Bilkent University

CS 449 Learning For Robotics

Homework 1: Report

Alper Mumcular 21902740

Vesile İrem Aydın 21902914

Ece Kahraman 21801879

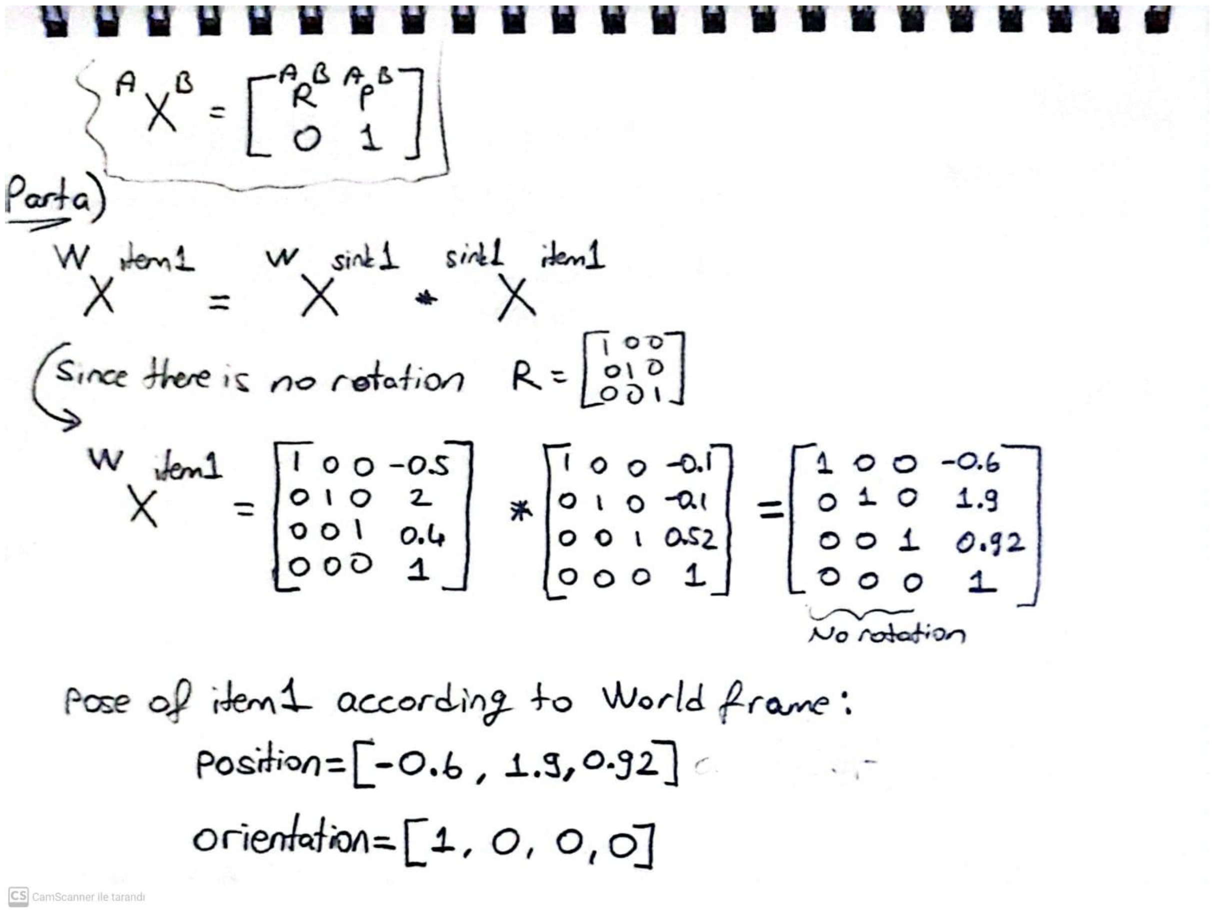
Gülçin Özkahya 21903129

Orkun Taha Öznergiz 21803058

1) Frames

1.1) Part a

The question asks us to find position and orientation. To find the pose of the object relative to the world frame, we can do the following calculations.



