

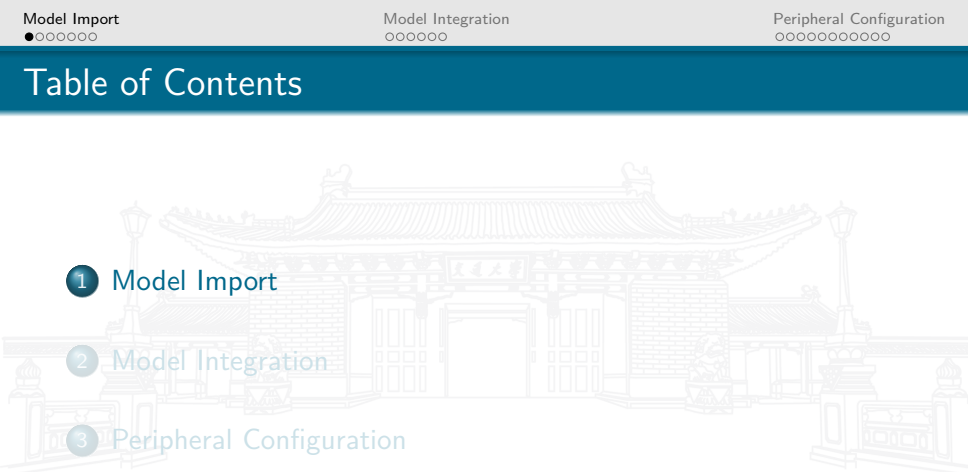
# Robotics (ME3403) Control System Configuration

Racheus Zhao

Team Roboticians  
School of Mechanical Engineering  
Shanghai Jiao Tong University

June 2, 2024

# Table of Contents

- 
- 1 Model Import
  - 2 Model Integration
  - 3 Peripheral Configuration



# Import Model to Simulink

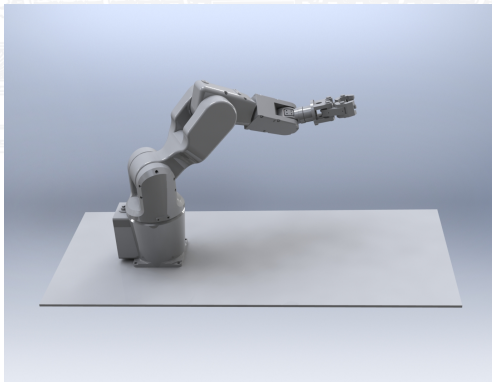
In Simulink, you can import models from other modeling environments, such as Solidworks.



Save the modeling file in XML format or URDF format, set a reasonable rotation coordinate system, and import it into Simulink for simulation.

# Model in Solidworks

Firstly, you should build a model in Solidworks. For this step I wish nothing but the best for you.



# Import Model to Simulink

Typing the following command in MATLAB command terminal to import the model.

## MATLAB Command

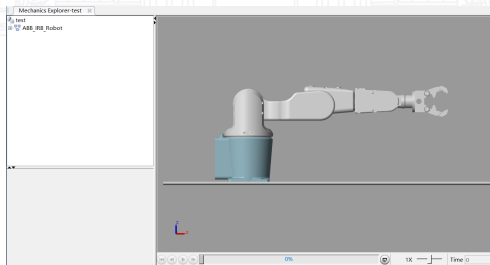
```
» smimport('model.xml')
```

# Import Model to Simulink

Typing the following command in MATLAB command terminal to import the model.

## MATLAB Command

```
» smimport('model.xml')
```



# Configuration of Model

In this type, we can see three types of data: **Rigid Transform**, **Solid** and **Revolute Joint**.

# Configuration of Model

In this type, we can see three types of data: **RigidTransform**, **Solid** and **Revolute Joint**.

- **RigidTransform**: The translation and rotation of the model.

## MATLAB Code

```
smiData.RigidTransform(6).translation = [170.00 45.99 19.99];  
smiData.RigidTransform(6).angle = 2.0944;  
smiData.RigidTransform(6).axis = [-0.577 -0.577 -0.577];  
smiData.RigidTransform(6).ID = "F[2-1:-:3-1]";
```



# Configuration of Model

In this type, we can see three types of data: **RigidTransform**, **Solid** and **Revolute Joint**.

- **RigidTransform**: The translation and rotation of the model.

## MATLAB Code

```
smiData.RigidTransform(6).translation = [170.00 45.99 19.99];  
smiData.RigidTransform(6).angle = 2.0944;  
smiData.RigidTransform(6).axis = [-0.577 -0.577 -0.577];  
smiData.RigidTransform(6).ID = "F[2-1:-:3-1]";
```

Translation means the position of the model, and angle is the rotation angle of the model, axis is the rotation axis of the model. ID is the only identification of the model.

