**PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA**

Ministry of Higher Education and Scientific Research

University of Djilali Bounaama - Khemis Miliana

FACULTY OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF TECHNOLOGY

Dissertation

Presented with A View to Obtaining the Degree of Master

In Mechanical Engineering

Option: Mechanical Construction

**Theme**

# Planning a Fast Movements for Robot Manipulators Using Sequential Quadratic Programming

Prepared by:

BOUCHERF Hadjira

Supervised by:

Dr. BENDALI Nadir

**2019-2020**

# Acknowledgments

In the Name of ALLAH, the Most Merciful, the Most Compassionate all praise be to ALLAH, the Lord of the worlds; and prayers and peace be upon MOHAMED his servant and messenger. First and foremost, I must acknowledge my limitless thanks to ALLAH, the ever-Magnificent; the ever-Thankful, for his help and bless. I thank God for the blessing of the mind and the soul, thank god for the good health, ‘Alhamdoulilah’ always and forever.

Special thanks are given to department of technologies, I would like to thank my supervisor, Dr. BENDALI Nadir for his enthusiasm, patience, insightful comments, advices and uncasing ideas that helped me in my research and writing of this thesis. I wish to express my sincere gratitude to Dr. ALLICHE Ridha for the big help he gave me, so I was able to complete the thesis successfully.

I would also like to express my deep gratitude to my family members which includes my parents for their financial support and encouragement throughout my life.

Nevertheless, it is lucky for me to meet some friends who inspirit my effort to overcome these difficulties, a big thanks to Hichem for the big help he gave, Abdou L and Nazih for the support and the positive vibes, Abdou K and Sedik.

# Dedication

This thesis is dedicated to:

My great parents, who never stop giving of themselves in countless ways, the symbol of love and giving, may Allah bless them and protect them… Amen,

My older sister, the free soul girl, the love giver…

My brothers, my peace of mind…

My little sister, who gives me hope and support when things look bleak…

My friends: Naima, Sedik, Rachid, Walid, Mohammed D, Kader B, Amine, Mohammed M, Fatima…

I cherish every moment I spent with all…

All the people in my life who touch my heart, I dedicate this research.

# Abstract

The trajectories planning of a manipulator robot leads to equations of motion that are generally difficult to solve given the mechanism of these robots. We approach in this work, to solve a problem of trajectory planning for two types of robots with two and three degrees of freedom which are subjected to kinematic and / or dynamic constraints and which execute repetitive operations, the trajectory is modeled using cubic spline functions then they are optimized, for this an objective function represents the execution time of the operation will be minimized by using a powerful optimization technique namely sequential quadratic programming (SQP), our goal is that the manipulator robots make fast movements and finish these tasks in a very short time. The results obtained are compared with others from the scientific literature to test the performance and effectiveness of our method.

**Keywords:** Robotic Manipulators, Motion Planning, Sequential quadratic programming, Kinodynamic Constraints, Function Objective.

# ملخص

يؤدي تخطيط مسارات الروبوت المناور إلى معادلات الحركة التي يصعب حلها بشكل عام نظرًا لآلية هذه الروبوتات. نقترب في هذا العمل من حل مشكلة تخطيط المسار لنوعين من الروبوتات بدرجتين وثلاث درجات من الحرية التي تخضع لقيود حركية و / أو ديناميكية تؤدي عمليات متكررة، يتم نمذجة المسار باستخدام وظائف العمود المكعب ثم يتم تحسينها لهذه الوظيفة الموضوعية التي تمثل وقت تنفيذ العملية وسيتم تصغيرها باستخدام تقنية تحسين قوية وهي البرمجة التربيعية المتسلسلة (SQP) و وهدفنا هو أن تقوم الروبوتات المناورة بإجراء حركات سريعة و، إنهاء هذه المهام في وقت قصير جدًا. تتم مقارنة النتائج التي تم الحصول عليها مع النتائج الأخرى من الأدبيات العلمية لاختبار أداء وفعالية طريقتنا.

**كلمات مفتاحية**

المتلاعبات الروبوتية، تخطيط الحركة، برمجة تربيعية متتابعة، القيود الديناميكية الحركية، الهدف الوظيفي

# Résumé

La planification des trajectoires d'un robot manipulateur mène à des équations de mouvement qui sont généralement délicates à résoudre vue le mécanisme de ces robots. On aborde dans ce travail, à résoudre un problème de planification des trajectoires pour deux types de robots avec deux et trois degrés de libertés qui sont soumise à des contraintes cinématiques et/ou dynamiques qui exécutent des opérations répétitives, la trajectoire est modélisée en utilisant les fonctions splines cubiques puis elles sont optimisée pour cela une fonction objective représente le temps d'exécution de l'opération sera minimisée en utilisant une technique d'optimisation puissante à savoir programmation quadratique séquentielle (SQP), notre but est que les robots manipulateurs fassent des mouvements rapide et terminent ses taches en un temps très courts. Les résultats obtenus sont comparés avec d'autres de la littérature scientifique pour tester les performances et l'efficacité de notre méthode.

**Mots clés :** Robots Manipulateur, Planification de Mouvement, Programmation Quadratique Séquentielle, Contraintes Cinématiques et Dynamiques, Fonction Objective.

# Contents

# List of figures

## Chapter I    Introduction to robots manipulator

# List of tables

# Nomenclature

| | |
|---|---|
| D-H: | Denavit-Harterberg |
| DOF: | Degrees of freedom |
| FKS: | Forward Kinematic Solution |
| IKS: | Inverse Kinematic Solution |
| NRBS: | Non-uniform rational basis spline |
| SQP: | Sequential Quadratic Programming |
| 2R: | Two links, R for Rotation |

# General introduction:

Mankind has always strived to give life like qualities to its artifacts. He tries to find substitute for himself to carry out his orders and also work in hostile environment. Broadly robot is a machine that looks and work like a human being. The industry is moving from automation to robotization to increase productivity and to deliver uniform quality.

Historical perspective:

The word robot was first used in 1921 by Czech play right karel Capek in his satirical drama titled Rossum's universal robots. Derived from Czech word robot which literally means "forced lab our".

The term robotics was coined by American author and professor of biochemistry at Boston university, Isaac Asimov in this short story titled runaround.

Robot definitions:

- Webster dictionary: an automatic apparatus or device that performs functions ordinarily as cribbed to humans or operates with what appears to be almost human intelligence.
- Robotic Institute of America (RIA): a robot is a reprogrammable, multifunctional manipulator designed to move material, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks.

Basic components of a robot system:

- Manipulators: series of rigid members called links connected by joints.
- Actuators: provide power to the manipulator.
- Security devices: to monitor position, speed, acceleration, torque etc.…
- Controller: provides the intelligence to make the manipulator perform in a certain manner.
- Power conversion unit: takes signal from controller and converts it into meaningful power level so that actuators can move.

We interduce serial link robot manipulators, the sort of robot arms you might have seen working in factories doing tasks like welding, spray painting or material transfer.

Our work consists in planning the trajectories for robots manipulator in operations where the terminal organ of the robots must pass through several points, for this a resolution approach has been proposed taking into account the nature of the model adopted.

Our thesis was exposed as follows:

The first chapter deals with a generality on robotics, the various robotic mechanical systems, the terminology used, the models of transformation between the operational space and the articular space and the direct and inverse kinematic and dynamic models of a manipulator robot.

Then the second chapter presents the planning of the trajectories for a manipulator arm by studying the different interpolation methods such as Bézier, NURBS and B-Spline.

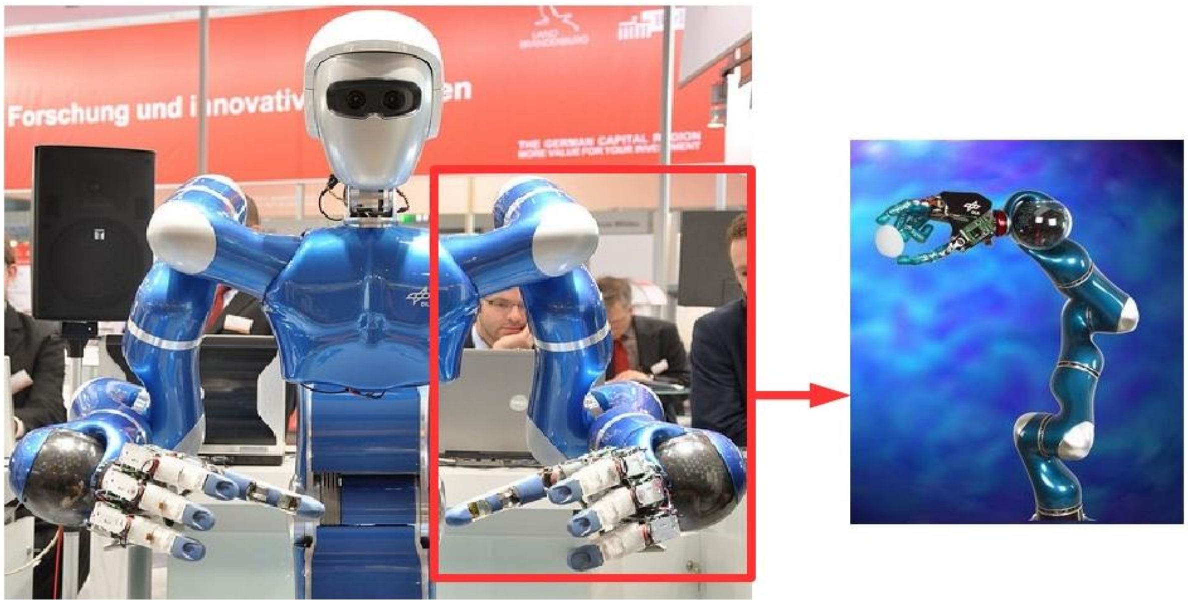
The last chapter is devoted to studying examples for robot manipulators such as: robot 2- DOF and 3-DOF robot.

Finally, the general conclusion which reflects the results of the chapters followed by the perspectives envisaged.

# Chapter I

# Introduction to Manipulator Robots

## I.1 Introduction:



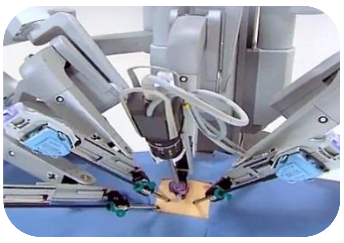
- By general agreement a robot is: [1]
    - A programmable machine that imitates the actions or appearance of an intelligent creature–usually a human.
- To qualify as a robot, a machine must be able to:
  1) Sensing and perception: get information from its surroundings
  2) Carry out different tasks: Locomotion or manipulation, do something physical–such as move or manipulate objects
  3) Re-programmable: can do different things
  4) Function autonomously and/or interact with human beings
- Robot Manipulation is a core robot technology

A. Basic fundamentals of robotics lies in robot manipulation
- kinematics & dynamics
- motion planning & control
- higher mathematics and AI

B. Direct application to industry
- robot manipulators are still a growing market
- technology is now being applied beyond conventional areas
- new research on interactive robots is centered around it

## I.2 New Application Areas:

Small Industries (e.g pharmacutical)


Medical &Surgery

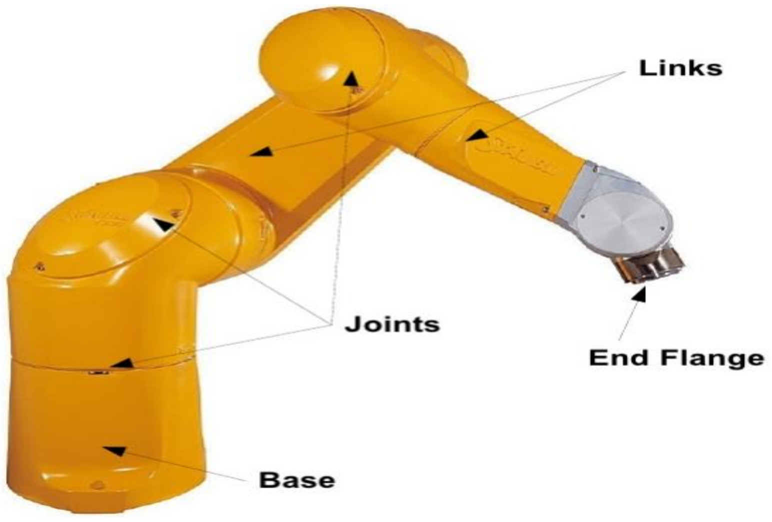
Military & Homeland security

Rehabilitation & Helpers

Future Sevice Robots

Figure 1.1 Application Areas [2]

## I.3 Structure of A Robot Manipulator:



➢ Robot Arm is a

KINEMATIC CHAIN

which is comprised of:
a. Links
b. Joints

Figure 1.2 A Robot Manipulator [2]

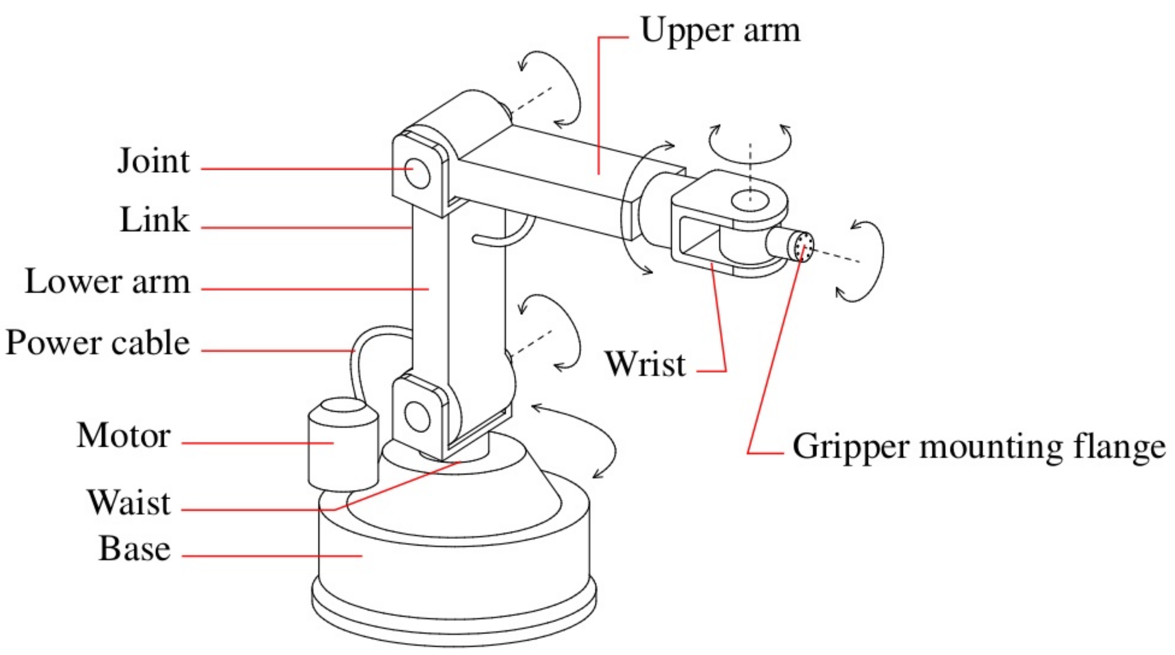


figure 1.3 a robotic arm [3]

### I.3.1 Types of Joints:

**Joints are of two types:**

1. Linear joint – links move in linear fashion with respect to its joint when actuated.
2. Rotary joint – links move in rotary fashion with respect to its joint when actuated.

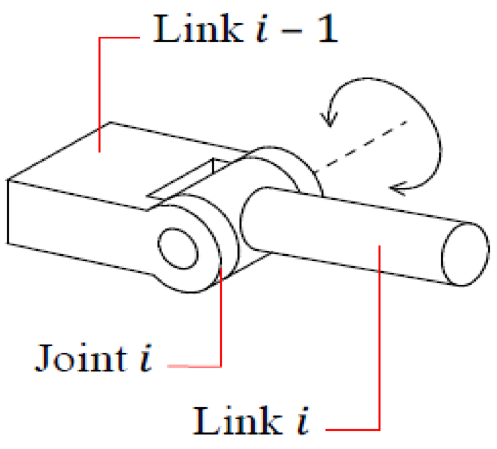


Figure 1.4 Rotary joint [3]

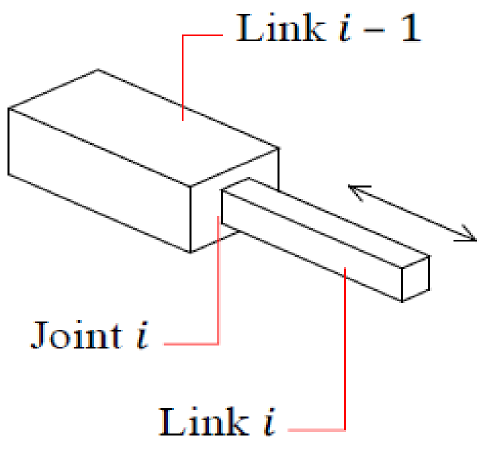


Figure 1.5 Linear joint [3]

### I.3.2 Degrees of Freedom:

a) Degrees of freedom (DOF) is defined as the ability of a joint to produce linear or rotary movement when actuated.
b) Number of DOF for a robot is equal to the number of joint axes in the robotic arm.[3]

### I.3.3 Lower Pair Joints:

1. A lower pair joint is the joint in which two contacting surfaces can slide over with one another in rotary or linear manner.[3]
2. They are of six types:
a) Revolute joint – 1 DOF
b) Prismatic joint – 1 DOF
c) Screw joint – 1 DOF
d) Cylindrical joint – 2 DOF
e) Planar joint – 3 DOF
f) Spherical joint – 3 DOF

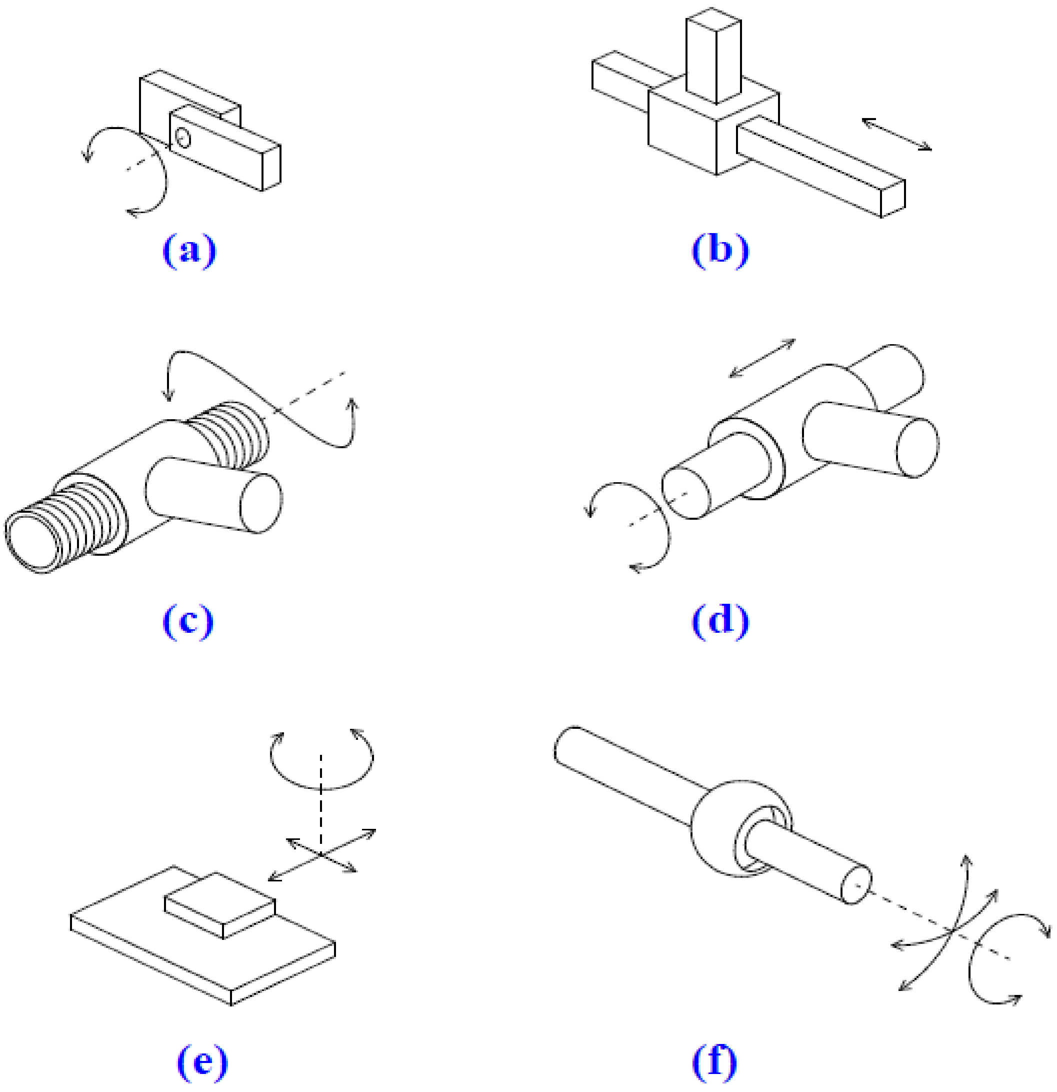


figure 1.6 Different types of lower pair joints [3]

### I.3.4 Link Parameters:

1. Link length – $a$

2. Twist angle – $\alpha$

3. Joint angle – $\theta$

4. Link offset – $d$

### I.3.5 Wrist Motion:

1. Yaw – Rotary motion executed about $z$ axis. Causes movement in left and right directions.
2. Pitch – Rotary motion executed about $y$ axis. Causes movement in up and down directions.
3. Roll – Rotary motion executed about $x$ axis.[3]

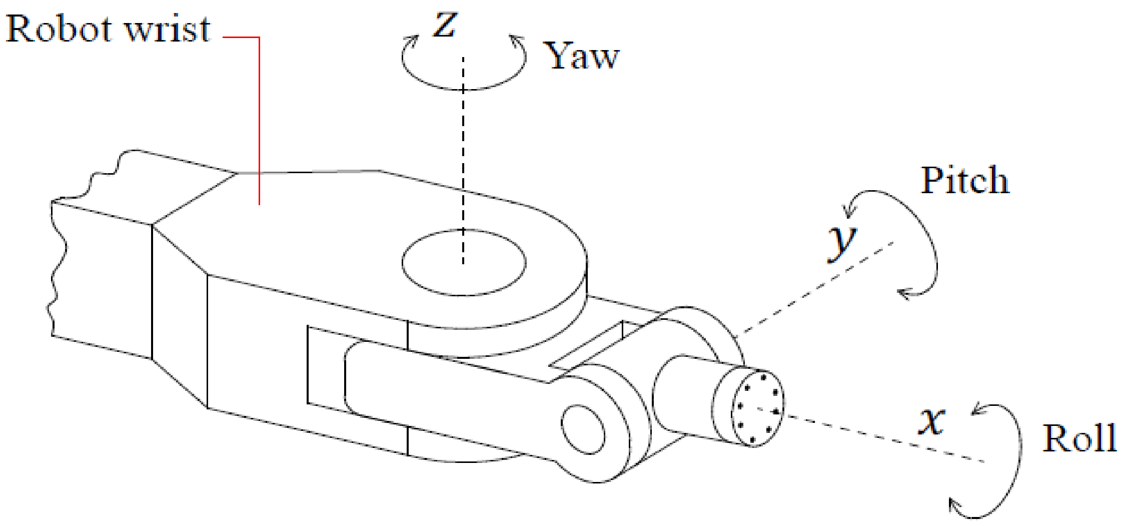


Figure 1.7 Wrist Motion [3]

### I.3.6 Robot's Work Volume:

The three-dimensional space around the robot where it can sweep its wrist end within the points of maximum and minimum reach is called as Robot's work Volume.

#### I.3.6.1 Maximum Reach:

Maximum Reach is the point where the wrist end can go as far as possible from its base.

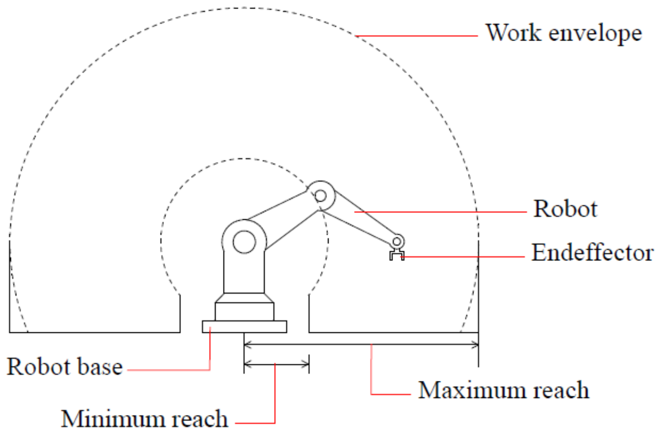


Figure 1.8 Maximum Reach [3]

### I.3.6.2 Minimum reach:

Minimum reach is the point where the wrist end can go as close as possible to its base.

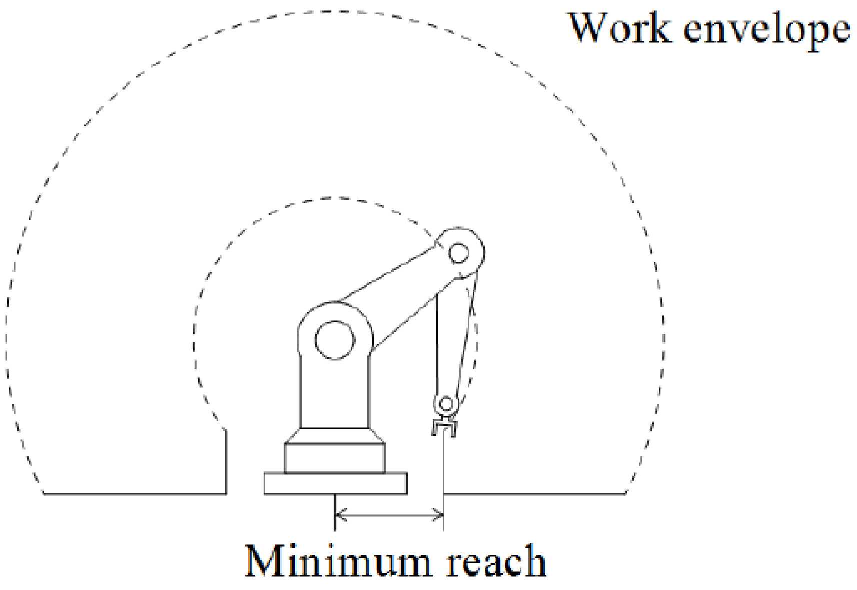


Figure 1.9 Minimum Reach [3]

## I.4 Classification of Manipulator:

1. Cartesian coordinate robot system
2. Cylindrical robot system
3. Polar robot system
4. Pendulum robot system
5. Articulated or Jointed arm robot system
   a) Horizontal axis jointed arm
   b) Vertical axis jointed arm
6. Multiple joint robot system

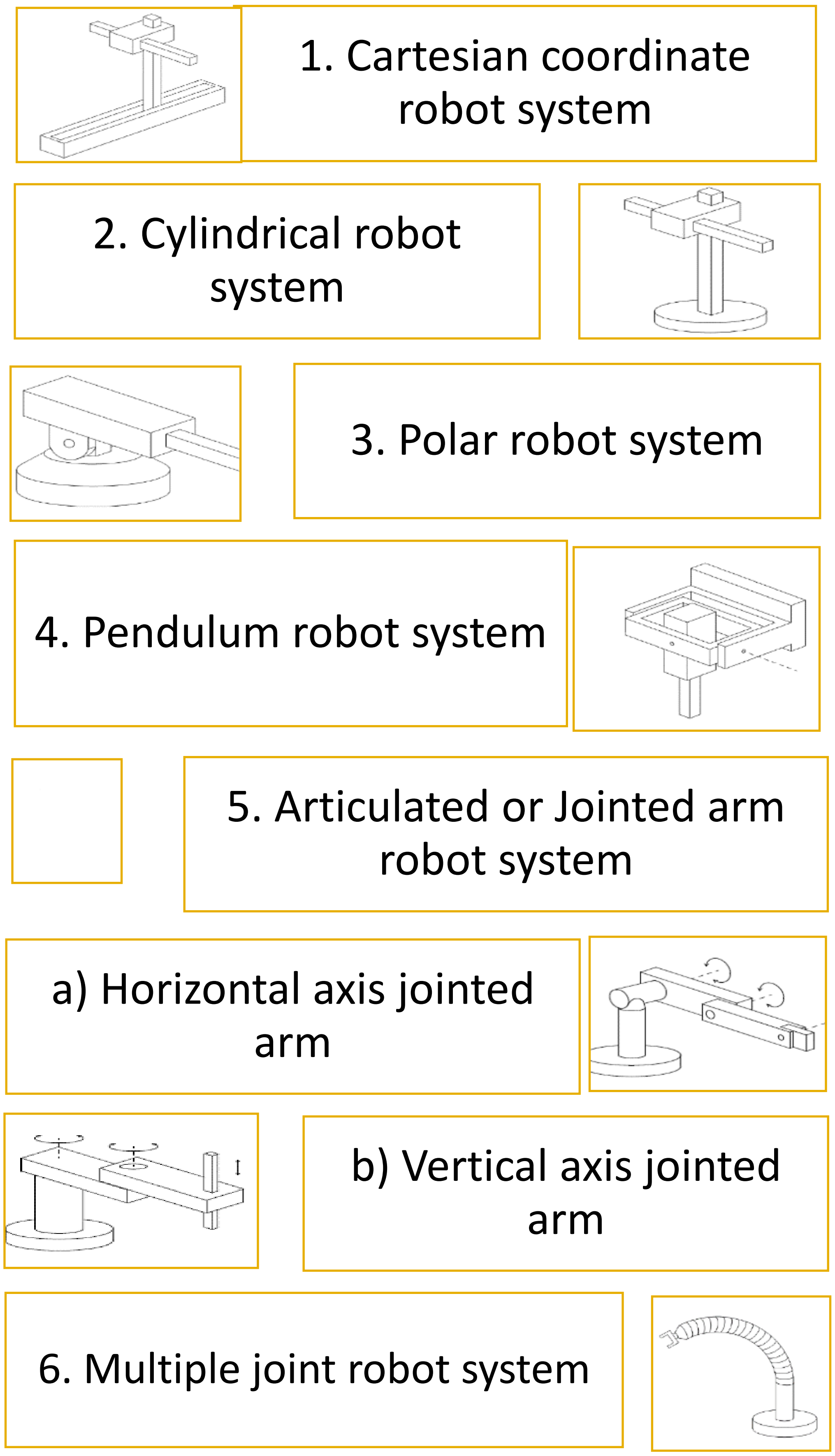
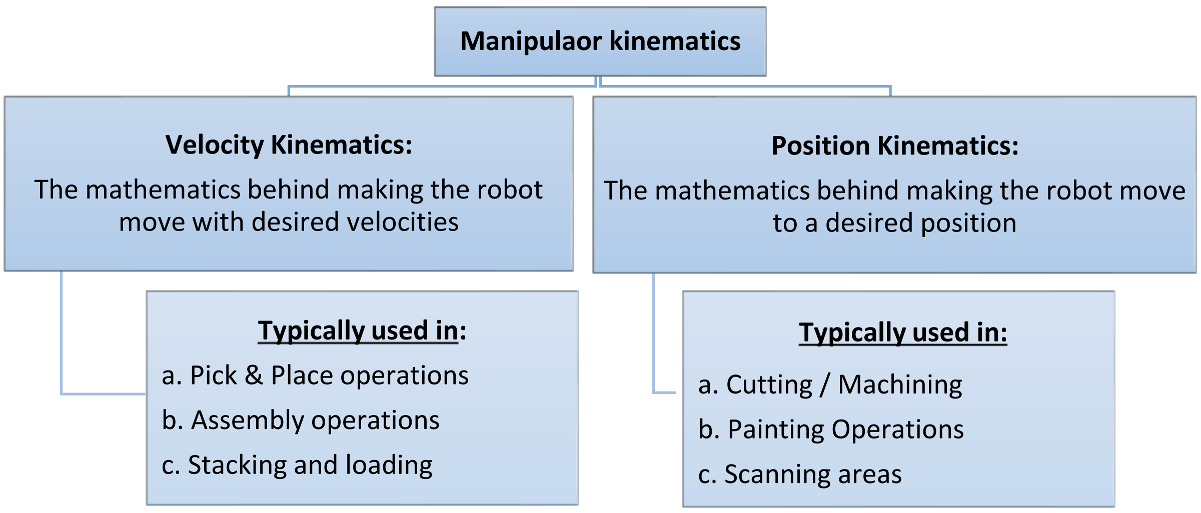

1. Cartesian coordinate robot system
2. Cylindrical robot system
3. Polar robot system
4. Pendulum robot system
5. Articulated or Jointed arm robot system
a) Horizontal axis jointed arm
b) Vertical axis jointed arm
6. Multiple joint robot system

Figure 1.10 Classification of manipulator [3]

## I.5 Manipulator Kinematics:


**Manipulaor kinematics**

**Velocity Kinematics:**
The mathematics behind making the robot move with desired velocities

**Typically used in:**
a. Pick & Place operations
b. Assembly operations
c. Stacking and loading

**Position Kinematics:**
The mathematics behind making the robot move to a desired position

**Typically used in:**
a. Cutting / Machining
b. Painting Operations
c. Scanning areas

### I.5.1 Forward & Inverse Kinematics:

**Forward Kinematics:**

**I know**: The position / velocity of each robot joint.
**I need to find out**: The end position / velocity of the robot

