Robotic Arm Impedance Control

With an implementation for a Kinova® Gen3 Robotic Arm

Open the project in MATLAB Online

Contents

What is Impedance Control?

Impedance control is a technique used in robotics to regulate the interaction forces between a robotic arm and its environment. Instead of directly controlling the position or velocity of the robotic arm's end-effector (e.g., gripper), impedance control focuses on controlling the stiffness, damping, and inertia of the arm's motion, mimicking the behavior of a mechanical spring-damper system.

Here's a breakdown of the key components of impedance control:

1. **Stiffness**: Stiffness refers to how resistant the robotic arm is to deformation when a force is applied to it. In impedance control, stiffness is typically represented as a virtual spring. Higher stiffness values result in less deformation in response to external forces.
2. **Damping**: Damping controls how quickly the robotic arm dissipates energy when subjected to external forces. It's akin to the resistance provided by a damper in a mechanical system. Higher damping values help absorb energy from external disturbances more quickly.

By adjusting these parameters, impedance control allows the robotic arm to respond appropriately to different environmental conditions and interaction forces. For example:

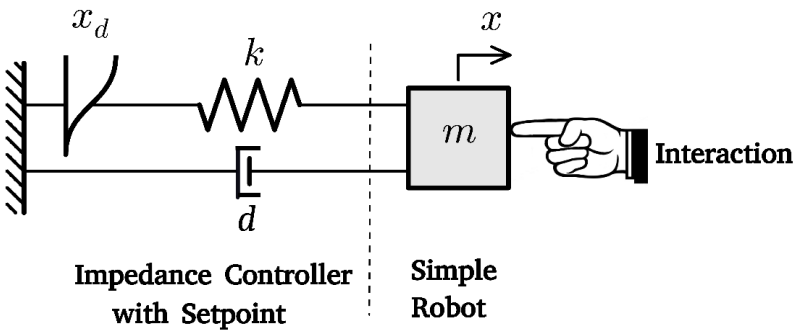
- **Contact with objects**: When the robotic arm comes into contact with an object, impedance control adjusts the stiffness and damping to provide the desired level of compliance. This allows the arm to maintain stable contact with the object while adapting to variations in surface geometry or external forces.
- **Safety and human-robot interaction**: Impedance control is particularly useful in applications where robots work alongside humans. By adjusting the impedance parameters, the robot can ensure safe interaction with humans by reducing the risk of injury in case of accidental contact.

- **Task execution**: In tasks such as assembly or manipulation, impedance control enables the robotic arm to precisely control the forces applied during interaction with objects. This helps improve accuracy and reliability in tasks where delicate handling or precise force control is required.

Overall, impedance control enhances the versatility and adaptability of robotic arms, enabling them to operate effectively in dynamic and uncertain environments while ensuring safe and efficient interaction with objects and humans.

1-DOF Robot Example

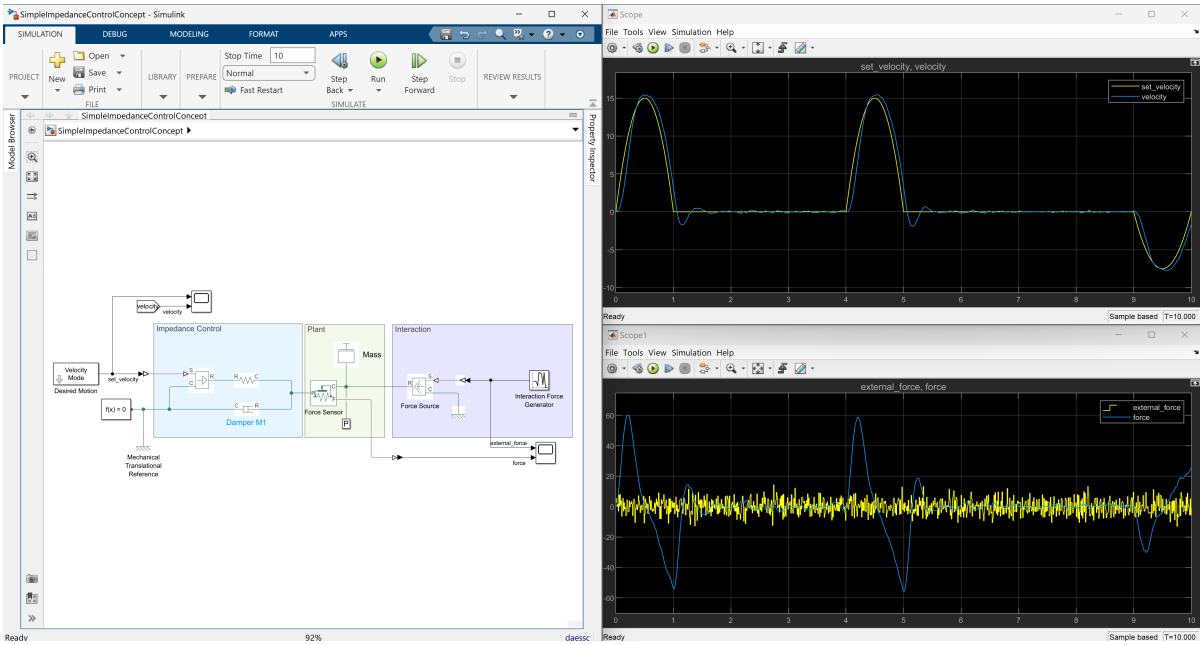
Consider a simplified 1-DOF robot modeled as a mass m at position X_m and it is to be moved to a desired position X_d, a simple physical controller to achieve this consists of a spring connected between the desired virtual point and the mass. The resulting mass-spring system is marginally stable and will continue to oscillate unless a damper is added to the system. In addition to placing the object at the desired position in a stable way, the value of K_c can be adjusted to give the system a desired compliance which will be felt by any interaction with the system. See Figure below.



Use the model below to explore a Simscape based implementation of this conceptual impedance controller.

```
open_system('SimpleImpedanceControlConcept');
```

Below figure shows the model and simulation results:



Check that

- expected impedance is felt during interaction
- increasing spring constant K_c results in a lower motion error
- decreasing damping coefficient D_c results in a higher vibration

Mathematically, the impedance controller can be defined as,

$$F_c = K_c \cdot (X_d - X_m) - D_c \cdot \dot{X}_m$$

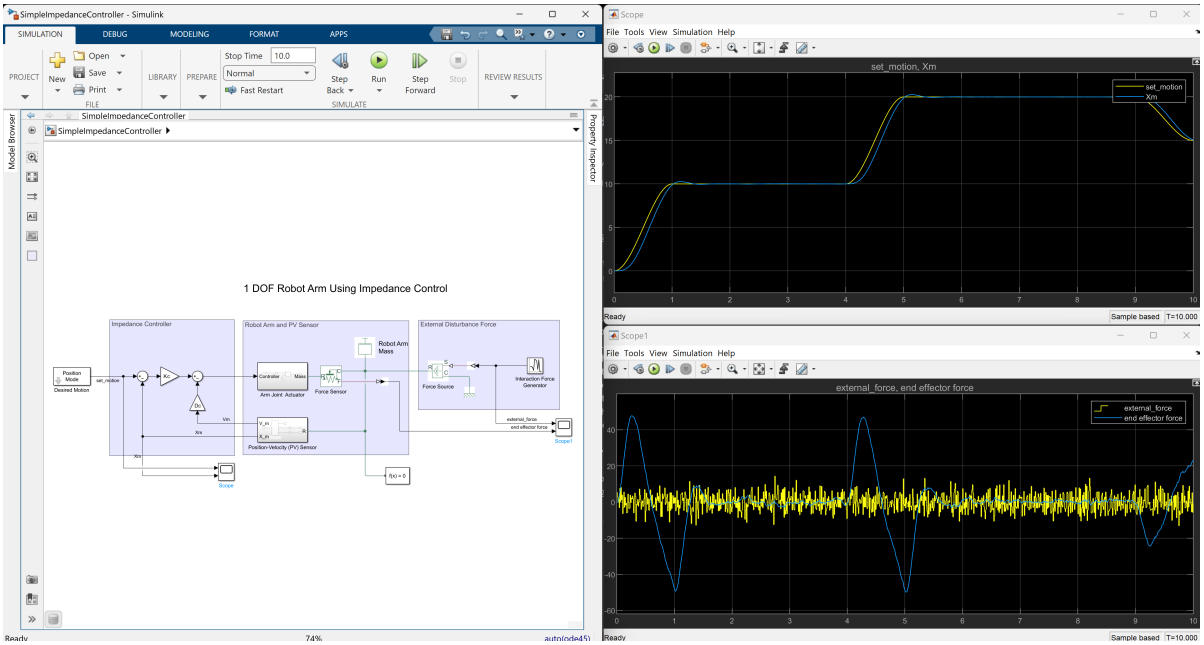
It resembles a conventional PD controller that guarantees closed loop stability for any positive gains, with the proportional term representing the spring and the derivative term representing the damper.

