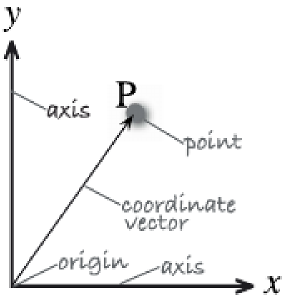
Fundamentals for robot manipulators

Universidade Federal de Pernambuco
Adrien Durand-Petiteville
adrien.durandpetiteville@ufpe.br

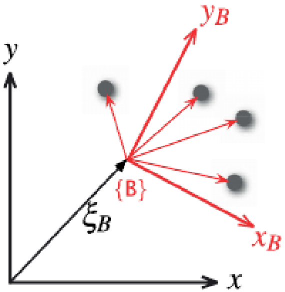
Point

- A fundamental requirement in robotics is to represent the position and orientation of objects in an environment
- A point in space can be described by a coordinate vector
- The vector represents the displacement of the point with respect to some reference coordinate frame
- A coordinate frame, or Cartesian coordinate system, is a set of orthogonal axes which intersect at a point known as the origin



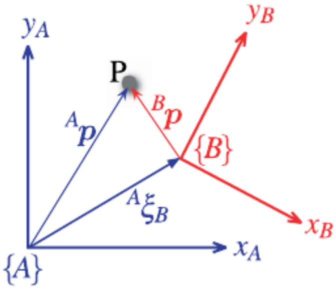
Set of points

- More frequently we need to consider a set of points that comprise some object.
- We assume that the object is rigid and that its constituent points maintain a constant relative position with respect to the object's coordinate frame.
- Instead of describing the individual points we describe the position and orientation of the object by the position and orientation of its coordinate frame $\{B\}$.



Relative pose

- The relative pose of a frame with respect to a reference coordinate frame is denoted by the symbol ξ
- Relative pose ${}^{A}\xi_{B}$ describes $\{B\}$ with respect to $\{A\}$
  - The superscript denotes the reference coordinate frame
  - The subscript denotes the frame being described

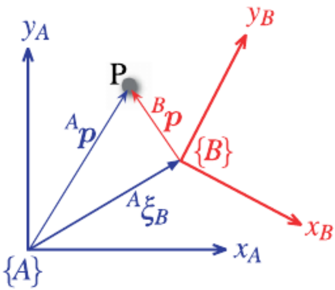


- We could also think of ${}^{A}\xi_{B}$ as describing some motion: applying a dis placement and a rotation on $\{A\}$ so that it is transformed to $\{B\}$

- The point **P** can be described with respect to either coordinate frame
- Formally we express this as:

$$^{A}\mathbf{p} = {}^{A}\xi_{B} . {}^{B}\mathbf{p}$$

- The operator . transforms the vector, resulting in a new vector that describes the same point but with respect to a different coordinate frame.

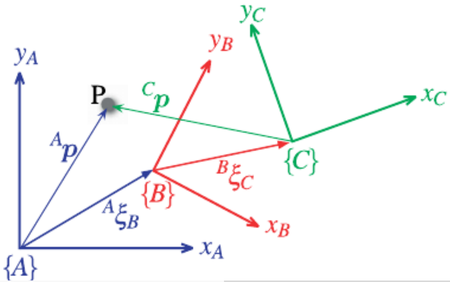


Composition of relative poses

- Relative poses can be composed
- If one frame can be described in terms of another by a relative pose then they can be applied sequentially:

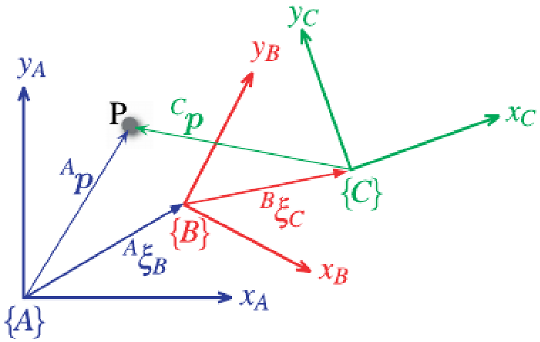
$$ {}^{A}\xi_{C} = {}^{A}\xi_{B} \oplus {}^{B}\xi_{C} $$

  - The pose of $\{C\}$ relative to $\{A\}$ can be obtained by compounding the relative poses from $\{A\}$ to $\{B\}$ and $\{B\}$ to $\{C\}$
  - We use the operator $\oplus$ to indicate composition of relative poses.



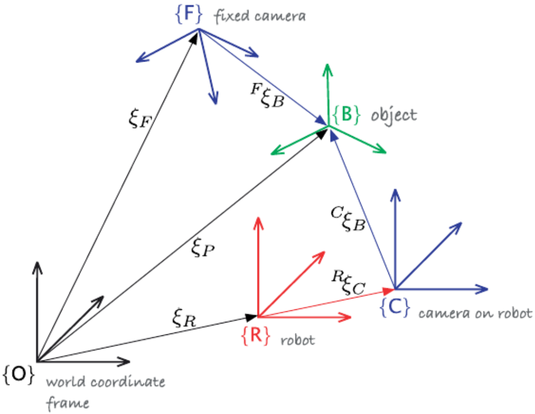
- For this case the point **P** can be described:

$$^{A}\mathbf{p} = (^{A}\xi_{B} \oplus {}^{B}\xi_{C}).{}^{C}\mathbf{p}$$



3-dimensional coordinate frames

- So far we have shown 2-dimensional coordinate frames
- For other problems we require 3-dimensional coordinate frames to describe objects in our 3-dimensional world such as the pose of the end of a tool carried by a robot arm



Directed graph

- An alternative representation of the spatial relationships is a directed graph
  - Each node represents a pose
  - Each edge represents a relative pose

- We can compose relative poses using the $\oplus$ operator:

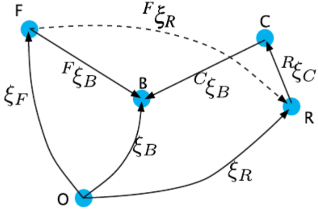
$$\xi_F \oplus {}^F\xi_B = \xi_R \oplus {}^R\xi_C \oplus {}^C\xi_B$$

$$\xi_F \oplus {}^F\xi_R = \xi_R$$

- We can subtract ξ_F from both sides of the equation by adding the inverse of ξ_F which we denote as $\ominus\xi_F$ and this gives:

$$\ominus\xi_F \oplus \xi_F \oplus {}^F\xi_R = \ominus\xi_F \oplus \xi_R$$

$$^F\xi_R = \ominus\xi_F \oplus \xi_R$$



Representing Pose in 2-Dimensions

Cartesian coordinate system

- To represent a 2-dimensional world, we use a Cartesian coordinate system
  - Two orthogonal axes denoted x and y and typically drawn with the x-axis horizontal and the y-axis vertical
  - The point of intersection is called the origin
- Unit-vectors parallel to the axes are denoted $\hat{x}$ and $\hat{y}$
- A point is represented by its x- and y-coordinates (x, y) or as a bound vector:

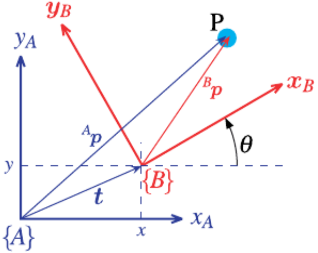
$$p = x\hat{x} + y\hat{y}$$

- Let's consider a coordinate frame $\{B\}$ to describe with respect to the reference frame $\{A\}$
- The origin of $\{B\}$ has been displaced by the vector $t = (x, y)$ and then rotated counter-clockwise by an angle θ

- A concrete representation of pose is therefore the 3-vector ${}^{A}\xi_{B} \sim (x, y, \theta)$ ($\sim$ denotes that the two representations are equivalent)
- Unfortunately this representation is not convenient for compounding since

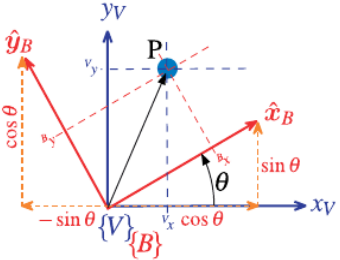
$$(x_1, y_1, \theta_1) \oplus (x_2, y_2, \theta_2)$$

is a complex trigonometric function of both poses



- Let's consider a point **P** with respect to each of the coordinate frames and to determine the relationship between ${}^{A}p$ and ${}^{B}p$
- We first consider the **rotation** problem
- We create a new frame $\{V\}$ whose axes are parallel to those of $\{A\}$ but whose origin is the same as $\{B\}$
- We can express the point P with respect to $\{V\}$ in terms of the unit-vectors that define the axes of the frame:

$$
\begin{aligned}
{}^{V}p &= {}^{V}x\hat{\boldsymbol{x}}_V + {}^{V}y\hat{\boldsymbol{y}}_V \\
&= \begin{pmatrix} \hat{\boldsymbol{x}}_V & \hat{\boldsymbol{y}}_V \end{pmatrix} \begin{pmatrix} {}^{V}x \\ {}^{V}y \end{pmatrix}
\end{aligned}
\qquad (1)
$$



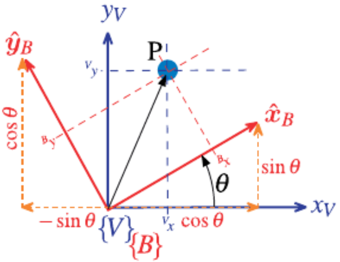
Rotation

- The coordinate frame $\{B\}$ is completely described by its two orthogonal axes which we represent by two unit vectors

$$\hat{\boldsymbol{x}}_B = \cos\theta \hat{\boldsymbol{x}}_V + \sin\theta \hat{\boldsymbol{y}}_V$$
$$\hat{\boldsymbol{y}}_B = -\sin\theta \hat{\boldsymbol{x}}_V + \cos\theta \hat{\boldsymbol{y}}_V$$

- It can be factorized into matrix form as

$$\begin{pmatrix} \hat{\boldsymbol{x}}_B & \hat{\boldsymbol{y}}_B \end{pmatrix} = \begin{pmatrix} \hat{\boldsymbol{x}}_V & \hat{\boldsymbol{y}}_V \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \tag{2}$$



- We can represent the point **P** with respect to $\{B\}$ as

$$\begin{aligned} {}^{B}p &= {}^{B}x\hat{\boldsymbol{x}}_B + {}^{B}y\hat{\boldsymbol{y}}_B \\ &= \begin{pmatrix} \hat{\boldsymbol{x}}_B & \hat{\boldsymbol{y}}_B \end{pmatrix} \begin{pmatrix} {}^{B}x \\ {}^{B}y \end{pmatrix} \end{aligned}$$

- Substituting (2) we write

$${}^{B}p = \begin{pmatrix} \hat{\boldsymbol{x}}_V & \hat{\boldsymbol{y}}_V \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} {}^{B}x \\ {}^{B}y \end{pmatrix} \tag{3}$$

- Now by equating the coefficients of the right-hand sides of (1) and (3) we write

$$\begin{pmatrix} {}^{V}x \\ {}^{V}y \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} {}^{B}x \\ {}^{B}y \end{pmatrix}$$

  which describes how points are transformed from frame $\{B\}$ to frame $\{V\}$ when the frame is rotated
- This type of matrix is known as a rotation matrix and denoted ${}^{V}\boldsymbol{R}_{B}$

$$\begin{pmatrix} {}^{V}x \\ {}^{V}y \end{pmatrix} = {}^{V}\boldsymbol{R}_{B} \begin{pmatrix} {}^{B}x \\ {}^{B}y \end{pmatrix}$$

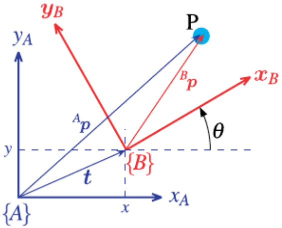
- The rotation matrix ${}^{V}\boldsymbol{R}_B$ is orthonormal
- Orthonormal matrices have the very convenient property that $\boldsymbol{R}^{-1} = \boldsymbol{R}^T$, that is, the inverse is the same as the transpose.

