Neuromechanics of Human Motion

Limb Dynamics

Joshua Cashaback, PhD

Recap — Kinematics

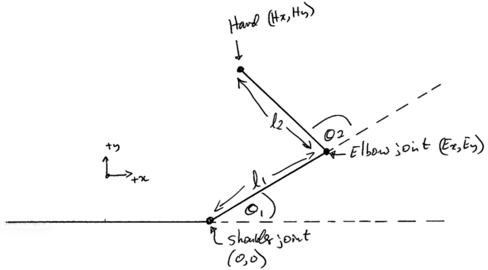
Forward Kinematics

Go from intrinsic variable (joint space) to extrinsic variables (hand space)

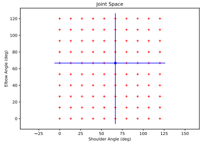
Inverse Kinematics

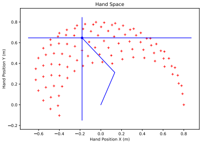
Go from extrinsic variables (hand space) to intrinsic variables (joint space)

Recap — Kinematics



Schematic of a simple kinematic model of a two-joint arm

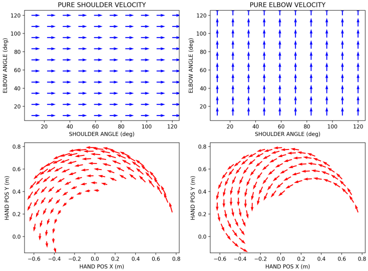




$$H_x = l_1 cos(\theta_1) + l_2 cos(\theta_1 + \theta_2)$$

$$H_y = l_1 sin(\theta_1) + l_2 sin(\theta_1 + \theta_2)$$

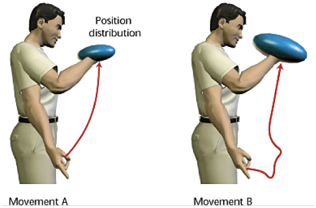
Recap — Kinematics

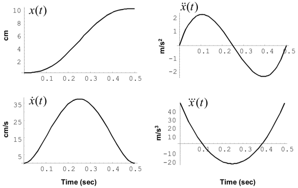


$$\dot{H} = J(\theta) \cdot \dot{\theta}$$

$$\ddot{H} = \dot{J}(\theta) \cdot \dot{\theta} + J(\theta) \cdot \ddot{\theta}$$

Recap — Kinematics





Lecture Objectives — Limb Dynamics

1. Lagrange Equations
   - Potential Energy
   - Kinetic Energy
2. 1DOF and 2DOF
3. Forward and Inverse Dynamics

How to Derive the Equations of Motion?

Many Different Ways

1. Newtonian Mechanics
2. Hamilton Mechanics
3. Kane Mechanics
4. Lagrange Equations

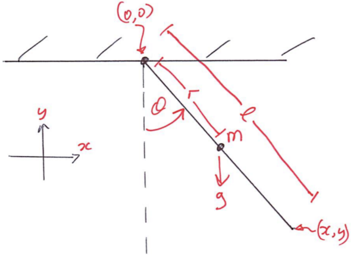
How to Derive Equations of Motion?

Many Different Ways

1. Newtonian Mechanics
2. Hamilton Mechanics
3. Kane Mechanics
4. Lagrange Mechanics
   - Energy Approach
   - Advantages: i. Ease with complex problems, ii. Any coordinate reference frame

1-Link Arm

1-Link Arm



Schematic of a simple one-joint arm in a vertical plane

$m(1.65kg)$ = mass; $l(1.0m)$ =rod length; $r(0.5m)$ = distance of mass from the origin (point the mass rotates about); $g(9.81m/s^2)$ = force of gravity; $\mathcal{I}(0.025kgm^2)$ = moment of inertia; $\theta(rad)$ = angle between negative y-axis and link

Setting up the Lagrange-Euler Equation

To Define the Equations of Motion we need to:

1. Define the Energy in the System
   - Potential
   - Kinetic
2. Define the kinematics of the system
3. Take derivatives based on the defined energy and kinematics

