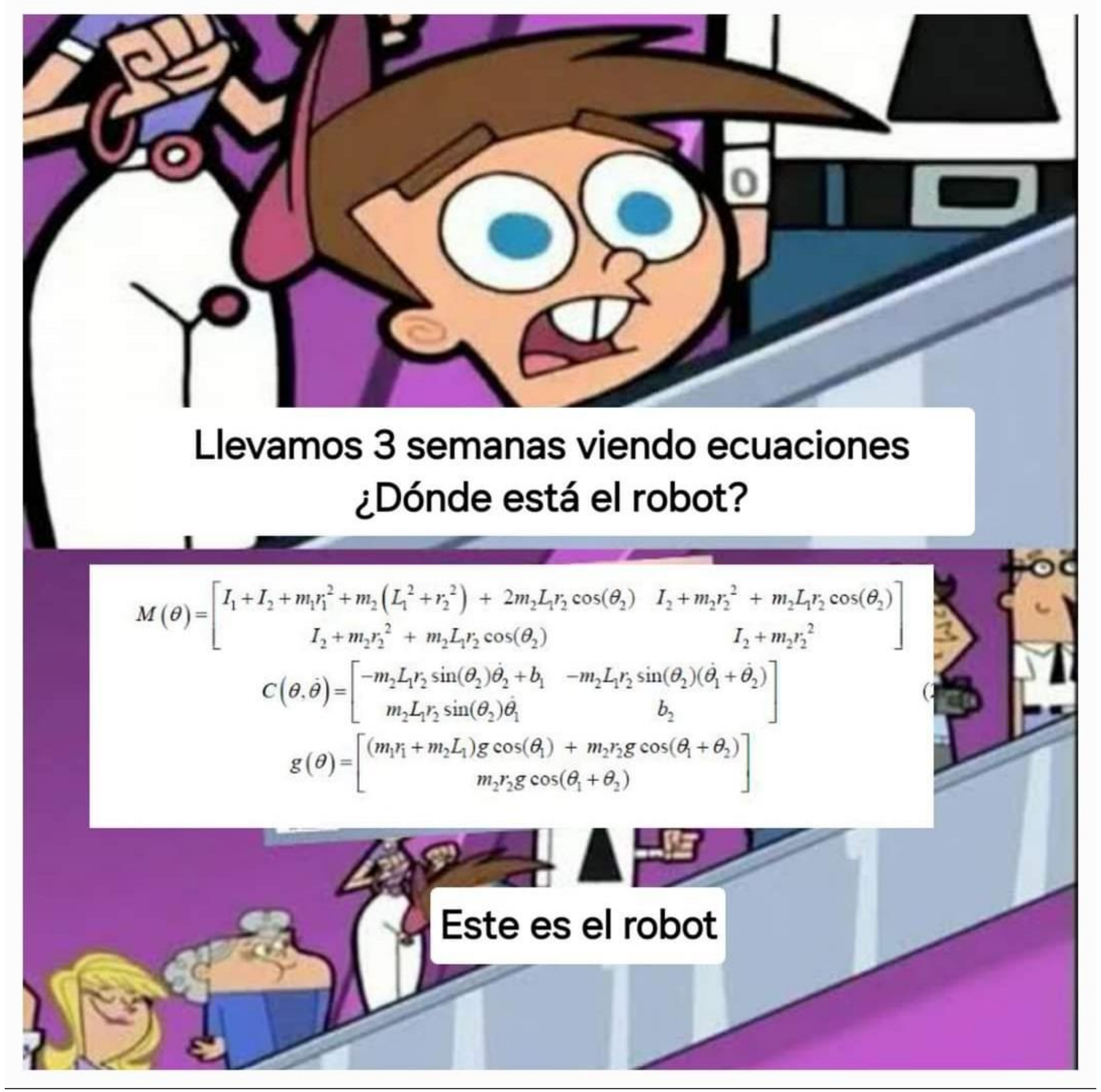
Reference Trajectories

SLM and DLM Trajectories

{Learn, Create, Innovate};

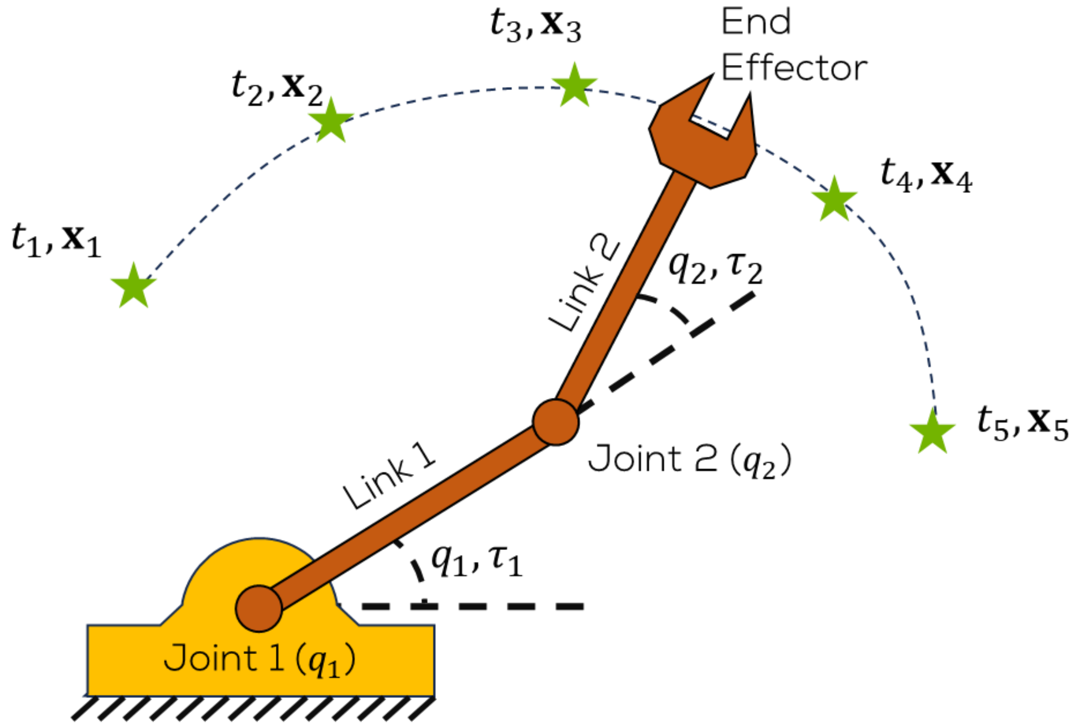

Llevamos 3 semanas viendo ecuaciones
¿Dónde está el robot?
$M(\theta)=\begin{bmatrix} I_1+I_2+m_1r_1^2+m_2\left(L_1^2+r_2^2\right)+2m_2L_1r_2\cos(\theta_2) & I_2+m_2r_2^2+m_2L_1r_2\cos(\theta_2) \\ I_2+m_2r_2^2+m_2L_1r_2\cos(\theta_2) & I_2+m_2r_2^2 \end{bmatrix}$
$C(\theta,\dot{\theta})=\begin{bmatrix} -m_2L_1r_2\sin(\theta_2)\dot{\theta}_2+b_1 & -m_2L_1r_2\sin(\theta_2)(\dot{\theta}_1+\dot{\theta}_2) \\ m_2L_1r_2\sin(\theta_2)\dot{\theta}_1 & b_2 \end{bmatrix}$
$g(\theta)=\begin{bmatrix} (m_1r_1+m_2L_1)g\cos(\theta_1)+m_2r_2g\cos(\theta_1+\theta_2) \\ m_2r_2g\cos(\theta_1+\theta_2) \end{bmatrix}$
Este es el robot

Introduction

The aim of this section is to describe how to **generate a reference trajectory**, using piecewise polynomials (splines), for a single link and double link manipulator

The objectives are:

1. Justification for **continuous reference trajectories**
2. **Cubic spline** representation
3. **Matrix**-based **interpolation** calculation
4. Single spline position trajectory **example**
5. **Manipulator trajectories: workspace** & **singularity** concerns



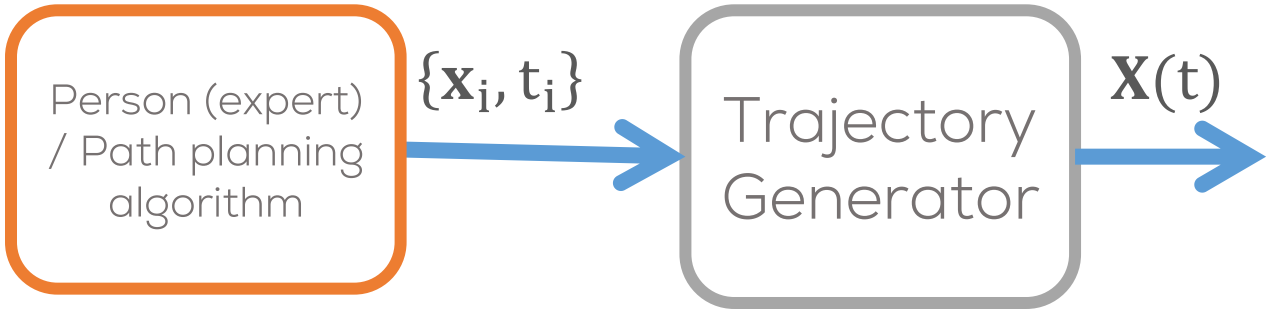
Trajectory Generation Problem

- Generally, the manipulator's (robot's) desired behaviour is specified as a **discrete set of points** (either Cartesian or joint space) through which the end effector must pass.
- The points are given by:
  - Some expert in the robot as a **`move`** command.

```
move(t1, x1, t2, x2, …)
```

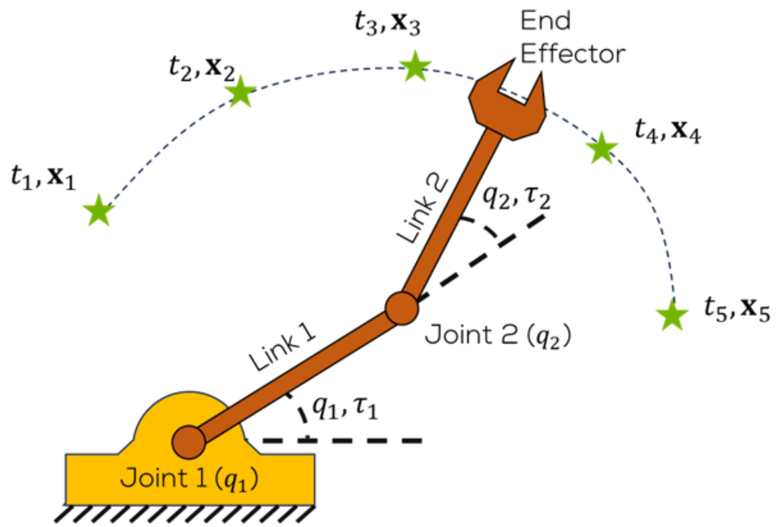
  - Provided by a high level path planning algorithm (Dijkstra).
- The points provided by the expert or the path planning must be **interpolated** to produce a continuous path $\mathbf{X}(t)$.

