Hand in assignment 1

Johannes Lundahl Daniel Söderqvist

September 15, 2023

Introduction

The task consist of calculating and determine the dynamics of a system. The system consists of a helicopter that flies in the sky but having a mass hanging from its bottom. The following figures describe the relation between the helicopter, the mass and angles etc.

(a) Schematic of a helicopter hovering a mass

(b) Top view of the system

Figure 1: Figure showing the dynamics of the helicopter and its hanging mass.

1 Hovering mass

a) The generalized coordinates were given as:

$$q = \begin{bmatrix} p_1 \\ \theta \\ \phi \end{bmatrix} \in \mathbb{R}^5 \tag{1}$$

We decided to put the coordinate frame in the center of mass 1 (m_1). Since mass 2 (m_2) is connected by a rigid link it can move in a spherical-like pattern and thus we use spherical coordinates to describe the position of (m_2) relative to our reference coordinate system. But since mass 1 (m_1) can move, we had to describe the position of mass 2 (m_2) expressed together with the position coordinates for mass 1 (m_1). Position vector 2 then became:

$$p_2 = \begin{bmatrix} p_{1x} + L \cdot \sin\phi \cdot \cos\theta \\ p_{1y} + L \cdot \sin\phi \cdot \sin\theta \\ p_{1z} - L \cdot \cos\phi \end{bmatrix} \tag{2}$$

Further, we calculated the kinetic and potential energies using equation (3.5) and (3.21) from the lecture notes respectively. Since we had to masses (m_1) and (m_2) we calculated the summation of the potential energy as $V_{tot} = V_1 + V_2$. Then we calculated the Lagrange function using equation (3.33) and later on the Euler-Lagrange function in its transposed version using (3.36). In the E-L function we calculated the

different parts of the equation separately in matlab. Instead of using the gradient function we used the Jacobian function, calculating the partial derivatives, and then transpose it, which can be shown to be the same according to equation (3.37). To be able to calculate the first part of the E-L equation which requires a total differentiation with respect to time operator we used the chain rule as in equation (3.39). On the other hand in matlab, we did not shorten the gradient terms using W(q), instead we used the transposed Jacobian function of Lagrange ($\mathcal{L}$) with respect to q and $\dot{q}$ to determine the $\nabla_q \mathcal{L}$ and $\nabla_{\dot{q}} \mathcal{L}$.

Since we had external forces $u \in \mathbb{R}^3$ acting on the helicopter the Euler Lagrange equation gets another term as in equation (3.112). Since $q \in \mathbb{R}^5$ the Q vector becomes:

$$Q = \begin{bmatrix} u_x \\ u_y \\ u_z \\ 0 \\ 0 \end{bmatrix} \tag{3}$$

We were then asked to specify the Euler Lagrange function in a restructured way as:

$$M(q)\dot{v} = b(q, \dot{q}, u) \tag{4}$$

What we did was just to move everything that did not have an $\dot{q}$-term to the right hand side of the E-L equation. Those terms implies the second part of the chain rule development of the $\frac{d}{dt}(\nabla_{\dot{q}} \mathcal{L})$ term and also the $\nabla_q \mathcal{L}$ term.

$$\underbrace{\frac{\partial}{\partial \dot{q}}(\nabla_{\dot{q}} \mathcal{L})\ddot{q}}_{\text{M(q)=}} = \underbrace{Q + \frac{\partial}{\partial q}(\nabla_{\dot{q}} \mathcal{L})\dot{q} + \nabla_q \mathcal{L}}_{\text{b(q,}\dot{q}\text{,u)}} \tag{5}$$

b) In this sub task we were supposed to do the same thing as in 1. a) but this time to include constraints to the system. The generalized coordinates are also changed, which now consists of only the positions of both the helicopter and the mass as follows:

$$q = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \in \mathbb{R}^6 \tag{6}$$

What we are doing when dealing with constrained Lagrange mechanics is actually to constrain the dynamical system to evolve on a certain manifold named $\mathbf{c}(\mathbf{q})$. The constrain $\mathbf{c}(\mathbf{q})$ is added to the Lagrange function as in equation (3.125) and the Euler-Lagrange function is then instead described as in equation (3.128). Since $q = p$

in this case, the W(q) term becomes constant and thus the final M and b matrices becomes:

$$\underbrace{\frac{\partial}{\partial \dot{q}}(\nabla_{\dot{q}}\mathcal{L})\ddot{q}}_{\text{M(q)=}} = \underbrace{Q - \nabla_q V - \nabla_q c \cdot z}_{\text{b(q,z,u)}} \tag{7}$$

Compare the complexity of your two models (i.e. the complexity of the symbolic expressions for M and b). What do you conclude?

Answer: When looking at the 2nd model with the constraint, the general coordinates can be expressed in terms of only the positions. This make the W(q) constant, and thus turn the M(q)-matrix described above into a diagonal-matrix that is easy to invert. In the 1st model you express the relation between the two masses within the general coordinates by using θ, ϕ and L. This in turn make the M(q)-matrix a bit more complicated and thus a bit trickier to invert. Although the the 2nd model including the constraint has an additional term z, which needs to be given to be able to execute the simulation.

2 Explicit vs. Implicit model

a) The Euler-Lagrange equation including constraints, equation (3.167a, 3.167b), can be rewritten on its implicit form by some small algebraic manipulations as follows:

$$\underbrace{\begin{bmatrix} W(q) & \nabla_q c \\ \nabla_q c^\top & 0 \end{bmatrix}}_{:=K} \begin{bmatrix} \ddot{q} \\ z \end{bmatrix} = \begin{bmatrix} Q - \frac{\partial}{\partial q}(W(q)\dot{q})\dot{q} + \nabla_q T - \nabla_q V \\ -\frac{\partial}{\partial q}(\frac{\partial c}{\partial q}\dot{q})\dot{q} \end{bmatrix} \tag{8}$$

Where in our case $a(t) = \nabla_q c$ which is just the gradient of the constraint with respect to q, and $c(q, \dot{q}, u)$ is the right hand side of equation (8). The matrices can be found in the attached matlab file.

b) When writing the model on its explicit form, one can see that it's the same except from that we have moved over the K-matrix presented above to the right hand side of equation (8). To be able to do that, we have to take the inverse of the K-matrix (K^{-1}). Since the matrix is quite big and we are using symbolic values, this operation requires a lot of computational power and it also yields a very complex matrix, see matlab code. If we want to obtain the second derivative of the state vector for simulation purpose, its better to use numerical values.