Exercises for CASPR

This documentation serves provides some exercises for new users of CASPR to try in order to verify their understanding of CASPR functionality. In solving these exercises please make use of the existing CASPR examples in addition to the documents CASPR_101.pdf and CASPR_Analysis.pdf.

1 Exercise 1 - Writing a New script

Write a new CASPR script that meets the following objectives:

1. Load the model `IPAnema2` with the cableset `original` .
2. Create a `IDSolverMinInfNorm` solver with the objective `IDObjectiveInfOptimallySafe`.
3. Create and Run an inverse dynamics simulator that uses the trajectory `traj_z_up`.
4. Plot the results of the simulator. This should include the following plots
   - A plot of the joint space variables.
   - A plot of the cable forces used throughout the trajectory.
   - A plot of the cable lengths.

Note: This new script should be placed into the folder ~/scripts/local/. All scripts within that folder will be ignored by git and will not be committed into the repository.

2 Exercise 2 - Adding a New Model

2.1 Single link CDPR

Add a new robot model `CASPR_test_SCDM` into CASPR. The robot should meet the following specifications:

- The robot possesses a single link with a spatial joint.
- The link possesses a mass of 10kg and can have its inertia arbitrarily set to a reasonable value. Assume that the body is a cube with 0.1m side lengths and has its centre of mass at the origin of the links.
- The robot should possess 8 ideal cables. The attachments for the cables are given in the Table 1 where all cables use the `com` as the reference point for the link.
- The desired trajectory for the robot is to traverse from a resting position of $(x, y, z, \alpha, \beta, \gamma) = (-1, -1, -1, 0, 0, 0)$ to a resting position of $(x, y, z, \alpha, \beta, \gamma) = (1, 1, 1, 0, 0, 0)$ where α, β and γ are euler angles. Assume that this trajectory is to be run for 1 seconds at a sampling time of 0.01 seconds.
- In this case there will be no operational space under consideration.

Once you have added the new model please modify the model configuration files and use the unit tests to confirm that your model has been successfully added. If this is the case please remove the new model since this should not be committed into CASPR.

2.2 Multilink CDPR (optional)

Add a new robot model `CASPR_test_MCDM` into CASPR. The robot should meet the following specifications:

- Comprised of two links that are connected together: link 1 has a spherical joint (with euler xyz coordinate representation) that connects it to the base and link 2 is connected to link 1 through revolute joint which allows rotation about the z axis.

Cable	Link A	Attachment Location A	Link B	Attachment Location B
1	0	$\begin{bmatrix}5.0 & 5.0 & 5.0\end{bmatrix}$	1	$\begin{bmatrix}0.1 & 0.1 & 0.1\end{bmatrix}$
2	0	$\begin{bmatrix}5.0 & 0.0 & 5.0\end{bmatrix}$	1	$\begin{bmatrix}0.1 & 0.0 & 0.1\end{bmatrix}$
3	0	$\begin{bmatrix}0.0 & 0.0 & 5.0\end{bmatrix}$	1	$\begin{bmatrix}0.0 & 0.0 & 0.1\end{bmatrix}$
4	0	$\begin{bmatrix}0.0 & 5.0 & 5.0\end{bmatrix}$	1	$\begin{bmatrix}0.0 & 0.1 & 0.1\end{bmatrix}$
5	0	$\begin{bmatrix}5.0 & 5.0 & 0.0\end{bmatrix}$	1	$\begin{bmatrix}0.1 & 0.1 & 0.0\end{bmatrix}$
6	0	$\begin{bmatrix}5.0 & 0.0 & 0.0\end{bmatrix}$	1	$\begin{bmatrix}0.1 & 0.0 & 0.0\end{bmatrix}$
7	0	$\begin{bmatrix}0.0 & 0.0 & 0.0\end{bmatrix}$	1	$\begin{bmatrix}0.0 & 0.0 & 0.0\end{bmatrix}$
8	0	$\begin{bmatrix}0.0 & 5.0 & 0.0\end{bmatrix}$	1	$\begin{bmatrix}0.0 & 0.1 & 0.0\end{bmatrix}$

Table 1: Attachment Locations for SCDM

- Each link should possess a mass of 1kg and can have its inertia arbitrarily set to a reasonable value. Assume that the links are $1m$ long (in the z direction) and have their COM exactly half way along the z axes.
- The robot should possess 6 linear spring cables (with stiffness 1000 N/m). The attachments for the cables are given in the Table 2 where all cables use the `com` as the reference point for the link.
- The desired trajectory for the robot is to traverse from a resting position of $(\alpha, \beta, \gamma, \theta) = (0, 0, 0, 0)$ to a resting position of $(\alpha, \beta, \gamma, \theta) = (0, 0, 0, 0)$ where α, β and γ are the euler angles and θ represents the z rotation. Assume that this trajectory is to be run for 5 seconds at a sampling time of 0.1 seconds.
- In this case there will be no operational space under consideration.

Cable	Link A	Attachment Location A	Link B	Attachment Location B
1	0	$\begin{bmatrix}0.5 & 0.0 & 0.0\end{bmatrix}$	1	$\begin{bmatrix}0.05 & 0.0 & 0.3\end{bmatrix}$
2	0	$\begin{bmatrix}-0.5 & 0.0 & 0.0\end{bmatrix}$	1	$\begin{bmatrix}-0.05 & 0.0 & 0.3\end{bmatrix}$
3	0	$\begin{bmatrix}0.0 & 0.5 & 0.0\end{bmatrix}$	1	$\begin{bmatrix}0.0 & 0.05 & 0.3\end{bmatrix}$
4	0	$\begin{bmatrix}0.0 & -0.5 & 0.0\end{bmatrix}$	1	$\begin{bmatrix}0.0 & -0.05 & 0.3\end{bmatrix}$
5	0	$\begin{bmatrix}0.8 & 0.0 & 0.0\end{bmatrix}$	2	$\begin{bmatrix}0.03 & 0.0 & 0.3\end{bmatrix}$
6	1	$\begin{bmatrix}-0.05 & 0.0 & 0.1\end{bmatrix}$	2	$\begin{bmatrix}-0.03 & 0.0 & 0.3\end{bmatrix}$

Table 2: Attachment Locations for MCDM

Once you have added the new model please modify the model configuration files and use the unit tests to confirm that your model has been successfully added. If this is the case please remove the new model since this should not be committed into CASPR.

3 Exercise 3 - Writing a New Analysis Tool

Create a new forward kinematics solver. This forward kinematics incorporate the following:

- The forward kinematics solver should implement the class FKAnalysisBase.
- The solver should compute a new joint pose $\mathbf{q}$ using the following algorithm

$$\mathbf{q} = \mathbf{q}_{prev} + \dot{\mathbf{q}}\Delta t, \tag{1}$$

where $\mathbf{q}_{prev}$ is the known previous joint position and $\dot{\mathbf{q}}$ is computed using the expression

$$\dot{\mathbf{q}} = L^{\dagger}\frac{(\mathbf{l} - \mathbf{l}_{prev})}{2\Delta t}, \tag{2}$$

where $L^{\dagger}$ can be produced in matlab using the operation $inv(L^T L)L^T$ and $\frac{(\mathbf{l}-\mathbf{l}_{prev})}{2\Delta t}$ is an approximation of $\dot{\mathbf{l}}$.

Note: This solver corresponds to the `FKDifferential` solver. You can verify your solution against this solution as you look to write the code.