$$\begin{pmatrix} {}^{B}x \\ {}^{B}y \end{pmatrix} = \left({}^{V}\boldsymbol{R}_B\right)^{-1} \begin{pmatrix} {}^{V}x \\ {}^{V}y \end{pmatrix} = \left({}^{V}\boldsymbol{R}_B\right)^{T} \begin{pmatrix} {}^{V}x \\ {}^{V}y \end{pmatrix} = {}^{B}\boldsymbol{R}_V \begin{pmatrix} {}^{V}x \\ {}^{V}y \end{pmatrix}$$

Translation

- The second part of representing pose is to account for the translation between the origins of the frames
- Since the axes $\{V\}$ and $\{A\}$ are parallel this is simply vectorial addition

$$
\begin{aligned}
\begin{pmatrix} {}^{A}x \\ {}^{A}y \end{pmatrix} &= \begin{pmatrix} {}^{V}x \\ {}^{V}y \end{pmatrix} + \begin{pmatrix} x \\ y \end{pmatrix} \\
&= \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} {}^{B}x \\ {}^{B}y \end{pmatrix} + \begin{pmatrix} x \\ y \end{pmatrix} \\
&= \begin{pmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \end{pmatrix} \begin{pmatrix} {}^{B}x \\ {}^{B}y \\ 1 \end{pmatrix}
\end{aligned}
$$



Homogeneous transformation

- More compactly

$$\begin{pmatrix} {}^{A}x \\ {}^{A}y \\ 1 \end{pmatrix} = \begin{pmatrix} {}^{A}\boldsymbol{R}_{B} & \boldsymbol{t} \\ \mathbf{0}_{1\times 2} & 1 \end{pmatrix} \begin{pmatrix} {}^{B}x \\ {}^{B}y \\ 1 \end{pmatrix}$$

  - $\boldsymbol{t} = (x, y)$ is the translation of the frame
  - ${}^{A}\boldsymbol{R}_{B}$ is the orientation of the frame

- The coordinate vectors for point **P** are now expressed in homogeneous form

$$\begin{aligned} {}^{A}\tilde{\boldsymbol{p}} &= \begin{pmatrix} {}^{A}\boldsymbol{R}_{B} & \boldsymbol{t} \\ \mathbf{0}_{1\times 2} & 1 \end{pmatrix} {}^{B}\tilde{\boldsymbol{p}} \\ &= {}^{A}\boldsymbol{T}_{B}\, {}^{B}\tilde{\boldsymbol{p}} \end{aligned}$$

${}^{A}\boldsymbol{T}_{B}$ is referred to as a **homogeneous transformation**

Relative pose

- ${}^{A}\boldsymbol{T}_{B}$ represents relative pose

$$\xi(x, y, \theta) \sim \begin{pmatrix} \cos\theta & \sin\theta & x \\ -\sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{pmatrix}$$

- **First rule:** $T_1 \oplus T_2 \mapsto T_1 T_2$

$$T_1 T_2 = \begin{pmatrix} R_1 & t_1 \\ 0_{1\times 2} & 1 \end{pmatrix} \begin{pmatrix} R_2 & t_2 \\ 0_{1\times 2} & 1 \end{pmatrix} = \begin{pmatrix} R_1 R_2 & t_1 + R_1 t_2 \\ 0_{1\times 2} & 1 \end{pmatrix}$$

- **Second rule:**

$$\boldsymbol{T}^{-1} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}_{1\times 2} & 1 \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{R}^T & -\boldsymbol{R}^T \boldsymbol{t} \\ \boldsymbol{0}_{1\times 2} & 1 \end{pmatrix}$$

Exercise - 1

- Spatialmath Python toolbox
- https://github.com/petercorke/spatialmath-python
- **SO(2): rotation in 2D**

SO(2) matrix

class **SO2**(**args, **kwargs*) [source]

Bases: `spatialmath.baseposematrix.BasePoseMatrix`

SO(2) matrix class

This subclass represents rotations in 2D space. Internally it is a 2x2 orthogonal matrix belonging to the group SO(2).

Exercise - 2

`__init__`(*arg=None, *, unit='rad', check=True*) [source]

Construct new SO(2) object

Parameters
- **unit** (*str, optional*) – angular units 'deg' or 'rad' [default] if applicable
- **check** (*bool*) – check for valid SO(2) elements if applicable, default to True

Returns SO(2) rotation

Return type SO2 instance

- `SO2()` is an SO2 instance representing a null rotation – the identity matrix.
- `SO2(θ)` is an SO2 instance representing a rotation by `θ` radians. If `θ` is array_like *[θ1, θ2, … θN]* then an SO2 instance containing a sequence of N rotations.
- `SO2(θ, unit='deg')` is an SO2 instance representing a rotation by `θ` degrees. If `θ` is array_like *[θ1, θ2, … θN]* then an SO2 instance containing a sequence of N rotations.

- **SE(2): pose (rotation + translation) in 2D**

SE(2) matrix

class **SE2**(**args, **kwargs*) [source]

Bases: `spatialmath.pose2d.SO2`

SE(2) matrix class

This subclass represents rigid-body motion (pose) in 2D space. Internally it is a 3x3 homogeneous transformation matrix belonging to the group SE(2).

Exercise - 4

__init__(*x=None, y=None, theta=None, *, unit='rad', check=True*) [source]

Construct new SE(2) object

Parameters
- **unit** (*str, optional*) – angular units 'deg' or 'rad' [default] if applicable
- **check** (*bool*) – check for valid SE(2) elements if applicable, default to True

Returns SE(2) matrix

Return type SE2 instance

- `SE2()` is an SE2 instance representing a null motion – the identity matrix
- `SE2(θ)` is an SE2 instance representing a pure rotation of `θ` radians
- `SE2(θ, unit='deg')` as above but `θ` in degrees
- `SE2(x, y)` is an SE2 instance representing a pure translation of (`x`, `y`)

