Biomechanics Gait Analysis Lab

Table of Contents

Summary of Code

```
%This script calculates the joint angles, velocities, and moments of a
%subject throughout his stance phase using motion capture data and a
 force
%plate. These claculations are made for three sets of movement data:
 when
%the subject is walking, fast walking, and jogging.
```

Import the data from the walk trial

```
clear all
```

```
load('normal03.mat')

%Downsample the data
digitaldata = downsample(digitaldata, 10);
COP = downsample(COP, 10);

%Apply Butterworth filter to remove high frequency noise from data
[a,b] = butter(2, 10/50);
[c,d] = butter(2, 5/50);
digitalDataFilt = filtfilt(a,b, digitaldata);
COP_filt = filtfilt(a,b,COP);
markerPositionFilt = filtfilt(c,d, markerposition);

%Ground Reaction Forces
grf = zeros(283,6);
F_M = digitalDataFilt;
grf(:,1) = -F_M(:,1);
grf(:,2) = -F_M(:,2);
grf(:,3) = -F_M(:,3);
grf(:,4) = -F_M(:,4);
grf(:,5) = -F_M(:,5);
grf(:,6) = -F_M(:,6);

%Acquire Center of Pressure in Meters
COP = COP_filt/1000;

%Acquire joint positions from the marker position matrix
RANKLE = markerPositionFilt(:, 1:3)/1000;
RTIP = markerPositionFilt(:, 4:6)/1000;
RHEEL = markerPositionFilt(:, 7:9)/1000;
RKNEE = markerPositionFilt(:, 10:12)/1000;
RTROCH = markerPositionFilt(:, 13:15)/1000;
LANKLE = markerPositionFilt(:, 16:18)/1000;
LTIP = markerPositionFilt(:, 19:21)/1000;
LHEEL = markerPositionFilt(:, 22:24)/1000;
LKNEE = markerPositionFilt(:, 25:27)/1000;
LTROCH = markerPositionFilt(:, 28:30)/1000;
RSHLDR = markerPositionFilt(:, 31:33)/1000;
LSHLDR = markerPositionFilt(:, 34:36)/1000;
RBACK = markerPositionFilt(:, 37:39)/1000;
```

Segment Angles for Walk(deg)

```
%Determine the range of one gait cycle of the subject from visual
%inspection
shift = 98;
finish = 177;
L = finish - shift;
stancePhase = 100.*([0:L-1]./(L-1));

%Initialize the segment angles
theta_foot = zeros(L-1,1);
theta_trunk = zeros(L-1,1);
```

```
theta_shank = zeros(L-1,1);
theta_thigh = zeros(L-1,1);

%Calculate the segment angles of the right leg utilizing the positions
 of
%the joints
for i = shift+1:finish

    theta_trunk(i-shift) = atand((RSHLDR(i,3) - RTROCH(i,3))/
(RSHLDR(i,1)-RTROCH(i,1)));
    if theta_trunk(i-shift) <0
        theta_trunk(i-shift) = 180 + theta_trunk(i-shift);
    end
    theta_thigh(i-shift) = atand((RTROCH(i,3) -RKNEE(i,3))/
(RTROCH(i,1)-RKNEE(i,1)));
    if theta_thigh(i-shift) <0
        theta_thigh(i-shift) = 180 + theta_thigh(i-shift);
    end
    theta_shank(i-shift) = atand((RKNEE(i,3) - RANKLE(i,3))/
(RKNEE(i,1)- RANKLE(i,1)));
    if theta_shank(i-shift) <0
        theta_shank(i-shift) = 180 + theta_shank(i-shift);
    end
    theta_foot(i-shift) = 180 + atand((RHEEL(i, 3) - RTIP(i,3))/
(RHEEL(i,1) -  RTIP(i,1)));

end
```

Joint Angles for Walk(deg)

```
%Joint Angles are calculated using the segment angles
theta_ankle = theta_foot - theta_shank - 90;
theta_hip = theta_thigh - theta_trunk;
theta_knee = theta_thigh - theta_shank;
```

Joint Angular Velocity for Walk(deg/s)

```
%Define new stance phase. This will be 2 indices shorter because due
 to the
%numerical derivative calculation
L2 = L-2;
stancePhase2 = ([0:L2-1]./(L2-1)).*100;

%Initialize angular velocity arrays
knee_avg = zeros(L2,1);
hip_avg = zeros(L2,1);
ankle_avg = zeros(L2,1);

%Calculate angular velocities as the chanage in jonit angle over two
 steps
for i = 1:L2
    knee_avg(i) = ((theta_knee(i+2) - theta_knee(i))/2)*100;
```

```
    hip_avg(i) = ((theta_hip(i+2) - theta_hip(i))/2)*100;
    ankle_avg(i) = ((theta_ankle(i+2) - theta_ankle(i))/2)*100;
end
```

Anthopometry for Walk

```
%The mass of the subject was measured and then the masses of the
 thigh,
%shank and foot were estimated
body_mass = 63; %Kg
thigh_m = .1416*body_mass;
shank_m = .0433*body_mass;
foot_m = .0137*body_mass;

%Determine the length of each segment using the average distance
 between
%the joint marker readings
thighLength = mean(((RTROCH(:,3) - RKNEE(:,3)).^2 + (RTROCH(:,1) -
 RKNEE(:,1)).^2).^.5);
shankLength = mean(((RKNEE(:,3) - RANKLE(:,3)).^2 + (RKNEE(:,1) -
 RANKLE(:,1)).^2).^.5);
footLength = mean(((RHEEL(:,3) - RTIP(:,3)).^2 + (RHEEL(:,1) -
 RTIP(:,1)).^2).^.5);

%Center of mass location along each segment is estimated
%using literature values
thighCOM_p = thighLength*.4095;
shankCOM_p = shankLength*.4459;
footCOM_p = .4415*footLength;
```