$${}^{A}X^{B} = \begin{bmatrix} {}^{A}R^{B} & {}^{A}P^{B} \\ 0 & 1 \end{bmatrix}$$

Part a)

$${}^{W}X^{item1} = {}^{W}X^{sink1} * {}^{sink1}X^{item1}$$

Since there is no rotation $R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$${}^{W}X^{item1} = \begin{bmatrix} 1 & 0 & 0 & -0.5 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0.4 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & -0.1 \\ 0 & 1 & 0 & -0.1 \\ 0 & 0 & 1 & 0.52 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -0.6 \\ 0 & 1 & 0 & 1.9 \\ 0 & 0 & 1 & 0.92 \\ \underbrace{0 \quad 0 \quad 0}_{\text{No rotation}} & & & 1 \end{bmatrix}$$

Pose of item1 according to World frame:

position=[-0.6, 1.9, 0.92]

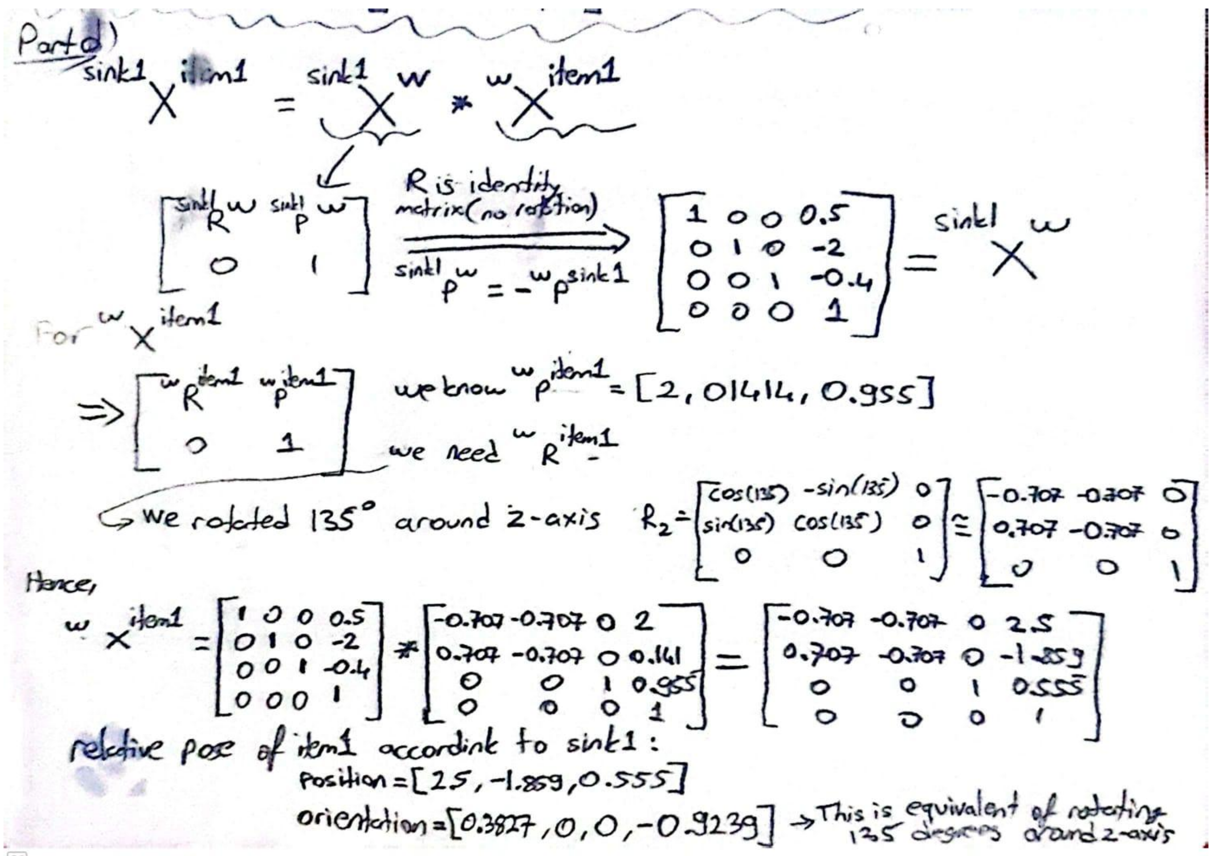
orientation=[1, 0, 0, 0]