# Configuration of Model

- **Solid:** Where the Solidworks model shows.

## MATLAB Code

```
smiData.Solid(11).mass = 5.2349061398896204;  
smiData.Solid(11).CoM = [-0.0002 85.3754 -7.55e-06];  
smiData.Solid(11).Mol = [34314.37 40472.28 51914.45];  
smiData.Solid(11).Pol = [0.01084 -0.0412 0.1905];  
smiData.Solid(11).ID = "1*:* 默认";
```

# Configuration of Model

- **Solid:** Where the Solidworks model shows.

## MATLAB Code

```
smiData.Solid(11).mass = 5.2349061398896204;  
smiData.Solid(11).CoM = [-0.0002 85.3754 -7.55e-06];  
smiData.Solid(11).Mol = [34314.37 40472.28 51914.45];  
smiData.Solid(11).Pol = [0.01084 -0.0412 0.1905];  
smiData.Solid(11).ID = "1*:* 默认";
```

Similarly, mass is the mass of the model, CoM is the center of mass of the model, Mol is the moment of inertia of the model, Pol is the product of inertia of the model, and ID is the only identification of the model.

# Configuration of Model

- **Revolute Joint:** The joint of the model, where revolute applies.

## MATLAB Code

```
smiData.RevoluteJoint(1).Rz.Pos = 115.30584675495646;  
smiData.RevoluteJoint(1).ID = "[0-1:-:1-1]";
```



# Configuration of Model

- **Revolute Joint:** The joint of the model, where revolute applies.

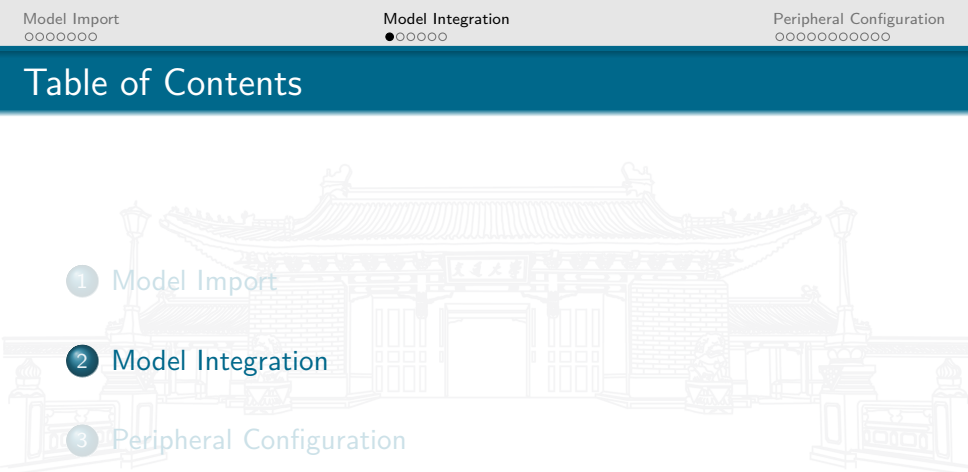
## MATLAB Code

```
smiData.RevoluteJoint(1).Rz.Pos = 115.30584675495646;  
smiData.RevoluteJoint(1).ID = "[0-1:-:1-1]";
```

Rz.Pos is the rotation angle of the joint, and ID is the only identification of the joint.

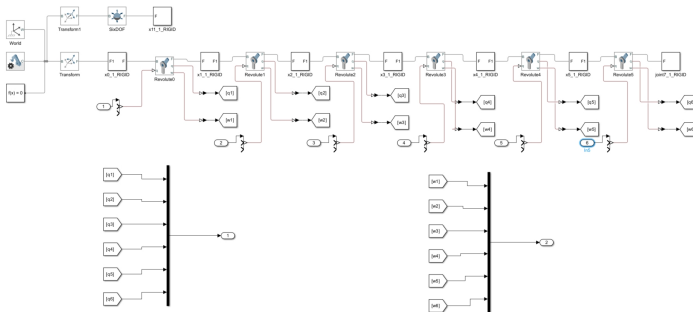
Now all the configurations are done, we can start the simulation.

# Table of Contents

- 
- 1 Model Import
  - 2 Model Integration
  - 3 Peripheral Configuration

# Simulink Model

There I 'd like to show you the ABB IRB 1200 model in Simulink/Simscape.I 've add some sensors and controllers to the model.



# Joint parameters

In this part we need to set the parameters of the joint, including the damping, stiffness and the initial angle of the joint.

