Robot Trajectory Planning
Robot control methods

1- Lead-through programming

• The human operator physically grabs the end-effector and shows the robot exactly what motions to make for a task, while the computer memorizes the motions (memorizing the joint positions, lengths and/or angles, to be played back during task execution).
2- Teach programming

- Move robot to required task positions via teach pendant; computer memorizes these configurations and plays them back in robot motion sequence. The teach pendant is a controller box that allows the human operator to position the robot by manipulating the buttons on the box. This type of control is adequate for simple, non-intelligent tasks.
3- Off-line programming
• Use of computer software with realistic graphics to plan and program motions without use of robot hardware (such as IGRIP).

4- Autonomous
• Controlled by computer, with sensor feedback, without human intervention. Computer control is required for intelligent robot control. In this type of control, the computer may send the robot preprogrammed positions and even manipulate the speed and direction of the robot as it moves, based on sensor feedback. The computer can also communicate with other devices to help guide the robot through its tasks.

5- Teleoperation
• Human-directed motion, via a joystick. Special joysticks that allow the human operator to feel what the robot feels are called haptic interfaces.

6- Telerobotic
• Combination of autonomous and teleoperation.
Control Circuit

Fig. 7.1 Basic structure of a feedback control system.

Block diagram of simplified open loop system with effective inertia and damp-

set-point tracking problem is now the problem of tracking a constant or step reference command $\theta^d$ while rejecting a constant disturbance, $d$. 
PD Compensator

\[
U(s) = K_p (\Theta^d(s) - \Theta(s)) - K_D s \Theta(s)
\]
Saturation

- Many manipulators incorporate current limiters in the servo-system to prevent damage that might result from overdrawing current.
PID Compensator

Fig. 7.14 Closed loop system with PID control.

\[ C(s) = K_P + K_D s + \frac{K_I}{s} \]
Fig. 7.18 Feedforward control scheme.
Fig. 7.21  Feedforward computed torque compensation.
Robot Motion Planning

- **Path planning**
  - Geometric path
  - Issues: obstacle avoidance, shortest path

- **Trajectory planning,**
  - “interpolate” or “approximate” the desired path by a class of polynomial functions and generates a sequence of time-based “control set points” for the control of manipulator from the initial configuration to its destination.
Path Plan
Trajectory Planning

Path constraints

(continuity, smoothness)

Path specification

Trajectory Planner

{\{q(t), \dot{q}(t), \ddot{q}(t)\}}

or

{\{p(t), v(t), a(t)\}}

cartesian space

sequence of control set points along desired trajectory
Trajectory planning

- Path Profile
- Velocity Profile
- Acceleration Profile
The boundary conditions

1) Initial position
2) Initial velocity
3) Initial acceleration
4) Lift-off position
5) Continuity in position at $t_1$
6) Continuity in velocity at $t_1$
7) Continuity in acceleration at $t_1$
8) Set-down position
9) Continuity in position at $t_2$
10) Continuity in velocity at $t_2$
11) Continuity in acceleration at $t_2$
12) Final position
13) Final velocity
14) Final acceleration
Requirements

• Initial Position
  – Position (given)
  – Velocity (given, normally zero)
  – Acceleration (given, normally zero)

• Final Position
  – Position (given)
  – Velocity (given, normally zero)
  – Acceleration (given, normally zero)
Requirements

• Intermediate positions
  – set-down position (given)
  – set-down position (continuous with previous trajectory segment)
  – Velocity (continuous with previous trajectory segment)
  – Acceleration (continuous with previous trajectory segment)
Requirements

• Intermediate positions
  – Lift-off position (given)
  – Lift-off position (continuous with previous trajectory segment)
  – Velocity (continuous with previous trajectory segment)
  – Acceleration (continuous with previous trajectory segment)
Trajectory Planning

• n-th order polynomial, must satisfy 14 conditions,

• 13-th order polynomial

\[ a_{13}t^{13} + \cdots + a_2 t^2 + a_1 t + a_0 = 0 \]

