Homework 8: Input/Output Calculations for the Phantom Robot

MEAM 520, University of Pennsylvania
Katherine J. Kuchenbecker, Ph.D.

November 12, 2013

This assignment is due on **Tuesday, November 19, by midnight (11:59:59 p.m.)**. You should aim to turn the paper part in during class that day. If you don't finish before class, you can turn the paper part in to the bin outside Professor Kuchenbecker's office, Towne 224. Your code should be submitted via email according to the instructions at the end of this document. Late submissions will be accepted until Thursday, November 21, by midnight (11:59:59 p.m.), but they will be penalized by 10% for each partial or full day late, up to 20%. After the late deadline, no further assignments may be submitted.

You may talk with other students about this assignment, ask the teaching team questions, use a calculator and other tools, and consult outside sources such as the Internet. To help you actually learn the material, what you write down must be your own work, not copied from any other individual or team. Any submissions suspected of violating Penn's Code of Academic Integrity will be reported to the Office of Student Conduct. If you get stuck, post a question on Piazza or go to office hours!

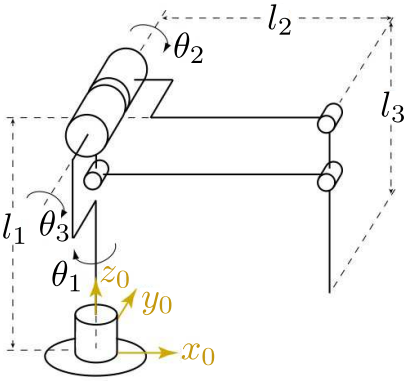
Individual vs. Pair Programming

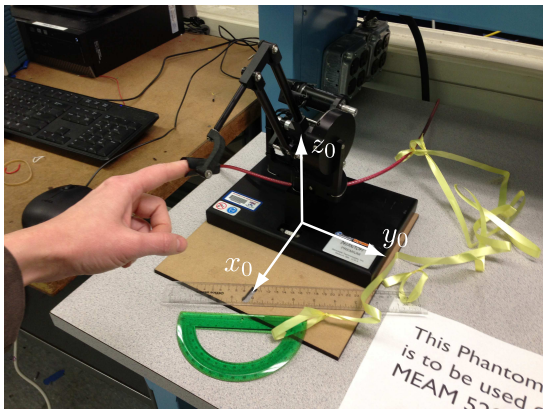
You may do this assignment either individually or with a partner. If you do this homework with a partner, you may work with anyone in the class. If you are in a pair, you should work closely with your partner throughout this assignment, following the paradigm of pair programming. You will turn in one paper assignment and one set of MATLAB files for which you are both jointly responsible, and you will both receive the same grade. Please follow the pair programming guidelines that have been shared before, and name your files with both of your PennKeys separated by an underscore character. Do not split the assignment in half; both of you should understand all steps of this assignment.

SensAble Phantom Premium 1.0

This entire assignment is focused on a particular robot – the SensAble Phantom Premium 1.0. As shown in the photo below, the Phantom is an impedance-type haptic interface with three actuated rotational joints. Designed to be lightweight, stiff, smooth, and easily backdrivable, this type of robotic device enables a human user to interact with a virtual environment or control the movement of a remote robot through the movement of their fingertip while simultaneously feeling force feedback.

A thimble is attached to the tip of the robot via a passive non-encoded three-axis gimbal to allow the user to move the robot around while freely rotating their fingertip. As shown in the diagram below left, the Phantom haptic device looks similar to the standard RRR articulated manipulator base, but it uses a unique four-bar mechanism to co-locate the shoulder and elbow joints while also keeping the upper arm and forearm in the plane that intersects the axis of the waist joint.





As shown in the photo above right, a Phantom is available on a table on the left side of the GM Lab (Towne 193) for you to look at. Many students in this class already have card-swipe access to the GM Lab; the door to this room is often open, or someone inside can let you in. Note that you do not need to connect the Phantom to a computer for any part of this assignment. You should just examine the Phantom to understand how it works and measure any parameters you need. A ruler and a protractor are available at the robot; please leave them attached to the robot's security cable.