The Lagrange-Euler Equation

The Lagrange

$$L = T - U$$

The Lagrange-Euler Equation:

$$Q_j = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_j}\right) - \frac{\partial L}{\partial q_j}$$

L: Lagrangian

T: Kinetic Energy

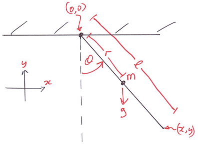
U: Potential Energy

q_j: some "generalized coordinate" (e.g., θ)

$_j$: index of some generalized coordinate

Q_j: some "generalized force" (e.g., torque)

Kinetic Energy

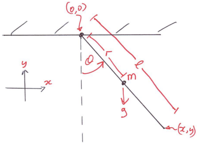


Schematic of a simple one-joint arm in a vertical plane

$$T = T^{rot} + T^{lin}$$

1. Rotational Kinetic Energy (T^{rot})
   . kinetic energy related to the rotation of the rod
2. Linear Kinetic Energy (T^{lin})
   . kinetic energy related to the movement of the centre of mass, m

Rotational Kinetic Energy

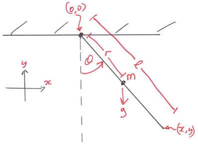


Schematic of a simple one-joint arm in a vertical plane

$$T^{rot} = \frac{1}{2}\mathcal{I}\dot{\theta}^2$$

If you are interested in the moment of inertia of different limbs, check out: Biomechanics and Motor Control of Human Movement: Second Edition (1990) by David A. Winter

Linear Kinetic Energy



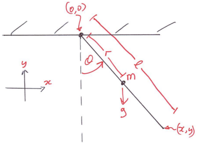
Schematic of a simple one-joint arm in a vertical plane

$$T^{lin} = \frac{1}{2}mv^2$$

$$T^{lin} = \frac{1}{2}m(\dot{x}^2 + \dot{y}^2)$$

$$T^{lin} = \frac{1}{2}m\dot{x}^2 + \frac{1}{2}m\dot{y}^2$$

Linear Kinetic Energy



Schematic of a simple one-joint arm in a vertical plane

$$T^{lin} = \frac{1}{2}mv^2$$

$$T^{lin} = \frac{1}{2}m(\dot{x}^2 + \dot{y}^2)$$

$$T^{lin} = \frac{1}{2}m\dot{x}^2 + \frac{1}{2}m\dot{y}^2$$

But, we need to express x and y in terms of generalized coordinates (i.e., θ)!

Linear Kinetic Energy

$$T^{lin} = \frac{1}{2}m\dot{x}^2 + \frac{1}{2}m\dot{y}^2 (Eq.1)$$

Lets express x and y in our generalized (polar) coordinates.

$$x = rsin(\theta); y = -rcos(\theta)$$

To calculate the linear kinetic energy, we need to take the time derivative of these terms (i.e., $\dot{x}$ and $\dot{y}$) to substitute into **Eq. 1**.

$$\frac{dx}{dt} = \frac{d(rsin(\theta))}{dt}; \frac{dy}{dt} = \frac{d(-rcos(\theta))}{dt}$$

Using the chain rule $(e.g., \frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx})$ for functions of the form, $z(y(x))$:

$$\dot{x} = rcos(\theta)\dot{\theta}; \dot{y} = rsin(\theta)\dot{\theta}$$

Linear Kinetic Energy

Substituting

$$\dot{x} = rcos(\theta)\dot{\theta}; \dot{y} = rsin(\theta)\dot{\theta}$$

into

$$T^{lin} = \frac{1}{2}m\dot{x}^2 + \frac{1}{2}m\dot{y}^2$$

gives

$$T^{lin} = \frac{1}{2}m(rcos(\theta)\dot{\theta})^2 + \frac{1}{2}m(rsin(\theta)\dot{\theta})^2.$$

Expanding and some slight rearrangement yields

$$T^{lin} = \frac{1}{2}mr^2\dot{\theta}^2cos^2(\theta) + \frac{1}{2}mr^2\dot{\theta}^2sin^2(\theta).$$

