SSY191 Individual Assignment 1

Abdulrahman Hameshli

April 2024

Problem 1

When a matrix is ortogonal, transpose is equal to its invers. To be orthonormal, a matrix's columns need to be the right length and at right angles to each other. In our case, since the Homogenise matrix has both translation and rotation, and since rotation can be described using cosine and sine, and the x, y, and z axes are at right angles to each other, our matrix ${}^{W}\xi_B$ is orthonormal. This means we can pick any point, P, in three dimensions using:

$$ {}^{B}\xi_W = \begin{bmatrix} R^T & -R^T * T \\ 0^{1x3} & 1 \end{bmatrix} $$

So:

$$ {}^{W}\xi_B = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

becomes:

$$ {}^{B}\xi W = \begin{bmatrix} r_{11} & r_{21} & r_{31} & -r_{11}t_1 - r_{21}t_2 - r_{31}t_3 \\ r_{12} & r_{22} & r_{32} & -r_{12}t_1 - r_{22}t_2 - r_{32}t_3 \\ r_{13} & r_{23} & r_{33} & -r_{13}t_1 - r_{23}t_2 - r_{33}t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Problem 2

The accelerometer records the forces acting on the Crazyflie drone relative to the world coordinate frame as:

$$^{B}f = m\,^{B}R_w \left(\begin{bmatrix} ^{w}a_x \\ ^{w}a_y \\ ^{w}a_z \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \right)$$

Here, ^{B}f represents the forces on the drone, $^{B}R_w$ is the rotation matrix from body to world, and m is the mass of the drone.

Assuming the drone flies at a constant velocity and forces are normalized to gravity, acceleration in the world frame equals zero. This simplifies the equation to:

$$^{B}f = m\,^{B}R_w \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Now using Euler-Angles YXZ our rotational matrix from world to base becomes:

$$^{w}R_B = R_z(\psi)R_x(\varphi)R_y(\theta)$$

And rotational matrix from base to world becomes:

$$^{B}R_w = (^{w}R_B)^T = R_y^T(\theta)R_x^T(\varphi)R_z^T(\psi)$$

With this inserted in the equation above, the following result yields:

$$\begin{bmatrix} F_y \\ F_x \\ F_z \end{bmatrix} = m \cdot R_y^T(\theta)R_x^T(\varphi)R_z^T(\psi) \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Using the MATLAB symbolic toolbox, the following result is obtained:

$$\begin{bmatrix} F_y \\ F_z \\ F_x \end{bmatrix} = m \cdot \begin{bmatrix} -\cos(\varphi)\sin(\theta) \\ \sin(\varphi) \\ \cos(\varphi)\cos(\theta) \end{bmatrix}$$

By comparing elements, we derive θ and φ:

$$\theta = \text{atan2}(-F_y, F_z)$$

$$\varphi = \text{atan2}\left(\frac{F_x}{\sqrt{F_y^2 + F_z^2}}, \sqrt{F_y^2 + F_z^2} \right)$$

Problem 3

We know that

$$G(S) = \frac{1}{\alpha s + 1}$$

$$\gamma = \frac{\alpha}{\alpha + h}$$

$$\theta_g(S) = \frac{1}{S} y(S)$$

$$\theta(S) = G(S)(\theta_a(S) - \theta_g(S)) + \theta_g(S)$$

$$\alpha S\theta(S) + \theta(S) = \theta_a(S) - \theta_g(S) + \alpha S\theta_g(S) + \theta_g(S)$$

$$\alpha S\theta(S) + \theta(S) = \theta_a(S) - \frac{1}{S}y(S) + \alpha S\frac{1}{S}y(S) + \frac{1}{S}y(S)$$

$$\alpha S\theta(S) + \theta(S) = \theta_a(S) + \alpha y(S)$$

Invers laplace transform gives:

$$\alpha\dot{\theta}(t) + \theta(t) = \theta_a(t) + \alpha y(t)$$

$$\dot{\theta}(t) = \frac{\theta_a(t) + \alpha y(t) - \theta(t)}{\alpha}$$

Euler backword approximation is :

$$\dot{\theta}(t) = \frac{\theta(kh) - \theta((k-1)h)}{h}$$

$$\frac{\theta(kh) - \theta((k-1)h)}{h} = \frac{\theta_a(kh)}{\alpha} + y(kh) + \frac{\theta(kh)}{\alpha}$$

$$\theta(kh)\left(\tfrac{1}{h}+\tfrac{1}{\alpha}\right)=\frac{\theta_a(kh)}{\alpha}+y(kh)+\frac{\theta((k-1)h)}{h}$$

$$\theta(kh)\frac{\alpha+h}{\alpha h}=\frac{\theta_a(kh)}{\alpha}+y(kh)+\frac{\theta((k-1)h)}{h}$$

$$\theta(kh)(\alpha+h)=h\theta_a(kh)+\alpha hy(kh)+\alpha\theta((k-1)h)$$

$$\theta(kh)=(1-\gamma)\theta_a(kh)+h\gamma y(kh)+\gamma\theta((k-1)h)$$

Problem 4

A

The automaton for the tank is showen in the following figure:

