Information Technology Institute

Lecture 7 – Friday March 22, 2013

Trajectory Planning

Objectives

When you have finished this lecture you should be able to:

- Understand the difference between **path** and **trajectory**
- Understand the difference between **joint-space and Cartesian-space** motion descriptions.
- Understand how to generate a **sequence of movements** that must be made to create a controlled movement between motion segments, whether in straight-line motions or sequential motions.

Outline

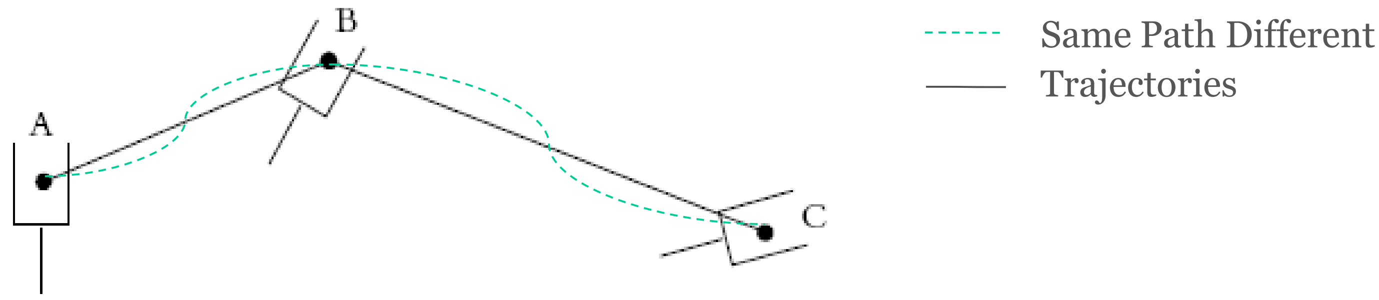
- Path versus Trajectory
- Joint-Space versus Cartesian-Space Descriptions
- Basics of Trajectory Planning
- Joint-Space Trajectory Planning
- Cartesian-Space Trajectories
- Summary

Outline

- **Path versus Trajectory**
- Joint-Space versus Cartesian-Space Descriptions
- Basics of Trajectory Planning
- Joint-Space Trajectory Planning
- Cartesian-Space Trajectories
- Summary

Path versus Trajectory

- A path is defined as the collection of **a sequence of configurations** a robot makes to go from one place to another **without** regard to the **timing** of these configurations.

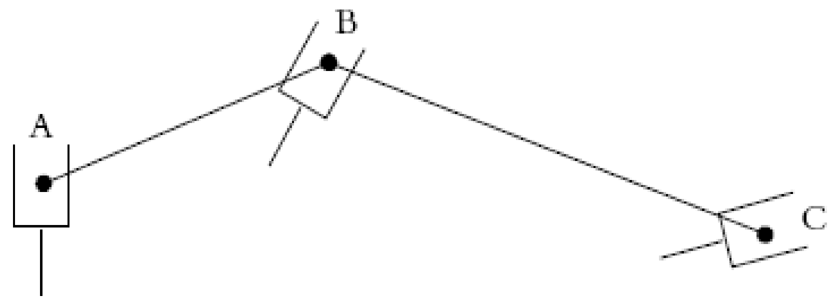


Sequential robot movements in a path

- A trajectory is related to the **timing** at which each part of the path must be attained.
- As a result, regardless of when points B and C in the figure are reached, the path is the same, whereas depending on how fast each portion of the path is traversed, the trajectory may differ.

Path versus Trajectory

- The points at which the robot may be on a path and on a trajectory at a given time may be different, even if the robot traverses the same points.
- On a **trajectory**, depending on the **velocities and accelerations**, points B and C may be reached at different times, creating different trajectories.



Sequential robot movements in a path

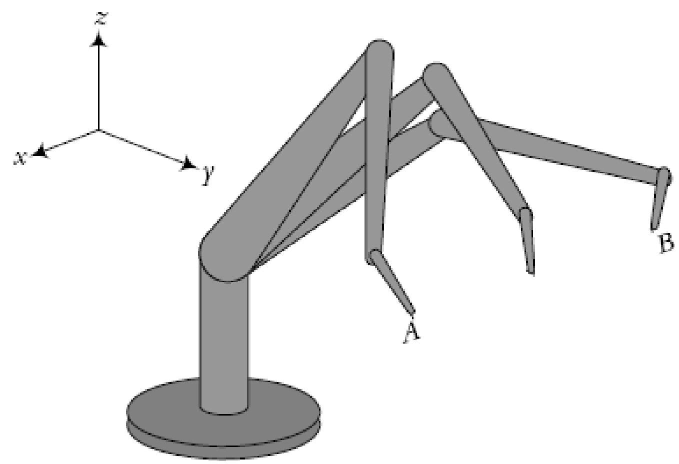
- In this lecture, we are not only concerned about the path a robot takes, but also its velocities and accelerations.

Outline

- Path versus Trajectory
- **Joint-Space versus Cartesian-Space Descriptions**
- Basics of Trajectory Planning
- Joint-Space Trajectory Planning
- Cartesian-Space Trajectories
- Summary

Joint-Space versus Cartesian-Space

- **Joint-space Motion Description**
  - ◇ The joint values calculated using **inverse kinematics** are used by the controller to drive the robot joints to their new values and, consequently, move the robot arm to its new position.



Desired End-effector Pose B

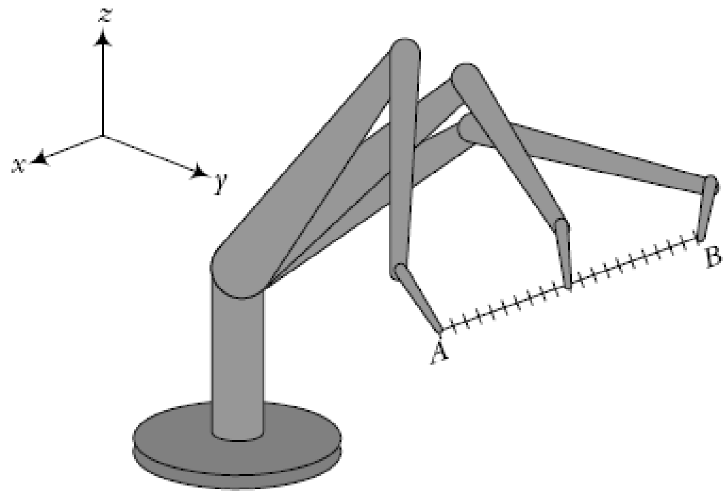
$$\begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \xrightarrow{\text{Inverse Kinematics}} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{bmatrix} \xrightarrow{\text{Joint Controllers}} \text{New Pose B}$$

Joint Values

◇ In this case, although the robot will eventually reach the **desired position**, but as we will see later, the motion between the two points is **unpredictable**.

Joint-Space versus Cartesian-Space

- **Cartesian-space Motion Description**
  - ◊ Now assume that a **straight line** is drawn between points A and B, and it is desirable to have the robot move from point A to point B in a straight line.

