CS225A : Experimental Robotics

Lecture 5 : Fine-Tuning Controllers for Real Robots

(Aka.. How to fix a misbehaving robot)

Samir Menon

Oct 07, 2014

Projects...

- Select team & project deadline => “Today”!!!

- Please email Samir & Tas:
  - Your team
  - Project title
  - A few paragraphs to describe the project
  - Platform: Robot/Haptics/Humanoid/Skeleton

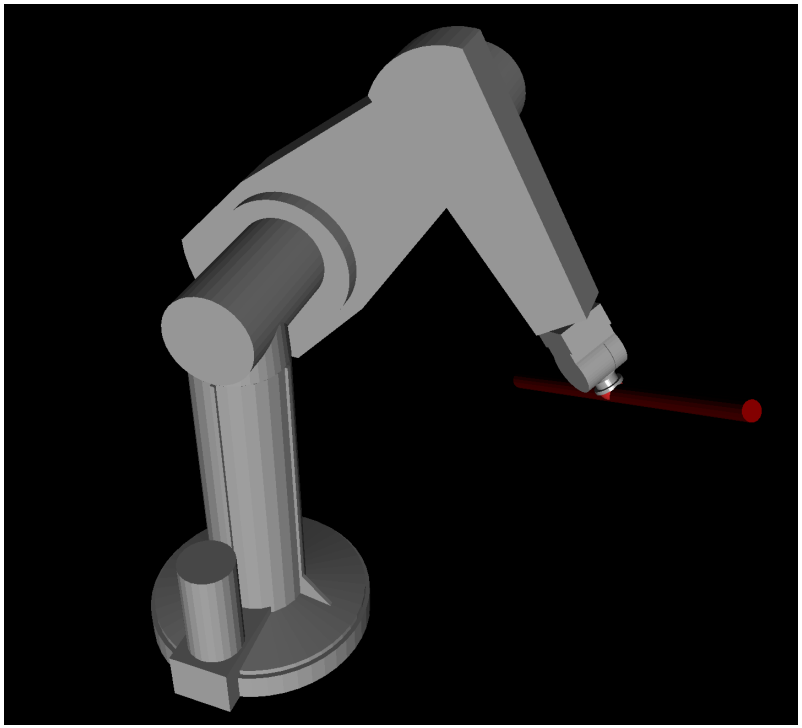
- If you are not in a group, stay around after class...
  - We will assign you to a group.

- If you have not discussed your project idea with Samir and/or one of the TAs.
  - Please do so after class.

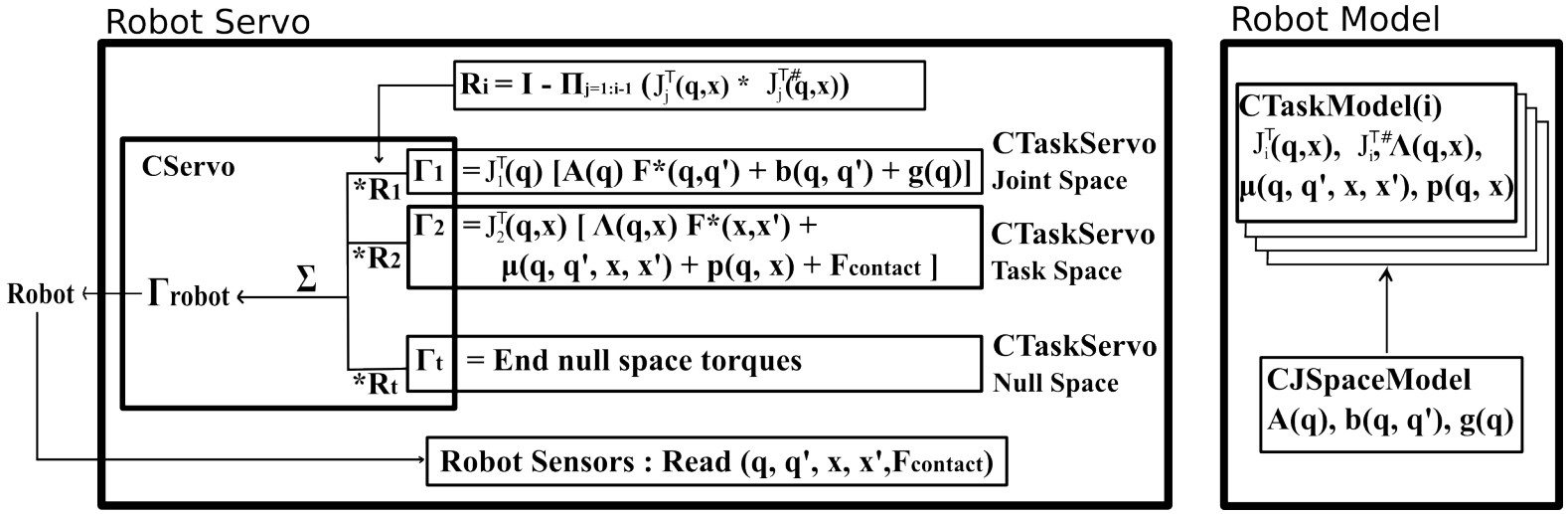
Control Formulation

$$\Gamma = J_t^{\mathsf{T}} \left(\Lambda(x)\ddot{x} + \mu(x, \dot{x}) + p(x) \right)$$

Theory



Practice...



- Gains?
- Rate of dynamics update?
- Inertial errors?

