Dymanics and Control

Steven Dirven, Massey University, 2015

Reflection on Forward and Inverse Kinematics

So far, the following concepts have been discussed:

1. Assignment of Denavit-Hartenberg frames, and recording of the parameters $\theta_j, d_j, a_j, \alpha_j$ into a table for each joint
2. Forward Kinematics is the process of finding a robot's pose (location and orientation in space) based on the specified configuration of each joint
3. Inverse Kinematics is the process of finding the joint configurations (angles for revolute, length for prismatic) based a robot's specified pose

Reflection on Trajectory Generation

4. Trajectory Generation can be achieved by a number of methods. We visited traditional methods of:

- **Cartesian space** trajectory generation.
  - The pose matrix T was interpolated from the initial to final pose
  - The resulting joint angles were found by **inverse** kinematics
- **Joint space** trajectory generation.
  - The q joint vector was interpolated from the initial to final configurations
  - The resulting poses were found by **forward** kinematics

Reflection on Velocity Relationships

5. A series of tools were investigating velocity within a frame, and then between frames have been investigated:

- Developing relationships between linear and rotational velocities and their effect on the end-effector
- Discovering the Jacobian matrix to transform velocity between frames
- Answered, how do small variations in joint angles with respect to time affect the pose of the end-effector?

Dynamics : Introduction

When a serial robot moves effects such as gravity, friction, and inertia need to be considered.

Dynamics is concerned with describing motion of the robot, and the relationships between velocity, acceleration, mass distribution, forces, and torques.

It is modelled by a linear sum of matrices that correspond to relationships between these effects.

So how does it relate to what has been investigated in earlier lectures?

Dynamics : Relationship to Previous Lectures

So far, tools towards investigating robotic motion have included:

1. Forward and inverse kinematics
   - Static investigation of relationship between joint angles and robot pose
2. Trajectory generation
   - Continuously describing motion along a path
3. Velocity relationships
   - Using the Jacobian to investigate propagation of velocity and forces through the robot

Dynamics is concerned with motion of the robot, and integrates each of the above elements.

Dynamics : Equations of Motion Introduction

Equations of Motion

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T g$$

where $\ddot{q}, \dot{q}$, and q are angular acceleration, velocity and position respectively, and:

Q = Vector of generalised actuator assertion in q
M = Joint space inertia matrix
C = Coriolis and centripetal coupling matrix
F = Friction force matrix
G = Gravity loading matrix

Each element of this system of equations will be investigated independently.

Dynamics : Breakdown of the Technique

$$Q = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T g$$

The equations of motion in this form can be used to find Q, the joint assertion magnitude (torque for each of the revolute joints).

This is based on the pose, velocity and acceleration.

This series of matrices will be broken down to look at the individual effects.

Dynamics : Investigating Contribution of Individual Terms : Inertia

$$Q = \boxed{M(q)\ddot{q}} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T g$$

Why is inertia important to consider in serial robots?

- Once actuators are selected for a robot, the maximum joint torque is fixed.
- Inertia is the resistance of mass to undertake motion, and limits the upper bound on acceleration
- Path following accuracy may be hindered if the robot cannot achieve sufficient acceleration

Dynamics : Investigating Contribution of Individual Terms : Inertia

$$Q = \boxed{M(q)\ddot{q}} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T g$$

The inertia matrix M is a function of the manipulator configuration q.

Inertia Matrix : PUMA PCT MATLAB Example

```
mdl_puma560
M = p560.inertia(qn)
```

```
M =
    3.6594   -0.4044    0.1006   -0.0025    0.0000   -0.0000
   -0.4044    4.4137    0.3509    0.0000    0.0024    0.0000
    0.1006    0.3509    0.9378    0.0000    0.0015    0.0000
   -0.0025    0.0000    0.0000    0.1925    0.0000    0.0000
    0.0000    0.0024    0.0015    0.0000    0.1713    0.0000
   -0.0000    0.0000    0.0000    0.0000    0.0000    0.1941
```

Dynamics : Investigating Contribution of Individual Terms : Inertia

```
M =
    3.6594   -0.4044    0.1006   -0.0025    0.0000   -0.0000
   -0.4044    4.4137    0.3509    0.0000    0.0024    0.0000
    0.1006    0.3509    0.9378    0.0000    0.0015    0.0000
   -0.0025    0.0000    0.0000    0.1925    0.0000    0.0000
    0.0000    0.0024    0.0015    0.0000    0.1713    0.0000

   -0.0000    0.0000    0.0000    0.0000    0.0000    0.1941
```