Each of the five questions below includes both a written explanation and the programming of a specific MATLAB function. For the paper parts, write in pencil, show your work clearly, box your answers, and staple your pages together. For the programming, download the starter code from the assignment resources section on Piazza, change all function and file names to include your team's PennKeys, comment your code, and follow the instructions at the end of this document to submit all of your MATLAB files.

1. **From Encoder Counts to Joint Angles**

   Imagine your boss bought this robot on eBay and has asked you to get it working. Each of the Phantom's motors includes a shaft-mounted HEDM-5500-B02 optical encoder. The data sheet for this family of encoders is available as `HEDM-5500.pdf` inside the zip file `phantom_files.zip` under assignment resources on Piazza. Drum-and-capstan cable drives are used to connect the motors to the respective joints.

   After figuring out the wiring, imagine you connected all three of the Phantom's incremental optical encoders to a quadrature encoder input card on your computer. You zeroed all of the encoders when the device was in the configuration shown in the schematic above, where the upper arm is horizontal, the lower arm is vertical, and the tip is located above the x_0-axis. This is the Phantom's zero configuration. Our Phantom sits on a board that includes a cut-out into which you can place the gimbal for more repeatable zeroing.

   As you played with the Phantom, you noticed that the encoder counts increase when each joint is rotated in the direction indicated on the schematic. You decide you will use these positive directions for your joint angles as well. You then moved the robot through an interesting trajectory (writing the word "phantom" in cursive in the horizontal plane; the text is written along a straight line that is parallel to the y_0-axis) and recorded the triplets of encoder count values $[Q_1 \;\; Q_2 \;\; Q_3]^T$ that occurred at 206 points along the way. This list of values is provided with the starter code as `phantom_encoder_counts.txt` and is automatically loaded into the `phantom_robot_yourpennkeys.m` script.

(a) Based on your inspection of the Phantom and the above information, write a careful explanation of how to convert its encoder counts, $[Q_1 \quad Q_2 \quad Q_3]^T$, to joint angles in radians, $[\theta_1 \quad \theta_2 \quad \theta_3]^T$, including step-wise calculations and any helpful diagrams.

(b) Implement this function in MATLAB as `phantom_counts_to_angles_yourpennkeys.m` by modifying the provided starter code.

2. **From Joint Angles to Robot Position**

Now it is time to express the position of the Phantom's key components in terms of its joint angles. Your boss specifically told you to use the coordinate frame indicated on the schematic above (even though it does not comply with DH conventions). Since the Phantom is not a serial manipulator, you realize that DH parameters are not a good method for analyzing its kinematics anyway; simple geometry is a better choice.

(a) Based on your inspection of the Phantom and the above information, write a precise explanation of how to convert joint angles in radians to the position of all points needed to create a simple stick-figure animation of the robot. At a minimum, your animation should include the origin of frame 0, two other useful points along the robot, and the tip of the robot. Express all positions in frame 0 using millimeters, include the physical values of any constants, and include any diagrams you used to figure this out.

(b) Implement this function in MATLAB as `phantom_angles_to_positions_yourpennkeys.m` by modifying the provided starter code. Use the graphing provided in the `phantom_robot_yourpennkeys.m` script to check your calculations, and fix any errors you find in this step or the previous one. For example, you can check your work by making sure the joint angles stay within the robot's joint limits and ensuring that the span of the robot's motion is physically realizable. Furthermore, the trajectory of the tip should look like the motion described above.

3. **From Force Vector to Joint Torques**

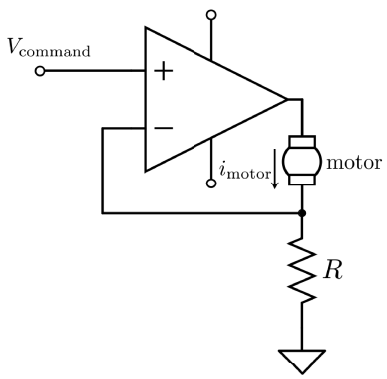
Now imagine you want to output a force vector at the tip of the robot for the user to feel. The force vector will be specified using newtons in frame 0 as $[F_x \quad F_y \quad F_z]^T$. You remember that a Jacobian Transpose controller is good for situations like this. The robot's joint torques follow the same positive direction convention as the angles; a positive torque will make each joint move in the positive direction.

