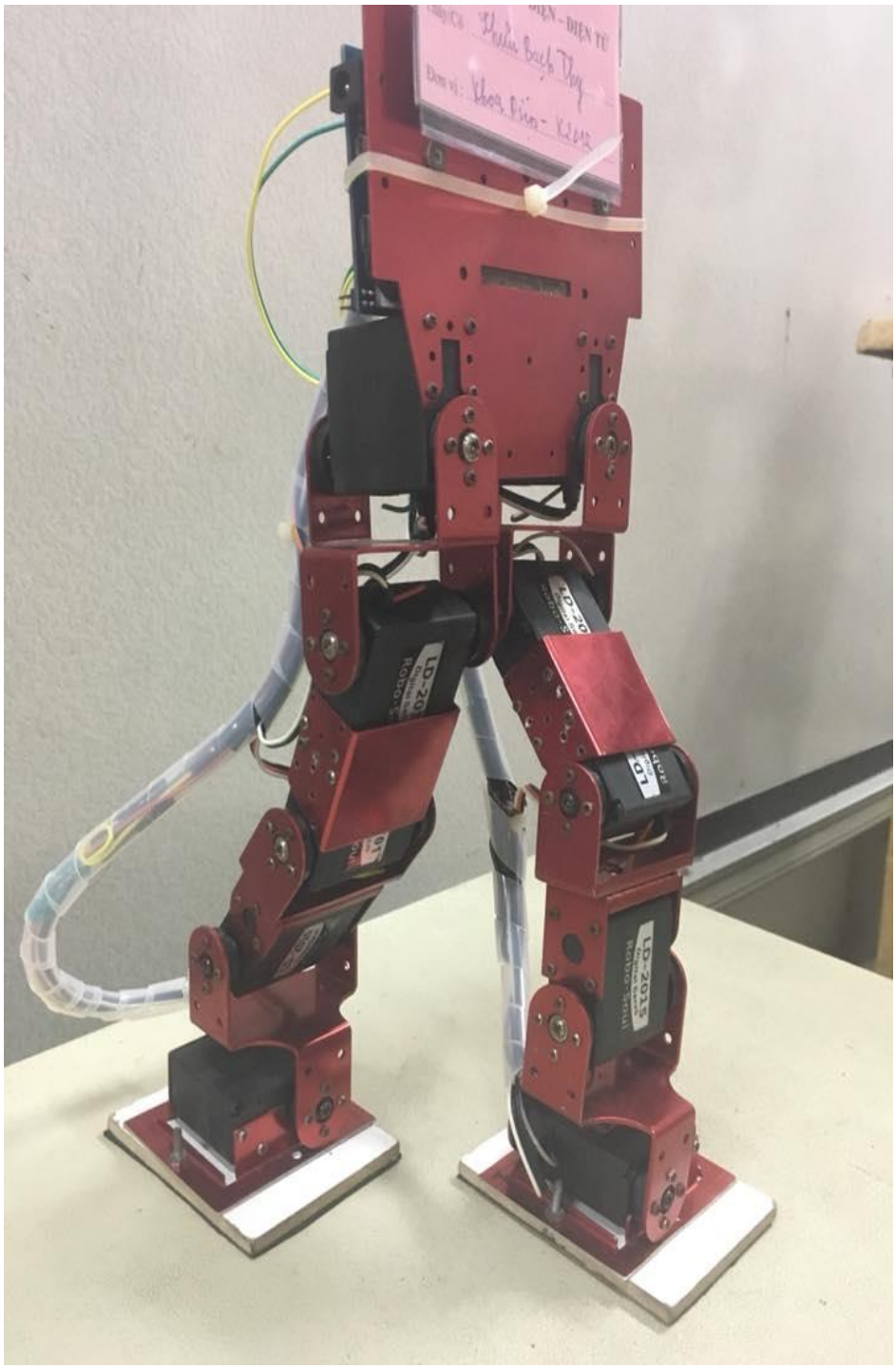
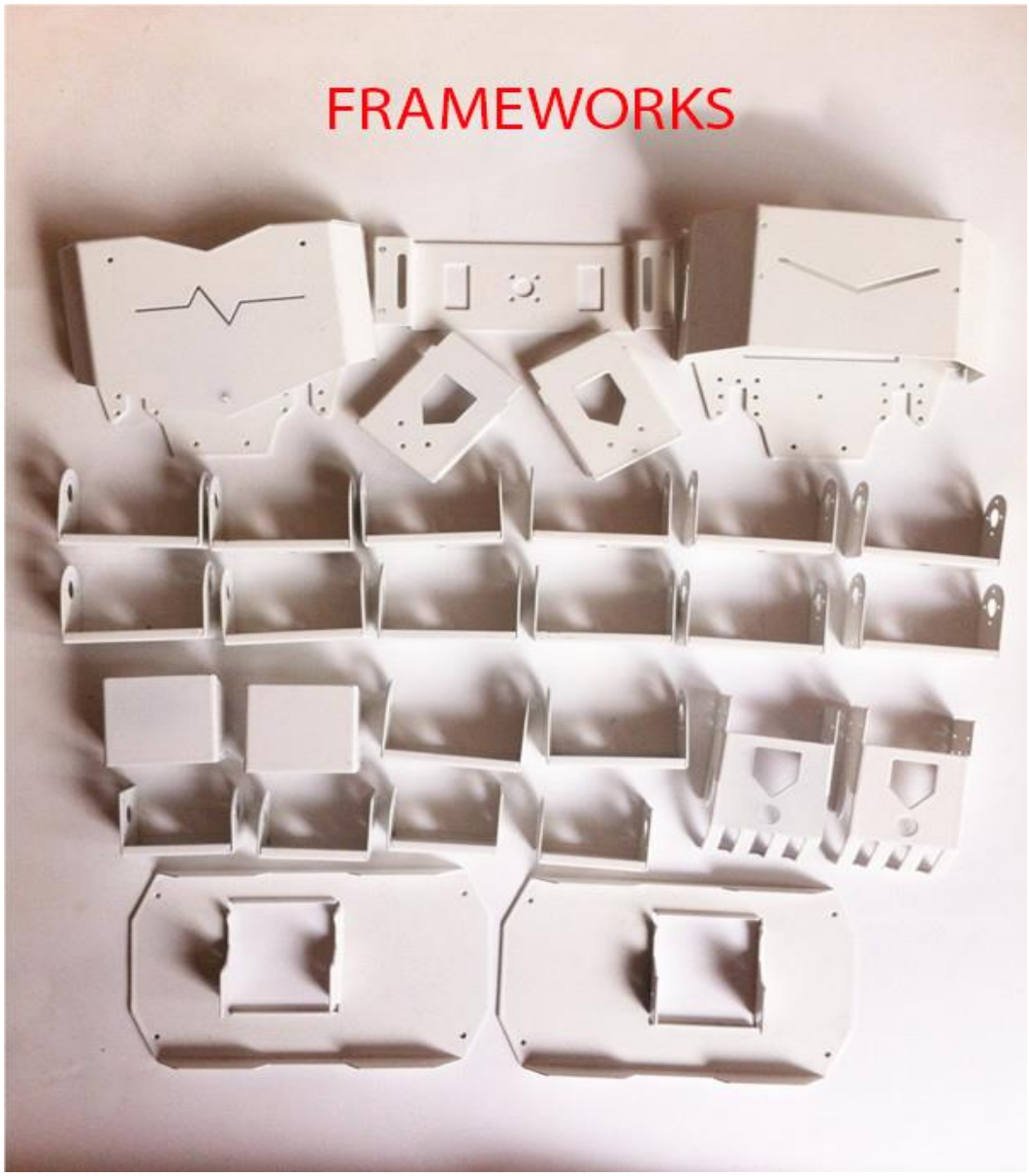


