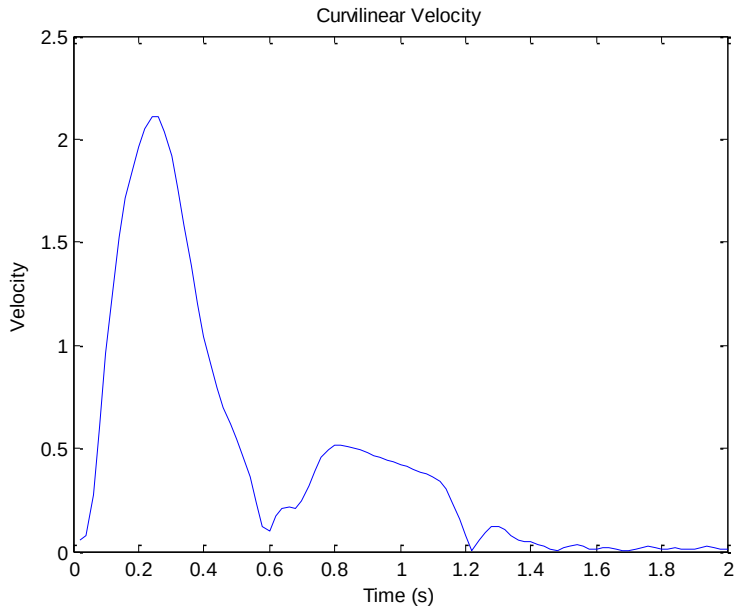
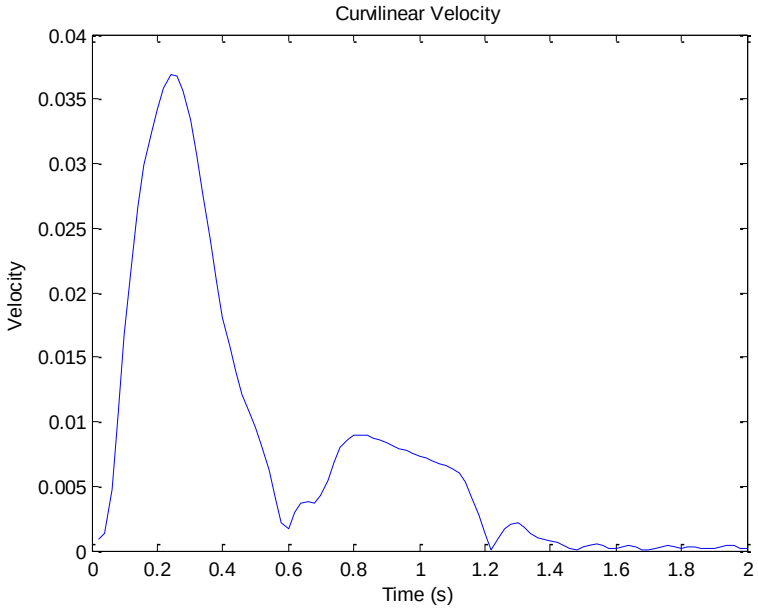
Debugging Velocity Plots Code, 12.02.14

If joint angles in radians are used for the calculation, and final velocity values are multiplied by (180/pi), the resulting velocity plot is: (units are presumably meters/second)