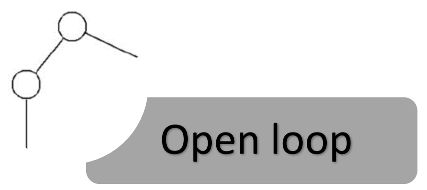
**EASY !!**

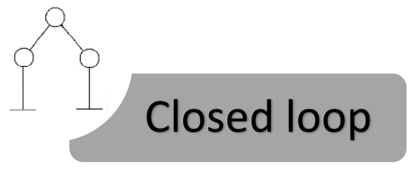
**Inverse Kinematics:**

**I know**: The desired position / velocity of the end point where the robot must reach
**I need to find out**: The position / velocity of each robot joint [2]

**COMPLEX !!**

### I.5.2 Kinematic chains:



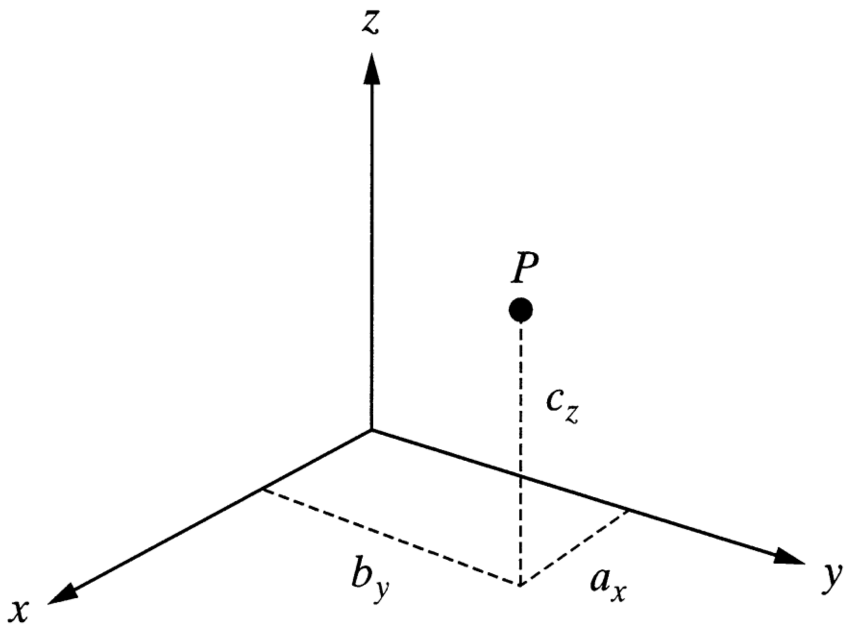


### 1.5.3 Position analysis:

#### 1.5.3.1 Matrix Representation:

##### 1.5.3.1.1 Representation of a Point in Space:

- A point P in space: 3 coordinates relative to a reference frame [4]

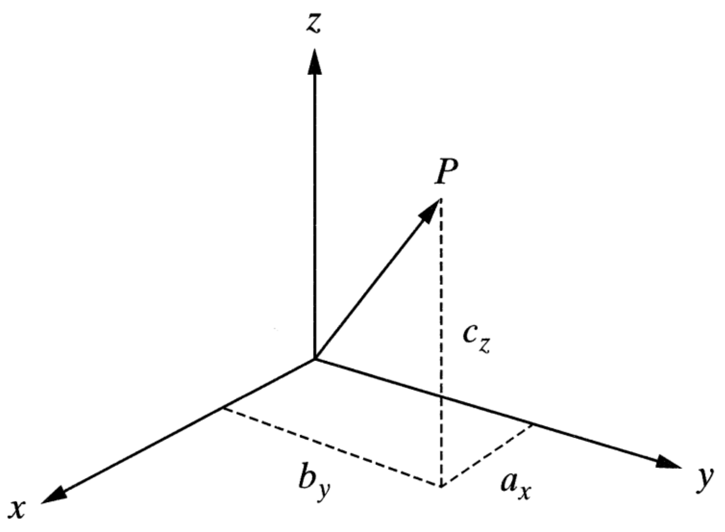


P = ax i +by j+ cz k

Figure 1.11 Representation of a point in space [4]

##### 1.5.3.1.2 Representation of a Vector in Space:

- A Vector P in space: 3 coordinates of its tail and of its head



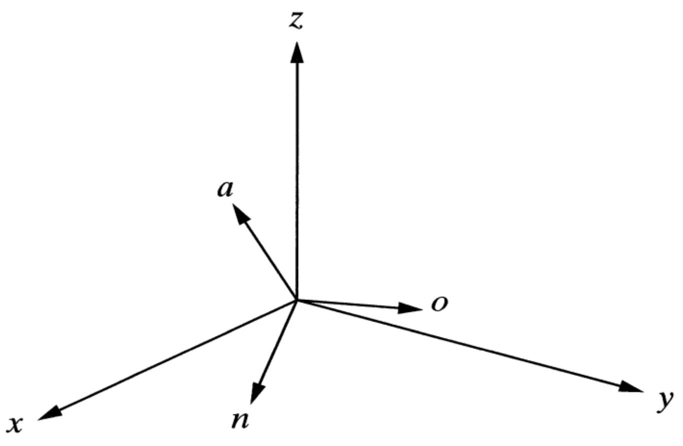
P = ax i+ by j+ cz k

$$P = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Figure 1.12 Representation of a vector in space [4]

#### 1.5.3.1.3 Representation of a Frame at the Origin of a Fixed-Reference Frame:

-Each Unit Vector is mutually perpendicular: normal, orientation, approach vector

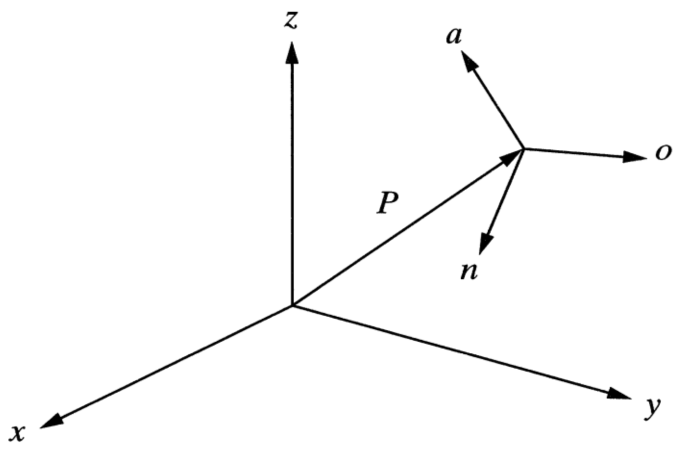


$$F = \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix}$$

Figure1.13 Representation of a frame at the origin of the reference frame [4]

#### 1.5.3.1.4 Representation of a Frame in a Fixed Reference Frame:

Each Unit Vector is mutually perpendicular: normal, orientation, approach vector

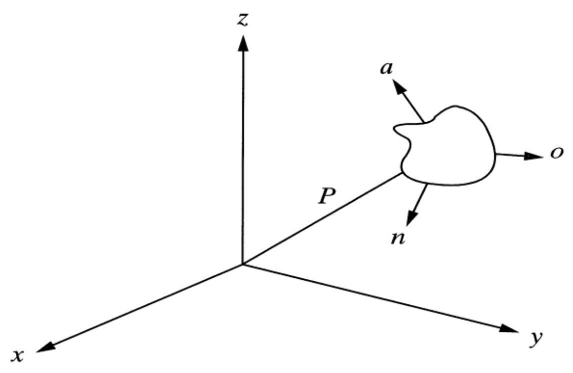


$$F = \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure1.14 Representation of a frame in a frame [4]

#### 1.5.3.1.5 Representation of a Rigid Body:

An object can be represented in space by attaching a frame to it and representing the frame in space.



$$F_{object} = \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure1.15 Representation of an object in space [4]

### 1.5.3.2 Homogeneous Transformation Matrices:

A transformation matrix must be in square form.

• It is much easier to calculate the inverse of square matrices.

• To multiply two matrices, their dimensions must match. [4]

$$F = \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

### 1.5.3.3 Representation of Transformations:

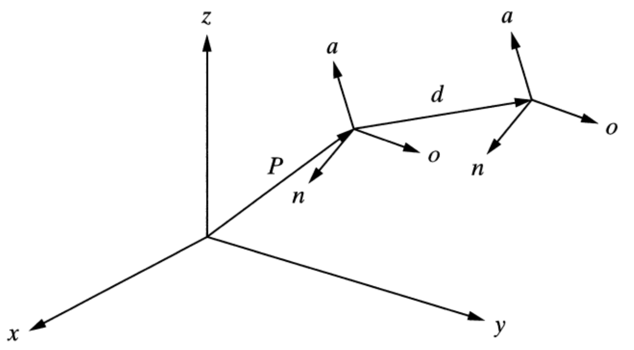
#### 1.5.3.3.1 Representation of a Pure Translation:

A transformation is defined as making a movement in space.

• A pure translation.

• A pure rotation about an axis.

• A combination of translation or rotations.



$$T = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure1.16 Representation of a pure translation in space [4]

#### 1.5.3.3.2 Representation of a Pure Rotation about an Axis:

Assumption: The frame is at the origin of the reference frame and parallel to it.

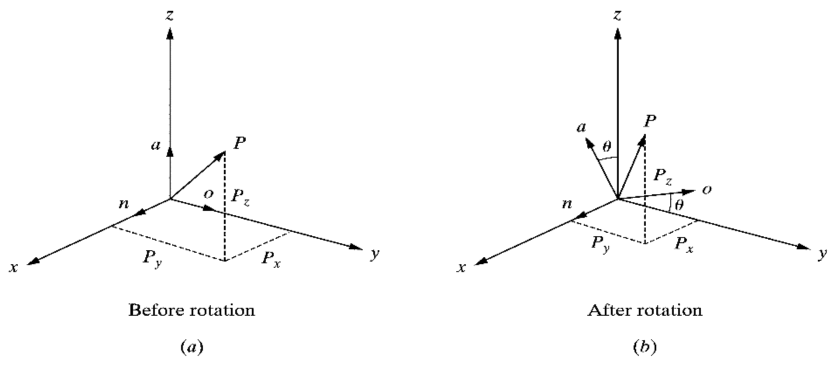


Figure1.17 Coordinates of a point in a rotating frame before and after rotation. [4]

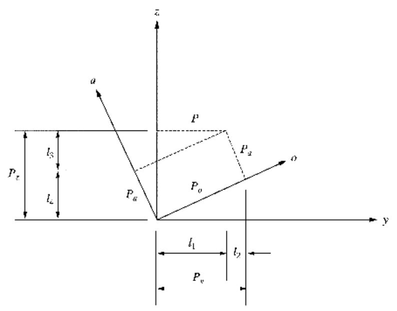


Figure1.18 Coordinates of a point relative to the reference frame and rotating frame as viewed from the x-axis.

### 1.5.3.3.3 Representation of Combined Transformations:

-A number of successive translations and rotations….

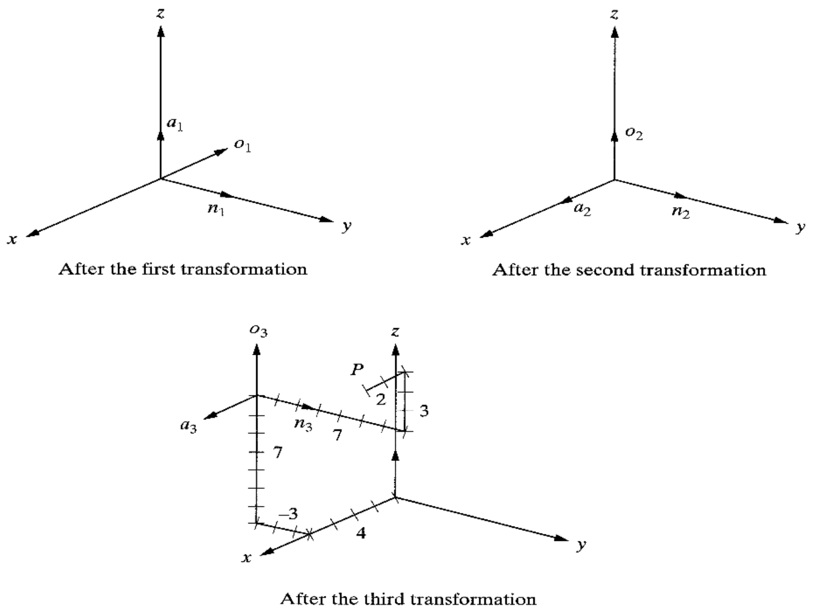

z
$a_1$
$o_1$
$n_1$
x
y
After the first transformation
z
$o_2$
$a_2$
$n_2$
x
y
After the second transformation
$o_3$
z
P
2
3
$a_3$
$n_3$
7
7
−3
4
x
y
After the third transformation

Figure1.19 Effects of three successive transformations

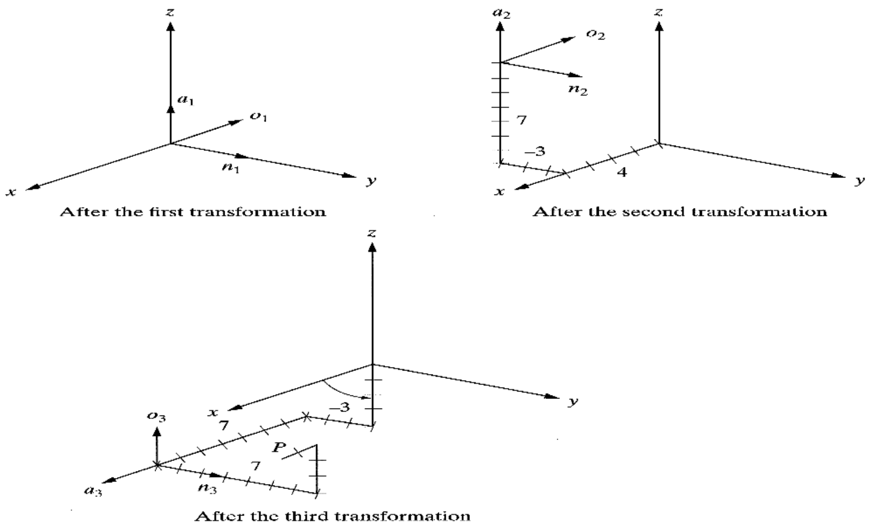

z
$a_1$
$o_1$
$n_1$
x
y
After the first transformation
$a_2$
$o_2$
z
$n_2$
7
−3
4
x
y
After the second transformation
z
y
−3
x
7
$o_3$
P
7
$a_3$
$n_3$
After the third transformation

Figure1.20 Changing the order of transformations will change the final result

#### 1.5.3.3.4 Transformations Relative to the Rotating Frame:

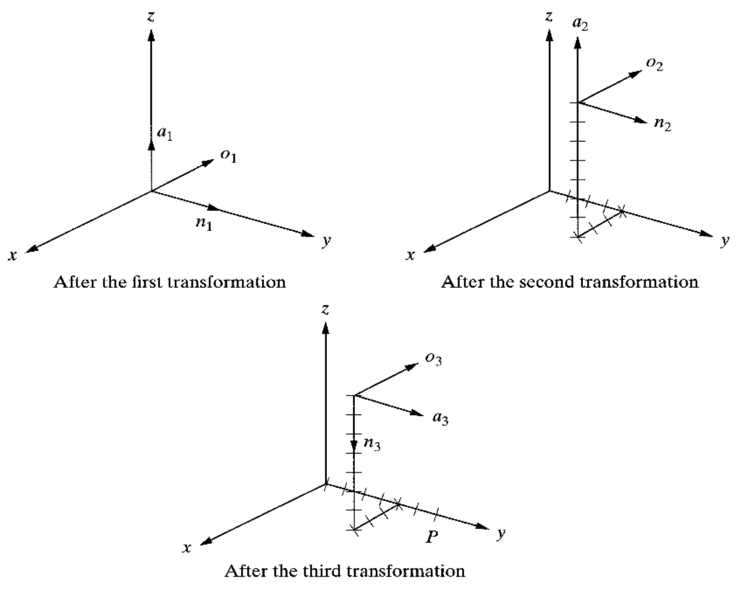

z
$a_1$
$o_1$
$n_1$
x
y
After the first transformation
z
$a_2$
$o_2$
$n_2$
x
y
After the second transformation
z
$o_3$
$a_3$
$n_3$
x
P
y
After the third transformation

Figure1.21 Transformations relative to the current frames. [4]

### 1.5.3.4 Inverse of Transformation Matrices:

Inverse of a matrix calculation steps:
• Calculate the determinant of the matrix.
• Transpose the matrix.
• Replace each element of the transposed matrix by its own minor (adjoint matrix).
• Divide the converted matrix by the determinant. [4]

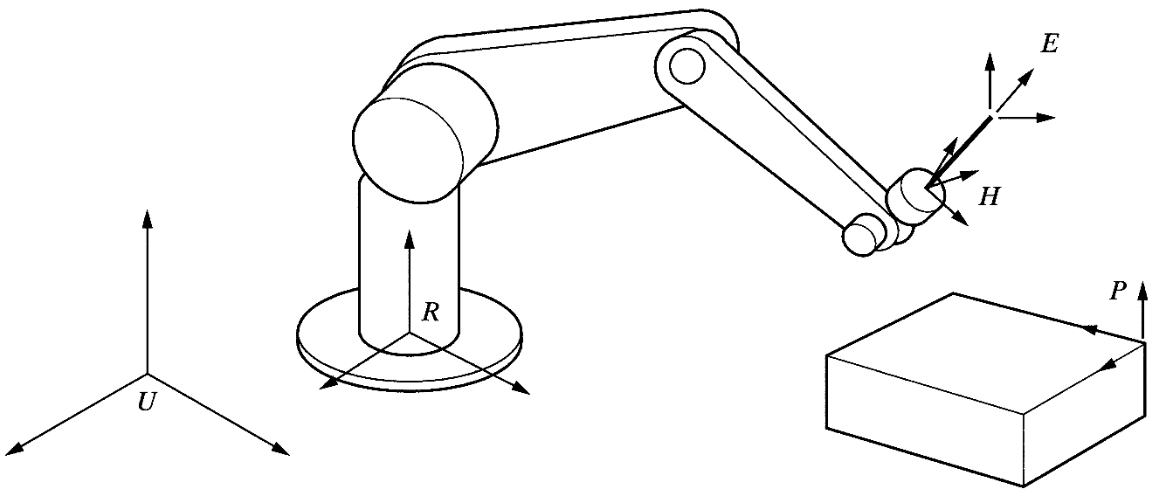


Figure1.22 The Universe, robot, hand, part, and end effecter frames.

### 1.5.3.5 Forward and Inverse Kinematics of Robots:

• Calculating the position and orientation of the hand of the robot.
• If all robot joint variables are known, one can calculate where the robot is at any instant.

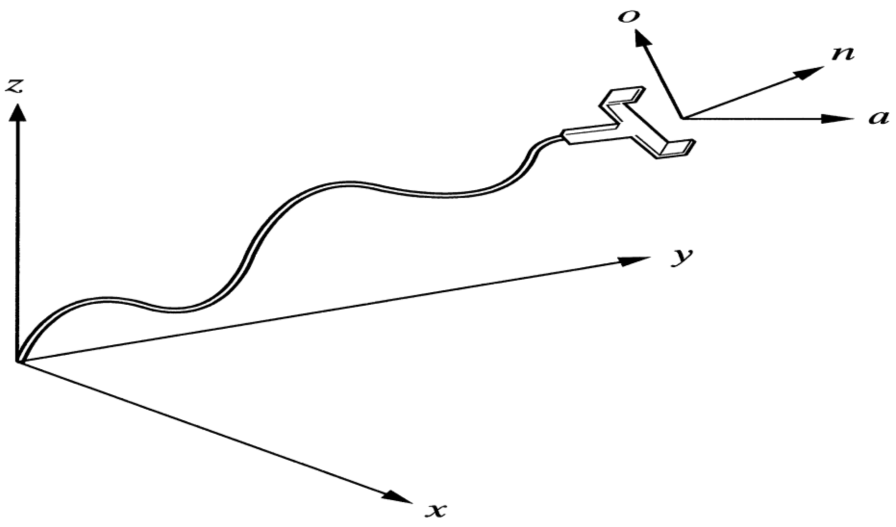


Figure1.23 The hand frame of the robot relative to the reference frame.

#### 1.5.3.5.1 Forward and Inverse Kinematics Equations for Position:

Forward Kinematics and Inverse Kinematics equation for position analysis:

(a) Cartesian (gantry, rectangular) coordinates.

(b) Cylindrical coordinates.

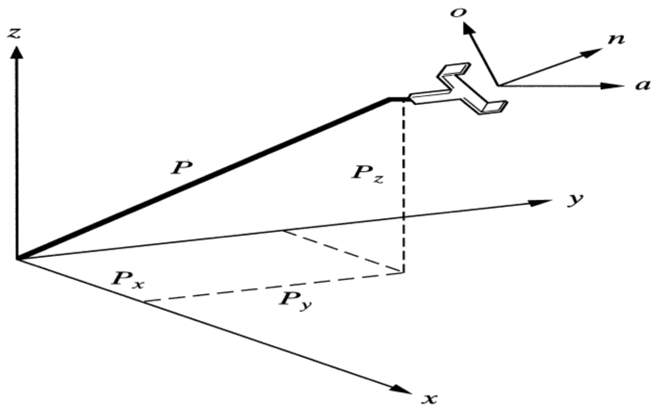
(c) Spherical coordinates.

(d) Articulated (anthropomorphic, or all-revolute) coordinates.

**<u>(a) Cartesian (Gantry, Rectangular) Coordinates:</u>**

IBM 7565 robot

• All actuator is linear.
• A gantry robot is a Cartesian robot.



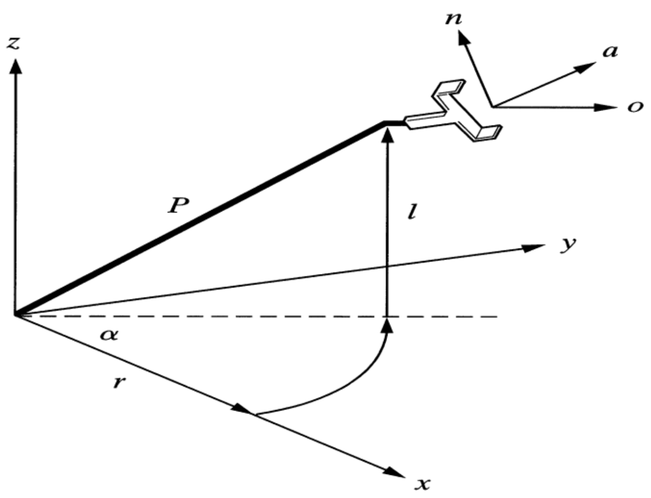
$$ {}^{R}T_{P} = T_{cart} = \begin{bmatrix} 1 & 0 & 0 & P_x \\ 0 & 1 & 0 & P_y \\ 0 & 0 & 1 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Figure1.24 Cartesian Coordinates.

**(b) Cylindrical Coordinates:**

**2 Linear translations and 1 rotation**

• translation of r along the x-axis
• rotation of α about the z-axis
• translation of l along the z-axis



$${}^{R}T_{P} = T_{cyl}(r,\alpha,l) = \text{Trans}(0,0,l)\text{Rot}(z,\alpha)\text{Trans}(r,0,0)$$

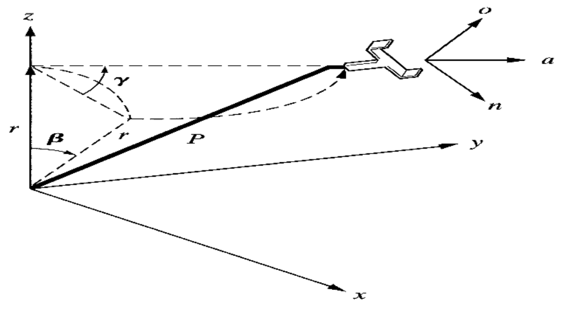
$${}^{R}T_{P} = T_{cyl} = \begin{bmatrix} C\alpha & -S\alpha & 0 & rC\alpha \\ S\alpha & C\alpha & 0 & rS\alpha \\ 0 & 0 & 1 & l \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure1.25 Cylindrical Coordinates.

**(c) Spherical Coordinates:**

**2 Linear translations and 1 rotation**

• translation of r along the z-axis
• rotation of β about the y-axis
• rotation of γ along the z-axis



$${}^{R}T_{P} = T_{cyl}(r,\alpha,l) = \text{Trans}(0,0,l)\text{Rot}(z,\alpha)\text{Trans}(r,0,0)$$

$${}^{R}T_{P} = T_{cyl} = \begin{bmatrix} C\alpha & -S\alpha & 0 & rC\alpha \\ S\alpha & C\alpha & 0 & rS\alpha \\ 0 & 0 & 1 & l \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure1.26 Spherical Coordinates.

**(d) Articulated Coordinates:**

3 rotations -> Denavit-Hartenberg representation

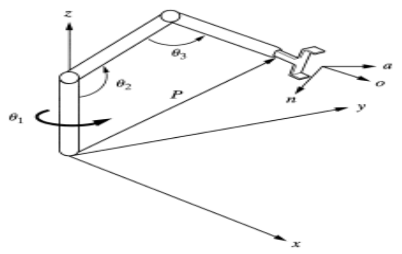


Figure1.27 Articulated Coordinates.

**1.5.3.5.2 Forward and Inverse Kinematics Equations for Orientation:**

Roll, Pitch, Yaw (RPY) angles
Euler angles
Articulated joints

**(a) Roll, Pitch, Yaw (RPY) Angles:**

Roll: Rotation of $\phi a$ about *a*-axis (z-axis of the moving frame)

Pitch: Rotation of $\phi o$ about *o*-axis (y-axis of the moving frame)

Yaw: Rotation of $\phi n$ about *n*-axis (x-axis of the moving frame)

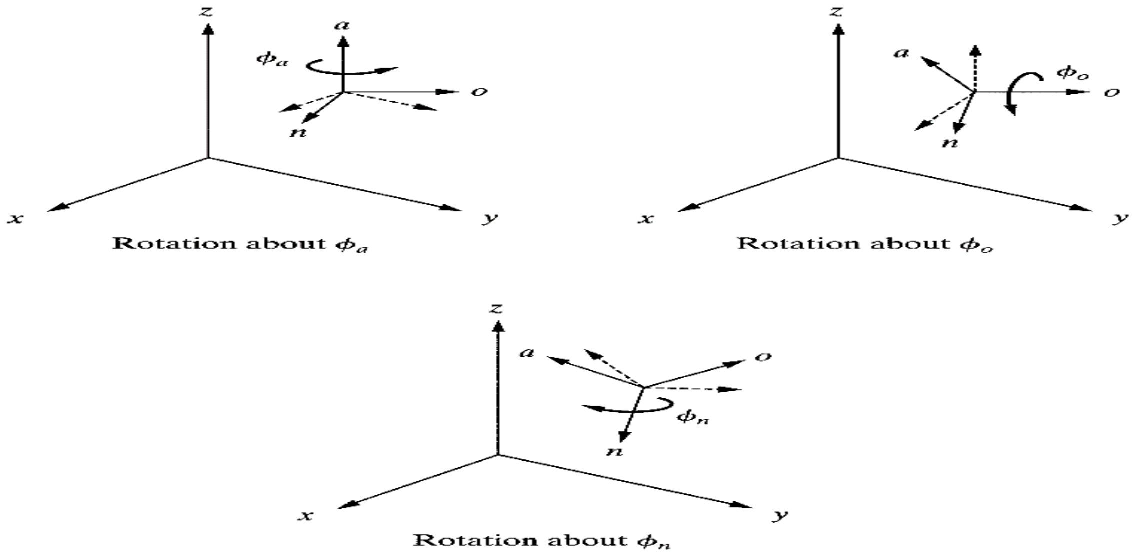


Figure1.28 RPY rotations about the current axes.

**(b) Euler Angles:**

Rotation of ϕ about a-axis (z-axis of the moving frame) followed by
Rotation of ϴ about o-axis (y-axis of the moving frame) followed by
Rotation of Ψ about a-axis (z-axis of the moving frame).

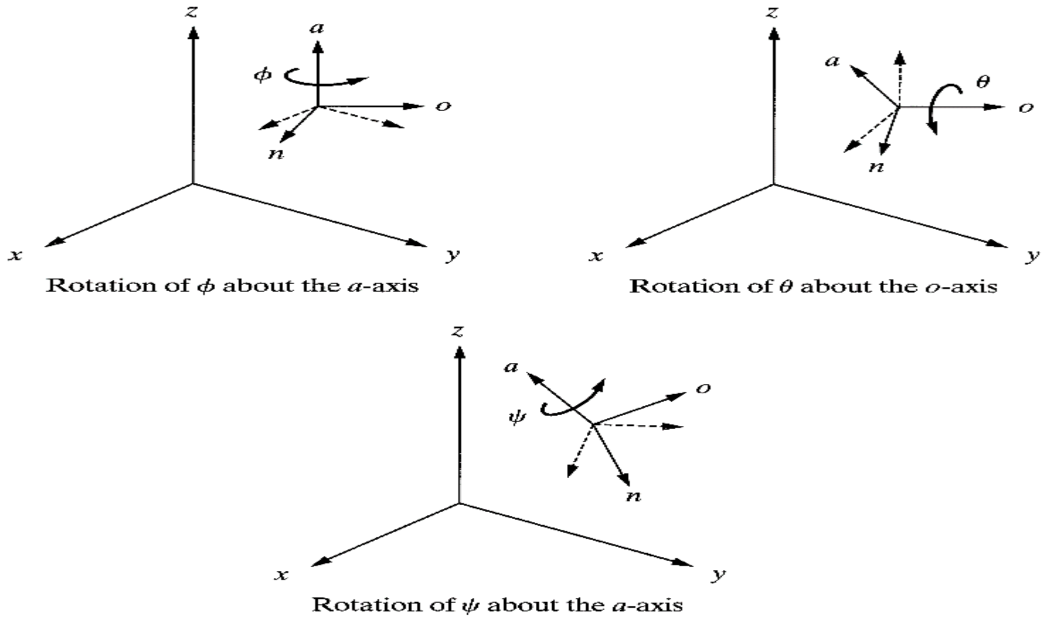

z
a
$\phi$
o
n
x
y
Rotation of $\phi$ about the $a$-axis
z
a
$\theta$
o
n
x
y
Rotation of $\theta$ about the $o$-axis
z
a
$\psi$
o
n
x
y
Rotation of $\psi$ about the $a$-axis

Figure 1.29 Euler rotations about the current axes.

**(c) Articulated Joints:**

Consult again section 1.5.3.5.1 (d)……

**1.5.3.5.3 Forward and Inverse Kinematics Equations for Orientation:**

Assumption: Robot is made of a Cartesian and an RPY set of joints.

$${}^{R}T_H = T_{cart}(P_x, P_y, P_z) \times RPY(\phi_a, \phi_o, \phi_n)$$

Assumption: Robot is made of a Spherical Coordinate and an Euler angle.

$${}^{R}T_H = T_{sph}(r, \beta, \gamma) \times Euler(\phi, \theta, \psi)$$

Another Combination can be possible……

Denavit-Hartenberg Representation

**1.5.3.4 Denavit-Hartenberg Representation of Forward Kinematic Equations of Robot:**

**Denavit-Hartenberg Representation:**

@ Simple way of modeling robot links and joints for any robot configuration, regardless of its sequence or complexity.

@ Transformations in any coordinates is possible.

@ Any possible combinations of joints and links and all-revolute articulated robots can be represented.

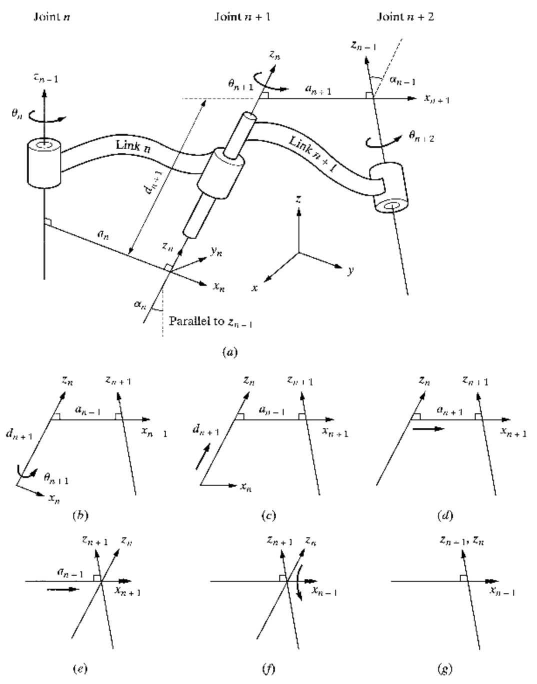


Figure1.30 A D-H representation of a general-purpose joint-link combination

Denavit-Hartenberg Representation procedures:
Start point:
Assign joint number n to the first shown joint.
Assign a local reference frame for each and every joint before or after these joints.
Y-axis does not used in D-H representation.
Procedures for assigning a local reference frame to each joint:
* All joints are represented by a z-axis.
(right-hand rule for rotational joint, linear movement for prismatic joint)
* The common normal is one line mutually perpendicular to any two skew lines.
* Parallel z-axes joints make a infinite number of common normal.
* Intersecting z-axes of two successive joints make no common normal between them (Length is 0.).
Symbol Terminologies:
⊙ q: A rotation about the z-axis.
⊙ d: The distance on the z-axis.
⊙ a: The length of each common normal (Joint offset).
⊙ a: The angle between two successive z-axes (Joint twist)

- Only q and d are joint variables.

The necessary motions to transform from one reference frame to the next.
(I) Rotate about the zn-axis an able of qn+1. (Coplanar)
(II) Translate along zn-axis a distance of dn+1 to make xn and xn+1 colinear.
(III) Translate along the xn-axis a distance of an+1 to bring the origins of xn+1 together.
(IV) Rotate zn-axis about xn+1 axis an angle of an+1 to align zn-axis with zn+1-axis.