Factoring leads to:

$$T^{lin} = \frac{1}{2}mr^2\dot{\theta}^2(cos^2(\theta) + sin^2(\theta)).$$

A 'commonly known' trig identity is: $cos^2(\theta) + sin^2(\theta) = 1$. Thus,

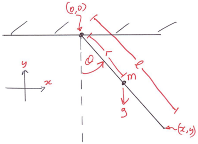
$$T^{lin} = \frac{1}{2}mr^2\dot{\theta}^2$$

Total Kinetic Energy

$$T = T^{rot} + T^{lin}$$

$$T = \frac{1}{2}\mathcal{I}\dot{\theta}^2 + \frac{1}{2}mr^2\dot{\theta}^2$$

Potential Energy (U)



Schematic of a simple one-joint arm in a vertical plane

$$U = mgh$$

where U is the potential energy and h is the height of the mass above the ground. Assuming the ground is defined as the y-axis position when the pendulum is pointed straight down,

$$U = mgr(1 - cos\theta)$$

Lagrangian

Putting all the energy terms together we get the Lagrangian (L)

$$L = T - U$$

$$L = \frac{1}{2}\mathcal{I}\dot{\theta}^2 + \frac{1}{2}mr^2\dot{\theta}^2 - mgr(1 - cos\theta)$$

Next we apply the Euler-Lagrange equation

Euler-Lagrange Equation

$$L = \frac{1}{2}\mathcal{I}\dot{\theta}^2 + \frac{1}{2}mr^2\dot{\theta}^2 - mgr(1 - cos\theta)$$

$$Q_j = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_j}\right) - \frac{\partial L}{\partial \theta_j}$$

Breaking this down

$$\frac{\partial L}{\partial \theta_j} = -mgrsin(\theta)$$

$$\frac{\partial L}{\partial \dot{\theta}_j} = \dot{\theta}(mr^2 + \mathcal{I})$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_j}\right) = \ddot{\theta}(mr^2 + \mathcal{I})$$

Thus,

$$Q_j = \ddot{\theta}(mr^2 + \mathcal{I}) + mgrsin(\theta)$$

1-Link Lagrange Equation with Sympy — Python Code

```
from sympy import *
# 1-Link Lagrange-Euler Derivation (rotation coor. frame relative to negative y-axis)
m,r,i,l,a,t,g = symbols('m r I l a t g')
theta  = Function('theta')(t)
# define x in general coordinates
x = r * sin(theta)
y = -r * cos(theta)
xd = diff(x,t)
yd = diff(y,t)
Tlin = 0.5 * m * ((xd*xd) + (yd*yd))
Tlin = simplify(Tlin)
ad = diff(theta,t)
Trot = 0.5 * I * ad ** 2
T = Tlin + Trot
T = simplify(T)
U = m * g * r * (1-cos(theta))
L = T - U
L = simplify(L)
Q = diff(diff(L,diff(theta)),t) - diff(L,theta)
pprint(Q)
```

Inverse Dynamics

Inverse Dynamics

Relies on the motion of the subject and a body model to compute the forces that were necessary to produce this movement.

$$Q_j = \ddot{\theta}(mr^2 + \mathcal{I}) + mgrsin(\theta)$$

Forward Dynamics

Forward Dynamics

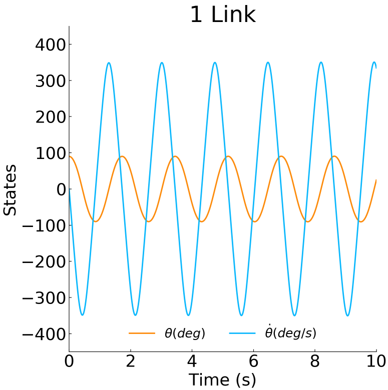
Uses joint torques/forces to predict resultant motions.