- Generally, the manipulator's (robot's) desired behaviour is specified as a **discrete set of points** (either Cartesian or joint space) through which the end effector must pass.
- The points are given by:
  - Some expert in the robot as a **move** command.
  - Provided by a high level path planning algorithm (Dijkstra).
- The points provided by the expert or the path planning must be **interpolated** to produce a continuous path $\mathbf{X}(t)$.



Why not use Step Reference Trajectories & Linear Feedback Control?

- Feedback control often assumes that the **controller** is **linear**, error **feedback** (PID or state space)
- Assumed that the **reference signal** is a step or a **sequence of steps** which specify the desired (joint) position at time points.
- These assumptions can presents several characteristically behaviours such as: non smooth trajectories and poor set point following.

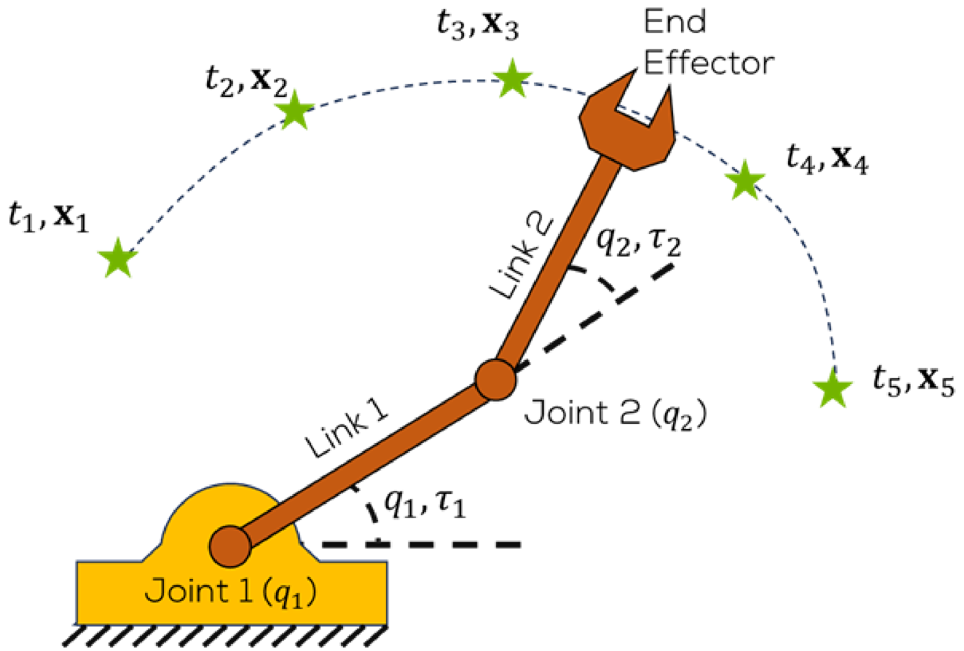


Step Reference trajectories disadvantages.

- Step angle / position commands do not specify how the robot actually responds so **collisions may occur** with the environment
- Smooth motion is required (not short steady state periods with sharp transients)
- Joint errors are largest at the start, so are the calculated torques and this can **significantly exceed the delivered torque**
- **PID control zeros** often produce **overshoot**
- Robots are often quite **non-linear**
- Error-based PID control is **reactive**, i.e. an error must exist for the control signal to be calculated
- Highly accurate position specification & control is required

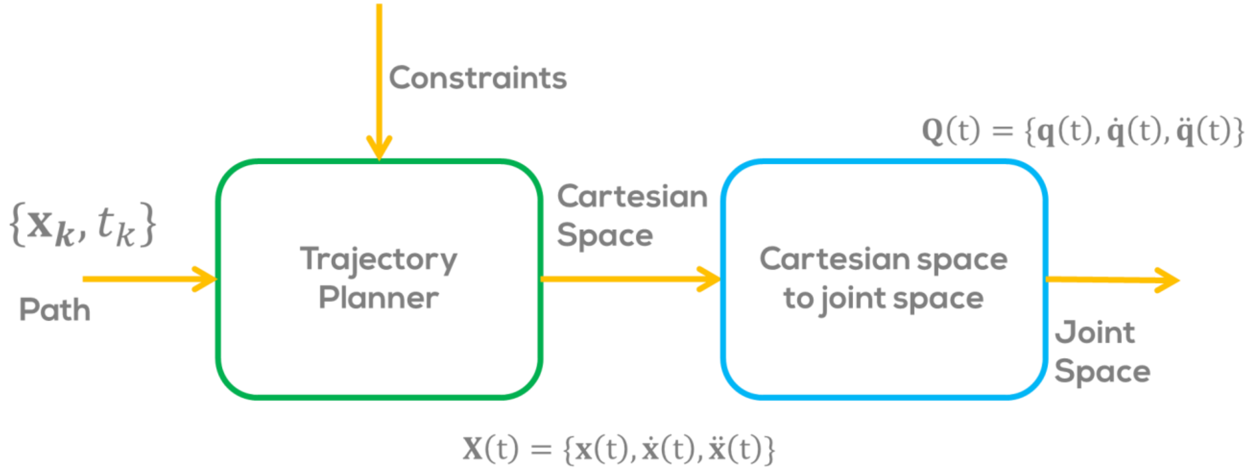
Step Reference trajectories disadvantages.

- Other type of controller are needed for these tasks.
- As an example it is possible to use Sliding mode control, feedback linearization, model predictive control.
- These types of controllers, use the model of the system's dynamical behaviour to follow reference trajectories in a smooth fashion.



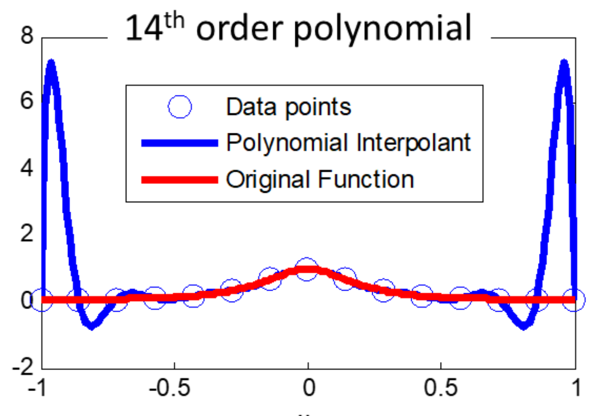
Trajectory generation

- The aim of the trajectory generator is to produce a continuous path from the discrete set of points $\{\mathbf{x}_\mathrm{i}, \mathrm{t}_\mathrm{i}\}$, that sample the desired trajectory.
- Usually the trajectories are specified in **cartesian space**, then converted to **joint space.**
- Some constraints are defined to make the trajectory smooth and continuous for the robot.



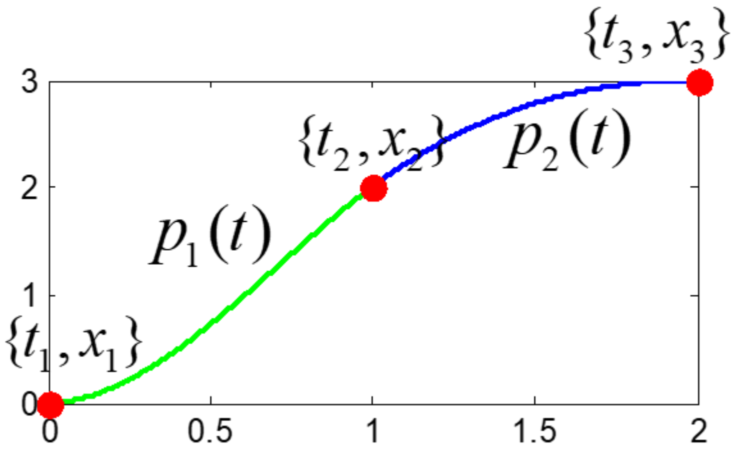
Polynomial Spline

- Interpolation of a set of points can be done in different ways.
- The two man ways are polynomial and spline interpolation.
- Polynomial interpolation is good with higher order polynomials, but presents several problems such as Runge's Phenomenon.



Spline interpolation

- A **spline** is a **numeric function** which is made of **piecewise polynomials**, $p_k(t)$.
- It's smooth at the **knots** – where the polynomial pieces connect and data points, $\{t_k, x_k\}$, are given.



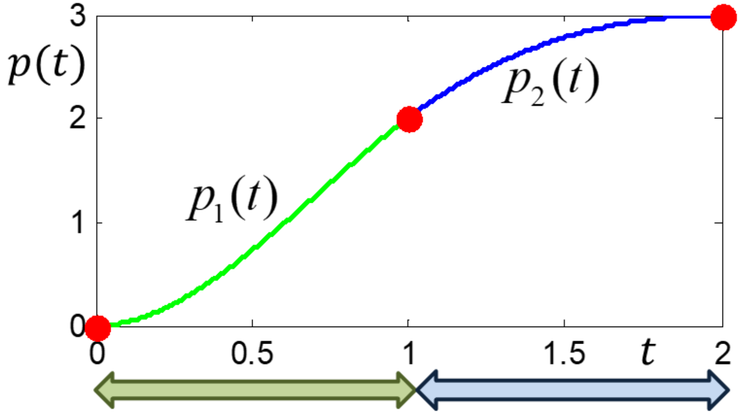
Cubic Polynomial Spline

- A cubic spline (order 3 polynomials) is usually used.

$$p(t) = a_1 + a_2 t + a_3 t^2 + a_4 t^3$$

- The spline $p(t)$, has time t as the independent variable.
- The four parameters are a_i: a_1 bias, a_2 linear, a_3 quadratic, a_4 cubic.
- The spline's output linearly depends on the parameters

- Four (independent) equations are needed to uniquely determine the parameter values (linear system of equations).
- A (**piecewise**) **cubic spline** is simply a set of cubic polynomials, one for each of the time intervals which are specified by adjacent knots



Time intervals

$$p_1(t) = a_{1,1} + a_{1,2}t + a_{1,3}t^2 + a_{1,4}t^3$$
$t \in [0,1)$

$$p_2(t) = a_{2,1} + a_{2,2}t + a_{2,3}t^2 + a_{2,4}t^3$$
$t \in [1,2]$

Why a Cubic Polynomial?