Location of Center Of Mass for Walk

```
%Inititialize
thigh_COM = zeros(L, 2);
shank_COM = zeros(L,2);
foot_COM = zeros(L,2);

%Calculate the location of Center Of Mass during the gait cycle
for i = 1:L
    thigh_COM(i,1) = RTROCH(i+shift,1) -
 thighCOM_p.*cosd(theta_thigh(i));
    thigh_COM(i,2) = RTROCH(i+shift,3) -
 thighCOM_p.*sind(theta_thigh(i));
    foot_COM(i,1) = RHEEL(i+shift,1) - footCOM_p.*cosd(theta_foot(i));
    foot_COM(i,2) = RHEEL(i+shift,3) - footCOM_p.*sind(theta_foot(i));
    shank_COM(i,1) = RKNEE(i+shift,1) -
 shankCOM_p.*cosd(theta_shank(i));
    shank_COM(i,2) = RKNEE(i+shift,3) -
 shankCOM_p.*sind(theta_shank(i));
end
```

Moment of Inertia and Radius of Gyration

```
%Radius of Gyration values found from literature (m)
thigh_r = 0.389*thighLength;
shank_r = .249*shankLength;
foot_r = .245*footLength;

%Moment of Intertia (Kgm^2)
thigh_I = thigh_m*(thigh_r^2);
shank_I = shank_m*(shank_r^2);
foot_I = foot_m*(foot_r^2);
```

Linear Velocity/Acceletation for Walk(m/s and m/s^2)

```
%Create new stance phase for the acceleration arrays
L3 = L2 - 2;
stancePhase3 = 100.*[0:L3-1]./(L3-1);

%Initialize the linear velocity arrays for each segment
shank_COM_lv = zeros(L2,2);
foot_COM_lv = zeros(L2,2);
thigh_COM_lv = zeros(L2,2);

%Calculate linear velocities of each segment
for i = 1:L2
    shank_COM_lv(i,:) = ((shank_COM(i+2,:) - shank_COM(i,:))/2)*100;
    foot_COM_lv(i,:) = ((foot_COM(i+2,:) - foot_COM(i,:))/2)*100;
    thigh_COM_lv(i,:) = ((thigh_COM(i+2,:) - thigh_COM(i,:))/2)*100;
end

%Initialize linear accerlation arrays for each segment
shank_COM_la = zeros(L3,2);
foot_COM_la = zeros(L3,2);
thigh_COM_la = zeros(L3,2);

%Calculate linear acceleration of each segment
for i = 1:L3
    shank_COM_la(i,:) = ((shank_COM_lv(i+2,:) -
 shank_COM_lv(i,:))/2)*100;
    foot_COM_la(i,:) = ((foot_COM_lv(i+2,:) -
 foot_COM_lv(i,:))/2)*100;
    thigh_COM_la(i,:) = ((thigh_COM_lv(i+2,:) -
 thigh_COM_lv(i,:))/2)*100;
end
```

Angular Acceleration for Walk(deg/s^2)

```
%Initialize angular velocity vectors
shank_COM_aa = zeros(L2,1);
foot_COM_aa= zeros(L2,1);
```

```
thigh_COM_aa = zeros(L2,1);

%Calculate angular velocity
for i=1:L2
    shank_COM_aa(i)=((theta_shank(i+2)+theta_shank(i)-2*theta_shank(i
+1))/1)*pi/180*10000;
    foot_COM_aa(i)=((theta_foot(i+2)+theta_foot(i)-2*theta_foot(i
+1))/1)*pi/180*10000;
    thigh_COM_aa(i)=((theta_thigh(i+2)+theta_thigh(i)-2*theta_thigh(i
+1))/1)*pi/180*10000;
end
```

Define the Reaction Forces and Segment Weights for Walk

```
%Making Indices Align
zReaction = grf(101:175,3);
xReaction = grf(101:175,1);
COPReaction = COP(101:175,1);

%Define weights of each segment
footWeight = foot_m*9.814;
thighWeight = thigh_m*9.814;
shankWeight = shank_m*9.814;

%Make the indices align with the reaction force data
thigh_COM = thigh_COM(3:length(thigh_COM)-2, :);
foot_COM = foot_COM(3:length(foot_COM)-2, :);
shank_COM = shank_COM(3:length(shank_COM)-2, :);

foot_COM_aa = foot_COM_aa(2:length(foot_COM_aa)-1);
thigh_COM_aa = thigh_COM_aa(2:length(shank_COM_aa)-1);
shank_COM_aa = shank_COM_aa(2:length(shank_COM_aa)-1);

RANKLE2 = RANKLE(101:175, 1:3);
RTROCH2 = RTROCH(101:175, 1:3);
RKNEE2 = RKNEE(101:175, 1:3);
```

