IN3140/IN4140 - Assignment 3

Due: 8. April 2022, 23:59 (24h)

Introduction

In this assignment we will look at modeling the dynamics of a robot manipulator. We will continue to work with the CrustCrawler robot throughout both tasks of this assignment. The first task is designed to be solved with pen and paper, and introduces a well known method for deriving *Equations of Motion*. The second task is designed to be solved by computation, and investigates a method to compute the dynamic equations for a n-link robot manipulator. As we progress through the assignment, we will build up the complexity of the dynamics, and in the end compute the complete *Equations of Motion* for the 3 link CrustCrawler robot with a pen attached as an end effector, (some simplifications have been made).

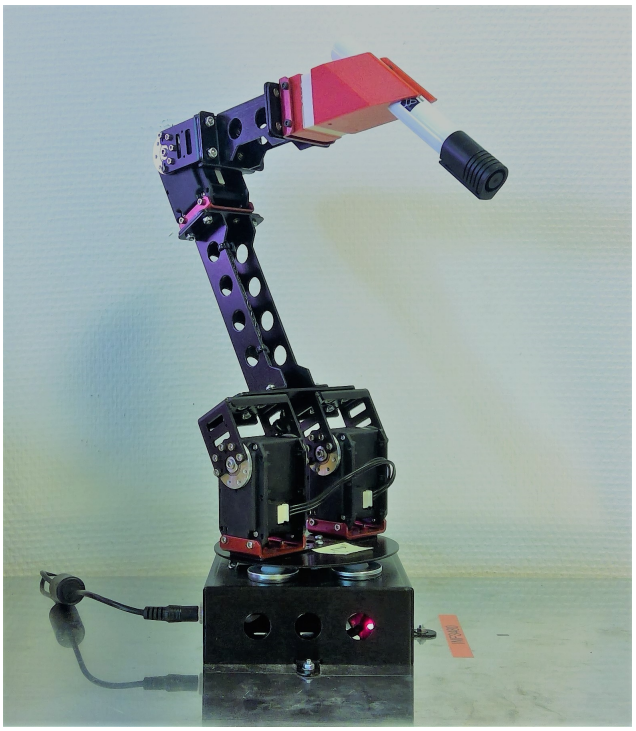


Figure 1: Three link CrustCrawler with Pen

Task 1: Dynamics I (45%)

In this exercise we will develop a simplified dynamical model for a simplified two link version of the real CrustCrawler robot. We will use the ***Euler-Lagrange Equations*** to derive the dynamical equations, also called *Equations of Motion*.

This method derives the ***Lagrangian,*** $\mathcal{L}$, with respect to each joint variable, where

$$\mathcal{L} = \mathcal{K} - \mathcal{P} \tag{1}$$

and assumes that the kinetic and potential energy can be expressed in terms of a set of generalized coordinates.

The general formula for potential energy can be written as

$$\mathcal{P} = mgh \tag{2}$$

where m is the mass, g is the gravitational constant and h is the height above the defined zero plane. For our robot, the base frame is on the zero plane.

The general formula for kinetic energy of a rigid body is given as

$$\mathcal{K} = \frac{1}{2} m \boldsymbol{v}^T \boldsymbol{v} + \frac{1}{2} \boldsymbol{\omega}^T \mathcal{I} \boldsymbol{\omega} \tag{3}$$

where $\mathcal{I}$ is the inertia tensor for the center of mass of the link (expressed relative to the base frame), the velocity vector $\boldsymbol{v}$ (linear velocity) and $\boldsymbol{\omega}$ (angular velocity) are three-dimensional vectors and can be expressed in any reference frame. The inner product of the velocities produces a scalar value. In this task the velocities must be relative to the base frame.

We will model the first motor, Link 1 and the second motor as a cylinder with radius r_1, height L_1 and an evenly distributed mass of m_1, see figure 2. In our case, rotating the cylinder around z_0 gives a moment of inertia about the z axis in the body attached frame (the link), given as

$$I_{1,zz} = \frac{m_1 r_1^2}{2} \tag{4}$$

For simplification, we set the rest of the inertia tensor $\mathbf{I}_1$ and all of the inertia tensor $\mathbf{I}_2$ to zero. We model the mass of link two as a point mass at the end of the link, together with the mass of the third motor. We denote this combined mass m_2.