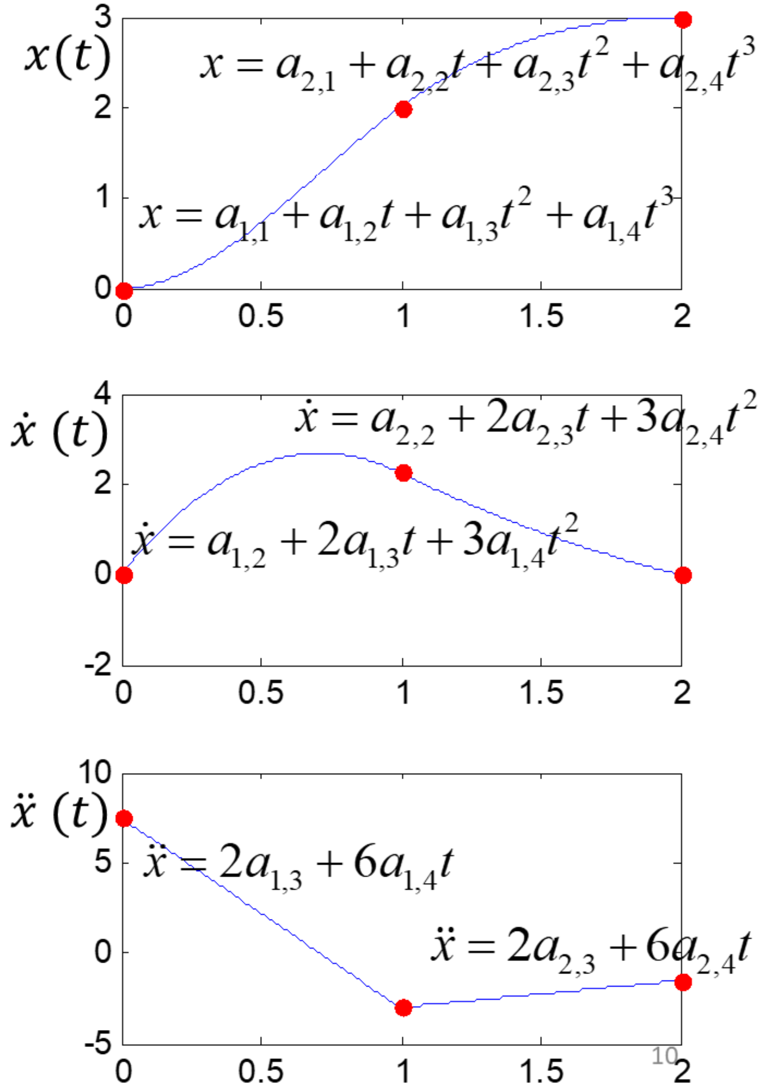
A cubic polynomial provides the following features.

Continuous position, x: This feature ensures a smooth and controlled movement of the manipulator, with the position being at least piecewise linear.

Continuous velocity, $\dot{x}$: No velocity jumps in mechanical systems. Therefore, they must be *at least piecewise quadratic*

Continuous acceleration, $\ddot{x}$: Torque must be continuous;s therefore, the joint or Cartesian acceleration must also be continuous.

Discontinuous jerk, $\dddot{x}$, is discontinuous (piecewise constant)

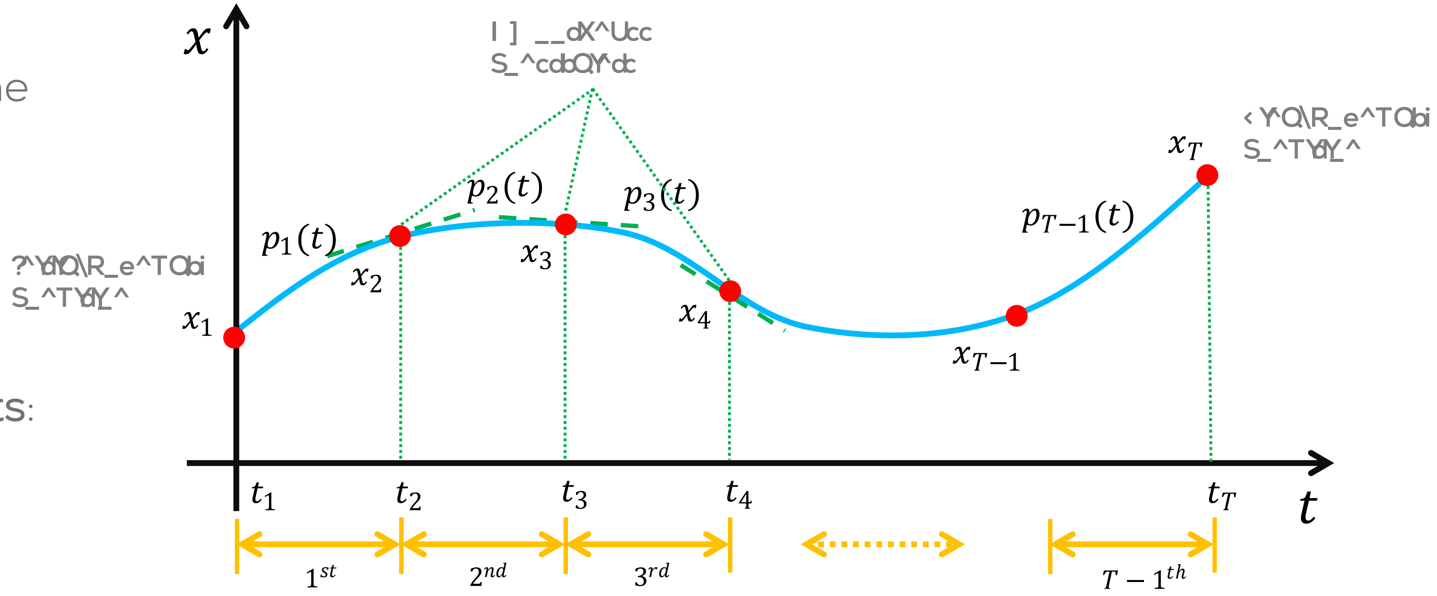


Parameters of a cubic spline

- Let T to be the number of points or knots to be to interpolated.
- To uniquely determine the parameters for the cubic spline, we need a set of $\mathbf{4 * (T-1)}$ equations.

1. **Interpolating data** at the "knots" $\{t_k, x_k\}$
2. **Smoothness constraints** at the **interior knots**: $\dot{p}_{k-1}(t_k) = \dot{p}_k(t_k)$
3. **Boundary conditions** at the **exterior knots**: $\dot{p}_1(t_1) = 0$

- All of these equations are **linearly dependent** on the parameters



Data Interpolation at Knots

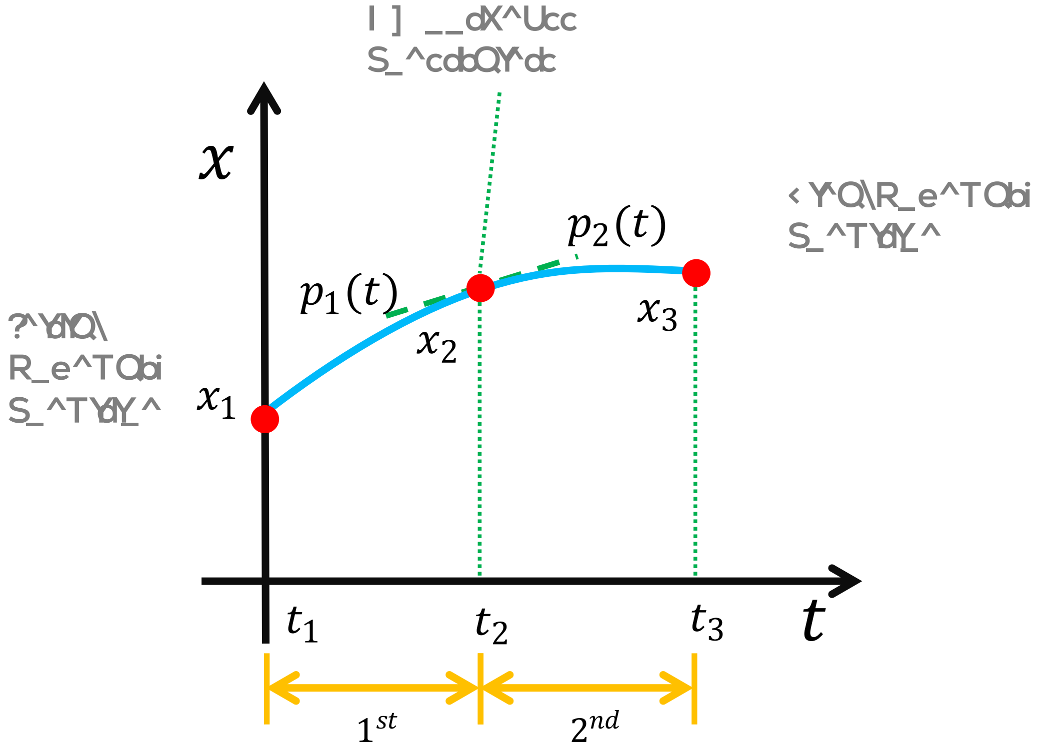
- Consider the spline on the right image containing 3 knots $x_1, x_2,$ and x_3 therefore $T = 3$, therefore we need 8 equations.
  - The exterior knot positions are:

$$p_1(t_1) = x_1, \qquad p_{T-1}(t_T) = \mathrm{p}_2(\mathrm{t}_3) = x_T = x_3$$

  - The Interior knots are given by

$$p_{k-1}(t_k) = p_1(\mathrm{t}_2) = x_k = x_2 = p_k(t_k) = p_2(t_2)$$

- The interpolation conditions generate $\mathbf{2} * (\boldsymbol{T} - \mathbf{1})$ linear equations.
- We need $\mathbf{4} * (\boldsymbol{T} - \mathbf{1})$ equations… we only have half of the required equations.