Inverse Dynamics for Walk

```
%Moment and joint reaction force of foot
RzFoot = -1.*zReaction + foot_COM_la(:,2).*foot_m + footWeight;
RxFoot = -1.*xReaction + foot_COM_la(:,1).*foot_m + footWeight;
MFoot = foot_I.*foot_COM_aa + zReaction.*(foot_COM(:,1) - COPReaction)
 + RzFoot.*(foot_COM(:,1)-RANKLE2(:,1)) + -1.*RxFoot.*(0 -
 RANKLE2(:,3));

%Initialize array for moment and joint reaction force of the knee
MKnee = zeros(L3,1);
RxKnee = zeros(L3,1);
RzKnee = zeros(L3,1);
```

```
%Calculate the moment and joint reaction forces of the knee
for i = 1:L3
    if theta_shank(i) > 0
        RzKnee(i) = shank_COM_la(i,2)*shank_m + RzFoot(i) +
 shankWeight;
        RxKnee(i) = shank_COM_la(i,1)*shank_m + RxFoot(i);
        MKnee(i) = shank_I*shank_COM_aa(i) +
 -1*RzFoot(i)*(shank_COM(i,1) - RANKLE2(i,1)) +
 RxFoot(i)*(shank_COM(i,2)- RANKLE2(i,3)) + MFoot(i)
 + RxKnee(i)*-1*(shank_COM(i,2)- RKNEE2(i,3)) +
 RzKnee(i)*(shank_COM(i,1) -RKNEE2(i,1));
    else
        RzKnee(i) = shank_COM_la(i,2)*shank_m + RzFoot(i) +
 shankWeight;
        RxKnee(i) = shank_COM_la(i,1)*shank_m + RxFoot(i);
        MKnee(i) = shank_I*shank_COM_aa(i)+RzFoot(i)*(shank_COM(i,1)
 - RANKLE2(i,1)) + RxFoot(i)*(shank_COM(i,2)- RANKLE2(i,3))
 + MFoot(i) + RxKnee(i)*-1*(shank_COM(i,2)- RKNEE2(i,3)) +
 -1*RzKnee(i)*(shank_COM(i,1) -RKNEE2(i,1));
    end
end

%Initialize hip moment and joint reaction forces
MHip = zeros(L3,1);
RxHip= zeros(L3,1);
RzHip= zeros(L3,1);

%Calculate hip moment and joint reaction forces
for i = 1:L3
    if theta_thigh(i) > 0
        RzHip(i) = thigh_COM_la(i,2)*thigh_m + RzKnee(i) +
 thighWeight;
        RxHip(i) = thigh_COM_la(i,1)*thigh_m + RxKnee(i);
        MHip(i) = thigh_I*thigh_COM_aa(i) -1*RzKnee(i)*(thigh_COM(i,1)
 - RKNEE2(i,1)) + RxKnee(i)*(thigh_COM(i,2)- RKNEE2(i,3)) + MKnee(i) +
 -1*RxHip(i)*(thigh_COM(i,2)- RTROCH2(i,3)) + RzHip(i)*(thigh_COM(i,1)
 -RTROCH2(i,1));
    else
        RzHip(i) = thigh_COM_la(i,2)*thigh_m + RzKnee(i) +
 thighWeight;
        RxHip(i) = thigh_COM_la(i,1)*thigh_m + RxKnee(i);
        MHip(i) = thigh_I*thigh_COM_aa(i)+
 -1*RzKnee(i)*(thigh_COM(i,1) - RKNEE2(i,1)) +
 RxKnee(i)*(thigh_COM(i,2)- RKNEE2(i,3)) + MKnee(i) +
 -1*RxHip(i)*(thigh_COM(i,2)- RTROCH2(i,3)) + RzHip(i)*(thigh_COM(i,1)
 -RTROCH2(i,1));
    end
end
```

Import the data for the Fast Walk Trial

```
load('fast03.mat')
```

```
%Downsample the Data
digitaldata = downsample(digitaldata, 10);
COP = downsample(COP, 10);

%Remove high frequency noise from data with butterworth filter
[a,b] = butter(2, 10/50);
[c,d] = butter(2, 5/50);
digitalDataFilt = filtfilt(a,b, digitaldata);
COP_filt = filtfilt(a,b,COP);
markerPositionFilt = filtfilt(c,d, markerposition);

%Ground Reaction Forces
grf = zeros(length(digitalDataFilt),6);
F_M = digitalDataFilt;
grf(:,1) = -F_M(:,1);
grf(:,2) = -F_M(:,2);
grf(:,3) = -F_M(:,3);
grf(:,4) = -F_M(:,4);
grf(:,5) = -F_M(:,5);
grf(:,6) = -F_M(:,6);

%Acquire center of pressure in meters
COP = COP_filt/1000;

%Extract each marker position array frmo marker position matrix
RANKLE = markerPositionFilt(:, 1:3)/1000;
RTIP = markerPositionFilt(:, 4:6)/1000;
RHEEL = markerPositionFilt(:, 7:9)/1000;
RKNEE = markerPositionFilt(:, 10:12)/1000;
RTROCH = markerPositionFilt(:, 13:15)/1000;
LANKLE = markerPositionFilt(:, 16:18)/1000;
LTIP = markerPositionFilt(:, 19:21)/1000;
LHEEL = markerPositionFilt(:, 22:24)/1000;
LKNEE = markerPositionFilt(:, 25:27)/1000;
LTROCH = markerPositionFilt(:, 28:30)/1000;
RSHLDR = markerPositionFilt(:, 31:33)/1000;
LSHLDR = markerPositionFilt(:, 34:36)/1000;
RBACK = markerPositionFilt(:, 37:39)/1000;
```

Segment Angles for Fast Walk(deg)

```
%Determine the range of one gait cycle of the subject from visual
%inspection
top = 124;
bottom = 68;
L = top - bottom +1;
shift = bottom-1;
stancePhaseF = 100.*([0:L-1]./(L-1));

%Initialize segment angles
theta_foot = zeros(L,1);
theta_trunk = zeros(L,1);
theta_shank = zeros(L,1);
```

```
theta_thigh = zeros(L,1);

%Calculate segment angles
for i = bottom:top

    theta_trunk(i-shift) = atand((RSHLDR(i,3) - RTROCH(i,3))/
(RSHLDR(i,1)-RTROCH(i,1)));
    if theta_trunk(i-shift) <0
        theta_trunk(i-shift) = 180 + theta_trunk(i-shift);
    end
    theta_thigh(i-shift) = atand((RTROCH(i,3) -RKNEE(i,3))/
(RTROCH(i,1)-RKNEE(i,1)));
    if theta_thigh(i-shift) <0
        theta_thigh(i-shift) = 180 + theta_thigh(i-shift);
    end
    theta_shank(i-shift) = atand((RKNEE(i,3) - RANKLE(i,3))/
(RKNEE(i,1)- RANKLE(i,1)));
    if theta_shank(i-shift) <0
        theta_shank(i-shift) = 180 + theta_shank(i-shift);
    end
    theta_foot(i-shift) = 180 + atand((RHEEL(i, 3) - RTIP(i,3))/
(RHEEL(i,1) -  RTIP(i,1)));

end
```

Joint Angles for Fast Walk(deg)

```
%Calculate joint angles
theta_ankleF = theta_foot - theta_shank - 90;
theta_hipF = theta_thigh - theta_trunk;
theta_kneeF = theta_thigh - theta_shank;
```