Finally, when we know $\mathcal{K}$ and $\mathcal{P}$, we can use the ***Euler-Lagrange Equations***, which for a n-DOF system are given as

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_k} - \frac{\partial \mathcal{L}}{\partial q_k} = \tau_k \quad , \quad k = 1, ..., n \quad (7.6) \tag{5}$$

where τ_k is the (generalized) force associated with the vector q_k consisting of the k Denavit-Hartenberg joint variables

$$q_k = q_1, q_2, q_3, ..., q_k \quad , \quad k \in \mathbb{N} \tag{6}$$

which serves as a set of generalized coordinates.

a) 5% Consider a one-dimensional system with mass m where y is the generalized coordinate and a particle is constrained to vertical movement. By Newton's second law, it can be described as:

$$m\ddot{y} = f - mg \tag{7}$$

where the gravitational force mg acts downwards, and an external force f acts upwards.

Use your knowledge about the ***Lagrangian***, to show how the given equation of motion for a particle (equation 7) can be written in terms of $\mathcal{L}$, as equation 5.

b) 20% Derive the ***Lagrangian*** for the two link CrustCrawler. You can find the Jacobian in equation 9 for calculating velocities. Write the ***Lagrangian*** equation on a form consisting only of non-factorized terms, meaning multiplying out all factors.

c) 20% Derive the dynamical model for the two link CrustCrawler using the ***Euler-Lagrange Equations***. The dynamical model for this robot is expressed on the form

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau \quad , \quad \tau \in \mathbb{R}^2 \tag{8}$$

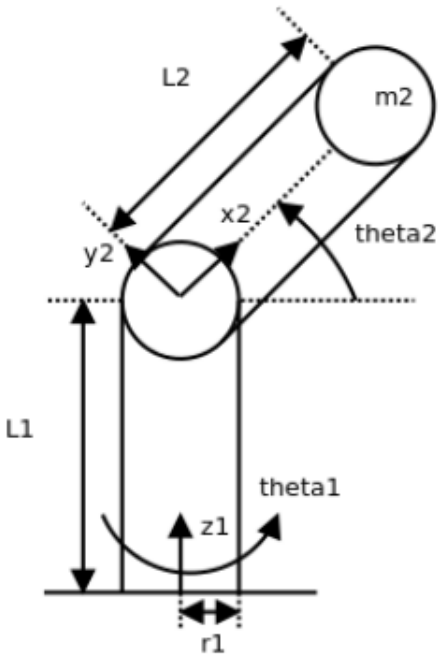


Figure 2: Two-link simplified CrustCrawler

Task 2: Dynamics II (55%)

We will now expand the model to include three links, and investigate a computational approach that is applicable to n links.

The choice of dynamical model greatly affects the accuracy of the results, and it is normally in our best interest to find the most realistic model we can. However, the procedures and calculations involved in such process may be considered to be outside the scope of this course. Therefore, we will settle for a model that is more complex than the one in the previous task, but not overly so.

We will consider each of the three links a rectangular solid with uniform mass and treat any motor attached to a link as a part of that link (in terms of mass). We will also assume that the mass center of each link is in the center of that link.

Figure 3 demonstrates the recommended body-attached frame rotations and positions of the mass centers:

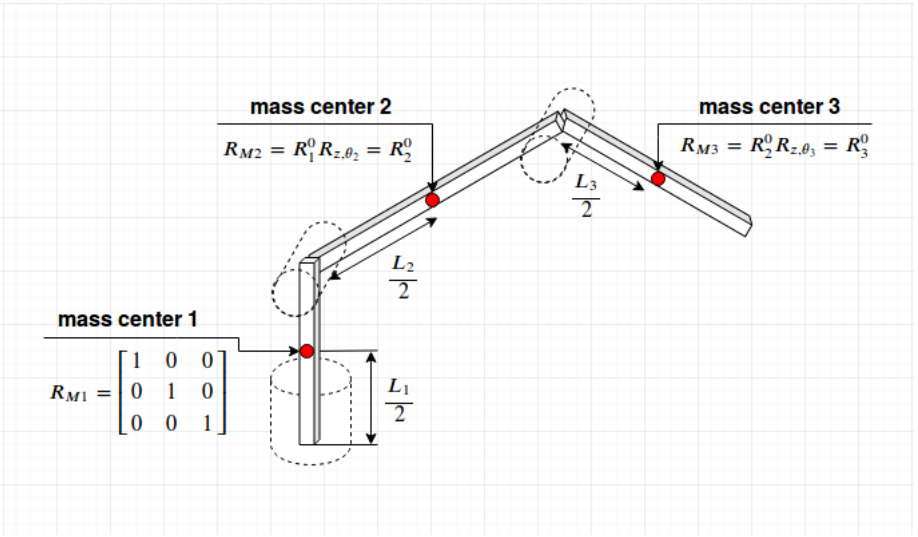


Figure 3: Three-link CrustCrawler dynamical model

You may assume that the link masses are $\mathbf{m_1 = 0.3833\ kg}$, $\mathbf{m_2 = 0.2724\ kg}$ and $\mathbf{m_3 = 0.1406\ kg}$, while the ***Inertia Tensors*** are as follows:

$$\mathbf{I_1} = \begin{bmatrix} I_{1,x} & 0 & 0 \\ 0 & I_{1,y} & 0 \\ 0 & 0 & I_{1,z} \end{bmatrix} \qquad \mathbf{I_2} = \begin{bmatrix} I_{2,x} & 0 & 0 \\ 0 & I_{2,y} & 0 \\ 0 & 0 & I_{2,z} \end{bmatrix} \qquad \mathbf{I_3} = \begin{bmatrix} I_{3,x} & 0 & 0 \\ 0 & I_{3,y} & 0 \\ 0 & 0 & I_{3,z} \end{bmatrix}$$

From assignment 2 we have the following Jacobian for the 3-DOF CrustCrawler:

$$\mathbf{J} = \begin{bmatrix} -s_1(L_2c_2 + L_3c_{23}) & -c_1(L_2s_2 + L_3s_{23}) & -c_1(L_3s_{23}) \\ c_1(L_2c_2 + L_3c_{23}) & -s_1(L_2s_2 + L_3s_{23}) & -s_1(L_3s_{23}) \\ 0 & (L_2c_2 + L_3c_{23}) & L_3c_{23} \\ 0 & s_1 & s_1 \\ 0 & -c_1 & -c_1 \\ 1 & 0 & 0 \end{bmatrix} \tag{9}$$

a) 5% Compute the potential energy for the three-link manipulator by implementing equation (7.52) in the textbook:

$$P = \sum_{i=1}^{n} P_i = \sum_{i=1}^{n} m_i g^T r_{ci}, \tag{10}$$

where r_{ci} is a vector from the origin in the base coordinate system, to the center of mass of link i.

b) 15% Compute the kinetic energy for the 3-link manipulator by implementing equation (7.48) in the textbook:

$$K = \frac{1}{2}\dot{q}^T \left[\sum_{i=1}^{n} m_i J_{vi}(q)^T J_{vi}(q) + J_{\omega i}(q)^T R_i(q) I_i R_i(q)^T J_{\omega i}(q) \right] \dot{q} \tag{11}$$

$$K = \frac{1}{2}\dot{q}^T D(q)\dot{q} \tag{12}$$

R_i is the rotation matrix expressing a rotation from the base coordinate system, to the center of mass of link i.

The intertia tensors I_i, are given above.

c) 10% You have now computed parts of the equations of motion. Show that the complete equation of motion, found in (7.63) in the textbook:

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau \tag{13}$$

can be written as equation (7.62):

$$\sum_{j=i}^{n} dkj(q)\ddot{q}_j + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ijk}(q)\dot{q}_i\dot{q}_j + g_k(q) = \tau_k, k = 1, ..., n \tag{14}$$

and describe the forces represented by each matrix.