$$\ddot{\theta} = \frac{Q_j - mgrsin(\theta)}{(mr^2 + \mathcal{I})}$$

Note that if the torque Q is zero, in other words if there is no **input moment (e.g., from muscle)** to the system:

$$\ddot{\theta} = \frac{mgrsin(\theta)}{(mr^2 + \mathcal{I})}$$

1-Link Arm — Forward Simulation

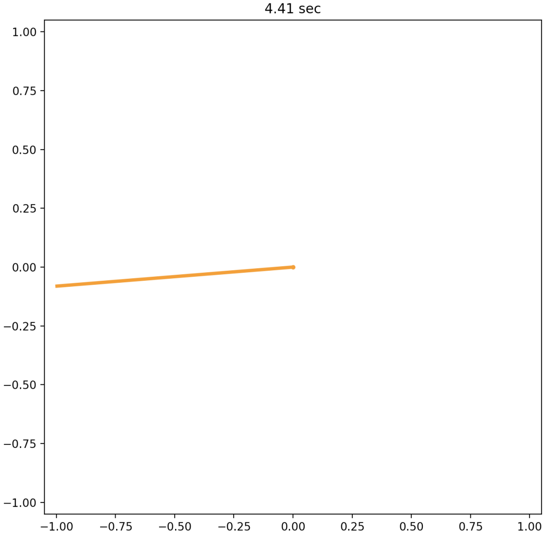


IC: $\theta = 90; \dot{\theta} = 0$; constants listed on previous slide

*Convert 2nd-order system to two, 1st-order ODEs

(refer to ODE lecture)

1-Link Arm — Animation

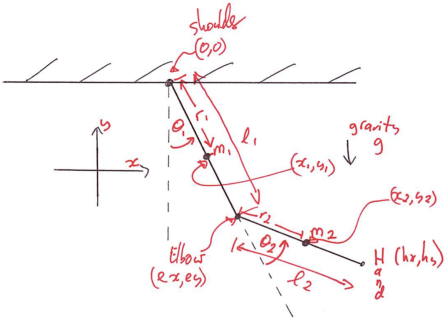


1-Link Arm Animation — Python Code

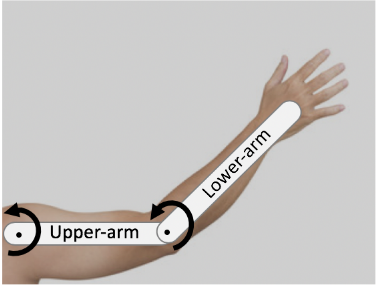
```
def animate_arm(state,t):
  l = 1.0
  figure = plt.subplots(figsize=(8,8))
  plot(0,0,'r.', color = '#FD8B0B')
  p, = plot((0,l*math.sin(THETA[0])),(0,-l*math.cos(THETA[0])),'-', color = '#FD8B0B')
  dt = t[1]-t[0]
  tt = title("")
  xlim([-l-.05,l+.05])
  ylim([-l-.05,l+.05])
  step = 100
  for i in range(0,len(THETA)-step,step):
    p.set_xdata((0,l*math.sin(THETA[i])))
    p.set_ydata((0,-l*math.cos(THETA[i])))
    tt.set_text("%4.2f sec" % (i*dt))
    pause(0.001)
    draw()

animate_arm(THETA,TIME)
```

2-Link Arm

2-Link Arm



Schematic of a two-joint arm in a vertical plane



$m_1(2.1), m_2(1.65), \mathcal{I}_1(0.025), \mathcal{I}_2(0.075), l_1(0.3384), l_2(0.4554)$
$r_1(0.1692), r_2(0.2277), g(9.81)$

The Lagrange-Euler Equation

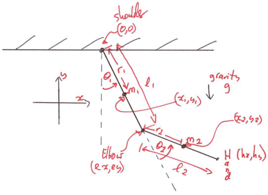
$$Q_j = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_j}\right) - \frac{\partial L}{\partial q_j}$$

1. q_j: here there are 2 "generalized coordinates" (θ_1 and θ_2)
2. Q_j: 2 "generalized forces" (shoulder and elbow moments)

$$Q_1 = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_1}\right) - \frac{\partial L}{\partial \theta_1}$$

$$Q_2 = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_2}\right) - \frac{\partial L}{\partial \theta_2}$$