### 1.5.3.5 The Inverse Kinematic Solution of Robot:

Determine the value of each joint to place the arm at a desired position and orientation.

$$^{R}T_{H} = A_1 A_2 A_3 A_4 A_5 A_6$$

$$= \begin{bmatrix} C_1(C_{234}C_5C_6 - S_{234}S_6) - S_1S_5C_6 & C_1(-C_{234}C_5C_6 - S_{234}C_6) + S_1S_5C_6 & C_1(C_{234}S_5) + S_1C_5 & C_1(C_{234}a_4 + C_{23}a_3 + C_2a_2) \\ S_1(C_{234}C_5C_6 - S_{234}S_6) + C_1S_5C_6 & S_1(-C_{234}C_5C_6 - S_{234}C_6) - C_1S_5C_6 & S_1(C_{234}S_5) - C_1C_5 & S_1(C_{234}a_4 + C_{23}a_3 + C_2a_2) \\ S_{234}C_5C_6 + C_{234}S_6 & -S_{234}C_5C_6 + C_{234}C_6 & S_{234}S_5 & S_{234}a_4 + S_{23}a_3 + S_2a_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_1^{-1}\times\begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_1^{-1}[RHS] = A_2A_3A_4A_5A_6$$

$$\begin{bmatrix} C_1 & S_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ S_1 & -C_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\times\begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_2A_3A_4A_5A_6$$

$$\theta_1 = \tan^{-1}\left(\frac{p_y}{p_x}\right)$$

$$\theta_2 = \tan^{-1}\frac{(C_3a_3+a_2)(p_z-S_{234}a_4)-S_3a_3(p_xC_1+p_yS_1-C_{234}a_4)}{(C_3a_3+a_2)(p_xC_1+p_yS_1-C_{234}a_4)+S_3a_3(P_z-S_{234}a_4)}$$

$$\theta_3 = \tan^{-1}\left(\frac{S_3}{C_3}\right)$$

$$\theta_4 = \theta_{234}-\theta_2-\theta_3$$

$$\theta_5 = \tan^{-1}\frac{C_{234}(C_1a_x+S_1a_y)+S_{234}a_z}{S_1a_x-C_1a_y}$$

$$\theta_6 = \tan^{-1}\frac{-S_{234}(C_1n_x+S_1n_y)+S_{234}n_z}{-S_{234}(C_1o_x+S_1o_y)+C_{234}o_z}$$

### 1.5.3.6 Inverse Kinematic Program of Robots:

· A robot has a predictable path on a straight line,
· Or an unpredictable path on a straight line.
* A predictable path is necessary to recalculate joint variables.
(Between 50 to 200 times a second)
* To make the robot follow a straight line, it is necessary to break the line into many small sections.
* All unnecessary computations should be eliminated.

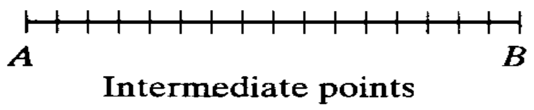


Figure1.31 Small sections of movement for straight-line motions

### 1.5.3.7 Degeneracy and Dexterity:

-Degeneracy: The robot loses a degree of freedom and thus cannot perform as desired.

* When the robot's joints reach their physical limits, and as a result, cannot move any further.

* In the middle point of its workspace if the z-axes of two similar joints becomes colinear.

-Dexterity: The volume of points where one can position the robot as desired, but not orientate it.

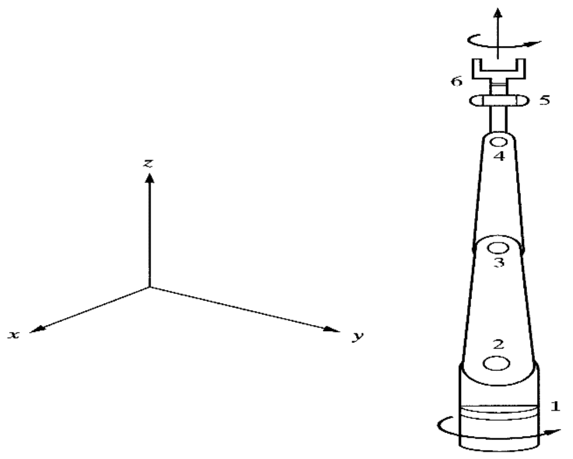


Figure1.32 An example of a robot in a degenerate position.

### 1.5.3.8 The Fundamental Problem With D-H Representation:

Defect of D-H presentation: D-H cannot represent any motion about the y-axis, because all motions are about the x- and z-axis.

Table 1.1 the parameters table for the stanford arm

| # | θ | d | a | α |
|---|---|---|---|---|
| **1** | $\theta_1$ | 0 | 0 | -90 |
| **2** | $\theta_2$ | $d_1$ | 0 | 90 |
| **3** | 0 | $d_2$ | 0 | 0 |
| **4** | $\theta_4$ | 0 | 0 | -90 |
| **5** | $\theta_5$ | 0 | 0 | 90 |
| **6** | $\theta_6$ | 0 | 0 | 0 |

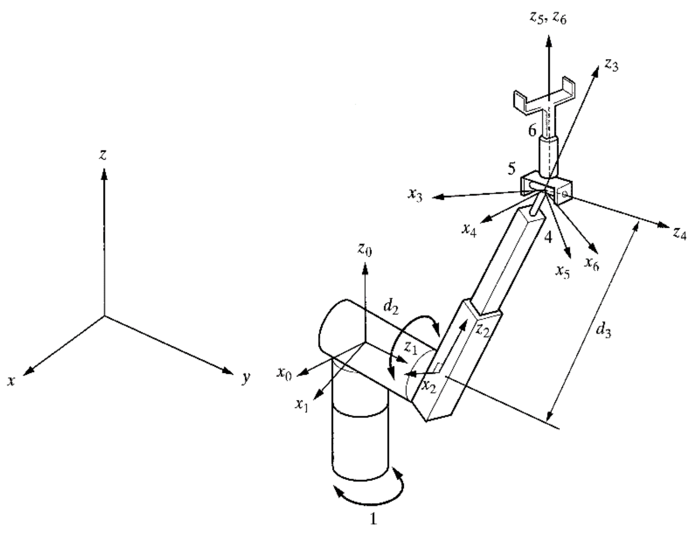


Figure1.33 The frames of the Stanford Arm.

## 1.6 Manipulator Dynamics:

Mathematical equations describing the dynamic behavior of the manipulator

- For computer simulation
- Design of suitable controller
- Evaluation of robot structure

Joint torques $\longleftrightarrow$ Robot motion, i.e. acceleration, velocity, position

### 1.6.1 Lagrange-Euler Formulation:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right)-\frac{\partial L}{\partial q_i}=\tau_i$$

Lagrange function is defined,

K: Total kinetic energy of robot

P: Total potential energy of robot

$q_i$: Joint variable of i-th joint

$\dot{q}_i$: first time derivative of i-th joint

$\tau_i$: Generalized force (torque) at i-th joint

### 1.6.2 Kinetic energy

- Single particle: $k=\frac{1}{2}mv^2$

- Rigid body in 3-D space with linear velocity (V) , and angular velocity ( ) about the center of mass $k=\frac{1}{2}mV^TV+\frac{1}{2}I\omega^T\omega$

$$I\equiv\begin{bmatrix}\int(y^2+z^2)dm & -\int xydm & -\int xzdm\\ -\int xydm & \int(x^2+z^2)dm & -\int yzdm\\ -\int xzdm & -\int yzdm & \int(x^2+y^2)dm\end{bmatrix}$$

**I: Inertia Tensor:**

- Diagonal terms: moments of inertia $I_{xx}=\int(y^2+z^2)dm$
- Off-diagonal terms: products of inertia $I_{xy}=\int(xy)dm$

➢ **Total kinetic energy of a robot arm**

$$K = \sum_{i=1}^{n} K_i = \frac{1}{2}\sum_{i=1}^{n} Tr\left[\sum_{p=1}^{i}\sum_{r=1}^{i} U_{ip}\left(\int r_i^i r_i^{iT} dm\right)U_{ir}^T \dot{q}_p \dot{q}_r\right]$$

$$= \frac{1}{2}\sum_{i=1}^{n}\sum_{p=1}^{i}\sum_{r=1}^{i}\left[Tr(U_{ip} I_i U_{ir}^T)\dot{q}_p \dot{q}_r\right]$$

Scalar quantity, function of $\mathbf{q_i}$ and, $\dot{\mathbf{q}}_{\mathbf{i}}$, i=1,2….n

$I_i$: Pseudo-inertia matrix of link i, dependent on the mass distribution of link i and are expressed w.r.t. the i-th frame,

Need to be computed once for evaluating the kinetic energy

**Potential energy of link i**

$$P_i = -m_i g \bar{r}_i^0 = -m_i g(T_i^0 \bar{r}_i^i)$$

$\bar{r}_i^0$ = center of mass w.r.t. base frame

$\bar{r}_i^i$ = center of mass w.r.t. i-th frame

$$g = (g_x, g_y, g_z, 0)$$

**|g|=9.8 m/sec²**

**g**= gravity row vector expressed in base frame

**Potential energy of a robot arm**

$$P = \sum_{i=1}^{n} P_i = \sum_{i=1}^{n}[-m_i g(T_i^0 \bar{r}_i^i)]$$

Function of $\mathbf{q_i}$

## I.6 Conclusion:

In this chapter, we have presented the terminology of robotics terms, the classification of manipulators, the direct and inverse kinematic model and the dynamic modeling of Euler-Lagrange with the necessary equations to calculate the force and / or the torque motor of each joint. The study done in this chapter has enabled us to prepare the various mathematical models for robots manipulator. The goal is to understand and introduce useful aspects to our approach in the rest of our work.

# Chapter II

# Motion planning (path and trajectory planning)

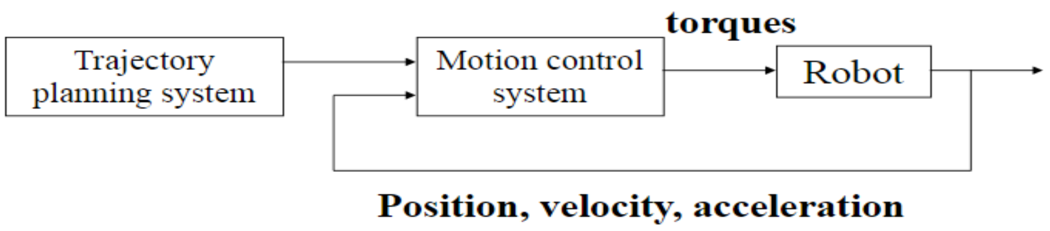
## II.1 Introduction:

- Path and trajectory planning mean the way that a robot is moved from one location to another in a controlled manner.
- The sequence of movements for a controlled movement between motion segment, in straight-line motion or in sequential motions.
- It requires the use of both kinematics and dynamics of robots.[6]

Trajectory planning is carried out in the articular space of the robot, after a kinematic inversion of the given geometric trajectory. The trajectories are then obtained through a functional interpolation which meets the constraints kinematics and dynamics imposed. Almost all the techniques found in the scientific literature on the problem of trajectory planning is based on the optimization of certain parameters or an objective function: minimize execution time, minimize energy and minimize the torque.[7] the goal is to generate the reference inputs to the motion control system which ensures that the manipulator executes the planned trajectory [8]

In this chapter, we will discuss the different methods of planning trajectories:

- Bézier;
- B-spline
- NURBS



## II.2 path vs. trajectory:

**Path**: A sequence of robot configurations in a particular order without regard to the timing of these configurations.

**Trajectory**: It concerned about when each part of the path must be attained, thus specifying timing.

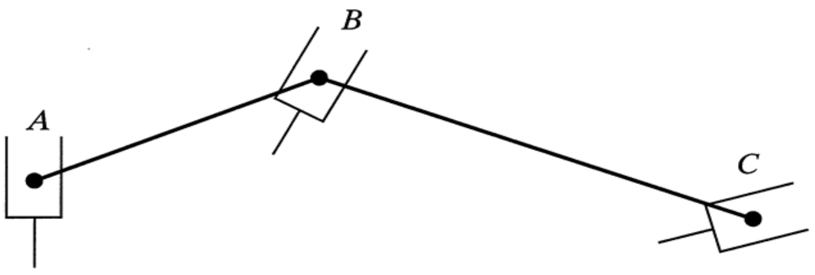
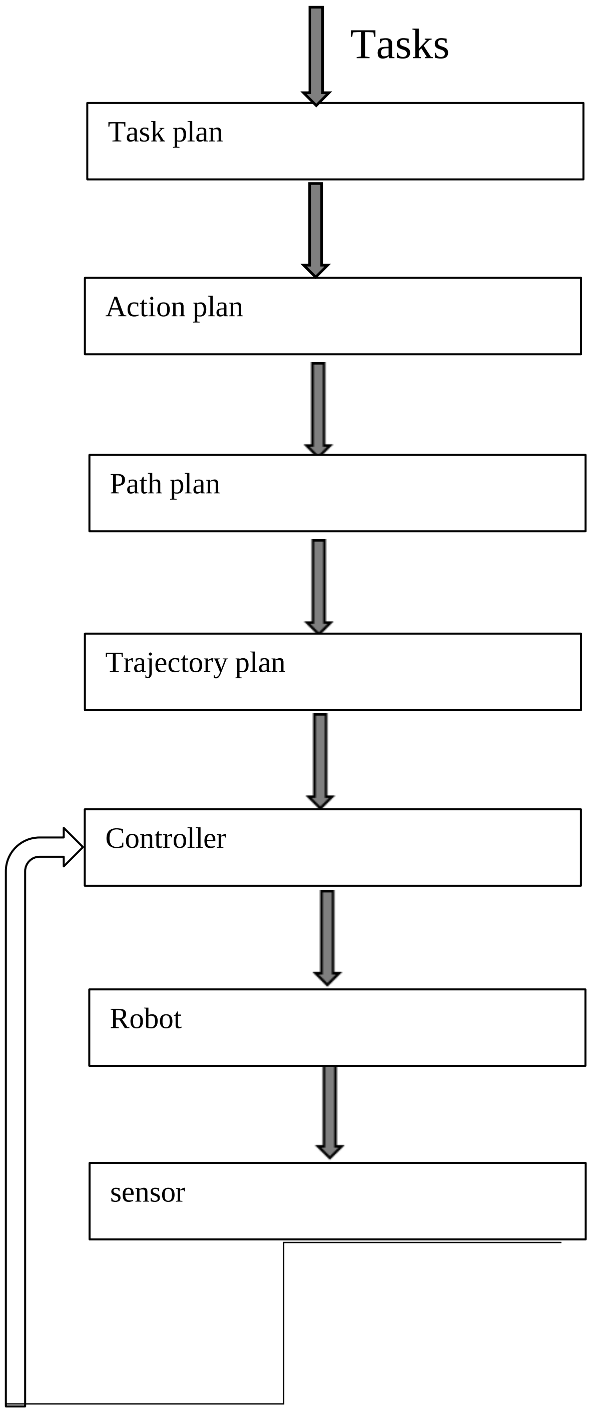


Figure2.1 Sequential robot movements in a path [8]



## II.3 Joint-Space Vs. Cartesian-Space Descriptions:

● Joint-space description:

- The description of the motion to be made by the robot by its joint values.
- The motion between the two points is unpredictable.

● Cartesian space description:

- The motion between the two points is known at all times and controllable.
- It is easy to visualize the trajectory but is difficult to ensure that singularity.[9]

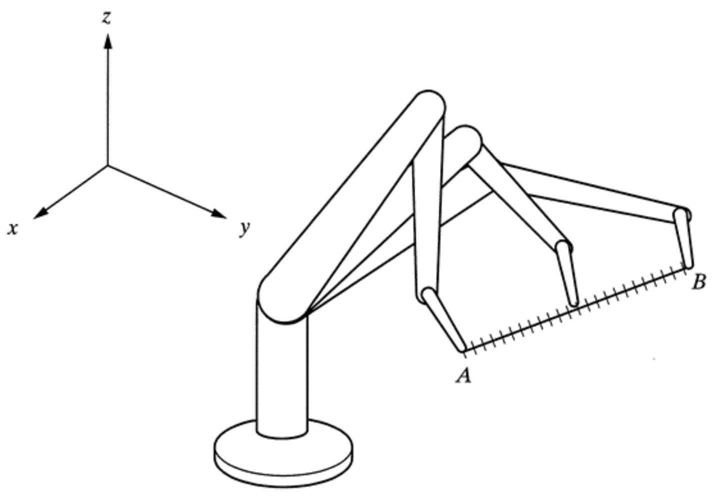


Figure2.2 Sequential motions of a robot to follow a straight line.[9]

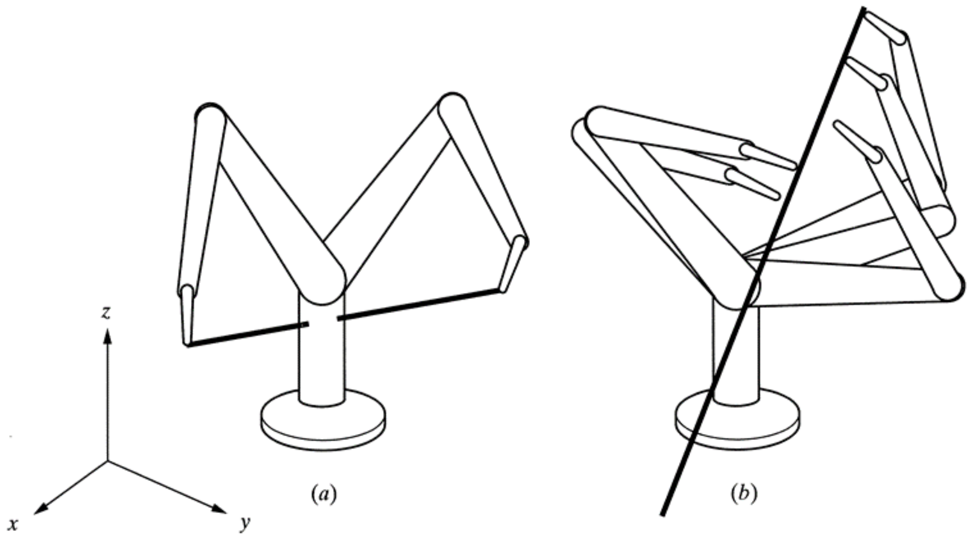


Figure2.3 Cartesian-space trajectory (a) The trajectory specified in Cartesian coordinates may force the robot to run into itself, and (b) the trajectory may require a sudden change in the joint angles.[11]

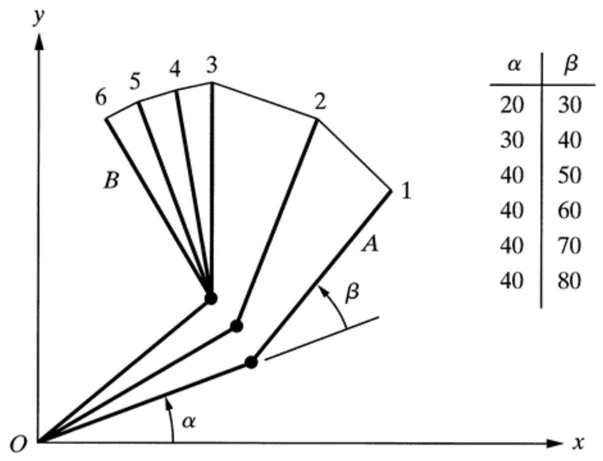
## II.4 Basics of Trajectory Planning:

• Let’s consider a simple 2 degree of freedom robot.

• We desire to move the robot from Point A to Point B.

• Let’s assume that both joints of the robot can move at the maximum rate of 10 degree/sec.

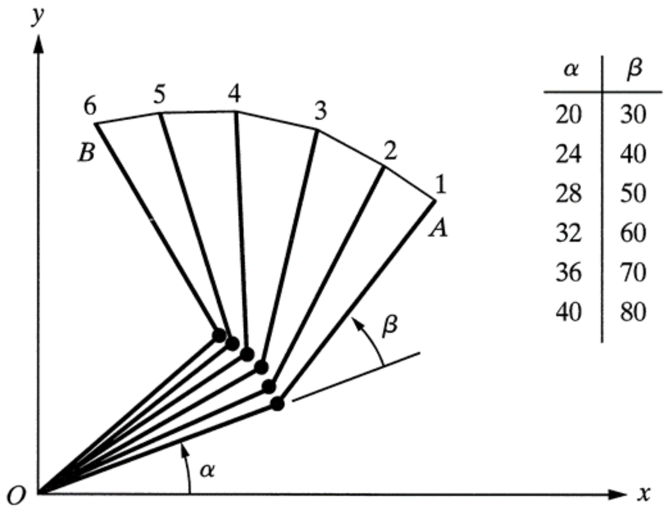
• Let’s assume that both joints of the robot can move at the maximum rate of 10 degree/sec. [11]



| α | β |
|---|---|
| 20 | 30 |
| 30 | 40 |
| 40 | 50 |
| 40 | 60 |
| 40 | 70 |
| 40 | 80 |

Figure2.4 Joint space nonnormalized movements of a robot with two degrees of freedom.[13]

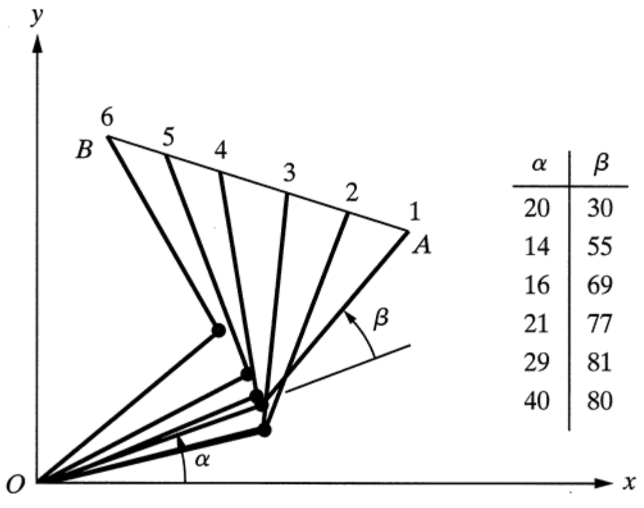
• Let’s assume that the motions of both joints are normalized by a <u>common factor</u> such that the joint with smaller motion will move proportionally slower and the both joints will start and stop their motion simultaneously.



| α | β |
|---|---|
| 20 | 30 |
| 24 | 40 |
| 28 | 50 |
| 32 | 60 |
| 36 | 70 |
| 40 | 80 |

Figure2.5 Joint-space, normalized movements of a robot with two degrees of freedom[13]

- Let’s assume that the robot’s hand follows a known path between point A to B with straight line.
- The simplest solution would be to draw a line between points A and B, so called interpolation.



| α | β |
|---|---|
| 20 | 30 |
| 14 | 55 |
| 16 | 69 |
| 21 | 77 |
| 29 | 81 |
| 40 | 80 |

Figure2.6 Cartesian-space movements of a two-degree-of-freedom robot.

- Let’s assume that the robot’s hand follows a known path between point A to B with straight line.
- The simplest solution would be to draw a line between points A and B, so called interpolation.

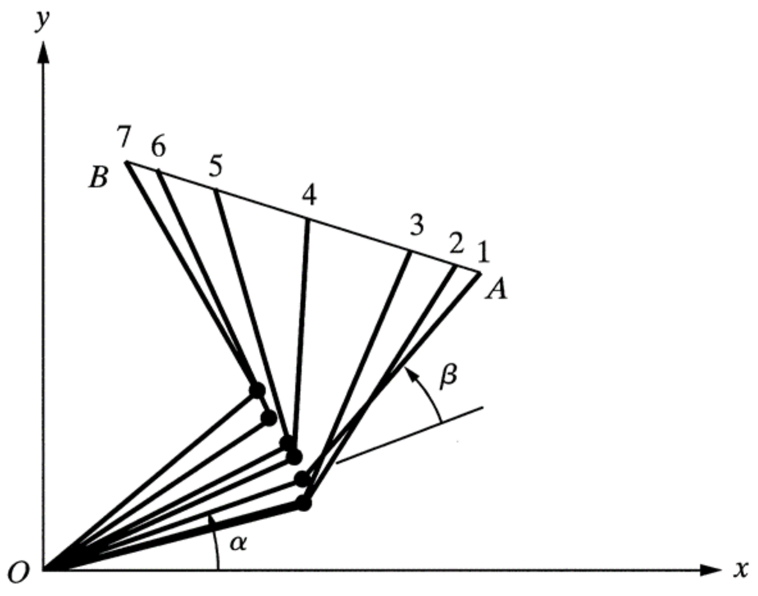


Figure2.7 Trajectory planning with an acceleration-deceleration regiment.

• Next level of trajectory planning is between multiple points for continuous movements.

• Stop-and-go motion create jerky motions with unnecessary stops.

**• Blend the two portions of the motion at point B.**

**• Specify two via point D and E before and after point B**

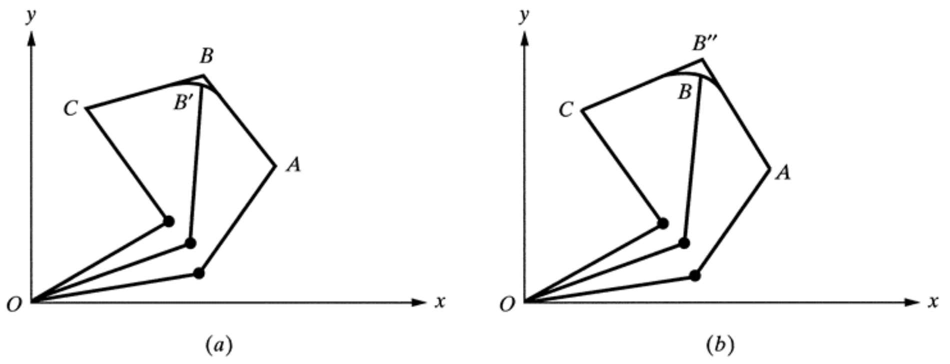


Figure2.8 Blending of different motion segments in a path

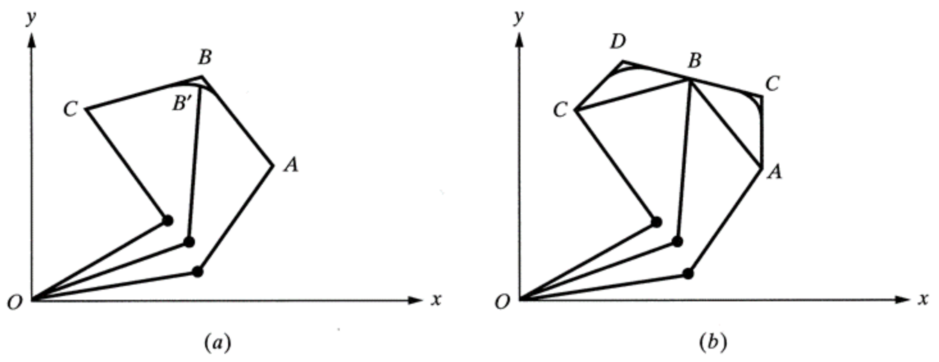


Figure2.9 An alternative scheme for ensuring that the robot will go through a specified point during blending of motion segments. Two via points D and E are picked such that point B will fall on the straight-line section of the segment ensuring that the robot will pass through point B.[13]

## II.5 Joint-Space Trajectory Planning:

### II.5.1 Third-Order Polynomial Trajectory Planning:

• How the motions of a robot can be planned in joint space with controlled characteristics.

• Polynomials of different orders

• Linear functions with parabolic blends

**• The initial location and orientation of the robot is known, and using the inverse kinematic equations, we find the final joint angles for the desired position and orientation.**

$$\theta(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3$$

$$\theta(t_i) = \theta_i$$
$$\theta(t_f) = \theta_f$$
$$\dot{\theta}(t_i) = 0$$
$$\dot{\theta}(t_f) = 0$$

- **Initial Condition**

$$\dot{\theta}(t) = c_1 + 2c_2 t + 3c_3 t^2$$

- **First derivative of the polynomial of equation**

$$\theta(t_i) = c_0 = \theta_i$$
$$\theta(t_f) = c_0 + c_1 t_f + c_2 t_f^2 + c_3 t_f^3$$
$$\dot{\theta}(t_i) = c_1 = 0$$
$$\dot{\theta}(t_f) = c_1 + 2c_2 t_f + 3c_3 t_f^2 = 0$$

- **Substituting the initial and final conditions**

## Example:

- It is desired to have the first joint of a six-axis robot go from initial angle of 30° to a final angle of 75° in 5 seconds. Using a third-order polynomial, calculate the joint angle at 1, 2 3, and 4 seconds.[15]

$$\theta(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3$$

$$\theta(0) = c_0 = 30$$

$$\dot{\theta}(0) = c_1 = 0$$

### II.5.2 Fifth-Order Polynomial Trajectory Planning:

• Specify the initial and ending accelerations for a segment.

• To use a fifth-order polynomial for planning a trajectory, the total number of boundary conditions is 6.[14]

• **Calculation of the coefficients of a fifth-order polynomial with position, velocity and an acceleration boundary conditions can be possible with below equations.**

$$\theta(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 + c_5 t^5$$
$$\dot{\theta}(t) = c_1 + 2c_2 t + 3c_3 t^2$$
$$\ddot{\theta}(t) = 2c_2 + 6c_3 t + 12c_4 t^2 + 20c_5 t^3$$

### II.5.3 Linear Segments with Parabolic Blends:

• Linear segment can be blended with parabolic sections at the beginning and the end of the motion segment, creating continuous position and velocity.

• Acceleration is constant for the parabolic sections, yielding a continuous velocity at the common points A and B.[15]

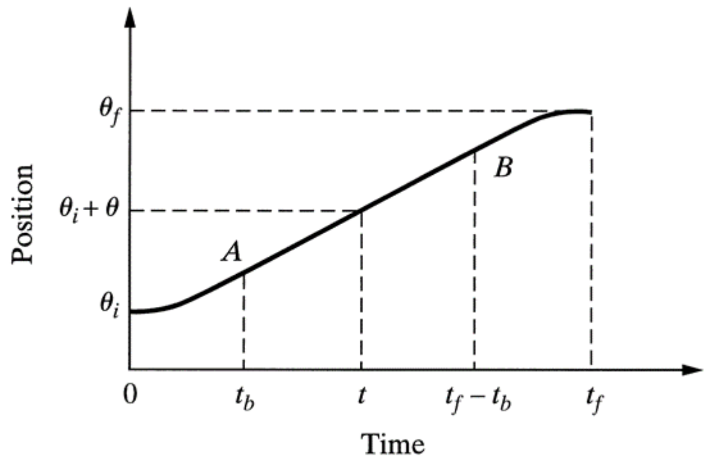


Figure2.10 Scheme for linear segments with parabolic blends.

$$\theta(t) = c_0 + c_1 t + \frac{1}{2} c_2 t^2$$
$$\dot{\theta}(t) = c_1 + c_2 t$$
$$\ddot{\theta}(t) = c_2$$

$$\Rightarrow$$

$$\theta(t) = \theta_i + \frac{1}{2} c_2 t^2$$
$$\dot{\theta}(t) = c_2 t$$
$$\ddot{\theta}(t) = c_2$$

### II.5.4 Linear Segments with Parabolic Blends and Via Points:

• The position of the robot at time $t_0$ is known and using the inverse kinematic equations of the robot, the joint angles at via points and at the end of the motion can be found.

• To blend the motion segments together, the boundary conditions of each point to calculate the coefficients of the parabolic segments is used. [14]

**• Maximum allowable accelerations should not be exceeded.**

### II.5.5 Higher Order Trajectories:

• Incorporating the initial and final boundary conditions together with this information enables us to use higher order polynomials in the below form, so that the trajectory will pass through all specified points.

$$\theta(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + \cdots\cdots + c_{n-1} t^{n-1} + c_n t^n$$

• It requires extensive calculation for each joint and higher order polynomial.

• Combinations of lower order polynomials for different segments of the trajectory and blending together to satisfy all required boundary conditions is required.

## II.6 Cartesian-Space Trajectories:

• Cartesian-space trajectories relate to the motions of a robot relative to the Cartesian reference frame.

• In Cartesian-space, the joint values must be repeatedly calculated through the inverse kinematic equations of the robot.

### • Computer Loop Algorithm

(1) Increment the time by t=t+Δt.

(2) Calculate the position and orientation of the hand based on the selected function for the trajectory.

(3) Calculate the joint values for the position and orientation through the inverse kinematic equations of the robot.

(4) Send the joint information to the controller.

