Walking Arm Trebuchet Dynamic Simulation - ME 326 Ex. 4

Original Author: Charlie Refvem

Student Author: Nash Elder

Last Modified: 06/05/2020

Academic Honesty Statement:

As a student of Cal Poly University, I, Nash Elder, pledge that I have notknowingly given nor received any inappropriate assistance regarding exercise 4 in accordance with the course syllabus.

Description:

The motion of the walking arm trebuchet as well as the projectile will be modeled and analyzed. The motion is analyzed in 4 regimes: falling, impact, swinging, and flying. Kinematics and vector loops are utilized, as well as kinetic analysis of Newton's 2nd Law and impulse-momentum. This results in a full simulation of the trebuchet and projectile.

Required Files:

`main.mlx` - Main script for entire project. Only run this file directly.

`reg_1_model.slx` - Block diagram for simulating Regime 1. Contains two matlab function blocks, **`reg_1_EOM`** and **`reg_1_output`** which handle the equations of motion and output equations respectively.

`jbi2jai.mlx` - Live function for handling Regime 2. Relates the system state just before impact to the system state just after impact.

`reg_3_model.slx` - Block diagram for simulating Regime 3. Contains two matlab function blocks, **`reg_3_EOM`** and **`reg_3_output`** which handle the equations of motion and output equations respectively.

`reg_4_model.slx` - Block diagram for simulating Regime 4. Contains two matlab function blocks, **`reg_4_EOM`** and **`reg_4_output`** which handle the equations of

motion and output equations respectively.

Table of Contents

Prepare Workspace

This section makes sets up the MATLAB workspace to help everything run smoothly. First, all the figure windows are closed before the script starts.

```
close all;
```

System Constants

This section defines all the constant parameters and settings that will be used across all regimes. First, an empty struct, `P`, is created to hold all of the constant parameters. This struct will be passed to all functions and Simulink models to guarantee that the parameters match between different parts of the simulation. Doing so guarantees that each constant is only defined one time as part of the struct.

```
P = struct();
```

All of the link lengths are defined in units of feet.

```
P.r_AC = 20/12;
P.r_CD = 20/12;
P.r_BC = 15/12;
P.r_DE = 18/12;
P.r_AF = P.r_AC/2;
P.r_CF = P.r_AC-P.r_AF;
P.r_BD = P.r_BC+P.r_CD;
```

Gravitational acceleration is in ft/s/s.

```
P.g = 32.17;
```

The linear weight density of a 2x2 in lbf/ft is used to find the weight of each link.

```
P.rho = 0.7;
```

Masses of each link are calculated in slugs. The mass of link 1 is calculated by assuming uniform density of a 2x2 over the full length between point A and point C.

```
P.m_1 = P.rho*P.r_AC/P.g;
```

Link 2 includes the mass of the 20 lbf counterweight in addition to the mass of the 2x2 going from point B to point D.

```
P.m_w = 20/P.g;
P.m_BD = P.rho*P.r_BD/P.g;
P.m_2 = P.m_w+P.m_BD;
```

Link 3 includes only the mass of the projectile because the string is of negligable mass.

```
P.m_3 = 0.25/P.g; % mass of projectile
```

With the masses defined, the location of point G, the center of mass for link 2, is determined. From this location, remaining dependant link lengths can be resolved.

```
P.r_DG = (P.r_BD*P.m_w + P.r_CD*P.m_BD)/(P.m_2);
P.r_CG = P.r_DG-P.r_CD;
P.r_BG = P.r_BD-P.r_DG;
```

The moments of inertia for link 1 and link 2 are calculated in slug*ft^2. Link 3 has no moment of inertia because the string is considered massless.

```
P.I_1 = 1/12*P.m_1*P.r_AC^2;
```

Link 2 is a composite body so the moment of inertia must be considered for both parts of the body.

```
P.I_2 = 1/12*P.m_BD*P.r_BD^2 ...                    % link BD
          + P.m_BD*(P.r_BD/2 - P.r_BG)^2 ...        % P.A.T. for link BD
          + P.m_w*P.r_BG^2;                         % counterweight
```

A framerate is selected that will eventually be used to set the framerate for an animation.

```
P.fps = 30;
```

A release condition is specified as the maximum angle between Link 2 and Link 3 before the projectile leaves the mechanism.

```
P.th_release = deg2rad(165);
```