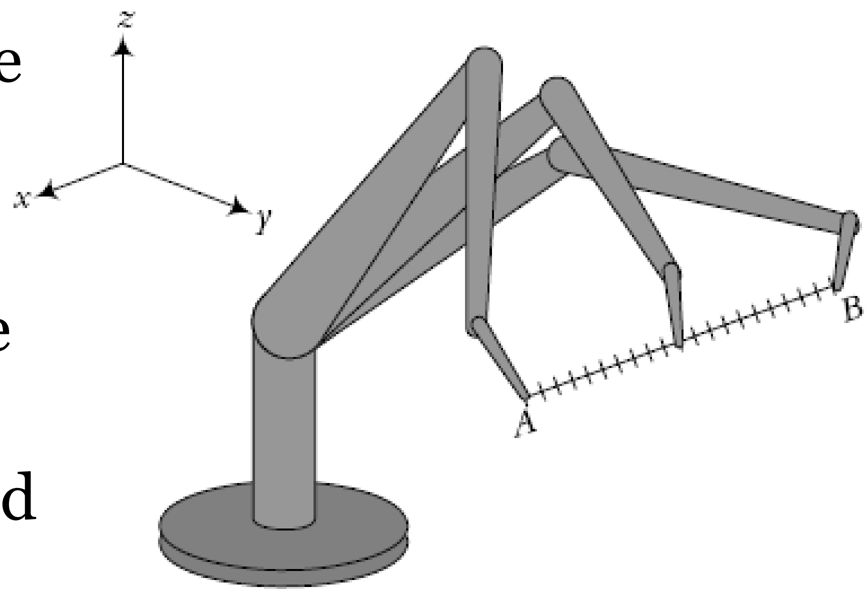


Sequential motions of a robot to follow a straight line

  - ◊ To do this, it will be necessary to **divide the line** into small **segments**, as shown below and to move the robot through all intermediate points.

Joint-Space versus Cartesian-Space

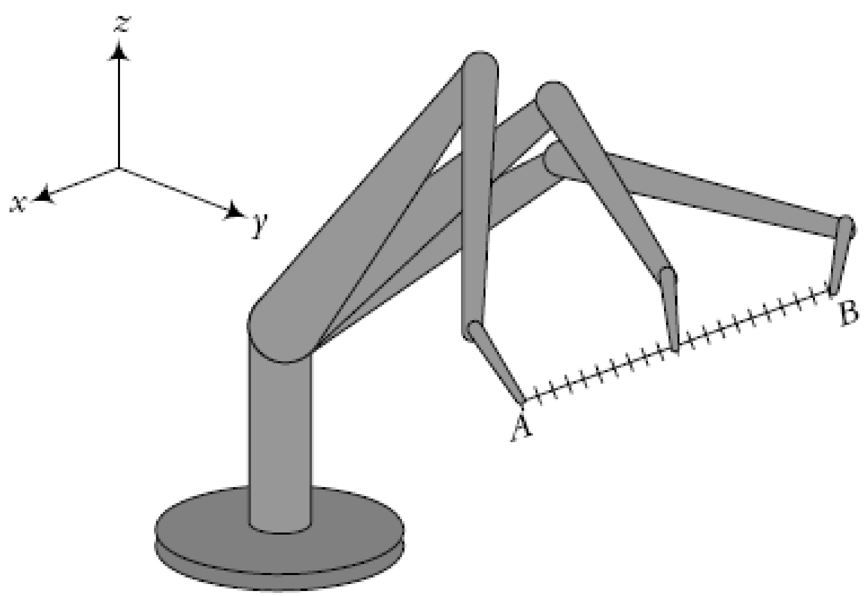
- **Cartesian-space Motion Description**
  - ◇ To accomplish this task, at each **intermediate location**, the robot's **inverse kinematic** equations are solved, a set of **joint variables** is calculated, and the controller is directed to drive the robot to those values. When **all segments** are completed, the robot will be at point B, as desired.
  - ◇ However, in this case, unlike the joint-space case, the motion is **known at all times**.
  - ◇ The sequence of movements the robot makes is described in Cartesian-space and is converted to joint-space at each segment.



Sequential motions of a robot to follow a straight line

Joint-Space versus Cartesian-Space

- **Cartesian-space Motion Description**
  - ◇ Cartesian-space trajectories are **very easy to visualize**. Since the trajectories are in the common Cartesian space in which we all operate, it is easy to visualize what the end-effector's trajectory must be.
  - ◇ However, Cartesian-space trajectories are **computationally expensive** and require a **faster processing time** for similar resolution than joint-space trajectories.

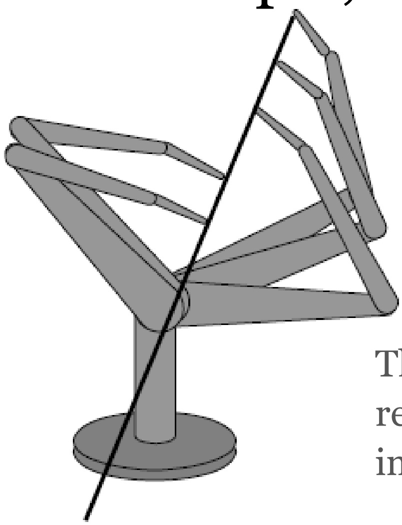


Sequential motions of a robot to follow a straight line

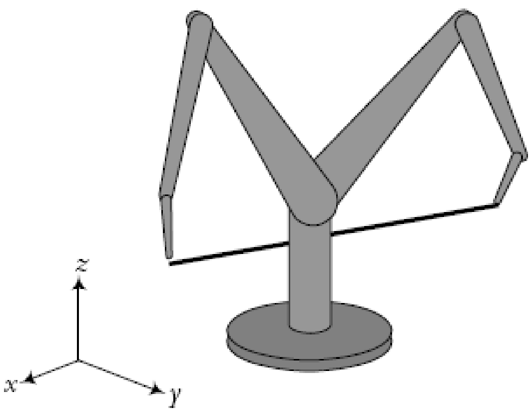
Joint-Space versus Cartesian-Space

- **Cartesian-space Motion Description**
  - ◇ Although it is easy to visualize the trajectory, it is difficult to visually ensure that **singularities** will not occur.

    For example, consider the situation shown here.



The trajectory may require a sudden change in the joint angles.



The trajectory specified in Cartesian coordinates may force the robot to run into itself.

  - ◇ If not careful, we may specify a trajectory that requires the robot to **run into itself** or to reach a point **outside of the work envelope**—which, of course, is impossible-and yields an unsatisfactory solution.