(5) Go to the beginning of the loop.[15]

## II.7 The different methods of trajectory planning:

### II.7.1 Interpolation polynomial: Lagrange:

The coefficients a0, a1, . . ., an−1 of the unique interpolation polynomial ("Matrix Method")

$$f(x) = \sum_{k=0}^{n-1} a_k . x^k$$

of degree ≤ n − 1 which passes through the data points (x1 , y1 ), . . . ,(xn, yn) is given by the following

**system of linear equations:**

$$\underbrace{\begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^{n-1} \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix}}_{\vec{a}} = \underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{\vec{y}}.$$

**Note:** The above matrix A is called the Vandermonde matrix defined by. $x_1, x_2, \dots, x_n$
Its determinant is called the Vandermonde determinant:

$$V(x_1, x_2, \dots, x_n) = \prod_{1 \le i < j \le n} (x_j - x_i).$$

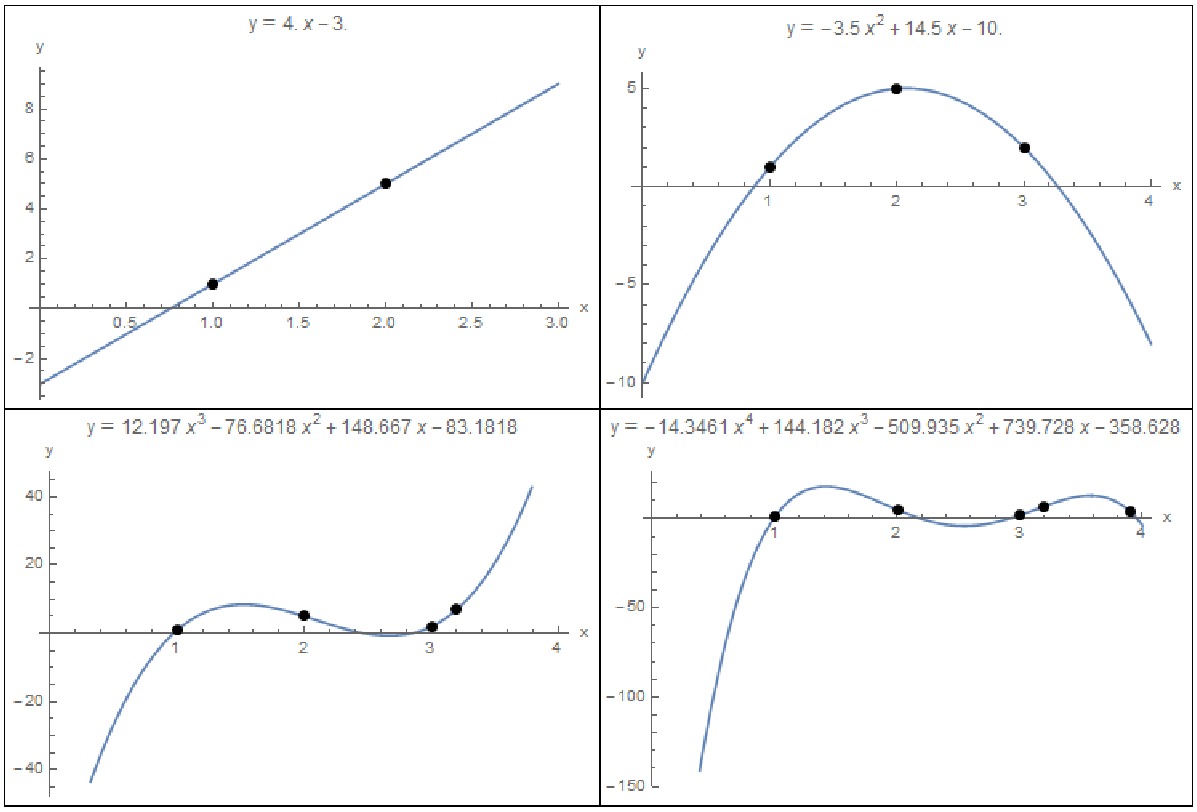


Figure2.11 interpolation polynomial Lagrange.

## Advantages and disadvantages:

➢ The advantage of Lagrange interpolation is that it is relatively simple and that it
is possible to connect a lot of points with a single polynomial function and not
several arcs of cubic or parables.

➢ The main disadvantage is that the curve can be completely aberrant.

➢ Indeed, if we try to interpolate a regular function like 1/1+ $x^2$, we would like the closer the points chosen on this curve the interpolation approaches the given function. Unfortunately, it is not the case "The more points we take the more the curve oscillates between these points". Interpolation de Lagrange is therefore not a very good method, and it is generally preferable to interpolate the points with several arcs of a lower degree than to connect them.

➢ This problem is very annoying and explains why the Lagrange interpolation is not very used.

➢ Another disadvantage is that the polynomial is of degree n, so if we have
a lot of points the manipulations can become heavy. It is also a
reason to prefer the interpolation by ends of cubic.[12]

### II.7.2 Bezier Curves:

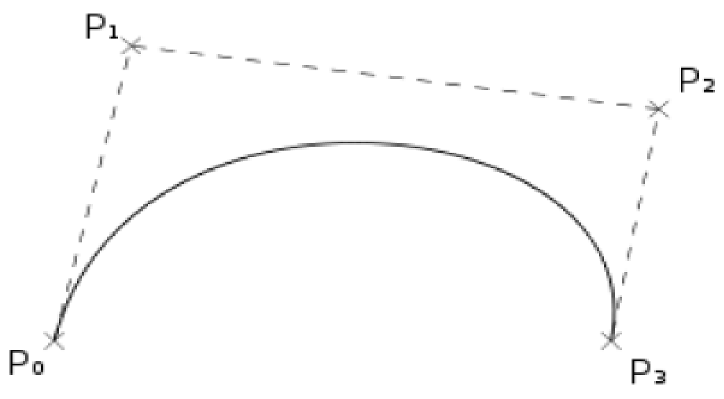
#### II.7.2.1 Introduction:

• A Bezier curve is a parametric curve frequently used in computer graphics and related fields. Generalizations of Bezier curves to higher dimensions are called Bezier surfaces, of which the Bezier triangle is a special case.

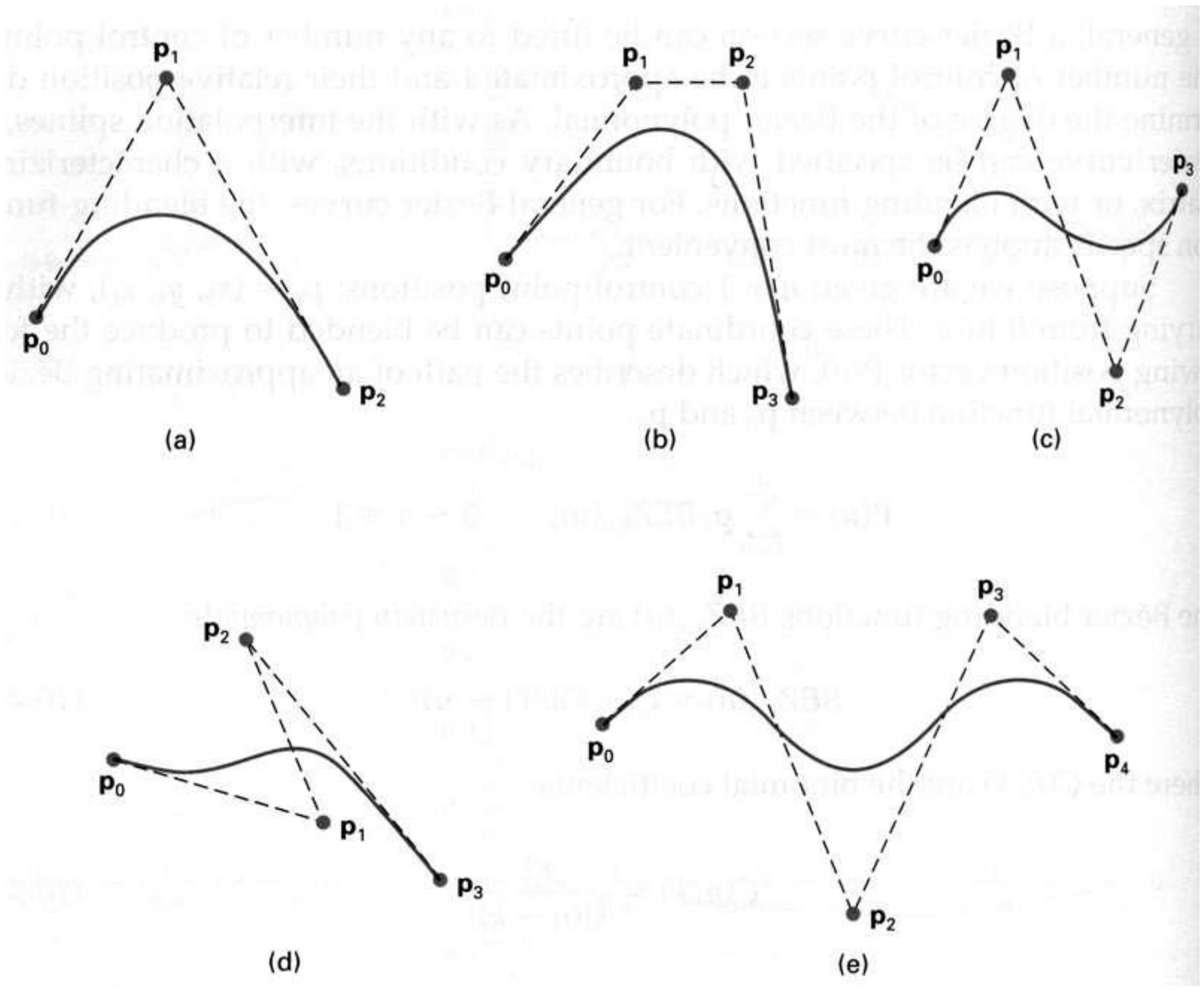
• Paths are not bound by the limits of rasterized images and are intuitive to modify. Bezier curves are also used in animation as a tool to control motion.

#### II.7.2.2 Definition:

A Bezier curve is defined by its order (l,inear, quadratic, cubic, etc.) and a set of *control points* **P**0 through **P***n*, the number *n* of which depends on the order (*n* = 2 for linear, 3 for quadratic, etc.). The first and last control points are always the end points of the curve; however, the intermediate control points (if any) generally do not lie on the curve. [12]



**SOME MORE BEZIER CURVES:**



### II.7.2.3 Properties of Bezier Curves:

•The first and last control points are interpolated.

•The tangent to the curve at the first control point is along the line joining the first and second control points.

•The tangent at the last control point is along the line joining the second last and last control points.

• The curve lies entirely within the convex hull of its control points.

•The Bernstein polynomials (the basic functions) sum to 1 and are everywhere positive.

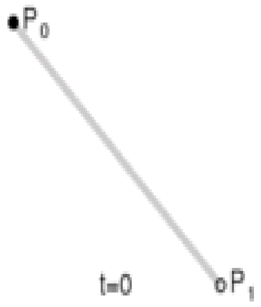
•They can be rendered in many ways.

**E.g.**: Convert to line segments with a subdivision algorithm.[9]

### II.7.2.4 Constructing Bezier Curves:

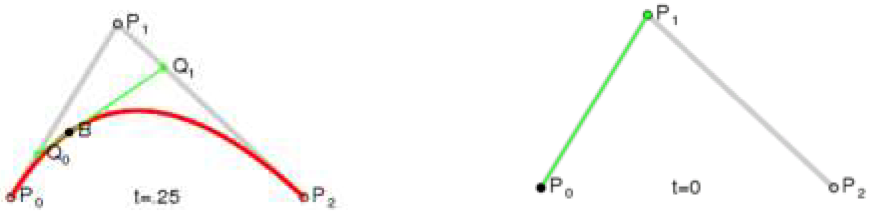
#### *II.7.2.4.1 Linear curves:*

The *t* in the function for a linear Bezier curve can be thought of as describing how far **B**(*t*) is from **P**0 to **P**1. For example, when *t=0.25*, **B**(*t*) is one quarter of the way from point **P**0 to **P**1. As *t* varies from 0 to 1, **B**(*t*) describes a straight line from **P**0 to **P**1.



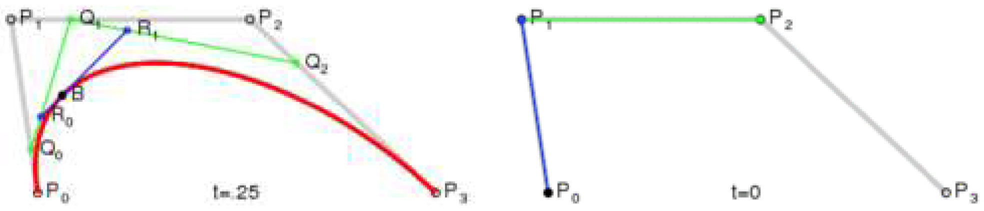
#### II.7.2.4.2 Quadratic curves:

- For quadratic Bezier curves one can construct intermediate points Q0 and Q1 such that as t varies from 0 to 1.
- Point Q0 varies from P0 to P1 and describes a linear Bezier curve.
- Point Q1 varies from P1 to P2 and describes a linear Bezier curve.
- Point B(t) varies from Q0 to Q1 and describes a quadratic Bezier curve.[10]



#### II.7.2.4.3 Higher-order curves:

For higher-order curves one needs correspondingly more intermediate points. For cubic curves one can construct intermediate points **Q**0, **Q**1, and **Q**2 that describe linear Bezier curves, and points **R**0 & **R**1 that describe quadratic Bezier.



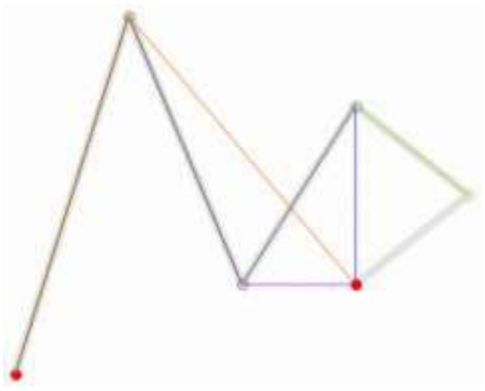
#### II.7.2.4.4 Fourth-order curves:

One can construct intermediate points **Q**0, **Q**1, **Q**2 &

**Q**3 that describe linear Bezier curves, points **R**0, **R**1 & **R**2 that describe quadratic Bezier curves, and points **S**0 & **S**1 that describe cubic Bezier curves.

#### II.7.2.4.5 Fifth-order curves:

One can construct similar intermediate points.



### II.7.2.5 Application:

• Bezier curves are used in the time domain, particularly in animation and interface design, e.g., a Bezier curve can be used to specify the velocity over time of an object such as an icon moving from A to B, rather than simply moving at a fixed number of pixels per step.

• Bezier curves are widely used in computer graphics to model smooth curves.

• Quadratic and cubic Bezier curves are most common; higher degree curves are more expensive to evaluate. When more complex shapes are needed, low order Bezier curves are patched together.[10]

## II.7.3 Splines:

Modelling segmental

- Piecewise polynomial segments
  - Modelling the curve locally by a segment
  - The segments are joined to get the global curve
- Name derives from shipbuilding
  - Splines are stripes of metal fixed at several points. They have the same smoothness as cubic B-Splines
- Global parameter *u* and for each segment $s_i$ a local parameter *t*

### II.7.3.1 Splines – parametrization:

- Globally parametrized over a sequence of nodes $u_i$
    - Each segment $s_i$ is locally parametrized over $[u_i, u_{i+1}]$

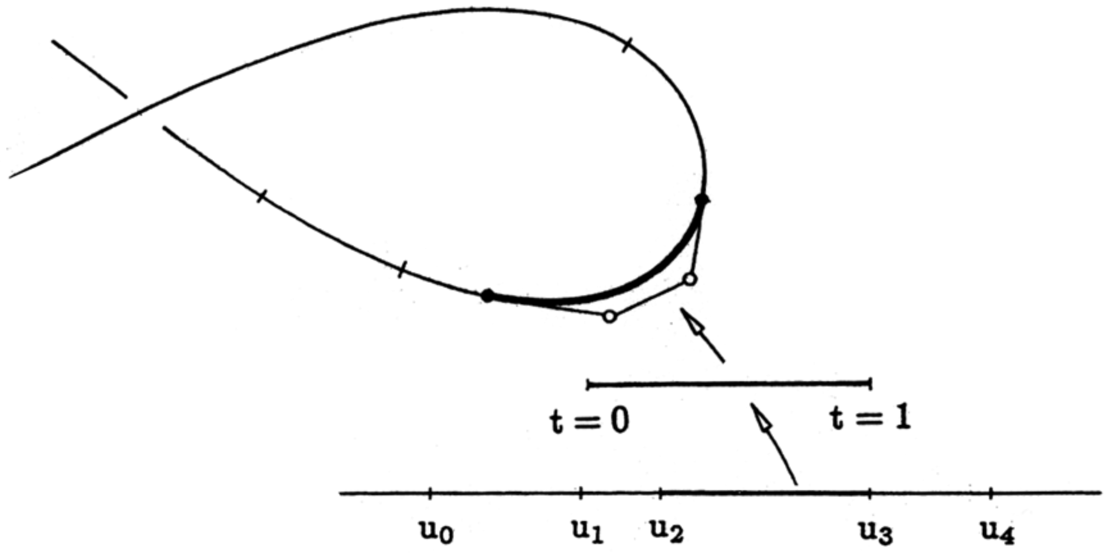
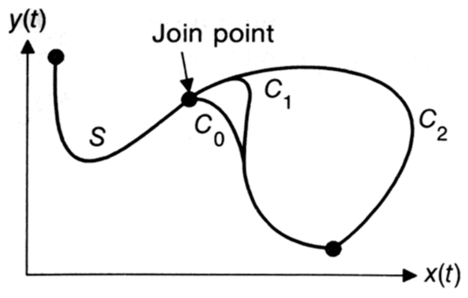


Fig. II.12 Globally parametrized over a sequence of nodes ui Splines

- Uniform splines
    - Uniformly distributed sequence over range of global parameter *u*
- Nonuniform splines
    - Sequence is not uniformly distributed
- Segments $s_i$ „live''over $[u_i, u_{i+1}]$ with local parameter
- $t = (u – u_i) / (u_{i+1} – u_{i–1})$

### II.7.3.2 Splines – smoothness:

- What does smoothness mean?
    - Graphically clear: No sharp bends
    - Mathematical abstraction: Order of continuous derivability
    - Only nodes and the corresponding knot-points are relevant
        - other points are infinitely often derivable



- **C$^0$-continuity**
  - The Segmentes meet at the nodes
- **C$^1$-continuity**
  - Curve must be continuously derivable after the global parameter *u* one time
  - The „speed «in respect to global parameter must be the same in the joint both segments
- **C$^2$-continuity**
  - Curve must be continuously derivable 2 times
  - The „acceleration «must be equal in the joint

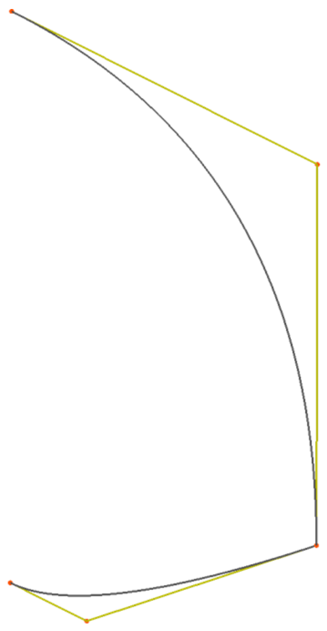


Figure2.13 C$^0$-continuous Spline

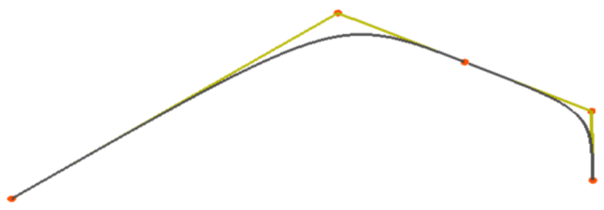


Figure2.14 C$^1$-continuous Spline [6]

- Geometric smoothness: **$G^1$-continuity**
- In every joint there exists a unique tangent
  - I.e. tangent vectors need not be equal in magnitude, only direction
- Does 't take global parametrization into account
- An example for **$G^1$**-continuous curve that is **not $C^1$**-continuous

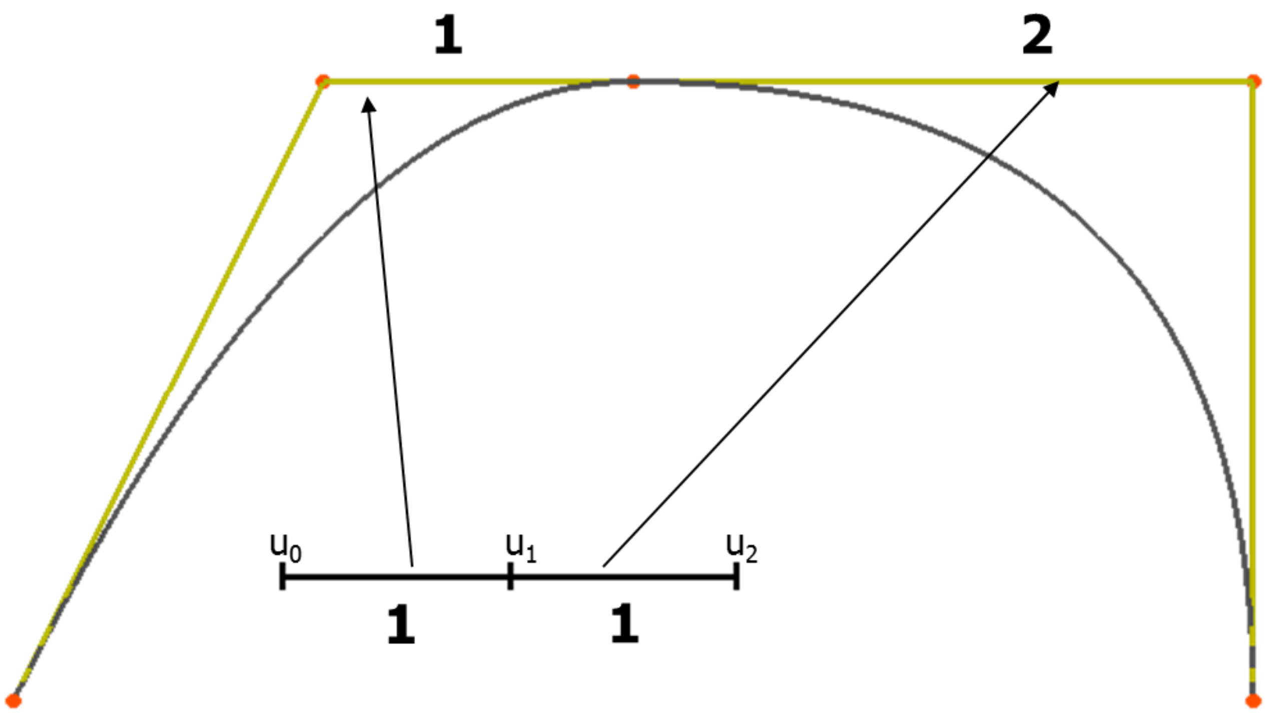
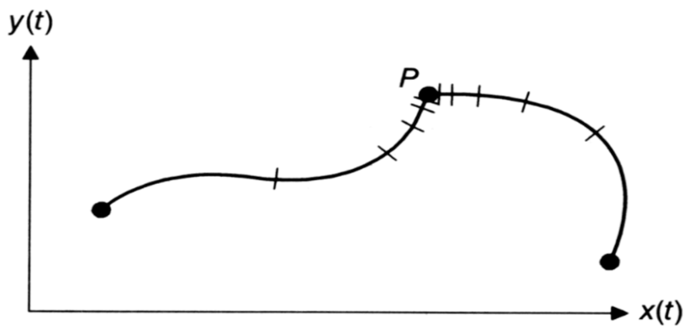
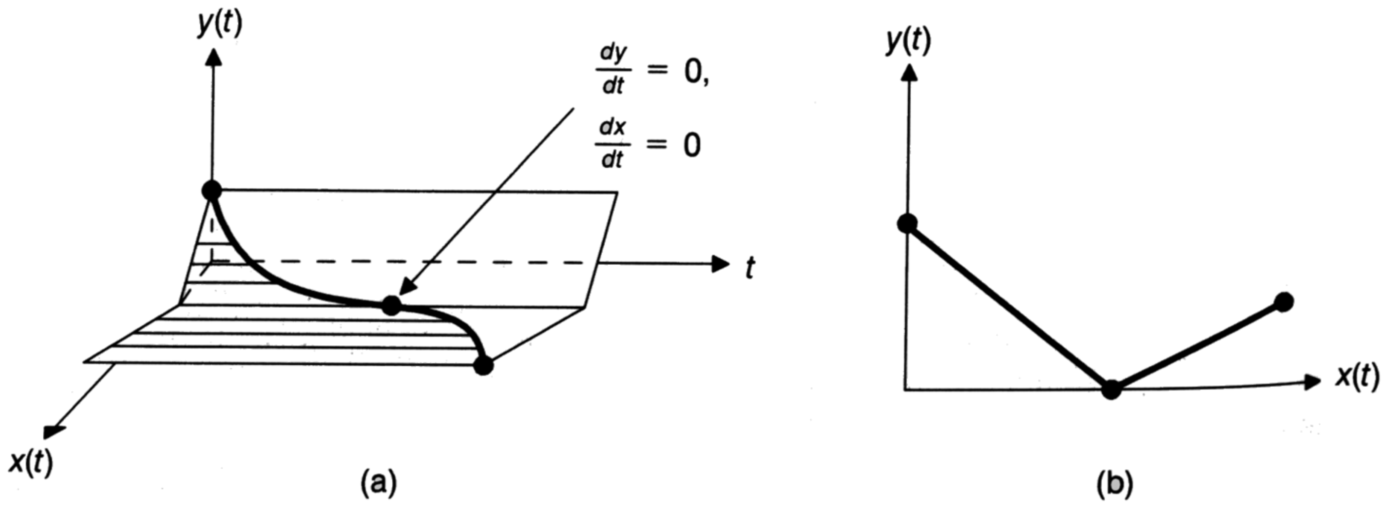


Figure2.15 G1-continuity "Geometric smoothness"

- In general
  - **$C^x$-continuity $\Rightarrow$ $G^x$-continuity**
  - But **not: $G^x$-Continuity $\Rightarrow$ $C^x$-continuity**
- Special case: Stationary point
  - All derivations are zero
  - Then **$C^x$-continuity $\Rightarrow$ $G^x$-continuity** doesn't hold





### II.7.4 B-Splines:

- Spline consisting of Bezier curves
- Depending on the degree *n* of the Bezier segments various orders of continuity possible
  - In general, up to $C^{n-1}$-continuouity possible

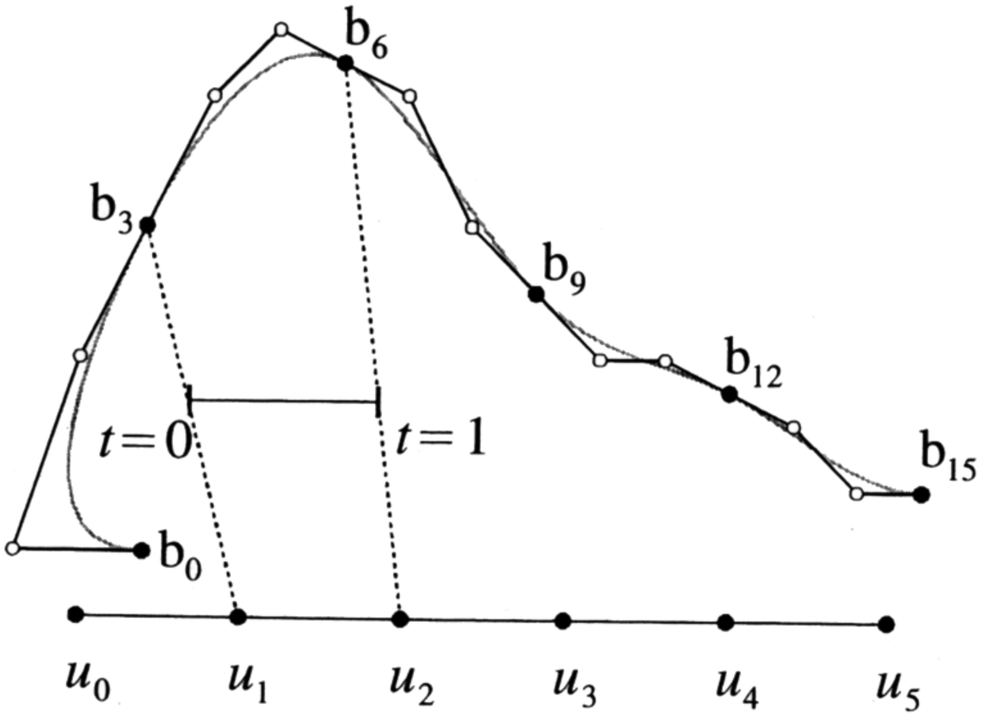


Figure2.16 B-Splines curve

### II.7.4.1 B-Splines – C$^1$-continuity:

- C$^1$-continuity at knot $b_i$
  - Tangent vectors must be equal
  - Bezier: Tangent vectors in $b_i$
    - $s_0$ - points from $b_{i-1}$ to $b_i$
    - $s_1$ - points from $b_i$ to $b_{i+1}$
  - This means $b_{i-1}$, $b_i$ *and* $b_{i+1}$ must be collinear

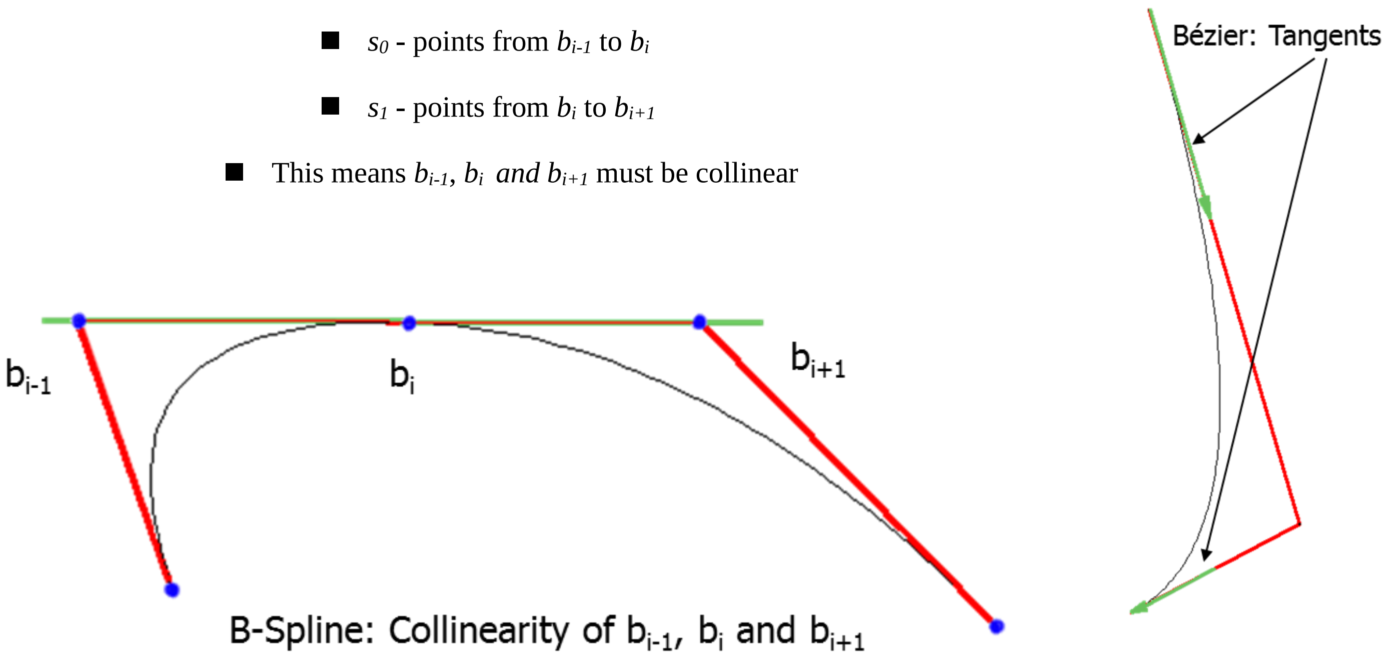
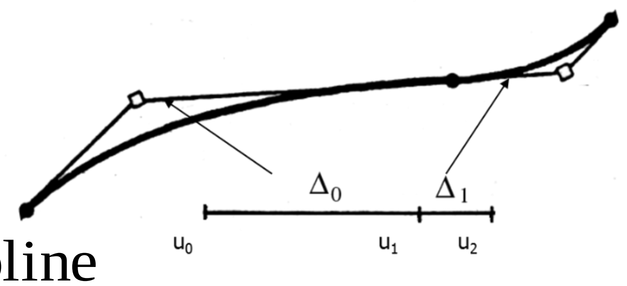


Figure2.17 B-Splines – C1-continuity

- But this alone isn't sufficient
  - Furthermore $b_{i-1}$, $b_i$ *and* $b_{i+1}$ must satisfy a specific ratio

$$\frac{|b_i - b_{i-1}|}{|b_{i+1} - b_i|} = \frac{u_1 - u_0}{u_2 - u_1} =: \frac{\Delta_0}{\Delta_1}$$

$$\frac{|b_i - b_{i-1}|}{|b_{i+1} - b_i|} = \frac{1}{1} \quad \text{(for uniform B-Spline}$$



- Why this?
- Because we are interested in derivability for the **global** parameter

  - Collinearity says that the „speed'' has the same direction, but not necessarily magnitude
- Mathematical derivation

$$\frac{d}{du}s(u)=\frac{d}{du}t(u)\cdot\frac{d}{dt}s_j(t)=\frac{1}{\Delta_j}\cdot\frac{d}{dt}s_j(t),\ \text{with}\ \Delta_j=u_{j+1}-u_j,\ t(u)=\frac{u-u_j}{\Delta_j},\ u\in[u_j,u_{j+1}]$$

$$\text{So we have to check}\ \frac{1}{\Delta_0}\frac{d}{dt}s_0(1)=\frac{1}{\Delta_1}\frac{d}{dt}s_1(0)$$

$$\Leftrightarrow \Delta_1(b_n-b_{n+1})=\Delta_0(b_{n+1}-b_n),\text{since}\ \frac{d}{dt}s_0(1)=n(b_n-b_{n-1})\ \text{and}\ s_1(0)=n(b_{n+1}-b_n)$$

- Given: Two Bezier curves
- How to test $C^1$-continuity
  - Consider the joint points of the spline and its direct neighbors
    - When no joint it isn't even $C^0$-continuous
  - Are they collinear?
    - No, then the spline is not $C^1$-continuous
  - Ok, they are collinear
    - Then check the ratio of the distance from the knot point neighbors to the knot point. It must be same ratio as of the corresponding nodes