SCL : Fine Tuning Controller Specification

- Understand how control and inertial specifications affect motion
- Understand how to debug control specifications
- Use advanced features in SCL to simplify your life
- Tutorial applications are basic and very verbose.
- Application to be used :
  - scl.git/applications-linux/scl_example_ctrl/
  - Offers easy manipulation with a GUI/keyboard

SCL : Run controller in real-time

- The default scl_eg_ctrl application runs (much) faster than real-time for simple robots

SCL : Run controller in real-time

- The default scl_eg_ctrl application runs (much) faster than real-time for simple robots

- So we will slow it down

SCL : Run controller in real-time

- The default scl_eg_ctrl application runs (much) faster than real-time for simple robots

- So we will slow it down

```
robot_.integrateDynamics();      //Integrate system

sutil::CSystemClock::tick(db_->sim_dt_);//Tick the clock.

/** Slow down sim to real time */
double tcurr = sutil::CSystemClock::getSysTime();
double tdiff = sutil::CSystemClock::getSimTime() - tcurr;
while(tdiff > 1)
{
  sleep(1);
  tcurr = sutil::CSystemClock::getSysTime();
  tdiff = sutil::CSystemClock::getSimTime() - tcurr;
}
if(tdiff > 0){
  timespec ts = {0, 0};
  ts.tv_nsec = tdiff*1e9;
  nanosleep(&ts,NULL);
  //std::cout<<"\n Sleeping : "<<tdiff;
}
```

SCL : Run controller for the Puma robot

```
scl_example_ctrl$ ./scl_eg_ctrl ../../specs/Puma/PumaCfg.xml PumaBot opc -op hand
```

SCL : Monitor robot stats

```
scl_example_ctrl$ ./scl_eg_ctrl ../../specs/Puma/PumaCfg.xml PumaBot opc -op hand
```

```
*** Robot printables ***

scl>> print <printable name>

Printable information                                Printable name

Printable: Robot:                                         PumaBotParsed
Printable: PumaBot SRigidBody                                      base
Printable: PumaBot SRigidBody                                 upper_arm
Printable: PumaBot SRigidBody                                 lower_arm
Printable: PumaBot SRigidBody                                wrist-hand
Printable: PumaBot SRigidBody                               wrist-finger
Printable: PumaBot SRigidBody                               end-effector
Printable: PumaBot SRigidBody                                     ground
Printable: Robot:                                               PumaBot
Printable: Task Controller:                                     opcpida
Printable: Task Controller:                                         opc
Printable: Task:                                              handpida
Printable: Task:                                 NullSpaceDampingTaskpida
Printable: Task:                                                   hand
Printable: Task:                                     NullSpaceDampingTask
Printable: UI-point                                                 ui0
Printable: UI-point                                                 ui1
Printable: UI-point                                                 ui2
Printable: UI-point                                                 ui3
Printable: UI-point                                                 ui4
Printable: UI-point                                                 ui5
Printable: UI-point                                                 ui6
Printable: UI-point                                                 ui7
Printable: UI-point                                                 ui8
Printable: UI-point                                                 ui9
Printable: UI-point                                                ui10
Printable: UI-point                                                ui11
```

SCL : Monitor robot stats (parsed spec)

```
scl>> print  PumaBotParsed

 Name: PumaBot(SRobotParsed)
  Dof: 6
  Jlim_max : 3.14159 3.14159 3.14159 3.14159 3.14159 1.14159
  Jlim_min : -3.14159 -3.14159 -3.14159 -3.14159 -3.14159 -1.14159
 Actuator_force_max : 3000 3000 3000 3000 3000 3000
 Actuator_force_min : -3000 -3000 -3000 -3000 -3000 -3000
 Flag_gc_pos_limits : 0
 Flag_gc_damping : 0
 Flag_force_limits : 1
 Flag_vel_limits : 1
 Flag_accel_limits : 1
```

SCL : Monitor robot stats (dynamic state)

```
scl>> print PumaBot

 Name: PumaBot(SRobotIO)
Dof: 6
Has been init: 1
  q:  0.000131399  2.02691e-06  5.41261e-05 -0.000135674  -0.00090126  0.000784804
 dq:  1.17616e-05  2.31701e-07  4.88433e-06 -1.25671e-05 -8.06785e-05   7.5669e-05
ddq: -1.32084e-07 -2.35087e-09 -3.35198e-08  5.93572e-08  9.05324e-07  2.07644e-07
Measured force: 2.36024e-05    -42.2382    0.248068 1.48885e-06  -0.0281938 3.31378e-08
 Command force: 2.36024e-05    -42.2382    0.248068 1.48885e-06  -0.0281938 3.31378e-08
```

- Press “enter” to repeat last command
- E.g., keep “enter” pressed to constantly print state