Data Interpolation at Knots

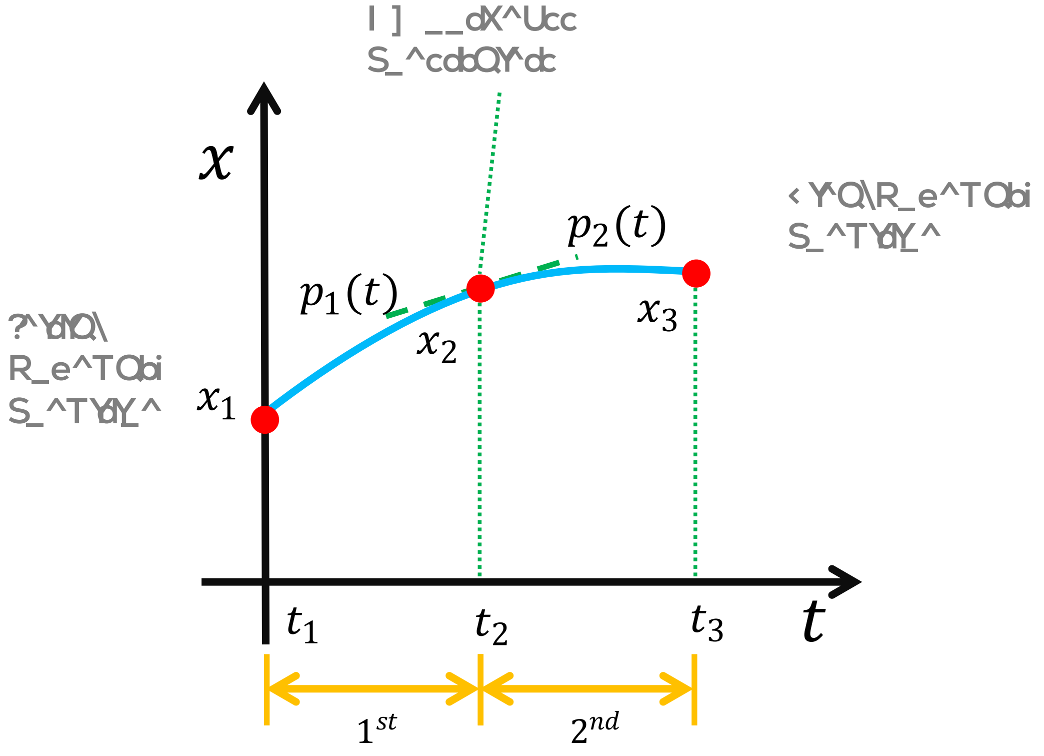
- The interior and exterior position conditions can be written in a matrix form as:

$$p_1(t_1) = x_1, \qquad p_2(t_3) = x_3$$

$$p_1(t_2) = x_2 = p_2(t_2)$$

$$\begin{array}{l} \text{Exterior: } t_1, x_1 \\ \text{Interior: } t_2, x_2 \\ \text{Interior: } t_2, x_2 \\ \text{Exterior: } t_3, x_3 \end{array}
\left[\begin{array}{cccc|cccc}
1 & t_1 & t_1^2 & t_1^3 & 0 & 0 & 0 & 0 \\
1 & t_2 & t_2^2 & t_2^3 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & t_2 & t_2^2 & t_2^3 \\
0 & 0 & 0 & 0 & 1 & t_3 & t_3^2 & t_3^3
\end{array}\right]
\begin{bmatrix} a_{1,1} \\ a_{1,2} \\ a_{1,3} \\ a_{1,4} \\ a_{2,1} \\ a_{2,2} \\ a_{2,3} \\ a_{2,4} \end{bmatrix}
=
\begin{bmatrix} x_1 \\ x_2 \\ x_2 \\ x_3 \end{bmatrix}$$

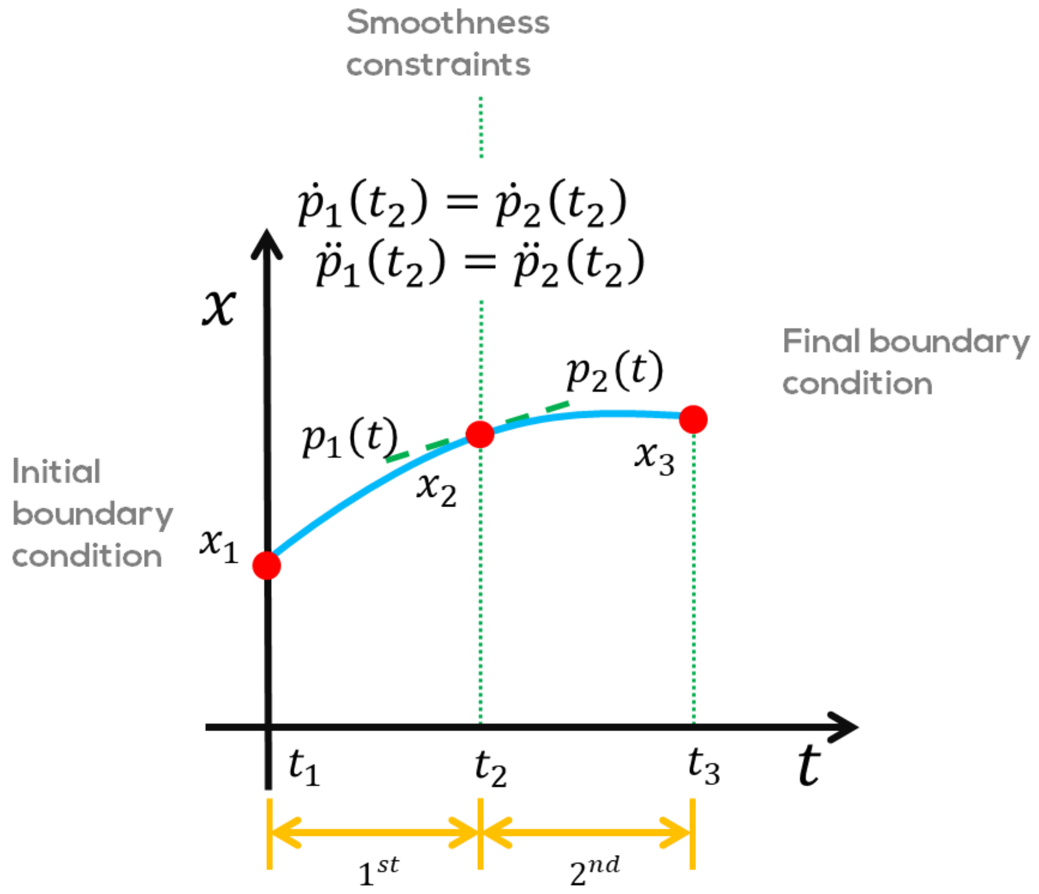
$$\xleftarrow{\quad p_1 \quad} \Big| \xrightarrow{\quad p_2 \quad}$$



Internal Smoothness Constraints

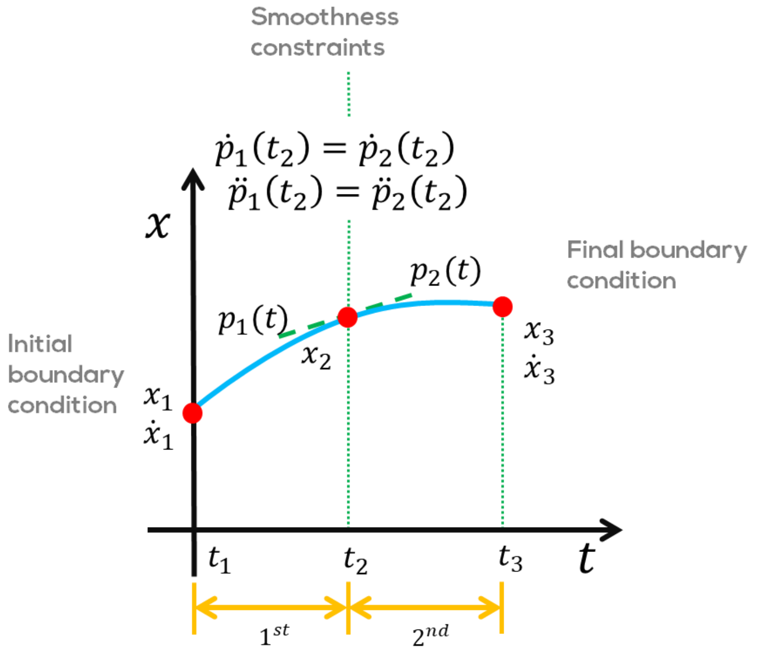
- **Smoothness constraints** ensure two neighbouring cubic polynomials **join smoothly** at the **interior knots**
- 1st derivative: $\dot{p}_{k-1}(t_k) = \dot{p}_k(t_k)$
- 2nd derivative: $\ddot{p}_{k-1}(t_k) = \ddot{p}_k(t_k)$
- This gives an additional **2*(T-2) linear equations**

$$
\begin{array}{l}
1^{st}\text{ deriv} \\
2^{nd}\text{ deriv}
\end{array}
\underbrace{\begin{bmatrix} 0 & 1 & 2t_k & 3t_k^2 & 0 & -1 & -2t_k & -3t_k^2 \\ 0 & 0 & 2 & 6t_k & 0 & 0 & -2 & -6t_k \end{bmatrix}}_{\longleftarrow p_{k-1} \;\vert\; p_k \longrightarrow}
\begin{bmatrix} a_{k-1,1} \\ a_{k-1,2} \\ a_{k-1,3} \\ a_{k-1,4} \\ a_{k,1} \\ a_{k,2} \\ a_{k,3} \\ a_{k,4} \end{bmatrix}
=
\begin{bmatrix} 0 \\ 0 \end{bmatrix}
$$



Boundary Conditions

- So far $\mathbf{4 * (T-2) + 2}$ equations have been generated in the $\mathbf{4 * (T-1)}$ parameters. Therefore 2 extra constraints must be specified
- This is typically done by specifying the derivatives at the two exterior knots, t_1 and t_T, to be zero
- Other similar constraints include having a zero acceleration at the exterior knots.
- The user can also specify the 1st and 2nd derivative at the first knot to smoothly join with another spline.

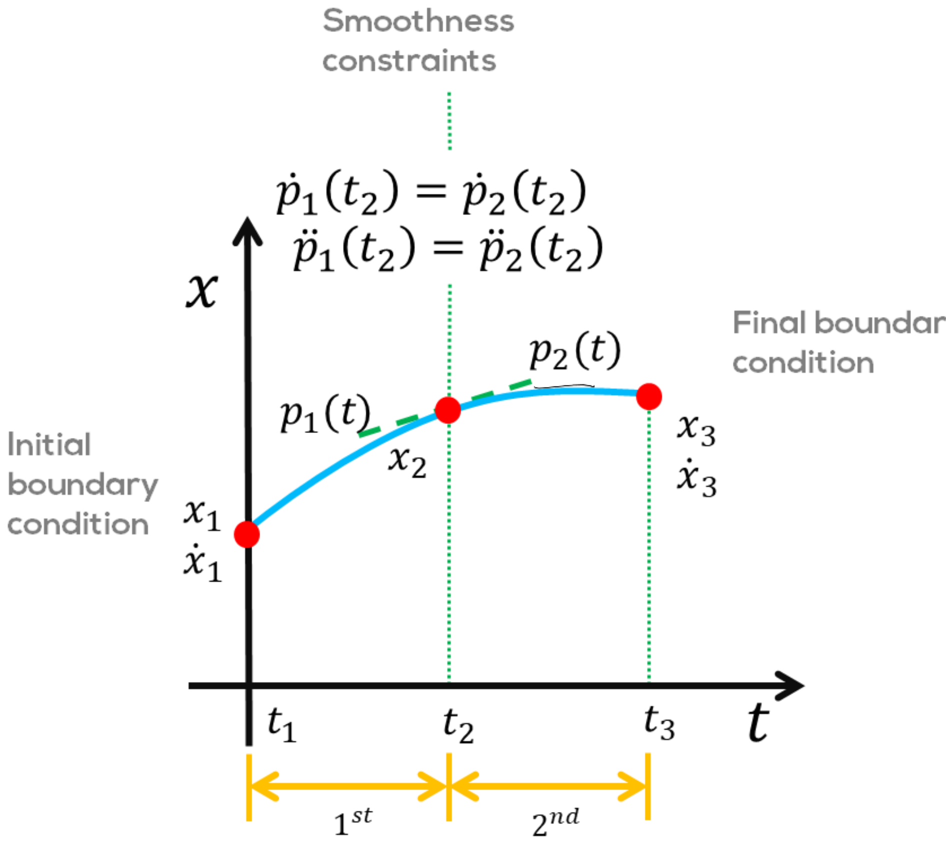


$$\left[\begin{array}{cccc|cccc} 0 & 1 & 2t_1 & 3t_1^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & t_3 & t_3^2 & t_3^3 \end{array}\right] \begin{bmatrix} a_{1,1} \\ a_{1,2} \\ a_{1,3} \\ a_{1,4} \\ a_{2,1} \\ a_{2,2} \\ a_{2,3} \\ a_{2,4} \end{bmatrix} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_3 \end{bmatrix}$$