Outline

- Path versus Trajectory
- Joint-Space versus Cartesian-Space Descriptions
- **Basics of Trajectory Planning**
- Joint-Space Trajectory Planning
- Cartesian-Space Trajectories
- Summary

Basics of Trajectory Planning

- **Given:** a simple 2-DOF robot (mechanism)
- **Required:**

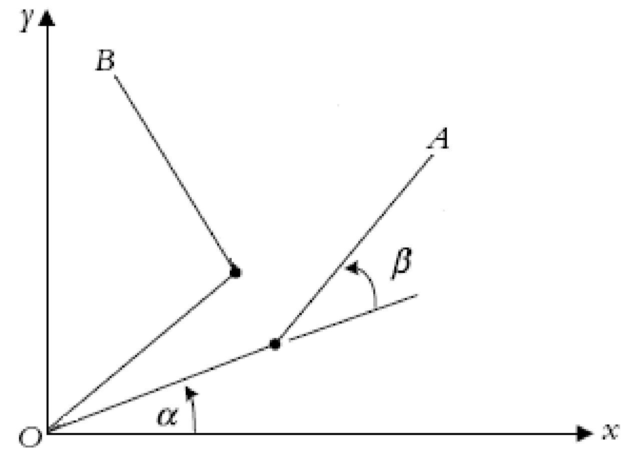
Move the robot from point A to point B.

Suppose that:

At initial point A: $\alpha=20^o$ & $\beta=30^o$.

At final point B: $\alpha=40^o$ & $\beta=80^o$.

Both joints of the robot can move at the maximum rate of 10 degrees/sec.



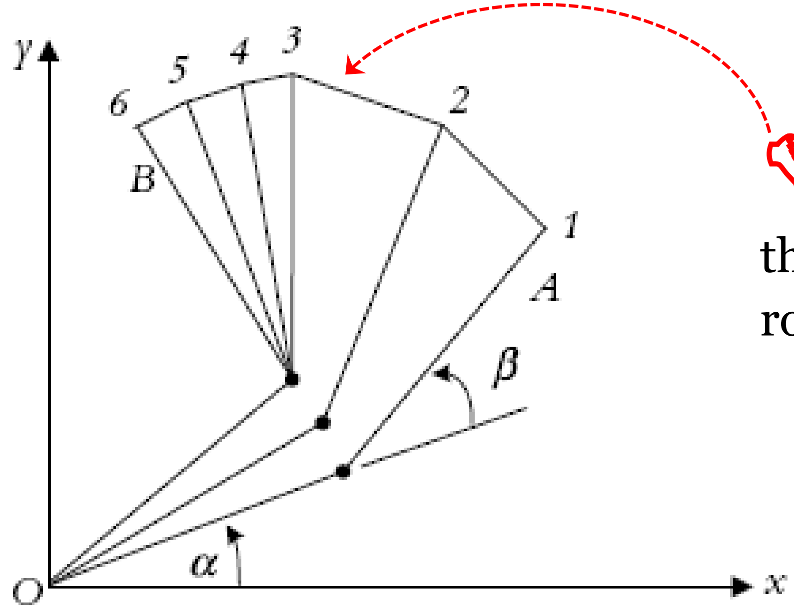
2-DOF Mechanism

Basics of Trajectory Planning

- **Joint-space, Non-normalized Movements:**

  One way to move the robot from point A to B is to run **both joints** at their **maximum angular velocities**. This means that at the end of the second time interval, the lower link of the robot will have finished its motion, while the upper link continues for another three seconds, as shown here:

Time (sec)	α	β
0	20	30
1	30	40
2	40	50
3	40	60
4	40	70
5	40	80



The path is **irregular**, and the distances traveled by the robot's end are **not uniform**.

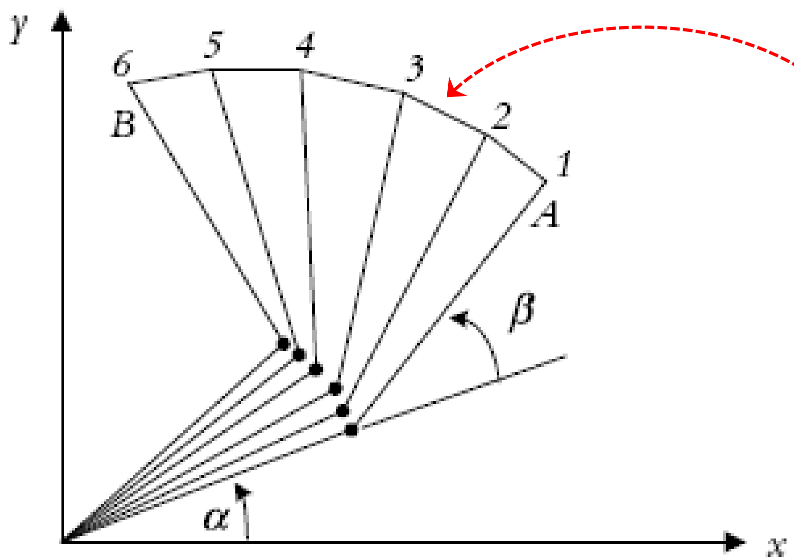
Joint-space, non-normalized movements of a 2-DOF Mechanism

Basics of Trajectory Planning

- **Joint-space, Normalized Movements:**

  The motions of both joints of the robot are normalized such that the joint with smaller motion will move proportionally slower so that both joints will start and stop their motion simultaneously. In this case, both joints move at different speeds, but move continuously together. **α changes 4 degrees/second** while **β changes 10 degrees/second**.

Time (sec)	α	β
0	20	30
1	24	40
2	28	50
3	32	60
4	36	70
5	40	80



Joint-space, normalized movements of a 2-DOF Mechanism

The **segments** of the movement are much **more similar** to each other than before, but the **path** is still **irregular** (and different from the previous case)

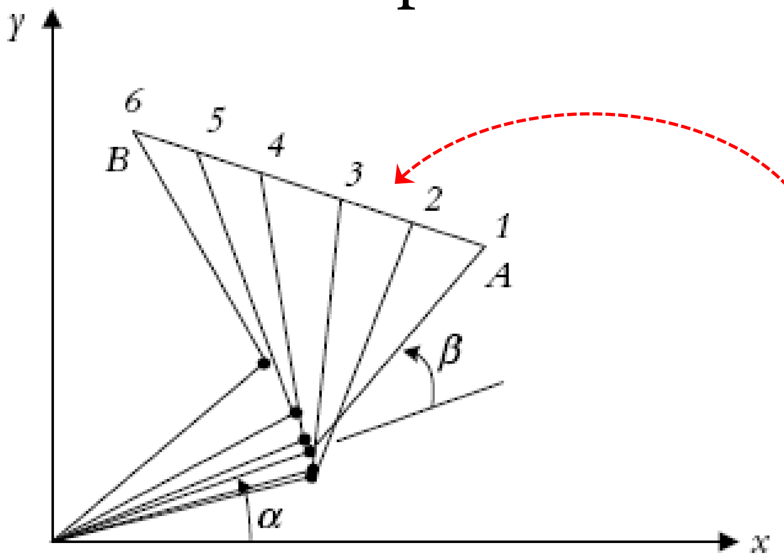
Basics of Trajectory Planning

- **Cartesian-space Movements:**

  Now suppose we want the robot's hand to follow a known path between points A and B, say, in a **straight line**.

  The simplest solution would be to draw a line between points A and B, **divide the line** into, say, **5 segments**, and solve for necessary **angles** α and β at each point. This is called **interpolation** between points A and B.