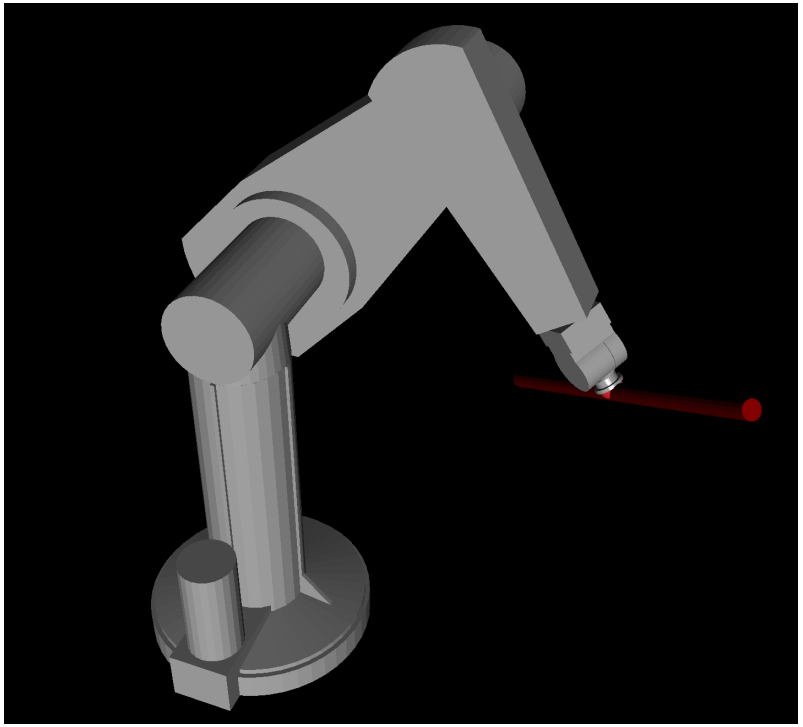
SCL : Monitor robot controller; one/all tasks

```
scl>> print opc

 Name: opc(SControllerMultiTask)
 Init      : 1

 === Task #0:
 Name: NullSpaceDampingTask(STaskBase). Parent(opc)
 Init/Active : 1/1
 Priority    : 2
 F_task :  0.00447506 8.86907e-05  0.00189555 -0.00492343  -0.0306976   0.0306376
 F_gc   :  -0.0452826  -0.0358453  -0.0356239 0.000249047   0.0337845  -0.0358882

 === Task #1:
 Name: hand(STaskBase). Parent(opc)
 Init/Active : 1/1
 Priority    : 0
 F_task : -3.47049 -1.15575  98.3928
 F_gc   :    0.0453092     -42.2026     0.283383 -0.000246922   -0.0619562    0.0358882

scl>> print hand

 Name: hand(STaskBase). Parent(opc)
 Init/Active : 1/1
 Priority    : 0
 F_task : -3.46894 -1.16042  98.3928
 F_gc   :    0.0431538     -42.2026     0.283447 -0.000308453   -0.0603911    0.0362089
```

SCL : XML Controller Specification

```
<controller name="opc">
    <type>task</type>
    <must_use_robot>PumaBot</must_use_robot>

    <task name="hand">
        <type>TaskOpPos</type>
        <parent_link>end-effector</parent_link>
        <pos_in_parent>0.01 0.00 0.00</pos_in_parent>
        <priority>0</priority>
        <task_dof>3</task_dof>
        <kp>1000</kp>
        <kv>200</kv>
        <ka>0</ka>
        <force_max>1000</force_max>
        <force_min>-1000</force_min>
    </task>

    <!--0 task_dof means a gc task, ie full dofs -->
    <task name="NullSpaceDampingTask">
        <type>TaskNullSpaceDamping</type>
        <priority>2</priority>
        <task_dof>0</task_dof>
        <kp>0</kp>
        <kv>400</kv>
        <ka>0</ka>
        <force_max>500</force_max>
        <force_min>-500</force_min>
    </task>
</controller>
```

SCL : PumaBot Demo



SCL : Effect of link inertia

```
<link>
  <link_name>end-effector</link_name>
  <position_in_parent>-0.070000 0.000000 0.000000</position_in_parent>
  <orientation_in_parent>0.7071 0 0 0.7071</orientation_in_parent>
  <mass>0.090000</mass>
  <inertia>0.100150 0.100150 0.193040</inertia>
  <center_of_mass>0.000000 0.000000 0.032000</center_of_mass>
  <joint_name>wrist-roll2</joint_name>
  <parent_link_name>wrist-finger</parent_link_name>
  <joint_type>rz</joint_type>
  <joint_limits>-1.141590 1.141590</joint_limits>
  <default_joint_position>0.000000</default_joint_position>
  <graphics>
    <obj_file>
      <name>Puma/graphics/shaft2_bl.obj</name>
      <position_in_parent>0.000000 0.000000 0.000000</position_in_parent>
      <orientation_in_parent>-0.707106 0 0 0.707106</orientation_in_parent>
      <collision_type>0</collision_type>
    </obj_file>
  </graphics>
</link>
```

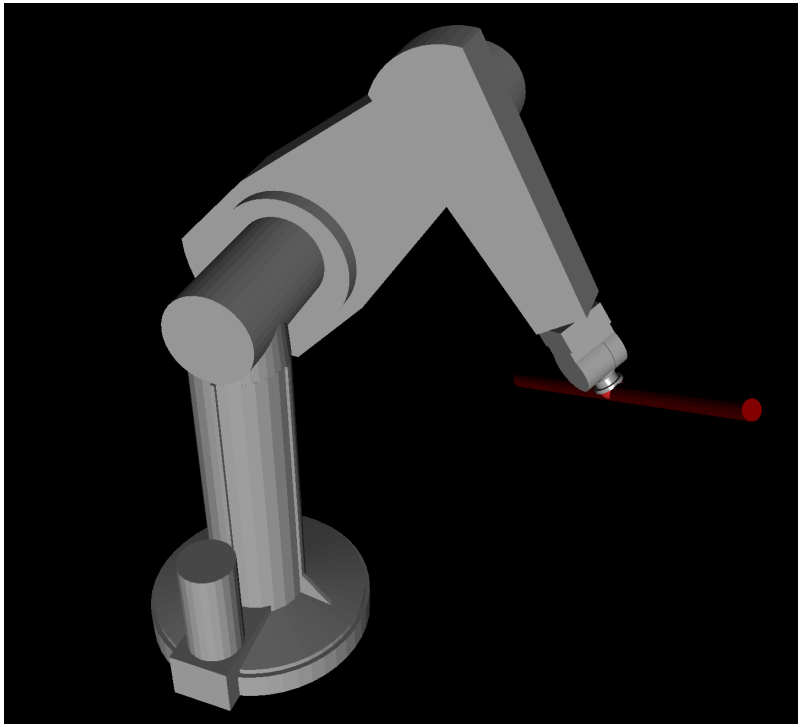
- Observe low inertias at the end-effector
- What aspect of the demo motion is affected?