Figure 1: Answer of Part a

1.2) Part c

It didn't change because the relative position of item1 according to item2 and the tray is still the same. The relative positions of item1 and item2 are still the same because we moved the tray, the parent object, when we transitioned from State1 to State2. Similarly, item1 did not change its position on the tray, so the relative position of item1 and the tray is still the same.

1.3) **Part d**



Part d)

$$^{sink1}X^{item1} = \underbrace{^{sink1}X^{w}} * \underbrace{^{w}X^{item1}}$$

$$\begin{bmatrix} ^{sink1}R^{w} & ^{sink1}p^{w} \\ 0 & 1 \end{bmatrix} \xrightarrow[^{sink1}p^{w} = -^{w}p^{sink1}]{\text{R is identity matrix(no rotation)}} \begin{bmatrix} 1 & 0 & 0 & 0.5 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & -0.4 \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^{sink1}X^{w}$$

For $^{w}X^{item1}$

$$\Rightarrow \begin{bmatrix} ^{w}R^{item1} & ^{w}p^{item1} \\ 0 & 1 \end{bmatrix}$$

we know $^{w}p^{item1} = [2, 0.1414, 0.955]$

we need $^{w}R^{item1}$

↳ we rotated $135°$ around z-axis $R_2 = \begin{bmatrix} \cos(135) & -\sin(135) & 0 \\ \sin(135) & \cos(135) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cong \begin{bmatrix} -0.707 & -0.707 & 0 \\ 0.707 & -0.707 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Hence,

$$^{w}X^{item1} = \begin{bmatrix} 1 & 0 & 0 & 0.5 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & -0.4 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} -0.707 & -0.707 & 0 & 2 \\ 0.707 & -0.707 & 0 & 0.141 \\ 0 & 0 & 1 & 0.955 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -0.707 & -0.707 & 0 & 2.5 \\ 0.707 & -0.707 & 0 & -1.859 \\ 0 & 0 & 1 & 0.555 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

relative pose of item1 accordink to sink1:

position $= [2.5, -1.859, 0.555]$

orientation $= [0.3827, 0, 0, -0.9239]$ → This is equivalent of rotating 135 degrees around z-axis

Figure 2: Answer of Part d

2) Two Link Manipulator

Forward Kinematics:

$^{A}p^{B} = (y, z) = (L\cos(q_0) , L\sin(q_0)) = (\cos(q_0) , \sin(q_0))$

$^{B}p^{C} = (L\cos(q_0+q_1) , L\sin(q_0+q_1)) = (\cos(q_0+q_1) , \sin(q_0+q_1))$

$^{A}p^{C} = {}^{A}p^{B} + {}^{B}p^{C} = (\cos(q_0) + \cos(q_0+q_1) ,\ \sin(q_0) + \sin(q_0+q_1))$

$^{A}p^{C} (q_0,q_1) = (\cos(q_0) + \cos(q_0+q_1) ,\ \sin(q_0) + \sin(q_0+q_1))$

As a result,

$$y = \cos(q_0) + \cos(q_0+q_1)$$

$$z = \sin(q_0) + \sin(q_0+q_1)$$

2.1 Part c

$J(q) = [\ \partial z/\partial q_0\ \ \partial z/\partial q_1\ \ = [\ \cos(q_0) + \cos(q_0+q_1),\quad \cos(q_0+q_1)$