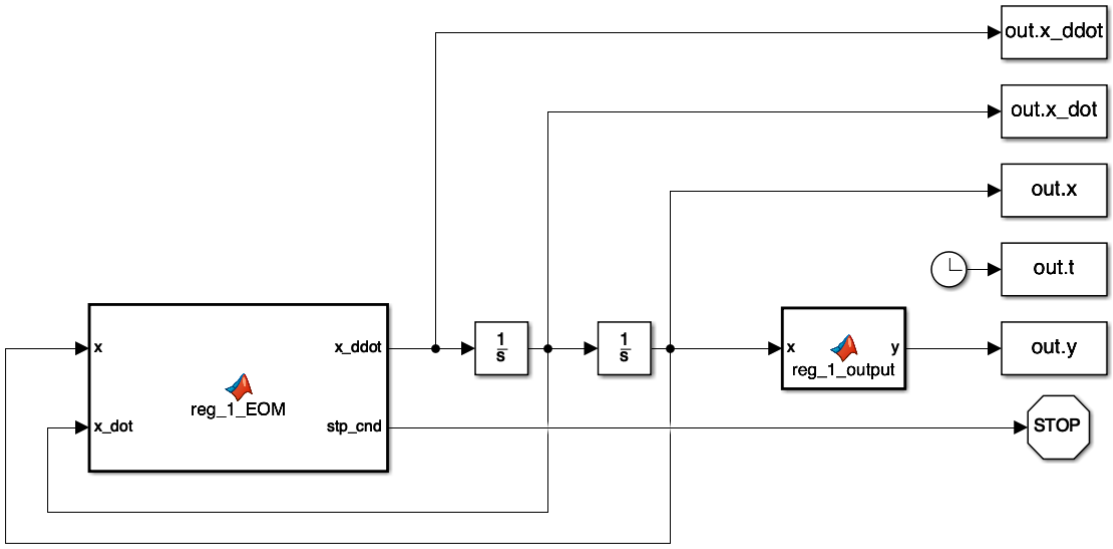
Finally, a struct will be created for each regime to hold the settings and results associated with that regime specifically.

```
reg1 = struct();
reg3 = struct();
reg4 = struct();
```

Regime 1 - Falling

This section will handle everything associated with Regime 1, which covers the falling motion of the trebuchet before it hits the ground.

Observe the regime 1 Simulink diagram based upon hand calculations and equations of motion:



Initial conditions

All initial angles, in radians, are defined. These angles can be adjusted to improve the performance of the trebuchet and prevent interference with the ground before the projectile flies off.

```
reg1.xi     = [ deg2rad(-35)     % th1                    <------
                deg2rad(95)      % th2                    <------
                deg2rad(100) ];  % th3                    <------
```

Similarly, the initial angular velocities are defined, all in radians per second. The system starts at rest, so these velocities are all zero.

```
reg1.xi_dot = [ 0                % th1_dot
                0                % th2_dot
                0 ];             % th3_dot
```

Simulation

The simulation for Regime 1 must run long enough to guarantee that the trebuchet impacts with the ground. This will be accomplished by running the simulation indefinitely and relying on a "Stop Simulation" block in Simulink to end the simulation. Even though the simulation will automatically determine when to end, the simulation does need to know when to start, so a start time is defined for the simulation.

```
reg1.ti     = 0;
```

Here the Simulink model is run and the output is stored for post processing.

```
reg1.out    = sim('reg_1_model');
```

Post Processing

This section resamples the output to achieve a constant sample rate with respect to time to use for the animation. A constant sample rate will help the animation to run closer to real-time. First the stop time for regime 1 is calculated and used to find a new set of time samples at a constant interval near the desired framerate.

First, the time of the final simulation step is recorded as the stop time for Regime 1.

```
reg1.tf     = reg1.out.t(end);
reg1.xf     = reg1.out.x(end,:)';
reg1.xf_dot = reg1.out.x_dot(end,:)';
```

The simulation of Regime 1 ended automatically when the trebuchet made impact with the ground. Because this end time occurs at an arbitrary point in time, it will not be possible to resample at the desired framrate and end up with an integer number of samples. Instead, the output data will be sampled at whatever framerate is closest to the desired framerate such that an integer number of samples are created.

The stop time and desired framerate are used to determine the number of data points to use in resampling.

```
reg1.N      = length(reg1.ti:(1/P.fps):reg1.tf);
```

The number of data points is used to create a new list of sample times with a sample rate very near the desired framerate.

```
reg1.t      = linspace(reg1.ti,reg1.tf,reg1.N)';
```

WIth the new list of sample times, the output of Regime 1 can be resampled using linear interpolation.

```
reg1.pts    = interp1(reg1.out.t,reg1.out.y,reg1.t);
```

The actual framerate can be determined after the new list of samples is created.

```
fprintf('Regime 1 has been resampled at a framerate of %2.2f FPS resulting in
%2.0f samples.', ...
    1/(reg1.t(2)-reg1.t(1)),reg1.N);
```