SCL : Effect of link inertia

```
<link>
  <link_name>end-effector</link_name>
  <position_in_parent>-0.070000 0.000000 0.000000</position_in_parent>
  <orientation_in_parent>0.7071 0 0 0.7071</orientation_in_parent>
  <mass>0.090000</mass>
  <inertia>0.100150 0.100150 0.193040</inertia>
  <center_of_ <link>
  <joint_nam    <link_name>end-effector</link_name>
  <parent_li    <position_in_parent>-0.070000 0.000000 0.000000</position_in_parent>
  <joint_typ    <orientation_in_parent>0.7071 0 0 0.7071</orientation_in_parent>
  <joint_lim    <mass>0.090000</mass>
  <default_j    <inertia>1.100150 1.100150 0.193040</inertia>
  <graphics>    <center_of_mass>0.000000 0.000000 0.032000</center_of_mass>
    <obj_fil    <joint_name>wrist-roll2</joint_name>
      <name>    <parent_link_name>wrist-finger</parent_link_name>
      <posit    <joint_type>rz</joint_type>
      <orien    <joint_limits>-1.141590 1.141590</joint_limits>
      <colli    <default_joint_position>0.000000</default_joint_position>
    </obj_fi    <graphics>
  </graphics>     <obj_file>
</link>             <name>Puma/graphics/shaft2_bl.obj</name>
                    <position_in_parent>0.000000 0.000000 0.000000</position_in_parent>
                    <orientation_in_parent>-0.707106 0 0 0.707106</orientation_in_parent>
                    <collision_type>0</collision_type>
                  </obj_file>
                </graphics>
              </link>
```

- Let us increase the inertia (artificially high now)
- Effect on motion?

SCL : PumaBot w/ Wrist Inertia Demo



SCL : Low Task-Space Damping Gain

```
<controller name="opc-lkv-task">
    <type>task</type>
    <must_use_robot>PumaBot</must_use_robot>

    <task name="hand">
        <type>TaskOpPos</type>
        <parent_link>end-effector</parent_link>
        <pos_in_parent>0.01 0.00 0.00</pos_in_parent>
        <priority>0</priority>
        <task_dof>3</task_dof>
        <kp>100</kp>
        <kv>0</kv>
        <ka>0</ka>
        <force_max>1000</force_max>
        <force_min>-1000</force_min>
    </task>

    <!--0 task_dof means a gc task, ie full dofs -->
    <task name="NullSpaceDampingTask">
        <type>TaskNullSpaceDamping</type>
        <priority>2</priority>
        <task_dof>0</task_dof>
        <kp>0</kp>
        <kv>400</kv>
        <ka>0</ka>
        <force_max>500</force_max>
        <force_min>-500</force_min>
    </task>
</controller>
```

SCL : Low Task-Space Damping Gain

```
<controller name="opc-lkv-task">
    <type>task</type>
    <must_use_robot>PumaBot</must_use_robot>

    <task name="hand">
        <type>TaskOpPos</type>
        <parent_link>end-effector</parent_link>
        <pos_in_parent>0.01 0.00 0.00</pos_in_parent>
        <priority>0</priority>
        <task_dof>3</task_dof>
        <kp>100</kp>
        <kv>0</kv>
        <ka>0</ka>
        <force_max>1000</force_max>
        <force_min>-1000</force_min>
    </task>

    <!--0 task_dof means a gc task, ie full dofs -->
    <task name="NullSpaceDampingTask">
        <type>TaskNullSpaceDamping</type>
        <priority>2</priority>
        <task_dof>0</task_dof>
        <kp>0</kp>
        <kv>400</kv>
        <ka>0</ka>
        <force_max>500</force_max>
        <force_min>-500</force_min>
    </task>
</controller>
```

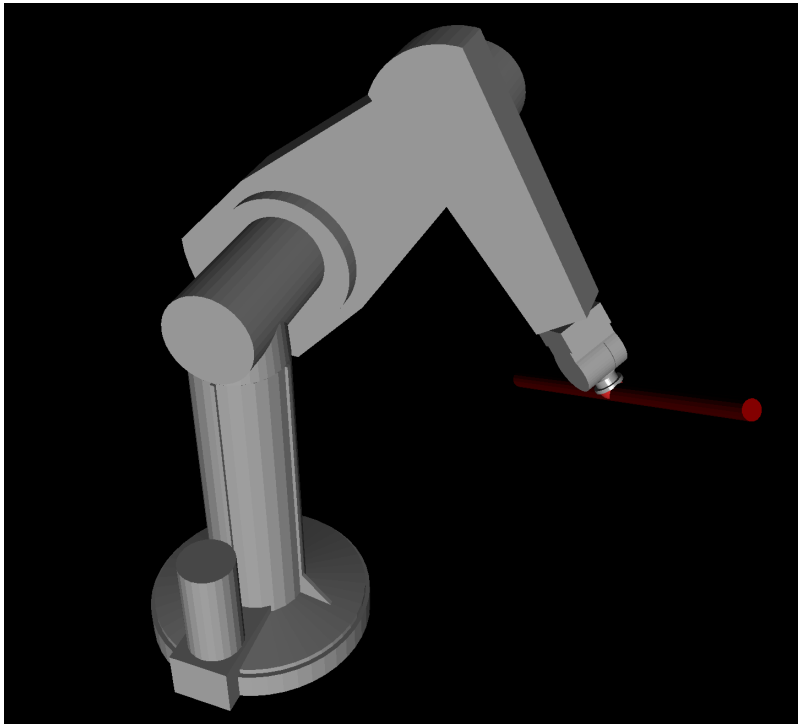
- Note high null-space damping gain...