The Langrangian



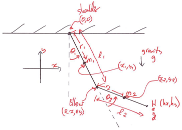
Schematic of a two-joint arm in a vertical plane

$$L = T - U$$

$$L = T^{rot} + T^{lin} - U$$

$$L = T_1^{rot} + T_2^{rot} + T_1^{lin} + T_2^{lin} - U_1 - U_2$$

Rotational Kinetic Energy



Schematic of a two-joint arm in a vertical plane

Generally,

$$T_j^{rot} = \frac{1}{2}\mathcal{I}_j\dot{\theta_j}^2$$

And for each generalized coordinate

$$T_1^{rot} = \frac{1}{2}\mathcal{I}_1\dot{\theta_1}^2$$

$$T_2^{rot} = \frac{1}{2}\mathcal{I}_2(\dot{\theta_1} + \dot{\theta_2})^2$$

Linear Kinetic Energy

Generally,

$$T_j^{lin} = \frac{1}{2} m_j \dot{v_j}^2$$

$$T_j^{lin} = \frac{1}{2} m_j (\dot{x_j}^2 + \dot{y_j}^2)$$

And for each generalized coordinate

$$T_1^{lin} = \frac{1}{2} m_1 (\dot{x_1}^2 + \dot{y_1}^2)$$

$$T_2^{lin} = \frac{1}{2} m_2 (\dot{x_2}^2 + \dot{y_2}^2)$$

Linear Kinetic Energy — Generalized Coordinates

Transforming cartesian coordinates (x_j, y_j) into generalized coordinates (θ_j) based on the link geometry

$$x_1 = r_1 sin(\theta_1)$$
$$y_1 = -r_1 cos(\theta_1)$$
$$x_2 = l_1 sin(\theta_1) + r_2 sin(\theta_1 + \theta_2)$$
$$y_2 = -l_1 cos(\theta_1) - r_2 cos(\theta_1 + \theta_2)$$

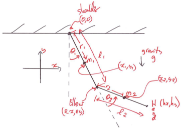
Linear Kinetic Energy — Generalized Coordinates

Transforming cartesian coordinates (x_j, y_j) into generalized coordinates (θ_j) based on the link geometry

$$x_1 = r_1 sin(\theta_1)$$
$$y_1 = -r_1 cos(\theta_1)$$
$$x_2 = l_1 sin(\theta_1) + r_2 sin(\theta_1 + \theta_2)$$
$$y_2 = -l_1 cos(\theta_1) - r_2 cos(\theta_1 + \theta_2)$$

Further below, we will find $\dot{x}$ and $\dot{y}$ in sympy

Potential Energy (U)


Schematic of a two-joint arm in a vertical plane

Generally,

$$U_j = m_j g h_j$$

And for each generalized coordinate

$$U_1 = m_1 g r_1 (1 - cos(\theta_1))$$

$$U_2 = m_2 g [l_1 (1 - cos(\theta_1)) + r_2 (1 - cos(\theta_1 + \theta_2))]$$

Summary

$$L = T_1^{rot} + T_2^{rot} + T_1^{lin} + T_2^{lin} - U_1 - U_2$$

Summary

$$L = T_1^{rot} + T_2^{rot} + T_1^{lin} + T_2^{lin} - U_1 - U_2$$

Shoulder Moment

$$Q_1 = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_1}\right) - \frac{\partial L}{\partial \theta_1}$$

Summary

$$L = T_1^{rot} + T_2^{rot} + T_1^{lin} + T_2^{lin} - U_1 - U_2$$

Shoulder Moment

$$Q_1 = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_1}\right) - \frac{\partial L}{\partial \theta_1}$$

Elbow Moment

$$Q_2 = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_2}\right) - \frac{\partial L}{\partial \theta_2}$$

Summary

$$L = T_1^{rot} + T_2^{rot} + T_1^{lin} + T_2^{lin} - U_1 - U_2$$

Shoulder Moment

$$Q_1 = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_1}\right) - \frac{\partial L}{\partial \theta_1}$$

Elbow Moment

$$Q_2 = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_2}\right) - \frac{\partial L}{\partial \theta_2}$$