Fig.5.1.3: Full view of biped

Specific parts



5.2. Controller board and power source

5.2.1. Controller board

The board which used for controlling biped is Arduino Due

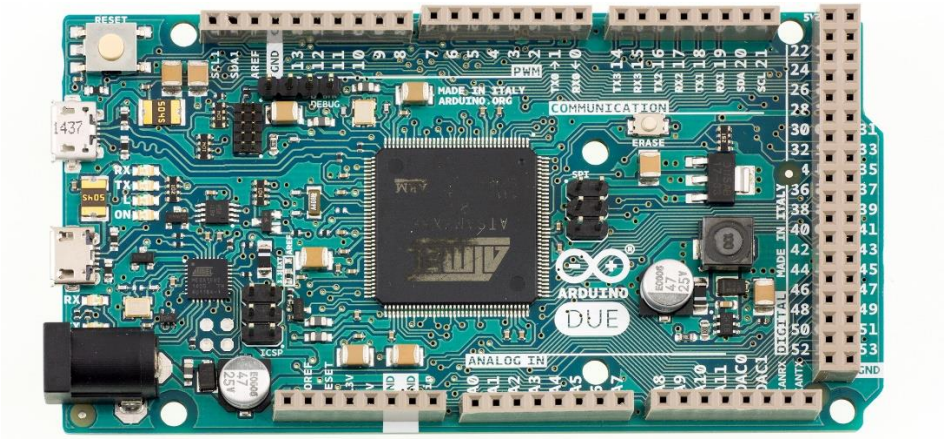


Fig.5.2.1: Arduino Due

The Arduino Due ADK is a microcontroller board based on the AT91SAM3X8E (datasheet). It has a USB host interface to connect with Android based phones, based on the MAX3421e IC. It has 54 digital input/output pins (of which 12 can be used as PWM outputs), 12 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.

The MEGA ADK, it features an ATmega16U2 programmed as a USB-to-serial converter. Revision 2 of the Mega ADK board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode.