SCL : Low Task-Space Damping Gain

```
<controller name="opc-lkv-task">
    <type>task</type>
    <must_use_robot>PumaBot</must_use_robot>

    <task name="hand">
        <type>TaskOpPos</type>
        <parent_link>end-effector</parent_link>
        <pos_in_parent>0.01 0.00 0.00</pos_in_parent>
        <priority>0</priority>
        <task_dof>3</task_dof>
        <kp>100</kp>
        <kv>0</kv>
        <ka>0</ka>
        <force_max>1000</force_max>
        <force_min>-1000</force_min>
    </task>

    <!--0 task_dof means a gc task, ie full dofs -->
    <task name="NullSpaceDampingTask">
        <type>TaskNullSpaceDamping</type>
        <priority>2</priority>
        <task_dof>0</task_dof>
        <kp>0</kp>
        <kv>400</kv>
        <ka>0</ka>
        <force_max>500</force_max>
        <force_min>-500</force_min>
    </task>
</controller>
```

- What motion do you predict?

SCL : PumaBot w/ Low Task-damping Demo



SCL : Low Null-Space Damping Gain

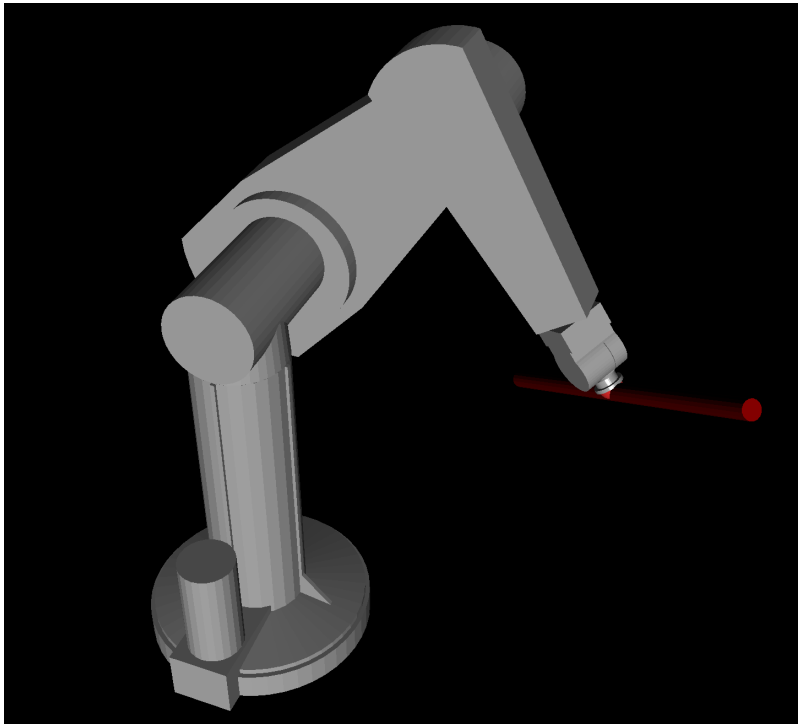
```
<controller name="opc-lkv-null">
    <type>task</type>
    <must_use_robot>PumaBot</must_use_robot>

    <task name="hand-n">
        <type>TaskOpPos</type>
        <parent_link>end-effector</parent_link>
        <pos_in_parent>0.01 0.00 0.00</pos_in_parent>
        <priority>0</priority>
        <task_dof>3</task_dof>
        <kp>100</kp>
        <kv>20</kv>
        <ka>0</ka>
        <force_max>1000</force_max>
        <force_min>-1000</force_min>
    </task>

    <!--0 task_dof means a gc task, ie full dofs -->
    <task name="NoNullDamp">
        <type>TaskNullSpaceDamping</type>
        <priority>2</priority>
        <task_dof>0</task_dof>
        <kp>0</kp>
        <kv>0</kv>
        <ka>0</ka>
        <force_max>500</force_max>
        <force_min>-500</force_min>
    </task>
</controller>
```

SCL : Low Null-Space Damping Gain

```
<controller name="opc-lkv-null">
    <type>task</type>
    <must_use_robot>PumaBot</must_use_robot>

    <task name="hand-n">
        <type>TaskOpPos</type>
        <parent_link>end-effector</parent_link>
        <pos_in_parent>0.01 0.00 0.00</pos_in_parent>
        <priority>0</priority>
        <task_dof>3</task_dof>
        <kp>100</kp>
        <kv>20</kv>
        <ka>0</ka>
        <force_max>1000</force_max>
        <force_min>-1000</force_min>
    </task>

    <!--0 task_dof means a gc task, ie full dofs -->
    <task name="NoNullDamp">
        <type>TaskNullSpaceDamping</type>
        <priority>2</priority>
        <task_dof>0</task_dof>
        <kp>0</kp>
        <kv>0</kv>
        <ka>0</ka>
        <force_max>500</force_max>
        <force_min>-500</force_min>
    </task>
</controller>
```

- What motion do you predict?