• 4-3-4 trajectory

\[
\begin{align*}
    h_1(t) &= a_{14} t^4 + a_{13} t^3 + a_{12} t^2 + a_{12} t + a_{10} \\
    h_2(t) &= a_{23} t^3 + a_{22} t^2 + a_{21} t + a_{20} \\
    h_n(t) &= a_{n4} t^4 + a_{n3} t^3 + a_{n2} t^2 + a_{n2} t + a_{n0}
\end{align*}
\]

t_0 \rightarrow t_1, 5 unknown

t_1 \rightarrow t_2, 4 unknown

t_2 \rightarrow t_f, 5 unknown

• 3-5-3 trajectory
Trajectories for Point to Point Motion

- As described above, the problem here is to find a trajectory that connects an initial to a final configuration while satisfying other specified constraints at the endpoints (e.g., velocity and/or acceleration constraints).
- Without loss of generality, we will consider planning the trajectory for a single joint, since the trajectories for the remaining joints will be created independently and in exactly the same way.
- Thus, we will concern ourselves with the problem of determining $q(t)$, where $q(t)$ is a scalar joint variable.
We suppose that at time $t_0$ the joint variable satisfies
\[
q(t_0) = q_0 \\
\dot{q}(t_0) = \nu_0
\]
and we wish to attain the values at $t_f$
\[
q(t_f) = q_f \\
\dot{q}(t_f) = \nu_f
\]
two the additional equations
\[
\ddot{q}(t_0) = \alpha_0 \\
\ddot{q}(t_f) = \alpha_f
\]
Cubic Polynomial Trajectories

we consider a cubic trajectory of the form

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

Then the desired velocity is given as

$$\dot{q}(t) = a_1 + 2a_2 t + 3a_3 t^2$$

with the four constraints yields

$$q_0 = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3$$
$$v_0 = a_1 + 2a_2 t_0 + 3a_3 t_0^2$$
$$q_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3$$
$$v_f = a_1 + 2a_2 t_f + 3a_3 t_f^2$$
These four equations can be combined into a single matrix equation

\[
\begin{bmatrix}
1 & t_0 & t_0^2 & t_0^3 \\
0 & 1 & 2t_0 & 3t_0^2 \\
1 & t_f & t_f^2 & t_f^3 \\
0 & 1 & 2t_f & 3t_f^2
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
a_3
\end{bmatrix}
=
\begin{bmatrix}
q_0 \\
v_0 \\
q_f \\
v_f
\end{bmatrix}
\]
Example

Suppose we take \( t_0 = 0 \) and \( t_f = 1 \) sec, with \( v_0 = 0 \quad v_f = 0 \)

then equivalent to the four equations

\[
\begin{align*}
    a_0 &= q_0 \\
    a_1 &= 0 \\
    a_2 + a_3 &= q_f - q_0 \\
    2a_2 + 3a_3 &= 0
\end{align*}
\]

The required cubic polynomial function is therefore

\[
q_i(t) = q_0 + 3(q_f - q_0)t^2 - 2(q_f - q_0)t^3
\]

with \( q_0 = 10^\circ \), \( q_f = -20^\circ \).

---

**Fig. 5.13**  (a) Cubic polynomial trajectory (b) Velocity profile for cubic polynomial trajectory (c) Acceleration profile for cubic polynomial trajectory
Quintic Polynomial Trajectories

• a cubic trajectory gives continuous positions and velocities at the start and finish points times but discontinuities in the acceleration. The derivative of acceleration is called the jerk. A discontinuity in acceleration leads to an impulsive jerk, which may excite vibrational modes in the manipulator and reduce tracking accuracy.