Revision 3 of the board has the following some features:

- 1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF allows an attached shield with the proper configuration to adapt to the voltage provided by the board. This enables shield compatibility with a 3.3V board like the Due and AVR-based boards which operate at 5V. An unconnected pin, reserved for future use.
- Stronger RESET circuit

We also use an Arduino mega proto shield V3 to connect controller board and servo

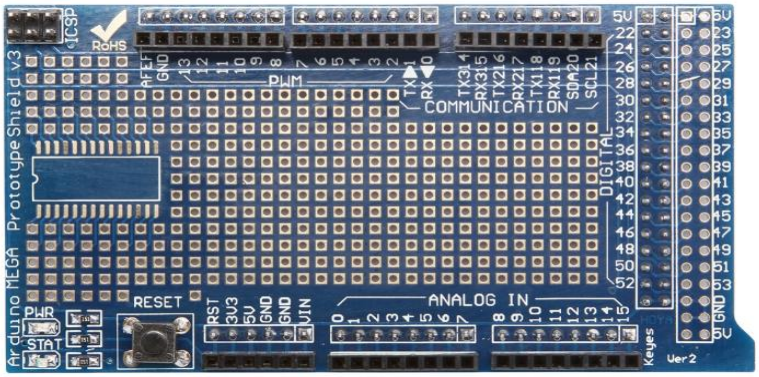


Fig.5.2.2: Arduino mega proto shield V3

Schematic, Reference Design & Pin Mapping

As shown in the figure Summary

Microcontroller	AT91SAM3X8E
Operating Voltage	3.3V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-16V
Digital I/O Pins	54 (of which 12 provide PWM output)
Analog Input Pins	12
Analog Output Pins	2 (DAC)
Total DC Output Current on all I/O lines	130 mA
DC Current for 3.3V Pin	800 mA
DC Current for 5V Pin	800 mA
Flash Memory	512 KB all available for the user applications
SRAM	96 KB (two banks: 64KB and 32KB)
Clock Speed	84 MHz
Length	101.52 mm
Width	53.3 mm
Weight	36 g

The Arduino MEGA ADK can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the GND and Vin pin headers of the POWER connector. The board can operate on an external supply of 5.5 to 16 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- VIN - The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

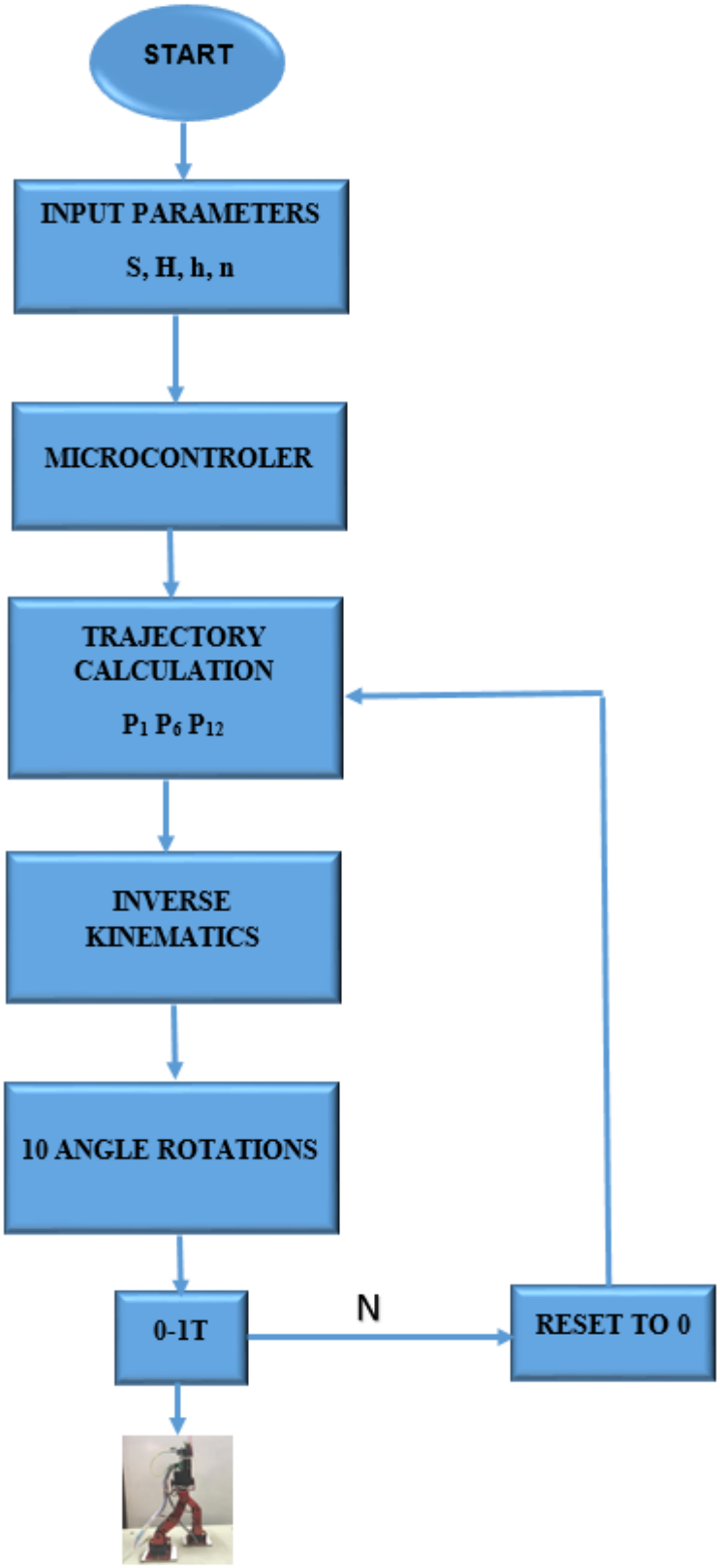
- 5V - This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.