Exercise - 5

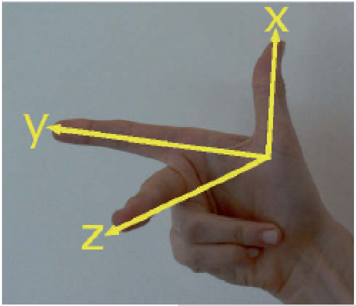
- SO(2)
  - Create a rotation matrix R_1 representing a $-\pi/2$ rad rotation
  - Create a rotation matrix R_2 representing a 90° rotation
  - Compute the new coordinates of $a = [1, 0]$ after a $-\pi/2$ rad rotation
  - Compute the new coordinates of $a = [1, 0]$ after a 90° rad rotation
- SE(2)
  - Create an homogenous matrix H_1 representing a $\pi/2$ rad rotation
  - Create an homogenous matrix H_2 representing a $[1, 2]$ translation
  - Create an homogenous matrix H_3 representing a $[1, 2]$ translation and a $\pi/2$ rad rotation
  - Compute the new coordinates of $a = [1, 0]$ after a $\pi/2$ rad rotation
  - Compute the new coordinates of $a = [1, 0]$ after a $[1, 2]$ translation
  - Compute the new coordinates of $a = [1, 0]$ after a $\pi/2$ rad rotation and a $[1, 2]$ translation

Representing Pose in 3-Dimensions

The 3-dimensional case

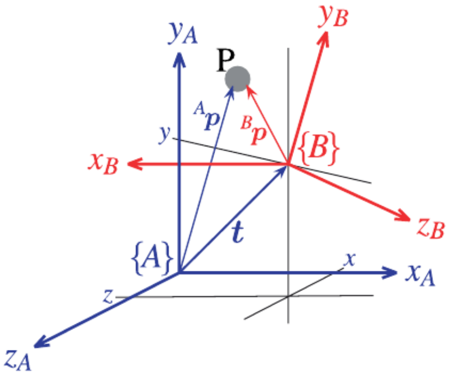
- The 3-dimensional case is an extension of the 2-dimensional
- We add an extra coordinate axis, typically denoted by z, that is orthogonal to both the x- and y-axes
- The direction of the z-axis obeys the right-hand rule and forms a right-handed coordinate frame
- A point **P** is represented by its x-, y- and z-coordinates (x, y, z) or as a bound vector

$$\boldsymbol{P} = x\hat{\boldsymbol{x}} + y\hat{\boldsymbol{y}} + z\hat{\boldsymbol{z}}$$



Representation of pose

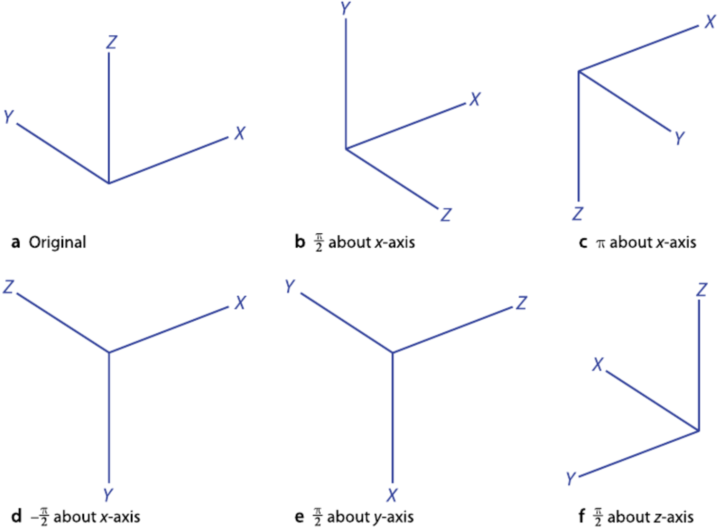
- A coordinate frame $\{B\}$ has to be described with respect to the reference frame $\{A\}$
- We can see clearly that the origin of $\{B\}$ has been displaced by the vector $\boldsymbol{t} = (x, y, z)$ and then rotated in some complex fashion
- Just as for the 2-dimensional case the way we represent orientation is very important



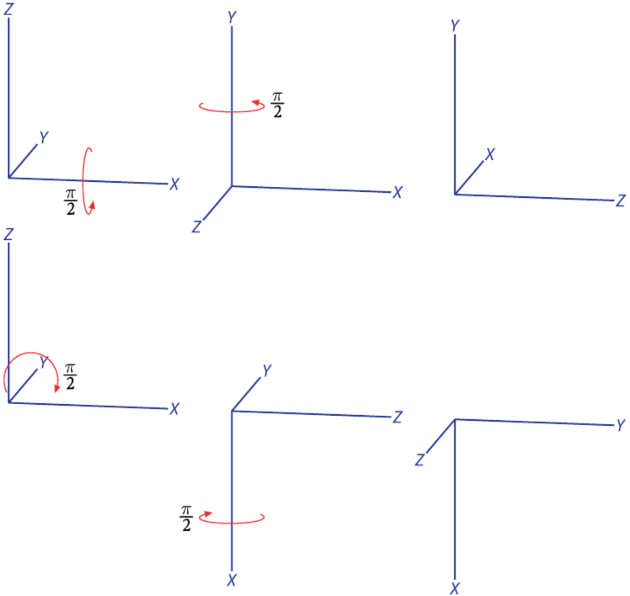
Representing Orientation in 3-Dimensions

Any two independent orthonormal coordinate frames can be related by a sequence of rotations (not more than three) about coordinate axes, where no two successive rotations may be about the same axis. Euler's rotation theorem (Kuipers 1999)

Rotation of a 3D coordinate frame



Example showing the non-commutivity of rotation



Example showing the non-commutivity of rotation - 2

Sequence of two rotations applied in different orders

- The final orientation depends on the order in which the rotations are applied
- This is a deep and confounding characteristic of the 3-dimensional world
- The implication for the pose algebra is that the $\oplus$ operator is not commutative

  The order in which rotations are applied is very important
- There exists many ways to represent rotation
  - Orthonormal rotation matrices
  - Euler and Cardan angles
  - Rotation axis and angle
  - Unit quaternions

Orthonormal Rotation Matrix

- Just as for the 2-dimensional case we can represent the orientation of a coordinate frame by its unit vectors expressed in terms of the reference coordinate frame
- Each unit vector has three elements and they form the columns of a 3 x 3 orthonormal matrix ${}^{A}\boldsymbol{R}_{B}$

$$\begin{pmatrix} {}^{A}x \\ {}^{A}y \\ {}^{A}z \end{pmatrix} = {}^{A}\boldsymbol{R}_{B} \begin{pmatrix} {}^{B}x \\ {}^{B}y \\ {}^{B}z \end{pmatrix}$$

which rotates a vector defined with respect to frame $\{B\}$ to a vector with respect to $\{A\}$