(a) Write a precise explanation of how to convert a desired three-dimensional force vector to the triplet of joint torques $[\tau_1 \quad \tau_2 \quad \tau_3]^T$ in newton·meters (not newton·millimeters) needed to produce that force vector at a given robot configuration $[\theta_1 \quad \theta_2 \quad \theta_3]^T$. Include step-by-step calculations.

(b) Implement this function in MATLAB as `phantom_force_to_torques_yourpennkeys.m` by modifying the provided starter code. You may set the desired force to any vector you want on lines 51 to 53 of the starter code. It should probably stay constant across this simulation to make it easier to debug. A scaled version of this force vector is graphed in gray to help you visualize the torques needed to create it. The graphing provided in `phantom_robot_yourpennkeys.m` should help you debug this function; you are welcome to update the graphing however you want, but please add comments to explain what you did. If you were actually using the Phantom for haptic rendering, you would calculate a force vector at each time step, based on the current tip position, the history of tip positions, and the geometry and dynamics of the virtual environment.

4. **From Joint Torques to DAC Voltages**

You now need to figure out how to create the torques needed on each joint of the robot. All three joints are driven by Maxon 118743 DC brushed motors. The data sheet for this family of motors is available as `Maxon-118743.pdf` inside the zip file `phantom_files.zip` under assignment resources on Piazza. Recall that drum-and-capstan cable drives are used to connect the motors to the respective joints.



You wire up each motor to a custom-built linear current amplifier. Each of these amplifiers uses one OPA544 high-voltage high-current operational amplifier (op-amp), configured in the circuit shown at left. The motor is connected as drawn, and you specify the desired motor current by using a DAC card on your computer to apply a command voltage V_{command} to the op-amp's non-inverting input, relative to ground. The op-amp supply voltages are symmetric and within the ranges specified on the OPA544 data sheet. The resistor is a high-precision power resistor with a resistance of $R = 5.00\ \Omega$. When analyzing this circuit to determine the relationship between command voltage and motor current, you should assume that the op-amp behaves ideally.

Your three linear current amplifiers cause three respective current to flow through the three motors of the Phantom, $[i_1 \;\; i_2 \;\; i_3]^T$. The current amplifiers are numbered the same as the joints, and you set it up so that positive current makes each joint want to move in its positive direction. You should never command more than the maximum steady-state current the motor can take.

(a) Write a precise explanation of how to convert a triplet of desired joint torques $[\tau_1 \;\; \tau_2 \;\; \tau_3]^T$ into the corresponding triplet of voltage commands $[V_1 \;\; V_2 \;\; V_3]^T$, making sure never to command more current than the motor can take in steady state. Include step-by-step calculations and the values of any physical constants needed.

(b) Implement this function in MATLAB as `phantom_torques_to_voltages_yourpennkeys.m`, and use the graphing provided in `phantom_robot_yourpennkeys.m` to debug this function.

5. **Motor Dynamics**

The final step in this homework is to complete a MATLAB simulation of one of the Phantom's brushed DC motors. As explained above, these motors are model 118743 from Maxon DC Motor Corporation, and the data sheet is provided in the zip file of resources for this homework.

The scenario that you should simulate is the **motor alone**, removed from the Phantom, with nothing attached to its output shaft and no encoder or other sensor on its sensor shaft. It should be driven using **voltage drive**, as opposed to the current drive explained above, so that we can understand its native electro-mechanical dynamics. In particular, the simulation has been set up to drive the motor with a 50% duty-cycle pulse waveform that alternates between 0 V and the motor's nominal voltage of 12 V with a period of 0.5 s.