Time (sec)	α	β
0	20	30
1	14	55
2	16	69
3	21	77
4	29	81
5	40	80

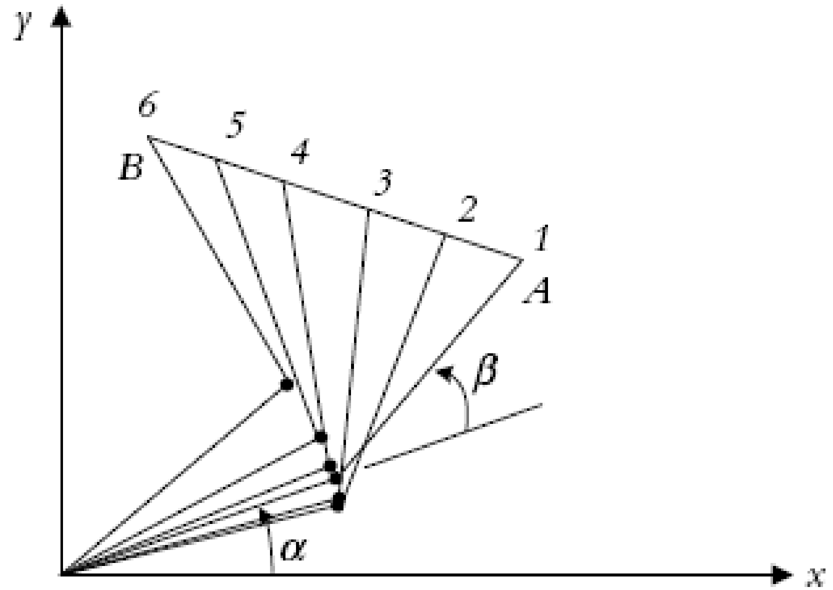


Cartesian-space movements of a 2-DOF Mechanism

The **path** is a **straight line**, but the **joint angles** are **not uniformly changing**.

Basics of Trajectory Planning

- **Cartesian-space Movements:**
  - ◊ This trajectory is in **Cartesian-space** since all segments of the motion must be calculated based on the information expressed in a **Cartesian frame**.



Cartesian-space movements

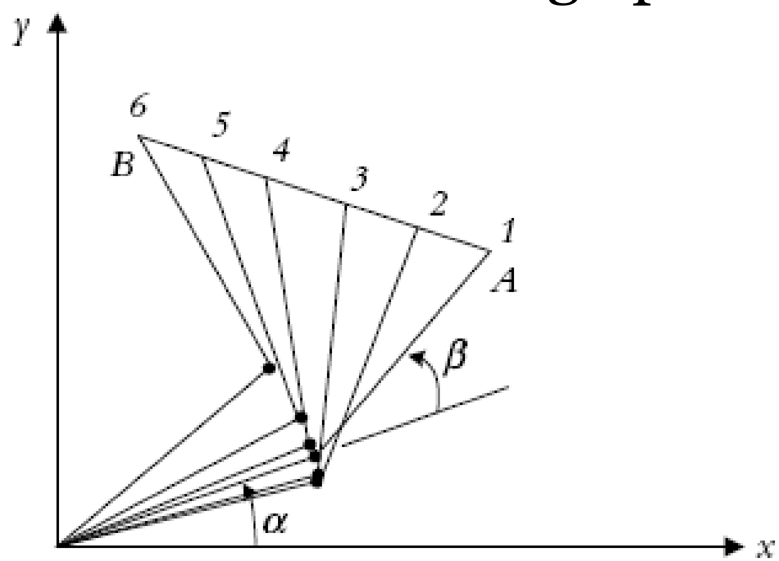
  - ◊ Although the resulting motion is a straight (and consequently, known) trajectory, it is necessary to **solve** for the **joint values at each point**.
  - ◊ Obviously, many **more points** must be calculated for **better accuracy**; with so few segments the robot will not exactly follow the lines at each segment.

Basics of Trajectory Planning

- **Cartesian-space Movements:**
  - ◇ Note how the difference between two consecutive values is **larger than** the **maximum specified joint velocity** of **10 degrees/second** (e.g., between times 0 and 1, the joint must move **25 degrees**).
  - ◇ Obviously, this is **not attainable**. Also note how, in this case, joint 1 moves downward first before moving up.

Time (sec)	α	β
0	20	30
1	14	55
2	16	69
3	21	77
4	29	81
5	40	80

!!

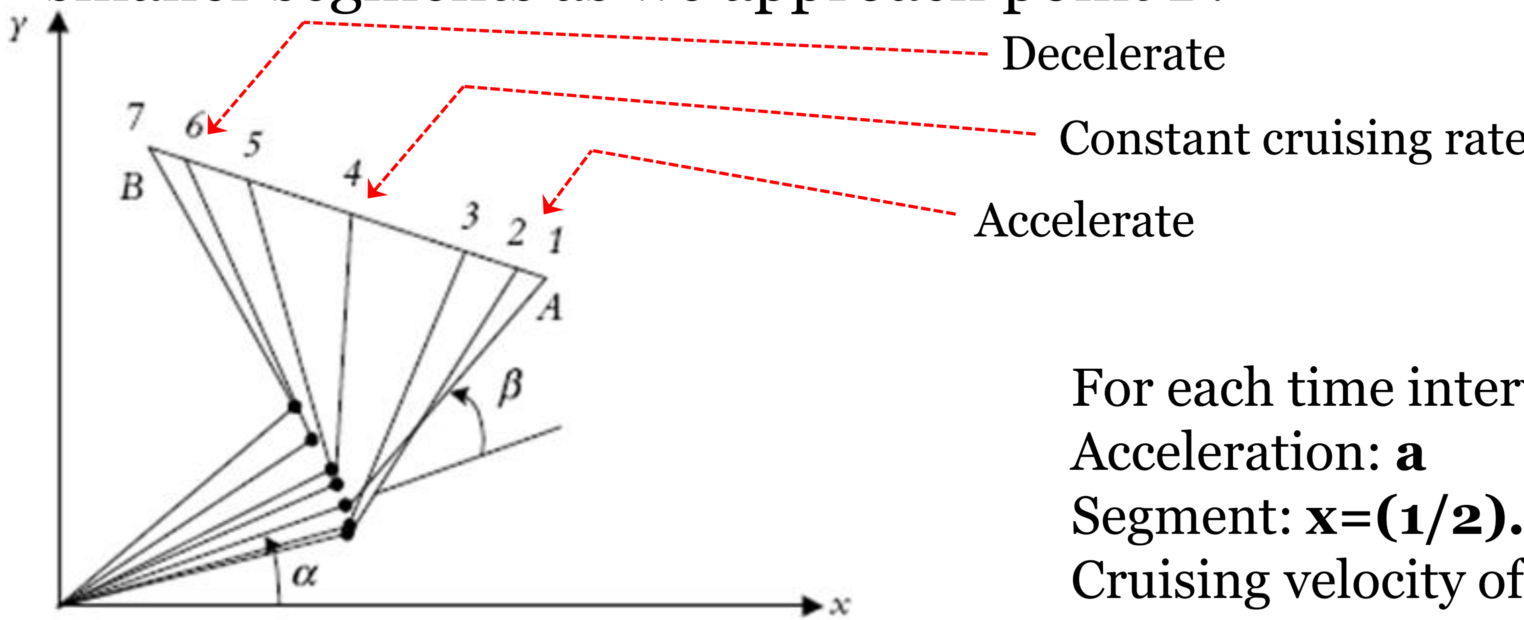


Cartesian-space movements of a 2-DOF Mechanism

Basics of Trajectory Planning

- **Cartesian-space Movements:**
  - ◇ **Trajectory planning with an acceleration/deceleration regiment:**