Regime 2 - Impact

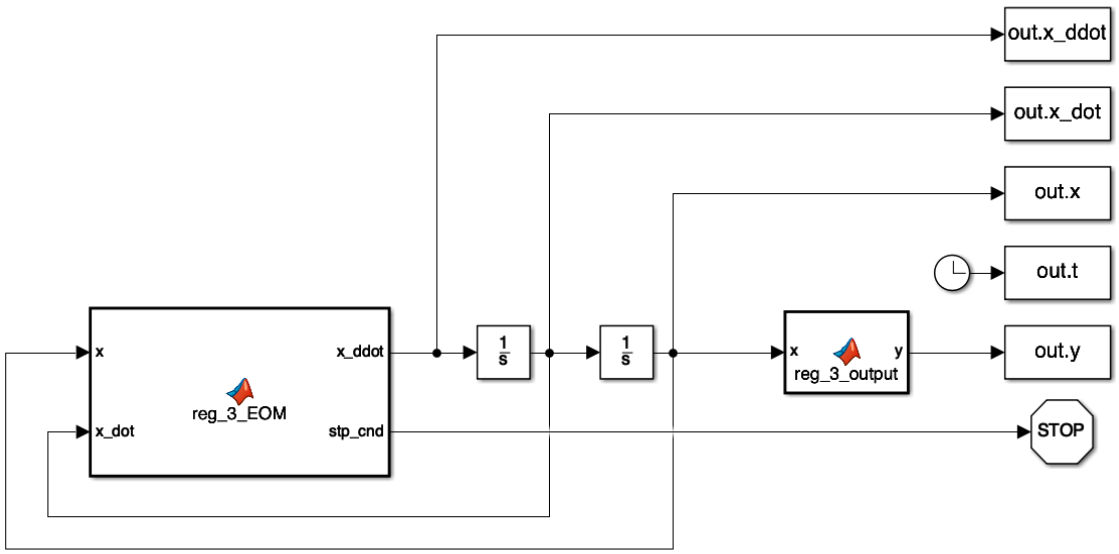
State Transformation

This section handles the instantaneous change in state between the falling regime and the swinging regime. This is achieved by running a function that analyzes the state of the system just before impact (JBI) to determine the state of the system just after impact (JAI). Regime 2 occurs instantaneously and does not require its own simulation.

```
[reg3.xi, reg3.xi_dot] = jbi2jai(reg1.xf, reg1.xf_dot, P);
```

Regime 3 - Swinging

Observe the regime 3 Simulink diagram based upon hand calculations and equations of motion:



Initial Conditions

The initial conditions for Regime 3 have already been handled during Regime 2.

Simulation

In theory, the start time for Regime 3 should be the stop time from Regime 1 if the impact is modeled as an impulse.

```
reg3.ti = 0;                                                % <------
reg3.out = sim('reg_3_model.slx');
```

Post Processing

This section resamples the output to achieve a constant sample rate with respect to time to use for the animation. A constant sample rate will help the animation to run closer to real-time. First the stop time for regime 1 is calculated and used to find a new set of time samples at a constant interval near the desired framerate.

First, the time of the final simulation step is recorded as the stop time for Regime 1.

```
reg3.tf     = reg3.out.t(end);
reg3.xf     = reg3.out.x(end,:)';
reg3.xf_dot = reg3.out.x_dot(end,:)';
```

The simulation of Regime 1 ended automatically when the trebuchet made impact with the ground. Because this end time occurs at an arbitrary point in time, it will not be possible to resample at the desired framrate and end up with an integer number of samples. Instead, the output data will be sampled at whatever framerate is closest to the desired framerate such that an integer number of samples are created.

The stop time and desired framerate are used to determine the number of data points to use in resampling.

```
reg3.N      = length(reg3.ti:(1/P.fps):reg3.tf);
```

The number of data points is used to create a new list of sample times with a sample rate very near the desired framerate.

```
reg3.t      = linspace(reg3.ti,reg3.tf,reg3.N)';
```

WIth the new list of sample times, the output of Regime 1 can be resampled using linear interpolation.