Let's open a different model that is configured as a typical controller, plant model, and sensor.

```
open_system('SimpleImpedanceController');
```

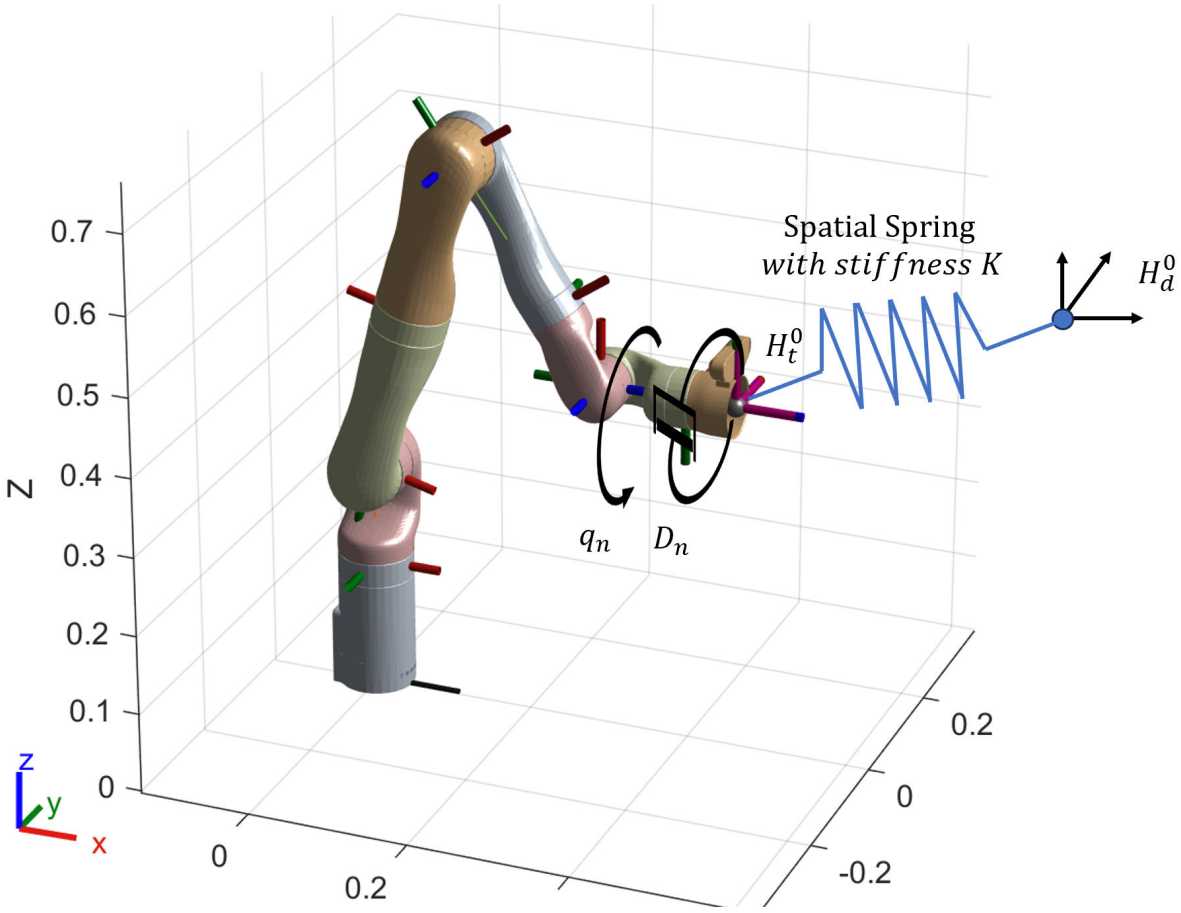
Modify the controller parameters (they are defined in the model workspace) and see if the system behaves as expected.

Note that, even in the presence of a large random disturbance (large in comparison to the arm mass), the robot motion (position and velocity) is smooth enough.



Multi-DOF Robot

For an n-DOF robotic manipulator, the cartesian impedance controller is physically described as a multidimensional virtual spring that attempts to co-align the robot end-effector pose $H_t^0(t)$ to a desired pose configurations $H_d^0(t)$. In order to achieve a satisfactory dynamic behavior and guarantee a robust asymptotic stability, damping is also injected into the system via a joint space damper D_n on each DOF. For a Kinova® Gen3 robot, $n = 1..7$.



This virtual spatial geometric spring is defined as a symmetric matrix $K \in \mathbf{R}^{6\times6}$

$$K = \begin{pmatrix} K_o & K_c \\ K_c^T & Kt \end{pmatrix}$$

where K_t, K_o and K_c are the symmetric translational, rotational, and coupling stiffness matrices.

The generalized force exerted by the spring on the robot is a function of the relative configuration $H_t^d(t)$.

The force can be defined in the coordinates of the end-effector as a 6 vector wrench $W^e = [m^e \ f^e]$ computed as follows [Ref]

$$\widetilde{m}^t = -2as(G_o R_t^d) - as(G_t R_d^t \widetilde{p}_t^d \widetilde{p}_t^d R_t^d) - 2as(G_c \widetilde{p}_t^d R_t^d)$$

$$\widetilde{f}^t = -R_d^t as(G_t \widetilde{p}_t^d) R_t^d - as(G_t R_d^t \widetilde{p}_t^d R_t^d) - 2as(G_c R_t^d)$$

where

- $as(\cdot)$ is an operator that gives the skew-symmetric part of a square matrix
- G_t, G_o and G_c are co-stifness matrices of the spatial spring calculated as $G_x = \frac{1}{2} tr(K_x) I - K_x$
- $tr()$ is the tensor trace operator

- $\sim$ is a tilde operator capturing cross product matrix form of a vector

After coordinate transformation of the wrench W^t to an inertial reference frame Ψ_0, joint torques τ that can emulate the desired end-effector wrench on the manipulator are calculated by the equations:

$$(W^0)^T = Ad^T_{H^t_0}(W^t)^T$$

$$\tau = J^T(q)W^0$$

where $Ad(\cdot)$ is the adjoint of the homogeneous matrix

The Cartesian Impedance Controller in the model is enhanced by a gravity compensation to improve the operational performance of the manipulator.