- 3V3 - A 3.3 volts supply generated by the on-board regulator. Maximum current draw is 50 mA.

- GND - Ground pins.

- IOREF - This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

The controlling loop:


START
INPUT PARAMETERS
S, H, h, n
MICROCONTROLER
TRAJECTORY CALCULATION
P1 P6 P12
INVERSE KINEMATICS
10 ANGLE ROTATIONS
0-1T
N
RESET TO 0

5.2.2. Power source

We use a 5V 20A AC-DC power source to supply power for both controller board and servos. This power source will convert 220V AC to 5V DC, which will provide a stable power supply for controller board and servos

Fig.5.2.3: The 5V - 20A AC-DC power

5.3. RVC Servos

The big difference between this and other motors is the fact that it has a built in gear box and the controller inside so it comes as a ready package which is far more accurate. Another big difference is the fact that most servo motors can only turn within 180 (degrees) of motion instead of the 360 that most dc motors turn.

In order to control servo, control signal is sent form controller board. The shaft will rotate to the angle, which respect to control signal. If the shaft is at the correct angle, then the motor shuts off. If the circuit finds that the angle is not correct, it will turn the motor the correct direction until the angle is correct. The output shaft of the servo is capable of travelling somewhere around 180 degrees. Usually, its somewhere in the 210 degrees range, but it varies by manufacturer. A normal servo is used to control an angular motion of between 0 and 180 (degrees). The control wire is used to communicate the

angle. The angle is determined by the duration of a pulse that is applied to the control wire. This is called Pulse Coded Modulation. The servo expects to see a pulse every 20 milliseconds (02 seconds). The length of the pulse will determine how far the motor turns. A 1.5 millisecond pulse, for example, will make the motor turn to the 90 (degrees) position (often called the neutral position). If the pulse is shorter than 1.5 ms, then the motor will turn the shaft to closer to 0 (degree). If the pulse is longer than 1.5 ms, the shaft turns closer to 180 degrees.

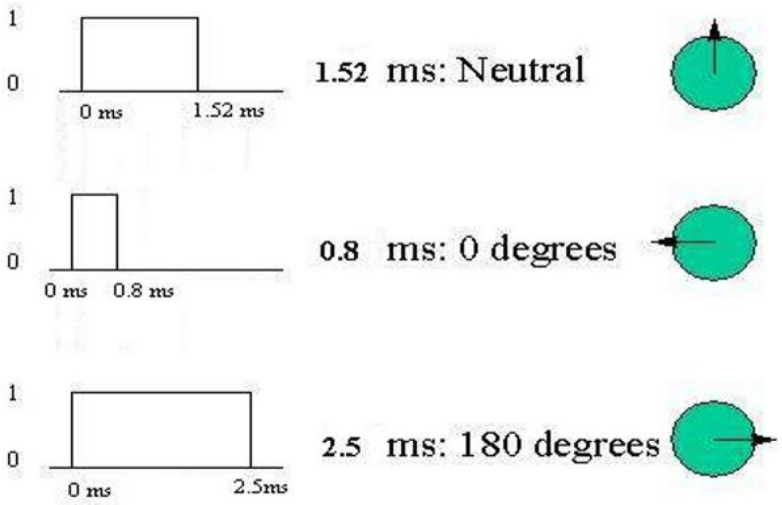


Fig.5.3: Pulse wide modulation

We install 10 Futaba servos to each joints of biped. Each servo will rotate to respective angle (θ_1, θ_2, θ_3, ..., θ_{12}) that we programed in controller board

CONCLUSION

Most of the parameters for the biped gait is looking from solving optimization problems using evolutionary algorithms on multi-objective Optimization Tool of MATLAB are run on the experimental model biped small size - 10 DOFs at 2 biped legs that did not fall. However, because of the biped controllers designed only open-loop, we simply verifiable sentiment is biped walking and falling. To be objective, we need to design closed-loop controllers to perform the calculation of the biped ZMP trajectory during walk with ZMP trajectory obtained from simulation programs gait ZMP trajectory as well as the original design oil.