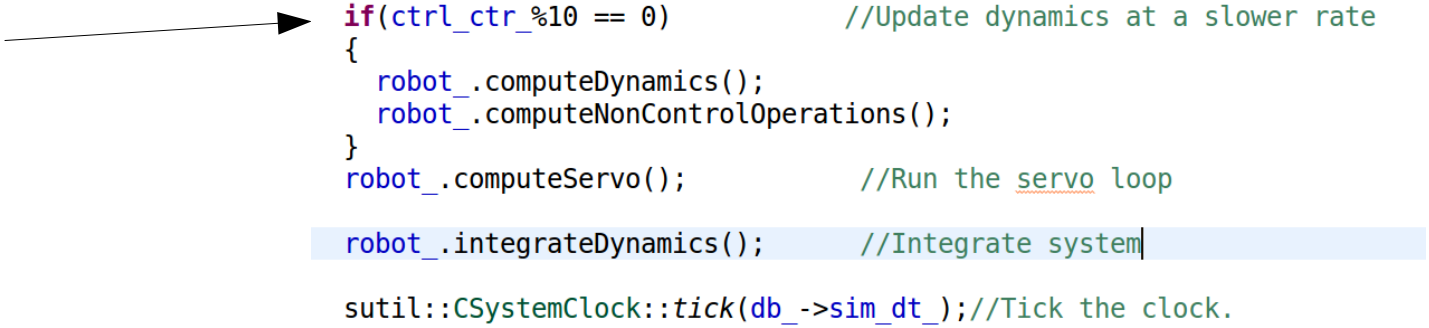
SCL : PumaBot w/ Low Null-damping Demo



SCL : Dynamics Update Rate

```
if(ctrl_ctr_%10 == 0)           //Update dynamics at a slower rate
{
  robot_.computeDynamics();
  robot_.computeNonControlOperations();
}
robot_.computeServo();           //Run the servo loop

robot_.integrateDynamics();      //Integrate system

sutil::CSystemClock::tick(db_->sim_dt_);//Tick the clock.
```

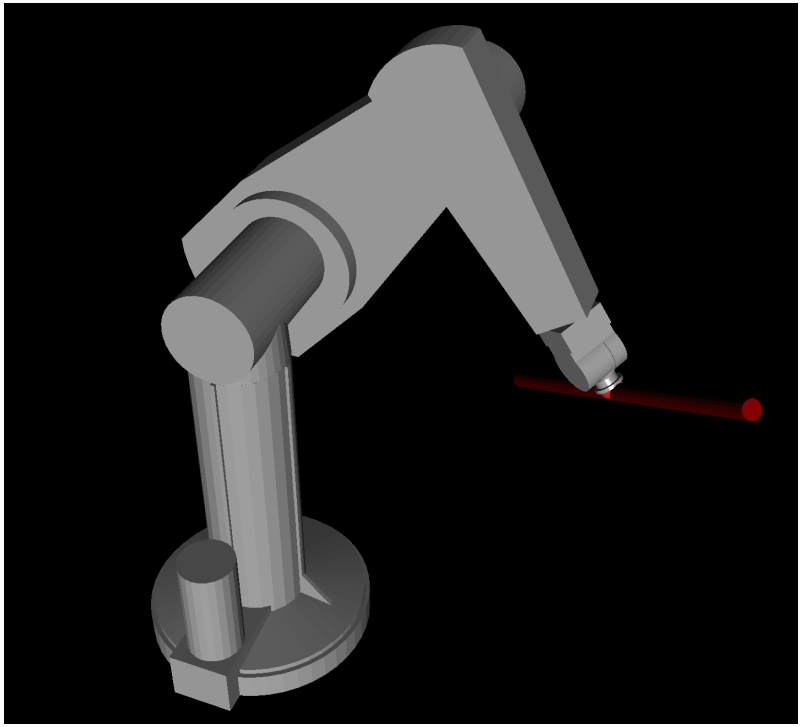
SCL : Dynamics Update Rate



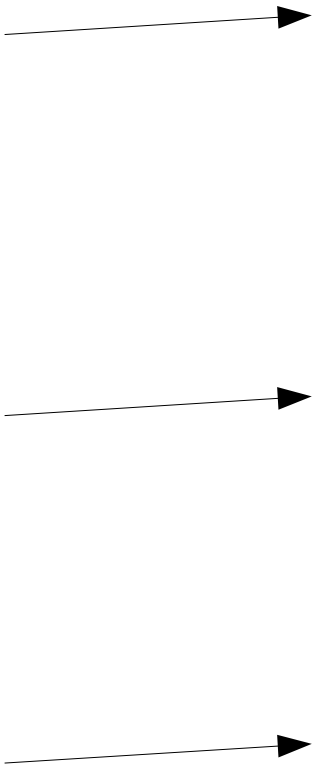
```
if(ctrl_ctr_%10 == 0)           //Update dynamics at a slower rate
{
  robot_.computeDynamics();
  robot_.computeNonControlOperations();
}
robot_.computeServo();           //Run the servo loop

robot_.integrateDynamics();      //Integrate system

sutil::CSystemClock::tick(db_->sim_dt_);//Tick the clock.
```

- Speeds up controller for large robots
- Not really a problem for the Puma (already fast)
- What motion do you predict?