Orthonormal Rotation Matrix

- The orthonormal rotation matrices for rotation of θ about the x-, y- and z-axes are

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$$

$$R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$

$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- **SO(3): rotation in 3D**

Constructor	rotation
SO3.Rx(theta)	about X-axis
SO3.Ry(theta)	about Y-axis
SO3.Rz(theta)	about Z-axis
SO3.RPY(rpy)	from roll-pitch-yaw angle vector
SO3.Eul(euler)	from Euler angle vector
SO3.AngVec(theta, v)	from rotation and axis
SO3.Exp(v)	from a twist vector
SO3.OA	from orientation and approach vectors

- SO(3)
  - Create a rotation matrix R_x representing a $\pi/2$ rad rotation around the x axis
  - Create a rotation matrix R_y representing a $\pi/2$ rad rotation around the y axis
  - Create a rotation matrix R_z representing a $\pi/2$ rad rotation around the z axis
  - Create a rotation matrix R_{xyz} representing three $\pi/2$ rad rotations around the x, y and then z axis
  - Create a rotation matrix R_{xzy} representing three $\pi/2$ rad rotations around the x, z and then y axis

Three-Angle Representations

- Euler's rotation theorem requires successive rotation about three axes such that no two successive rotations are about the same axis
- There are two classes of rotation sequence: Eulerian and Cardanian, named after Euler and Cardano respectively
- The Eulerian type involves repetition, but not successive, of rotations about one particular axis: XYX, XZX, YXY, YZY, ZXZ, or ZYZ
- The Cardanian type is characterized by rotations about all three axes: XYZ, XZY, YZX, YXZ, ZXY, or ZYX
- In common usage all these sequences are called Euler angles and there are a total of twelve to choose from

Euler angles

- It is common practice to refer to all 3-angle representations as Euler angles but this is underspecified since there are twelve different types to choose from
- The ZYZ sequence $\boldsymbol{R} = \boldsymbol{R}_z(\phi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_z(\psi)$ is commonly used in aeronautics and mechanical dynamics
- Another widely used convention is the roll-pitch-yaw angle sequence angle $\boldsymbol{R} = \boldsymbol{R}_x(\theta_r)\,\boldsymbol{R}_y(\theta_p)\,\boldsymbol{R}_z(\theta_y)$ which are intuitive when describing the attitude of vehicles such as ships, aircraft and cars
- A fundamental problem with the three-angle representations just described is **singularity**
- **This occurs when the rotational axis of the middle term in the sequence becomes parallel to the rotation axis of the first or third term**

Unit Quaternions

- Quaternions were discovered by W. R. Hamilton over 150 years ago and have great utility for roboticists
- The quaternion is an extension of the complex number - a hyper-complex number - and is written as a scalar plus a vector

$$
\begin{aligned}
\mathring{q} &= s + \boldsymbol{v} \\
&= s + v_1 i + v_2 j + v_3 k
\end{aligned}
$$

where $s \in \mathbb{R}$, $\boldsymbol{v} \in \mathbb{R}^3$ and the orthogonal complex numbers i, j and k are defined such that

$$
i^2 = j^2 = k^2 = ijk = -1
$$

Unit Quaternions - 2

- We denote a quaternion as

$$\overset{\circ}{q} = s < v_1, v_2, v_3 >$$

- The quaternion or Hamilton product is

$$\overset{\circ}{q}_1 \oplus \overset{\circ}{q}_2 \mapsto s_1 s_2 - \boldsymbol{v}_1 \cdot \boldsymbol{v}_2, < s_1 \boldsymbol{v}_2 + s_2 \boldsymbol{v}_1 + \boldsymbol{v}_1 \times \boldsymbol{v}_2 >$$

- The quaternion conjugate is

$$\ominus \overset{\circ}{q} \mapsto \overset{\circ}{q}^{-1} = s, < -\boldsymbol{v} >$$

- The identity quaternion is

$$0 \mapsto 1 < 0, 0, 0 >$$

Exercise

- SO(3)
  - Create a rotation matrix R_{euler} representing three $\pi/2$ rad rotations using the Euler angles
  - Create a rotation matrix R_{rpy} representing three $\pi/2$ rad rotations using the Roll, Pitch, Yaw angles

Combining translation and orientation

- We return now to representing relative **pose** in three dimensions, the **position and orientation** change, between two coordinate frames
- The two most practical representations are:
  - the quaternion vector pair
  - 4x4 homogeneous transformation matrix
- A homogeneous transformation matrix describes rotation and translation:

$$\begin{pmatrix} {}^{A}x \\ {}^{A}y \\ {}^{A}z \\ 1 \end{pmatrix} = \begin{pmatrix} {}^{A}\boldsymbol{R}_{B} & \boldsymbol{t} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \begin{pmatrix} {}^{B}x \\ {}^{B}y \\ {}^{B}z \\ 1 \end{pmatrix}$$

  - $\boldsymbol{t}$: Cartesian translation vector between the origin of the coordinates frames
  - 3x3 orthonormal submatrix $\boldsymbol{R}$: change in orientation

Homogeneous transformation matrix

- With the vectors expressed in homogeneous form, we write:

$$
\begin{aligned}
{}^{A}\tilde{\boldsymbol{p}} &= \begin{pmatrix} {}^{A}\boldsymbol{R}_{B} & \boldsymbol{t} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} {}^{B}\tilde{\boldsymbol{p}} \\
&= {}^{A}\boldsymbol{T}_{B}\, {}^{B}\tilde{\boldsymbol{p}}
\end{aligned}
$$

- ${}^{A}\boldsymbol{T}_{B}$ is a 4x4 homogeneous transformation
- Standard matrix multiplication

$$
\boldsymbol{T}_1\boldsymbol{T}_2 = \begin{pmatrix} \boldsymbol{R}_1 & \boldsymbol{t}_1 \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{R}_2 & \boldsymbol{t}_2 \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{R}_1\boldsymbol{R}_2 & \boldsymbol{t}_1 + \boldsymbol{R}_1\boldsymbol{t}_2 \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix}
$$