"Building a model small size experimental biped - (open loop control)": Most of the parameters for the biped gait is looking from solving optimization problems using evolutionary algorithms on multi-objective Optimization Tool of MATLAB are run on the experimental model biped small size - 10 feet which biped DOF in 2 non-selfs. However, because of the biped controllers designed only open-loop, we simply verifiable sentiment is biped walking and falling. To be objective, we need to design closed-loop controllers to perform the calculation of the biped ZMP trajectory during walking compared with ZMP trajectory obtained from simulation programs gait and ZMP trajectory design original.

APPENDIX 1

```
function [P1,P6,P12] = gait_trajectory(S,H,h,n)

% Thong so robot
S=11;
H=2;
h=3;
n=5;

d1=4;
d2=4.5;
d3=6.2;
d5=6.2;
w=6.6;
%Thoi gian cua quy dao P1,P6,P12
t1=0:0.01:0.2;
t2=0.21:0.01:0.8;
t3=0.81:0.01:1;

t4=0:0.01:1;
t5=0:0.01:0.2;
t6=0.21:0.01:1;
t7=0:0.01:0.2;
t8=0.21:0.01:1;

%Toa do P1,P6,P12
P12x1=-ones(size(t5))*(S/2);
P12x2=(S/2)*sin((2*pi/1.58)*t6+3.87);

P12z1=ones(size(t7))*0;
P12z2=H*sin(((2*pi)/1.58)*t8-0.832921);

P6y1=n*sin(((2*pi)/0.8)*t1);
P6y2=ones(size(t2))*n;
P6y3=n*sin(((2*pi)/0.8)*t3+pi/2);

P1.x=zeros(101,1);
P1.y=zeros(101,1);
P1.z=zeros(101,1);

P6.x=transpose((S/4)*sin(-(2*pi/2)*t4-pi/2));
P6.y=transpose([P6y1 P6y2 P6y3]);
P6.z=transpose(repelem(d1+d2+d3+d5-h,101));

P12.x=transpose([P12x1 P12x2]);
P12.y=transpose(repelem(-w,101));
P12.z=transpose([P12z1 P12z2]);

end
```

APPENDIX 2

```
function [P1,P2,P3,P5,P6,P7,P8,P10,P11,P12] =
forward_kinamatics(deta1,deta2,deta3,deta5,deta7,deta8,deta10,deta11)

%Lengths between joints
% S=6;
% H=5;
% h=4;
% n=2;
d1=4;
d2=4.5;
d3=6.2;
d5=6.2;

d7=6.2;
d8=6.2;
d10=4.5;
d11=4;
w=6.6;
% T=1;
% t1=0:0.01:0.2;
% t2=0.21:0.01:0.8;
% t3=0.81:0.01:1;
%
% t4=0:0.01:1;
% t5=0:0.01:0.2;
% t6=0.21:0.01:1;
% t7=0:0.01:0.2;
% t8=0.21:0.01:1;

%Length
matrix_row = 101; %row matrix
matrix_line = 1;  %line matrix

%Position (x,y,z) P0,P1,P2,...,P13
P1.x = zeros(matrix_row,matrix_line);
P1.y = zeros(matrix_row,matrix_line);
P1.z = zeros(matrix_row,matrix_line);

P2.x = P1.x;
P2.z = d1.*cos(deta1);
P2.y = P2.z.*sin(deta1);

P3.x = d2.*sin(deta2);
P3.z = P2.z + d2.*cos(deta2).*cos(deta1);
P3.y = P3.z.*sin(deta1);

P5.x = P3.x + d3.*sin(deta2-deta3);
P5.z = P3.z + d3.*cos(deta2-deta3).*cos(deta1);
P5.y = P5.z.*sin(deta1);

P6.x = P5.x + d5.*sin(deta2-deta3+deta5);
P6.z = P5.z + d5.*cos(deta2-deta3+deta5).*cos(deta1);
P6.y = P6.z.*sin(deta1);

P7.x = P6.x;
```

```
P7.y = P6.y - w;
P7.z = P6.z;

P8.x = P7.x;
P8.z = P7.z - d7.*cos(deta7);
P8.y = P7.y - (P7.z-P8.z).*sin(deta7);

P10.x = P8.x + d8.*sin(deta8);
P10.z = P8.z - d8.*cos(deta8).*cos(deta7);
P10.y = P7.y - (P7.z-P10.z).*sin(deta7);

P11.x = P10.x + d10.*sin(deta8-deta10);
P11.z = P10.z - d10.*cos(deta8-deta10).*cos(deta7);
P11.y = P7.y - (P7.z-P11.z).*sin(deta7);

P12.x = P11.x + d11.*sin(deta8-deta10+deta11);
P12.z = P11.z - d11.*cos(deta8-deta10+deta11).*cos(deta7);
P12.y = P7.y - (P7.z-P12.z).*sin(deta7);

end
```