### II.7.4.2 B-Splines – $C^2$-continuity:

- $C^2$-continuity at knot $b_i$
  - Precondition: $C^1$-continuity
- Nodes $b_{n-2}$, $b_{n-1}$, $b_n$ and $b_n$, $b_{n+1}$, $b_{n+2}$ must describe a unique global quadratic Bzier curve
  - So, there must exist a (unique) point $d$, so that $b_{n-2}$, $d$, $b_{n+2}$ describe this curve
- The point $d$ must fulfill two conditions

$$b_{n-1}=(1-t)\,b_{n-2}+t\,d$$
$$b_{n+1}=(1-t)\,d+t\,b_{n+2}$$

$$\text{at which}\ t=\frac{\Delta_0}{\Delta_0+\Delta_1}$$

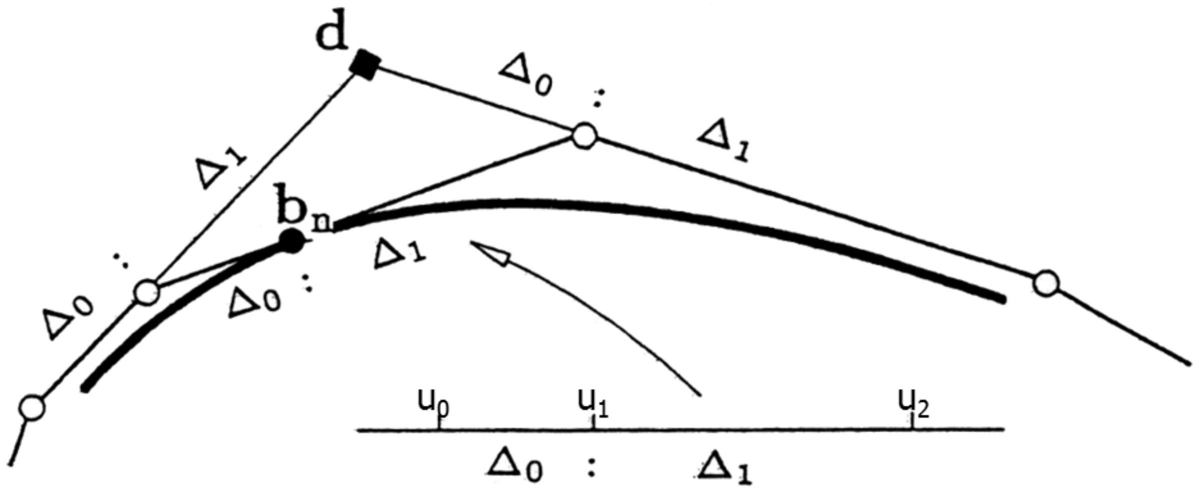


Figure2.18 B-Splines – C2-continuity

- How to test for $C^2$-continuity
- First of all, test for $C^1$-continuity
    - If it is not, it cannot be $C^2$-continuous
- Check for uniqueness of point *d*
    - Extrapolate $d_-$ from $b_{n-2}$ and $b_{n-1}$
    - Extrapolate $d_+$ from $b_{n+2}$ and $b_{n+1}$
        - $d_-$ and $d_+$ must be equal
            - In Practice specify a tolerance

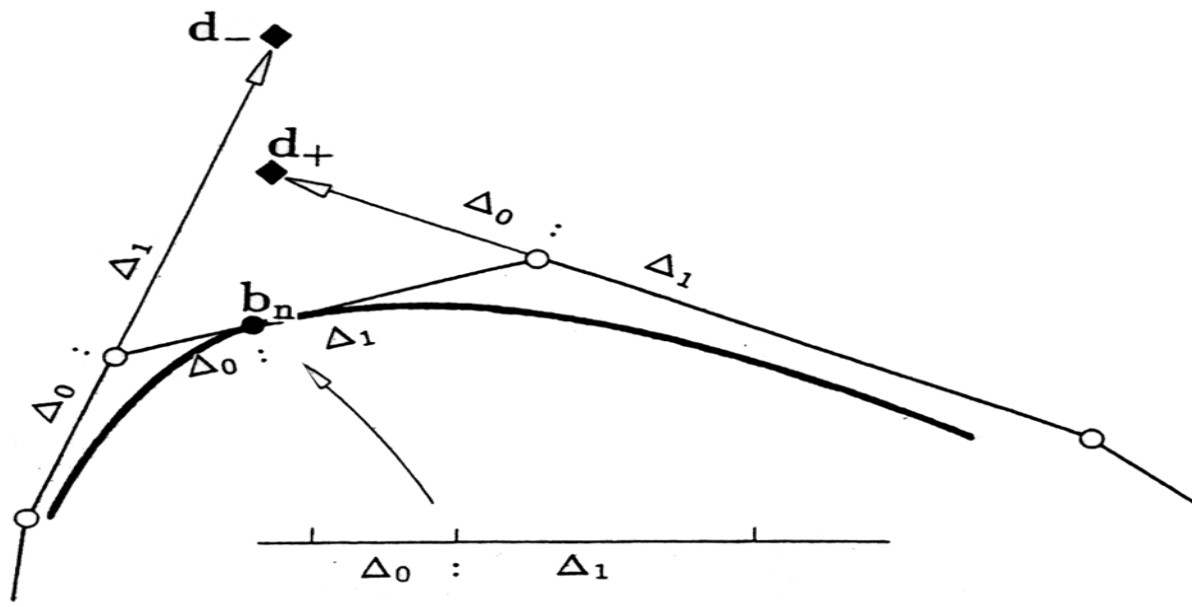


Figure2.19 B-Splines – C2-continuity

### II.7.4.3 Quadratic B-Splines:

- Construction of a $C^1$-continuous quadratic B-Spline
- Different approach
  - Per definition we want a $C^1$-continuous curve
  - We do **not** want to test two segments for $C^1$-continuity anymore
- Given are *n*+1 control points $d_0,\dots,d_n$ spanning the *control polygon* of the B-Spline
- Constructing Bezier control points for a $C^1$-continuous quadratic B-Spline
- Construction of Bezier polygon:

$$b_0 := d_0 \qquad b_{2(n-2)} := d_n \qquad b_{2i+1} := d_i,\ \ i=0,\dots,n-3$$

$$b_{2i} := \frac{\Delta_i}{\Delta_{i-1}+\Delta_i} b_{2i-1} + \frac{\Delta_{i-1}}{\Delta_{i-1}+\Delta_i} b_{2i+1} \qquad\qquad b_{2i} = \frac{1}{2} b_{2i-1} + \frac{1}{2} b_{2i+1}$$ For uniform B-Splines

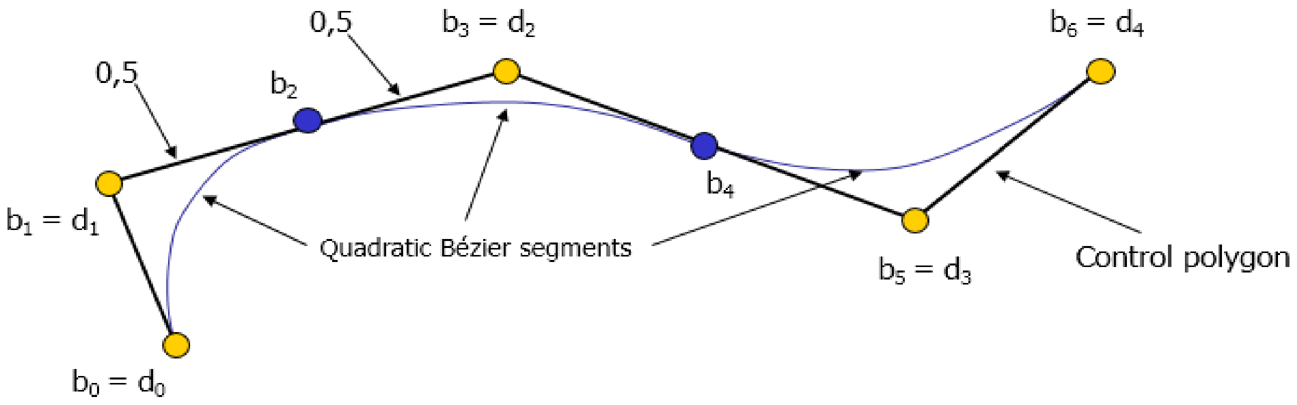


Figure2.20 Construction of a uniform quadratic $C^1$-continuous B-Spline with n=5

### II.7.4.4 Cubic B-Splines:

- Construction of a $C^2$-continuous cubic B-Spline
  - As for quadratic B-Splines we construct the Bezier control points $b_i$ from the B-Spline polygon spanned by $d_i$
- Given are *n*+1 control points $d_0,\dots,d_n$ again spanning the *control polygon* of the B-Spline
- Constructing Bezier control points for a $C^2$-continuous cubic B-Spline

- Formulas for uniform case (nonuniform version is too ugly ;)

$$\text{Left end:}\, b_0 := d_0,\quad b_1 := d_1,\quad b_2 := \frac{1}{2} d_1 + \frac{1}{2} d_2$$

$$\text{Right end:}\, b_{3(n-2)} := d_n,\quad b_{3(n-2)-1} := d_{n-1},\quad b_{3(n-2)-2} := \frac{1}{2} d_{n-2} + \frac{1}{2} d_{n-1}$$

$$b_{3i-2} := \frac{2}{3} d_{i-1} + \frac{1}{3} d_i$$

$$b_{3i-1} := \frac{1}{3} d_{i-1} + \frac{2}{3} d_i$$

$$b_{3i} := \frac{1}{2} b_{3i-1} + \frac{1}{2} b_{3i+1} \qquad \text{(Remember } C^1\text{-condition)}$$

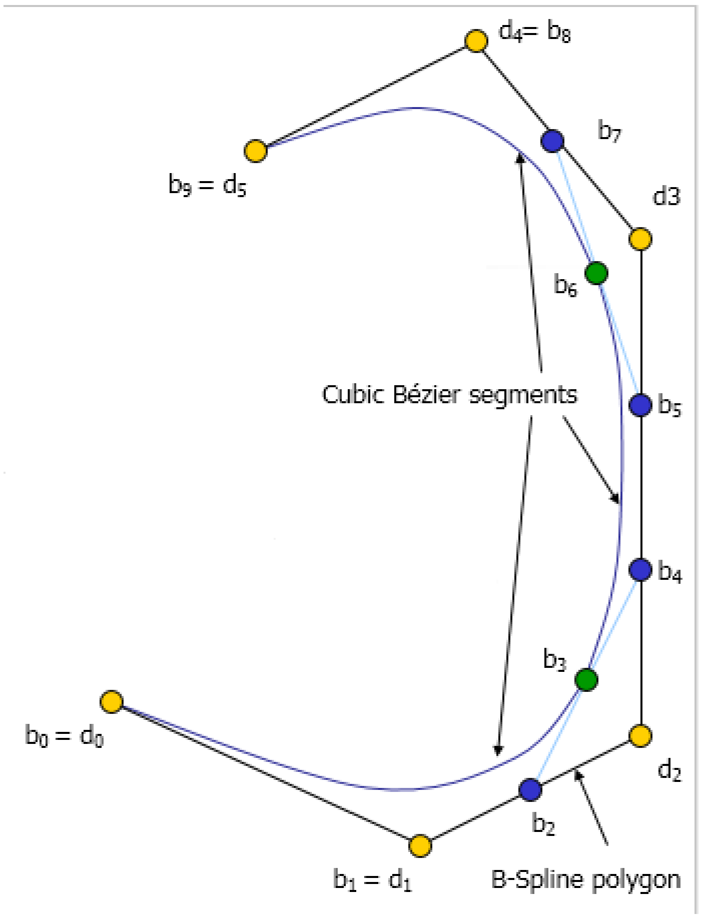


Figure2.21 Construction of a uniform C2-continuous cubic B-Spline

### II.7.4.5 Cubic B-Splines – storage:

- A little off-topic aspect: How to store a B-Spline
  - For example, it could be approximated linearly, and these linear segments could be saved
    - Obviously inefficient
  - All Bezier control points could be saved
    - It’s more efficient but not perfect
- So, what to save?
  - Answer: Only the $d_i$ need to be saved
  - Curve can be reconstructed as shown before
- The advantages are obvious
  - No loss of data. The curve can be *exactly* reconstructed
    - No problems with rounding errors
  - Little storage space needed
  - The storage is distinct when low-level storage information is known
  - Theoretical interchangeability of file format saving B-Splines
    - Practice is evidently different

### II.7.4.6 B-Splines – basis:

- What does the „B in the name stand for?
  - It does **not** stand for „Bezier’’ as you may guess, but for „Basis “
    - Since there exists a basis representation for the spline in contrary to natural splines
  - The basic functions are called the *minimum support splines*

- **Definition** of B-Splines via basis splines

$$s(u) := \sum_{i=0}^{N+n-1} d_i \cdot N_i^n(u), \quad \text{where } N_i^n(u) \text{ are the basis spline}$$

- **Definition** of the basis splines

- **Global** over the whole domain of *u*
- So that $N_i^n(u_i) = 1$
- $C^{n-1}$-smoothness is assured
- The support is minimal
  - That is the range in which it doesn't vanish

- Ok but how does this look like?

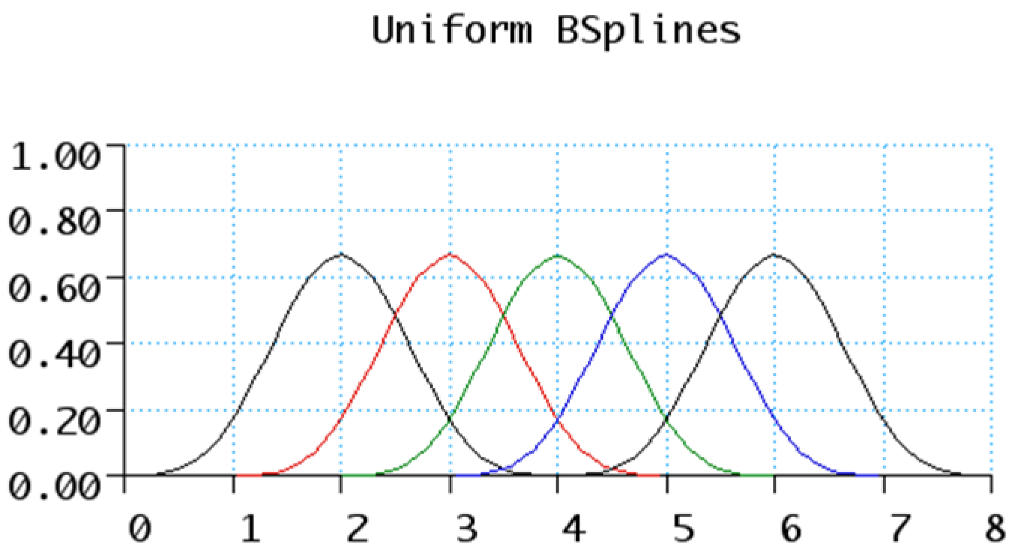


Figure2.22 The basis splines over the full domain of u

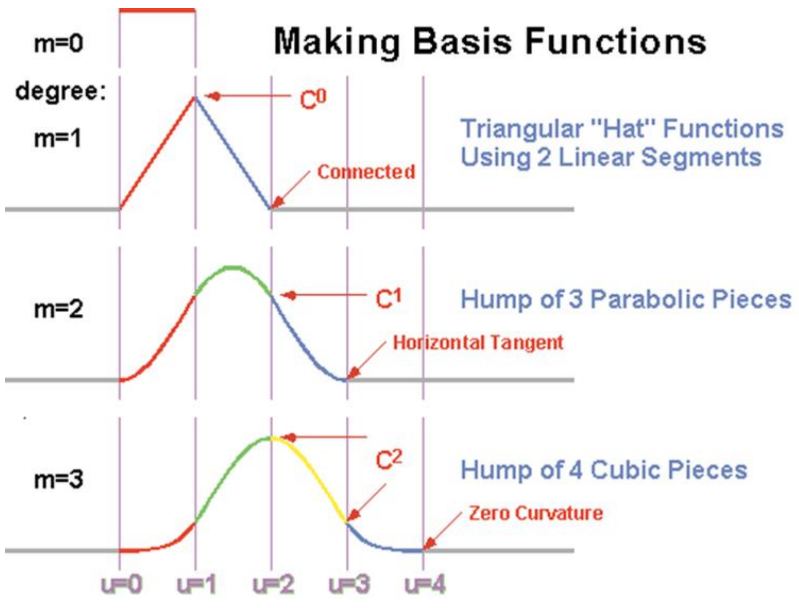


Figure2.23 basis functions

### II.7.4.7 B-Splines – characteristics:

- Invariant under affine transformations
    - You can scale, rotate and translate the curve
        - I.e. the control points and get the same result as doing so with all the points on the curve
- Interpolation of end points
- Excellent locality
    - Change of one control points affects at most *n+1* segment
- Lies within convex closure of the control points
    - Stricter than for Bezier curve: Lies within the union of the convex closures of all segments

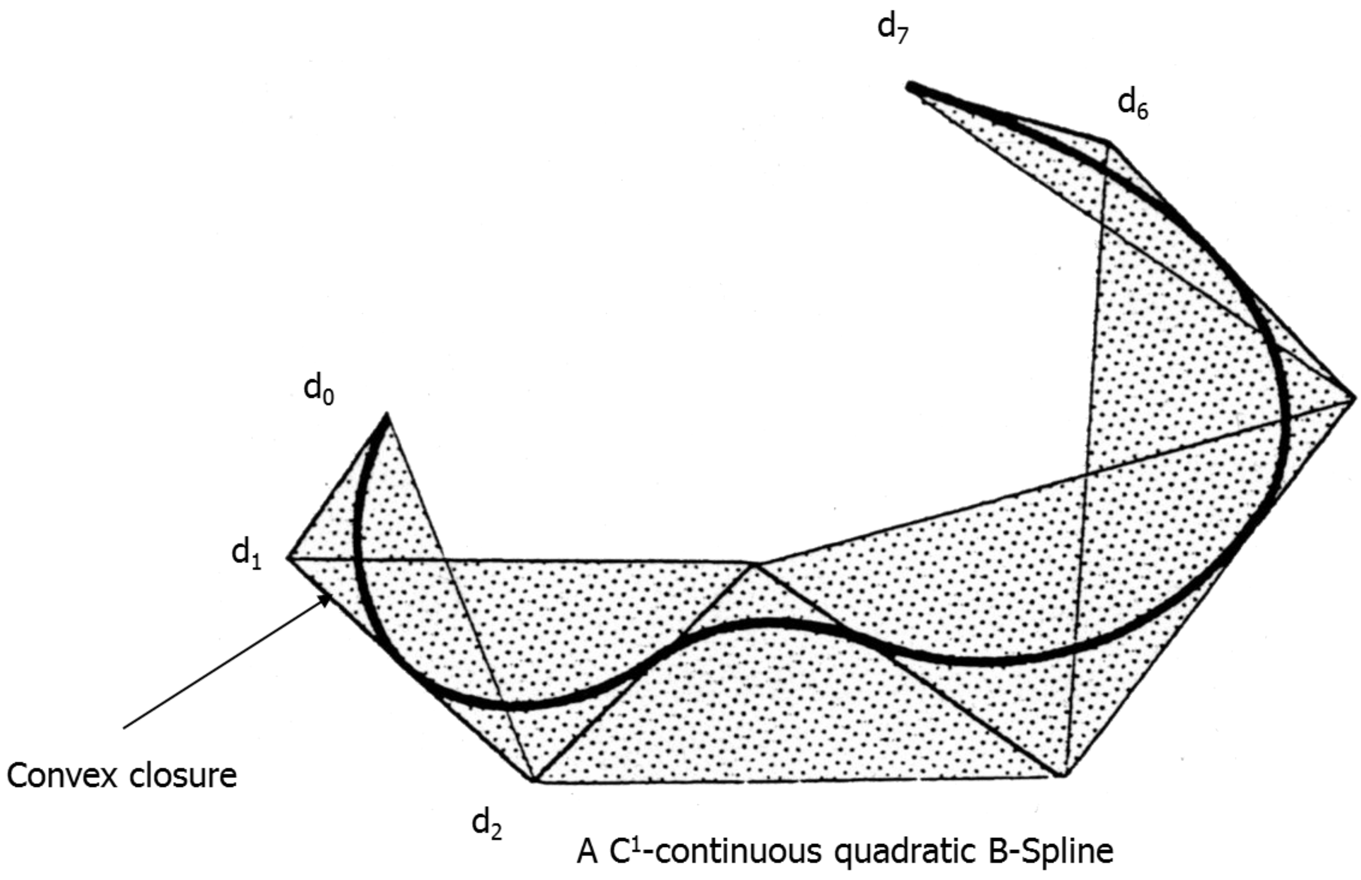


Figure2.24 B-Splines – characteristics

- Modelling of straight lines possible
- Even if all local segments are planar, modelling of true 3D curves is possible
  - E.g. quadratic B-Splines are planar in each segment, but the global curve may be true 3D
- The degree of the global curve doesn't depend on the number of points Efficient for modelling curves with many points

## II.7.5 Rational Bezier curves:

An even more powerful approach

- One major disadvantage of Bezier curves is that they are **not** invariant under projection
  - This means when perspectival projecting a 3D Bezier curve on the screen the result is no longer a Bezier curve
  - For CAGD applications 3D curves are often projected to be displayed on 2D screens
  - When projected, the original nonrational curve results in a rational curve
- This leads us to rational Bezier curves
  - They are invariant under projective transformation
- **Definition** of a rational Bezier curve

$$x(t) := \frac{\sum_{i=0}^{n} w_i \cdot b_i \cdot B_i^n(t)}{\sum_{i=0}^{n} w_i \cdot B_i^n(t)}$$

- As before the $b_i$ are the control points
- Additionally, every point $b_i$ has a weight $w_i$
- You can think of a rational Bezier curve of the projection of a nonrational one from 4D space to the hyper-plane w=1
- The control polyhedron of this is spanned by the 4D points ($b_i \cdot w_i$ , $w_i$)
- The $b_i$ are the projection of the $b_i \cdot w_i$ to the hyperplane $w$=1
- Let's visualize this for a 2D rational Bezier curve as a projected Bezier curve in 3D to plane $z$=1

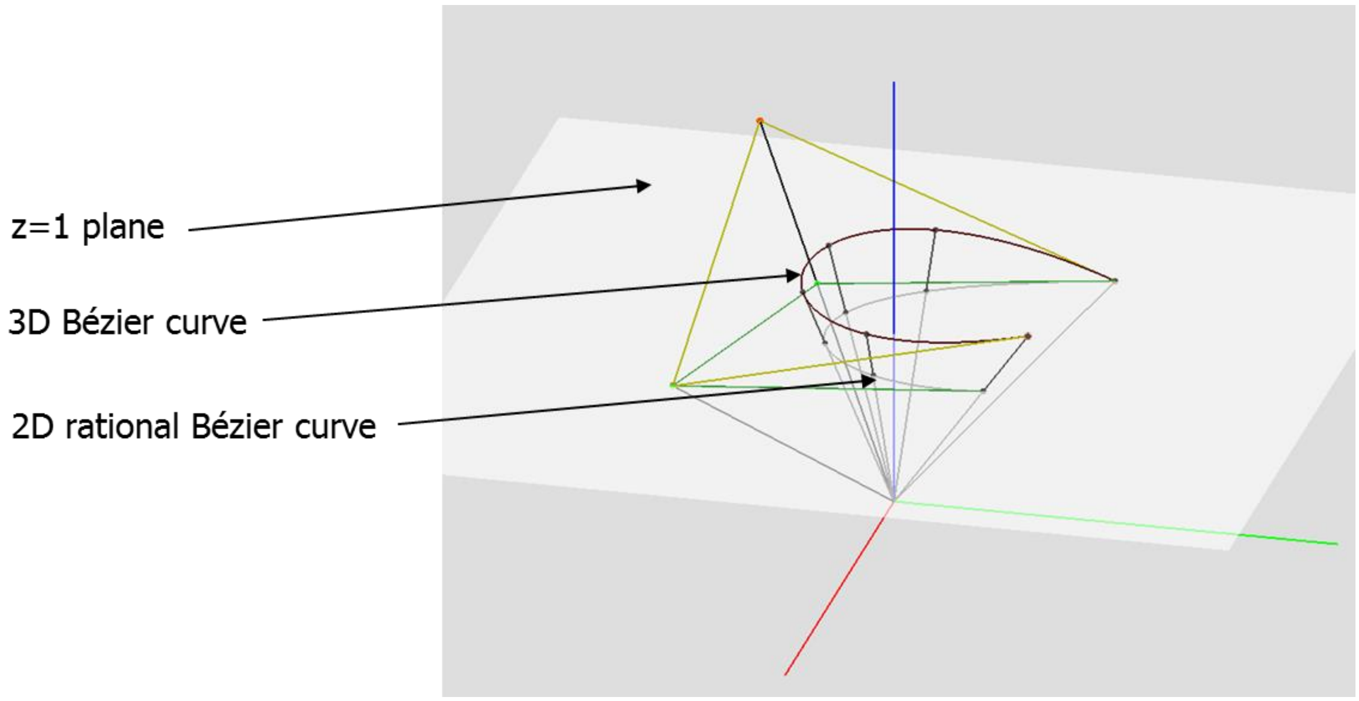


Figure2.25 rational Bezier curve

- The weights give a further degree of liberty to the designer

Changing the weight differs from moving the control point

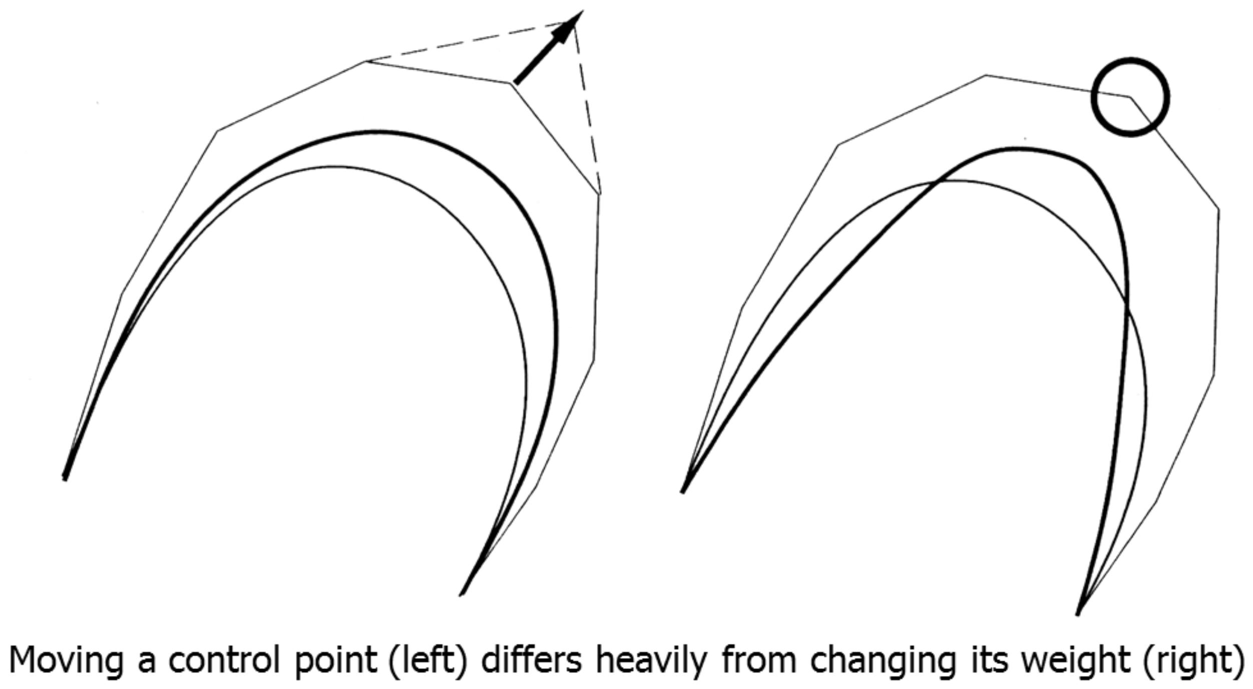

Moving a control point (left) differs heavily from changing its weight (right)

Figure2.26 Changing the weight differs from moving the control point

- Because they are invariant under projection the projected curve can be modeled further
  - E.g. a car designer may change the projected car body
- Bezier curves are a subset of the rational Bezier curves
  - Special case: Equal weights

### II.7.6 NURBS:

Puzzling with rational curves

- **N**on-Uniform **R**ational **B-S**plines
- B-Splines with rational Bezier curves
  - Since 3D rational Bezier curves can be thought of as nonrational in 4D, all results of normal B-Splines apply
  - Especially the continuity conditions are exactly identical, but with 4D control points
  - This implies that their construction is mathematical identical to that of B-Splines (but as well in 4D)
- Note: NURBS can be defined as well as B-Splines with Basis Splines

$$x(t) := \frac{\sum_{i=0}^{N+n-1} w_i \cdot b_i \cdot N_i^n(t)}{\sum_{i=0}^{N+n-1} w_i \cdot N_i^n(t)}$$

#### II.7.6.1 NURBS – characteristics:

- All conic sections (a circle for example) can be modelled exactly

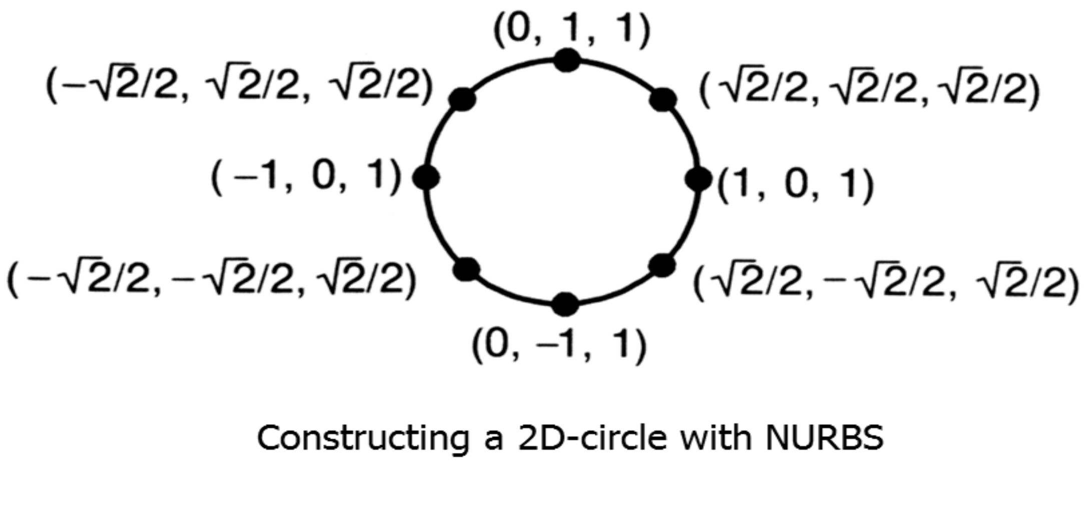


Figure2. II.27 Circle of NURBS

- It inherits all advantages of B-Splines (excellent locality, ...) while extending the liberty of modelling
- Therefore, today NURBS are standard for modelling
- The next logical step is to model not only curves but surfaces
    - This will be discussed in the next lecture

## II.8 Conclusion:

In this chapter, we have presented in a general way the subject of trajectory planning in robotics. After having detailed the representations of Bézier and the different B-Splines and at the end the representation of NURBS, then we gave a small comparison between them. In the next chapter, we will deal with examples of the planning of the trajectories of manipulator robots with two degrees of freedom of the revolute type (2R) and of three degrees of freedom of the revolute type (3R) by minimizing an objective function representing the time of the operation performed by the robot.

# Chapter III

# Results and Interpretations

## III.1 Introduction:

Trajectory planning is an essential part of the software dedicated to the operation of a manipulator robot. Generally, it is not enough to settle for the first feasible solution, but the current demands on robots make it necessary to seek among the most suitable feasible solutions or the optimal solution. In the field of robotics, several criteria have been used to assess the optimal solution (minimum time, minimization of the efforts provided by the actuators, minimization of jerk, etc.). In addition to these criteria, the trajectory obtained must meet the constraints imposed by the dynamics of the robot, the limits on the torques provided by the actuators in order to avoid overloading, the limits on positions, velocities and joint accelerations.

In this chapter we will do a detailed study of a method of planning an optimal trajectory for robot manipulator with two & three degrees of freedom executing complex tasks, this method aims to minimize the transfer time of the terminal organ from an initial point to an end point passing through intermediate points such as tasks of (continuous welding, polishing, or painting)

Initially, we will start by generating the trajectory by the cubic spline functions. which will allow us to have the control points to optimize. The analytical expression of the gradient of the cost function with respect to these control points allows the optimization procedure to easily converge to the minimum and makes it more robust. We will thus have transformed the problem of finding an optimal command which minimizes the criterion into an optimization problem.

## III.2 Description of the MATLAB Fmincon function (SQP):

*Fmincon* is the abbreviation for "*Find Minimum of Constrained Non-Linear Multivariable Function*" or find the minimum of a multivariable function with nonlinear constraint.

The goal of this function is to minimize a function ***f*** over a set ***S***.

*Fmincon* is the MATLAB (optimization toolbox version 2.2) function for constrained optimization problems. For general nonlinear optimization problems, *Fmincon* uses the SQP (*Sequential Quadratic Programming*) method.

The initial estimate of the Lagrangian Essian is the identity matrix. Its correction is made by the formula of Broyden, Fletcher, Goldfarb and Shanno (BFGS). If the gradient of the cost function is not provided, MATLAB estimates it by finite difference.

*Fmincon* searches for the minimum of a function subjected to linear and / or nonlinear constraints in a given interval. The goal of the function is the minimization of a function $\boldsymbol{f}$ over a set $\boldsymbol{S}$. If we are at a point $\boldsymbol{xk}$, we want to go to a point $\boldsymbol{y}$ whose image by $\boldsymbol{f}$ is smaller. For that, we approximate $\boldsymbol{f}$ by a relatively simple function $\boldsymbol{q}$, which reproduces $\boldsymbol{f}$ fairly well in a neighborhood $\boldsymbol{N}$ of$\boldsymbol{x}$. This neighborhood is called a “region of trust”.