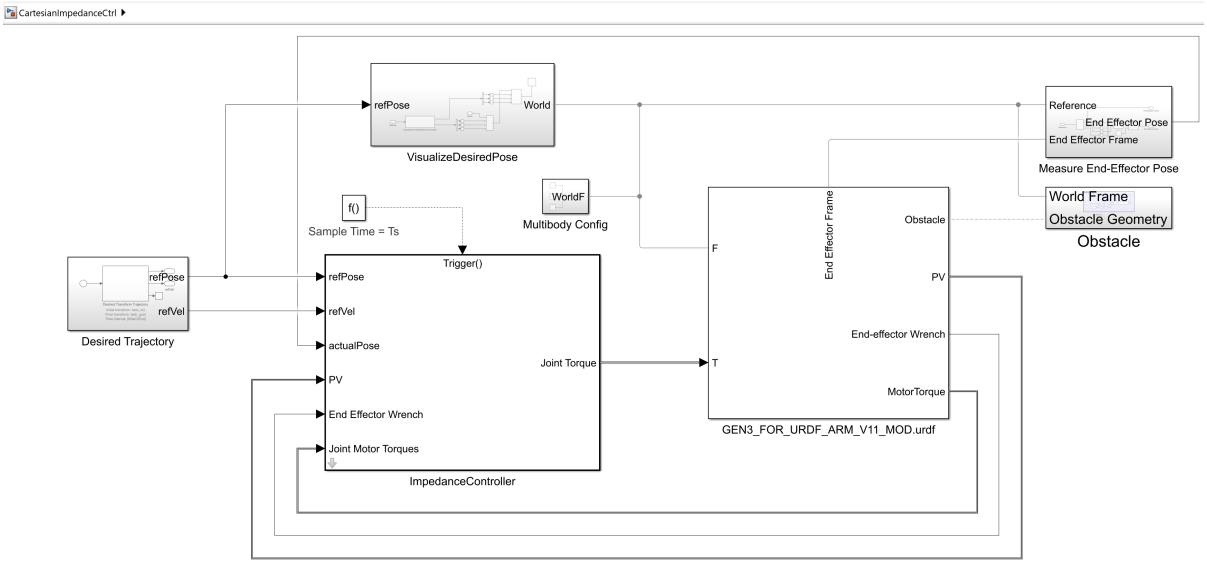
Thus, the final controller output is a joint-space torque vector which is the sum of torques from:

- Spatial spring
- Gravity compensation
- Joint Damping injection

Model Description

In this example we show how to design a cartesian impedance control for a Kinova® Gen3 robotic arm. A multidimensional virtual spring is connected between the actual and desired end-effector's configuration, and, additionally, damping is added at each joint.

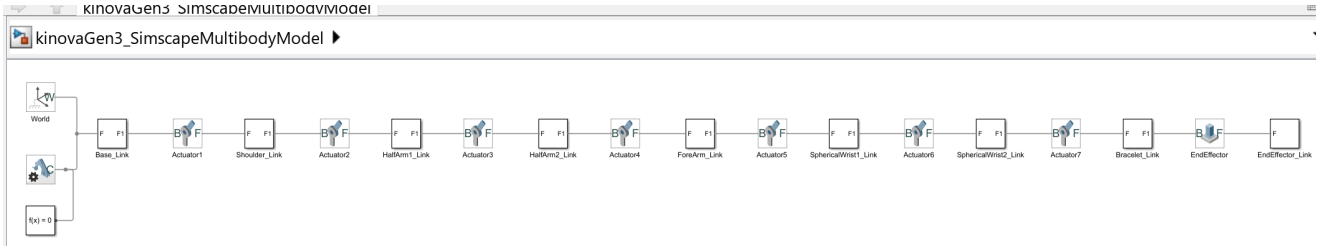
The simulation model is shown below. It consists of a trajectory generator, an impedance controller block, the robot arm, an obstacle, and a contact force transducer model. Note that the contact force is not used in the controller, but can be used to develop more elaborate safe control algorithms in the future, if the End Effector is fitted with a force or pressure transducer.



The robot block, shown in the figure below, contains geometry modelled using Simscape™ Multibody™, a simple obstacle (a vertical wall), and a model of a contact force sensor (a force transducer model).

The robot geometry model can be created by using `smimport` command. This will automatically generate a Simulink™ model that contains Simscape™ Multibody™ components, for instance bodies and joints.

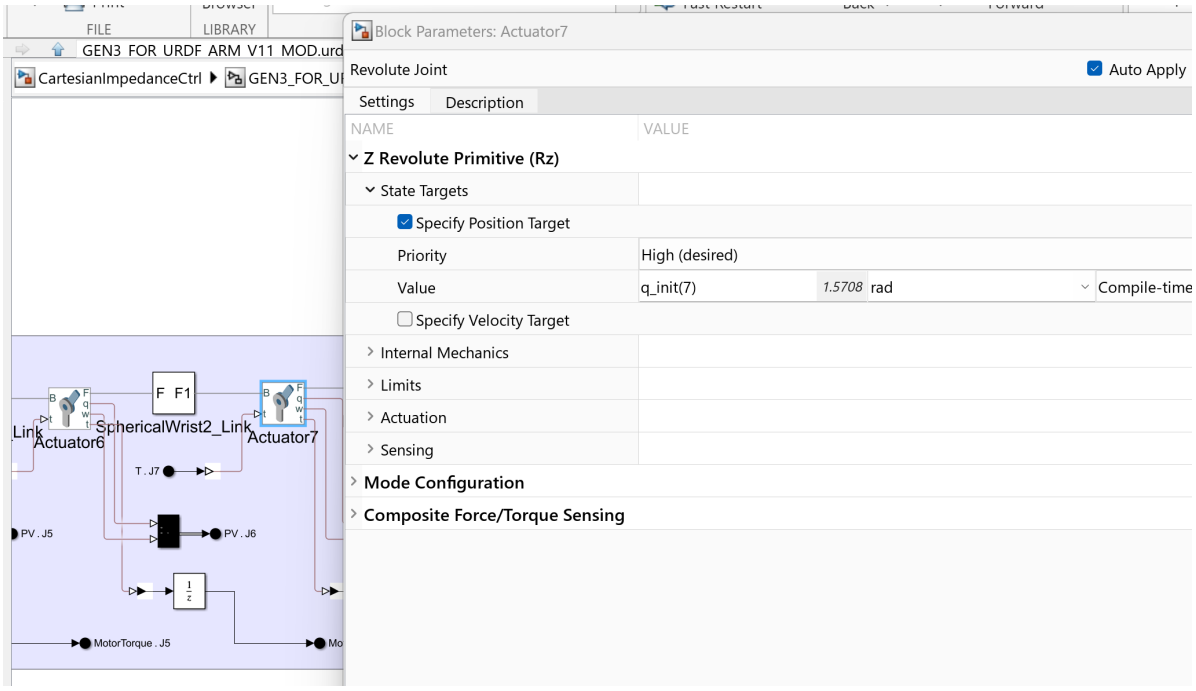
```
% Gen 3 robotic arm manipulator
robotSM = smimport(kinovaGen3,ModelName="kinovaGen3_SimscapeMultibodyModel");
```



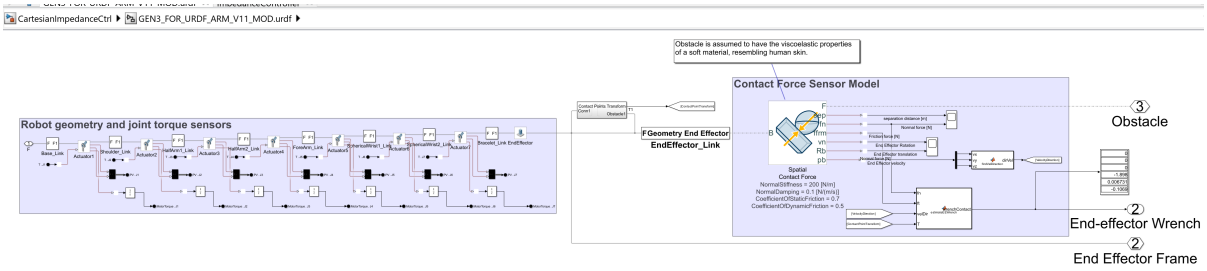
Next, configure the Simscape™ Multibody™ model. Prepare the model to accept the joint torque inputs and return the joint positions and velocities. For more details, consult the documentation example Design Position Controlled Manipulator Using Simscape on how to automatically instrument the robot model with postion, velocity, or torque sensors. You can use the `helperInstrumentSMModels` helper function (found in the exmaple Design Position Controlled Manipulator Using Simscape) to automatically configure the model.