- $$
\boldsymbol{T}^{-1} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{R}^{T} & -\boldsymbol{R}^{T}\boldsymbol{t} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix}
$$

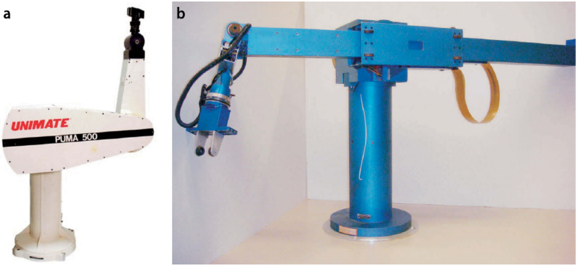
Exercise

- SE(3)
  - Create an homogenous matrix H_{tx} representing a 1 m translation along the x axis
  - Create an homogenous matrix H_{ty} representing a 2 m translation along the y axis
  - Create an homogenous matrix H_{tz} representing a 3 m translation along the y axis
  - Create an homogenous matrix H_t representing three 1, 2 and 3 m translations along the x, y and z axis
  - Compute the new coordinates of $a = [1, 0, 0]$ after three 1, 2 and 3 m translations along the x, y and z axis
  - Create an homogenous matrix H_r representing a $\pi/2$ rad rotation around the x axis
  - Compute the new coordinates of $a = [1, 0, 0]$ after three 1, 2 and 3 m translations along the x, y and z axis and one $\pi/2$ rad rotation around the x axis

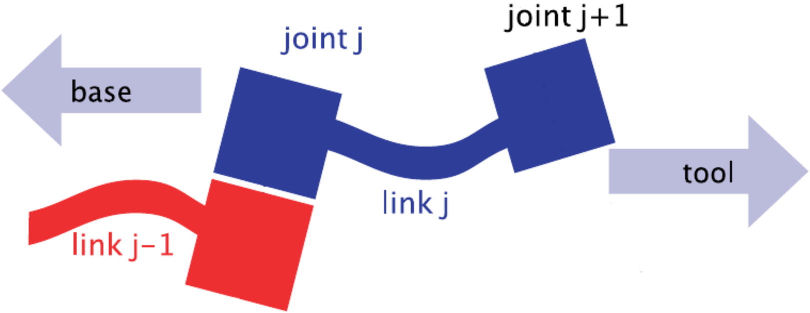
Robot Arm Kinematics

Serial-link manipulator

- A serial-link manipulator comprises a set of bodies, called **links**, in a chain and connected by **joints**
- Each joint has one degree of freedom:
  - either translational (a sliding or prismatic joint)
  - or rotational (a revolute joint)
- Motion of the joint changes the relative angle or position of its neighbouring links
- One end of the chain, the base, is generally fixed and the other end is free to move in space and holds the tool or end-effector.

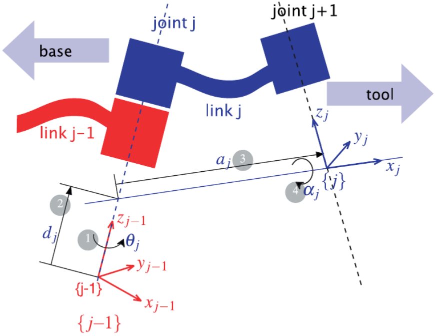


- For a manipulator with N joints numbered from 1 to N, there are $N + 1$ links, numbered from 0 to N
- Link 0 is the base of the manipulator and link N carries the end-effector or tool
- Joint j connects link $j - 1$ to link j and therefore joint j moves link j
- A link is considered a rigid body that defines the spatial relationship between two neighbouring joint axes



Parameters

- A link can be specified by two parameters
  - its length a_j
  - its twist α_j
- Joints are also described by two parameters.
  - The link offset d_j is the distance from one link coordinate frame to the next along the axis of the joint
  - The joint angle θ_j is the rotation of one link with respect to the next about the joint axis



Link transformation

- The transformation from link coordinate frame $j-1$ to frame j is defined in terms of elementary rotations and translations as

$$^{j-1}\boldsymbol{A}_j\left(\theta_j, d_j, a_j, \alpha_j\right) = \boldsymbol{T}_{Rz}\left(\theta_j\right)\boldsymbol{T}_z\left(d_j\right)\boldsymbol{T}_x\left(a_j\right)\boldsymbol{T}_{Rx}\left(\alpha_j\right)$$

which can be expanded as

$$^{j-1}A_j = \begin{pmatrix} \cos\theta_j & -\sin\theta_j\cos\alpha_j & \sin\theta_j\sin\alpha_j & a_j\cos\theta_j \\ \sin\theta_j & \cos\theta_j\cos\alpha_j & -\cos\theta_j\sin\alpha_j & a_j\sin\theta_i \\ 0 & \sin\alpha_j & \cos\alpha_j & d_j \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Denavit-Hartenberg parameters

Joint angle	θ_j	the angle between the x_{j-1} and x_j axes about the z_{j-1} axis	revolute joint variable
Link offset	d_j	the distance from the origin of frame $j-1$ to the x_j axis along the z_{j-1} axis	prismatic joint variable
Link length	a_j	the distance between the z_{j-1} and z_j axes along the x_j axis; for intersecting axes is parallel to $\hat{z}_{j-1} \times \hat{z}_j$	constant
Link twist	α_j	the angle from the z_{j-1} axis to the z_j axis about the x_j axis	constant
Joint type	σ_j	$\sigma = 0$ for a revolute joint, $\sigma = 1$ for a prismatic joint	constant

- The parameters α_j and a_j are always constant
- For a **revolute** joint θ_j is the joint variable and d_j is constant
- For a **prismatic** joint d_j is variable, θ_j is constant and $\alpha_j = 0$

Generalized joint coordinates

- In many of the formulations that follow we use **generalized joint** coordinates

$$q_j = \begin{cases} \theta_j & \sigma_j = 0, \text{ for a revolute joint} \\ d_j & \sigma_j = 1, \text{ for a prismatic joint} \end{cases}$$

- For an N-axis robot the generalized joint coordinates $q \in \mathcal{C}$ where $\mathcal{C} \subset \mathbb{R}^N$ is called the joint space or configuration space
- The joint coordinates are also referred to as the pose of the manipulator
- The pose of the end-effector is a Cartesian pose ξ