We then minimize $\boldsymbol{q}$ on $\boldsymbol{N}$, which gives us the point $\boldsymbol{y}$. Then, we pose (since we are trying to minimize$\boldsymbol{f}$):

$$\boldsymbol{x_{k+1}} = \boldsymbol{y} \ \text{si}\ \boldsymbol{f}(\boldsymbol{y}) < f(\boldsymbol{x_k})$$

And if not, we narrow the confidence region and we approximate $\boldsymbol{f}$ again by a function $\boldsymbol{q}$ on the new neighborhood.

From an implementation point of view, the gradient $\boldsymbol{g}$ is partially approximated in all directions, and the Hessian matrix H is recursively calculated by a quasi-Newtonian method, in this case the BFGS algorithm. Thus, each iteration of the routine *Fmincon* costs in the order of $\boldsymbol{n}$ evaluations of the function $\boldsymbol{f}$ if $\boldsymbol{n}$ is the dimension of the problem.

Indeed, the gradient approximation is done by the finite difference method, that is to say exactly as in the descent method.

So we have a problem of the following form:

$$\min_x(F(x))$$

Such as: $$:|\begin{cases} C(x) \leq 0 \\ C_{eq} = 0 \\ Ax \leq B \\ A_{eq}x = B_{eq} \\ LB \leq x < UB \end{cases}$$

$x$, $B$, $Beq$, $LB\ et\ UB$ Are vectors, $A\ et\ Aeq$ are matrices, $C\ (x)\ et\ Ceq\ (x)$ are functions with vector values, $\boldsymbol{F}\ (\boldsymbol{x})$ is a scalar function. $\boldsymbol{f}\ (\boldsymbol{x})$, $C\ (x)\ et\ Ceq\ (x)$ Can be nonlinear functions.

## III.3 The 2R robot manipulator models:

### III.3.1. Direct kinematic:

In the case of our manipulator arm we found the following equations:

$$\begin{cases} X = L1\cos\alpha + L2\cos(\alpha+\beta) \\ Y = L1\sin\alpha + L2\sin(\alpha+\beta) \end{cases}$$

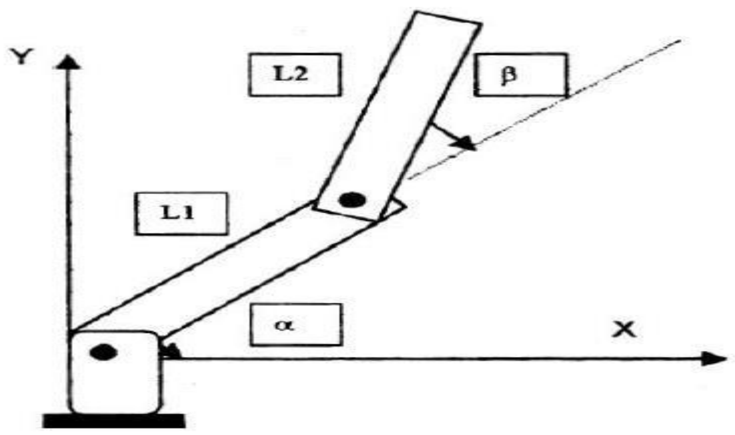


Figure3.1: Robot arm with 2R.

## III.3.2. Inverse Kinematics:

After calculations we find that:

$$\alpha = \operatorname{arctg}\left(YZ \pm \left(X^2+Y^2-Z^2\right)^{1/2}\right) \Big/ \left(ZX \pm \left(X^2+Y^2-Z^2\right)^{1/2}\right)$$

And

$$\beta = \operatorname{arctg}\left(X^3+XY^2-L_1ZX \pm \left(X^2+Y^2-Z^2\right)^{1/2}\right) \Big/ \left(Y^3+YX^2-L_1ZX \pm \left(X^2+Y^2-Z^2\right)^{1/2}\right) - \alpha$$

3.3.3. Dynamic of 2R Robot:

The model obtained can be put in the following matrix form:

$$\tau = M(q)\ddot{q} + N(q,\dot{q})\dot{q} + G(q) \qquad (3.1)$$

by applying the formulation of Lagrange:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = \tau_i$$

With:

K: Total kinetic energy of robot

P: Total potential energy of robot

$q_i$: Joint variable of i-th joint

$\dot{q}_i$: first time derivative of

$\tau_i$: Generalized force (torque) at i-th joint

We can find:

1) The inertia matrix:

$$M(q)=\begin{pmatrix} m_2l_2^2+2m_2l_2l_1c_2+(m_2+m_1)l_1^2 & m_2l_2^2+m_2l_2l_1c_2 \\ m_2l_2^2+m_2l_2l_1c_2 & m_2l_2^2 \end{pmatrix}$$

2) Coriolis and centrifuge matrix:

$$M(q,\dot{q})=\begin{pmatrix} -2m_2l_2l_1s_2\dot{\theta}_2 & -m_2l_2l_1s_2\dot{\theta}_2 \\ m_2l_2l_1s_2\dot{\theta}_1 & 0 \end{pmatrix}$$

3) Vectors of gravity forces:

$$G(q)=\begin{pmatrix} m_2l_2gc_{12}+(m_2+m_1)l_1gc_1 \\ m_2l_2gc_{12} \end{pmatrix}$$

We apply equation (3.1) we find:

$$\begin{aligned} \tau_1 &= m_2l_2^2(\ddot{\theta}_1+\ddot{\theta}_2)+m_2l_1l_2c_2(2\ddot{\theta}_1+\ddot{\theta}_2)+(m_1+m_2)l_1^2\ddot{\theta}_1- \\ & m_2l_1l_2s_2\dot{\theta}_2^2-2m_2l_1l_2s_2\dot{\theta}_1\dot{\theta}_2+m_2l_2gc_{12}+(m_1+m_2)l_1gc_1 \\ \tau_2 &= m_2l_1l_2c_2\ddot{\theta}_1+m_2l_1l_2s_2\dot{\theta}_1^2+m_2l_2gc_{12}+m_2l_2^2(\ddot{\theta}_1+\ddot{\theta}_2) \end{aligned} \tag{3.2}$$

## III.4 Optimization of trajectory planar 2-DOF robot passing through three points:

For a first application, we will deal with the case for planar 2-DOF robot. The geometrical parameters of the arms and the limited performances are indicated successively in Table 3.1. Table 3.2. presents the kinematic limits of the joints, including velocity, acceleration, jerk and dynamic constraints.

This manipulator robot must follow a path in the horizontal plane (XY) which passes through three points starting with (-0.1 rad, 2.8 rad) via (0.2 rad, 0.65 rad) to (0.9 rad, 1.25 rad), and the goal of this application is to minimize an objective function which is in the following form:

$$F_{obj} = Time$$

Subject to:

$$\begin{cases} |\dot{Q}(t)| \le \dot{Q}^{\max},\ j=\cdots 1,2,\dots,N. \\ |\ddot{Q}(t)| \le \ddot{Q}^{\max},\ j=\cdots 1,2,\dots,N \\ |\dddot{Q}(t)| \le \dddot{Q}^{\max},\ j=\cdots 1,2,\dots,N \\ |\tau(t)| \le \tau^{\max},\ j=\cdots 1,2,\dots,N \end{cases}$$

Where:

$\dot{Q}(t), \ddot{Q}(t)\, \dddot{Q}(t)$ and $\tau(t)$ represents the velocity, acceleration, jerk and torque vectors of the *j*th joint.

*N* represent the number of robotic manipulator joints

Using the sequential quadratic programming optimization technique, we get the following results:

Table 3.1: Geometric and inertial parameters of the planar 2-DOF robot.

| | Lien 1 | Lien 2 |
|---|---|---|
| M (kg) | 30 | 19.2 |
| D (m) | 0.15 | 0.16 |
| L (m) | 0.30 | 0.32 |
| Izz (kg m$^2$) | 0.225 | 0.16 |

Table 3.2: Kinematic and dynamic constraints of the robot.

| | Lien 1 | Lien 2 |
|---|---|---|
| Couple (N m) | 40 | 40 |
| Vitesse (rad s$^{-1}$) | 5 | 5 |
| Accélération (rad s$^{-2}$) | 20 | 20 |
| Jerk (rad s$^{-3}$) | 30 | 30 |

For this example: we have processed a trajectory with three points (initial point, intermediate point and the end point) and according to the results obtained, the minimum value found to execute this task $F_{obj}$= 1.8057 s (see figure 3.6), the intervals $h_i$ are represented in the table 3.3, Note that the various kinematic and dynamic constraints have been respected (see table 3.4) and the results found show that the values of the velocities, accelerations, jerk and the torques do not exceed the limited values as shown in figures 3.3, 3.4 3.5 & 3.6 respectively.

Table 3.3: Time intervals of the 2R Planar Robot.

| Interval N° | $h_1$ | $h_2$ | $h_3$ | $h_4$ |
|---|---|---|---|---|
| Time Interval | 0.3388 | 0.6845 | 0.5641 | 0.2183 |

Table 3.4: Maximum kinematic values of the 2R Planar Robot.

| Joint | Velocity (°/s) | Acceleration (°/s$^2$) | Jerk (°/s$^3$) | Torque (N.m) |
|---|---|---|---|---|
| 1 | 1.3898 | 5.0085 | 22.9398 | 20.1838 |
| 2 | 3.4426 | 10.3726 | 30.0000 | 4.9010 |

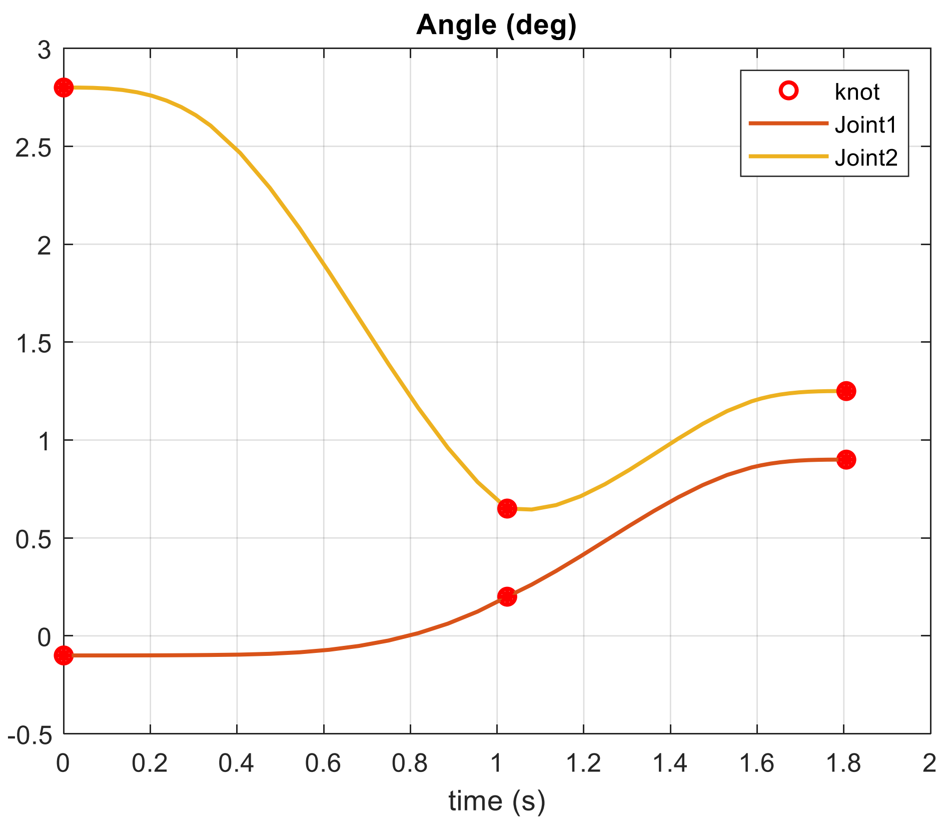


Figure. 3.2: Angle of the 2R- Robot for the trajectory passing through three points under kino-dynamic constraints

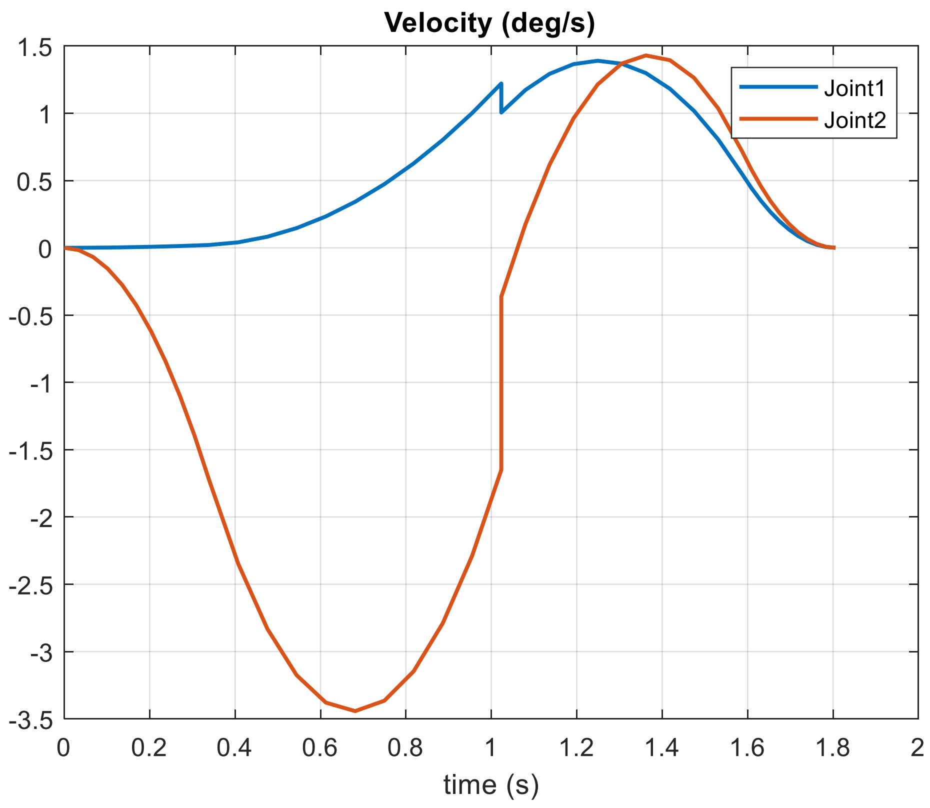


Figure. 3.3: Velocity of the 2R- Robot under kino-dynamic constraints

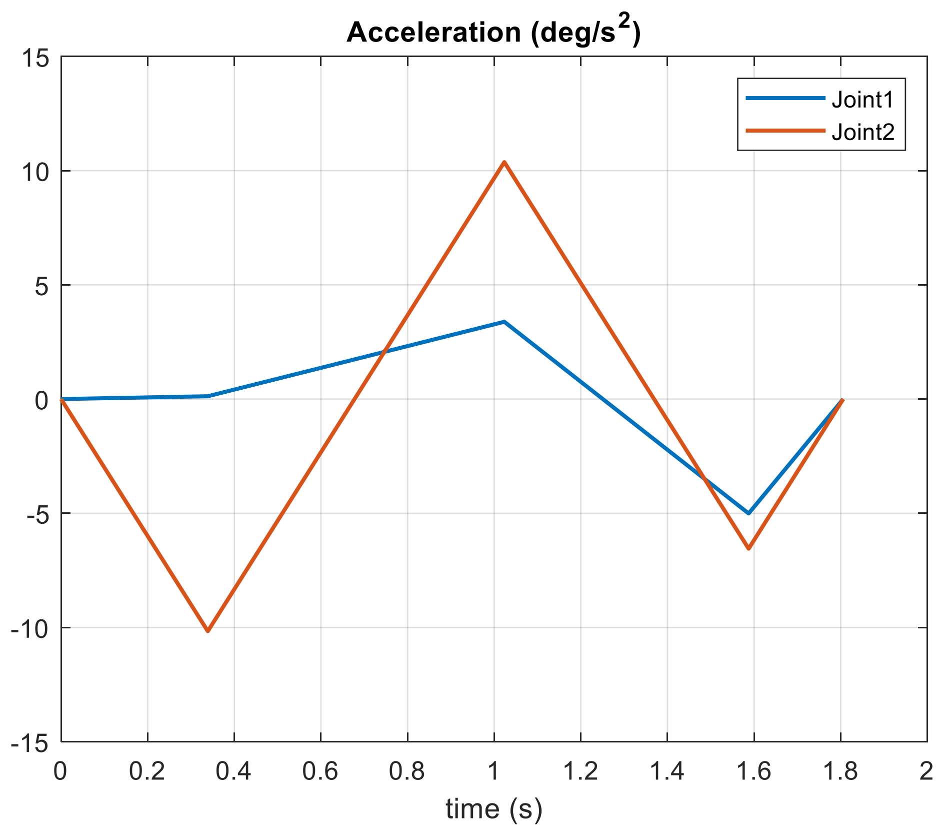


Figure. 3.4: Acceleration of the 2R- Robot under kino-dynamic constraints

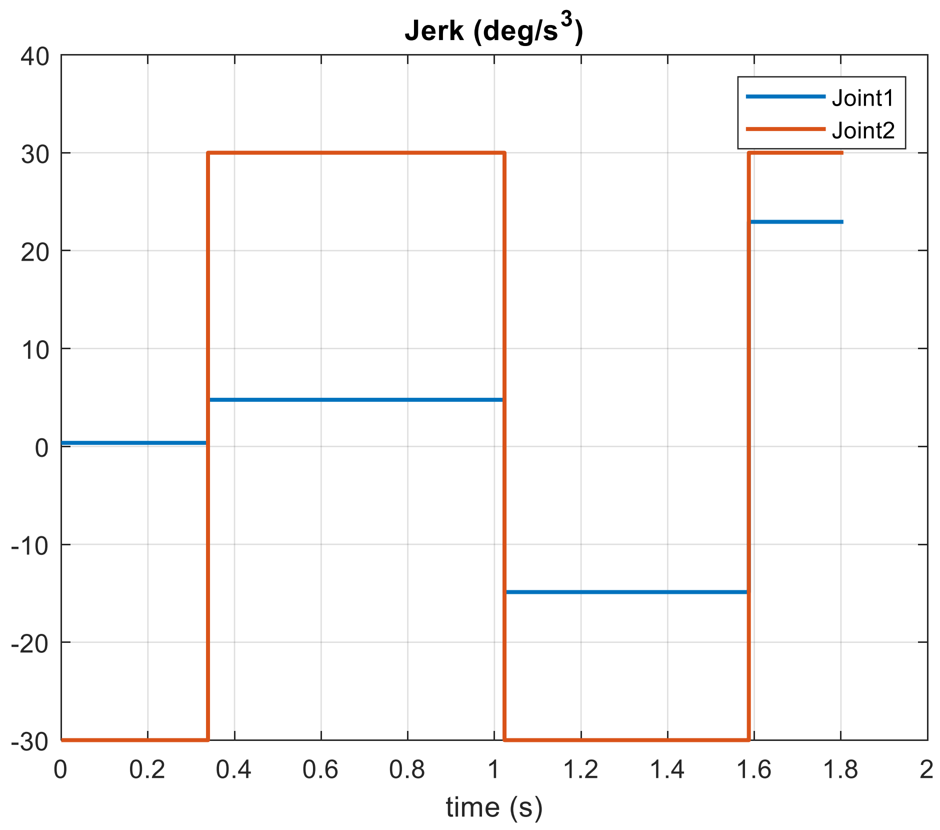


Figure. 3.5: Jerk of the 2R- Robot under kino-dynamic constraints

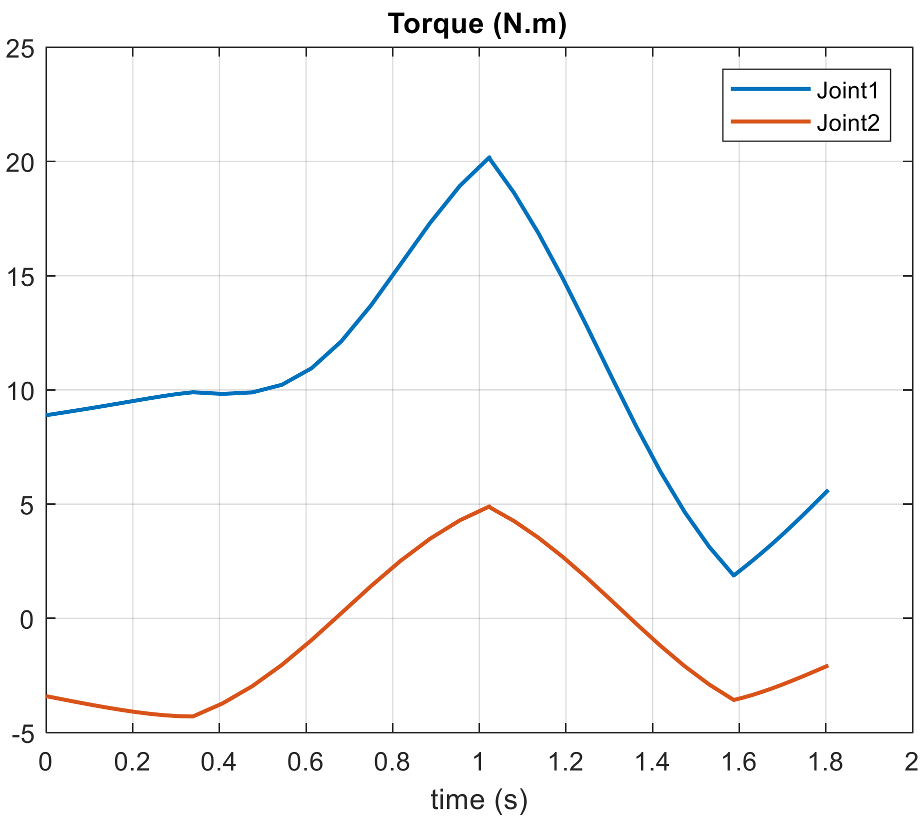


Figure. 3.6: Torque of the 2R- Robot under kino-dynamic constraints

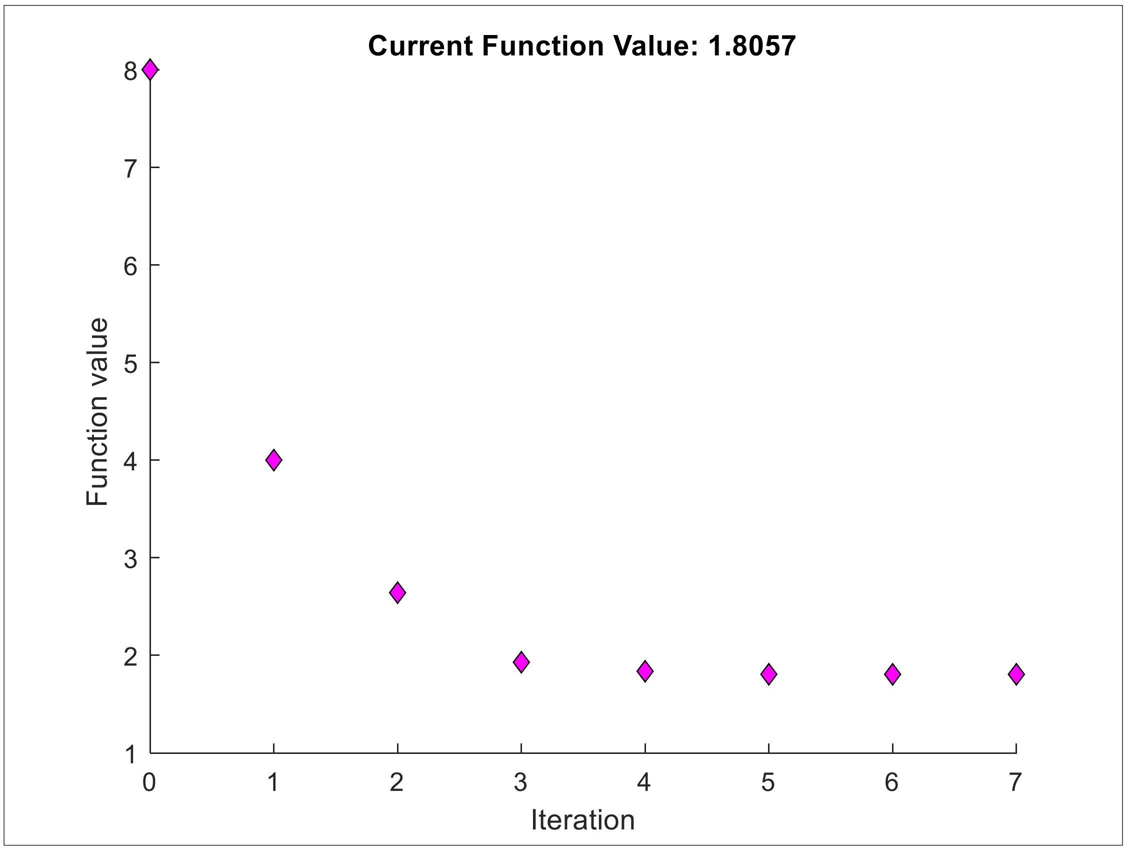


Figure 3.7: Generation results of 2R robot using SQP

## III.5 Optimization of trajectory planar 2-DOF robot passing through 11 points:

In this example we will treat a planar 2R robot (see Fig 3.8) which is subjected to dynamic constraints due to the couples, this robot must pass through eleven imposed points showed in Table 3.5 corresponding to the examples given by *Bianco etal*. [17] The second point is calculated by taking the sum of points 1 and 3 and dividing on 2 and the same thing for the tenth point we add the points 9 and 11 and we divide on 2.

The expression of the torque is given by the equation (3.2). The imposed torque limit vector is $\tau = [500 \quad 200]^T \; Nm$. The list of the robot parameters is reported in Table 3.6. We opt for a trajectory with a minimal time therefore we form an objective function which minimizes the execution time of the task expressed as:

$$F_{obj} = Time$$

Subject to:

$$|\tau(t)| \leq \tau^{\max}, \; j = \cdots 1, 2, ..., N$$

Where:

$\tau(t)$ represents the torque vector of the *j*th joint.

*N* represent the number of robotic manipulator joints.

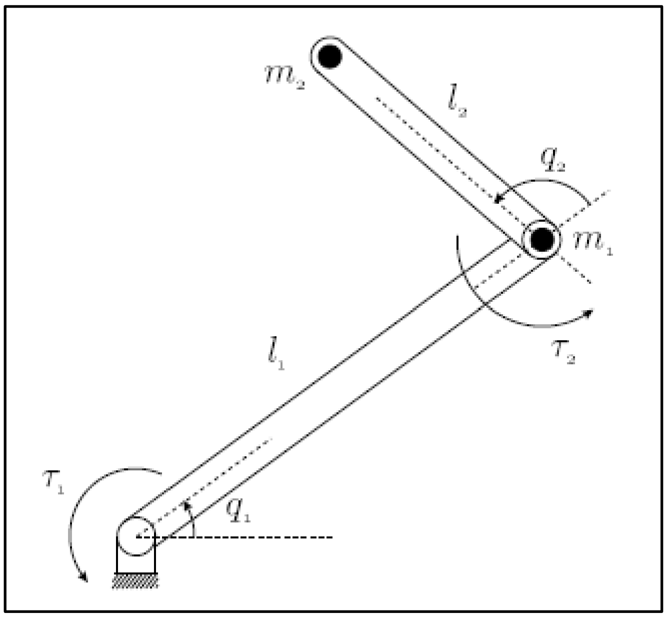


Figure 3.8: Schematic representation of the two-link planar robot.

Table 3.5: End-Effector of the two planar arms passes through the points expressed in radians.

| | |
|---|---|
| **$q^0_1$= -0.334990** | **$q^0_2$=1.40335** |
| **$q^2_1$=1.05271** | $q^2_2$=1.42444 |
| **$q^3_1$=0.94284** | $q^3_2$=1.50157 |
| **$q4_1$=0.76891** | $q4_2$=1.42444 |
| **$q5_1$=0.58780** | $q5_2$=1.35666 |
| **$q6_1$=0.33385** | $q6_2$=1.35666 |
| **$q7_1$=0.12541** | $q7_2$=1.42444 |
| **$q8_1$=-0.07136** | $q8_2$=1.50157 |
| **$q9_1$=-0.15882** | $q9_2$=1.42470 |
| **$q11_1$=1.23590** | $q11_2$=1.40335 |

Table 3.6: Parameters of the two-link arm.

| | |
|---|---|
| $l_1 = 4.0$ m | $m_1 = 2.0$ Kg |
| $l_2 = 1.5$ m | $m_2 = 1.0$ Kg |

Using the sequential quadratic programming optimization technique, we get the following results:

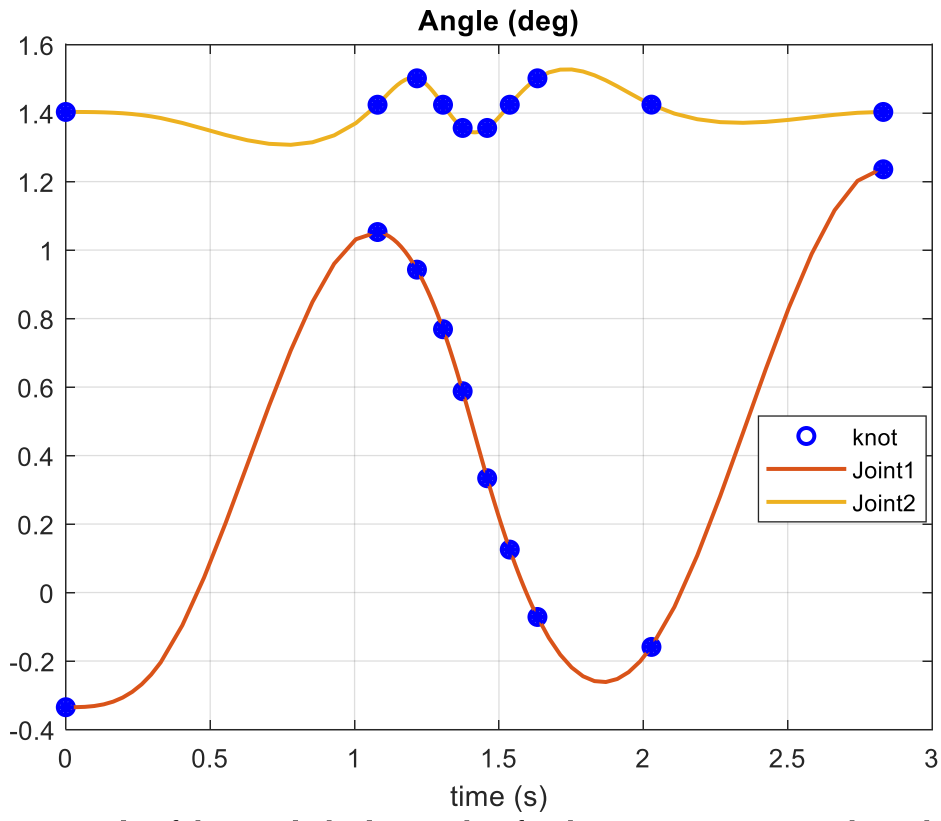


Figure. 3.9: Angle of the two-link planar robot for the trajectory passing through eleven points under dynamic constraints

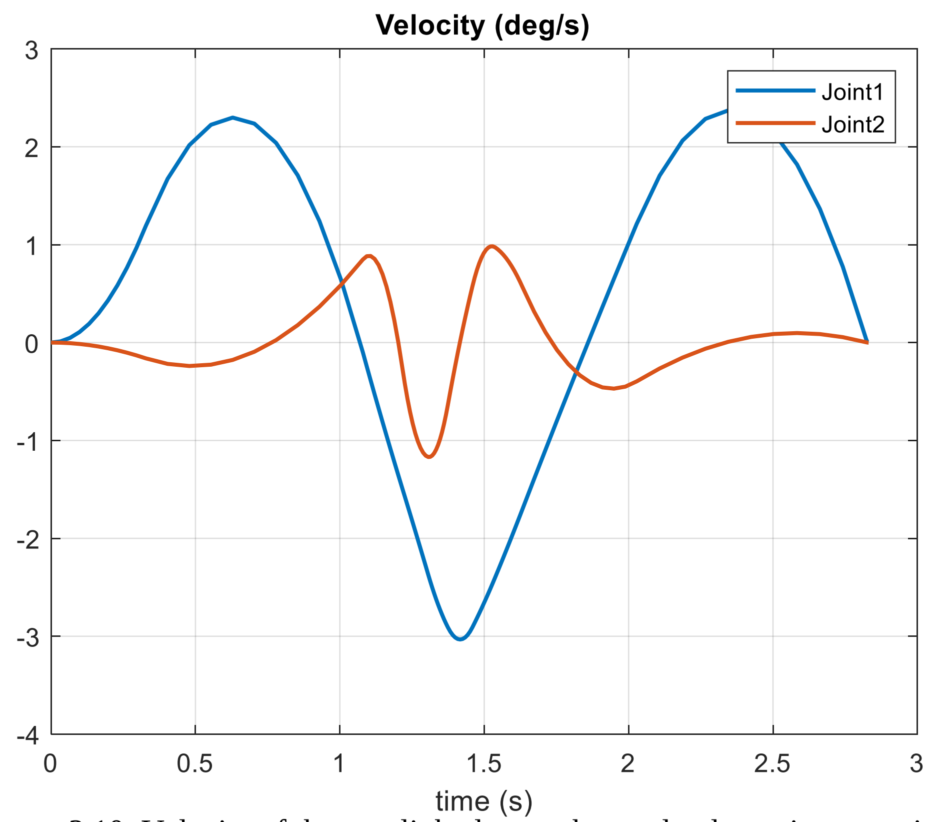


Figure. 3.10: Velocity of the two-link planar robot under dynamic constraints

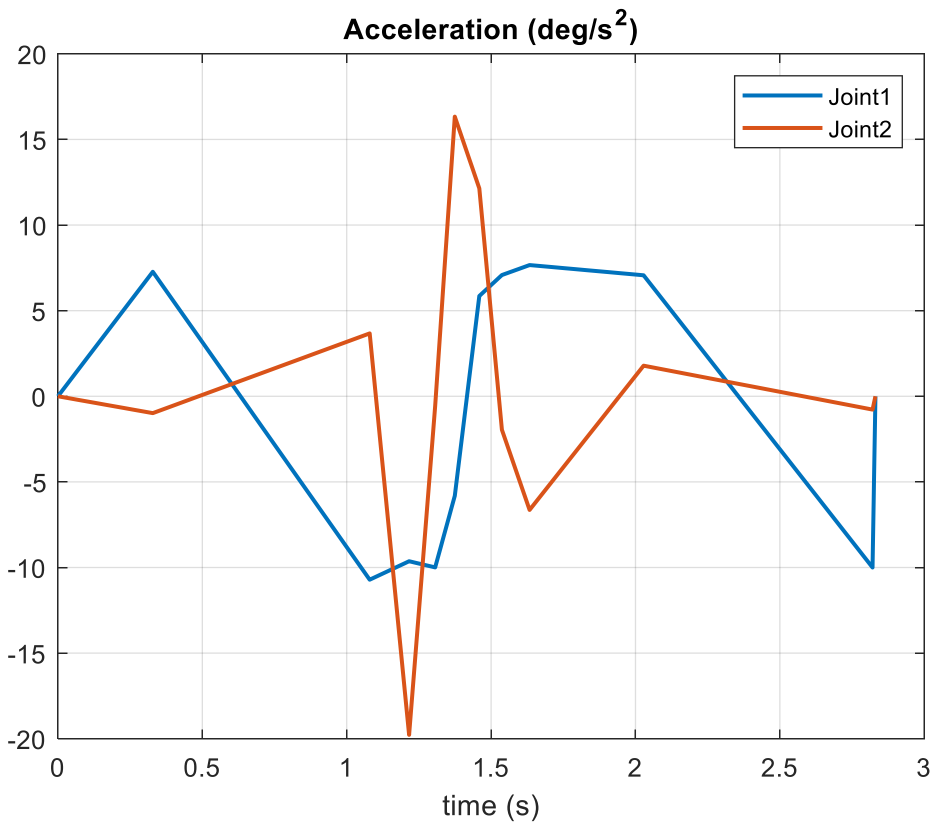


Figure. 3.11: Acceleration of the two-link planar robot under dynamic constraints