p_{k-1} p_k

$$
\begin{array}{r}
p_1(t_1) = x_1 \\
\dot{p}_1(t_1) = \dot{x}_1 \\
p_1(t_2) = x_2 \\
\dot{p}_1(t_2) = \dot{p}_2(t_2) \\
\ddot{p}_1(t_2) = \ddot{p}_2(t_2) \\
p_2(t_2) = x_2 \\
p_2(t_3) = x_3 \\
\dot{p}_2(t_3) = \dot{x}_3
\end{array}
\begin{bmatrix}
1 & t_1 & t_1^2 & t_1^3 & 0 & 0 & 0 & 0 \\
0 & 1 & 2t_1 & 3t_1^2 & 0 & 0 & 0 & 0 \\
1 & t_2 & t_2^2 & t_2^3 & 0 & 0 & 0 & 0 \\
0 & 1 & 2t_2 & 3t_2^2 & 0 & -1 & -2t_2 & -3t_2^2 \\
0 & 0 & 2 & 6t_2 & 0 & 0 & -2 & -6t_2 \\
0 & 0 & 0 & 0 & 1 & t_2 & t_2^2 & t_2^3 \\
0 & 0 & 0 & 0 & 1 & t_3 & t_3^2 & t_3^3 \\
0 & 0 & 0 & 0 & 0 & 1 & 2t_3 & 3t_3^2
\end{bmatrix}
\cdot
\begin{bmatrix}
a_{1,1} \\ a_{1,2} \\ a_{1,3} \\ a_{1,4} \\ a_{2,1} \\ a_{2,2} \\ a_{2,3} \\ a_{2,4}
\end{bmatrix}
=
\begin{bmatrix}
x_1 \\ \dot{x}_1 \\ x_2 \\ 0 \\ 0 \\ x_2 \\ x_3 \\ \dot{x}_3
\end{bmatrix}
$$

$\leftarrow p_1$ $p2 \rightarrow$



{Learn, Create, Innovate};

Example: Cubic Spline Trajectory

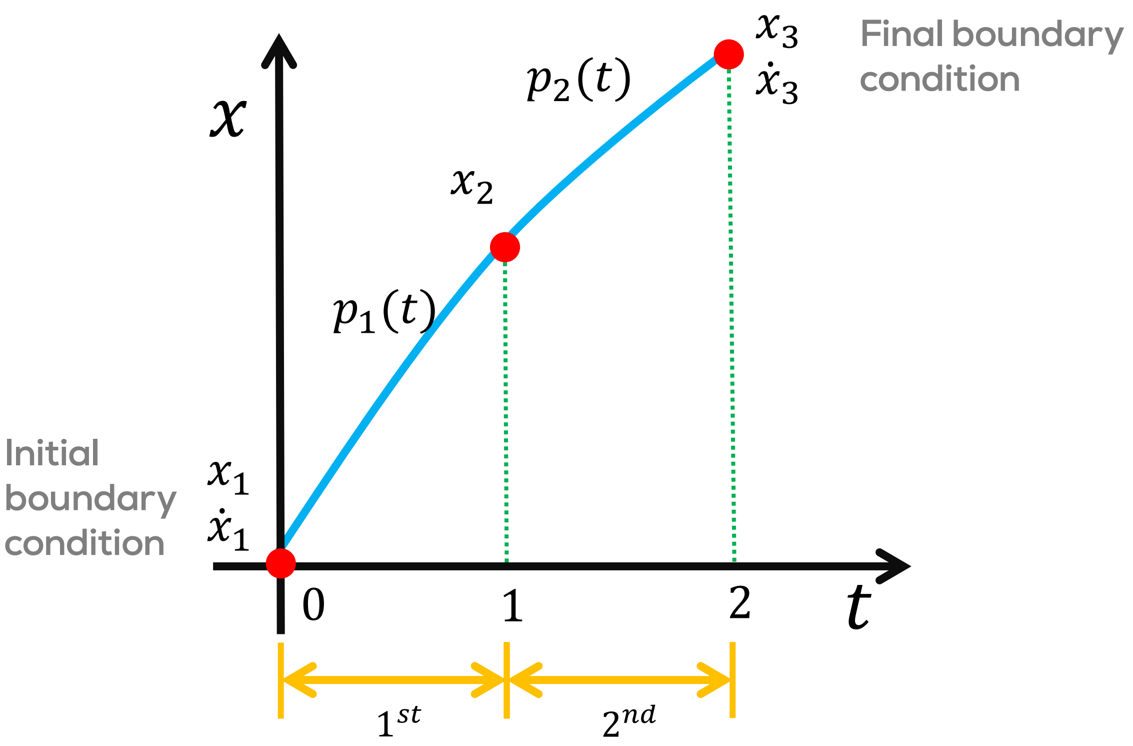
- This example will calculate a cubic spline trajectory for the following points:

k	t_k	x_k	$\dot{x}_k$
1	0	0	0
2	1	2	
3	2	3	0

- Continuous position, velocity and acceleration.

Note:

- The variable could be joint, y-position, ….
- There is no requirement for the points to be equally spaced
- There is generally more than 2 intervals & polynomials (or equivalently one interior knot)



Example: Cubic Spline Trajectory

- The cubic spline trajectory has two cubic polynomials, p_1 and p_2, defined on the intervals [0,1] and [1,2], which have 4 parameters each, $a_{1,1:4}$ and $a_{2,1:4}$.
- The 4 interpolation, 2 smoothness and 2 boundary conditions gives the following set of 8 linear equations

- Substituting the values in the Matrix we have:

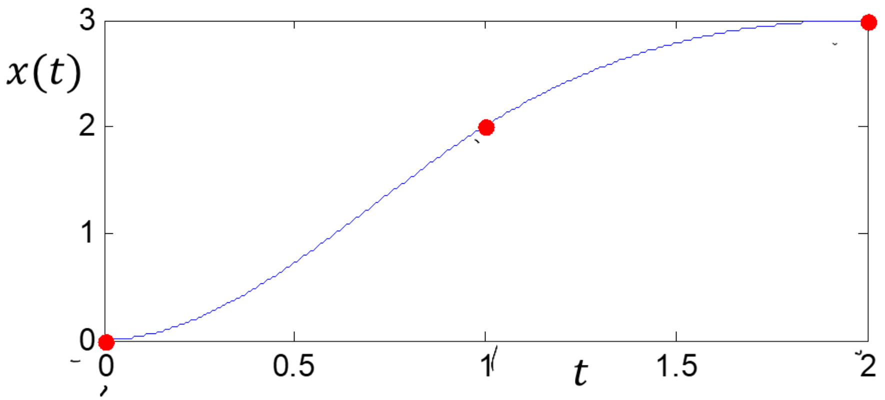
$$
\begin{array}{r}
p_1(t_1) = 0 \\
\dot{p}_1(t_1) = 0 \\
p_1(t_2) = 2 \\
\dot{p}_1(t_2) = \dot{p}_2(t_2) \\
\ddot{p}_1(t_2) = \ddot{p}_2(t_2) \\
p_2(t_2) = 2 \\
p_2(t_3) = 3 \\
\dot{p}_2(t_3) = 0
\end{array}
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 2 & 3 & 0 & -1 & -2 & -3 \\
0 & 0 & 1 & 6 & 0 & 0 & -1 & -6 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 2 & 4 & 8 \\
0 & 0 & 0 & 0 & 0 & 1 & 4 & 12
\end{bmatrix}
\cdot
\begin{bmatrix}
a_{1,1} \\ a_{1,2} \\ a_{1,3} \\ a_{1,4} \\ a_{2,1} \\ a_{2,2} \\ a_{2,3} \\ a_{2,4}
\end{bmatrix}
=
\begin{bmatrix}
0 \\ 0 \\ 2 \\ 0 \\ 0 \\ 2 \\ 3 \\ 0
\end{bmatrix}
$$

- Inverting the matrix and solving for the vector of parameters $\boldsymbol{a}$ in MATLAB

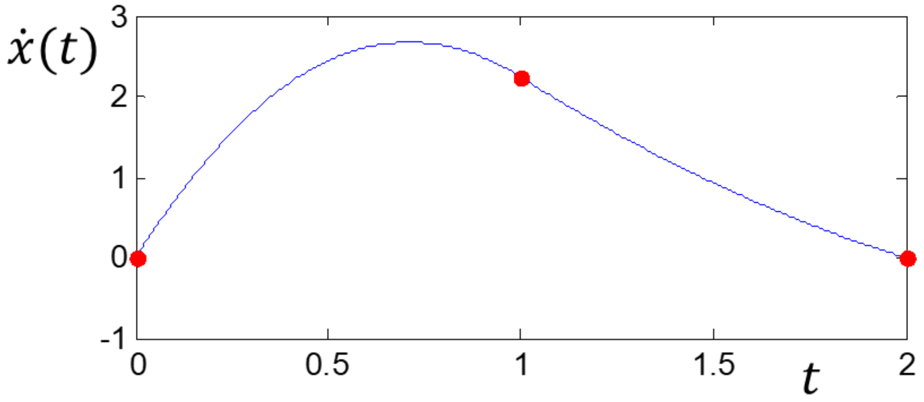
$$\boldsymbol{a} = [0, 0, 3.75, -1.75, -2, 6, -2.25, 0.25]^T$$

Results

- The original data points and the interpolating spline trajectory can therefore be plotted.
- Each segment is cubic.