Angular Velocity for Fast Walk(deg/sec)

```
%Define new stance phase for derivative calculation
L2 = L-2;
stancePhase2F = ([0:L2-1]./(L2-1)).*100;

%Initialize the angular velocity arrays
knee_avgF = zeros(L2,1);
hip_avgF = zeros(L2,1);
ankle_avgF = zeros(L2,1);

%Calculate joint angular velocities
for i = 1:L2
    knee_avgF(i) = ((theta_kneeF(i+2) - theta_kneeF(i))/2)*100;
    hip_avgF(i) = ((theta_hipF(i+2) - theta_hipF(i))/2)*100;
    ankle_avgF(i) = ((theta_ankleF(i+2) - theta_ankleF(i))/2)*100;
end
```

Anthopometry for Fast Walk

```
%Find the length of each segment using the average length betwene
 joints
thighLength = mean(((RTROCH(:,3) - RKNEE(:,3)).^2 + (RTROCH(:,1) -
 RKNEE(:,1)).^2).^.5);
shankLength = mean(((RKNEE(:,3) - RANKLE(:,3)).^2 + (RKNEE(:,1) -
 RANKLE(:,1)).^2).^.5);
footLength = mean(((RHEEL(:,3) - RTIP(:,3)).^2 + (RHEEL(:,1) -
 RTIP(:,1)).^2).^.5);

%Center of mass of ecah segment angle from literature values
thighCOM_p = thighLength*.4095;
shankCOM_p = shankLength*.4459;
footCOM_p = .4415*footLength;
```

Location of Center Of Mass for Fast Walk

```
%Inititialize center os mass arrays
thigh_COM = zeros(L, 2);
shank_COM = zeros(L,2);
foot_COM = zeros(L,2);

%Calculate the location of the center of mass during the stance phase
for i = 1:L
    thigh_COM(i,1) = RTROCH(i+shift,1) -
 thighCOM_p.*cosd(theta_thigh(i));
    thigh_COM(i,2) = RTROCH(i+shift,3) -
 thighCOM_p.*sind(theta_thigh(i));
    foot_COM(i,1) = RHEEL(i+shift,1) - footCOM_p.*cosd(theta_foot(i));
    foot_COM(i,2) = RHEEL(i+shift,3) - footCOM_p.*sind(theta_foot(i));
    shank_COM(i,1) = RKNEE(i+shift,1) -
 shankCOM_p.*cosd(theta_shank(i));
    shank_COM(i,2) = RKNEE(i+shift,3) -
 shankCOM_p.*sind(theta_shank(i));
end
```

Linear Velocity/Acceletation for Fast Walk(m/s and m/s^2)

```
%Define new stance phase for the acceleration calculations
L3 = L2 - 2;
stancePhase3F = 100.*[0:L3-1]./(L3-1);

%Inititialize linear velocity arrays for segments
shank_COM_lv = zeros(L2,2);
foot_COM_lv = zeros(L2,2);
thigh_COM_lv = zeros(L2,2);

%Calculate segment linar velocities
```

```
for i = 1:L2
    shank_COM_lv(i,:) = ((shank_COM(i+2,:) - shank_COM(i,:))/2)*100;
    foot_COM_lv(i,:) = ((foot_COM(i+2,:) - foot_COM(i,:))/2)*100;
    thigh_COM_lv(i,:) = ((thigh_COM(i+2,:) - thigh_COM(i,:))/2)*100;
end

%Initialize linear accleration arrays for segments
shank_COM_la = zeros(L3,2);
foot_COM_la = zeros(L3,2);
thigh_COM_la = zeros(L3,2);

%Calculate linear velocities for segments
for i = 1:L3
    shank_COM_la(i,:) = ((shank_COM_lv(i+2,:) -
 shank_COM_lv(i,:))/2)*100;
    foot_COM_la(i,:) = ((foot_COM_lv(i+2,:) -
 foot_COM_lv(i,:))/2)*100;
    thigh_COM_la(i,:) = ((thigh_COM_lv(i+2,:) -
 thigh_COM_lv(i,:))/2)*100;
end
```

Angular Acceleration for Fast Walk(deg/s)

```
%Initialize angular acceleration arrays for segments
shank_COM_aa = zeros(L2,1);
foot_COM_aa= zeros(L2,1);
thigh_COM_aa = zeros(L2,1);

%Calculate angular accelerations of segments
for i=1:L2
    shank_COM_aa(i)=((theta_shank(i+2)+theta_shank(i)-2*theta_shank(i
+1))/1)*pi/180*10000;
    foot_COM_aa(i)=((theta_foot(i+2)+theta_foot(i)-2*theta_foot(i
+1))/1)*pi/180*10000;
    thigh_COM_aa(i)=((theta_thigh(i+2)+theta_thigh(i)-2*theta_thigh(i
+1))/1)*pi/180*10000;
end
```

Defining the Reaction Forces and Segment Weights for Fast Walk

```
%Making Indices of reaction forces align
zReaction = grf(bottom+2:top-2,3);
xReaction = grf(bottom+2:top-2,1);
COPReaction = COP(bottom+2:top-2,1);

%Make the indices align with the reaction force data
thigh_COM = thigh_COM(3:length(thigh_COM)-2, :);
foot_COM = foot_COM(3:length(foot_COM)-2, :);
shank_COM = shank_COM(3:length(shank_COM)-2, :);
```

```
foot_COM_aa = foot_COM_aa(2:length(foot_COM_aa)-1);
thigh_COM_aa = thigh_COM_aa(2:length(shank_COM_aa)-1);
shank_COM_aa = shank_COM_aa(2:length(shank_COM_aa)-1);

RANKLE2 = RANKLE(bottom+2:top-2, 1:3);
RTROCH2 = RTROCH(bottom+2:top-2, 1:3);
RKNEE2 = RKNEE(bottom+2:top-2, 1:3);
```