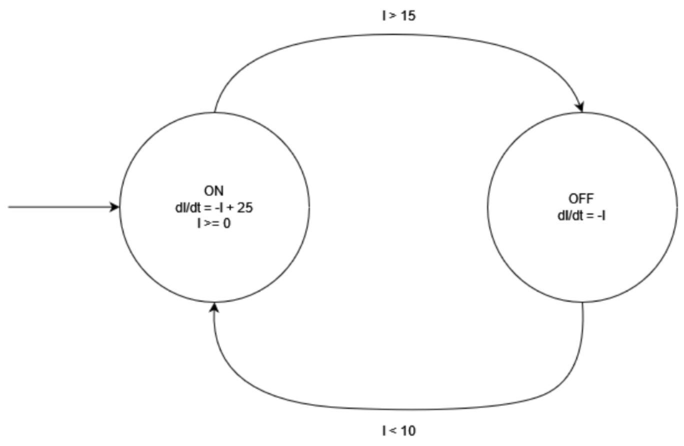

Figure 1: Water tank automaton

B

The figures "Zero-crossing detection enabled" and "Zero-crossing detection disabled" show how a certain feature affects the simulation. When it's enabled, things work better because it helps identify when certain values cross zero. Without it, there are problems with accuracy, especially noticeable in examples like the bouncing ball. Despite these issues, the system generally behaves as expected, with the water level staying within the desired range, as seen in the "System modeled in Simulink" figure.

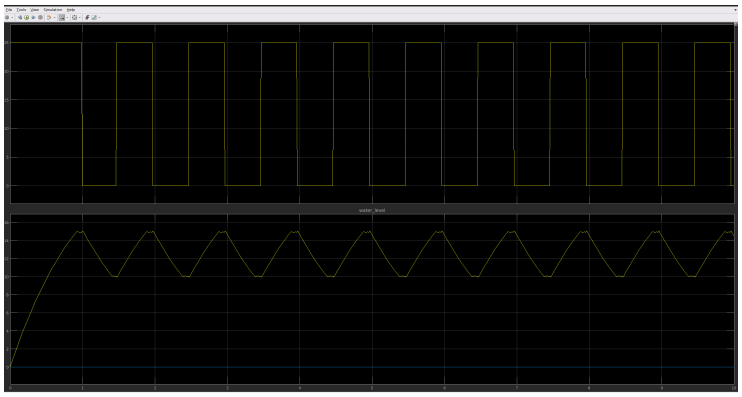

Figure 2: Disable zero-cross detection

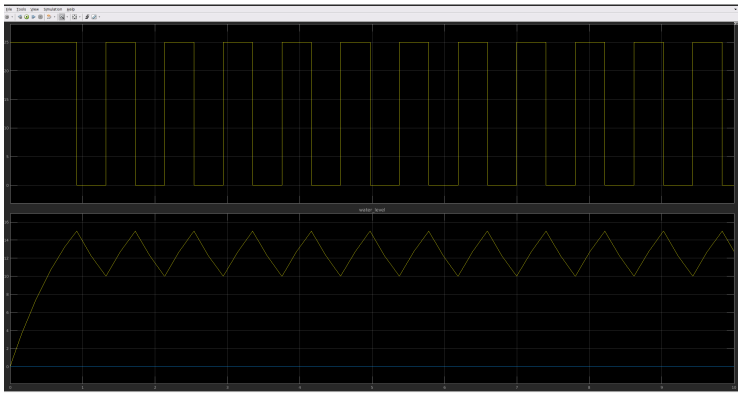

Figure 3: enable zero-cross detection

C

The Zeno effect is when a system attempts to perform an infinite number of actions in a limited time. In this situation, there are no Zeno effects occurring. Zeno solutions arise when there's a pattern causing states or modes to repeat infinitely. Looking at Figure 1, it's evident that such a pattern doesn't exist, preventing an infinite loop of recursion.