Lets make our lives a bit easier and carry this out in Sympy

2-Link Equation of Motion — Python I

```
# ( rotation coor . frame relative to negative y-axis )
m1, m2, I1 , I2 , r1 , r2 , l1 , g , t = symbols('m1 m2 I1 I2 r1 r2 l1 g t')
theta , alpha = Function('theta')(t), Function('alpha')(t)
x1=r1*sin(theta)
y1 =- r1 * cos(theta)
x2 = l1 * sin(theta) + r2 * sin(theta + alpha)
y2 =-l1 * cos(theta) - r2 * cos(theta + alpha)
x1d = diff(x1, t)
y1d = diff(y1, t)
x2d = diff(x2, t)
y2d = diff(y2, t)
thetad = diff(theta, t)
alphad = diff(alpha, t)
Tlin1 = 1/2. * m1 * (x1d ** 2 + y1d ** 2) # v**2 = (x**2 + y ** 2)
Tlin2 = 1/2. * m2 * (x2d ** 2 + y2d ** 2)
Trot1 = 1/2. * I1 * (thetad) ** 2
Trot2 = 1/2. * I2 * (thetad + alphad) ** 2
Ttotal = Tlin1 + Tlin2 + Trot1 + Trot2
U1 = r1 * m1 * g * (1 - cos(theta))
U2 = l1 * m2 * g * (1 - cos(theta)) + r2 * m2 * g * (1 - cos(theta + alpha))
Utotal = U1 + U2
L = Ttotal - Utotal
L = simplify(L)

*continued on next slide
```

2-Link Equation of Motion — Python II

```
Q1 = diff(diff(L,diff(theta)),t) - diff(L,theta)
Q2 = diff(diff(L,diff(alpha)),t) - diff(L,alpha)
# converts floats that are really integers to integers , gets rid of ?1.0?
Q1 = simplify ( nsimplify (Q1))
Q2 = simplify ( nsimplify (Q2))
# magic sauce to further simplify with some trigonometric identities
Q1 = Q1.rewrite(exp).expand().powsimp().rewrite(sin).expand()
Q2 = Q2.rewrite(exp).expand().powsimp().rewrite(sin).expand()
# collect derivative terms
Q1 = collect(Q1, Derivative(Derivative(theta,t),t))
Q1 = collect(Q1, Derivative(Derivative(alpha,t),t))
Q2 = collect(Q2, Derivative(Derivative(theta,t),t))
Q2 = collect(Q2, Derivative(Derivative(alpha,t),t))
# collect sin () terms
Q1 = collect(Q1, sin(theta))
Q2 = collect(Q2, sin(alpha))
pprint (Q1)
pprint (Q2)

From this, you'll get a big printout for Q1 and Q2.
```

Shoulder and Elbow Moments

$$
\begin{aligned}
Q_1 =& (\mathcal{I}_1 + \mathcal{I}_2 + m_1 \cdot r_1^2 + m_2(l_1^2 + r_2^2 + 2 \cdot l_1 \cdot r_2 \cdot cos(\theta_2)))\ddot{\theta_1} + \\
& (\mathcal{I}_2 + m_2[r_2^2 + l_1 \cdot r_2 \cdot cos(\theta_2)])\ddot{\theta_2} - \\
& l_1 \cdot m_2 \cdot r_2 \cdot sin(\theta_2) \cdot \dot{\theta_2}^2 - 2 \cdot l_1 \cdot m_2 \cdot r_2 \cdot sin(\theta_2) \cdot \dot{\theta_1} \cdot \dot{\theta_2} + \\
& g \cdot sin(\theta_1) \cdot (m_2 \cdot l_1 + m_1 \cdot r_1) + g \cdot m_2 \cdot r_2 \cdot sin(\theta_1 + \theta_2) \\
Q_2 =& (\mathcal{I}_2 + m_2 \cdot r_2^2)\ddot{\theta_2} + \\
& (\mathcal{I}_2 + m_2[r_2^2 + l_1 \cdot r_2 \cdot cos(\theta_2)])\ddot{\theta_1} + \\
& l_1 \cdot m_2 \cdot r_2 \cdot sin(\theta_2)\dot{\theta_1}^2 + \\
& g \cdot m_2 \cdot r_2 \cdot sin(\theta_1 + \theta_2)
\end{aligned}
$$