APPENDIX 3

```
function
[deta1,deta2,deta3,deta4,deta5,deta6,deta7,deta8,deta9,deta10,deta11,deta12]
= inverse_kinematics(P1,P6,P12)
d1=4;
d2=4.5;
d3=6.2;
d5=6.2;

l.x=P1.x-P6.x;
l.y=-P6.y;
l.z=P1.z-P6.z;

r.x=P12.x-P6.x;
r.y=-P6.y;
r.z=P12.z-P6.z;

deta1=atan(l.y./l.z);
deta6=deta1;
deta12=atan(r.y./r.z);
deta7=deta12;

l_l=sqrt(l.x.^2+0.25.*(l.y.^2)+(l.z+d1+d5).^2);
l_r=sqrt(r.x.^2+0.25.*(r.y.^2)+(r.z+d1+d5).^2);

detaA=acos((d2^2+d3^2-l_l.^2)/(2*d2*d3));
detaB=acos((d3.*sin(detaA))./l_l);
detaC=acos((d2^2+d3^2-l_r.^2)/(2*d2*d3));
detad=acos((d3.*sin(detaC))./l_r);

deta5=pi/2 - detaA + detaB + asin(l.x./l_l);
deta3=pi-detaA;
deta2=(deta3-deta5);

deta8=pi/2 - detaC + detad + asin(r.x./l_r);
deta10=pi-detaC;
deta11=deta10-deta8;

deta4=0;
deta9=0;

end
```