Inverse Dynamics for Fast Walk

```
%Moment and Joint reaction forces of foot
RzFoot = -1.*zReaction + foot_COM_la(:,2).*foot_m + footWeight;
RxFoot = -1.*xReaction + foot_COM_la(:,1).*foot_m + footWeight;
MFootF = foot_I.*foot_COM_aa + zReaction.*(foot_COM(:,1) -
 COPReaction) + RzFoot.*(foot_COM(:,1)-RANKLE2(:,1)) + -1.*RxFoot.*(0
 - RANKLE2(:,3));

%Initialize moment and joint reaction forces of knee
MKneeF = zeros(L3,1);
RxKnee = zeros(L3,1);
RzKnee = zeros(L3,1);

%Moment and Joint reaction forces of knee
for i = 1:L3
    RzKnee(i) = shank_COM_la(i,2)*shank_m + RzFoot(i) + shankWeight;
    RxKnee(i) = shank_COM_la(i,1)*shank_m + RxFoot(i);
    MKneeF(i) = shank_I*shank_COM_aa(i) + -1*RzFoot(i)*(shank_COM(i,1)
 - RANKLE2(i,1)) + RxFoot(i)*(shank_COM(i,2)- RANKLE2(i,3))
 + MFootF(i) + RxKnee(i)*-1*(shank_COM(i,2)- RKNEE2(i,3)) +
 RzKnee(i)*(shank_COM(i,1) -RKNEE2(i,1));
end

%Initialize moment and joint reaction forces for thigh
MHipF = zeros(L3,1);
RxHip = zeros(L3,1);
RzHip = zeros(L3,1);

%Calculate thigh moment and joint reaction forces
for i = 1:L3
    RzHip(i) = thigh_COM_la(i,2)*thigh_m + RzKnee(i) + thighWeight;
    RxHip(i) = thigh_COM_la(i,1)*thigh_m + RxKnee(i);
    MHipF(i) = thigh_I*thigh_COM_aa(i) -1*RzKnee(i)*(thigh_COM(i,1) -
 RKNEE2(i,1)) + RxKnee(i)*(thigh_COM(i,2)- RKNEE2(i,3)) + MKneeF(i) +
 -1*RxHip(i)*(thigh_COM(i,2)- RTROCH2(i,3)) + RzHip(i)*(thigh_COM(i,1)
 -RTROCH2(i,1));
end
```

Import the Data from the Jog Trial

```
load('jogging09.mat')

%Downsample the data
```

```
digitaldata = downsample(digitaldata, 10);
COP = downsample(COP, 10);

%Remove hgh frequency noise with butterworth filter
[a,b] = butter(2, 10/50);
[c,d] = butter(2, 5/50);
digitalDataFilt = filtfilt(a,b, digitaldata);
COP_filt = filtfilt(a,b,COP);
markerPositionFilt = filtfilt(c,d, markerposition);

%Ground Reaction Forces
grf = zeros(length(digitalDataFilt),6);
F_M = digitalDataFilt;
grf(:,1) = -F_M(:,1);
grf(:,2) = -F_M(:,2);
grf(:,3) = -F_M(:,3);
grf(:,4) = -F_M(:,4);
grf(:,5) = -F_M(:,5);
grf(:,6) = -F_M(:,6);

%Acquire the center of pressure in meters
COP = COP_filt/1000;

%Extract markers from markerPosition matrix
RANKLE = markerPositionFilt(:, 1:3)/1000;
RTIP = markerPositionFilt(:, 4:6)/1000;
RHEEL = markerPositionFilt(:, 7:9)/1000;
RKNEE = markerPositionFilt(:, 10:12)/1000;
RTROCH = markerPositionFilt(:, 13:15)/1000;
LANKLE = markerPositionFilt(:, 16:18)/1000;
LTIP = markerPositionFilt(:, 19:21)/1000;
LHEEL = markerPositionFilt(:, 22:24)/1000;
LKNEE = markerPositionFilt(:, 25:27)/1000;
LTROCH = markerPositionFilt(:, 28:30)/1000;
RSHLDR = markerPositionFilt(:, 31:33)/1000;
LSHLDR = markerPositionFilt(:, 34:36)/1000;
RBACK = markerPositionFilt(:, 37:39)/1000;
```

Segment Angles for Jog(deg)

```
%Determine the range of one gait cycle of the subject from visual
%inspection
top = 109;
bottom = 65;
L = top - bottom +1;
shift = bottom-1;
stancePhaseR = 100.*([0:L-1]./(L-1));

%Initialize segment angle arrays
theta_foot = zeros(L,1);
theta_trunk = zeros(L,1);
theta_shank = zeros(L,1);
theta_thigh = zeros(L,1);
```

```
%Calculate the segment angles
for i = bottom:top

    theta_trunk(i-shift) = atand((RSHLDR(i,3) - RTROCH(i,3))/
(RSHLDR(i,1)-RTROCH(i,1)));
    if theta_trunk(i-shift) <0
        theta_trunk(i-shift) = 180 + theta_trunk(i-shift);
    end
    theta_thigh(i-shift) = atand((RTROCH(i,3) -RKNEE(i,3))/
(RTROCH(i,1)-RKNEE(i,1)));
    if theta_thigh(i-shift) <0
        theta_thigh(i-shift) = 180 + theta_thigh(i-shift);
    end
    theta_shank(i-shift) = atand((RKNEE(i,3) - RANKLE(i,3))/
(RKNEE(i,1)- RANKLE(i,1)));
    if theta_shank(i-shift) <0
        theta_shank(i-shift) = 180 + theta_shank(i-shift);
    end
    theta_foot(i-shift) = 180 + atand((RHEEL(i, 3) - RTIP(i,3))/
(RHEEL(i,1) -  RTIP(i,1)));

end
```

Joint Angles for Jog(deg)

```
%Calculate the joint angles
theta_ankleR = theta_foot - theta_shank - 90;
theta_hipR = theta_thigh - theta_trunk;
theta_kneeR = theta_thigh - theta_shank;
```

Joint Angular velocity for Jog(deg/s)

```
%Define new stance phase for velocity calculation
L2 = L-2;
stancePhase2R = ([0:L2-1]./(L2-1)).*100;

%Initialize angular velocity arrays
knee_avgR = zeros(L2,1);
hip_avgR = zeros(L2,1);
ankle_avgR = zeros(L2,1);

%Calculate joint angular velocity
for i = 1:L2
    knee_avgR(i) = ((theta_kneeR(i+2) - theta_kneeR(i))/2)*100;
    hip_avgR(i) = ((theta_hipR(i+2) - theta_hipR(i))/2)*100;
    ankle_avgR(i) = ((theta_ankleR(i+2) - theta_ankleR(i))/2)*100;
end
```