HINT *Remember that D and C in equation 13 will always be nxn matrices.*

d) 20% Compute the two remaining terms of equation 14.

HINT *See page 256 in the textbook (2006), and page 258 in Siciliano.*

e) 5% Compute the dynamical model (τ) of the three-link manipulator and submit your code (submitting the output is not necessary).

Appendix

Matrices:

The only python3 package used in the solution is Sympy 1.2. The only function to instantiate data types used is Matrix(). See the Sympy 1.2 tutorial on matrices. In particular matrices are usually instantiated in the following three ways:

- Matrix(column-1, column-2, ..., column-n)
  - Matrix([a,b,c]) will generate a column vector [a,b,c].
  - Matrix([[a,b,c]]) will generate a row vector [a,b,c].
  - Matrix([[a,b,c],[d,e,f]]) a 2by3 matrix.
- Matrix(rows,columns, list of a rows*columns-number of elements) will generate a matrix with the corresponding dimensions and content. range(n) will generate a list [0,1,...,n-1,n].
- zeros(n number of rows, n number of columns), is a way to instantiate a n-by-n matrix zero matrix.

Matrix transpose can be found by using '.T' after the respective matrix. For example, 'matrixA.T'.

Keep track on matrix dimensions to avoid problems with multiplication and addition.

Lists:

A list will equal a row vector, but lists and sympy matrices can't be included in the same mathematical operations.

Lists should be used only as a iterable object, and not for calculations.

Differentiation:

Differentiation can be done by using the diff(equation, variable) function. See the Sympy 1.2 tutorial on Calculus

Differentiation by time on symbols can be done by instantiating dynamicsymbols using the dynamicsymbols package.

- For example, $diff(equation, dynamicsymbols.t)$ or $time_derivative(equation, ReferenceFrame)$
- To find time derivatives, it can be useful to do the following imports:
  - '$from$ $sympy.physics.vector$ $import$ $ReferenceFrame, dynamicsymbols$'
  - '$from$ $sympy.physics.vector$ $import$ $time_derivative$'

Printing:

To print more readable equations, pprint() and simplify() can be used in conjunction. For example, pprint(simplify(equation))

REQUIREMENTS:
Obtain a total score of at least 40%.
Each student must hand in their own assignment, and you are required to have read the following declaration to student submissions at the department of informatics: https://www.uio.no/studier/eksamen/obligatoriske-aktiviteter/mn-ifi-obliger-retningslinjer.html

IMPORTANT! Name the pdf file;

"IN3140(IN4140)_oblig3_*your_username.pdf*"

Submit your assignment at https://devilry.ifi.uio.no.
Your submission must include:

1.) A pdf-document with answers to the questions.

2.) The code asked for in Task 2.

3.) **A README.txt containing a short reflection on the assignment**; what was dificult, what was easy, was there anything you could have done better?

Where you have used MATLAB, Python or other tools to compute an answer, your approach and solution must be illustrated and explained thoroughly in the pdf file. By illustrating, it's meant that you must paste the images of the functions you were asked to implement, and images of the output printed to terminal, ***in the PDF file***. The files containing the code must also be delivered and named;

"in3140(in4140)_oblig3_taskXX_*your_username.py .m*"

You are free too use whatever programming languages and tools you are familiar with, but we strongly recommend doing task 1 by hand and task 2 with Python or Matlab.
In order to get a passed grade you will need to answer satisfactory on 40% of the assignment. We strongly recommend that you do all tasks, as all compulsory assignments build on each other. All tasks are directly relevant to the exam.

Deadline: 8. April 2022, 23:59 (24h)

You can use the mattermost channel *IN3140* for general questions about the assignment and for discussion.

Do not hesitate to contact us if you have any further questions.

Kristian Roa Gran - *krisrgra@ifi.uio.no*
Claudia Andreea Badescu - *claudaba@ifi.uio.no*
Oscar Hæstad Bjørnstad - *oscarhbj@ifi.uio.no*
Thao Le - *thanhtl@ifi.uio.no*