As discussed in lecture and in SHV 6.1 and 6.2, a brushed DC motor's movement is governed by a set of coupled differential equations that relate the first and second derivatives of the motor shaft angle with the applied voltage, the instantaneous current through the armature, the first derivative of the armature current, the load torque, and all of the motor's parameters. The provided script `simulate_motor_starter.m` sets up the simulation to track the following three state variables:

- θ_m = `thetam` The angle of the motor shaft, in radians.
- ω_m = `omegam` The angular velocity of the motor shaft, in radians per second.
- i_a = `ia` The current in the motor's armature, in amps.

Change the names of both `simulate_motor_starter.m` and `compute_motor_derivatives_starter.m` to include your PennKeys separated by an underscore character, or just your PennKey if you're working alone. In the first file, change the call to `compute_motor_derivatives_starter.m` on line 63 to match your function name. Also change the function declaration on the first line of the second file.

Your job is to update the provided function `compute_motor_derivatives_starter.m` so that it correctly represents the motor's dynamics. The inputs to this function are the present time `t` in seconds

and the present states of the motor, assembled into the column vector `X`. The function is already set up to pull the instantaneous values of the motor's three states out of this vector so that you can use them by their names (`thetam`, `omegam`, and `ia`). The script also compares the present time to the global variable that sets the period of the voltage pulse to calculate the present value of `V`, the voltage to be applied to the motor.

You must use your knowledge of motor dynamics to compute the **instantaneous time derivatives** of all three of the motor's states; store these values in `thetamdot`, `omegamdot`, and `iadot`. You will need to consult the motor's data sheet to obtain the values of its parameters. Be sure to convert everything to compatible units (V, A, H, Ω, Nm, kg·m^2, rad/s, etc.) Note that the value of the torque constant has already been specified in the function as $k_t = 0.023467\,\text{Nm/A}$ – this value is more accurate than what is specified on the data sheet, so please use it. One parameter that you need (B_m, the motor's frictional damping coefficient in Nm/(rad/s)) is not specified on the data sheet; you must find the value (within two significant digits) that makes your simulation's behavior match the data sheet.

When the variable `testtype` is equal to zero, your simulation should perform a no-load test, where the load torque $\tau_l =$ `taul` $= 0$. When the variable `testtype` is equal to one, your simulation should perform a stall test, where the load torque is a rotational spring and damper to simulate you holding the shaft almost stationary with your fingers: $\tau_{l,\text{stall}} = -(2\text{ Nm/rad})\,\theta_m - (0.0005\text{ Nm/(rad/s)})\,\omega_m$. Compare the output of your simulation during these two tests to the motor's data sheet to check the accuracy of your simulation and find the correct value for B_m. Remember that you should **not** do any integration inside your compute derivatives function; `ode45` does all of the integration for you.

(a) Write out all of the equations that govern the motion of the motor, plus a list of the motor's parameter values.

(b) Implement these dynamics in MATLAB as `compute_motor_derivatives_yourpennkeys.m`, and use the graphing provided in `simulate_motor_yourpennkeys.m` to debug this function.

Submitting Your Code

Submit the paper part of this assignment as specified on the first page. Follow these instructions to submit your code:

1. Start an email to `meam520@seas.upenn.edu`
2. Make the subject *Homework 8: Your Name* or *Homework 8: Your Name and Your Teammate's Name*, replacing *Your Name* and *Your Teammate's Name* with the appropriate full names.
3. Attach your seven correctly named MATLAB files to the email as individual attachments; please do not zip them together or include any other attachments. The files should be the following:
   - `phantom_robot_yourpennkeys.m`
   - `phantom_counts_to_angles_yourpennkeys.m`
   - `phantom_angles_to_positions_yourpennkeys.m`
   - `phantom_force_to_torques_yourpennkeys.m`
   - `phantom_torques_to_voltages_yourpennkeys.m`
   - `simulate_motor_yourpennkeys.m`
   - `compute_motor_derivatives_yourpennkeys.m`
4. Optionally include any comments you have about this assignment.
5. Send the email.

You are welcome to resubmit your code if you want to make corrections. To avoid confusion, please state in the new email that it is a resubmission, and include all of your MATLAB files, even if you have updated only some of them.