Anthopometry for Jog

```
%Average segment length is calculated from the marker data
```

```
thighLength = mean(((RTROCH(:,3) - RKNEE(:,3)).^2 + (RTROCH(:,1) -
 RKNEE(:,1)).^2).^.5);
shankLength = mean(((RKNEE(:,3) - RANKLE(:,3)).^2 + (RKNEE(:,1) -
 RANKLE(:,1)).^2).^.5);
footLength = mean(((RHEEL(:,3) - RTIP(:,3)).^2 + (RHEEL(:,1) -
 RTIP(:,1)).^2).^.5);

%Center of mass location along segments is estimated with literature
 values
thighCOM_p = thighLength*.4095;
shankCOM_p = shankLength*.4459;
footCOM_p = .4415*footLength;
```

Location of Center Of Mass for Jog

```
%initialize
thigh_COM = zeros(L, 2);
shank_COM = zeros(L,2);
foot_COM = zeros(L,2);

%Calculate center of mass location during stance phase
for i = 1:L
    thigh_COM(i,1) = RTROCH(i+shift,1) -
 thighCOM_p.*cosd(theta_thigh(i));
    thigh_COM(i,2) = RTROCH(i+shift,3) -
 thighCOM_p.*sind(theta_thigh(i));
    foot_COM(i,1) = RHEEL(i+shift,1) - footCOM_p.*cosd(theta_foot(i));
    foot_COM(i,2) = RHEEL(i+shift,3) - footCOM_p.*sind(theta_foot(i));
    shank_COM(i,1) = RKNEE(i+shift,1) -
 shankCOM_p.*cosd(theta_shank(i));
    shank_COM(i,2) = RKNEE(i+shift,3) -
 shankCOM_p.*sind(theta_shank(i));
end
```

Linear Velocity/Acceletation for Jog(m/s and m/s^2)

```
%Define new stance phase for acceleration calculations
L3 = L2 - 2;
stancePhase3R = 100.*[0:L3-1]./(L3-1);

%Initialize linear velocity arrays
shank_COM_lv = zeros(L2,2);
foot_COM_lv = zeros(L2,2);
thigh_COM_lv = zeros(L2,2);

%Calculate linear velocity of segments
for i = 1:L2
    shank_COM_lv(i,:) = ((shank_COM(i+2,:) - shank_COM(i,:))/2)*100;
    foot_COM_lv(i,:) = ((foot_COM(i+2,:) - foot_COM(i,:))/2)*100;
    thigh_COM_lv(i,:) = ((thigh_COM(i+2,:) - thigh_COM(i,:))/2)*100;
```

```
end

%Inialize linear acceleration arrays
shank_COM_la = zeros(L3,2);
foot_COM_la = zeros(L3,2);
thigh_COM_la = zeros(L3,2);

%Calculate linear accelerations
for i = 1:L3
    shank_COM_la(i,:) = ((shank_COM_lv(i+2,:) -
 shank_COM_lv(i,:))/2)*100;
    foot_COM_la(i,:) = ((foot_COM_lv(i+2,:) -
 foot_COM_lv(i,:))/2)*100;
    thigh_COM_la(i,:) = ((thigh_COM_lv(i+2,:) -
 thigh_COM_lv(i,:))/2)*100;
end
```

Angular Acceleration for Jog(deg/s^2)

```
%Initialize
shank_COM_aa = zeros(L2,1);
foot_COM_aa= zeros(L2,1);
thigh_COM_aa = zeros(L2,1);

%Calculate joint angular accelerations
for i=1:L2
    shank_COM_aa(i)=((theta_shank(i+2)+theta_shank(i)-2*theta_shank(i
+1))/1)*pi/180*10000;
    foot_COM_aa(i)=((theta_foot(i+2)+theta_foot(i)-2*theta_foot(i
+1))/1)*pi/180*10000;
    thigh_COM_aa(i)=((theta_thigh(i+2)+theta_thigh(i)-2*theta_thigh(i
+1))/1)*pi/180*10000;
end
```

Defining the Reaction Forces and Segment Weights for Jog

```
%Making Indices of reaction forces align
zReaction = grf(bottom+2:top-2,3);
xReaction = grf(bottom+2:top-2,1);
COPReaction = COP(bottom+2:top-2,1);

%Make the indices align with the reaction force data
thigh_COM = thigh_COM(3:length(thigh_COM)-2, :);
foot_COM = foot_COM(3:length(foot_COM)-2, :);
shank_COM = shank_COM(3:length(shank_COM)-2, :);

foot_COM_aa = foot_COM_aa(2:length(foot_COM_aa)-1);
thigh_COM_aa = thigh_COM_aa(2:length(shank_COM_aa)-1);
shank_COM_aa = shank_COM_aa(2:length(shank_COM_aa)-1);
```

```
RANKLE2 = RANKLE(bottom+2:top-2, 1:3);
RTROCH2 = RTROCH(bottom+2:top-2, 1:3);
RKNEE2 = RKNEE(bottom+2:top-2, 1:3);
```