Finally, assign the initial configuration, `q0`, to each joint. This parameter is used in the Multibody model joint blocks, to initialize the robot pose.

It will de defined numerically later in this program.

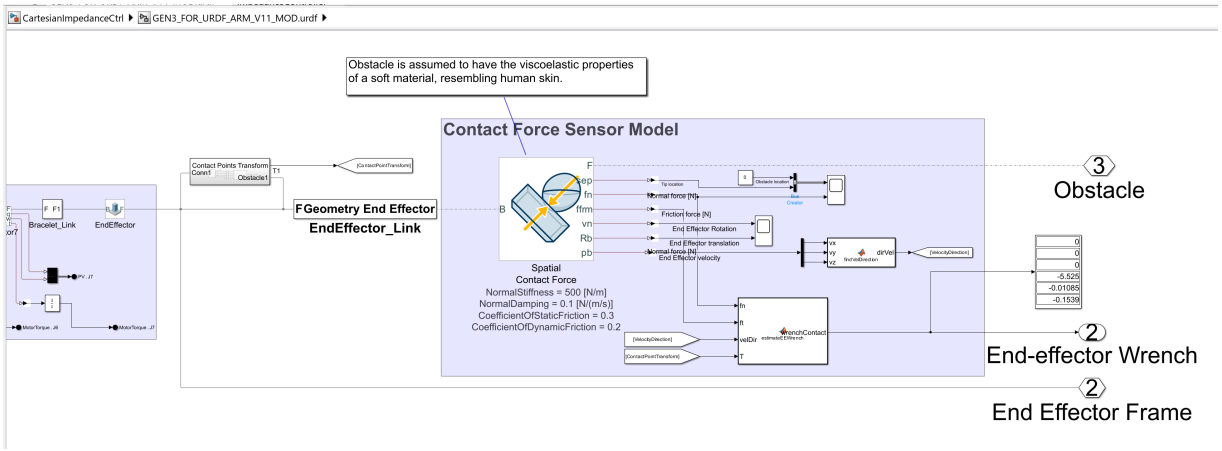


This Simscape™ Multibody™ model can then be copied to the impedance control model, as below.



Next, we add a contact force model, that senses the forces at the tip.

We assume that the obstacle has the viscoelastic properties of a soft material, resembling human skin, with a normal stiffness of 500 N/m and a damping of 0.1 N/(m/s).



Next, we leverage Robotics Toolbox to design the impedance controller.

Kinova® Gen3 Robot Import

Reference: https://www.mathworks.com/help/robotics/robotmanipulator/ug/get-latest-urdf.html

```
% Robot's URDF file
% Note: Any different URDF version can be imported in MATLAB using importrobot
function.
robotFile = "GEN3_FOR_URDF_ARM_V11_MOD.urdf" ;

kinovaGen3 = importrobot(robotFile,DataFormat="column");
kinovaGen3.Gravity=[0, 0, -9.8067];
```

If necessary, use below command to visualize the robot

```
show(kinovaGen3,'visuals','on','collision','on');
```

```
% Show link names and bodies
kinovaGen3.BodyNames'
```

```
ans = 8×1 cell
'Shoulder_Link'
'HalfArm1_Link'
'HalfArm2_Link'
'ForeArm_Link'
```

```
'SphericalWrist1_Link'
'SphericalWrist2_Link'
'Bracelet_Link'
'EndEffector_Link'
```

```
showdetails(kinovaGen3)
```

```
--------------------
Robot: (8 bodies)

 Idx                 Body Name          Joint Name         Joint Type                  Parent Name(Idx)   Children
 ---                 ---------          ----------         ----------                  ----------------   --------
   1              Shoulder_Link           Actuator1           revolute                      Base_Link(0)   HalfArm1
   2              HalfArm1_Link           Actuator2           revolute                  Shoulder_Link(1)   HalfArm2
   3              HalfArm2_Link           Actuator3           revolute                  HalfArm1_Link(2)   ForeArm_
   4               ForeArm_Link           Actuator4           revolute                  HalfArm2_Link(3)   Spherica
   5       SphericalWrist1_Link           Actuator5           revolute                   ForeArm_Link(4)   Spherica
   6       SphericalWrist2_Link           Actuator6           revolute           SphericalWrist1_Link(5)   Bracelet
   7              Bracelet_Link           Actuator7           revolute           SphericalWrist2_Link(6)   EndEffec
   8           EndEffector_Link         EndEffector              fixed                  Bracelet_Link(7)
--------------------
```

```
% Set robot home position
q_home = [0 15 180 -130 0 55 90]'*pi/180;
```

If necessary, use below command to visualize the robot:

```
show(kinovaGen3, q_home);
axis auto;
```

```
% There must be some damping in the actual manipulator, but typically this is not a
known quantity.
% Here we assume the following value for joint damping
% (this is a very small value but it can help stabilize the robot model during
simulation).
jointDamping = 0.001; % [Nm/rad/s]
```

Initialization and Trajectory Definition

Create a multiconstraint inverse kinematics solver.

```
gik = generalizedInverseKinematics(RigidBodyTree=kinovaGen3, ...
    ConstraintInputs={'pose','jointbounds'});
gik.SolverParameters.AllowRandomRestart = false;

% set joint constraints
jointConst = constraintJointBounds(kinovaGen3);
jointConst.Bounds(1,:) = q_home(1);
jointConst.Bounds(3,:) = q_home(3);
jointConst.Bounds(5,:) = q_home(5);
jointConst.Bounds(7,:) = q_home(7);

% get pose target
eeName = 'EndEffector_Link';
```

```
poseTgt = constraintPoseTarget(eeName);
poseTgt.ReferenceBody = kinovaGen3.BaseName;
```

Define initial and final pose.

```
% set initial position
q_init = q_home;
task_init = getTransform(kinovaGen3,q_init,eeName);
```

If needed, visualize the robot:

```
show(kinovaGen3,q_init);
axis auto;
view([60,10]);
hold on
```

On purpose, we set the task goal (end point) to be well inside the obstacle, to check if the impedance control can achieve a safe contact with small contact forces.

```
% get home location
task_goal = getTransform(kinovaGen3,q_home,eeName);

% set obstacle location
objectPos = task_goal(1,4)+0.1;

% set task end location to be inside the obstacle
depthOfIngress = 0.05; % m
task_goal(1,4) = objectPos  + depthOfIngress;

% assign target transform to pose
poseTgt.TargetTransform = task_goal;

% Use the multiconstraint inverse kinematics solver to get the goal pose
[q_goal,solnInfo] = gik(q_init ,poseTgt, jointConst);

% If necessary, uncomment to visualize the robot
% show(kinovaGen3,q_goal);
```

Next, assign the start time and duration for the trajectory.

```
tStart = 0.5;% maneuver start time
tEnd = 2; % maneuver end time
```