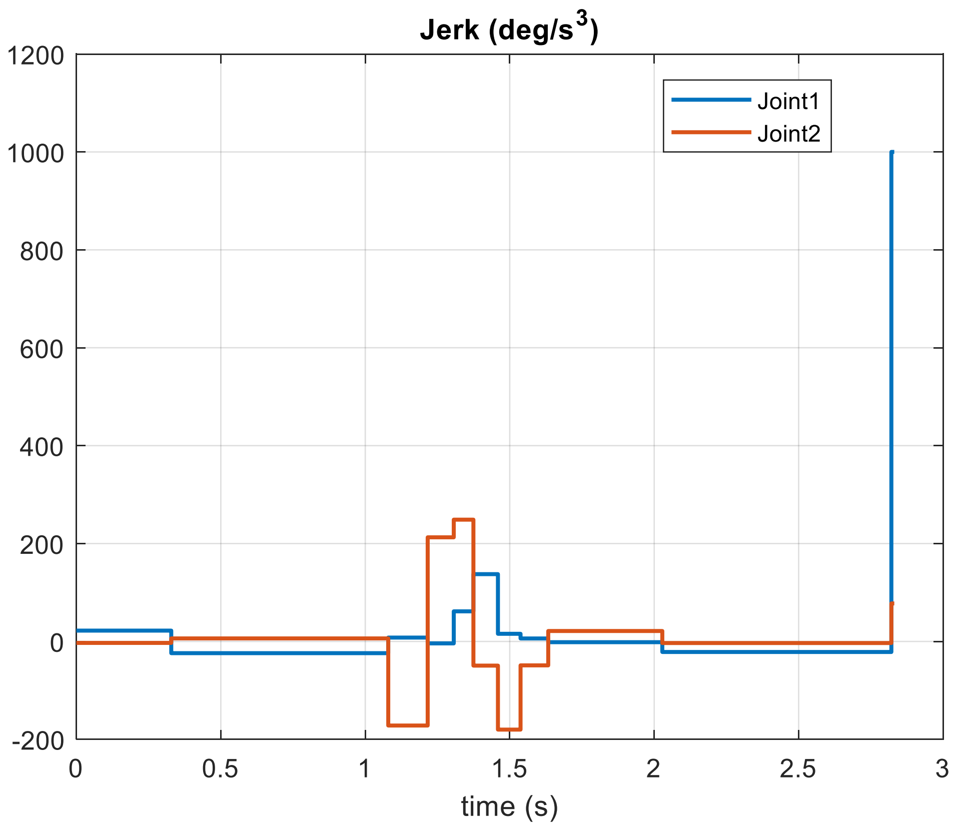


Figure. 3.12: Jerk of the two-link planar robot under dynamic constraints

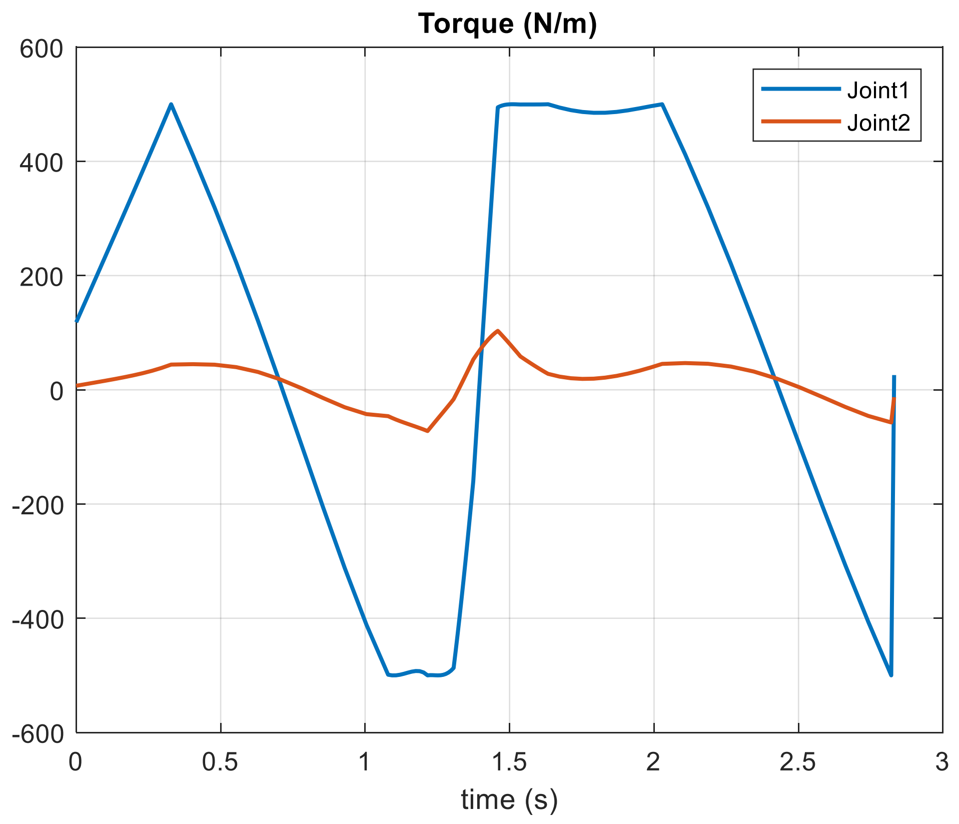


Figure. 3.13: Torque of the two-link planar robot under dynamic constraints

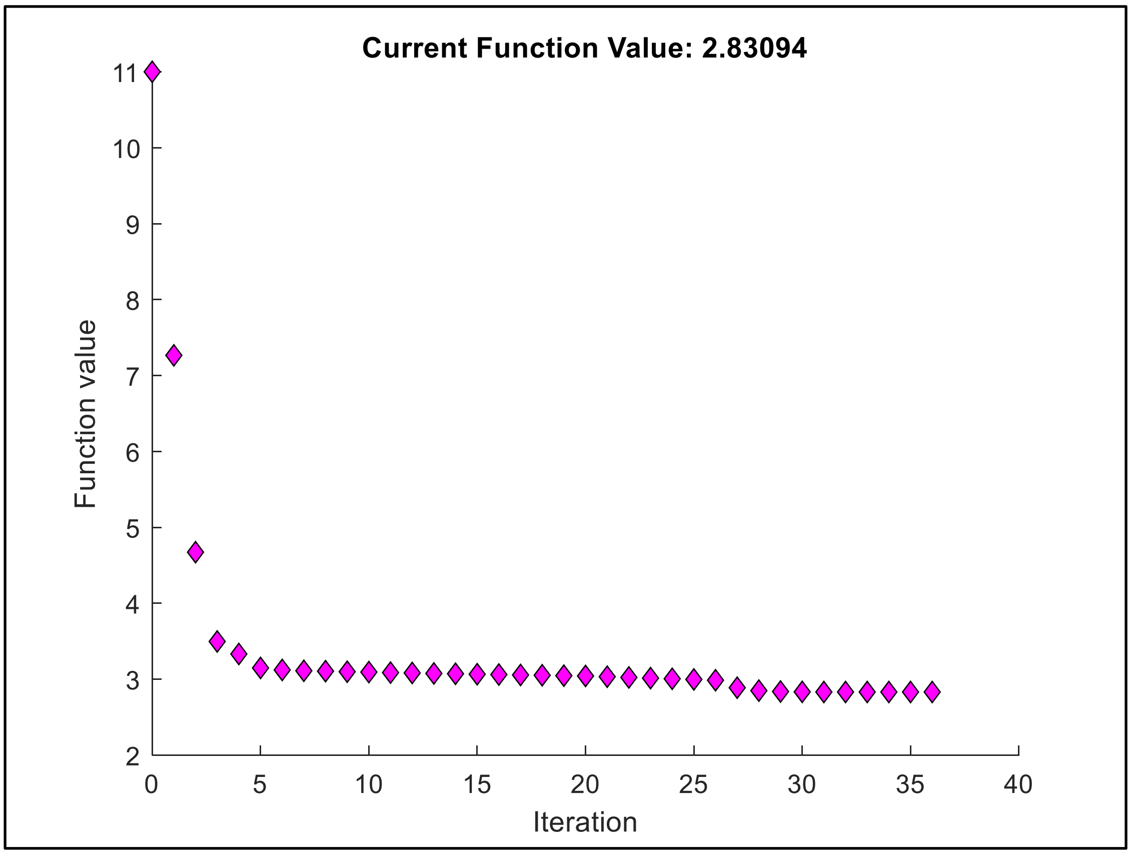


Figure 3.14: Generation results of the two-link planar robot using SQP

Table 3.7: Time intervals of the two-link planar robot using SQP.

| Interval | Time Interval found by our method | Time Interval found by [17] |
|---|---|---|
| $h_1$ | 0.3285 sec | 0.29536 sec |
| $h_2$ | 0.7511 sec | 0.77166 sec |
| $h_3$ | 0.1365 sec | 0.15203 sec |
| $h_4$ | 0.0903 sec | 0.09065 sec |
| $h_5$ | 0.0680 sec | 0.07087 sec |
| $h_6$ | 0.0849 sec | 0.08920 sec |
| $h_7$ | 0.0783 sec | 0.08525 sec |
| $h_8$ | 0.0958 sec | 0.10951 sec |
| $h_9$ | 0.3949 sec | 0.29057 sec |
| $h_{10}$ | 0.7928 sec | 0.86583 sec |
| $h_{11}$ | 0.0100 sec | 0.05000 sec |
| $F_{obj} = Time = \sum_{i=1}^{11} h_i$ | **2.8309 sec** | **2.87094 sec** |

For this example, we have treated an eleven-point trajectory with an initial point and an end point and nine intermediate points, by applying our programs under MATLAB and respecting the imposed dynamic constraints, an objective function which depends only on time was minimized and we found as optimal function $F_{obj} = \sum_{i=1}^{11} h_i = 2.8309$ sec. Note that for the same task an optimal objective function found in [17] using genetic/interval algorithm $F_{obj} = \sum_{i=1}^{11} h_i = 2.87094$ sec.

Hence, our method mains to the lowest value of time tasks while guaranteeing a high degree of smoothness as showed in Figs. 3.9–3.13.

This demonstrates the fact that the proposed technique is capable of constructing high speed trajectories, in order to achieve shorter production times, within specified dynamic constraints while taking into account end conditions of velocities and accelerations.

## III.6 Optimization of trajectory for planar 3-DOF robot passing through nine points:

In this example we will treat a planar 3-DOF robot (see Fig 3.15) which is subjected to dynamic constraints due to the couples, this robot must pass through nine imposed points showed in Table 3.8 corresponding to the examples given by *Shaotian et al*[18]. The second point is calculated by taking the sum of points 1 and 3 and dividing on 2 and the same thing for the tenth point we add the points 7 and 9 and we divide on 2.

The kinematic constraints of the velocity, acceleration, and jerk of the joints are 3 deg/s, 3 deg/s2, and 5 deg/s3, respectively. The tested mission is tracking a circular path in plane. In Figure 3.15, $q1$, $q2$, and $q3$ are joint angles.

The robot geometric parameters are defined as:

$l1$ = 2,080 mm, $l2$ = 2,080 mm, and $l3$ = 430 mm.

Because the robot has a redundant degree of freedom, the configuration control approach is utilized to determine its inverse kinematics. The initial position of the end effector is (4000,0) (mm), with corresponding joint angles of (30°, −49.03°, −38.27°), and the radius $r$ and the center of the track circle are 400mmand (3600, 0) (mm), respectively.

We opt for a trajectory with a minimal time therefore we form an objective function which minimizes the execution time of the task expressed as:

$$F_{obj} = Time$$

Subject to:

$$\begin{cases} \left|\dot{Q}(t)\right| \le \dot{Q}^{\max}, \ j = \cdots 1, 2, ..., N. \\ \left|\ddot{Q}(t)\right| \le \ddot{Q}^{\max}, \ j = \cdots 1, 2, ..., N \\ \left|\dddot{Q}(t)\right| \le \dddot{Q}^{\max}, \ j = \cdots 1, 2, ..., N \end{cases}$$

Where:

$\dot{Q}(t), \ddot{Q}(t)$ and $\dddot{Q}(t)$ represents the velocity, acceleration and jerk vectors of the *j*th joint. *N* represent the number of robotic manipulator joints

Table 3.8: Input data for trajectory planning.

| **Joint** | **Knot angles(degrees)** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **1** | 30.00 | Virtual | 39.94 | 49.24 | 47.66 | 37.60 | 29.48 | virtual | 30.04 |
| **2** | -49.03 | | -57.25 | -72.45 | -79.89 | -72.42 | -57.16 | | -49.14 |
| **3** | -38.27 | | -42.00 | -49.16 | -52.48 | -49.25 | -42.30 | | -37.96 |

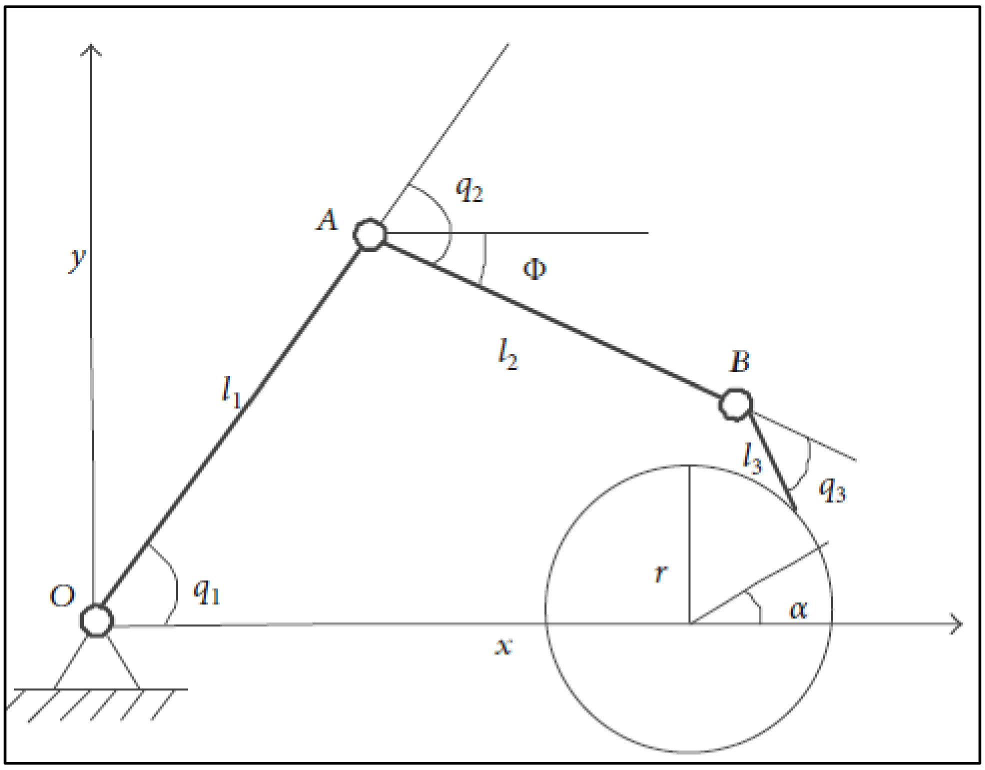


Figure 3.15: Schematic diagram of the planar 3-DOF robot.

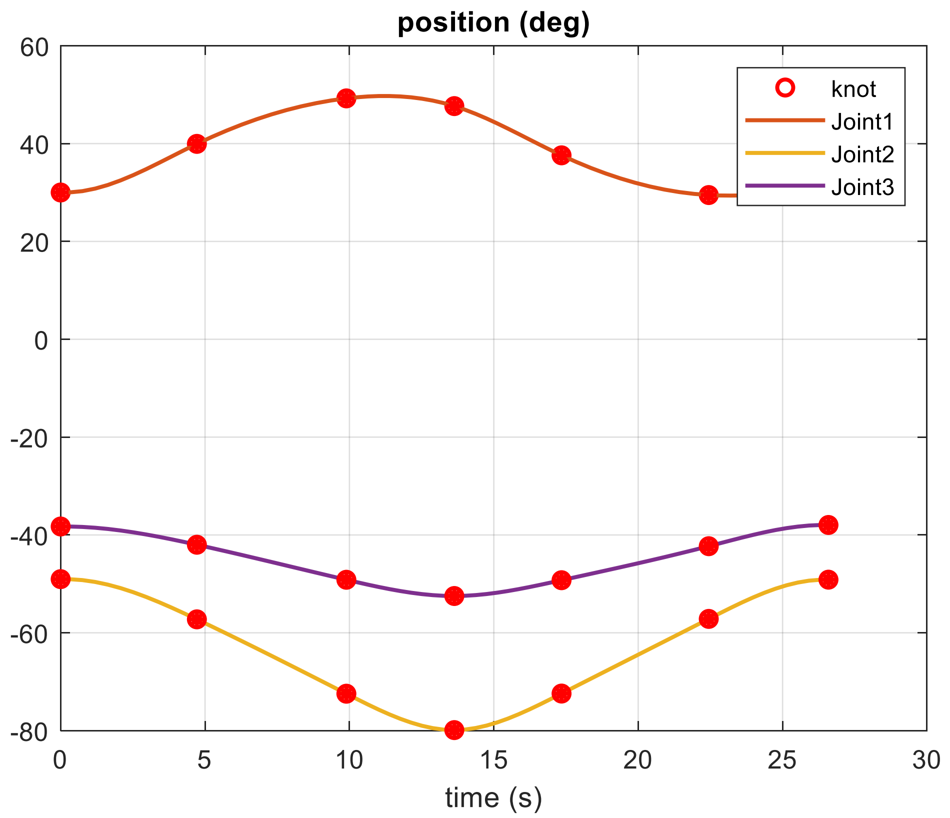


Figure. 3.16: Angle of the planar 3-DOF robot for the trajectory passing through nine points under kinematics constraints

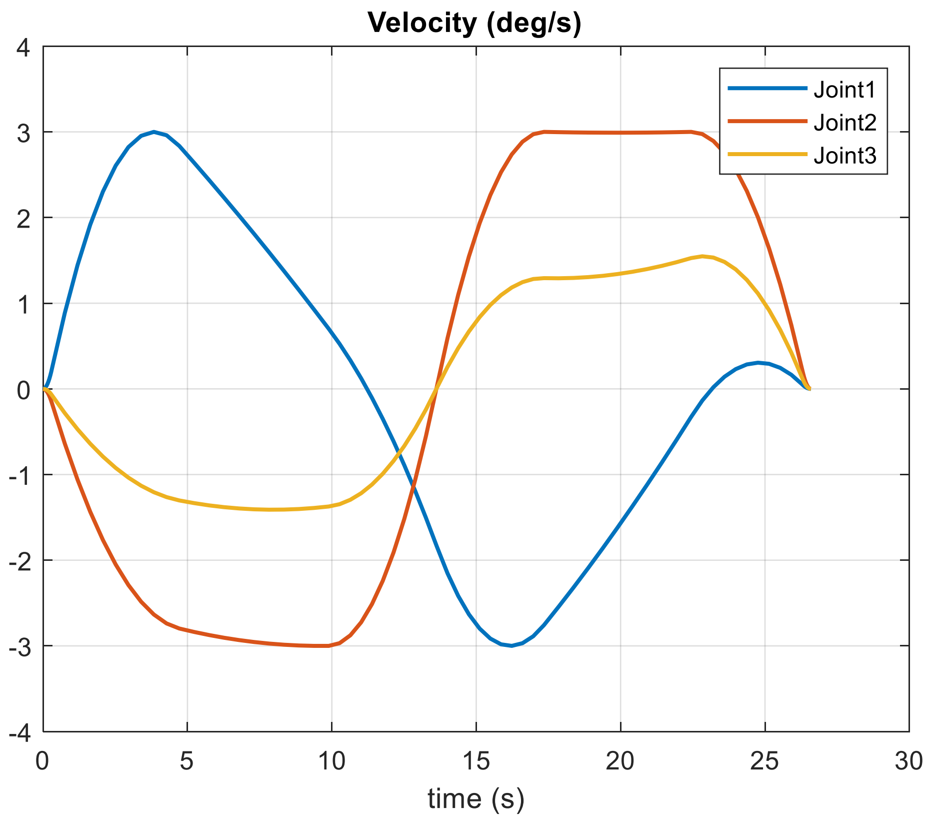


Figure. 3.17: Velocity of the two-link planar robot under kinematics constraints

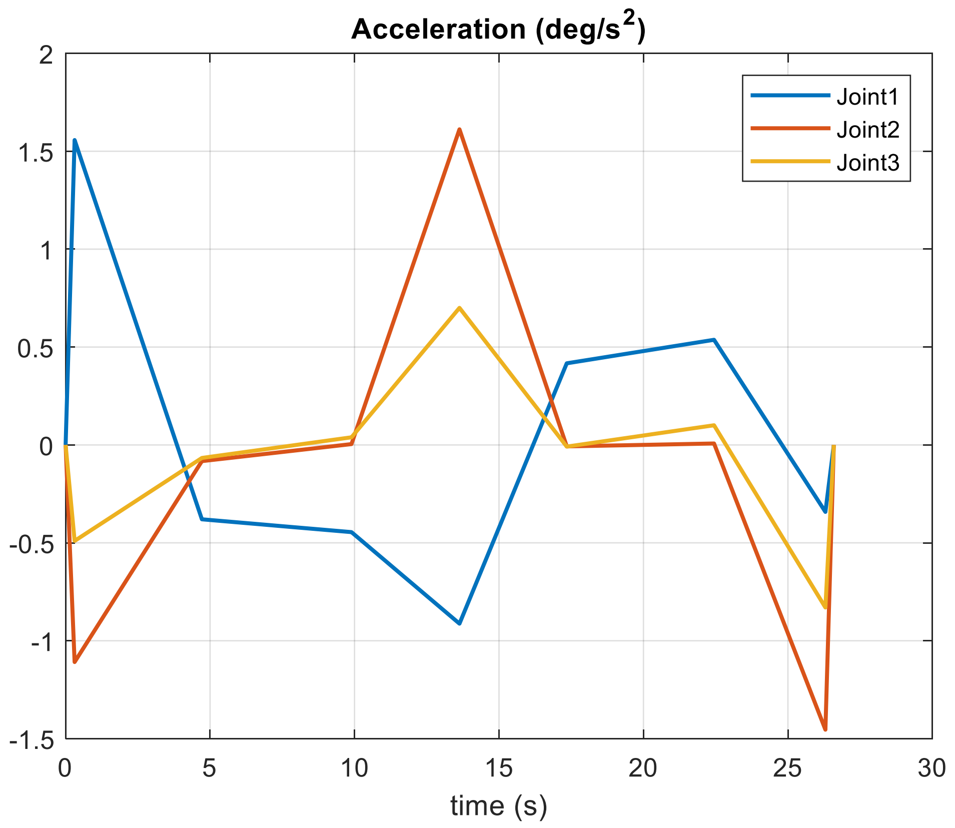


Figure. 3.18: Acceleration of the two-link planar robot under kinematics constraints

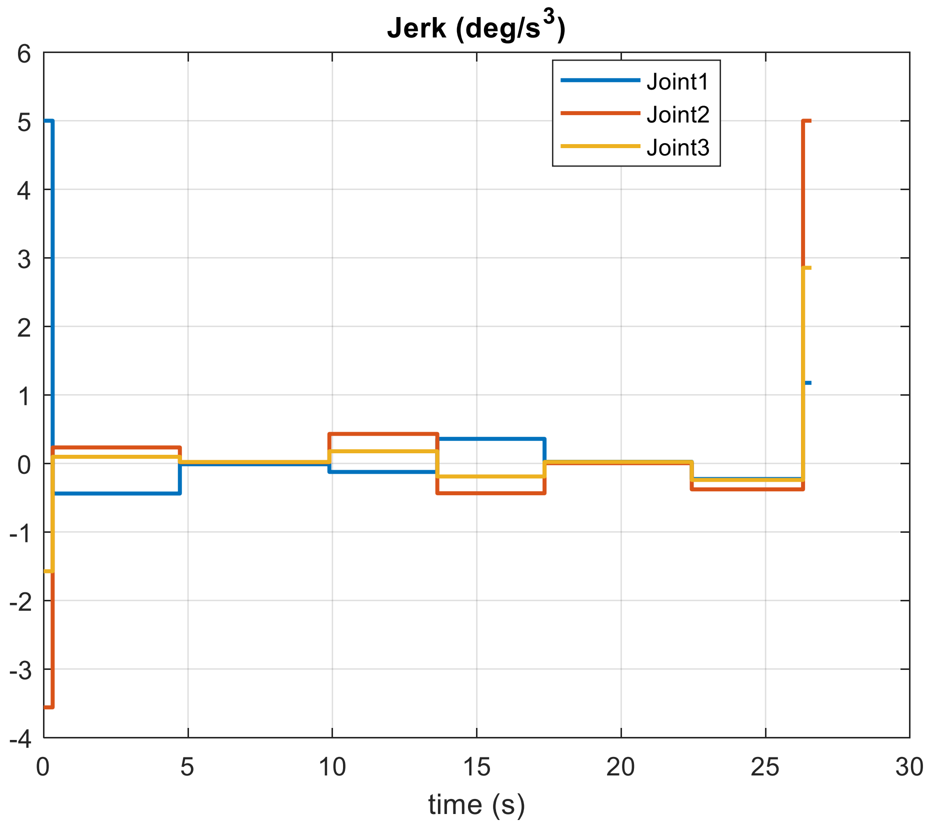


Figure. 3.19: Jerk of the two-link planar robot under kinematics constraints

Table 3.9: Time intervals of the planar 3-DOF robot using SQP.

| Interval | Time Interval found by our method | Time Interval found by ALCPSO algorithm |
|---|---|---|
| $h_1$ | 0.3114 sec | 0.2527 sec |
| $h_2$ | 4.4058 sec | 4.8729 sec |
| $h_3$ | 5.1768 sec | 5.2656 sec |
| $h_4$ | 3.7380 sec | 3.2660 sec |
| $h_5$ | 3.7137 sec | 5.1017 sec |
| $h_6$ | 5.0974 sec | 5.2044 sec |
| $h_7$ | 3.8547 sec | 3.9063 sec |
| $h_8$ | 0.2908 sec | 0.2826 sec |
| $F_{obj} = Time = \sum_{i=1}^{11} h_i$ | **26.5886 sec** | **28.1523 sec** |
| ***Time (sec)*** | **1.754557** | **82.125** |

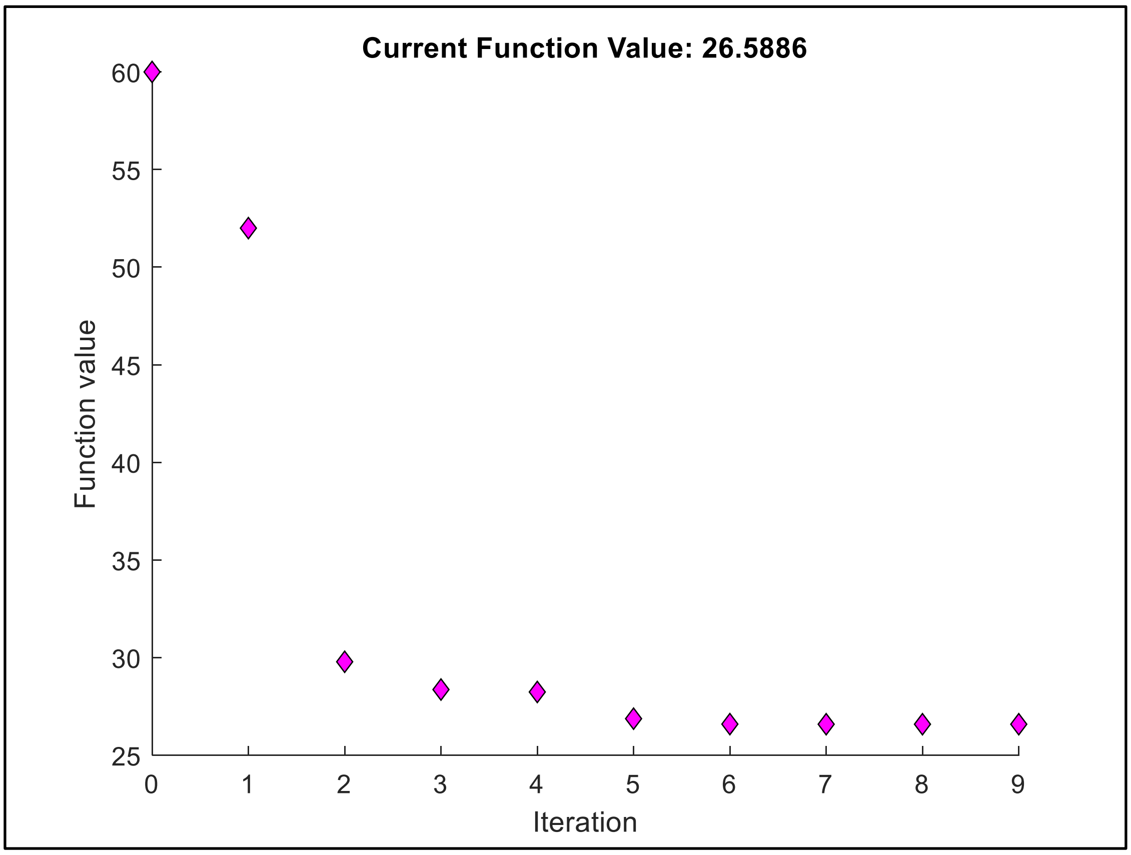


Figure 3.20: Generation results of planar 3-DOF robot using SQP

For this example: we have processed a trajectory with nine points and according to the results obtained, the minimum value found to execute this task $F_{obj}$= *26.5886 sec* (see figure 3.6), while the transfer time found with *Lagrange Constrained Particle Swarm Optimization* (LCPSO) is $F_{obj}$= 28.1523 *sec*. The intervals $h_i$ are represented in the table 3.9.

Moreover, Table 3.9 shows that our method spent the least execution time out of the other method (LCPSO).

Note that the various kinematic and dynamic constraints have been respected (see table 3.10) and the results found show that the values of the velocities, accelerations and jerk do not exceed the limited values as shown in figures 3.17, 3.18 & 3.19 respectively.