Inverse Dynamics for Jog

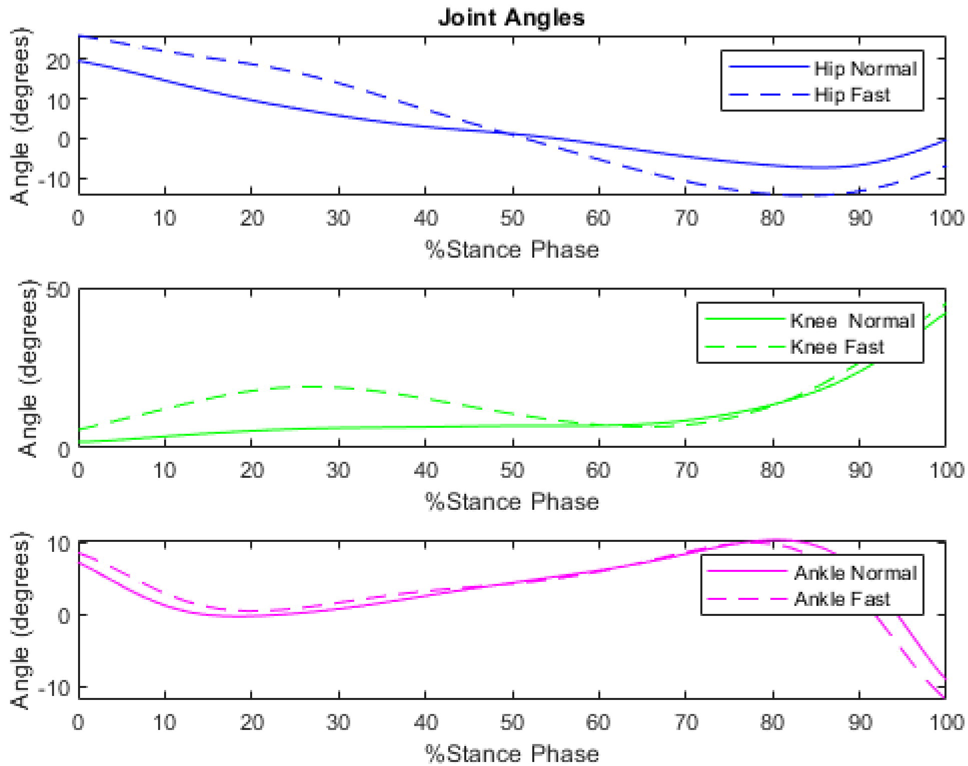
```
%Moment and Joint reaction forces of foot
RzFoot = -1.*zReaction + foot_COM_la(:,2).*foot_m + footWeight;
RxFoot = -1.*xReaction + foot_COM_la(:,1).*foot_m + footWeight;
MFootR = foot_I.*foot_COM_aa + zReaction.*(foot_COM(:,1) -
 COPReaction) + RzFoot.*(foot_COM(:,1)-RANKLE2(:,1)) + -1.*RxFoot.*(0
 - RANKLE2(:,3));

%Initialize moment and joint reaction force of knee arrays
MKneeR = zeros(L3,1);
RxKnee = zeros(L3,1);
RzKnee = zeros(L3,1);

%Moment and Joint reaction forces of knee
for i = 1:L3
    RzKnee(i) = shank_COM_la(i,2)*shank_m + RzFoot(i) + shankWeight;
    RxKnee(i) = shank_COM_la(i,1)*shank_m + RxFoot(i);
    MKneeR(i) = shank_I*shank_COM_aa(i) + -1*RzFoot(i)*(shank_COM(i,1)
 - RANKLE2(i,1)) + RxFoot(i)*(shank_COM(i,2)- RANKLE2(i,3))
 + MFootF(i) + RxKnee(i)*-1*(shank_COM(i,2)- RKNEE2(i,3)) +
 RzKnee(i)*(shank_COM(i,1) -RKNEE2(i,1));
end

%Initialize moment and joint reaction force of thigh arrays
MHipR = zeros(L3,1);
RxHip= zeros(L3,1);
RzHip= zeros(L3,1);

%Moment and Joint reaction forces of thigh
for i = 1:L3
    RzHip(i) = thigh_COM_la(i,2)*thigh_m + RzKnee(i) + shankWeight;
    RxHip(i) = thigh_COM_la(i,1)*thigh_m + RxKnee(i);
    MHipR(i) = thigh_I*thigh_COM_aa(i) -1*RzKnee(i)*(thigh_COM(i,1) -
 RKNEE2(i,1)) + RxKnee(i)*(thigh_COM(i,2)- RKNEE2(i,3)) + MKneeF(i) +
 -1*RxHip(i)*(thigh_COM(i,2)- RTROCH2(i,3)) + RzHip(i)*(thigh_COM(i,1)
 -RTROCH2(i,1));
end
```