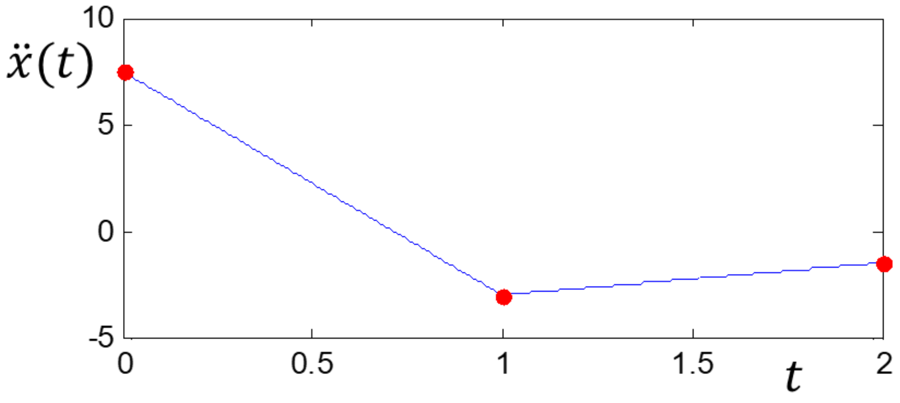


- Velocity (first derivative) of the cubic spline.
- Each segment is quadratic.
- First derivative meets specified boundary conditions.
- Still appears smooth.



Results

- **Acceleration** (second derivative) of the cubic spline.
- Each segment is linear.
- Second derivative (acceleration) is piecewise linear.
- Discontinuous in jerk.




```
%% MATLAB CODE
% Points to interpolate
td = [0 1 2];
xp = [0 2 3];

% Estimate parameters of cubic spline
T = [1 td(1) td(1)^2 td(1)^3 0 0 0 0; ...
    0 1 2*td(1) 3*td(1)^2 0 0 0 0; ...
    1 td(2) td(2)^2 td(2)^3 0 0 0 0; ...
    0 1 2*td(2) 3*td(2)^2 0 -1 -2*td(2) -3*td(2)^2; ...
    0 0 2 6*td(2) 0 0 -2 -6*td(2); ...
    0 0 0 0 1 td(2) td(2)^2 td(2)^3; ...
    0 0 0 0 1 td(3) td(3)^2 td(3)^3; ...
    0 0 0 0 0 1 2*td(3) 3*td(3)^2];
x = [xp(1) 0 xp(2) 0 0 xp(2) xp(3) 0]';
a = inv(T)*x;
```

Kinematic / Joint Considerations

- In practice the trajectories are as $\{t_k, x_k, y_k, z_k\}$
- In this case, each of the signals must be interpolated.
- After the interpolation, they must be then converted into a joint space reference trajectory.

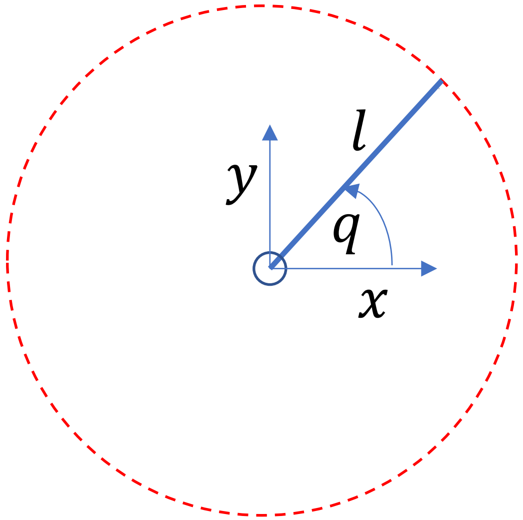
- As engineers, we need to ensure

1. That the trajectory is **reachable** (lies in the **workspace**)
2. **Singular configurations** are **avoided**

   Excessive torques (joint accelerations) are not demanded

Single Joint Manipulator

For a single joint manipulator, the spline would need to be specified in joint space, as the workspace is a circle of radius l centred on the joint



Workspace

$$\mathbf{x} = l \begin{bmatrix} \cos(q) \\ \sin(q) \end{bmatrix}$$

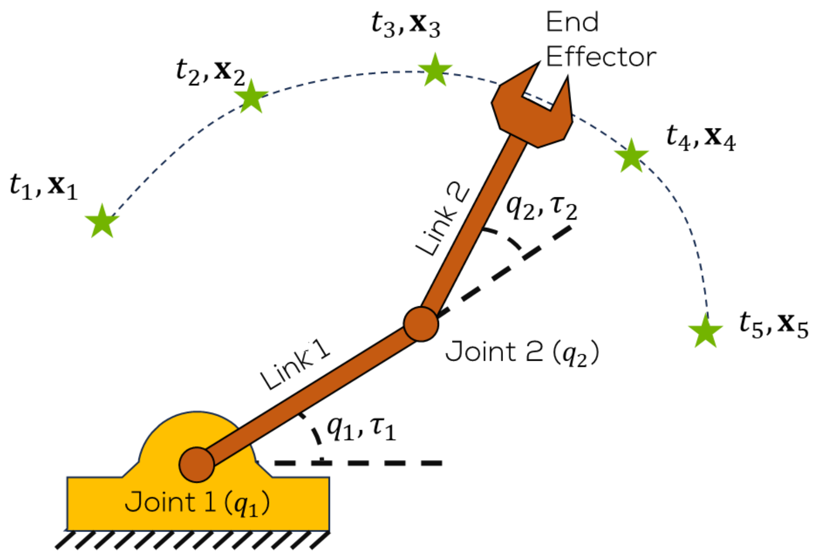
If you specify a reference trajectory via a cubic spline in Cartesian space, it will lie outside the circle (workspace) and would not be realisable; in other words, the inverse kinematics will not give a solution

A realisable (cubic spline) trajectory would have to be specified in joint space

Dual Link Manipulator

Build a reference trajectory for a dual link manipulator. For this example (for simplicity), the links are $l_1 = l_2 = 1\ m$, and the trajectory is specified by the points:

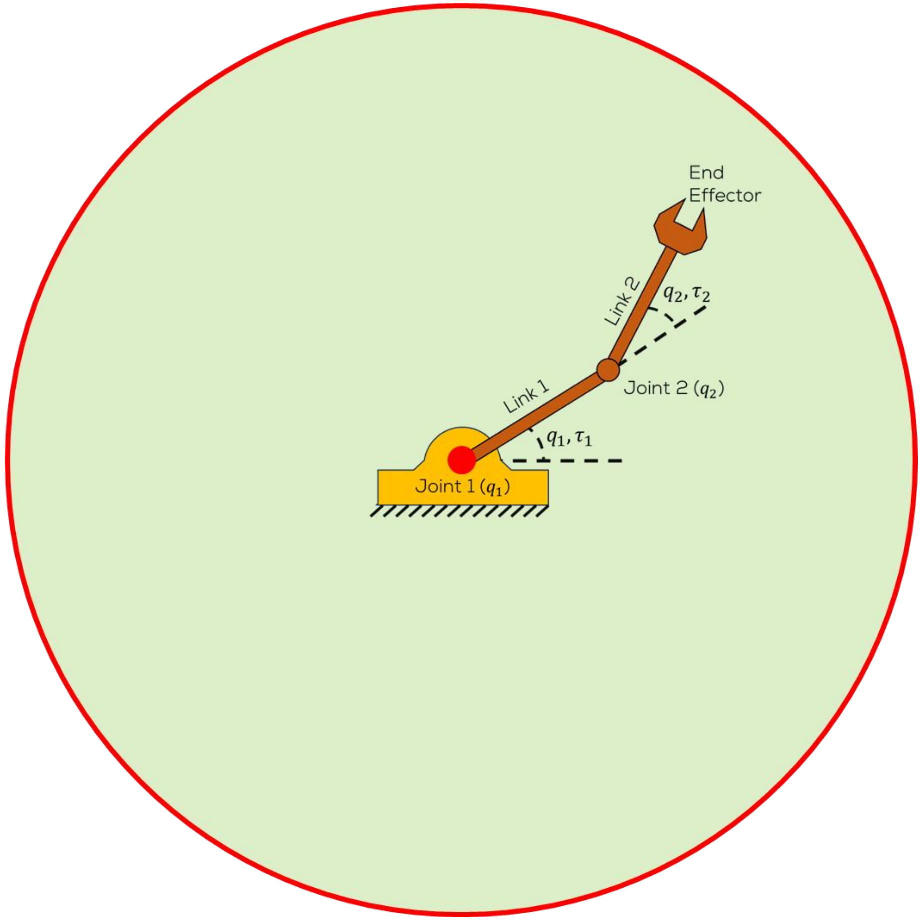
k	t_k	x_k	$\dot{x}_k$	y_k	$\dot{y}_k$
1	0	0.1	0	0	0
2	1	1		1	
3	2	1.99		0	
4	3	1		-1	
5	4	0.1	0	0	0



The splines in Cartesian or joint space will give different trajectories, because the inverse kinematics are non-linear.

$x - y$ Splines Preamble

- The workspace for this example is a circle of radius 2 (green).
- The only singularities occur along the perimeter and at the centre (red).
- In this example, the x & y splines will be fitted in Cartesian space and differentiated (1st and 2nd) to produce Cartesian velocity and accelerations.
- The inverse kinematics map will also be used to generate the corresponding signals (position, velocity and acceleration) in joint space.