Observations of the inertia matrix:

- The inertia matrix is symmetric (eg. $M_{ij} = M_{ji}$).
- Diagonal elements describe inertia observed at the joint (eg. $Q_j = M_{jj}\ddot{q}_j$)
- Diagonal elements up the top left are large due to heavy upper and lower arm links
- Off-diagonal terms M_{ij} where ij represent coupling from joint j to generalised force on joint i

Dynamics : Investigating Contribution of Individual Terms : Coriolis Matrix

$$Q = M(q)\ddot{q} + \boxed{C(q,\dot{q})\dot{q}} + F(\dot{q}) + G(q) + J(q)^T g$$

What are centripetal and coriolis torques?

- Centripetal torques act in the axis of rotation, and are proportional to $\dot{q}_i^2$.
- Coriolis torques act perpendicular to the axis of rotation and the direction of motion (right-handed cross product) and are proportional to $\dot{q}_i\dot{q}_j$

Dynamics : Investigating Contribution of Individual Terms : Coriolis Matrix

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T g$$

The coriolis matrix C is a function of the manipulator configuration q, and rotational velocity $\dot{q}$.

Coriolis Matrix : PUMA PCT MATLAB Example

```
mdl_puma560
qd = 0.5*[1 1 1 1 1 1]
C = p560.coriolis(qn,qd)
```

```
C =
   -0.1335   -0.6453    0.0848   -0.0002   -0.0014    0.0000
    0.3137    0.1929    0.3857   -0.0008   -0.0001   -0.0000
   -0.1804   -0.1933   -0.0005   -0.0005   -0.0014   -0.0000
    0.0002    0.0003   -0.0000    0.0001    0.0001   -0.0000
   -0.0001    0.0005    0.0009   -0.0001   -0.0000   -0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000         0
```

Dynamics : Investigating Contribution of Individual Terms : Coriolis Matrix

```
C =
   -0.1335   -0.6453    0.0848   -0.0002   -0.0014    0.0000
    0.3137    0.1929    0.3857   -0.0008   -0.0001   -0.0000
   -0.1804   -0.1933   -0.0005   -0.0005   -0.0014   -0.0000
    0.0002    0.0003   -0.0000    0.0001    0.0001   -0.0000
   -0.0001    0.0005    0.0009   -0.0001   -0.0000   -0.0000

    0.0000    0.0000    0.0000    0.0000    0.0000         0
```

Observations of the coriolis matrix:

- The off-diagonal terms represent coupling of joint j velocity to the generalised force acting on joint i

Multiplying C by $\dot{q}$ yields torque components of:

```
Q_C =

  -0.3478    0.4457   -0.1880    0.0003    0.0006    0.0000
```

Which are the torque contributions to each joint

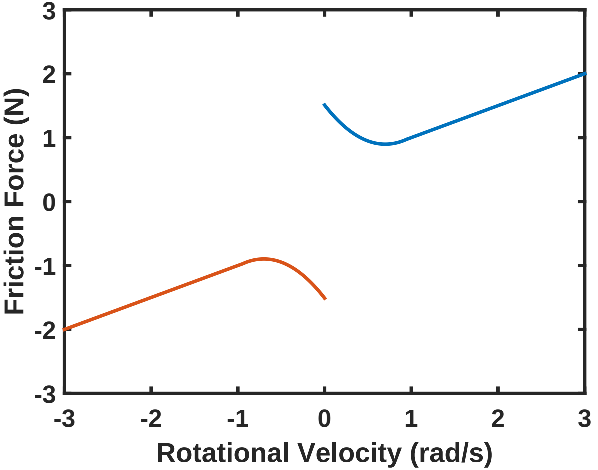
Dynamics : Investigating Contribution of Individual Terms : Friction

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + \boxed{F(\dot{q})} + G(q) + J(q)^T g$$

Why is friction important to consider in serial robots?