• For this reason, one may wish to specify constraints on the acceleration as well as on the position and velocity. In this case, we have six constraints (one each for initial and final configurations, initial and final velocities, and initial and final accelerations). Therefore we require a fifth order polynomial

\[ q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \]
\[ q_0 = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 + a_4 t_0^4 + a_5 t_0^5 \]
\[ v_0 = a_1 + 2a_2 t_0 + 3a_3 t_0^2 + 4a_4 t_0^3 + 5a_5 t_0^4 \]
\[ \alpha_0 = 2a_2 + 6a_3 t_0 + 12a_4 t_0^2 + 20a_5 t_0^3 \]
\[ q_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5 \]
\[ v_f = a_1 + 2a_2 t_f + 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4 \]
\[ \alpha_f = 2a_2 + 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3 \]

which can be written as

\[
\begin{bmatrix}
1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\
0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\
0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\
1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\
0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\
0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
a_3 \\
a_4 \\
a_5 
\end{bmatrix}
= 
\begin{bmatrix}
q_0 \\
v_0 \\
\alpha_0 \\
q_f \\
v_f \\
\alpha_f 
\end{bmatrix}
\]
Ex:
a quintic polynomial trajectory with $q(0) = 0$, $q(2) = 40$ with zero initial and final velocities and accelerations.

Fig. 5.15  (a) Quintic Polynomial Trajectory. (b) Velocity Profile for Quintic Polynomial Trajectory. (c) Acceleration Profile for Quintic Polynomial Trajectory
Linear Segments with Parabolic Blends (LSPB)

• Another way to generate suitable joint space trajectories is by so-called Linear Segments with Parabolic Blends or (LSPB) for short. This type of trajectory is appropriate when a constant velocity is desired along a portion of the path. The LSPB trajectory is such that the velocity is initially “ramped up” to its desired value and then “ramped down” when it approaches the goal position.

• To achieve this we specify the desired trajectory in three parts. The first part from time $t_0$ to time $t_b$ is a quadratic polynomial. This results in a linear “ramp” velocity.

• At time $t_b$, called the blend time, the trajectory switches to a linear function. This corresponds to a constant velocity.

• Finally, at time $t_f - t_b$ the trajectory switches once again, this time to a quadratic polynomial so that the velocity is linear.
Fig. 5.16  Blend times for LSPB trajectory
• We choose the blend time $t_b$ so that the position curve is symmetric. For convenience suppose that $t_0 = 0$ and $q'(t_0) = 0 = q'(0)$. Then between times 0 and $t_b$ we have

$$q(t) = a_0 + a_1 t + a_2 t^2$$

so that the velocity is

$$\dot{q}(t) = a_1 + 2a_2 t$$

The constraints $q_0 = 0$ and $\dot{q}(0) = 0$ imply that

$$a_0 = q_0$$
$$a_1 = 0$$

At time $t_b$ we want the velocity to equal a given constant, say $V$.

which implies that

$$a_2 = \frac{V}{2t_b}$$
Therefore the required trajectory between 0 and $t_b$ is given as

$$q(t) = q_0 + \frac{V}{2t_b} t^2$$

$$= q_0 + \frac{\alpha}{2} t^2$$

$$\dot{q}(t) = \frac{V}{t_b} t = \alpha t$$

$$\ddot{q} = \frac{V}{t_b} = \alpha$$

where $\alpha$ denotes the acceleration.
between time $t_f$ and $t_f - t_b$, the trajectory is a linear segment (corresponding to a constant velocity $V$)

$$q(t) = a_0 + a_1 t = a_0 + V t$$

Since, by symmetry,

$$q \left( \frac{t_f}{2} \right) = \frac{q_0 + q_f}{2}$$

we have

$$\frac{q_0 + q_f}{2} = a_0 + V \frac{t_f}{2}$$

which implies that

$$a_0 = \frac{q_0 + q_f - V t_f}{2}$$

Since the two segments must “blend” at time $t_b$ we require

$$q_0 + \frac{V}{2} t_b = \frac{q_0 + q_f - V t_f}{2} + V t_b$$

which gives upon solving for the blend time $t_b$

$$t_b = \frac{q_0 - q_f + V t_f}{V}$$
Note that we have the constraint $0 < t_b \leq \frac{t_f}{2}$. This leads to

$$\frac{q_f - q_0}{V} < t_f \leq \frac{2(q_f - q_0)}{V}$$

To put it another way we have the inequality

$$\frac{q_f - q_0}{t_f} < V \leq \frac{2(q_f - q_0)}{t_f}$$

Thus the specified velocity must be between these limits or the motion is not possible.