SCL : PumaBot w/ Slow Dyn Update Demo

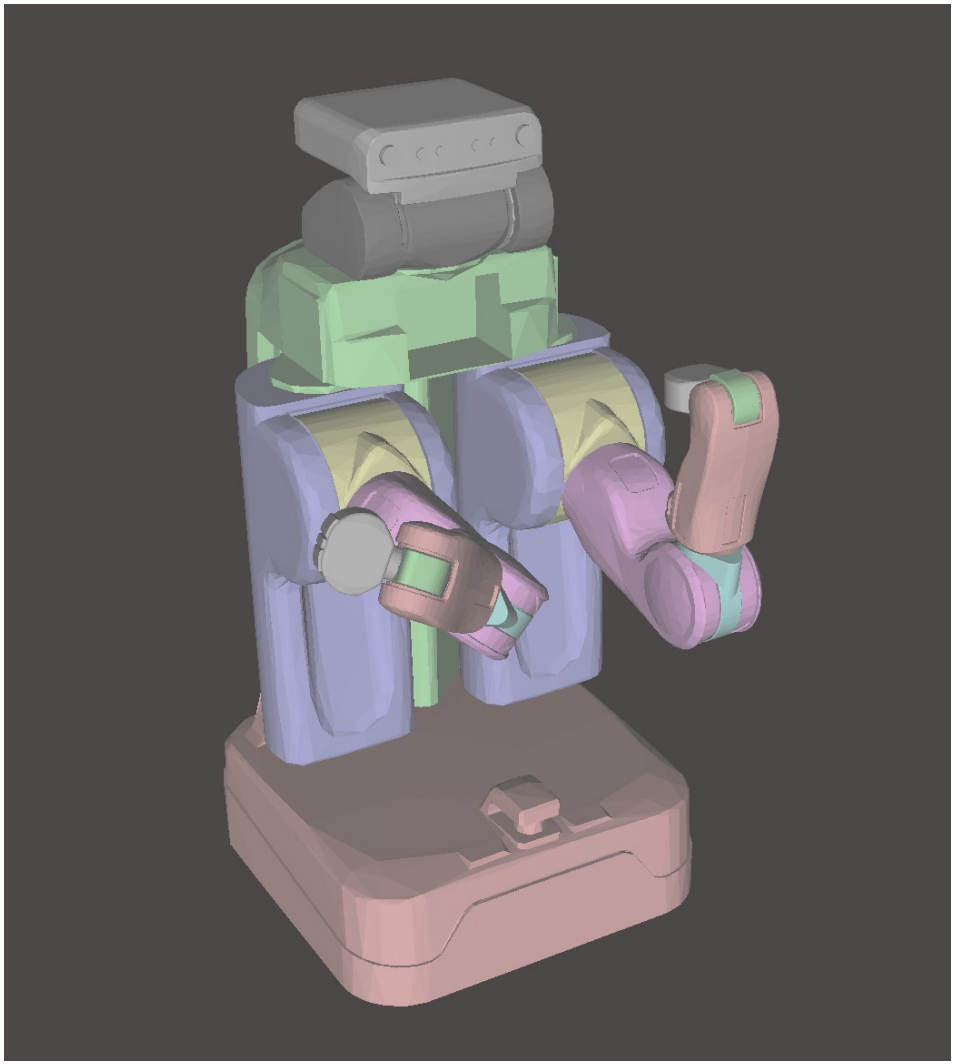


SCL : Multiple Tasks



```
<controller name="opc">
    <type>task</type>
    <must_use_robot>Pr2Bot</must_use_robot>
    <task name="hand">
        <type>TaskOpPos</type>
        <parent_link>l_wrist_roll_link</parent_link>
        <pos_in_parent>0.1 0.0 0.0</pos_in_parent>
        <priority>0</priority>
        <task_dof>3</task_dof>
        <kp>100</kp>
        <kv>25</kv>
        <ka>0</ka>
        <force_max>3000</force_max>
        <force_min>-3000</force_min>
    </task>

    <!--Activate this by passing its name as a second argument on
        the command line -->
    <task name="hand2">
        <type>TaskOpPos</type>
        <parent_link>r_wrist_roll_link</parent_link>
        <pos_in_parent>0.1 0.0 0.0</pos_in_parent>
        <priority>0</priority>
        <task_dof>3</task_dof>
        <kp>100</kp>
        <kv>25</kv>
        <ka>0</ka>
        <force_max>3000</force_max>
        <force_min>-3000</force_min>
    </task>

    <task name="GcTask">
        <type>TaskGc</type>
        <priority>1</priority>
        <task_dof>0</task_dof>
        <kp>30</kp>
        <kv>6</kv>
        <ka>0</ka>
        <force_max>3000</force_max>
        <force_min>-3000</force_min>
    </task>
</controller>
```

SCL : Pr2 w/ Equal Priority Hand Tasks



SCL : Multiple Prioritized Tasks

```
<task name="hand2-pri-high">
    <type>TaskOpPos</type>
    <parent_link>r_forearm_roll_link</parent_link>
    <pos_in_parent>0.1 0.0 0.0</pos_in_parent>
    <priority>0</priority>
    <task_dof>3</task_dof>
    <kp>100</kp>
    <kv>25</kv>
    <ka>0</ka>
    <force_max>3000</force_max>
    <force_min>-3000</force_min>
</task>

<!--Activate this by passing its name as a second argument on
    the command line -->
<task name="hand2-pri-low">
    <type>TaskOpPos</type>
    <parent_link>r_wrist_roll_link</parent_link>
    <pos_in_parent>0.1 0.0 0.0</pos_in_parent>
    <priority>1</priority>
    <task_dof>3</task_dof>
    <kp>100</kp>
    <kv>25</kv>
    <ka>0</ka>
    <force_max>3000</force_max>
    <force_min>-3000</force_min>
</task>
```