APPENDIX 4

```
function X_dd = acceleration(Xn)

Ts = 0.01;
[m,n] = size(Xn);     %101x1, m =101, n=1;

Xn1=[0;Xn(1:m-1,:)];
Xn2=[0;0;Xn(1:m-2,:)];

X_dd = (Xn-2*Xn1+Xn2)/Ts^2;
end
```

```
function com = com(P1,P2,P3,P5,P6,P7,P8,P10,P11,P12)
%m = [m1 m2 m3 m5 mm m7 m8 m10 m11]
M = [55 65 90 55 400 55 90 65 55];

PP1.x = (P1.x + P2.x)/2;
PP1.y = (P1.y + P2.y)/2;
PP1.z = (P1.z + P2.z)/2;

PP2.x = (P2.x + P3.x)/2;
PP2.y = (P2.y + P3.y)/2;
PP2.z = (P2.z + P3.z)/2;

PP3.x = (P3.x + P5.x)/2;
PP3.y = (P3.y + P5.y)/2;
PP3.z = (P3.z + P5.z)/2;

PP5.x = (P5.x + P6.x)/2;
PP5.y = (P5.y + P6.y)/2;
PP5.z = (P5.z + P6.z)/2;

PPM.x = (P6.x + P7.x)/2;
PPM.y = (P6.y + P7.y)/2;
PPM.z = (P6.z + P7.z)/2 + 6;

PP7.x = (P7.x + P8.x)/2;
PP7.y = (P7.y + P8.y)/2;
PP7.z = (P7.z + P8.z)/2;

PP8.x = (P8.x + P10.x)/2;
PP8.y = (P8.y + P10.y)/2;
PP8.z = (P8.z + P10.z)/2;

PP10.x = (P10.x + P11.x)/2;
PP10.y = (P10.y + P11.y)/2;
PP10.z = (P10.z + P11.z)/2;

PP11.x = (P11.x + P12.x)/2;
PP11.y = (P11.y + P12.y)/2;
PP11.z = (P11.z + P12.z)/2;

PPx = [PP1.x PP2.x PP3.x PP5.x PPM.x PP7.x PP8.x PP10.x PP11.x];
PPy = [PP1.y PP2.y PP3.y PP5.y PPM.y PP7.y PP8.y PP10.y PP11.y];
PPz = [PP1.z PP2.z PP3.z PP5.z PPM.z PP7.z PP8.z PP10.z PP11.z];

com.x = PPx*M'/sum(M);
com.y = PPy*M'/sum(M);
com.z = PPz*M'/sum(M);

end
```

```
function [zmp,com] = zmp(P1,P2,P3,P5,P6,P7,P8,P10,P11,P12)

M = [55 65 90 55 400 55 90 65 55];
g = 981;

PP1.x = (P1.x + P2.x)/2;
PP1.y = (P1.y + P2.y)/2;
PP1.z = (P1.z + P2.z)/2;

PP2.x = (P2.x + P3.x)/2;
PP2.y = (P2.y + P3.y)/2;
PP2.z = (P2.z + P3.z)/2;

PP3.x = (P3.x + P5.x)/2;
PP3.y = (P3.y + P5.y)/2;
PP3.z = (P3.z + P5.z)/2;

PP5.x = (P5.x + P6.x)/2;
PP5.y = (P5.y + P6.y)/2;
PP5.z = (P5.z + P6.z)/2;

PPM.x = (P6.x + P7.x)/2;
PPM.y = (P6.y + P7.y)/2;
PPM.z = (P6.z + P7.z)/2 + 6;

PP7.x = (P7.x + P8.x)/2;
PP7.y = (P7.y + P8.y)/2;
PP7.z = (P7.z + P8.z)/2;

PP8.x = (P8.x + P10.x)/2;
PP8.y = (P8.y + P10.y)/2;
PP8.z = (P8.z + P10.z)/2;

PP10.x = (P10.x + P11.x)/2;
PP10.y = (P10.y + P11.y)/2;
PP10.z = (P10.z + P11.z)/2;

PP11.x = (P11.x + P12.x)/2;
PP11.y = (P11.y + P12.y)/2;
PP11.z = (P11.z + P12.z)/2;

PPx = [PP1.x PP2.x PP3.x PP5.x PPM.x PP7.x PP8.x PP10.x PP11.x];
PPy = [PP1.y PP2.y PP3.y PP5.y PPM.y PP7.y PP8.y PP10.y PP11.y];
PPz = [PP1.z PP2.z PP3.z PP5.z PPM.z PP7.z PP8.z PP10.z PP11.z];

PP1x_dd = acceleration(PP1.x);
PP2x_dd = acceleration(PP2.x);
PP3x_dd = acceleration(PP3.x);
PP5x_dd = acceleration(PP5.x);
PPMx_dd = acceleration(PPM.x);
PP7x_dd = acceleration(PP7.x);
PP8x_dd = acceleration(PP8.x);
PP10x_dd = acceleration(PP10.x);
PP11x_dd = acceleration(PP11.x);
```

```
PP1y_dd = acceleration(PP1.y);
PP2y_dd = acceleration(PP2.y);
PP3y_dd = acceleration(PP3.y);
PP5y_dd = acceleration(PP5.y);
PPMy_dd = acceleration(PPM.y);
PP7y_dd = acceleration(PP7.y);
PP8y_dd = acceleration(PP8.y);
PP10y_dd = acceleration(PP10.y);
PP11y_dd = acceleration(PP11.y);

PP1z_dd = acceleration(PP1.z);
PP2z_dd = acceleration(PP2.z);
PP3z_dd = acceleration(PP3.z);
PP5z_dd = acceleration(PP5.z);
PPMz_dd = acceleration(PPM.z);
PP7z_dd = acceleration(PP7.z);
PP8z_dd = acceleration(PP8.z);
PP10z_dd = acceleration(PP10.z);
PP11z_dd = acceleration(PP11.z);

PPx_dd = [PP1x_dd PP2x_dd PP3x_dd PP5x_dd PPMx_dd PP7x_dd PP8x_dd PP10x_dd
PP11x_dd];
PPy_dd = [PP1y_dd PP2y_dd PP3y_dd PP5y_dd PPMy_dd PP7y_dd PP8y_dd PP10y_dd
PP11y_dd];
PPz_dd = [PP1z_dd PP2z_dd PP3z_dd PP5z_dd PPMz_dd PP7z_dd PP8z_dd PP10z_dd
PP11z_dd];

com.x = PPx*M'/sum(M);
com.y = PPy*M'/sum(M);
com.z = PPz*M'/sum(M);

zmp.x = com.x + ((PPx.*PPz_dd)*M'-(PPz.*PPx_dd)*M')./(g*sum(M));
zmp.y = com.y + ((PPy.*PPz_dd)*M'-(PPz.*PPy_dd)*M')./(g*sum(M));
end
```

APPENDIX 5

```
function j=fitnessfunction(u)

S=u(1);
H=u(2);
h=u(3);
n=u(4);
d1=4;
d2=4.5;
d3=6.2;
d5=6.2;
w1=6.6;
x1=-2.2;
x2=-1.5;
x3=2.5;
x4=3.3;
y1=-3.3;
y2=-1.2;
y3=-1.2;
y4=-3.3;
```

```
T=1;
t1=0:0.01:0.2;
t2=0.21:0.01:0.8;
t3=0.81:0.01:1;
t4=0:0.01:1;
t5=0:0.01:0.2;
t6=0.21:0.01:1;
t7=0:0.01:0.2;
t8=0.21:0.01:1;
[P1,P6,P12] = hamtoado(d1,d2,d3,d5,S,H,h,n,T,w1,t1,t2,t3,t4,t5,t6,t7,t8);

[deta1,deta2,deta3,deta4,deta5,deta6,deta7,deta8,deta9,deta10,deta11,deta12] = biped_backward(P1,P6,P12);

[P1,P2,P3,P5,P6,P7,P8,P10,P11,P12] = biped_forward(deta1,deta2,deta3,deta5,deta7,deta8,deta10,deta11);

%
[zmp,com] = biped_zmp(P1,P2,P3,P5,P6,P7,P8,P10,P11,P12);
%
zmprefx1=transpose(5*(x2-x1)*t1+x1);
zmprefx2=transpose(5/3*(x3-x2)*t2+1/3*(4*x2-x3));
zmprefx3=transpose((5*(x4-x3)*t3+5*x3-4*x4));

zmprefx=[zmprefx1;zmprefx2;zmprefx3];

zmprefy1=transpose(5*(y2-y1)*t1+y1);
zmprefy2=transpose(5/3*(y3-y2)*t2+1/3*(4*y2-y3));
zmprefy3=transpose((5*(y4-y3)*t3+5*y3-4*y4));

zmprefy=[zmprefy1;zmprefy2;zmprefy3];
```

```
zmprefy=[zmprefy1;zmprefy2;zmprefy3];

A=(com.x(1:end)-zmprefx(1:end)).^2;
B=(com.y(1:end)-zmprefy(1:end)).^2;

j=sum(A)+sum(B);
% plot(zmprefx(1:end),zmprefy(1:end));
% hold on;
% plot(com.x(1:end),com.y(1:end))
```