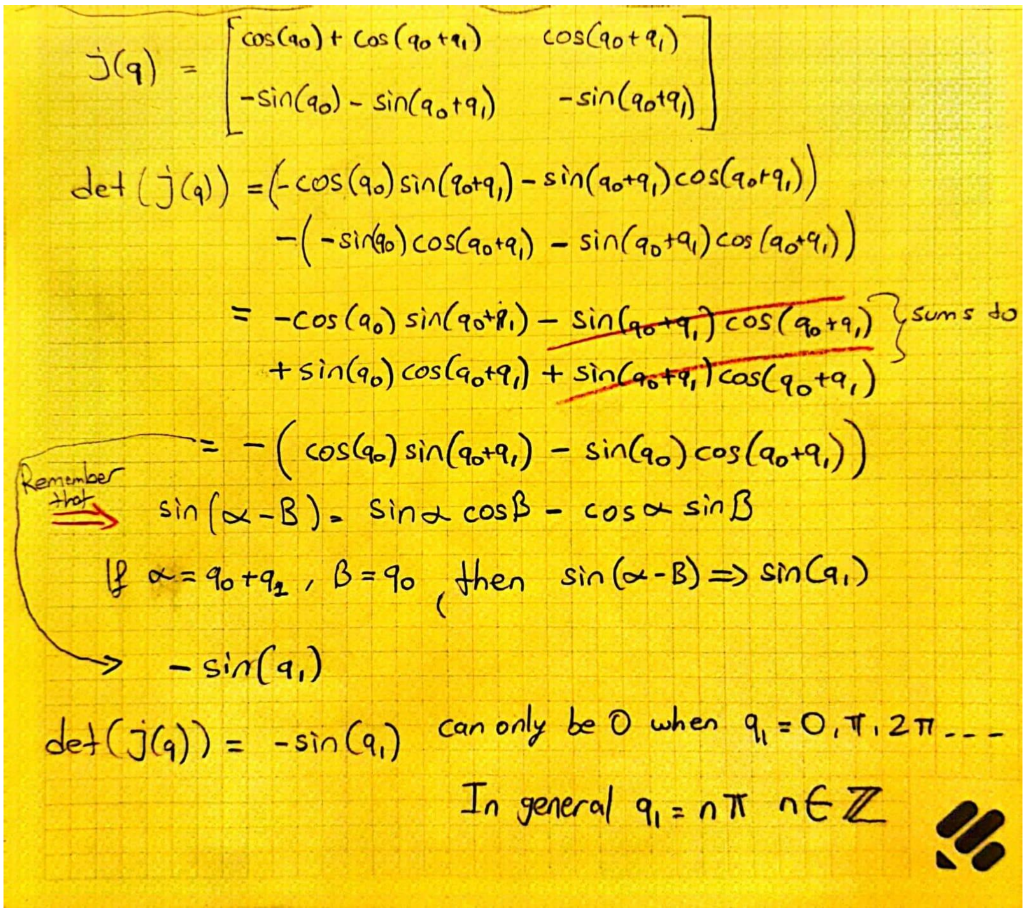
$\qquad\quad \partial y/\partial q_0\ \ \partial y/\partial q_1]\qquad -\sin(q_0) - \sin(q_0+q_1),\quad -\sin(q_0+q_1)\ ]$

2.2 Part d

The Jacobian matrix is a 2x2 square matrix, and a square matrix cannot be inverted when its determinant is equal to 0.

$J(q) = (\ \partial z/\partial q_0\ \ \partial z/\partial q_1$

$\qquad\quad \partial y/\partial q_0\ \ \partial y/\partial q_1)$



$$J(q) = \begin{bmatrix} \cos(q_0) + \cos(q_0+q_1) & \cos(q_0+q_1) \\ -\sin(q_0) - \sin(q_0+q_1) & -\sin(q_0+q_1) \end{bmatrix}$$

$$\det(J(q)) = \left(-\cos(q_0)\sin(q_0+q_1) - \sin(q_0+q_1)\cos(q_0+q_1)\right) - \left(-\sin(q_0)\cos(q_0+q_1) - \sin(q_0+q_1)\cos(q_0+q_1)\right)$$

$$= \left.\begin{matrix} -\cos(q_0)\sin(q_0+q_1) - \cancel{\sin(q_0+q_1)\cos(q_0+q_1)} \\ +\sin(q_0)\cos(q_0+q_1) + \cancel{\sin(q_0+q_1)\cos(q_0+q_1)} \end{matrix}\right\} \text{sums do}$$

$$= -\left(\cos(q_0)\sin(q_0+q_1) - \sin(q_0)\cos(q_0+q_1)\right)$$

Remember that $\Rightarrow$ $\sin(\alpha - \beta) = \sin\alpha\cos\beta - \cos\alpha\sin\beta$