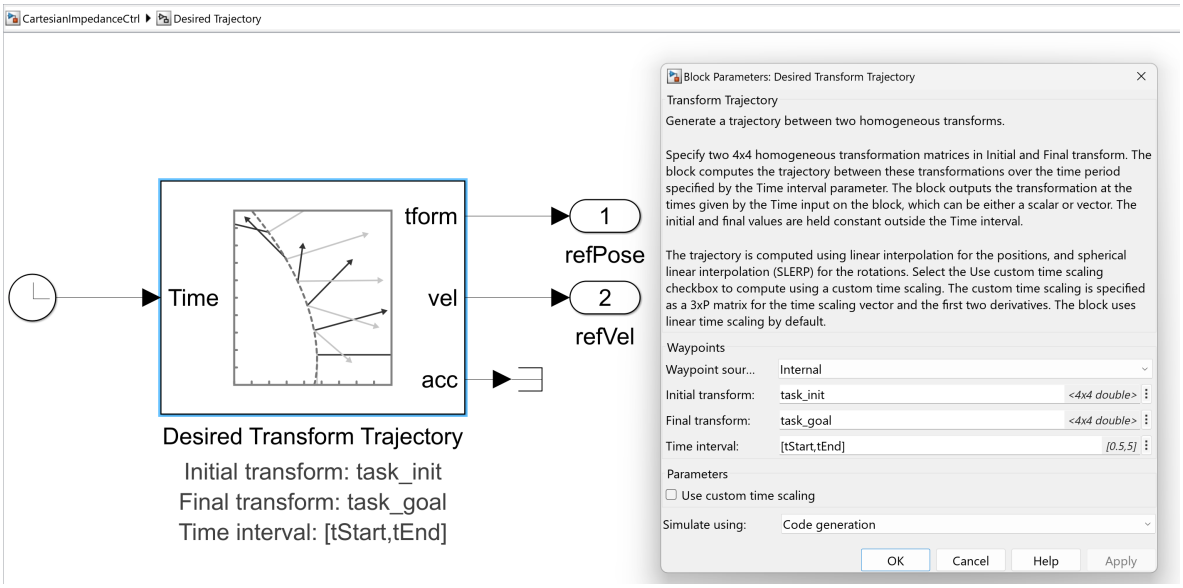
Finally, assign the initial and target configuration.

`q0` is the initial configuration. This parameter is used in the Multibody model joint blocks, to initialize the robot pose

```
q0 = q_init; % q0 is the parameter used in the Multibody model joint blocks, to
initialize the robot pose
```

Trajectory is implemented in Simulink® using **Transform Trajectory** block.

The **Transform Trajectory** block generates an interpolated **trajectory** between two homogenous transformation matrices.



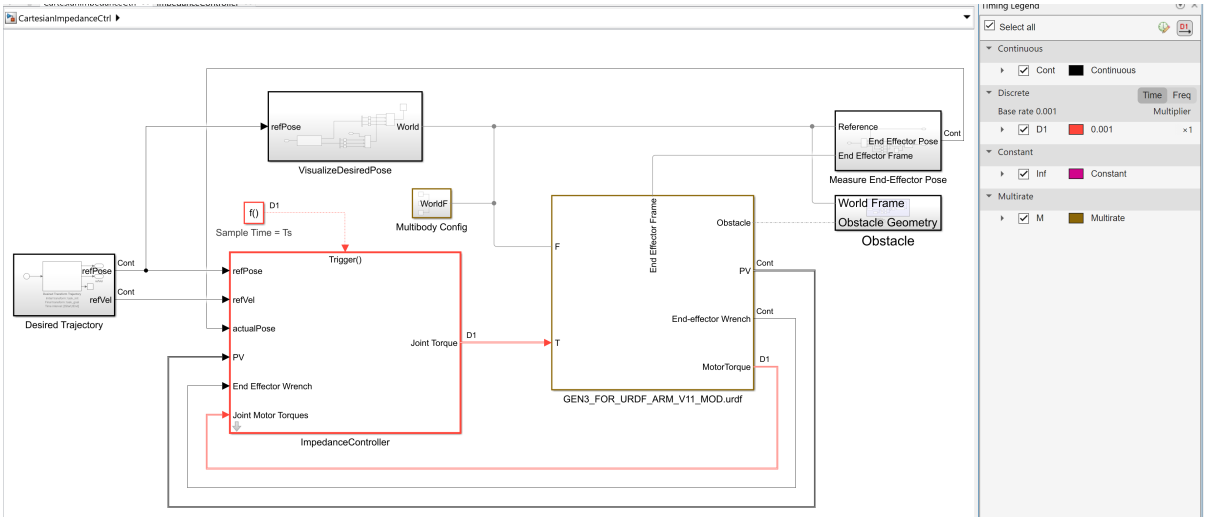
Impedance Control Model

Impedance control is a type of indirect force control at which we do not specify the forces to be applied by end-effector. Instead the interaction between the end-effector and environment is modeled as spring-mass-damper system. The impedance control aims to minimize contact forces between the end-effector and environment (object). This in turn makes impedance control is desirable for applications that require unexpected contact due to unmodeled environment. Compared this to the position control widely used in robotics that aims to track a motion trajectory. In the case of position control the discrepancy caused by unmodeled environment will result in high interaction forces that could be detrimental to the robot and enviroment.

In this example, we use an impedance controller that works in the cartesian space.

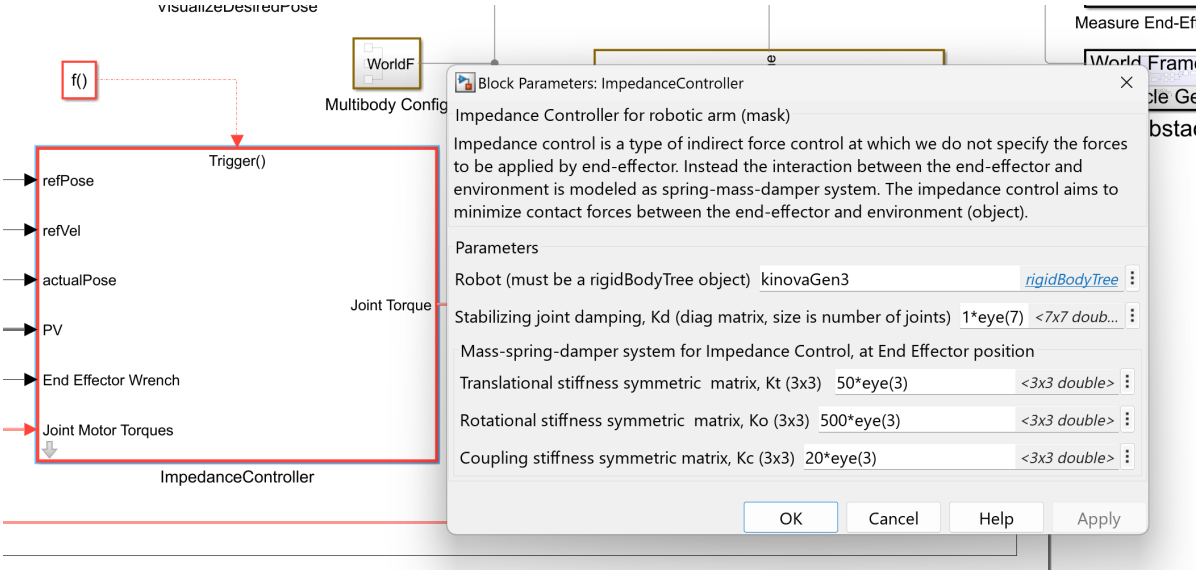
The controller is implemented in the **`ImpedanceController`** block, which is a discrete-time model:

```
% discrete controller time step
Ts = 0.001;
```