REFERENCES

[1] http://www.asimo.honda.com/

[2] T. Takenaka, T. Matsumoto, T. Yoshiike, and S. Shirokura, “Real Time Motion Generation and Control for Biped Robot -2nd Report: Running Gait Pattern Generation-”, In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009.

[3] K. Kaneko, F. Kanehiro, S. Kajita, H. Hirukawa, T. Kawasaki, M. Hirata, K. Akachi, T. Isozumi. Humanoid Robot HRP-2, Proceedings of the 2002 IEEE IROS, EPFL, Lausanene, Switzerland, Oct, 2002.

[4] Ho Pham Huy Anh, “A New Approach of Robust Walking Planning for Biped Robot HUBOT-3”, The Second Vietnam Conference On Control and Automation, pp. 64-69, Da Nang-VietNam, 11-2013.

[5] Dip Goswami, Prahlad Vadakkepat, “Genetic Algorithm-based Optimal Bipeda Walking Gait Synthesis considering Tradeoff between Stability Margin and Speed”, Robotica, volume 27, pp. 355-365, 2009.

[6] Ho Pham Huy Anh, Nguyen Thanh Nam, Tran Thien Huan, “A New Approach Of Humanoid Robot Walking Gait Optimization Using Modified Genetic Algorithms”, Hoi Nghi Co Khí Toàn Quoc, pp 324-329, Ha Noi - Viet Nam, 03/2013.

[7] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, “Biped Walking Pattern Generation by using Preview Control of Zero Moment Point”, In Proceedings of the 2003 IEEE International Conference on Robotics and Automation, pp. 1620-1626, 2003.

[8] Qiang Huang, Shuuji Kajia, Noriho Koachi, Kenji Kaneko, Kazuhio Yokoi, Hirohiko Arai, Kiyoshi Komoriya, Kazuo Tanie, “A High Stability, Smooth Walking Patern for a Biped Robot”, Proc. IEEE Inter. Conf. on Robotics and Automation, pp. 65-71, Detroit, Michigan, May 1999.

[9] Shuhei Shimmyo, Tomoya Sato, Kouhei Ohnishi, “Biped Walking Pattern Generation by Using Preview Control Based on Three-Mass Model”, IEEE Transactions on Industrial Electronics, Vol. 60, No. 11, November 2013.

[10] Jin Yongying, Dip Goswami and Prahlad Vadakkepat, “Walking Gait Generation For Humanoid Robot Brail 1.0”, National University of Singapore, Electrical and Computer Engineering, 4 Engineering Drive 3, Singapore – 117576

[11] Philippe Bidaud, Mohammad O.Tokhi, Christophe Grand and Gurvinder S.Virk, “Field Robotics: Proceedings of the 14th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines”, University Pierre et Marie Curie (UPMC), Paris, France, 6-8 September 2011

[12] Huỳnh Thái Hoàng, “Hệ thống điều khiển thông minh”, Nhà xuất bản Đại học Quốc Gia Thành Phố Hồ Chí Minh, 2014

[13] Rania Hassan, Babak Cohanim, Olivier de Weck, Gerhard Vente r, “A comparison of particle swarm optimization and the genetic algorithm”, 2005
[14] Banzhaf, Wolfgang; Nordin, Peter; Keller, Robert; Francone, Frank . “Genetic Programming – An Introduction”. San Francisco, CA: Morgan Kaufmann, 1998.
[15] Kennedy, J.; Eberhart, R. "Particle Swarm Optimization". Proceedings of IEEE International Conference on Neural Networks, 1995.
[16] Poli, R. "An analysis of publications on particle swarm optimisation applications" (PDF). Technical Report CSM-469. Department of Computer Science, University of Essex, UK, 2007