Configuration space vs task space

- Configuration space
  - The configuration of a robot can be completely described by a parameter $\mathbf{q}$ which is called its **generalized coordinate**
  - The set of all possible configurations is the configuration space, or C-space, denoted by $\mathcal{C}$ and $\mathbf{q} \in \mathcal{C}$
- Task space
  - The task space is the set of all possible poses ξ of the robot and $\xi \in \mathcal{T}$
  - The task space depends on the application or task
    - If we cared only about the position of the train in a plane then $\mathcal{T} \subset \mathbb{R}^2$
    - If we considered a 3-dimensional world then $\mathcal{T} \subset SE3$

Forward Kinematics

- The forward kinematics determines the pose of the end-effector given the joint coordinates
- The forward kinematics is often expressed in functional form

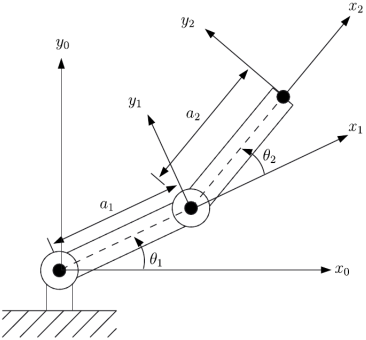
$$\xi_E = \mathcal{K}(\boldsymbol{q})$$

with the end-effector pose as a function of joint coordinates
- Using homogeneous transformations this is simply the product of the individual link transformation matrices

$$\xi_E \sim {}^0\boldsymbol{T}_E = {}^0\boldsymbol{A}_1 {}^1\boldsymbol{A}_2 \cdots {}^{N-1}\boldsymbol{A}_N$$

- The forward kinematic solution can be computed for **any** serial-link manipulator irrespective of the number of joints or the types of joints

Example: A 2-Link Robot



- We consider a two-link planar manipulator
- It has the following Denavit-Hartenberg parameters

Link	θ_i	d_i	a_i	α_i	σ_i
1	q_1	0	1	0	0
2	q_2	0	1	0	0

- Given the desired pose of the end-effector ξ_E what are the required joint coordinates?
- This is the inverse kinematics problem which is written in functional form as

$$\boldsymbol{q} = \mathcal{K}^{-1}(\xi)$$

- In general this function is not unique and for some classes of manipulator no closed- form solution exists, necessitating a numerical solution

Example

- RoboticsToolbox for Python
- https://github.com/petercorke/robotics-toolbox-python
- Planar robot with 2 revolute joints

Path and trajectories

Path and trajectories

- A **path** is a spatial construct - a locus in space that leads from an initial pose to a final pose.
  For example, there is a path from A to B.
- A **trajectory** is a path with specified timing.
  For example, there is a trajectory from A to B in 10 s or at 2 m/s.

Smooth One-Dimensional Trajectories - 1

- We start our discussion with a scalar function of time.
- Important characteristics of this function are that its initial and final value are specified and that it is smooth.
- Smoothness in this context means that its first few temporal derivatives are continuous.
- Typically velocity and acceleration are required to be continuous and sometimes also the derivative of acceleration or jerk.

Smooth One-Dimensional Trajectories - 2

- An obvious candidate for such a function is a polynomial function of time.
- Polynomials are simple to compute and can easily provide the required smoothness and boundary conditions.
- A quintic (fifth-order) polynomial is often used

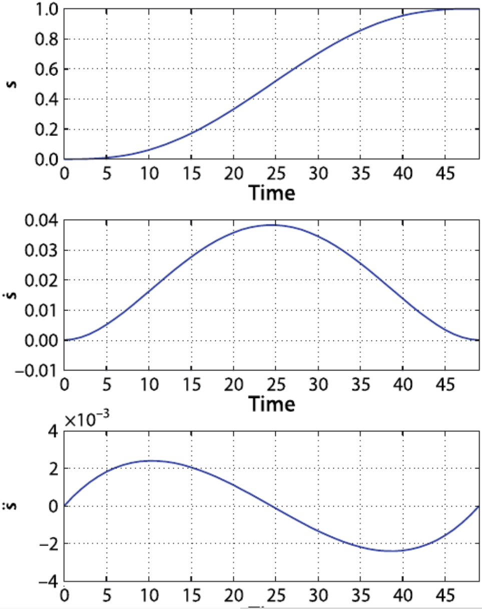
$$S\langle t\rangle = At^5 + Bt^4 + Ct^3 + Dt^2 + Et + F$$

  where time $t \in [0, T]$.
- The first- and second-derivatives are also smooth polynomials

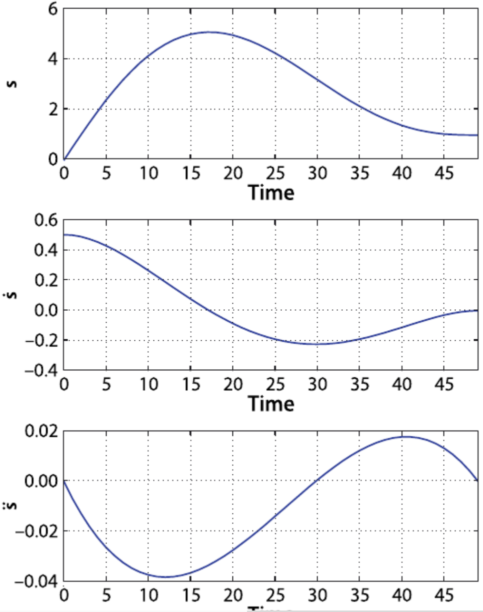
$$\dot{S}\langle t\rangle = 5At^4 + 4Bt^3 + 3Ct^2 + 2Dt + E$$