The portion of the trajectory between $t_f - t_b$ and $t_f$ is now found by symmetry
The complete LSPB trajectory is given by

\[
q(t) = \begin{cases} 
q_0 + \frac{a}{2}t^2 & 0 \leq t \leq t_b \\
\frac{q_f + q_0 - Vt_f}{2} + Vt & t_b < t \leq t_f - t_b \\
q_f - \frac{at_f^2}{2} + at_ft - \frac{a}{2}t^2 & t_f - t_b < t \leq t_f
\end{cases}
\]
Fig. 5.17  (a) LSPB trajectory (b) Velocity profile for LSPB trajectory (c) Acceleration for LSPB trajectory
Linear Interpolation with blends for several segments
Given:
- positions $u_i, u_j, u_k, u_l, u_m$
- desired time durations $t_{dij}, t_{dkj}, t_{dkl}, t_{dlm}$
- the magnitudes of the accelerations: $|\ddot{u}_i|, |\ddot{u}_j|, |\ddot{u}_k|, |\ddot{u}_l|$

Compute:
- blends times $t_i, t_j, t_k, t_l, t_m$
- straight segment times $t_{ij}, t_{jk}, t_{kl}, t_{lm}$
- slopes (velocities) $\dot{u}_{ij}, \dot{u}_{jk}, \dot{u}_{kl}, \dot{u}_{lm}$
- signed accelerations

System usually calculates or uses default values for accelerations. The system can also calculate desired time durations based on default velocities.
First segment

\[
\ddot{u}_1 = \text{sign}(u_2 - u_1)|\ddot{u}_1|
\]

\[
t_1 = t_{d_{12}} - \sqrt{t_{d_{12}}^2 - \frac{2(u_2 - u_1)}{\ddot{u}_1}}
\]

\[
\dot{u}_{12} = \frac{u_2 - u_1}{t_{d_{12}} - \frac{1}{2}t_1}
\]

\[
t_{12} = t_{d_{12}} - t_1 - \frac{1}{2}t_2
\]
Inside segments

\[ \dot{u}_{jk} = \frac{u_k - u_j}{t_{djk}} \]

\[ \ddot{u}_k = \text{sign}(\dot{u}_{kl} - \dot{u}_{jk})|\ddot{u}_k| \]

\[ t_k = \frac{\dot{u}_{kl} - \dot{u}_{jk}}{\ddot{u}_k} \]

\[ t_{jk} = t_{djk} - \frac{1}{2} t_j - \frac{1}{2} t_k \]
Last segment

\[ \ddot{u}_n = \text{sign}(u_{n-1} - u_n)|\ddot{u}_n| \]

\[ t_n = t_{d(n-1)n} - \sqrt{t_{d(n-1)n}^2 - \frac{2(u_n - u_{n-1})}{\ddot{u}_n}} \]

\[ \dot{u}_{(n-1)n} = \frac{u_n - u_{n-1}}{t_{d(n-1)n} - \frac{1}{2}t_n} \]

\[ t_{(n-1)n} = t_{d(n-1)n} - t_n - \frac{1}{2}t_{n-1} \]
To go through the **actual** via points:
- Introduce “Pseudo Via Points”

![Graph showing pseudo via points and original via point]

- Use sufficiently high acceleration
- If we want to stop there, simply repeat the via point
Trajectory Planning with Obstacles

- Path planning for the whole manipulator
  - Local vs. Global Motion Planning
    - Gross motion planning for relatively uncluttered environments
    - Fine motion planning for the end-effector frame
  - Configuration space (C-space) approach
- Planning for a point robot
  - graph representation of the free space, quadtree
  - Artificial Potential Field method
- Multiple robots, moving robots and/or obstacles