If $\alpha = q_0 + q_2$, $\beta = q_0$, then $\sin(\alpha - \beta) \Rightarrow \sin(q_1)$

$\rightarrow\ -\sin(q_1)$

$\det(J(q)) = -\sin(q_1)$ can only be 0 when $q_1 = 0, \pi, 2\pi \ldots$

In general $q_1 = n\pi\ \ n \in \mathbb{Z}$

Figure 1: Determinant of the Jacobian

We can say that the determinant of Jacobian is equal to 0 when (see Figure 1 for the solution):

$$q_1 = n * \pi \qquad \text{where } n \in Z$$

3) Exploring the Jacobian

3.1) Part a

Size of the Jacobian of the planar three-link manipulator = 2x3

The Jacobian will be the following:

$$J(q) = \begin{pmatrix} \partial x/\partial q_0 & \partial x/\partial q_1 & \partial x/\partial q_2 \\ \partial y/\partial q_0 & \partial y/\partial q_1 & \partial y/\partial q_2 \end{pmatrix}$$

3.2) Part b

When the Jacobian matrix is square (n x n size) and has full rank, we can compute its exact inverse.

We cannot compute its exact inverse when it is not square or in full rank. We can compute its pseudo-inverse when the exact inverse is not possible.

In brief,

n x n size → Exact Inverse

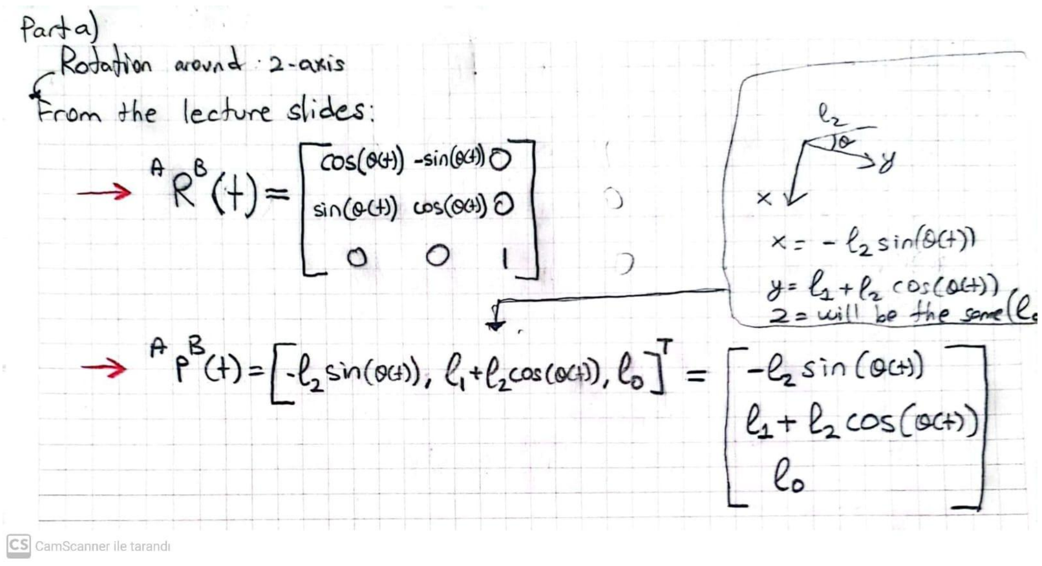
m x n size → Pseudo-inverse (where $m \neq n$)

3.3) Part c

It means the Jacobian loses rank, which results in singularity. Whenever this happens, the manipulability ellipsoid collapses into a line segment.