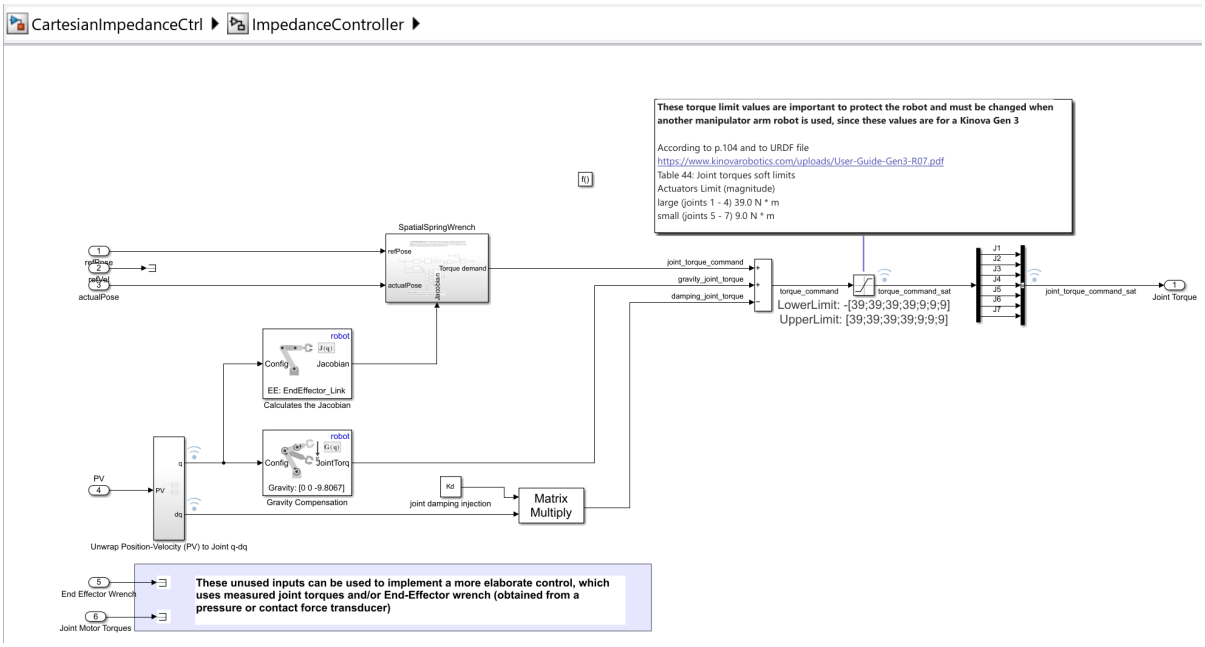


The impedance controller uses the **reference pose, sensed robot pose, and each joint's position and velocity**.

The impedance control parameters are defined in the block mask, as below. The meaning of the parameters is explained later.

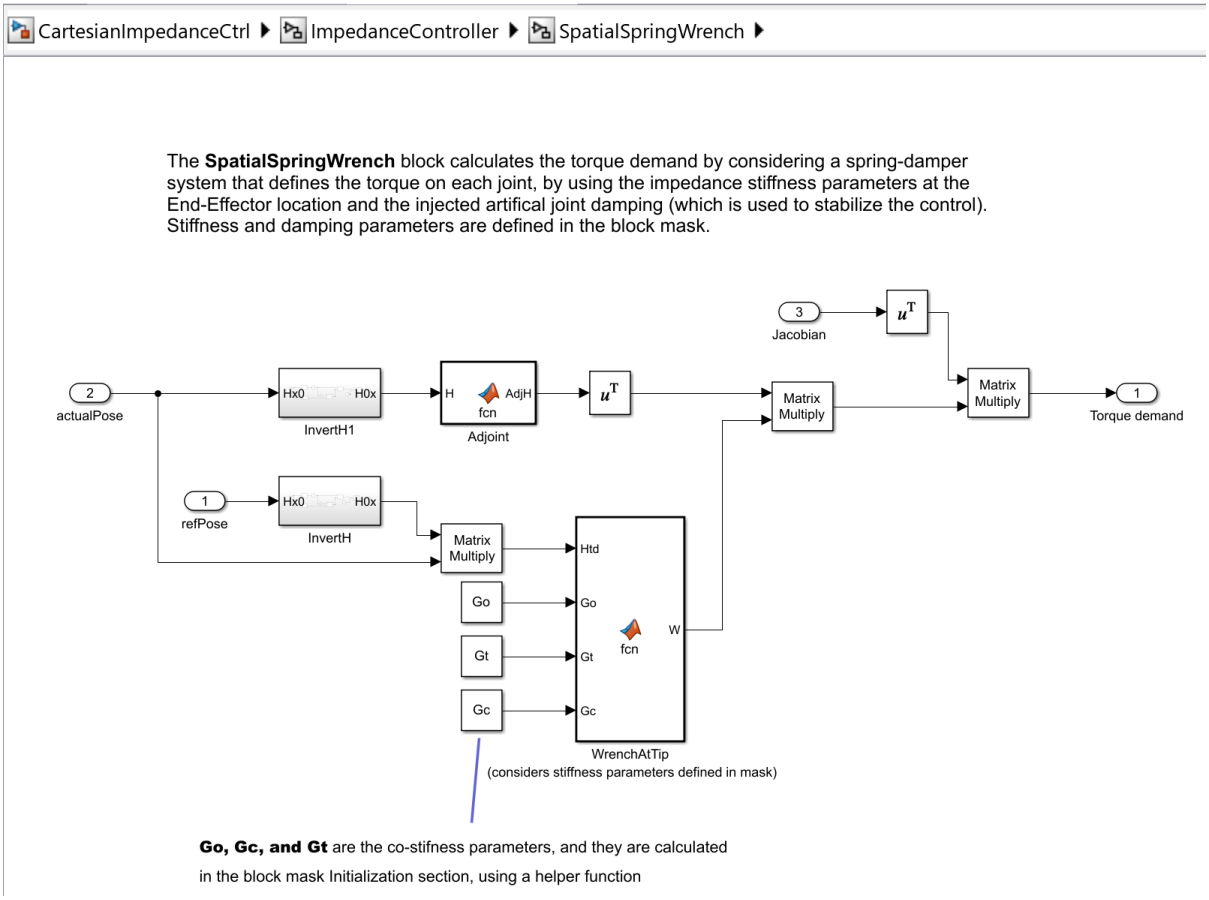


End-effector impedance is modelled in the SpatialSpringWrench block, using the equations described above, in the Multi-DOF Robot section.

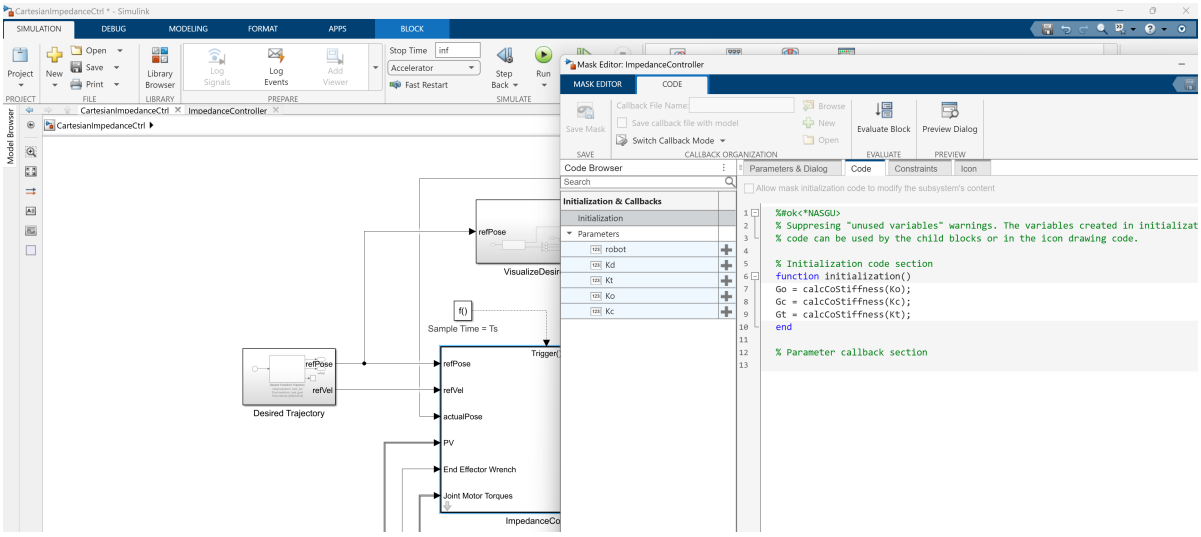


Note that the contact force model and joint torques are **not used** in the present controller, but can be used to develop more elaborate safe control algorithms in the future.