- Produces force, or torque, that opposes motion.
- Friction is an inherent behaviour of actuators
- Transmission or drive train systems typically also exhibit friction
- For most electrical drive robots, friction is the second most dominant torque sink, after gravity

Dynamics : Investigating Contribution of Individual Terms : Friction



Dynamics : Investigating Contribution of Individual Terms : Friction

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T g$$

Friction Behaviour : PUMA PCT MATLAB Example : Single Joint (eg. joint 2)

```
mdl_puma560
p560.links(2).dyn
```

```
theta=q, d= 0, a= 0.4318, alpha= 0, offset= 0 (R,stdDH)
m    =        17.4
r    =     -0.3638       0.006      0.2275
I    = |      0.13           0           0 |
       |         0       0.524           0 |
       |         0           0       0.539 |
Jm   =      0.0002
Bm   =    0.000817
Tc   =       0.126(+)      -0.071(-)
G    =       107.8
qlim = -0.785398 to 3.926991
```

Dynamics : Investigating Contribution of Individual Terms : Friction

The bottom elements, apart from the joint angle limits correspond to the friction values.

```
Bm   =    0.000817
Tc   =       0.126(+)      -0.071(-)
G    =       107.8
```

Bm is the viscous friction gradient
Tc is the Coulomb friction offset
G is the Gear ratio

The relationship between the friction parameters is as follows: $Q_f = Bm\dot{q} + Tc$

Dynamics : Investigating Contribution of Individual Terms : Gravity

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T g$$

How does gravity affect the motion of serial robots, and how can this be overcome?

- Gravity is required to be overcome for static poses, as well as dynamically-moving behaviour
- The greater horizontal robot reach from the base, the more base torque experienced
- Sometimes counter-balances or springs are used to aid robot joints

Dynamics : Investigating Contribution of Individual Terms : Gravity

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T g$$

The gravity contribution to joint torques is a function of the joint angles, as well as the mass distributed throughout the links.

Gravity Torque Vector : PUMA PCT MATLAB Example

```
mdl_puma560
grav = p560.gravity
gravload = p560.gravload(qn)
```

```
grav =
         0           0      9.8100

gravload =
  -0.0000   31.6399   6.0351   0.0000   0.0283         0
```

Dynamics : Investigating Contribution of Individual Terms : Gravity

```
grav =
         0           0      9.8100

gravload =
  -0.0000   31.6399   6.0351   0.0000   0.0283        0
```

Gravity is 9.81 m/s vertically downward

The magnitudes in the first couple of joints, perpendicular to gravity, exhibit considerable torque (eg. 31.6399 Nm)

The torque exerted on a joint is much larger when the arm is stretched out horizontally, more so than vertically (where there is only a small moment arm)

Dynamics : Investigating Contribution of Individual Terms : Wrench

$$Q = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T g$$

How does wrench affect serial robots?

- In a previous lecture, the propagation of wrench (force and moment) components throughout the frames of a serial robot were investigated.
- Wrench, in this context is transformed back into the joints to find the required joint torques to maintain the pose (similar to the static case)
- The method is similar to transforming velocities, though uses the transposed Jacobian

Dynamics : Investigating Contribution of Individual Terms : Wrench

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T g$$

- The Jacobian was investigated for transforming velocities between frames. If this Jacobian is transposed, it can be used to transform external wrenches into joint torques.
- The vector g has elements of force in X,Y,Z and moments about the three axes as well.

Dynamics : First Equation Summary

$$Q = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T g$$

The torque vector Q records the torques required in the individual joints and consists of contributions from :

M = Joint space inertia
C = Coriolis and centripetal coupling
F = Friction force
G = Gravity loading
$J(q)^T g$ = External wrench

In order to solve this equation in MATLAB a recursive Newton-Euler numerical method is used.

Dynamics : Implementation in MATLAB

The recursive Newton-Euler method starts at the robot's base. It adds the velocity/acceleration components towards the end-effector.

Then, it works back towards the base - finding how the forces and moments propagate through the robot in order to find the joint torques.

Finding Q by Recursive Newton-Euler Method : PUMA PCT MATLAB Example

```
Syntax:
Q = p560.rne(position,velocity,acceleration)

mdl_puma560
Q = p560.rne(qn,qz,qz)
```

eg. nominal pose, zero velocity and zero acceleration

Dynamics : Implementation in MATLAB

The code from the previous slide under the conditions of nominal pose, zero velocity and zero acceleration yields a torque requirement of:

```
Q′ =   -0.0000   31.6399    6.0351    0.0000    0.0283         0
```

What happens if joint 1 was moving with a rotational velocity of 1 rad/s? To observe this component independently, gravity will be turned off for this

Finding Q by Recursive Newton-Euler Method : PUMA PCT MATLAB Example : Joint 1 at 1 rad/s with no Gravity

```
Q = p560.rne(qn,[1 0 0 0 0 0],qz,[0 0 0]’)
```

```
Q′ =   30.5332    0.6280   -0.3607   -0.0003   -0.0000         0
```

Of course, these vectors need to be added together.