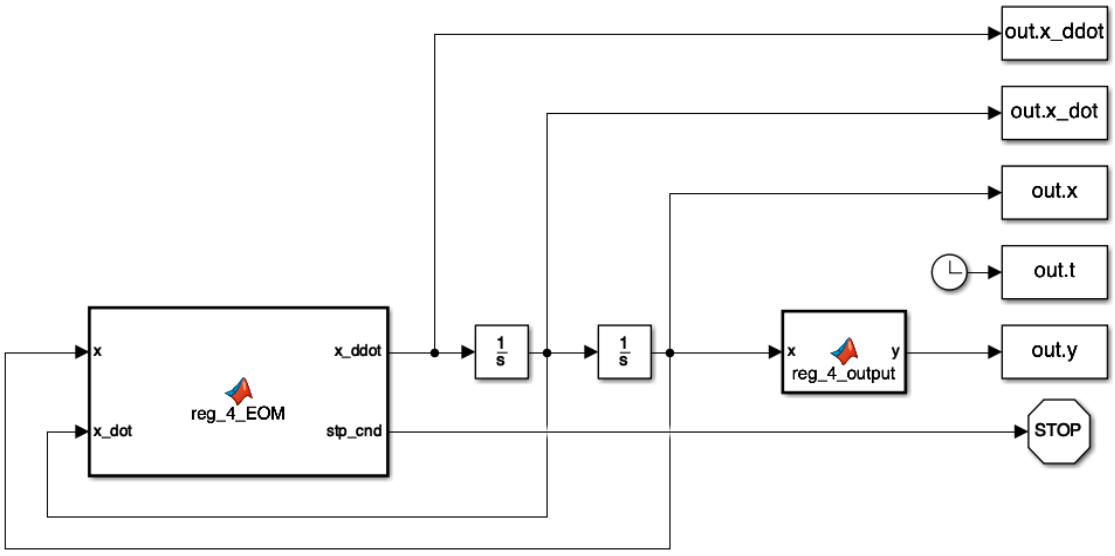
```
reg3.pts    = interp1(reg3.out.t,reg3.out.y,reg3.t);
```

The actual framerate can be determined after the new list of samples is created.

```
fprintf('Regime 3 has been resampled at a framerate of %2.2f FPS resulting in
%2.0f samples.', ...
    1/(reg3.t(2)-reg3.t(1)),reg3.N);
```

Regime 4 - Flying

Observe the regime 3 Simulink diagram based upon hand calculations and equations of motion:



Initial Conditions

```
reg4.xi     = [  0                                                  % <------
                 0 ];                                               % <------
reg4.xi_dot = [  0                                                  % <------
                 0 ];                                               % <------
```

Simulation

```
reg4.ti = 0;                                                        % <------
reg4.out = sim('reg_4_model.slx');
```

Post Processing

First, the time of the final simulation step is recorded as the stop time for Regime 1.

```
reg4.tf     = reg4.out.t(end);
reg4.xf     = reg4.out.x(end,:)';
reg4.xf_dot = reg4.out.x_dot(end,:)';
```

The stop time and desired framerate are used to determine the number of data points to use in resampling.

```
reg4.N      = length(reg4.ti:(1/P.fps):reg4.tf);
```

The number of data points is used to create a new list of sample times with a sample rate very near the desired framerate.

```
reg4.t      = linspace(reg4.ti,reg4.tf,reg4.N)';
```

WIth the new list of sample times, the output of Regime 4 can be resampled using linear interpolation.

```
reg4.pts    = interp1(reg4.out.t,reg4.out.y,reg4.t);
```

The actual framerate can be determined after the new list of samples is created.

```
 fprintf('Regime 4 has been resampled at a framerate of %2.2f FPS resulting in
%2.0f samples.', ...
     1/(reg4.t(2)-reg4.t(1)),reg4.N);
```

Animation

First, a new figure window is created and configured to be visible outside of this live script with the desired formatting. The animations for each regime will all show on the same figure.

```
anim = figure(1);
anim.Visible = 'On';
```

```
anim.Position = [600 200 800 600];
anim.Renderer = 'painters';
anim.Color = 'white';
```

Regime 1

This section loops through all of the points for Regime 1 while plotting each frame one by one.

```
for n=1:reg1.N

    % Find range of points to plot for trails.
    reg1.ind = reg1.out.t <= reg1.t(n);

    % Re-focus on previously defined figure
    figure(anim)

    % Plot the "ground"
    fill([-100 100 100 -100],[-100 -100 0 0],[0.5 0.5 0.5]);

    hold('on');

    % Plot the trail for each point of interest (E and G)
    plot(reg1.out.y(reg1.ind,9),
reg1.out.y(reg1.ind,10),':black','linewidth',1);

plot(reg1.out.y(reg1.ind,13),reg1.out.y(reg1.ind,14),':black','linewidth',2);

    % Plot the mechanism outline (D -> B -> C -> A)
    plot(reg1.pts(n,[7 3 5 1]),reg1.pts(n,[8 4 6 2]),'color',[0.75 0.75
0.75],'linewidth',3);

    % Plot the dots for each joint of the mechanism (A, B, C, E)
    plot(reg1.pts(n,[1 3 5 7 9]),reg1.pts(n,[2 4 6 8
10]),'o','MarkerFaceColor','black','MarkerSize',3);

    % Plot the string (D -> E)
    plot(reg1.pts(n,[7 9]),reg1.pts(n,[8 10]),'color',[0.75 0.75
0.75],'linewidth',1);

    % Plot the dot for the counterweight (G)

plot(reg1.pts(n,13),reg1.pts(n,14),'o','MarkerFaceColor','black','MarkerSize',6)
;

    % Plot the dot for the projectile (E)
```

```
plot(reg1.pts(n,9),reg1.pts(n,10),'o','MarkerFaceColor','yellow','MarkerSize',8)
;

     hold('off');

     xlabel('Horizontal Displacement, [ft]');
     ylabel('Vertical Displacement, [ft]');

     axis('equal');
     axis([-4 2 -1 5]);

     grid('on');

     drawnow;
 end
```