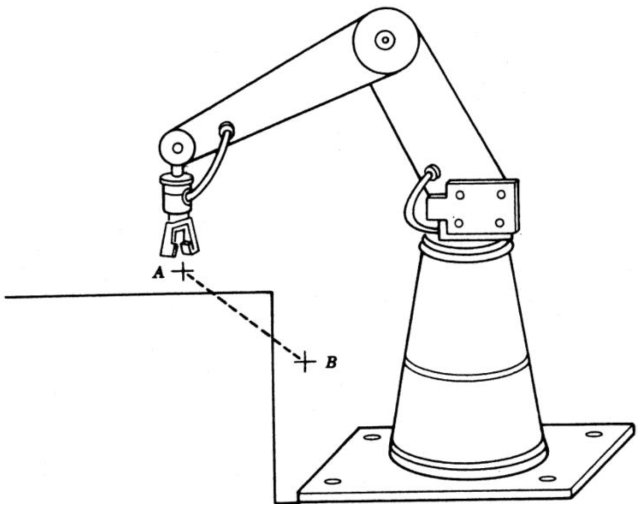
    Divide the segments differently by starting the arm with smaller segments and, as we **speed up** the arm, going at a **constant cruising rate** and finally **decelerating** with smaller segments as we approach point B.



For each time interval **t**:
Acceleration: **a**
Segment: $\mathbf{x=(1/2).at^2}$
Cruising velocity of $\mathbf{v=at}$

Basics of Trajectory Planning

- **Cartesian-space Movements:**
  - ◊ Another variation to this trajectory planning is to plan a path that is **not straight**, but one that follows some **desired path**, for example a **quadratic equation**.



A case where straight line path is not recommended.

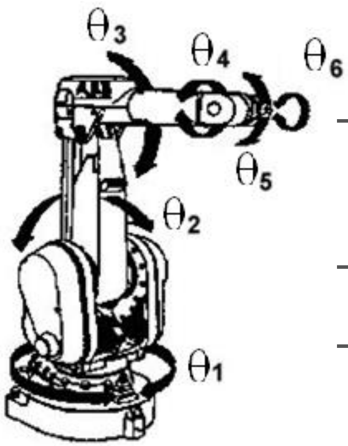
To do this, the coordinates of each segment are calculated based on the desired path and are used to calculate joint variables at each segment; therefore, the trajectory of the robot can be planned for any desired path.

Outline

- Path versus Trajectory
- Joint-Space versus Cartesian-Space Descriptions
- Basics of Trajectory Planning
- **<u>Joint-Space Trajectory Planning</u>**
- Cartesian-Space Trajectories
- Summary

Joint-Space Trajectory Planning

- **Third-Order Polynomial Trajectory Planning:**
  - ◊ In this application, the initial location and orientation of the robot are known and, using the inverse kinematic equations, the final joint angles for the desired position and orientation are found.



– Initial Location and orientation of the robot
– Initial Joint Angles
– Desired Location and orientation of the robot.

Inverse Kinematics → Final Joint Angles

  - ◊ However, the motions of **each joint** of the robot must be **planned individually**.

Joint-Space Trajectory Planning

- **Third-Order Polynomial Trajectory Planning:**
  - ◇ Consider one of the joints,

    At the beginning of the motion segment at time $\mathbf{t_i}$

    The joint angle is $\boldsymbol{\theta_i}$

θ_i following 3rd order polynomial trajectory θ_f

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

4 Unknowns: $c_o, c_1, c_2 \,\&\, c_3$

4 pieces of information:

$$\theta(t_i) = \theta_i$$

$$\theta(t_f) = \theta_f$$

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

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

Joint-Space Trajectory Planning

- **Third-Order Polynomial Trajectory Planning:**
  - ◇ Consider one of the joints,

θ_i following 3rd order polynomial trajectory θ_f

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

Taking the first derivative of the polynomial:

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

Substituting the initial and final conditions:

$$\theta(t_i) = c_o = \theta_i$$

$$\theta(t_f) = c_o + c_1 t_f + c_2 t_f^2 + c_3 t_f^3 = \theta_f$$

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

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

In matrix form

$$\begin{bmatrix} \theta_i \\ \theta_f \\ \dot{\theta}_i \\ \dot{\theta}_f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{bmatrix} c_o \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Joint-Space Trajectory Planning

- **Third-Order Polynomial Trajectory Planning:**

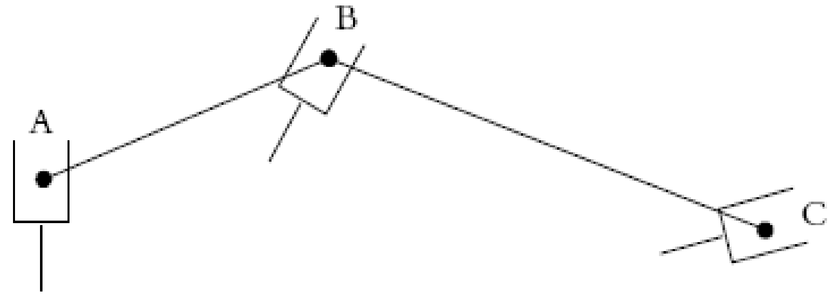
$$\begin{bmatrix} \theta_i \\ \theta_f \\ \dot{\theta}_i \\ \dot{\theta}_f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{bmatrix} c_o \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

◇ By solving these four equations simultaneously, we get the necessary values for the constants. This allows us to calculate the **joint position** at **any interval of time**, which can be used by the controller to drive the joint to position. The same process must be used for each joint individually, but they are all driven together from start to finish.

◇ Applying this third-order polynomial to each joint motion creates a **motion profile** that can be used to drive each joint.

Joint-Space Trajectory Planning

- **Third-Order Polynomial Trajectory Planning:**
  - ◊ If more than two points are specified, such that the robot will go through the points **successively**, the **final velocities and positions** at the **conclusion of each segment** can be used as the **initial values for the next segments**.



Sequential robot movements in a path

  - ◊ Similar third-order polynomials can be used to plan each section. However, although positions and velocities are continuous, accelerations are not, which may cause problems.