Dynamics : Effect of Payload

Real robots have a maximum specified payload dictated by 2 dynamic effects:

1. Mass at the end of the robot will increase the inertia observed by the joints, which reduces acceleration and dynamic performance
2. mass generates a weight force, which the joints need to be able to support.

eg. a PUMA robot has a maximum payload of 2.5kg. What happens if a 2.5kg point-mass load is added to the robot in simulation?

Dynamics : Effect of Payload

Inertia Before and After Adding 2.5kg Point-mass Load on a 0.1m Link in Z : PUMA PCT MATLAB Example

```
mdl_puma560
M_before = p560.inertia(qn)
p560.payload(2.5,[0 0 0.1])
M_after = p560.inertia(qn)
```

```
M_before =
    3.6594   -0.4044    0.1006   -0.0025    0.0000   -0.0000
   -0.4044    4.4137    0.3509    0.0000    0.0024    0.0000
    0.1006    0.3509    0.9378    0.0000    0.0015    0.0000
   -0.0025    0.0000    0.0000    0.1925    0.0000    0.0000
    0.0000    0.0024    0.0015    0.0000    0.1713    0.0000
   -0.0000    0.0000    0.0000    0.0000    0.0000    0.1941

M_after =
    4.8902   -0.3992    0.2162   -0.1243    0.0000   -0.0000
   -0.3992    5.5908    1.0243    0.0000    0.1746    0.0000
    0.2162    1.0243    1.5569   -0.0000    0.0983    0.0000
   -0.1243    0.0000   -0.0000    0.2050    0.0000    0.0000
    0.0000    0.1746    0.0983    0.0000    0.1963    0.0000
   -0.0000    0.0000    0.0000    0.0000    0.0000    0.1941
```

Dynamics : Effect of Payload

```
M_before =
    3.6594   -0.4044    0.1006   -0.0025    0.0000   -0.0000
   -0.4044    4.4137    0.3509    0.0000    0.0024    0.0000
    0.1006    0.3509    0.9378    0.0000    0.0015    0.0000
   -0.0025    0.0000    0.0000    0.1925    0.0000    0.0000
    0.0000    0.0024    0.0015    0.0000    0.1713    0.0000
   -0.0000    0.0000    0.0000    0.0000    0.0000    0.1941

M_after =
    4.8902   -0.3992    0.2162   -0.1243    0.0000   -0.0000
   -0.3992    5.5908    1.0243    0.0000    0.1746    0.0000
    0.2162    1.0243    1.5569   -0.0000    0.0983    0.0000
   -0.1243    0.0000   -0.0000    0.2050    0.0000    0.0000
    0.0000    0.1746    0.0983    0.0000    0.1963    0.0000
   -0.0000    0.0000    0.0000    0.0000    0.0000    0.1941
```

Recall that inertia matrix M is a symmetric matrix.

Can see that the diagonal elements (Observed inertia) have increased significantly.

Dynamics : Base wrench (forces and moments)

A moving robot exerts a wrench at its base:

- Vertical and horizontal forces to maintain its position
- Moments / torques as the arm moves around.

Base forces are especially important in situations where the robot does not have a rigid base.

Eg. On a satellite, boat, submersible, or vehicle with soft suspension

Finding Wrench g at the Robot's Base

```
[Q,g] = p560.rne(qn,qz,qz)
```

Forward Dynamics : Calculating Motion From Known Internal Forces and Torques

The aforementioned equation:

$$Q = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + F(\dot{q}) + G(q) + J(q)^T g$$

Can be rearranged for $\ddot{q}$ in the following manner:

$$M(q)\ddot{q} = Q - C(q, \dot{q})\dot{q} - F(\dot{q}) - G(q)$$

$J(q)^T g$ is ignored as the external wrench can be subtracted later. Then,

$$\ddot{q} = M(q)^{-1}(Q - C(q, \dot{q})\dot{q} - F(\dot{q}) - G(q))$$