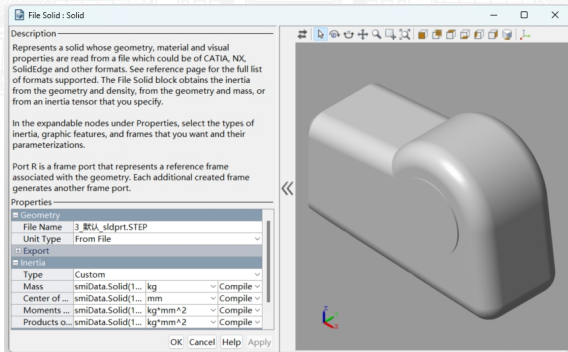
名称	值
<b>▼ Z Revolute Primitive (Rz)</b>	
<b>▼ State Targets</b>	
<input checked="" type="checkbox"/> Specify Position Target	
Priority	Low (approximate) ▼
Value	90 deg ▼ Compile-time ▼
<input type="checkbox"/> Specify Velocity Target	
<b>▼ Internal Mechanics</b>	
Equilibrium Position	0 rad ▼ Compile-time ▼
Spring Stiffness	20 N*m/rad ▼ Compile-time ▼
Damping Coefficient	5 N*m*s/rad ▼ Compile-time ▼
<b>▼ Limits</b>	
<input type="checkbox"/> Specify Lower Limit	
<input type="checkbox"/> Specify Upper Limit	
<b>▼ Actuation</b>	
Torque	Automatically Computed ▼
Motion	Provided by Input ▼
<b>▼ Sensing</b>	
<input checked="" type="checkbox"/> Position	
<input checked="" type="checkbox"/> Velocity	

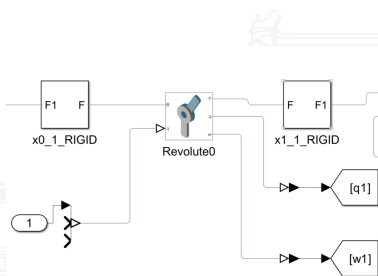


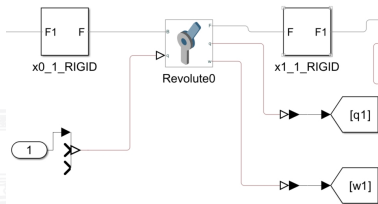
# Soild parameters

In this part we need to set the parameters of the solid, including the mass, the Col, the Mol and the Pol of the solid.

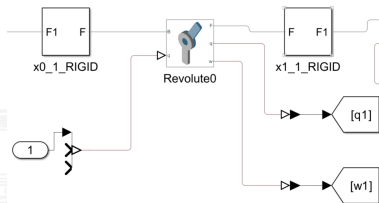
You **DON'T** need to modify any parameters of the solid, cause the parameters are imported from the Solidworks model.







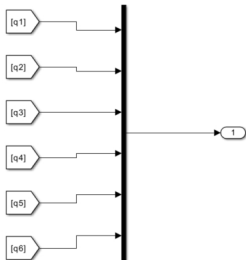
The input of the joint has two types, one is the torque, the other is the angle. Usually when we need to control the joint, we use the torque as the input, and if we need to see the position of the joint, we use the angle as the input.



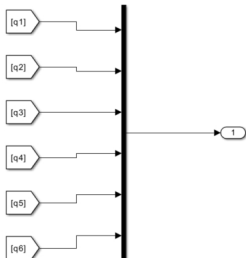
The input of the joint has two types, one is the torque, the other is the angle. Usually when we need to control the joint, we use the torque as the input, and if we need to see the position of the joint, we use the angle as the input.

As for the output of the joint (Sensor), we usually detect the angle of the joint ( $q(\theta)$ ), the velocity of the joint ( $\dot{q}(\omega)$ ) and the acceleration of the joint ( $\ddot{q}(\alpha)$ ), which is important in dynamic analysis.