Joint-Space Trajectory Planning

- **Third-Order Polynomial Trajectory Planning:**
  - ◊ ***Example-1:*** It is desired to have the first joint of a 6-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.
  - ◊ ***Given:***

$$t_i = 0 \qquad \theta(t_i) = 30$$

$$t_f = 5 \qquad \theta(t_f) = 75$$

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

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

  - ◊ ***Required:***

$$\theta \text{ at } t = 1,2,3 \text{ and } 4$$

Joint-Space Trajectory Planning

- **Third-Order Polynomial Trajectory Planning:**
  - ◊ ***Example-1 (cont'd):***
  - ◊ ***Solution:*** Substituting the boundary conditions:

$$\begin{cases} \theta(t_i) = c_o = \theta_i \\ \theta(t_f) = c_o + c_1 t_f + c_2 t_f^2 + c_3 t_f^3 = \theta_f \\ \dot{\theta}(t_i) = c_1 = 0 \\ \dot{\theta}(t_f) = c_1 + 2c_2 t_f + 3c_3 t_f^2 = 0 \end{cases} \rightarrow \begin{cases} \theta(t_i) = c_o = 30 \\ \theta(t_f) = c_o + c_1(5) + c_2(5)^2 + c_3(5)^3 = 75 \\ \dot{\theta}(t_i) = c_1 = 0 \\ \dot{\theta}(t_f) = c_1 + 2c_2(5) + 3c_3(5)^2 = 0 \end{cases}$$

$$\downarrow$$

$$c_o = 30, c_1 = 0, c_2 = 5.4, c_3 = -0.72$$

Joint-Space Trajectory Planning

- **Third-Order Polynomial Trajectory Planning:**
  - ◊ ***Example-1 (cont'd):*** This results in the following cubic polynomial equation for position as well as the velocity and acceleration equations for joint 1:

$$\theta(t) = 30 + 5.4t^2 - 0.72t^3$$

$$\dot{\theta}(t) = 10.84t - 2.16t^2$$

$$\ddot{\theta}(t) = 10.84 - 4.32t$$

Substituting the desired time intervals into the motion equation will result in:

$$\theta(1) = 34.68^o, \quad \theta(2) = 45.84^o, \quad \theta(3) = 59.16^o, \quad \theta(4) = 70.32^o$$

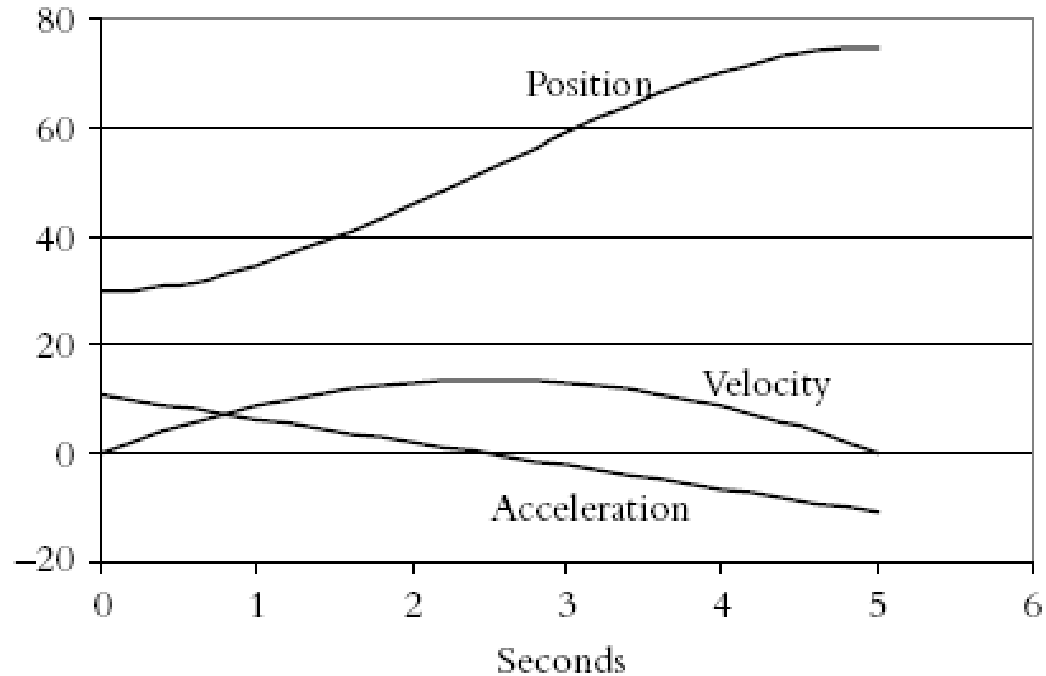
Joint-Space Trajectory Planning

- **Third-Order Polynomial Trajectory Planning:**
  - ◇ ***Example-1 (cont'd):*** The joint angles, velocities, and accelerations are shown below. Notice that in this case, the acceleration needed at the beginning of the motion is 10.8°/sec² (as well as -10.8°/sec² deceleration at the conclusion of the motion).

$$\theta(t) = 30 + 5.4t^2 - 0.72t^3$$

$$\dot{\theta}(t) = 10.84t - 2.16t^2$$

$$\ddot{\theta}(t) = 10.84 - 4.32t$$



Joint positions, velocities, and accelerations

Joint-Space Trajectory Planning

- **Fifth-Order Polynomial Trajectory Planning:**
  - ◊ Specifying the initial and ending positions, velocities, and accelerations of a segment yields **six pieces of information**, enabling us to use a **fifth-order polynomial** to plan a trajectory, as follows:

$$\theta(t) = c_o + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 + c_5 t^5$$

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

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

These equations allow us to calculate the coefficients of a fifth-order polynomial with position, velocity, and acceleration boundary conditions.

Joint-Space Trajectory Planning

- **Fifth-Order Polynomial Trajectory Planning:**
  - ◊ ***Example-2:*** Repeat Example-1, but assume the initial acceleration and final deceleration will be $5^o/\sec^2$.
  - ◊ ***Solution:*** From Example-1 and the given accelerations, we have:

$$\theta_i = 30^o \quad \dot{\theta}_i = 0^o/\sec \quad \ddot{\theta}_i = 5^o/\sec^2$$

$$\theta_f = 75^o \quad \dot{\theta}_f = 0^o/\sec \quad \ddot{\theta}_f = -5^o/\sec^2$$

Substituting in the following equations will result in:

$$\begin{cases} \theta(t) = c_o + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 + c_5 t^5 \\ \dot{\theta}(t) = c_1 + 2c_2 t + 3c_3 t^2 + 4c_4 t^3 + 5c_5 t^4 \\ \ddot{\theta}(t) = 2c_2 + 6c_3 t + 12c_4 t^2 + 20c_5 t^3 \end{cases} \rightarrow \begin{array}{lll} c_o = 30 & c_1 = 0 & c_2 = 2.5 \\ c_3 = 1.6 & c_4 = -0.58 & c_5 = 0.0464 \end{array}$$