Regime 3

This section loops through all of the points for Regime 3 while plotting each frame one by one.

```
 for n=1:reg3.N

     % Find range of points to plot for trails.
     reg3.ind = reg3.out.t <= reg3.t(n);

     % Re-focus on previously defined figure
     figure(anim)

     % Plot the "ground"
     fill([-100 100 100 -100],[-100 -100 0 0],[0.5 0.5 0.5]);

     hold('on');

     % Plot the trail for each point of interest (E and G)
     plot(reg3.out.y(reg3.ind,9),
reg3.out.y(reg3.ind,10),':black','linewidth',1);

plot(reg3.out.y(reg3.ind,13),reg3.out.y(reg3.ind,14),':black','linewidth',2);

     % Plot the mechanism outline (D -> B -> C -> A)
     plot(reg3.pts(n,[7 3 5 1]),reg3.pts(n,[8 4 6 2]),'color',[0.75 0.75
0.75],'linewidth',3);

     % Plot the dots for each joint of the mechanism (A, B, C, E)
```

```
    plot(reg3.pts(n,[1 3 5 7 9]),reg3.pts(n,[2 4 6 8
10]),'o','MarkerFaceColor','black','MarkerSize',3);

    % Plot the string (D -> E)
    plot(reg3.pts(n,[7 9]),reg3.pts(n,[8 10]),'color',[0.75 0.75
0.75],'linewidth',1);

    % Plot the dot for the counterweight (G)

plot(reg3.pts(n,13),reg3.pts(n,14),'o','MarkerFaceColor','black','MarkerSize',6)
;

    % Plot the dot for the projectile (E)

plot(reg3.pts(n,9),reg3.pts(n,10),'o','MarkerFaceColor','yellow','MarkerSize',8)
;


    hold('off');

    xlabel('Horizontal Displacement, [ft]');
    ylabel('Vertical Displacement, [ft]');

    axis('equal');
    axis([-4 2 -1 5]);

    grid('on');

    drawnow;
end
```

Regime 4

This section loops through all of the points for Regime 3 while plotting each frame one by one.

```
for n=1:reg4.N

    % Find range of points to plot for trails.
    reg4.ind = reg4.out.t <= reg4.t(n);

    % Re-focus on previously defined figure
    figure(anim)

    % Plot the "ground"
    fill([-100 100 100 -100],[-100 -100 0 0],[0.5 0.5 0.5]);

    hold('on');
```

```
    % Plot the trail for each point of interest (E and G)
    plot(reg4.out.y(reg4.ind,9),
reg4.out.y(reg4.ind,10),':black','linewidth',1);

plot(reg4.out.y(reg4.ind,13),reg4.out.y(reg4.ind,14),':black','linewidth',2);

    % Plot the mechanism outline (D -> B -> C -> A)
    plot(reg4.pts(n,[7 3 5 1]),reg4.pts(n,[8 4 6 2]),'color',[0.75 0.75
0.75],'linewidth',3);

    % Plot the dots for each joint of the mechanism (A, B, C, E)
    plot(reg4.pts(n,[1 3 5 7 9]),reg4.pts(n,[2 4 6 8
10]),'o','MarkerFaceColor','black','MarkerSize',3);

    % Plot the string (D -> E)
    plot(reg4.pts(n,[7 9]),reg4.pts(n,[8 10]),'color',[0.75 0.75
0.75],'linewidth',1);

    % Plot the dot for the counterweight (G)

plot(reg4.pts(n,13),reg4.pts(n,14),'o','MarkerFaceColor','black','MarkerSize',6)
;

    % Plot the dot for the projectile (E)

plot(reg4.pts(n,9),reg4.pts(n,10),'o','MarkerFaceColor','yellow','MarkerSize',8)
;

    hold('off');

    xlabel('Horizontal Displacement, [ft]');
    ylabel('Vertical Displacement, [ft]');

    axis('equal');
    axis([-4 2 -1 5]);

    grid('on');

    drawnow;
 end
```

Discussion Questions

1. In this assignment you are not required to simulate the motion of the mechanism once the projectile departs. Describe how you could analyze the residual motion of the mechanism by simulating an additional regime of motion. In your description provide details about your proposed method, including, but not limited to, the number and selection of state variables.