Expressing Joint Moment in Matrix Form

We want to express the equations of motion in matrix form,

$$Q = M\ddot{\theta} + C + G$$

so lets start off by write the equations of motion as

$$Q_1 = M_{11}\ddot{\theta}_1 + M_{12}\ddot{\theta}_2 + C_1 + G_1$$

$$Q_2 = M_{21}\ddot{\theta}_1 + M_{22}\ddot{\theta}_2 + C_2 + G_2$$

M represent inertial terms, C represent coriolis-centrifugal terms, and G are gravitational terms

Expressing Joint Moment in Matrix Form

Defining M, C, and G:

$$
\begin{aligned}
M_{11} =& \mathcal{I}_1 + \mathcal{I}_2 + m_1 \cdot r_1^2 + m_2(l_1^2 + r_2^2 + 2 \cdot l_1 \cdot r_2 \cdot cos(\theta_2)) \\
M_{12} =& \mathcal{I}_2 + m_2[r_2^2 + l_1 \cdot r_2 \cdot cos(\theta_2)] \\
M_{21} =& \mathcal{I}_2 + m_2[r_2^2 + l_1 \cdot r_2 \cdot cos(\theta_2)] \\
M_{22} =& \mathcal{I}_2 + m_2 \cdot r_2^2 \\
C_1 =& -l_1 \cdot m_2 \cdot r_2 \cdot sin(\theta_2) \cdot \dot{\theta_2}^2 - 2 \cdot l_1 \cdot m_2 \cdot r_2 \cdot sin(\theta_2) \cdot \dot{\theta_1} \cdot \dot{\theta_2} \\
C_2 =& l_1 \cdot m_2 \cdot r_2 \cdot sin(\theta_2) \dot{\theta_1}^2 \\
G_1 =& g \cdot sin(\theta_1) \cdot (m_2 \cdot l_1 + m_1 \cdot r_1) + g \cdot m_2 \cdot r_2 \cdot sin(\theta_1 + \theta_2) \\
G_2 =& g \cdot m_2 \cdot r_2 \cdot sin(\theta_1 + \theta_2)
\end{aligned}
$$

Expressing Joint Moment in Matrix Form

Defining M, C, and G:

$$
\begin{aligned}
M_{11} =& \mathcal{I}_1 + \mathcal{I}_2 + m_1 \cdot r_1^2 + m_2(l_1^2 + r_2^2 + 2 \cdot l_1 \cdot r_2 \cdot cos(\theta_2)) \\
M_{12} =& M_{21} = \mathcal{I}_2 + m_2[r_2^2 + l_1 \cdot r_2 \cdot cos(\theta_2)] \\
M_{22} =& \mathcal{I}_2 + m_2 \cdot r_2^2 \\
C_1 =& - l_1 \cdot m_2 \cdot r_2 \cdot sin(\theta_2) \cdot \dot{\theta_2}^2 - 2 \cdot l_1 \cdot m_2 \cdot r_2 \cdot sin(\theta_2) \cdot \dot{\theta_1} \cdot \dot{\theta_2} \\
C_2 =& l_1 \cdot m_2 \cdot r_2 \cdot sin(\theta_2) \dot{\theta_1}^2 \\
G_1 =& g \cdot sin(\theta_1) \cdot (m_2 \cdot l_1 + m_1 \cdot r_1) + g \cdot m_2 \cdot r_2 \cdot sin(\theta_1 + \theta_2) \\
G_2 =& g \cdot m_2 \cdot r_2 \cdot sin(\theta_1 + \theta_2)
\end{aligned}
$$