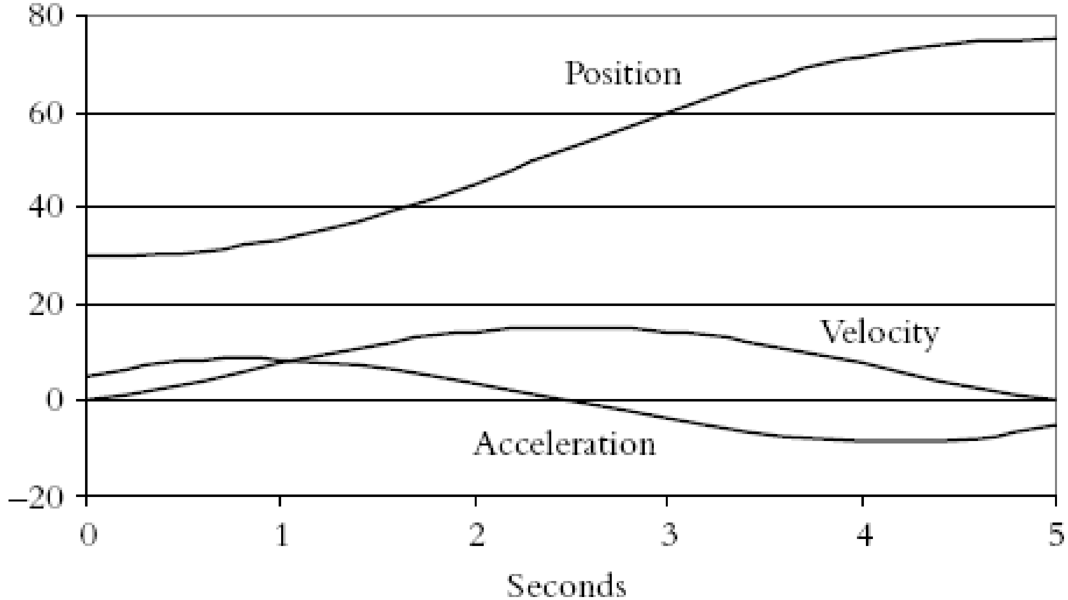
Joint-Space Trajectory Planning

- **Fifth-Order Polynomial Trajectory Planning:**
  - ◊ ***Example-2 (cont'd):*** This results in the following motion equations:

$$\theta(t) = 30 + 2.5t^2 + 1.6t^3 - 0.58t^4 + 0.0464\,t^5$$

$$\dot{\theta}(t) = 5t + 4.8t^2 - 2.32t^3 + 0.232\,t^4$$

$$\ddot{\theta}(t) = 5 + 9.6t - 6.9t^2 + 0.928\,t^3$$



Joint positions, velocities, and accelerations

Joint-Space Trajectory Planning

- **Fifth-Order Polynomial Trajectory Planning:**
  - ◊ To ensure that the robot's accelerations will not exceed its capabilities, acceleration limits may be used to calculate the necessary time to reach the target.

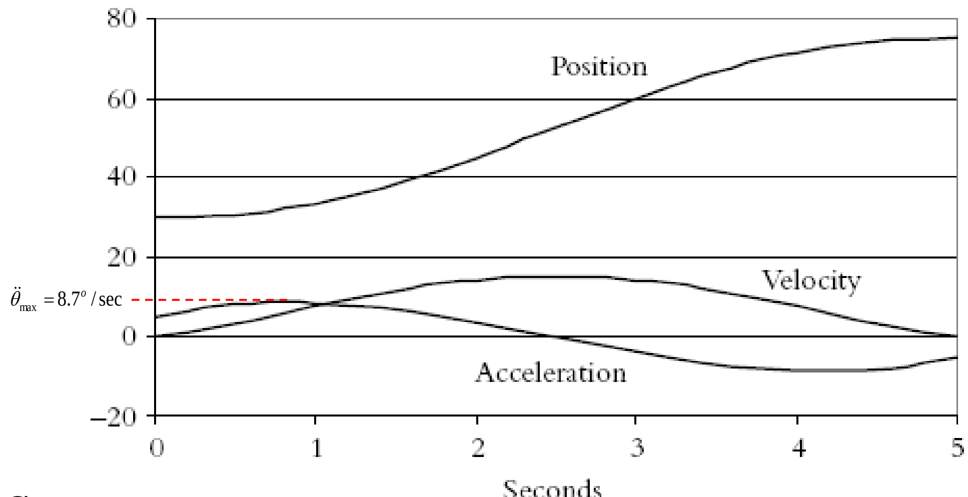
For $\dot{\theta}_i = 0$ and $\dot{\theta}_f = 0$

$$\left|\ddot{\theta}\right|_{\max} = \left|\frac{6(\theta_f - \theta_i)}{(t_f - t_i)^2}\right|$$

In example-2:

$$\left|\ddot{\theta}\right|_{\max} = \left|\frac{6(75-30)}{(5-0)^2}\right| = 10.8$$

The maximum acceleration is $8.7^{o}/\sec^2 < \left|\ddot{\theta}\right|_{\max}$



Joint positions, velocities, and accelerations

Outline

- Path versus Trajectory
- Joint-Space versus Cartesian-Space Descriptions
- Basics of Trajectory Planning
- Joint-Space Trajectory Planning
- **Cartesian-Space Trajectories**
- Summary

Cartesian-Space Trajectories

- Cartesian-space trajectories relate to the motions of a robot relative to the Cartesian reference frame, as followed by the **position and orientation of the robot's hand**.
- In addition to simple **straight-line trajectories**, many other schemes may be deployed to drive the robot in its path between different points.
- In fact, **all of the schemes** used for joint-space trajectory planning can also be used for Cartesian-space trajectories.
- The basic difference is that for Cartesian-space, the joint values must be **repeatedly calculated** through the **inverse kinematic** equations of the robot.

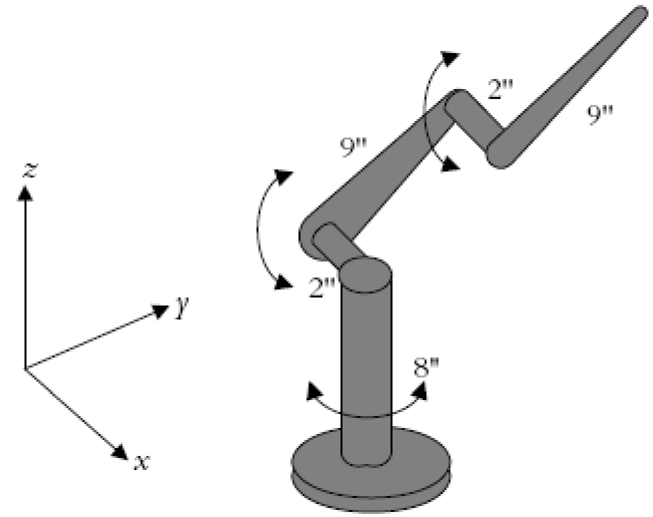
Cartesian-Space Trajectories

- **Procedure:**

1. Increment the time by **$t=t+\Delta t$**.
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.

Cartesian-Space Trajectories

- ***Example:*** A 3-DOF robot designed for lab experimentation has two links, each 9 inches long. As shown in the figure, the coordinate frames of the joints are such that when all angles are zero, the arm is pointed upward.



The inverse kinematic equations of the robot are also given below.

We want to move the robot from point **(9,6,10)** to point **(3,5,8)** along a **straight line**.