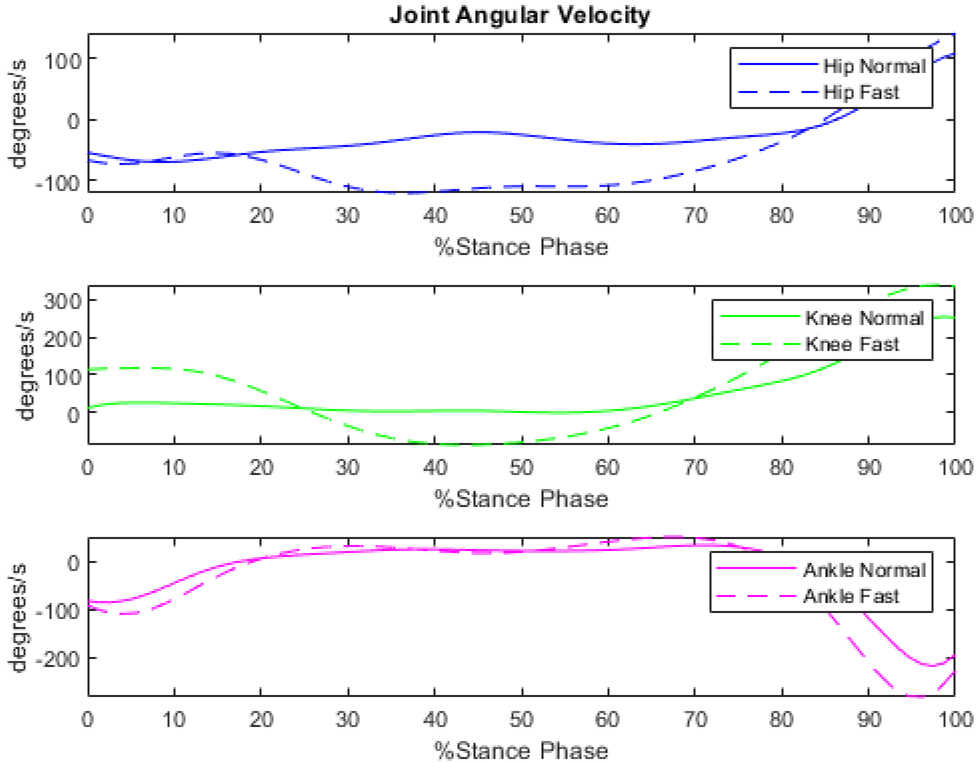
Final Plot of Results

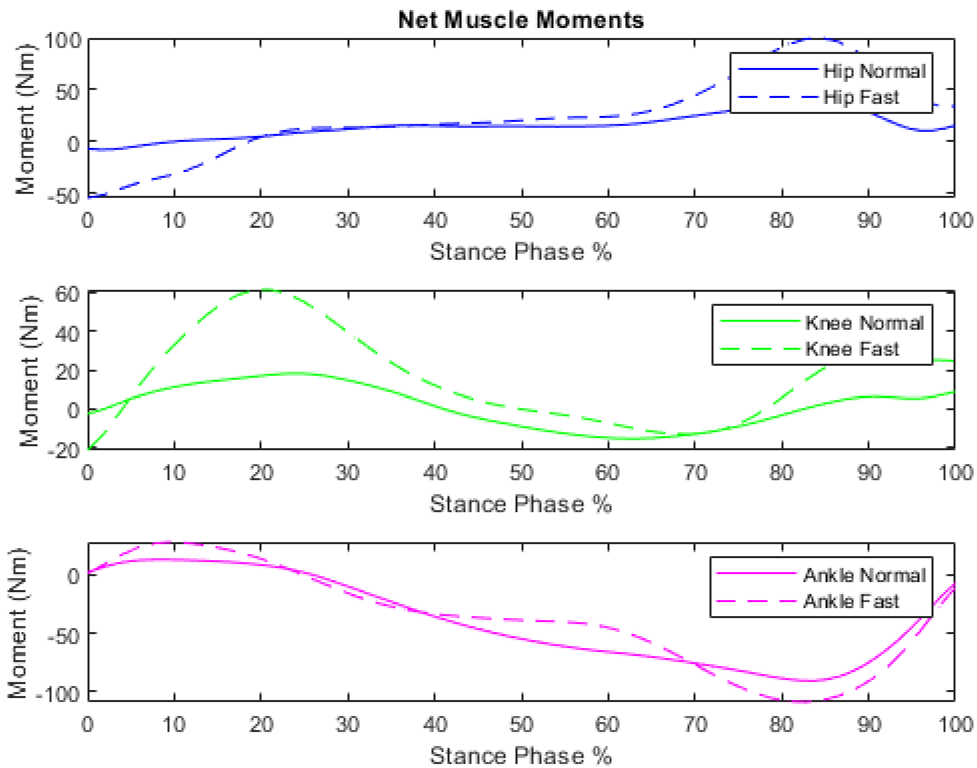
```
%Joint Angles
figure(1)
subplot(3,1,1)
plot(stancePhase, theta_hip,'b-')
hold on
plot(stancePhaseF, theta_hipF,'b--')
title('Joint Angles')
```

```
xlabel('%Stance Phase');
ylabel('Angle (degrees)');
legend('Hip Normal', 'Hip Fast')
hold off
subplot(3,1,2)
plot(stancePhase, theta_knee, 'g-')
hold on
plot(stancePhaseF, theta_kneeF, 'g--')
xlabel('%Stance Phase');
ylabel('Angle (degrees)');
legend('Knee  Normal', 'Knee Fast')
hold off
subplot(3,1,3)
plot(stancePhase, theta_ankle, 'm-')
hold on
plot(stancePhaseF, theta_ankleF, 'm--')
xlabel('%Stance Phase');
ylabel('Angle (degrees)');
legend('Ankle Normal', 'Ankle Fast');
hold off

%Joint Angular Velocity
figure(2)
subplot(3,1,1)
plot(stancePhase2, hip_avg,'b-')
hold on
plot(stancePhase2F, hip_avgF,'b--')
title('Joint Angular Velocity')
xlabel('%Stance Phase');
ylabel('degrees/s');
legend('Hip Normal', 'Hip Fast')
hold off
subplot(3,1,2)
plot(stancePhase2, knee_avg, 'g-')
hold on
plot(stancePhase2F, knee_avgF, 'g--')
xlabel('%Stance Phase');
ylabel('degrees/s');
legend('Knee Normal', 'Knee Fast')
hold off
subplot(3,1,3)
plot(stancePhase2, ankle_avg, 'm-')
hold on
plot(stancePhase2F, ankle_avgF, 'm--')
xlabel('%Stance Phase');
ylabel('degrees/s');
legend('Ankle Normal', 'Ankle Fast');
hold off

%Moments
figure(3)
subplot(3,1,1)
plot(stancePhase3, MHip, 'b-')
hold on
```

```
plot(stancePhase3F, MHipF, 'b--')
legend('Hip Normal', 'Hip Fast')
title('Net Muscle Moments')
xlabel('Stance Phase %')
ylabel('Moment (Nm)')
hold off
subplot(3,1,2)
plot(stancePhase3, MKnee, 'g-')
hold on
plot(stancePhase3F, MKneeF, 'g--')
legend('Knee Normal', 'Knee Fast')
xlabel('Stance Phase %')
ylabel('Moment (Nm)')
hold off
subplot(3,1,3)
plot(stancePhase3, MFoot, 'm-')
hold on
plot(stancePhase3F, MFootF, 'm--')
legend('Ankle Normal', 'Ankle Fast')
xlabel('Stance Phase %')
ylabel('Moment (Nm)')
hold off
```




Joint Angular Velocity
degrees/s
Hip Normal
Hip Fast
%Stance Phase
degrees/s
Knee Normal
Knee Fast
%Stance Phase
degrees/s
Ankle Normal
Ankle Fast
%Stance Phase


Net Muscle Moments
Moment (Nm)
Hip Normal
Hip Fast
Stance Phase %
Moment (Nm)
Knee Normal
Knee Fast
Stance Phase %
Moment (Nm)
Ankle Normal
Ankle Fast
Stance Phase %

Published with MATLAB® R2021a