# Collated Output



# Collated Output

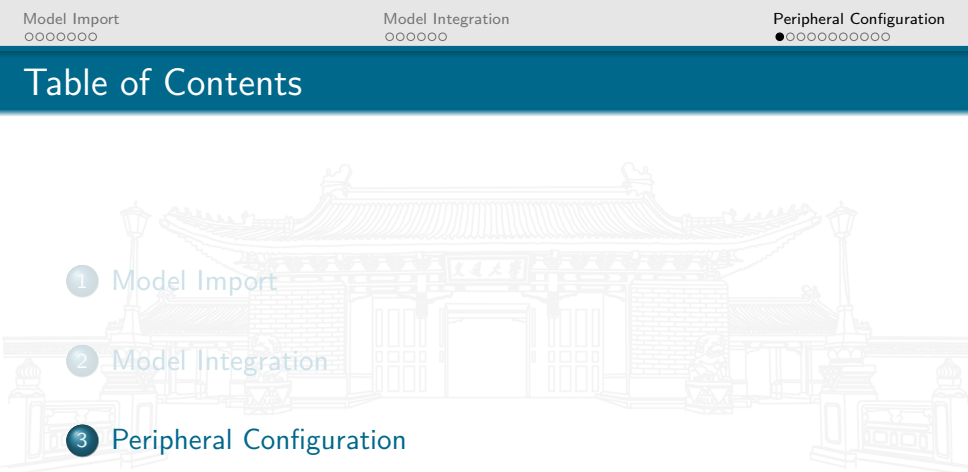


Using "Mux" to collate the output of the  $\dot{q}$  and  $\ddot{q}$ , and then make the robot as a whole. This **subsystem** is widely used in Simulink, which is a good way to service users.



ABB IRB Robot

# Table of Contents

- 
- 1 Model Import
  - 2 Model Integration
  - 3 Peripheral Configuration

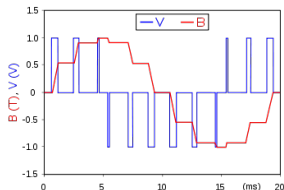
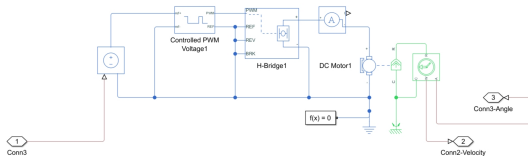
# DC Motor Configuration

When we Ctrl+G to make the ABB IRB 1200 as a subsystem, we should consider other parts of the robot, such as the DC motor.



# DC Motor Configuration

When we Ctrl+G to make the ABB IRB 1200 as a subsystem, we should consider other parts of the robot, such as the DC motor.



Pulse Width Modulation(PWM), is a technique for getting analog results with digital means.

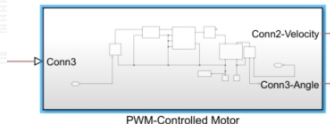
# DC Motor Configuration

- **H-Bridge:** The H-Bridge is a circuit that enables a low-voltage digital output to control a high-voltage motor.
- **DC Motor:** The DC motor is a device that converts electrical energy into mechanical energy.

Settings	Description
名称	值
Modeling option	No thermal port
Selected part	<click to select>
▼ <b>Electrical Torque</b>	
Field type	Permanent magnet
Model parameterization	By rated load and speed
Armature inductance	0.01 H
▼ No-load speed	400 rpm
Configurability	Compile-time
▼ Rated speed (at rated load)	250 rpm
Configurability	Compile-time
▼ Rated load (mechanical power)	10 W
Configurability	Compile-time
▼ Rated DC supply voltage	12 V
Configurability	Compile-time
Rotor damping parameterization	By damping value
> <b>Mechanical</b>	
> <b>Faults</b>	

# DC Motor Configuration

Then we can make it a subsystem and connect it to the ABB IRB 1200 model. The output of the DC motor in our Simulink model is the **Angle** of the joint.



If we use "Conn3-Angle" to the Revolute Joint(Motion Actuation),the process can be corresponded.The input of PWM-Controlled Motor's signal should be made by Simulink,too.

# Signal Design

- **Trapezoidal Signal:** The trapezoidal signal is a signal that is widely used in the control system. Relatively it is simple to calculate and implement, suitable for many industrial applications.

## Motion Profiles Trapezoidal Velocity Profile

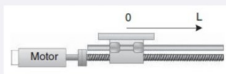
It shows the **jerk** (also called **jolt**), which is the **derivative of acceleration**. As it can be seen, the jerk is infinite at four points in this motion profile due to the discontinuities in the acceleration at corners of the velocity profiles. There are three distinct phases in the motion:

(1) acceleration, (2) constant velocity (zero acceleration), and (3) deceleration.

To move an axis of the machine, usually the following desired motion parameters are known:

- Move velocity  $zm$
- Acceleration  $a$
- Distance  $s$  to be traveled by the axis.

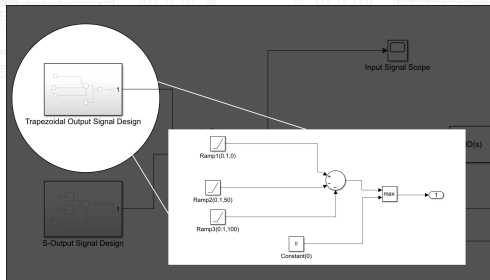
The desired motion profile can be programmed into the motion controller by first specifying the move velocity and move time for the motion. Then, the program commands the axis to move through distance  $s$ .