4) Spatial Velocity for Moving Frame

4.1) Part a



Part a)

Rotation around z-axis

From the lecture slides:

$$\rightarrow \; {}^{A}R^{B}(t) = \begin{bmatrix} \cos(\theta(t)) & -\sin(\theta(t)) & 0 \\ \sin(\theta(t)) & \cos(\theta(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

l_2 θ y x

$x = -l_2 \sin(\theta(t))$

$y = l_1 + l_2 \cos(\theta(t))$

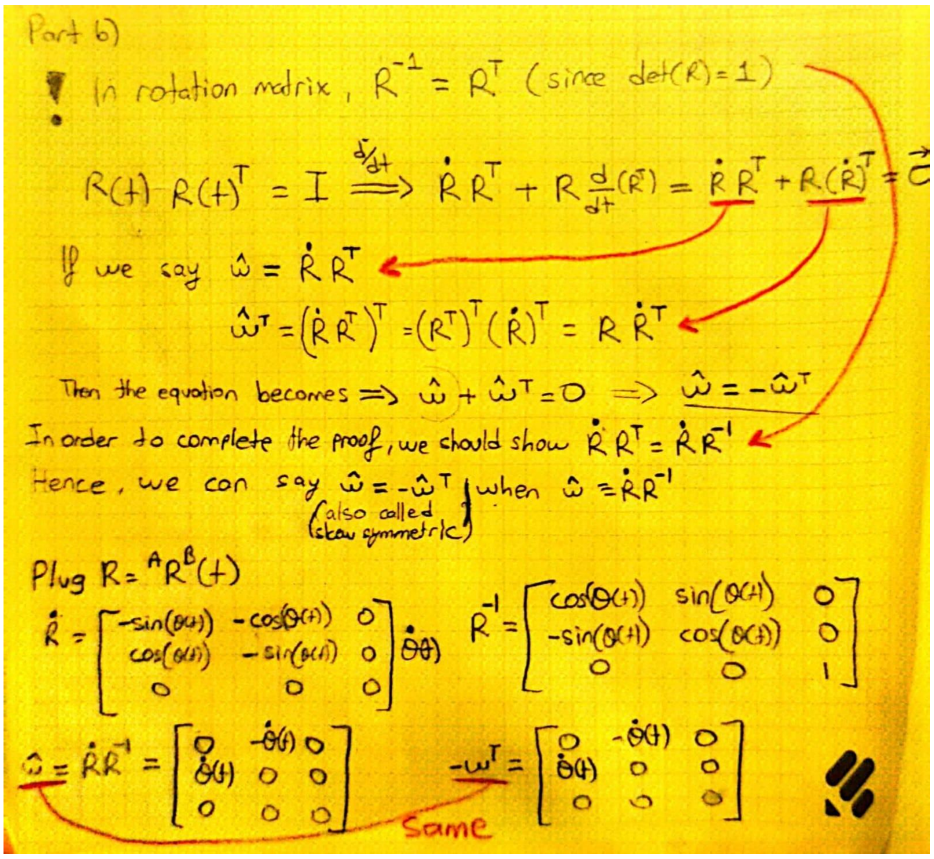
z = will be the same (l_0

$$\rightarrow \; {}^{A}p^{B}(t) = \left[-l_2 \sin(\theta(t)),\ l_1 + l_2 \cos(\theta(t)),\ l_0\right]^T = \begin{bmatrix} -l_2 \sin(\theta(t)) \\ l_1 + l_2 \cos(\theta(t)) \\ l_0 \end{bmatrix}$$

CamScanner ile tarandı

Figure 2: Answer of Part 4.1

4.2) Part b



Part b)

! In rotation matrix, $R^{-1} = R^T$ (since $\det(R) = 1$)

$$R(t)\,R(t)^T = I \xrightarrow{d/dt} \dot{R}R^T + R\frac{d}{dt}(R^T) = \dot{R}R^T + R(\dot{R})^T = \vec{0}$$