$$\ddot{S}\langle t\rangle = 20At^3 + 12Bt^2 + 6Ct + 2D$$

With zero-velocity boundary conditions



Initial velocity of 0.5 and a final velocity of 0



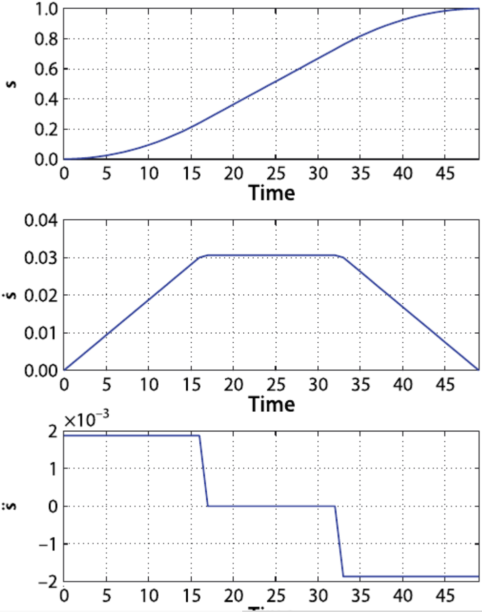
Comments

- The second results illustrate a problem with polynomials.
- The non-zero initial velocity causes the polynomial to overshoot the terminal value – it peaks at 5 on a trajectory from 0 to 1.
- Another problem with polynomials, a very practical one, can be seen in the velocity graph.
- The velocity peaks at $t = 25$ which means that for most of the time the velocity is far less than the maximum.
- A real robot joint has a well defined maximum velocity and for minimum-time motion we want to be operating at that maximum for as much of the time as possible.

LSPB - 1

- A well known alternative is a hybrid trajectory which has a constant velocity segment with polynomial segments for acceleration and deceleration.
- The trajectory comprises a linear segment (constant velocity) with parabolic blends.
- The term blend is commonly used to refer to a trajectory segment that smoothly joins linear segments.
- This type of trajectory is also referred to as trapezoidal due to the shape of the velocity curve versus time, and is commonly used in industrial motor drives.

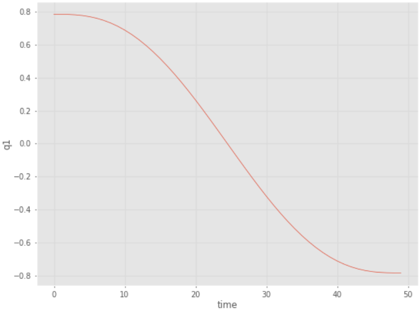
LSPB - 2


1.0
0.8
0.6
0.4
0.2
0.0
s
0 5 10 15 20 25 30 35 40 45
Time
0.04
0.03
0.02
0.01
0.00
ṡ
0 5 10 15 20 25 30 35 40 45
Time
×10^-3
2
1
0
−1
−2
s̈
0 5 10 15 20 25 30 35 40 45

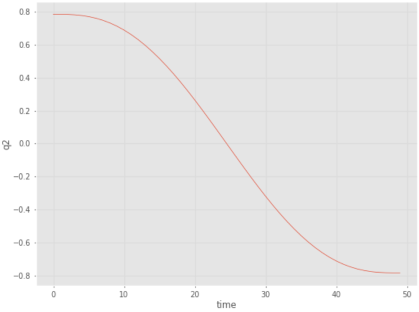
Trajectories

- One of the most common requirements in robotics is to move the end-effector smoothly from pose A to pose B
- We discuss two approaches to generating such trajectories:
  - straight lines the joint space
  - straight lines the Cartesian space
- These are known respectively as joint-space and Cartesian motion.

Joint space - 1

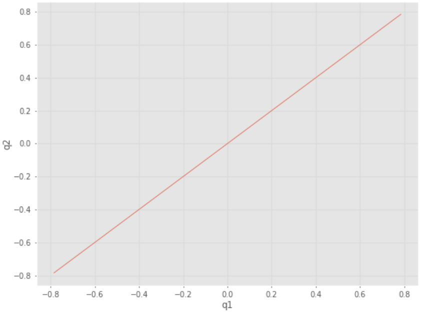


q1

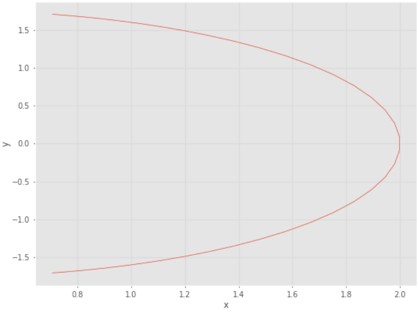


q2

Joint space - 2

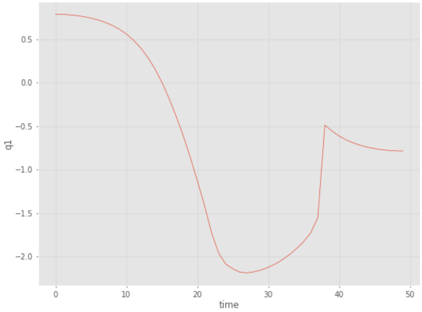


q1-q2

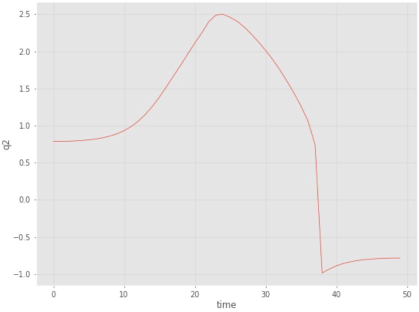


x-y

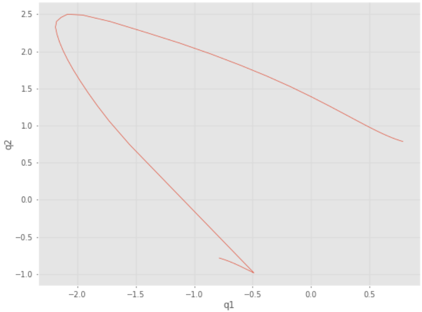
Cartesian space - 1



q1

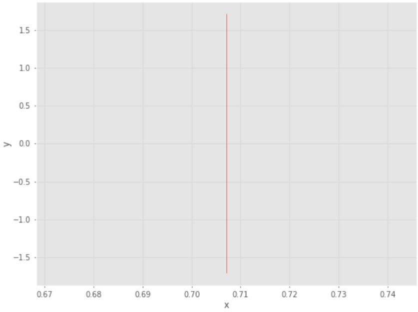


q2



q2
q1
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5

q1-q2



y
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.67
0.68
0.69
0.70
0.71
0.72
0.73
0.74

x-y