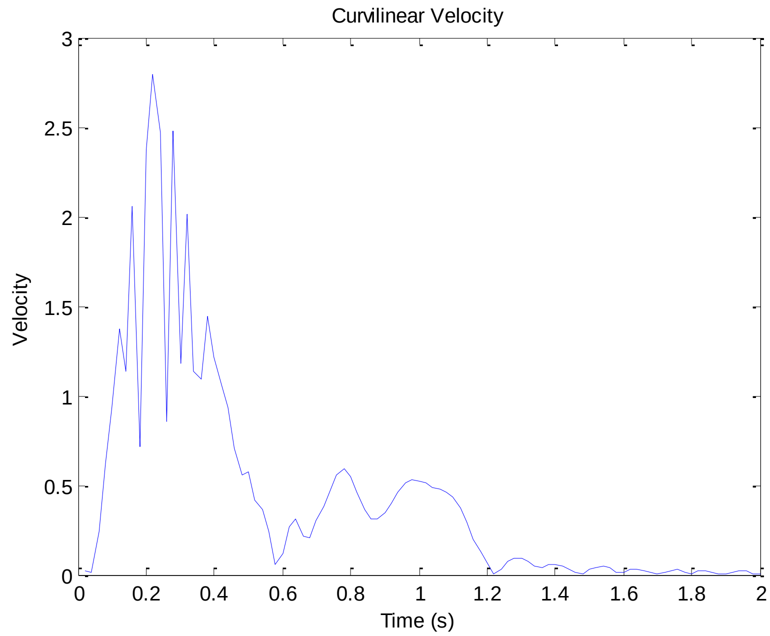
If joint angles in radians are used for the calculation, and there is no final conversion of velocity units, the resulting velocity plot is:

~ velocity values are much too small to correspond to any practical units



If joint angles in degrees are used for the calculation, and there is no final conversion of velocity units, the resulting velocity plot is:

~ note that this has the same general shape as the first plot above, but the main peak is much more jagged and has a larger peak value than in the first plot above.



~~~~~~

Matlab code used to create the above plots:

```
load JointAngles_Data.out;

jointAngles = JointAngles_Data;

% Data format:
%  Column 1: Task Number
%  Column 2: Timestep (20 ms)
%  Column 3: Shoulder Angle (in degrees)
%  Column 4: Elbow Angle (in degrees)
%  Column 5: Target/Goal Shoulder Angle (in degrees)
%  Column 6: Target/Goal Elbow Angle (in degrees)

% Select one of the two following units for the calculation:
q = jointAngles(:,3:6)*(3.14159/180.0);  % joint angles in units of radians
q = jointAngles(:,3:6);                  % joint angles in units of degrees

derivative = diff(q);   % Take the derivative of the joint angle data set

% correct for missing derivative at last timestep of movement
derivative(100,:) = derivative(99, :);

% Arm segment lengths, in meters
L1 = 0.33;
L2 = 0.32;
```
~~~~~~

```
% Calculate X and Y velocities using the Jacobian
for i = 1:100,   % for each timestep of the movement
    deriv1 = derivative(i,1);
    deriv2 = derivative(i,2);

    xComponent1 = -L1*sin(q(i,1));
    xComponent2 = -L2*sin(q(i,1)+q(i,2));
    xComponent3 = -L2*sin(q(i,1)+q(i,2));

    yComponent1 = L1*cos(q(i,1));
    yComponent2 = L2*cos(q(i,1)+q(i,2));
    yComponent3 = L2*cos(q(i,1)+q(i,2));

    xTerm1 = (xComponent1 + xComponent2)*deriv1;
    xTerm2 = (xComponent3)*deriv2;
    yTerm1 = (yComponent1 + yComponent2)*deriv1;
    yTerm2 = (yComponent3)*deriv2;

    velocity_x_singleStep = xTerm1 + xTerm2; % note that - sign is accounted for above
    velocity_y_singleStep = yTerm1 + yTerm2;

    velocity_x(i) = velocity_x_singleStep;
    velocity_y(i) = velocity_y_singleStep;

    deriv1 = 0.0;
    deriv2 = 0.0;

    velocity_x_singleStep = 0.0;
    velocity_y_singleStep = 0.0;

end


% Calculate curvilinear velocity from X and Y velocities
for k = 1:100,
   velocity_curvilinear(k) = sqrt((velocity_x(k)*velocity_x(k)) +
(velocity_y(k)*velocity_y(k)));
end


time_array = 0.02:0.02:2.0;

figure;
%plot(time_array, velocity_curvilinear.*(180.0/3.14159));  % convert from rad to degrees
plot(time_array, velocity_curvilinear);   % Don't perform any data conversion
title('Curvilinear Velocity');
xlabel('Time (s)');
ylabel('Velocity');
```