Table 3.10: Maximum kinematic values of the Planar 3-DOF Robot.

| **Joint** | **Velocity (°/s)** | **Acceleration (°/s$^2$)** | **Jerk (°/s$^3$)** |
|---|---|---|---|
| **1** | 3.0000 | 1.5571 | 5.0000 |
| **2** | 3.0000 | 1.6119 | 5.0000 |
| **3** | 1.5479 | 0.8302 | 2.8544 |

### III.6.1 Applying D-H:

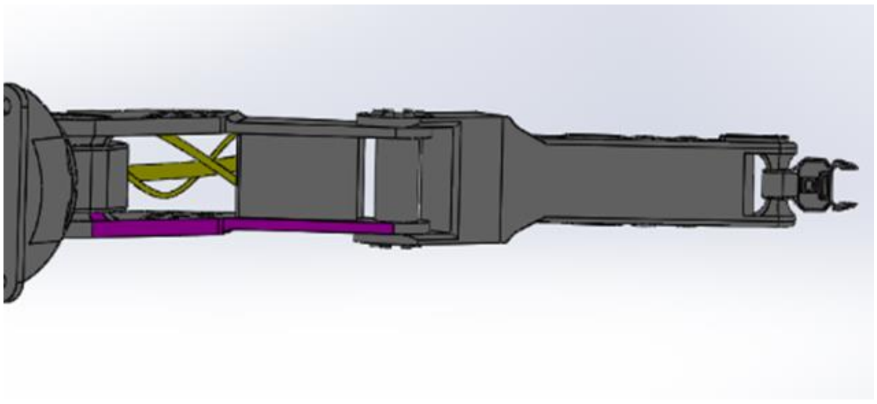


Figure3.21 A bottom view of the robot arm

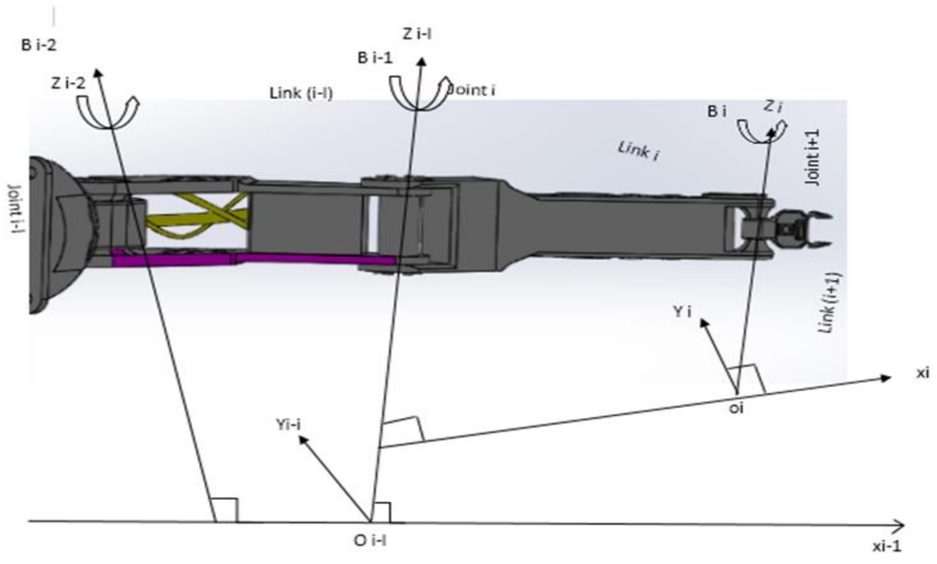


Figure3.22 The dh's applying

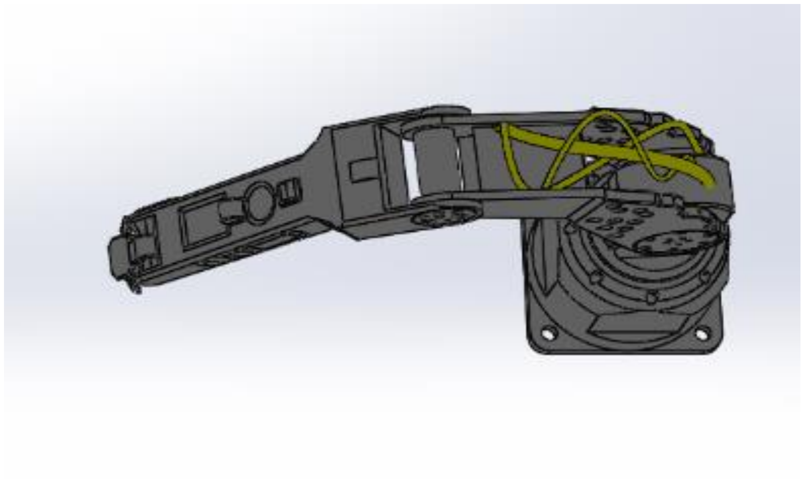


Figure3.23 A Top view of the robot arm

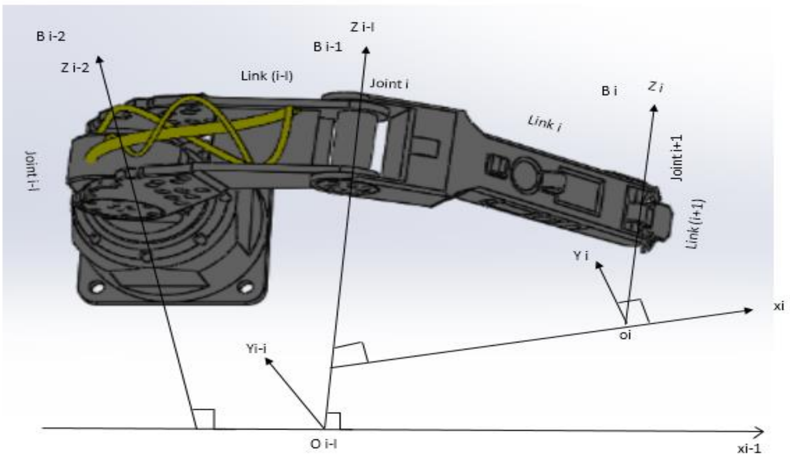


Figure3.24 the dh's applying on the robot arm

Doing a Pure IKS solution: the 2R Manipulator

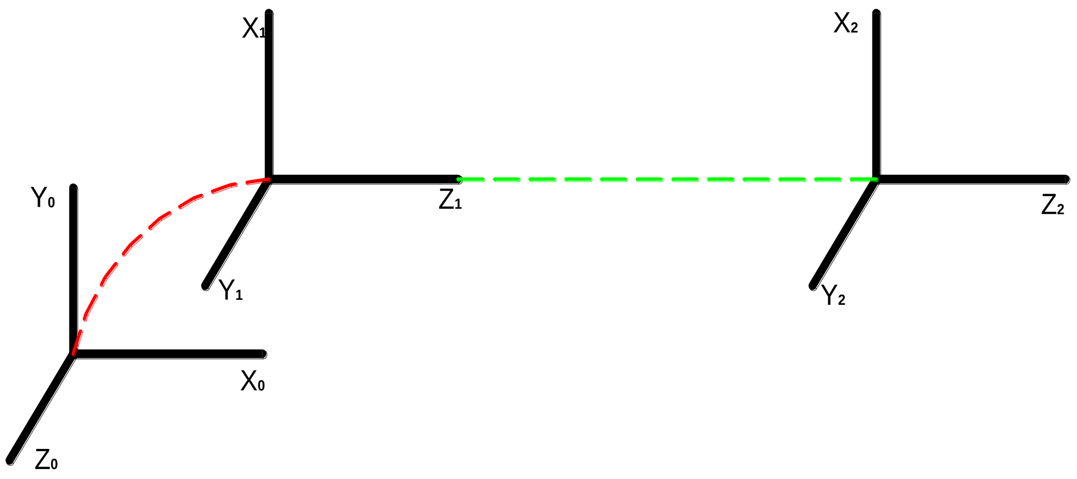


Figure3.25 “2R Frame Skeleton (as DH Suggest!)

Table 3.11: LP Table and Ai’s

| Frames | link | var | α | d | I | θ | s | c | s | c |
|---|---|---|---|---|---|---|---|---|---|---|
| **0-1** | 1 | R | α + 90 | 0 | 0 | 90 | 1 | 0 | C1 | -s1 |
| **1-2** | 2 | P | 0 | D2+cl2 | 0 | 0 | 1 | 0 | 1 | 0 |

Solving:

•Selecting and Equating terms (0,1)

• 0 = -S1*dx + C1*dy

• Solving: S1*dx = C1*dy

•Tan (θ) = (S1/C1) = (dy/dx)

• θ = Atan2(dx, dy)

•Selecting and Equating terms (1,2) -- after back substituting θ solution – and note, after the above step, θ is known as an angle

• d2 + cl2 = C1*dx + S1*dy

• d2 = C1*dx + S1*dy - cl2

•d2 = Cos [Atan2(dx, dy)] *dx + Sin [Atan2(dx, dy)] *dy –cl2

## III.7 Conclusion:

After the execution of the two tests, we managed to apply the parameters studied in the previous chapters and thus we proved the efficiency of the proposed method and the reliability of the results obtained by comparing with others of the scientific literature.

In the first step, we used Lagrange's formulation for the calculation of the dynamic model of the robot which will be used to determine the couples of the actuators as a function of joint velocities and accelerations.

The second step was devoted to the modeling of the trajectory using the cubic spline method.

In the third step, we proceeded to determine the optimal trajectory using the MATLAB function ‘FMINCON’ guided by the gradient of the objective function.

# General conclusion

In this dissertation, we dealt with the problem of trajectory planning for manipulator robots performing fast operations, for that we proposed a technique which focuses on the modeling of the trajectories using cubic splines functions and the optimization of an objective function which represents the time execution of the operations for robots manipulator using the technique of sequential quadratic programming (SQP).

To see the effectiveness of our approach, we have treated three examples. The results found and compared with others already published in the journals show that our approach can give very satisfactory and encouraging results.

As perspective, we propose to minimize other objective function / multi-objective functions, to apply this approach to other rigid (PUMA, Stanford, …etc.) robots or flexible systems.

Unfortunately, our study did not achieve its objectives previously targeted because of the exceptional case that we live with measures related to Covid-19.

# References

[1]: Dr.Jizhong Xiao  Department of electrical engineering “city college of New York ’

[2]: www.robots_alive.com

[3]: Design and applications of industrial robots “SABARIVAN.R’

[4]: Cours link in learning: chapter 2 – Robot Kinematics: position analysis

[5]: Azorobatics.com -Article-

[6]: Chapter five trajectory planning by Russell Fletcher

[7]: chapter three trajectory planning by bendali nadir

[8]: Robotics – Path and Trajectory Planning, Dr. Amit Goradia

[9]: Farin, G.: Kurven und Flächen im Computer Aided Geometric Design

[10]: H.-J. Bungartz, M. Griebel, C. Zenger: Einführung in die Computergraphik

[11]: Foley, van Dam, Feiner, Hughes: Computer Graphics: Principles and Practice - Second Edition in C

[12]: http://olli.informatik.uni-oldenburg.de/Grafiti3/grafiti/flow10/page1.html

[13]: http://www.cs.berkeley.edu/~sequin/CS284/IMGS/

[14]: http://escience.anu.edu.au/lecture/cg/Spline/bSplineFunction.en.html

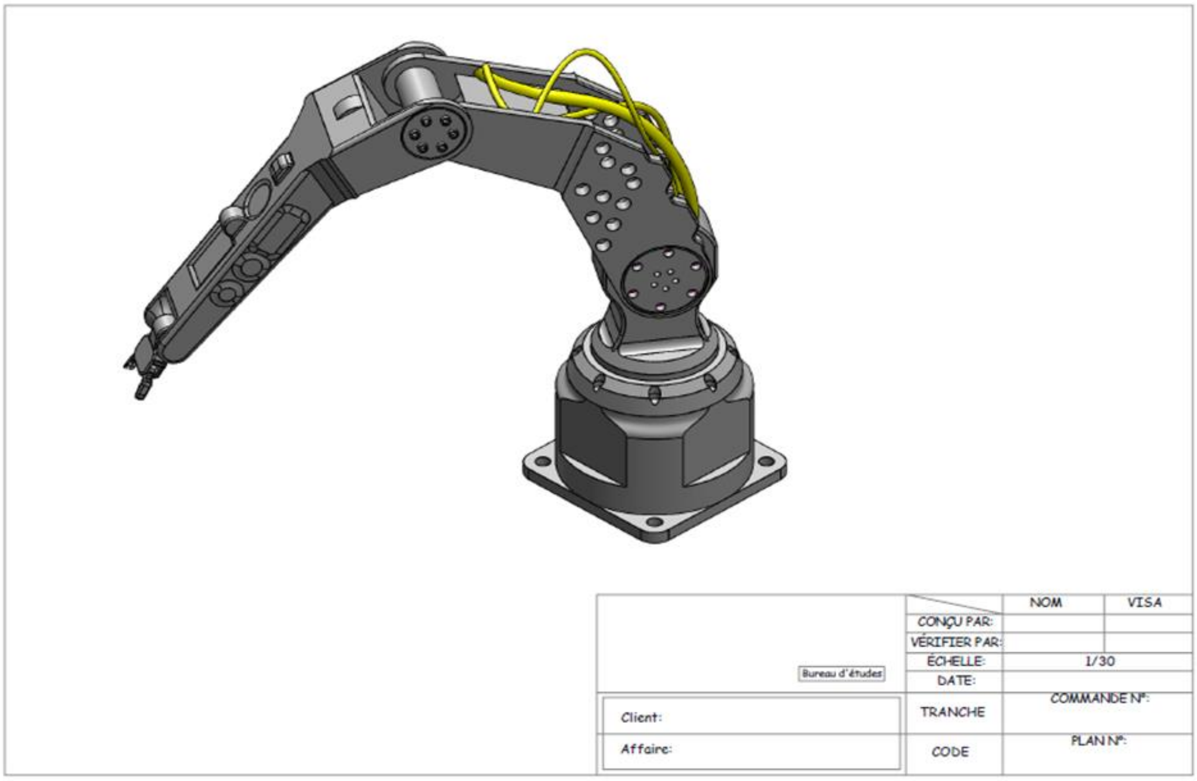
[15]: ME 4135 – Robotics and Controls

[16]: Dr. Jizhong Xiao, ME 3230, Kinematics and Mechatronics Dr. R. Lindeke, Department of Electrical Engineering City College of New York

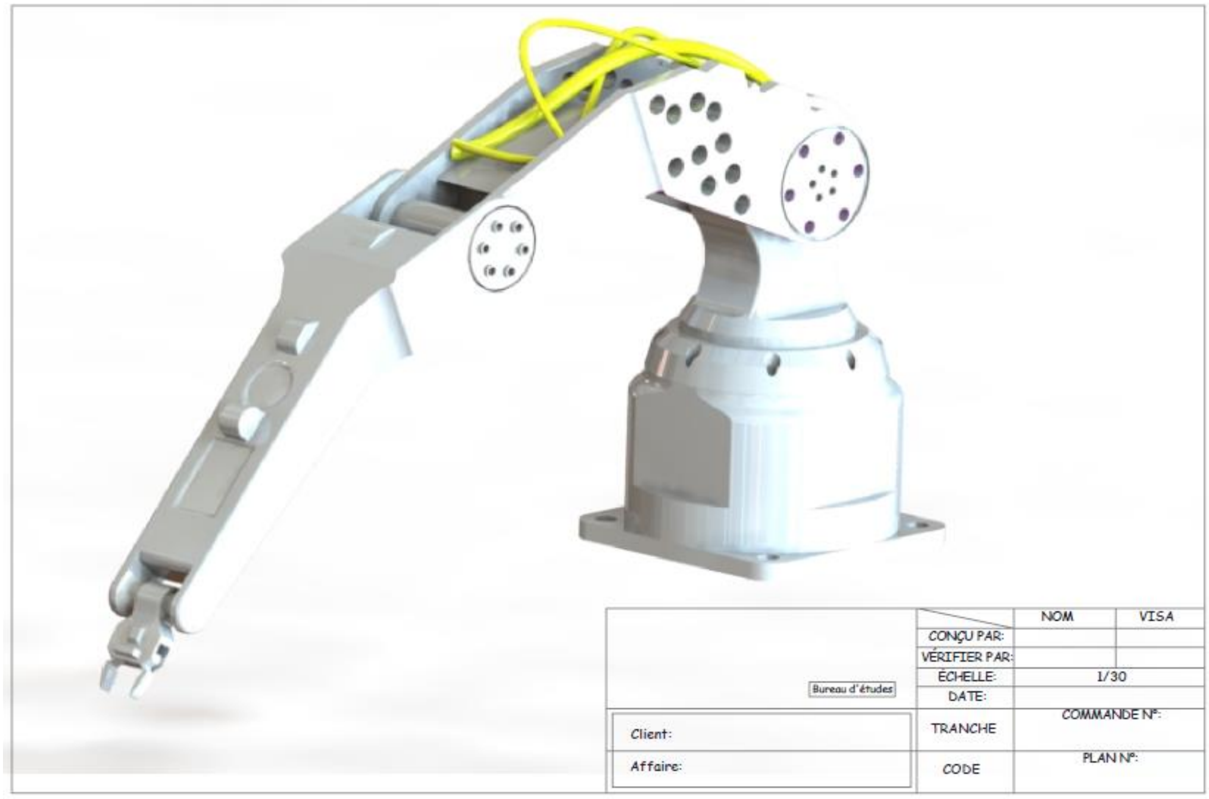
[17] Bianco, C. Guarino Lo, and A. Piazzi. "A genetic/interval approach to optimal trajectory planning of industrial robots under torque constraints." 1999 European Control Conference (ECC). IEEE, 1999.

[18] LU, Shaotian, ZHAO, Jingdong, JIANG, Li, et al. “Solving the time-jerk optimal trajectory planning problem of a robot using augmented lagrange constrained particle swarm optimization”. Mathematical Problems in Engineering, 2017, vol. 2017.

# Annexes

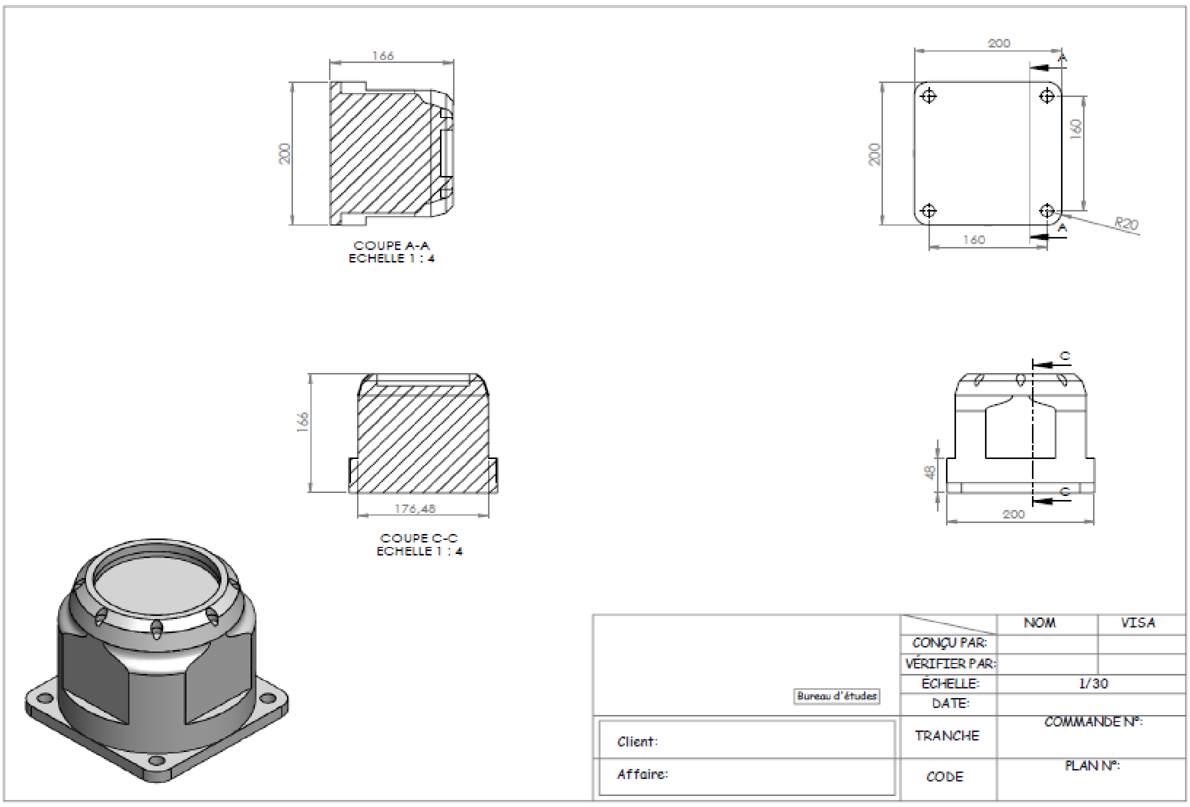


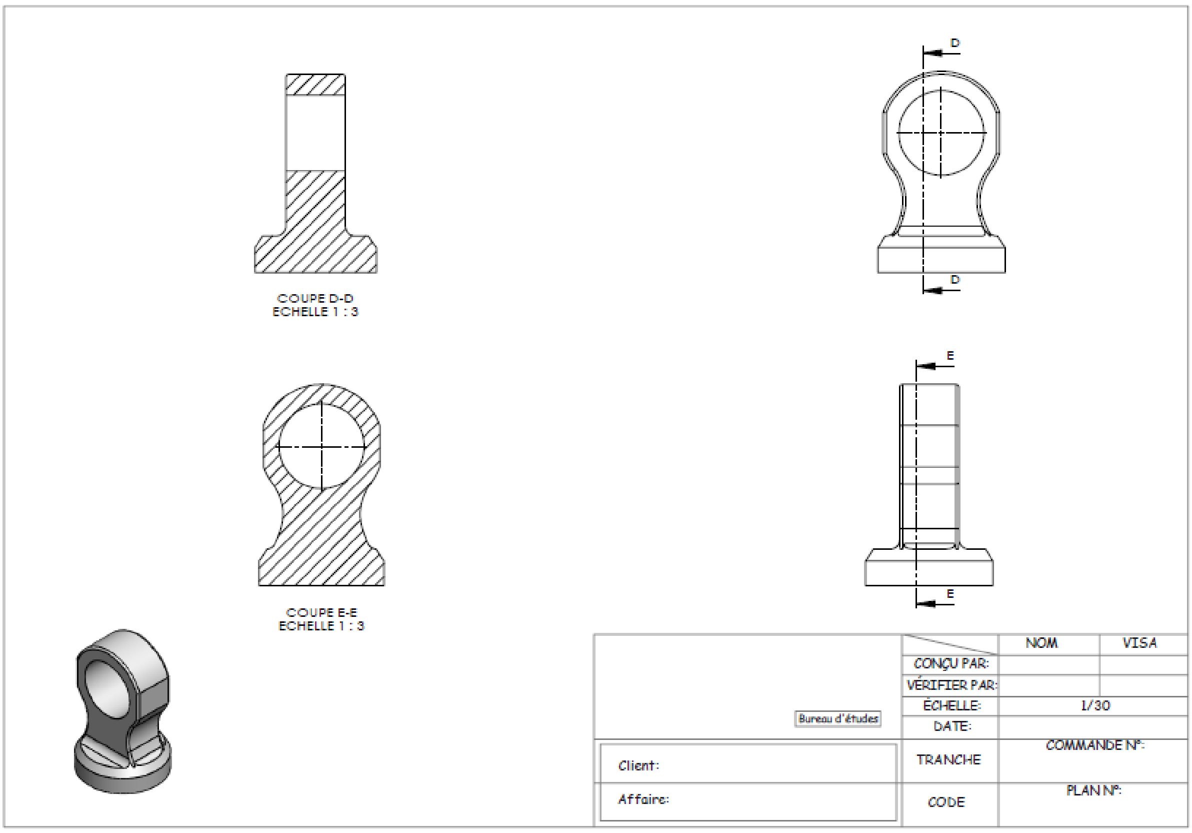
| | | NOM | VISA |
|---|---|---|---|
| | CONÇU PAR: | | |
| | VÉRIFIER PAR: | | |
| | ÉCHELLE: | 1/30 | |
| Bureau d'études | DATE: | | |
| Client: | TRANCHE | COMMANDE N°: | |
| Affaire: | CODE | PLAN N°: | |

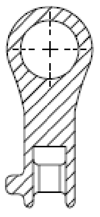


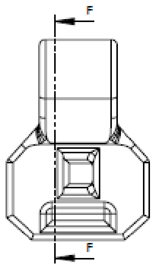
| | | NOM | VISA |
|---|---|---|---|
| | CONÇU PAR: | | |
| | VÉRIFIER PAR: | | |
| | ÉCHELLE: | 1/30 | |
| Bureau d'études | DATE: | | |
| Client: | TRANCHE | COMMANDE N°: | |
| Affaire: | CODE | PLAN N°: | |

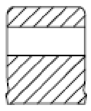
**The identification drawings of the manipulator robot's pieces:**

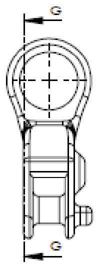


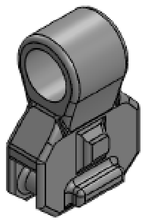



COUPE F-F
ECHELLE 1 : 1

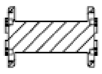


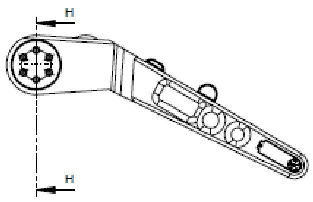

COUPE G-G
ECHELLE 1 : 1

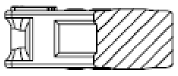


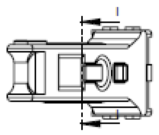


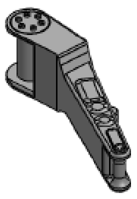
| | | NOM | VISA |
|---|---|---|---|
| Bureau d'études | CONÇU PAR: | | |
| | VÉRIFIER PAR: | | |
| | ÉCHELLE: | 1/30 | |
| | DATE: | | |
| Client: | TRANCHE | COMMANDE N°: | |
| Affaire: | CODE | PLAN N°: | |


COUPE H-H
ECHELLE 1 : 4

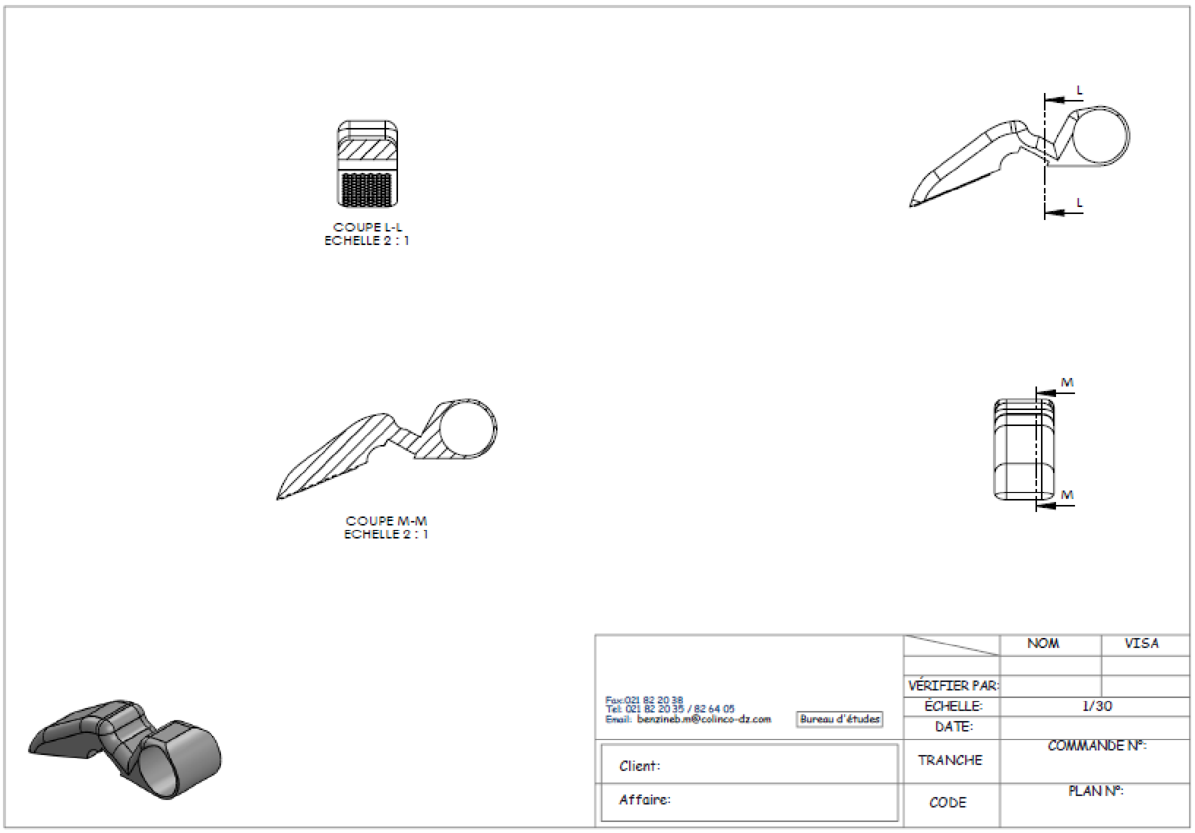



COUPE H
ECHELLE 1 : 4

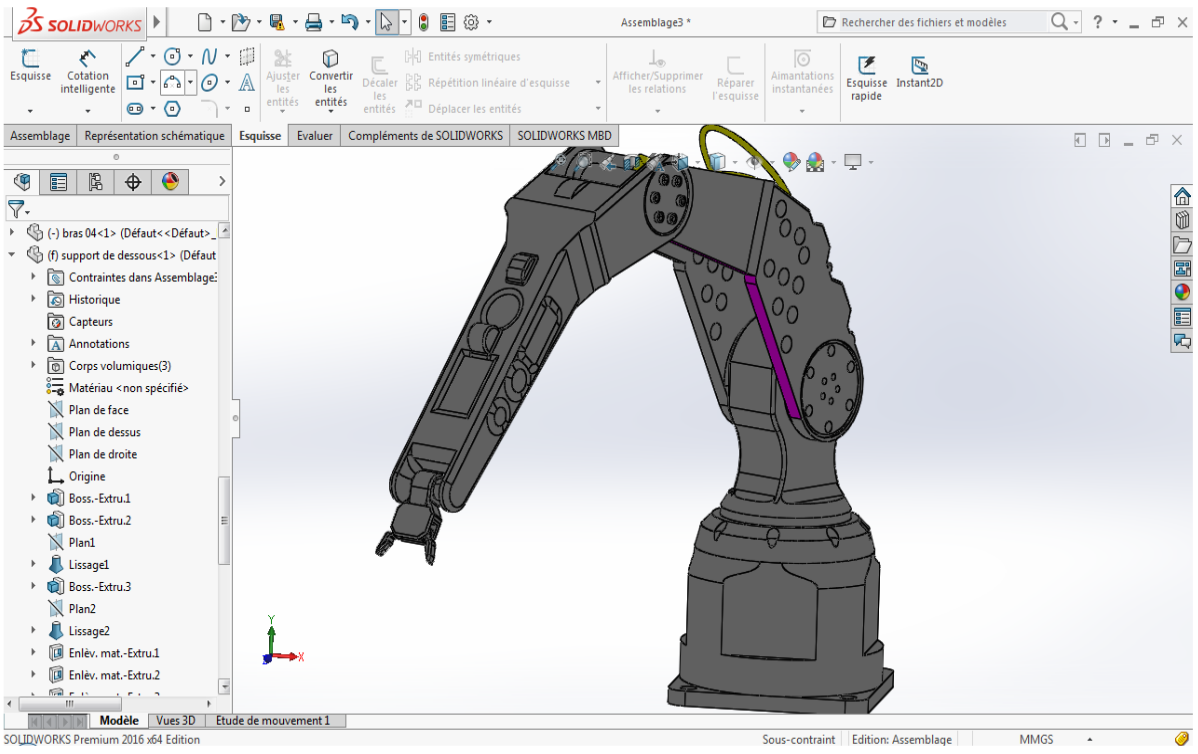


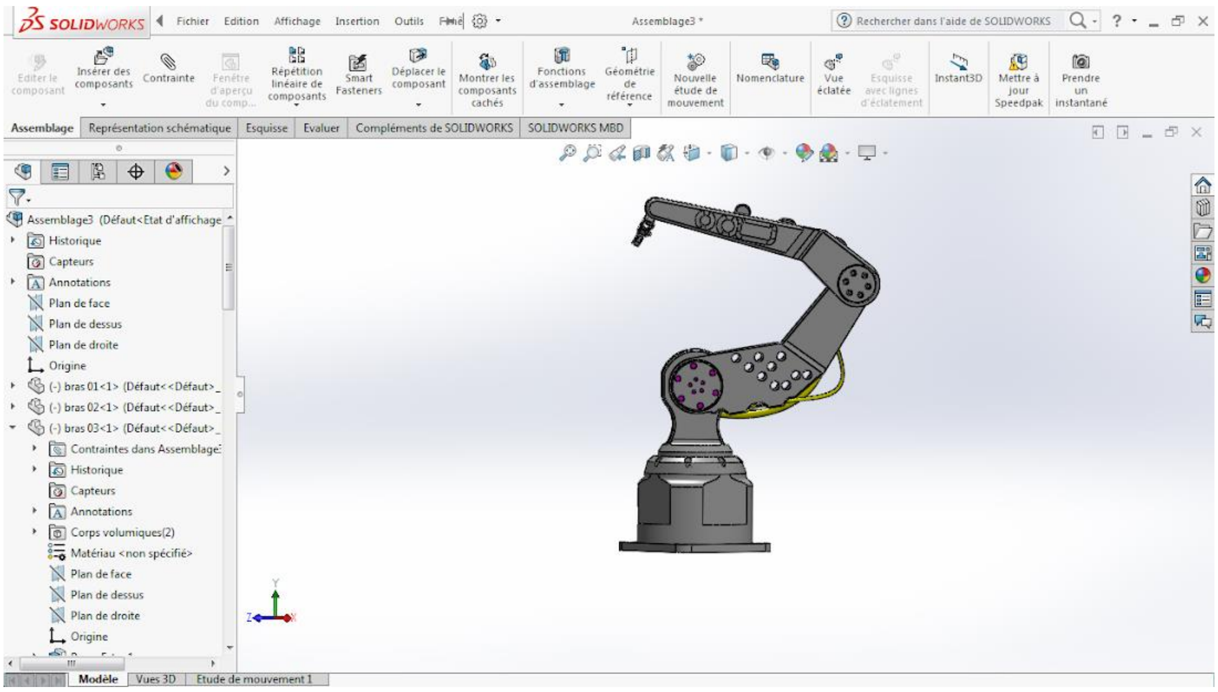


| | | NOM | VISA |
|---|---|---|---|
| Fax:021 82 20 38<br>Tel: 021 82 20 35 / 82 64 05<br>Email: benzineb.m@colinco-dz.com<br>Bureau d'études | CONÇU PAR: | | |
| | VÉRIFIER PAR: | | |
| | ÉCHELLE: | 1/30 | |
| | DATE: | | |
| Client: | TRANCHE | COMMANDE N°: | |
| Affaire: | CODE | PLAN N°: | |



**The manipulator robot arm Solid Works:**





**Zoom to the end flange of the robot (open loop):**