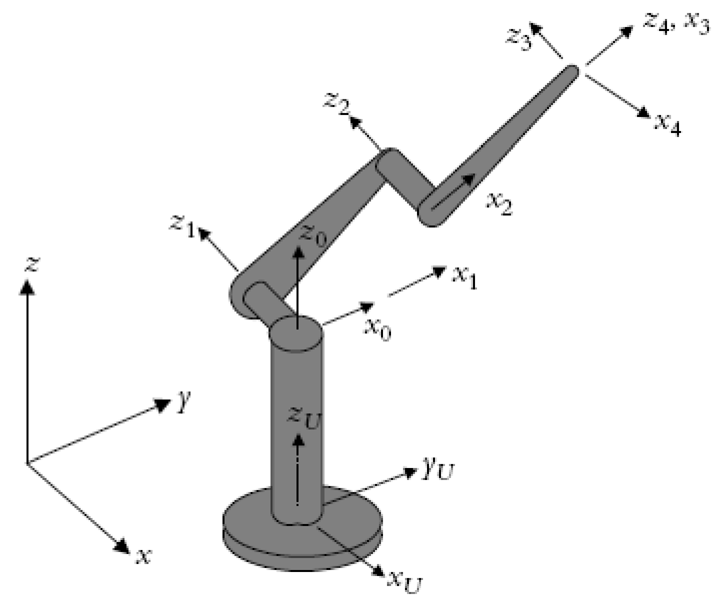
Find the angles of the three joints for each intermediate point and plot the results.

Cartesian-Space Trajectories

- ***Example (cont'd):***

Given:

$$A(9,6,10) \xrightarrow{\text{straight line}} B(3,5,8)$$



Inverse Kinematics Solution

$$\theta_1 = \tan^{-1}(P_x / P_y)$$

$$\theta_3 = \cos^{-1}\left[\left(\left(P_y / C_1 \right)^2 + \left(P_z - 8 \right)^2 - 162 \right) \Big/ 162 \right]$$

$$\theta_2 = \cos^{-1}\left[\left(C_1 \left(P_z - 8 \right)\left(1 + C_3 \right) + P_y S_3 \right) \Big/ \left(18 \left(1 + C_3 \right) C_1 \right) \right]$$

Required:

Angles of the three joints for each intermediate point and plot the results.

Cartesian-Space Trajectories

- ***Example (cont'd):*** **Solution**

We **divide the distance** between the start and the end points into **10 segments**, although in reality, it is divided into many more sections. The coordinates of each intermediate point are found by dividing the distance between the initial and the end points into **10 equal parts**.

Straight line equation between point (x_1,y_1,z_1) and point (x_2,y_2,z_2) is

$$\frac{x-x_1}{x_2-x_1}=\frac{y-y_1}{y_2-y_1}=\frac{z-z_1}{z_2-z_1}$$

$$\frac{x-9}{3-9}=\frac{y-6}{5-6}=\frac{z-10}{8-10}$$

$$(x-9)/6=y-6=0.5(z-10)$$

The Hand Frame Coordinates and Joint Angles for the Robot

P_x	P_y	P_z	θ_1	θ_2	θ_3
9	6	10	56.3	27.2	104.7
8.4	5.9	9.8	54.9	25.4	109.2
7.8	5.8	9.6	53.4	23.8	113.6
7.2	5.7	9.4	51.6	22.4	117.9
6.6	5.6	9.2	49.7	21.2	121.9
6	5.5	9	47.5	20.1	125.8
5.4	5.4	8.8	45	19.3	129.5
4.8	5.3	8.6	42.2	18.7	133
4.2	5.2	8.4	38.9	18.4	136.3
3.6	5.1	8.2	35.2	18.5	139.4
3	5	8	31	18.9	142.2

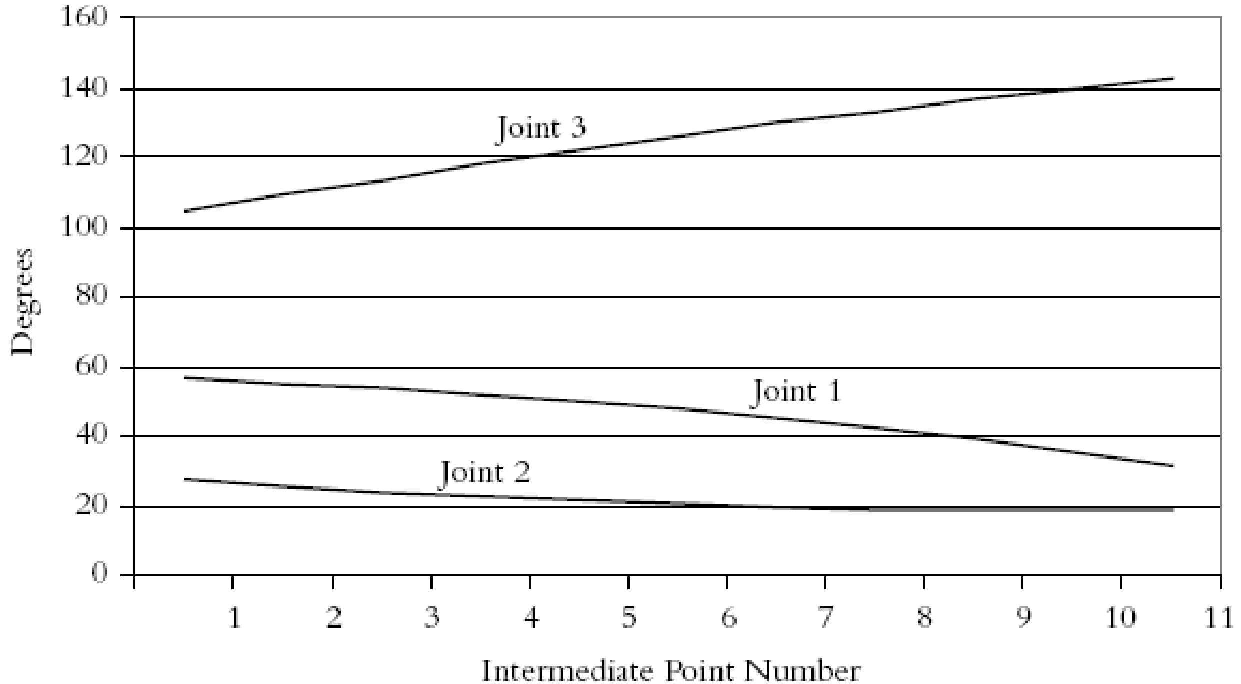
Cartesian-Space Trajectories

- ***Example (cont'd):***

Solution:

The **inverse kinematic** equations are used to calculate the **joint angles** for **each intermediate point**, as shown in the table.

The **joint angles** are shown here.



Joint angles

Outline

- Path versus Trajectory
- Joint-Space versus Cartesian-Space Descriptions
- Basics of Trajectory Planning
- Joint-Space Trajectory Planning
- Cartesian-Space Trajectories
- **Summary**

Summary

- Trajectories may be planned in joint-space or in Cartesian-space.
- Trajectories in each space may be planned through a number of different methods. Many of these methods may actually be used for both the Cartesian-space and the joint-space. However, although Cartesian-space trajectories are more realistic and can be visualized more easily, they are more difficult to calculate and plan.
- Obviously, a specific path such as a straight-line motion must be planned in Cartesian-space to be straight. But if the robot is not to follow a specific path, joint-space trajectories are easier to calculate and generate.