If we say $\hat{\omega} = \dot{R}R^T$

$$\hat{\omega}^T = (\dot{R}R^T)^T = (R^T)^T(\dot{R})^T = R\dot{R}^T$$

Then the equation becomes $\Rightarrow \hat{\omega} + \hat{\omega}^T = 0 \Rightarrow \underline{\hat{\omega} = -\hat{\omega}^T}$

In order to complete the proof, we should show $\dot{R}R^T = \dot{R}R^{-1}$

Hence, we can say $\hat{\omega} = -\hat{\omega}^T$ (also called skew symmetric) when $\hat{\omega} = \dot{R}R^{-1}$

Plug $R = {}^{A}R^{B}(t)$

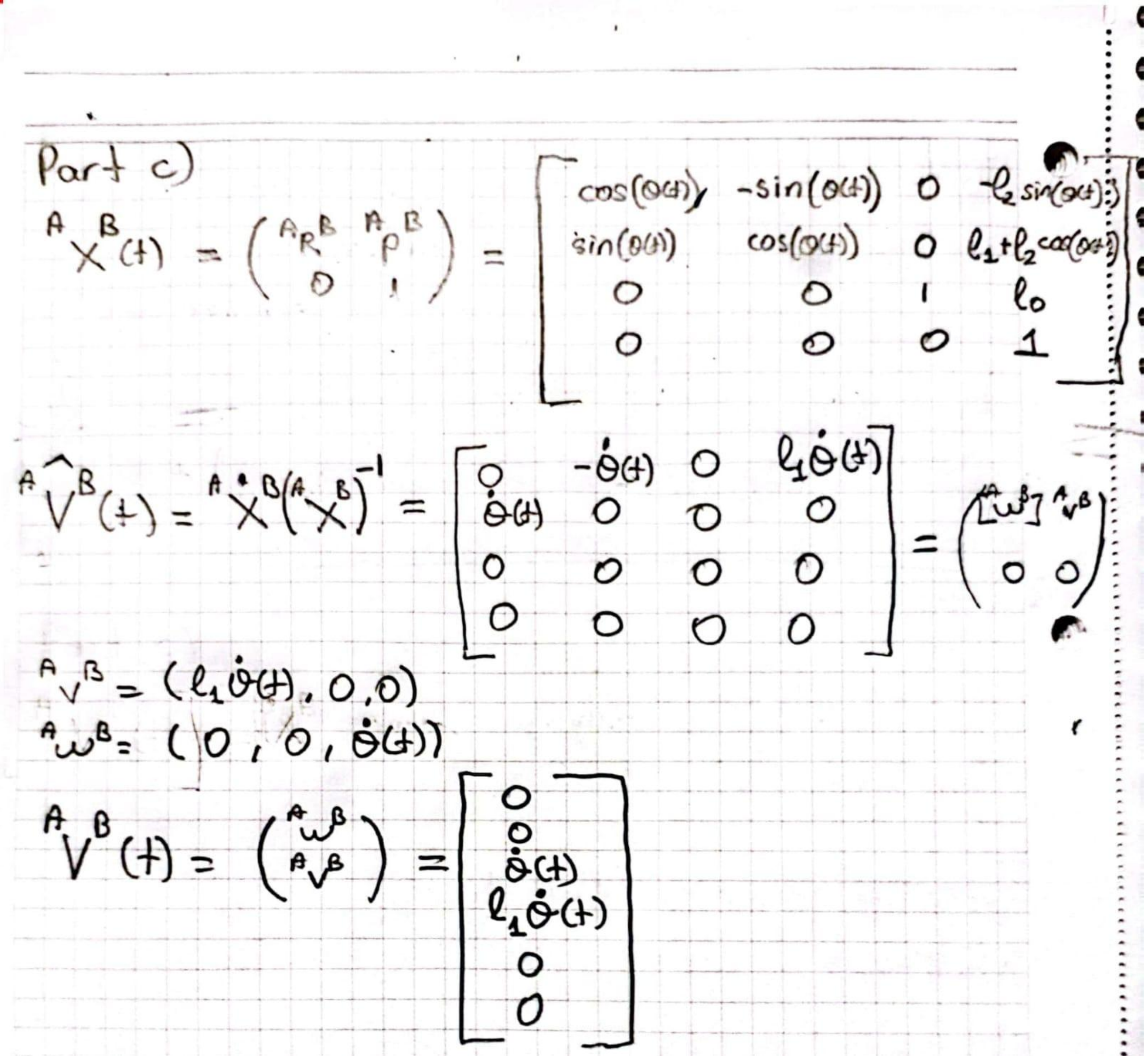
$$\dot{R} = \begin{bmatrix} -\sin(\theta(t)) & -\cos(\theta(t)) & 0 \\ \cos(\theta(t)) & -\sin(\theta(t)) & 0 \\ 0 & 0 & 0 \end{bmatrix} \dot{\theta}(t) \qquad R^{-1} = \begin{bmatrix} \cos(\theta(t)) & \sin(\theta(t)) & 0 \\ -\sin(\theta(t)) & \cos(\theta(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\underline{\hat{\omega}} = \dot{R}R^{-1} = \begin{bmatrix} 0 & -\dot{\theta}(t) & 0 \\ \dot{\theta}(t) & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \underline{-\omega^T} = \begin{bmatrix} 0 & -\dot{\theta}(t) & 0 \\ \dot{\theta}(t) & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Same