The **SpatialSpringWrench** block calculates the torque demand by considering a spring-damper system that defines the torque on each joint, by using the impedance stiffness parameters at the End-Effector location and the injected artifical joint damping (which is used to stabilize the control). Stiffness and damping parameters are defined in the block mask.



In the figure above, **Go, Gc, and Gt** are the co-stifness parameters, and they are calculated in the block mask Initialization section, using a helper function, as below.



Co-stiffness, often denoted as G_x, is a concept used in impedance control to incorporate additional damping into the control scheme based on the stiffness matrix K_x of the robotic arm. This term helps improve the stability and performance of the control system, especially in scenarios where high stiffness values alone may lead to instability or undesirable behavior.

The equation for G_x is:

$G_x = 0.5 \times \text{trace}(K_x) \times \text{eye}(3) - K_x$

Let's break down this equation:

- $\text{trace}(K_x)$ represents the sum of the diagonal elements of the stiffness matrix K_x. It gives us a measure of the overall stiffness of the system across all degrees of freedom.

-$\text{eye}(3)$ is the identity matrix of size 3x3. This matrix has ones on the diagonal and zeros elsewhere. Multiplying by $\text{trace}(K_x)$ effectively scales the identity matrix by the overall stiffness of the system.

-$0.5 \times \text{trace}(K_x) \times \text{eye}(3)$ computes a scaled version of the identity matrix based on the stiffness.

- Subtracting K_x from this scaled identity matrix effectively reduces the stiffness along each degree of freedom, introducing damping effects proportional to the stiffness.

In essence, G_x modifies the stiffness matrix $K_x$to introduce damping effects that are proportional to the overall stiffness of the system. This helps to balance the stiffness-damping trade-off and improve the stability and performance of the impedance control system.

In practical terms, incorporating co-stiffness into impedance control can enhance the behavior of robotic arms, allowing for more precise and stable interaction with the environment while mitigating the risk of oscillations or instability that may arise from high stiffness values alone.

We are now ready to run a simple simulation scenario, where the robot arm is hitting a soft obstacle that mimicks human skin and investigate how the robot reacts using the impedance controller. By controlling the stiffness and damping parameters of the impedance control, one can create a soft contact for the end-effector.

Simulation

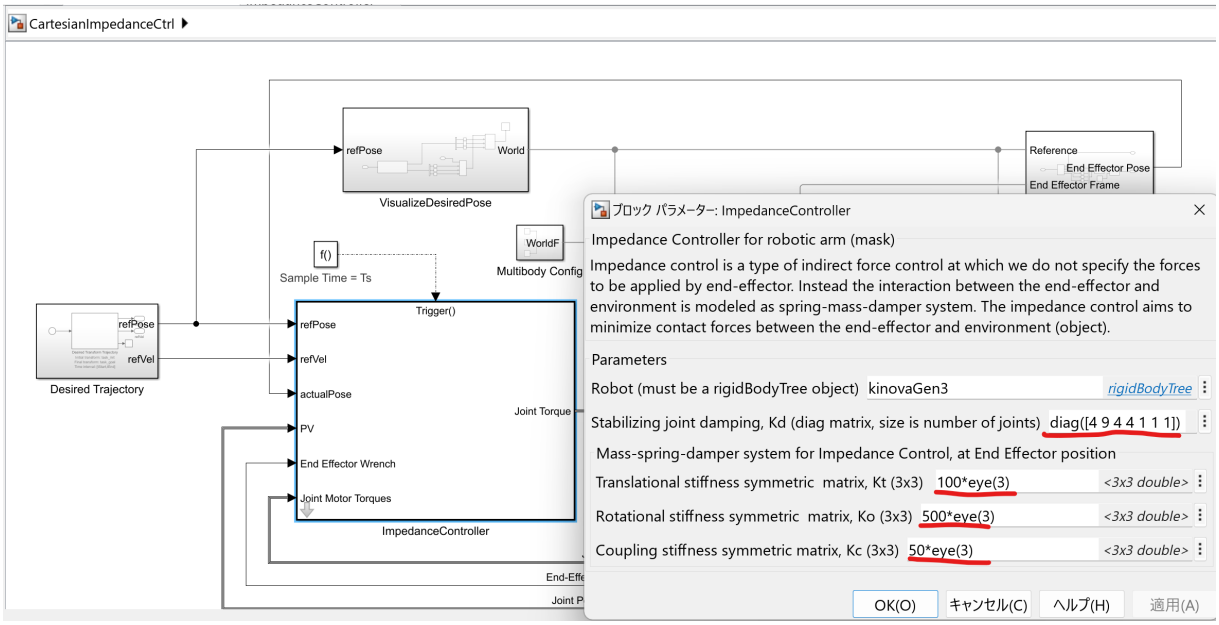
```
% obstacle position
objectPos = 0.6;

mdl = "CartesianImpedanceCtrl";
open_system(mdl);
open_system(mdl+"/GEN3_FOR_URDF_ARM_V11_MOD.urdf/Scope");

sim(mdl);
```

If necessary, adjust the controller parameters Kt, Ko, Kc, and Kd.

For the simulation we have used below values:

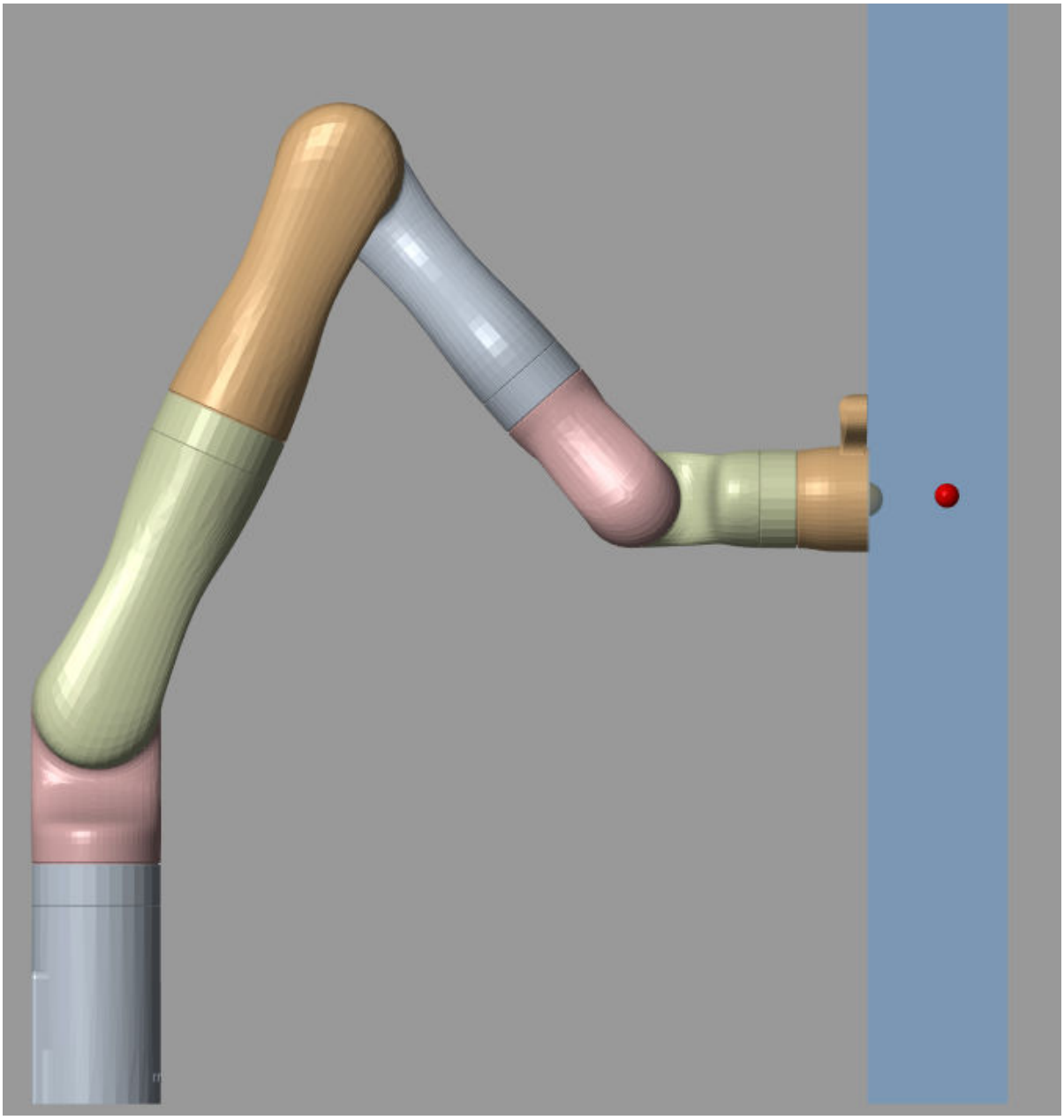


For Kd, we have used the values recommended for the admittance control by Kinova® in their User Guide for Gen 2 (page 36).

These values can be regarded as ballpark values, one must tune them for better performance. For example, controller parameters can be tuned using an optimization algorithm (check Simulink® Design Optimization™ documentation for more details).

Run the model and observe the behavior of Kinova® arm in the Mechanics Explorer.

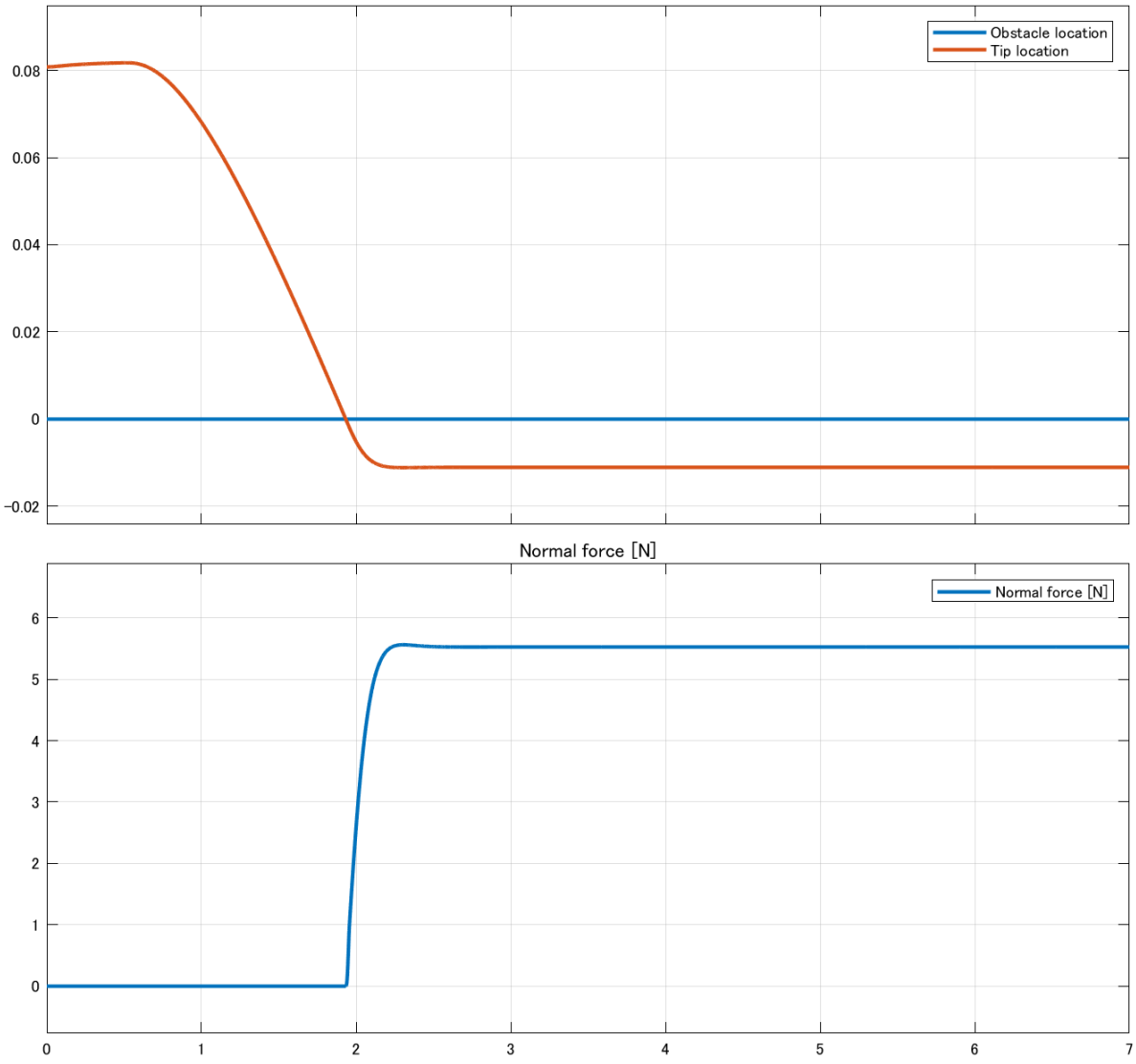
```
in = Simulink.SimulationInput(mdl);
out = sim(in);
open_system(mdl+"/GEN3_FOR_URDF_ARM_V11_MOD.urdf/Scope");
```



The results show the manipulator tip following a desired end-effector trajectory shown in red sphere. Note that the task goal is well inside the obstacle (set up by the `depthOfIngress` parameter defined previously). A recorded video of the test scenario is in ./images/CartesianImpedanceCtrl.mp4.

Discussion

Let's display the force at the tip and its position during the maneuver. As seen from the plot, the tip pushes the soft material about 1 cm only and then stops. The normal force remains very small, at less than 6 N (0.6 kg force). These values should pose no problem for a human being. If ncessary, the controller parameters can be adjusted for an even safer contact.



Conclusions

We have briefly introduced the reader to the still-active area of research of robotic impedance control, and have shown a simple implementation that can safely operate a Kinova® Gen3 arm close to a soft material resembling a human body.

Products needed to run the demo

1. MATLAB® version R2023b
2. Simulink®
3. Robotics System Toolbox™
4. Simscape™
5. Simscape™ Multibody™

Reference

1. G. Raiola, C. A. Cardenas, T. S. Tadele, T. de Vries and S. Stramigioli, "Development of a Safety- and Energy-Aware Impedance Controller for Collaborative Robots," in *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1237-1244, April 2018, doi: 10.1109/LRA.2018.2795639. keywords: {Robots;Impedance;Safety;Damping;Asymptotic stability;Measurement;Symmetric matrices;Robot safety;physical human–robot interaction;compliance and impedance control}, can be retrieved from https://iit-dlslab.github.io/papers/raiola18ral.pdf
2. Stefano Stramigioli, Modeling and IPC control of interactive mechanical systems. A coordinate-free approach. *Lecture Notes in Control and Information Sciences*. *Springer*, 2001.
3. KINOVA® Gen2 Ultra lightweight robot User Guide (link retrieved at April 16, 2024)
4. Kinova® Gen3 Ultra lightweight robot User Guide (link retrieved at April 16, 2024)