After the projectile leaves, there is no point mass at E. This means that the link 3 can be ignored. Analysis could be done using links 1 and 2 and finding equations of motion for the links with a pivot point at A. This analysis would utilize the state variable xdot1, ydot1, xddot1, yddot1, xdot2, ydot2, xddot2, yddot2, thetadot1, thetaddot1, thetadot2, thetaddot2, thetadot3, and thetaddot3. As in regimes 1 and 3, kinematic analysis using these state vectors can be done, as well as kinetic analysis using Newton Euler method, FBD = KD. Constraints for this could be when the trebuchet comes to rest when point D touches the ground. Using this, a Simulink model and Matlab code could be created similar to Regime 3.

2. Determine the theoretical maximum distance that the projectile could travel under ideal conditions, assuming that the ground is flat and level. Compare this theoretical maximum distance to the actual distance that your simulated projectile travels. Why does your simulation throw the projectile a shorter distance than the theoretical maximum? What adjustments or tuning could be performed to maximize the travel distance?

I cannot give a value for maximum distance, however the theoretical value would be greater than the actual distance traveled. The trebuchet will not perfectly convert potential energy into kinetic energy, and this is why the projectile will not travel the theoretical maximum distance. Also the theoretical maximum might not be accounting for aerodynamic drag on the projectile in air, which would lead to a shorter distance. Adjusting to a more ideal drop angle, such as they do in the videos on Instructables, would lead to a more optimized system and greater distance. Optimizing the amount of weights put on the end of link 2 would also help with distance.

3. In regard to your simulation results, what percentage of the initial potential energy (of the whole system) is converted to kinetic energy of the projectile at the time of release? In other words, how efficient is the trebuchet at converting potential energy to kinetic energy? Considering your answer to the previous question, does maximum efficiency translate to maximum distance traveled?

I cannot give a value for this percentage. Maximum efficiency of the system does not totally translate to maximum distance. The system would still need to be optimized for ideal drop angle, as well as the proper amount of weights added. Maximum efficiency with unideal launch conditions would improve range certainly, however the other conditions must be ideal to reach the maximum distance.

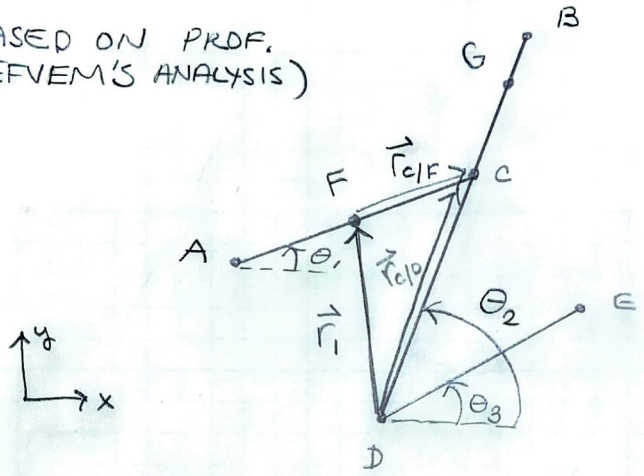
Hand Calculations

REGIME 1 FALLING

- SINGLE RIGID BODY

VECTOR LOOPS: LINK 1

(BASED ON PROF. REFVEM'S ANALYSIS)



$$\vec{r}_1 + \vec{r}_{C/F} = \vec{r}_{C/D}$$
$$\vec{r}_1 = \vec{r}_{C/D} - \vec{r}_{C/F}$$

COMPONENTS:

X - DIR

$$x_1 = r_{CD} c_2 - r_{CF} c_1$$
$$\dot{x}_1 = -r_{CD} s_2 \dot{\theta}_2 + r_{CF} s_1 \dot{\theta}_1$$
$$\ddot{x}_1 = -r_{CD} c_2 \dot{\theta}_2^2 - r_{CD} s_2 \ddot{\theta}_2 + r_{CF} c_1 \dot{\theta}_1^2 + r_{CF} s_1 \ddot{\theta}_1$$
$$-r_{CF} s_1 \ddot{\theta}_1 + r_{CD} s_2 \ddot{\theta}_2 + \ddot{x}_1 = -r_{CD} c_2 \dot{\theta}_2^2 + r_{CF} c_1 \dot{\theta}_1^2$$