Inverse Dynamics

$$Q = M\ddot{\theta} + C + G$$

where,

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}$$

$$\ddot{\theta} = \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix}$$

$$C = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}$$

$$G = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix}$$

Forward Dynamics

The inverse dynamics equation in matrix form is

$$Q = M\ddot{\theta} + C + G$$

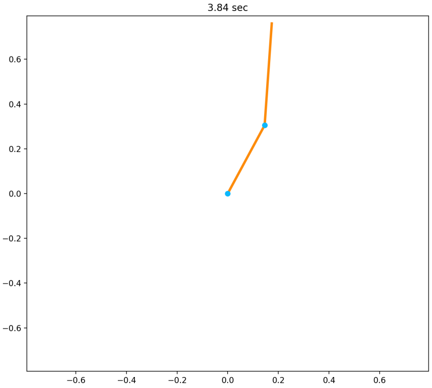
We can rearrange this equation to obtain the following forward dynamics equation:

$$\ddot{\theta} = (M)^{-1}(Q - C - G)$$

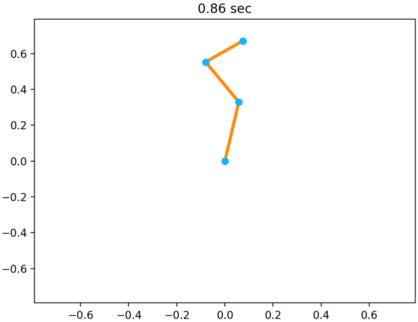
2-Link ODE — Pseudo Code

```
from scipy.integrate import odeint
import numpy as np
def arm_2dof(state, t):
    # unpack the state vector
    theta1, theta2 = state[0], state[1]
    theta1dot, theta2dot = state[2], state[3]
    # INPUT YOUR CONSTANTS HERE (m1, m2, l1, etc)
    # DEFINE YOUR M, C, G, AND Q MATRICES HERE
    # note: set Q matrix to zero but it can also be a time varying vector
    # USE FORWARD DYNAMICS TO CALCULATE ACCELERATION
    # DECOMPOSE YOUR ACCELERATION MATRIX (thetaddot) AS FOLLOWS
    theta1ddot = thetaddot[0,0]
    theta2ddot = thetaddot[1,0]
    # return the state derivatives
    return [theta1dot, theta2dot, theta1ddot, theta2ddot]

# initial conditions
theta1_0, theta2_0 = 180.0 * math.pi / 180.0, 1.0 * math.pi / 180.0 #rads
thetadot1_0, theta2dot_0 = 0.0 * math.pi / 180.0, 0.0 * math.pi / 180.0 #rads
state0 = np.array([theta1_0, theta2_0, thetadot1_0, theta2dot_0])
# time vector
tstart, tend, timestep = 0.0, 10.0, 0.01
t = arange(tstart, tend, timestep)
# differential equations
state = odeint(arm_2dof, state0, t)
```

2-Link Arm — Animation



3-Link Arm — Animation



Does the Brain Care about Dynamics?

Minimize:

- Joint Moments
- Muscle Force (or Activity)
- Energetics
- Time derivatives of these quantities

Does the Brain Care about Dynamics?

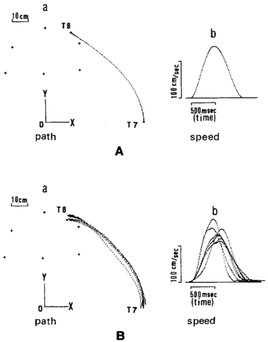


Fig. 4A and B. Large free movements between two targets ($T7 \rightarrow T8$); the starting posture is stretching an arm in the side direction and the end point is approximately in front of the body. **A** Hand trajectory predicted by the minimum torque-change model. **a** shows the path and **b** shows the corresponding speed profile. **B** Observed hand trajectories for the seven subjects. **a** shows the paths and **b** shows the corresponding speed profiles

Uno et al (1989). Biological cybernetics, 61(2), 89-101.

Min[d(joint moment)/dt]

Questions???

Next Class

Internal Models and the Cerebellum

- Brief Overview of Brain Regions
- What is an Internal Model
- Interaction Torques
- Cerebellum Disorders

Assignment 4

See Handout

Acknowledgements

Paul Gribble

Dinant Kistemaker