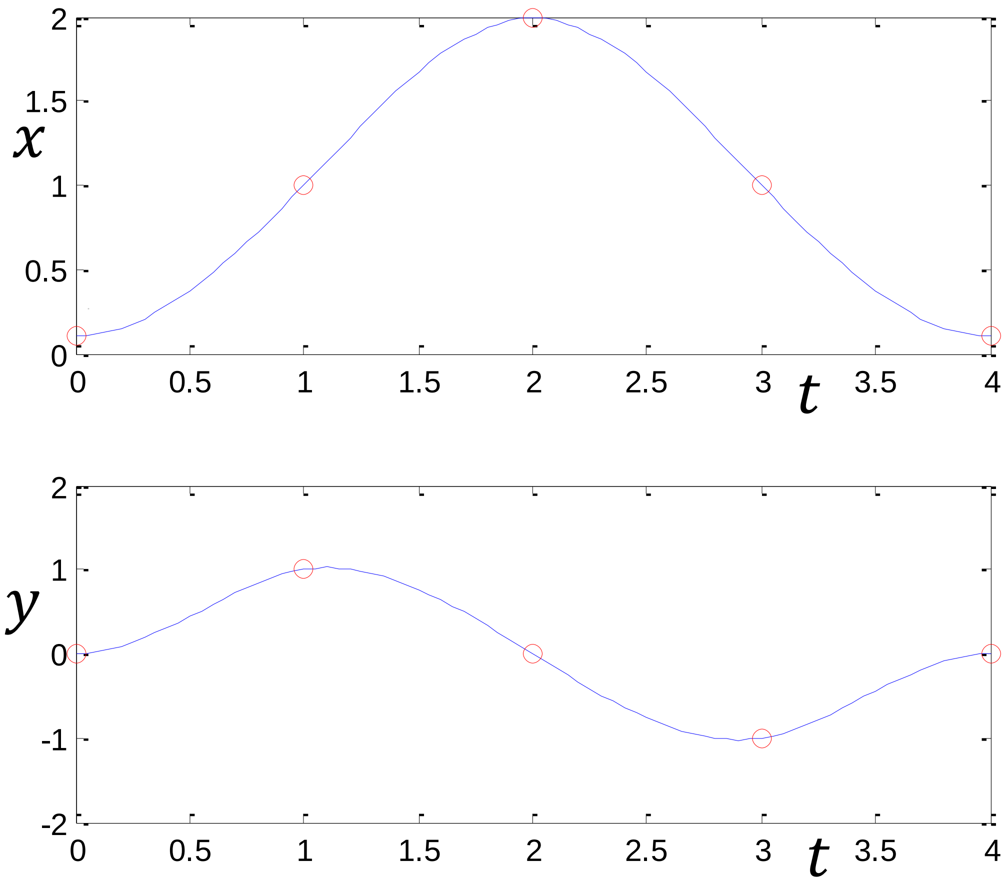


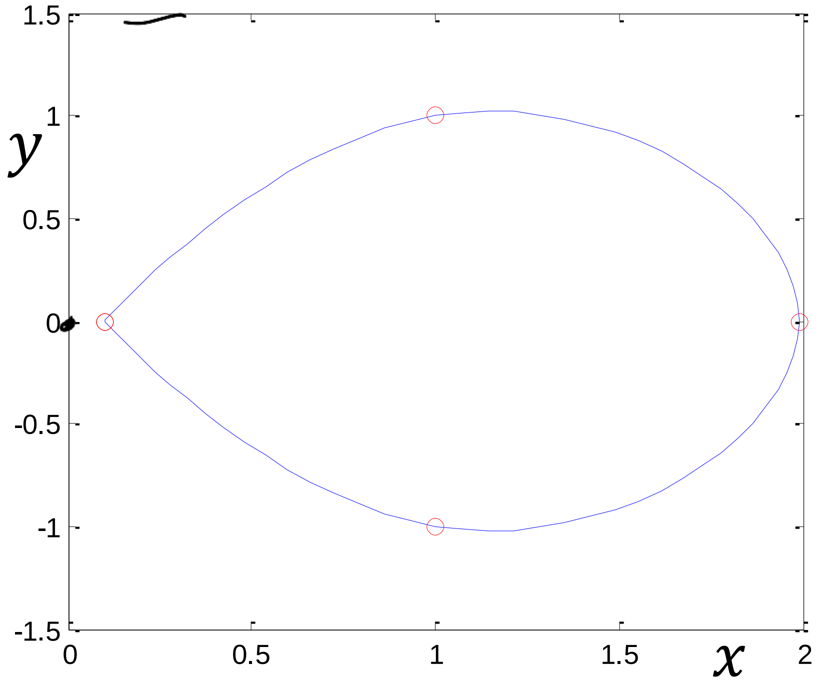
$x - y$ Splines Preamble

- For each of the two splines, there are
- 4 intervals
- 16 parameters
- 6 internal smoothness constraints,
- 8 data interpolation constraints and
- 2 velocity boundary conditions.
- Therefore, there is a 16*16 matrix which must be inverted to estimate the parameters.

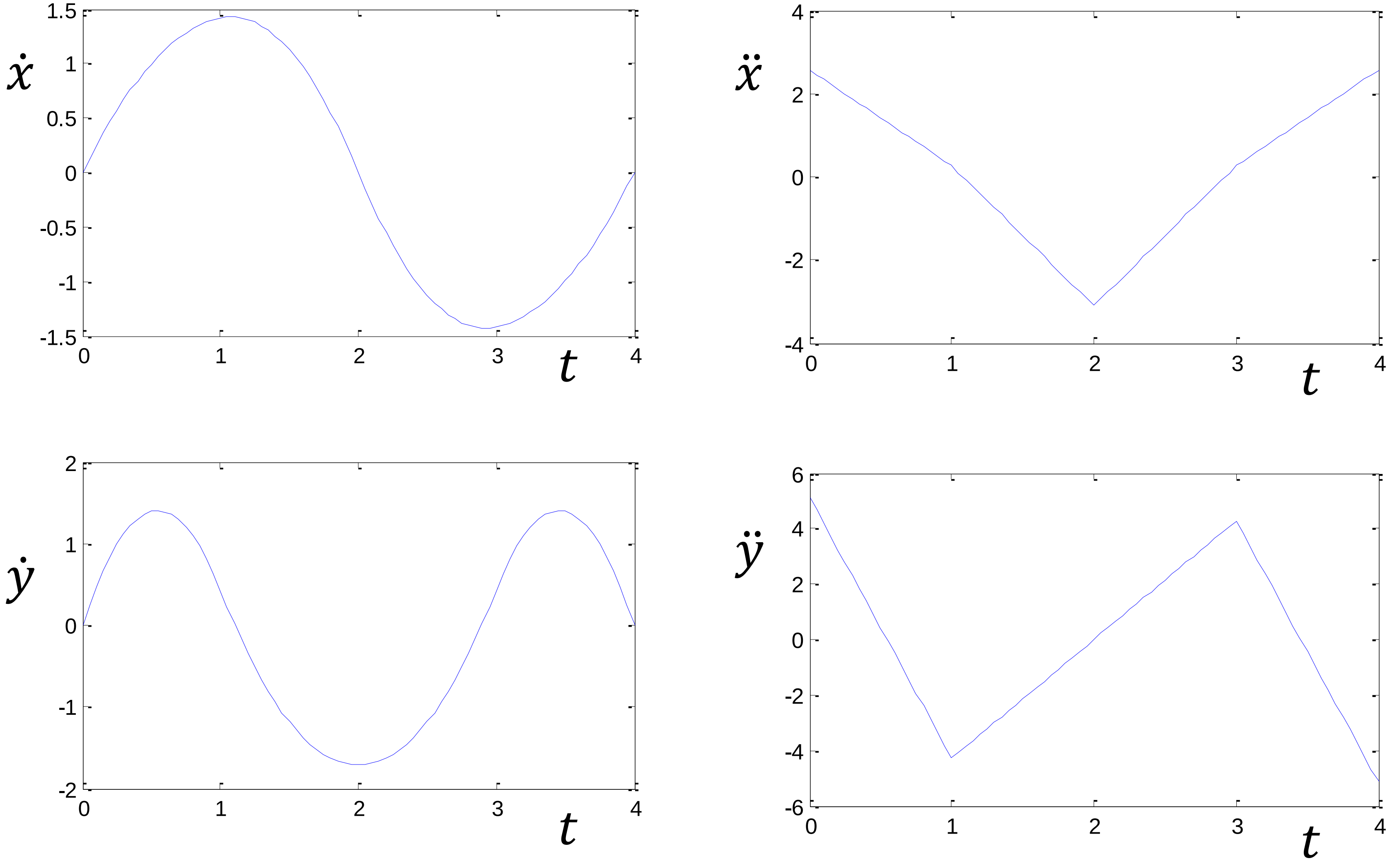
Result: End-Effector Trajectories

Fitting the two splines Cartesian x – y splines produces:



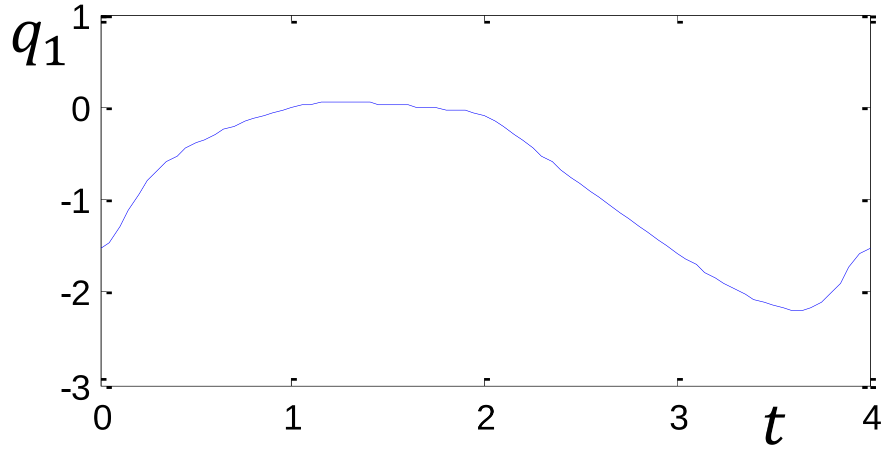


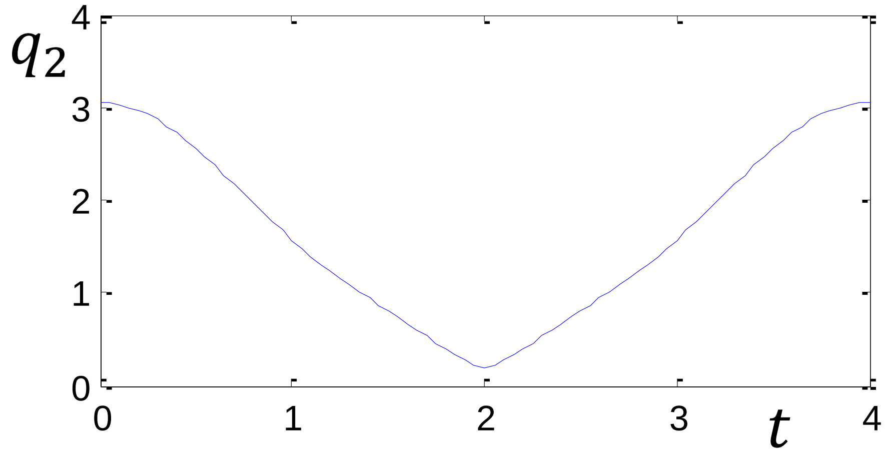
Cartesian Velocity & Accelerations

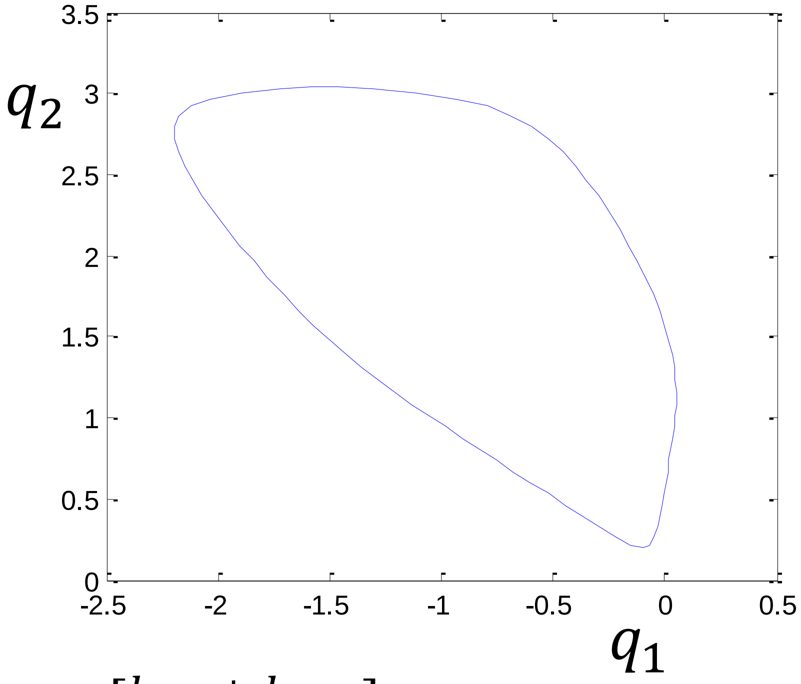


Piecewise quadratic velocity and linear accelerations

Joint Trajectories







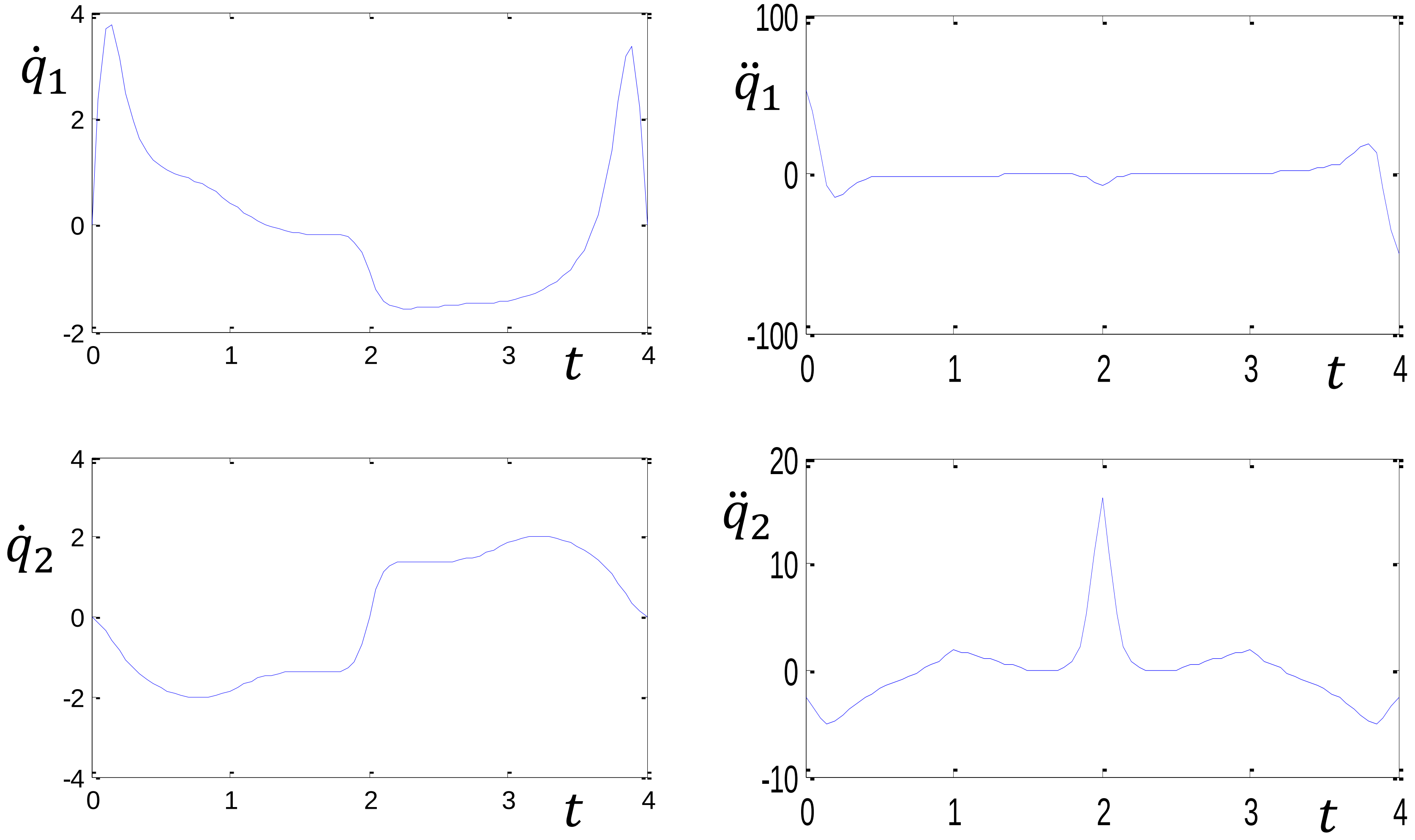
Using the inverse kinematics map allows us to view the splines / trajectories in joint space

$$\mathbf{x} = \begin{bmatrix} l_1 c_1 + l_2 c_{12} \\ l_1 s_1 + l_2 s_{12} \end{bmatrix}$$

$$\dot{\mathbf{x}} = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$$

$$\ddot{\mathbf{x}} = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} - \begin{bmatrix} l_1 c_1 \dot{q}_1^2 + l_2 c_{12} (\dot{q}_1 + \dot{q}_2)^2 \\ l_1 s_1 \dot{q}_1^2 + l_2 s_{12} (\dot{q}_1 + \dot{q}_2)^2 \end{bmatrix}$$

Joint Velocities and Accelerations

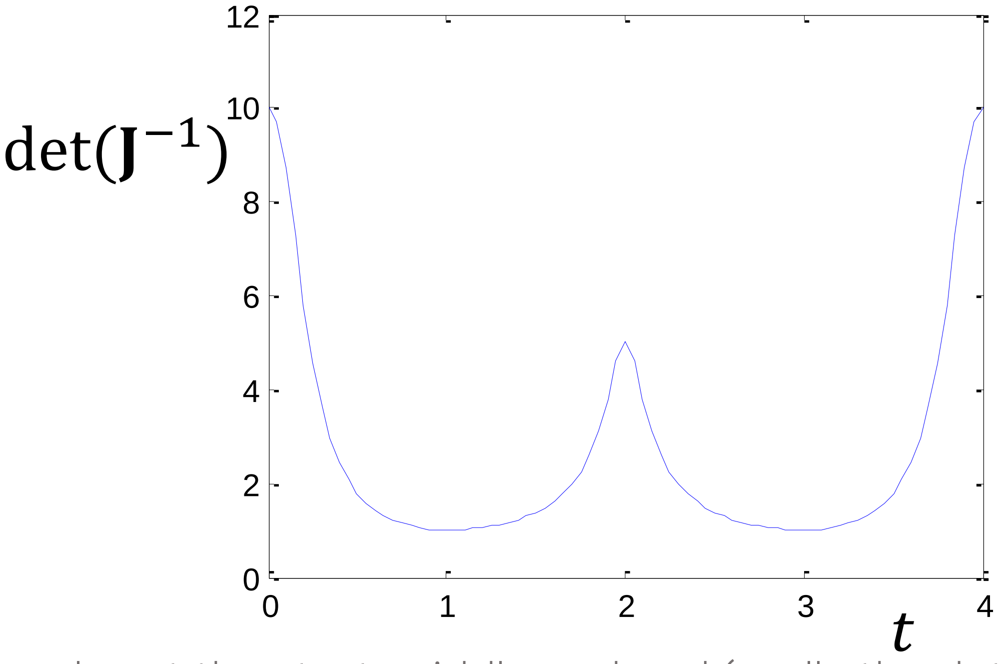


Joint accelerations are peaking at the start, end (joint 1) and middle (joint 2)

Result: Jacobian Singularities

The high joint accelerations may be suspected from the other plots, but they're not easy to see.

We must calculate the $\det(\mathbf{J})$ or $\det(\mathbf{J}^{-1})$



$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$$

$$\dot{\mathbf{x}} = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}$$

The matrix is becoming singular at the start, middle and end (really the determinant is much higher / closer to zero before we call it near singular)

Conclusions

- Precise trajectory specification / determination as well as joint control is a defining feature of how robotic manipulators and locomotion is implemented (not transient set point changes)
- This ensures robots behave in a predictable fashion, avoiding obstacles, foot placement on stairs, …
- Done by interpolating set-points, specified either in joint or Cartesian (operational) space
- Using piecewise cubic polynomial ensures that a continuous (piecewise linear) acceleration is achieved
- Polynomial parameter determination is formed as a linear matrix problem
- Interpolation, smoothness and end-point constraints are enough to uniquely determine the parameters
- Must analyse trajectory in joint space as well as Cartesian

Thank you

{Learn, Create, Innovate};

T&C

Terms and conditions

{Learn, Create, Innovate};

Terms and conditions

- *THE PIECES, IMAGES, VIDEOS, DOCUMENTATION, ETC. SHOWN HERE ARE FOR INFORMATIVE PURPOSES ONLY. THE DESIGN IS PROPRIETARY AND CONFIDENTIAL TO MANCHESTER ROBOTICS LTD. (MCR2). THE INFORMATION, CODE, SIMULATORS, DRAWINGS, VIDEOS PRESENTATIONS ETC. CONTAINED IN THIS PRESENTATION IS THE SOLE PROPERTY OF MANCHESTER ROBOTICS LTD. ANY REPRODUCTION, RESELL, REDISTRIBUTION OR USAGE IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF MANCHESTER ROBOTICS LTD. IS STRICTLY PROHIBITED.*
- *THIS PRESENTATION MAY CONTAIN LINKS TO OTHER WEBSITES OR CONTENT BELONGING TO OR ORIGINATING FROM THIRD PARTIES OR LINKS TO WEBSITES AND FEATURES IN BANNERS OR OTHER ADVERTISING. SUCH EXTERNAL LINKS ARE NOT INVESTIGATED, MONITORED, OR CHECKED FOR ACCURACY, ADEQUACY, VALIDITY, RELIABILITY, AVAILABILITY OR COMPLETENESS BY US.*
- *WE DO NOT WARRANT, ENDORSE, GUARANTEE, OR ASSUME RESPONSIBILITY FOR THE ACCURACY OR RELIABILITY OF ANY INFORMATION OFFERED BY THIRD-PARTY WEBSITES LINKED THROUGH THE SITE OR ANY WEBSITE OR FEATURE LINKED IN ANY BANNER OR OTHER ADVERTISING.*