SCL : Multiple Tasks

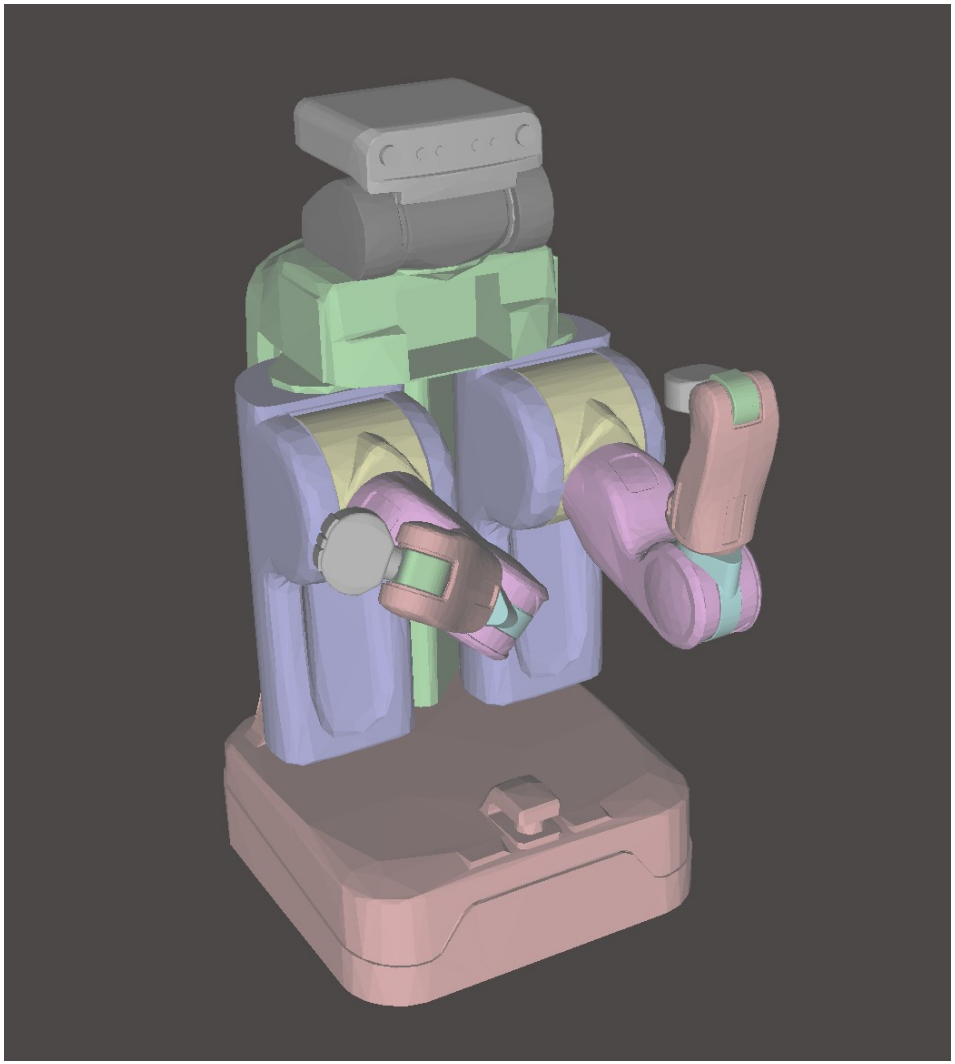
```
<task name="hand2-pri-high">
    <type>TaskOpPos</type>
    <parent_link>r_forearm_roll_link</parent_link>
    <pos_in_parent>0.1 0.0 0.0</pos_in_parent>
    <priority>0</priority>
    <task_dof>3</task_dof>
    <kp>100</kp>
    <kv>25</kv>
    <ka>0</ka>
    <force_max>3000</force_max>
    <force_min>-3000</force_min>
</task>

<!--Activate this by passing its name as a second argument on
    the command line -->
<task name="hand2-pri-low">
    <type>TaskOpPos</type>
    <parent_link>r_wrist_roll_link</parent_link>
    <pos_in_parent>0.1 0.0 0.0</pos_in_parent>
    <priority>1</priority>
    <task_dof>3</task_dof>
    <kp>100</kp>
    <kv>25</kv>
    <ka>0</ka>
    <force_max>3000</force_max>
    <force_min>-3000</force_min>
</task>
```

SCL : Multiple Tasks

```
<task name="hand2-pri-high">
    <type>TaskOpPos</type>
    <parent_link>r_forearm_roll_link</parent_link>
    <pos_in_parent>0.1 0.0 0.0</pos_in_parent>
    <priority>0</priority>
    <task_dof>3</task_dof>
    <kp>100</kp>
    <kv>25</kv>
    <ka>0</ka>
    <force_max>3000</force_max>
    <force_min>-3000</force_min>
</task>

<!--Activate this by passing its name as a second argument on
    the command line -->
<task name="hand2-pri-low">
    <type>TaskOpPos</type>
    <parent_link>r_wrist_roll_link</parent_link>
    <pos_in_parent>0.1 0.0 0.0</pos_in_parent>
    <priority>1</priority>
    <task_dof>3</task_dof>
    <kp>100</kp>
    <kv>25</kv>
    <ka>0</ka>
    <force_max>3000</force_max>
    <force_min>-3000</force_min>
</task>
```

- What motion do you predict?

SCL : Pr2 w/ Prioritized Hand Tasks



Final lessons

- If you think you have a bug in your controller
  - Look at the motion
  - Associate it with a specific type of error
  - Go fix the code/robot specification

- Goal of this class :
  - You should be able to look at a robot's motion and state what is wrong with it..
  - You're on the path to being a roboticist!

More Controller Twiddling?

Groups introduce themselves!!!