Figure 3: Answer of Part 4.2

4.3) Part c



Part c)

$${}^{A}X^{B}(t) = \begin{pmatrix} {}^{A}R^{B} & {}^{A}p^{B} \\ 0 & 1 \end{pmatrix} = \begin{bmatrix} \cos(\theta(t)) & -\sin(\theta(t)) & 0 & -\ell_2 \sin(\theta(t)) \\ \sin(\theta(t)) & \cos(\theta(t)) & 0 & \ell_1 + \ell_2 \cos(\theta(t)) \\ 0 & 0 & 1 & \ell_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^{A}\hat{V}^{B}(t) = {}^{A}\dot{X}^{B}({}^{A}X^{B})^{-1} = \begin{bmatrix} 0 & -\dot{\theta}(t) & 0 & \ell_1\dot{\theta}(t) \\ \dot{\theta}(t) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{pmatrix} [{}^{A}\omega^{B}] & {}^{A}v^{B} \\ 0 & 0 \end{pmatrix}$$

$${}^{A}v^{B} = (\ell_1\dot{\theta}(t), 0, 0)$$

$${}^{A}\omega^{B} = (0, 0, \dot{\theta}(t))$$

$${}^{A}V^{B}(t) = \begin{pmatrix} {}^{A}\omega^{B} \\ {}^{A}v^{B} \end{pmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}(t) \\ \ell_1\dot{\theta}(t) \\ 0 \\ 0 \end{bmatrix}$$

Figure 4: Answer of Part 4.3

5) Make Your Own Robot

5.1) Comparison of Optimization Based Method and Computational Method

In both IK methods, we tried to find the right joint angles of the robot arm to place the end effector in the correct position and orientation. However, we add different objective functions to the KOMO in the optimization-based method. By doing this, we tried to minimize the cost, or we defined constraint - by specifying that the gripper position should be equal to the target position. The advantage of the optimization-based IK is that it can be solved when there are various kinds of constraints in the problem, like overcoming obstacles -as we saw in tutorial 2. We were able to reach targets in the kitchen by using KOMO, so it accurately gave a solution to the problem.

A disadvantage of the KOMO can be considered that when there are multiple optimal points, such as local optimums and global optimums, it can find the local optimum. It can give a local optimum point as a solution, which is a suboptimal solution.

In the computational IK, we computed the end-effector's appropriate joint angles by benefitting from matrix operations using Jacobian. The advantage of this method may be that it is more computationally efficient than the optimization-based methods since there are more underlying operations in the optimization methods. Another advantage of computational IK might be that it is less likely to give a local minimum point as a solution, so probably computational IK gives a direct solution. However, the disadvantage of this method might be that computational IK may not overcome complex constraints as in the optimization-based IK.