# Signal Design

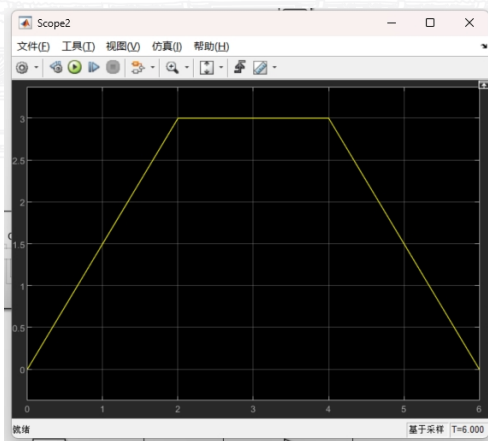
How to design: It is constructed by delaying and adding and subtracting the ramp signal, and starts zero-crossing detection.

$$\text{Ramp}(k, t) = \begin{cases} 0 & , t < 0 \\ kt & , 0 \leq t \leq \infty \end{cases}$$



# Signal Design

$$\text{Trapezoidal}(k, t) = \text{Ramp}(k, t) - \text{Ramp}(k, t - T_1) - \text{Ramp}(k, t - T_2)$$



# Signal Design

- **S-Curve Signal:** The S-Curve signal is a signal that is widely used in the control system. The acceleration gradually increases from zero to the maximum value, and then decreases from the maximum value to zero. During the whole process, the acceleration change is continuous without sudden changes.

# Signal Design

- **S-Curve Signal:** The S-Curve signal is a signal that is widely used in the control system. The acceleration gradually increases from zero to the maximum value, and then decreases from the maximum value to zero. During the whole process, the acceleration change is continuous without sudden changes.

$$f_S(x; a, b) = \begin{cases} 0 & , x \leq a \\ 2\left(\frac{x-a}{b-a}\right)^2 & , a \leq x \leq \frac{a+b}{2} \\ 1 - 2\left(\frac{x-b}{b-a}\right)^2 & , \frac{a+b}{2} \leq x \leq b \\ 1 & , x \geq b \end{cases}$$



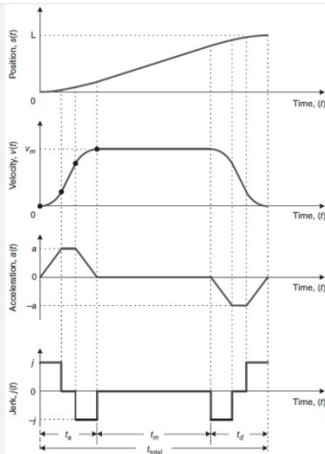
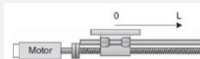
# Signal Design

## Motion Profiles S-curve Velocity Profile

The trapezoidal velocity profile is simple to compute but it has a major drawback. The sharp corners of the trapezoid cause discontinuities in the acceleration which then lead to infinite (or in practice large) jolts in the system.

To smooth out the acceleration, the sharp corners are rounded to create the so-called *S-curve velocity profile*.

The rounded corners are made up of second-order polynomials. This reshaping of the velocity profile smooths the transitions between positive, zero, and negative acceleration phases. Note that unlike the trapezoidal velocity profile, the acceleration is not constant.

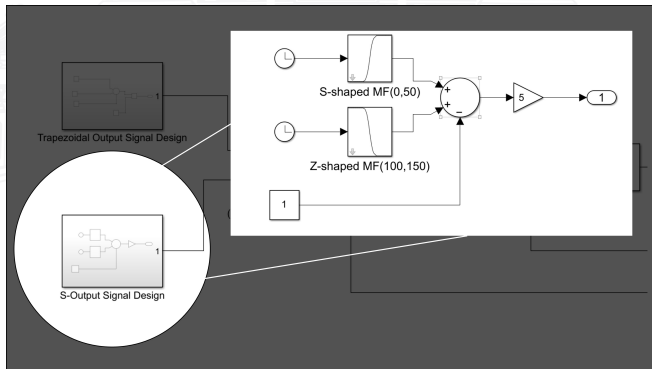
page  
019

## From Motion Control System, AU3324



# Signal Design

In Simulink, the S-curve signal can be created by the **S-shaped MF** and **Z-shaped MF** module, which derives from the **Fuzzy Logic Toolbox**.



# Signal Design

Put a scope to see the S-curve signal:

