RBE 595 — Reinforcement Learning
Week #2 Assignment

Arjan Gupta

Problem 1

What is the benefit of incremental update for action-value function, versus non-incremental?

Answer

The non-incremental implementation of the action-value function requires that we store all of the rewards that have been seen so far in each time-step. The formula for this would be,

$$Q_n = \frac{R_1 + R_2 + \cdots + R_{n-1}}{n-1}$$

Where Q_n is the current estimation of the reward R after $n-1$ time-steps. The problem with this approach is that, as the number of time-steps increases, the amount of memory required to store all of the rewards increases linearly, i.e. the space complexity is $\mathcal{O}(n)$.

Instead, we use the incremental update approach, which uses $\mathcal{O}(1)$ space complexity (constant memory). The formula for this can be derived as follows,

$$\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}(R_n + \sum_{i=1}^{n-1} R_i) \\
&= \frac{1}{n}(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i) \\
&= \frac{1}{n}(R_n + (n-1)Q_n) \\
&= \frac{1}{n}(R_n + nQ_n - Q_n) \\
&= Q_n + \frac{1}{n}(R_n - Q_n)
\end{aligned}$$

Computationally, this incremental approach is better as well, because it only requires one addition, one subtraction, and one division per time-step. The non-incremental approach requires $n-1$ additions, 1 subtraction, and one division per time-step. Therefore, the incremental approach is $\mathcal{O}(1)$ in terms of time complexity, while the non-incremental approach is $\mathcal{O}(n)$.

Problem 4.10

Two frames $o_0x_0y_0z_0$ and $o_1x_1y_1z_1$ are related by the homogeneous transformation

$$H = \begin{bmatrix} 0 & -1 & 0 & 1 \\ 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A particle has velocity $v_1(t) = (3, 1, 0)$ relative to frame $o_1x_1y_1z_1$. What is the velocity of the particle in frame $o_0x_0y_0z_0$?

Solution

The given H is the homogeneous transformation H_1^0. Therefore, $R_1^0 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and $o_1^0 = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}$.

At any given point in time, the position of the particle with respect to frame $o_1x_1y_1z_1$ is given as $p^1(t)$. We know that

$$p^0(t) = R_1^0 p^1(t) + o_1^0$$

Taking the derivative of both sides and using the product rule, we get

$$v^0(t) = \dot{R}_1^0 p^1(t) + R_1^0 v^1(t) + 0$$

But, $\dot{R}_1^0 = 0$ since R_1^0 is a constant in time. Therefore,

$$\begin{aligned} v^0(t) &= R_1^0 v^1(t) \\ v^0(t) &= \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \\ 0 \end{bmatrix} \\ v^0(t) &= \begin{bmatrix} -1 \\ 3 \\ 0 \end{bmatrix} \end{aligned}$$

Which was calculated by the following MATLAB script

Problem 4.15

Find the 6×3 Jacobian for the three links of the cylindrical manipulator of Figure 3.7. Find the singular configurations for this arm.

Solution

Figure 3.7 of our book is the following.

We can see that we have 3 joints, so $n = 3$. Let us also form the table given in the lecture videos:

	Linear component	Angular component
Revolute joint	$J_{v_i} = z_{i-1}^0 \times (o_n^0 - o_{i-1}^0)$	$J_{v_i} = z_{i-1}^0$
Prismatic joint	$J_{\omega_i} = z_{i-1}^0$	$J_{\omega_i} = 0$

Using this table, and the fact that the upper half of the Jacobian contains linear components while the bottom half contains angular components, we have

$$J = \begin{bmatrix} z_0 \times (o_3 - o_0) & z_1 & z_2 \\ z_0 & 0 & 0 \end{bmatrix}$$

In this Jacobian matrix, we know that $z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ and $o_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$. To find z_1, z_2, and o_3, we need to find $T_n^0 = A_1 \dots A_n$ for $n = 1, 2, 3$. Following the DH convention (which Figure 3.7 already abides by), we have the following table for quantities $\alpha_i, a_i, \theta_i, d_i$.

Link	α_i	a_i	θ_i	d_i
1	0	0	θ_1	d_1
2	-90°	0	0	d_2
3	0	0	0	d_3

Which gives us the following A_i matrices.

$$A_1 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, A_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}, A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Now we can compute our T matrices using the following MATLAB code,

$$T_1^0 = A_1 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^0 = A_1 A_2 = \begin{bmatrix} \cos(\theta_1) & 0 & -\sin(\theta_1) & 0 \\ \sin(\theta_1) & 0 & \cos(\theta_1) & 0 \\ 0 & -1 & 0 & d_1 + d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^0 = A_1A_2A_3 = \begin{bmatrix} \cos(\theta_1) & 0 & -\sin(\theta_1) & -d_3\,\sin(\theta_1) \\ \sin(\theta_1) & 0 & \cos(\theta_1) & d_3\,\cos(\theta_1) \\ 0 & -1 & 0 & d_1 + d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

From these T matrices, we get

$$z_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, z_2 = \begin{bmatrix} -\sin(\theta_1) \\ \cos(\theta_1) \\ 0 \end{bmatrix}, o_3 = \begin{bmatrix} -d_3\,\sin(\theta_1) \\ d_3\,\cos(\theta_1) \\ d_1 + d_2 \end{bmatrix}$$

Now we are ready to compute our Jacobian matrix. We do this using the following MATLAB code.
Which gives us the following Jacobian,

$$J = \begin{bmatrix} -d_3\,\cos(\theta_1) & 0 & -\sin(\theta_1) \\ -d_3\,\sin(\theta_1) & 0 & \cos(\theta_1) \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} = J_P = \begin{bmatrix} J_{11} \\ \hline J_{21} \end{bmatrix}$$

Where J_{11} and J_{21} are 3×3 matrices. By setting $\det J_{11} = 0$ we can find the singular configurations for the arm portion of the manipulator (first three joints). We calculate the determinant using the following MATLAB code,

Which gives us,

$$J_{11} = d_3 \cos^2(\theta_1) + d_3 \sin^2(\theta_1) = d_3(\cos^2(\theta_1) + \sin^2(\theta_1)) = d_3$$

When we set $J_{11} = 0$ here, we get $d_3 = 0$. Therefore, the only singular configuration for this manipulator will occur when the third joint variable (which is a prismatic joint) goes to 0.