Y - DIR

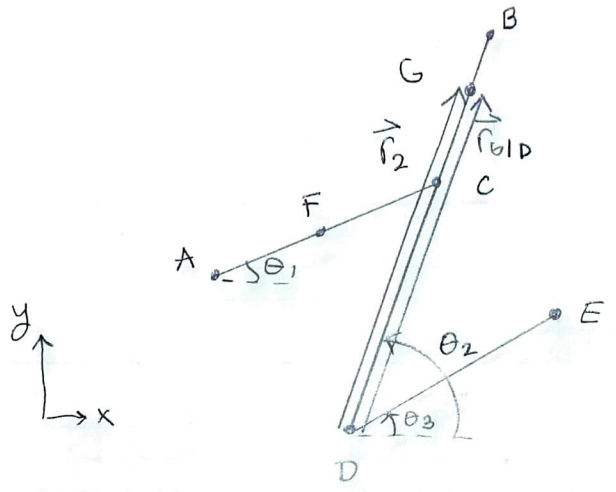
$$y_1 = r_{CD} s_2 - r_{CF} s_1$$
$$\dot{y}_1 = r_{CD} c_2 \dot{\theta}_2 - r_{CF} c_1 \dot{\theta}_1$$
$$\ddot{y}_1 = -r_{CD} s_2 \dot{\theta}_2^2 + r_{CD} c_2 \ddot{\theta}_2 + r_{CF} s_1 \dot{\theta}_1^2 - r_{CF} c_1 \ddot{\theta}_1$$
$$r_{CF} c_1 \ddot{\theta}_1 - r_{CD} c_2 \ddot{\theta}_2 + \ddot{y}_1 = -r_{CD} s_2 \dot{\theta}_2^2 + r_{CF} s_1 \dot{\theta}_1^2$$

NOTE: $c_1 = \cos\theta_1$ $\quad s_1 = \sin\theta_1$

$c_2 = \cos\theta_2$ $\quad s_2 = \sin\theta_2$

$c_3 = \cos\theta_3$ $\quad s_3 = \sin\theta_3$

VECTOR LOOPS CONTD: LINK 2



$$\vec{r}_2 = \vec{r}_{G/D}$$

COMPONENTS:

X-DIR:

$$x_2 = r_{DG} c_2$$

$$\dot{x}_2 = -r_{DG} s_2 \dot{\theta}_2$$

$$\ddot{x}_2 = -r_{DG} c_2 \dot{\theta}_2^2 - r_{DG} s_2 \ddot{\theta}_2$$

$$r_{DG} s_2 \ddot{\theta}_2 + \ddot{x}_2 = -r_{DG} c_2 \dot{\theta}_2^2$$

Y-DIR:

$$y_2 = r_{DG} s_2$$

$$\dot{y}_2 = r_{DG} c_2 \dot{\theta}_2$$

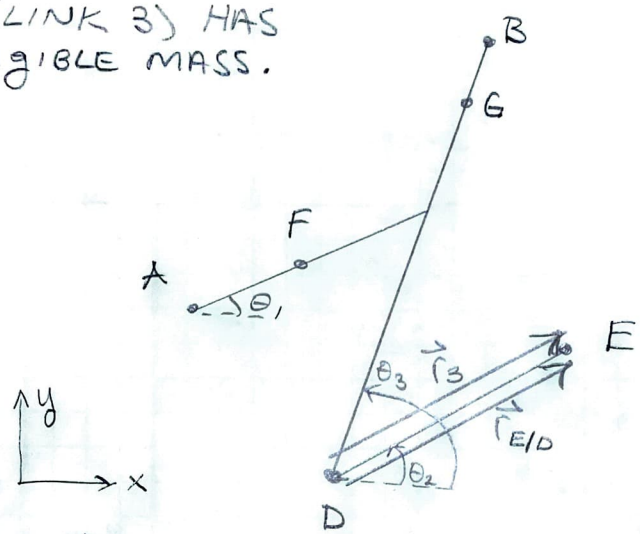
$$\ddot{y}_2 = -r_{DG} s_2 \dot{\theta}_2^2 + r_{DG} c_2 \ddot{\theta}_2$$

$$-r_{DG} c_2 \ddot{\theta}_2 + \ddot{y}_2 = -r_{DG} s_2 \dot{\theta}_2^2$$

VECTOR LOOPS CONTD. : LINK 3

POINT E IS A POINT MASS.

ASSUME: MEMBER DE (LINK 3) HAS NEGLIGIBLE MASS.



$$\vec{r}_3 = \vec{r}_{E/D}$$

COMPONENTS:

X-DIR:

$$x_3 = r_{DE} c_3$$

$$\dot{x}_3 = -r_{DE} s_3 \dot{\theta}_3$$

$$\ddot{x}_3 = -r_{DE} c_3 \dot{\theta}_3^2 - r_{DE} s_3 \ddot{\theta}_3$$

$$r_{DE} s_3 \ddot{\theta}_3 + \ddot{x}_3 = -r_{DE} c_3 \dot{\theta}_3^2$$

Y-DIR

$$y_3 = r_{DE} s_3$$

$$\dot{y}_3 = r_{DE} c_3 \dot{\theta}_3$$

$$\ddot{y}_3 = -r_{DE} s_3 \dot{\theta}_3^2 + r_{DE} c_3 \ddot{\theta}_3$$

$$-r_{DE} c_3 \ddot{\theta}_3 + \ddot{y}_3 = -r_{DE} s_3 \dot{\theta}_3^2$$

MATRICIES BEFORE IMPACT

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{x}_1 \\ \dot{y}_1 \\ \dot{x}_2 \\ \dot{y}_2 \\ \dot{x}_3 \\ \dot{y}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ r_{CF}S_1 & -r_{CD}S_2 & 0 \\ -r_{CF}C_1 & r_{CD}C_2 & 0 \\ 0 & -r_{DG}S_2 & 0 \\ 0 & r_{DG}C_2 & 0 \\ 0 & 0 & -r_{DE}S_3 \\ 0 & 0 & r_{DE}C_3 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}$$

$$(\dot{\vec{X}} = A\dot{\vec{x}})$$

$$\begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \\ \ddot{x}_1 \\ \ddot{y}_1 \\ \ddot{x}_2 \\ \ddot{y}_2 \\ \ddot{x}_3 \\ \ddot{y}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ r_{CF}S_1 & -r_{CD}S_2 & 0 \\ -r_{CF}C_1 & r_{CD}C_2 & 0 \\ 0 & -r_{DG}S_2 & 0 \\ 0 & r_{DG}C_2 & 0 \\ 0 & 0 & -r_{DE}S_3 \\ 0 & 0 & r_{DE}C_3 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix}$$

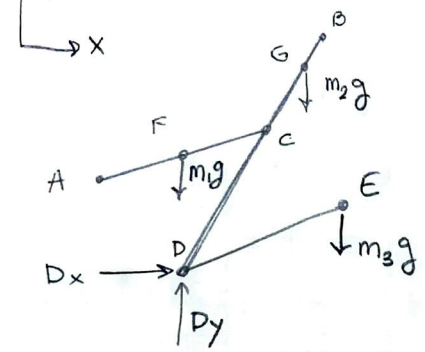
$$+ \begin{bmatrix} 0 \\ 0 \\ 0 \\ r_{CF}C_1\dot{\theta}_1^2 - r_{CD}C_2\dot{\theta}_2^2 \\ r_{CF}S_1\dot{\theta}_1^2 - r_{CD}S_2\dot{\theta}_2^2 \\ -r_{DG}C_2\dot{\theta}_2^2 \\ -r_{DG}S_2\dot{\theta}_2^2 \\ -r_{DE}C_3\dot{\theta}_3^2 \\ -r_{DE}S_3\dot{\theta}_3^2 \end{bmatrix}$$

$$(\ddot{\vec{X}} = A\ddot{\vec{x}} + \vec{a})$$

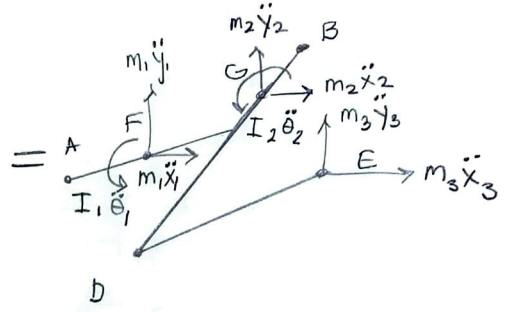
BEFORE IMPACT KINETICS

USING NEWTON - EULER

FBD



KD



SUM MOMENTS: FBD

$$\Sigma \vec{M}_D = \vec{r}_{F/D} \times \vec{\omega}_1 + \vec{r}_{G/D} \times \vec{\omega}_2 + \vec{r}_{E/D} \times \vec{\omega}_3$$

$$\Sigma M_D \hat{k} = ((r_{CD} c_2 - r_{CF} c_1)\hat{i} + (r_{CD} s_2 - r_{CF} s_1)\hat{j}) \times (-m_1 g \hat{j})$$
$$+ (r_{DG} c_2 \hat{i} + r_{DG} s_2 \hat{j}) \times (-m_2 g \hat{j})$$
$$+ (r_{DE} c_3 \hat{i} + r_{DE} s_3 \hat{j}) \times (-m_3 g \hat{j})$$

$$= -(r_{CD} c_2 - r_{CF} c_1) m_1 g - (r_{DG} c_2) m_2 g - (r_{DE} c_3) m_3 g$$

SUM MOMENTS: KD

$$\Sigma \vec{M}_D = \bar{\bar{I}}_1 \vec{\alpha}_1 + \vec{r}_{F/D} \times m_1 \vec{a}_F + \bar{\bar{I}}_2 \vec{\alpha}_2 + \vec{r}_{G/D} \times m_2 \vec{a}_G + \vec{r}_{E/D} \times m_2 \vec{a}_E$$

$$\Sigma M_D \hat{k} = I_1 \ddot{\theta}_1 \hat{k} + ((r_{CD} c_2 - r_{CF} c_1)\hat{i} + (r_{CD} s_2 - r_{CF} s_1)\hat{j}) \times m_1 (\ddot{x}_1 \hat{i} + \ddot{y}_1 \hat{j})$$
$$+ I_2 \ddot{\theta}_2 \hat{k} + (r_{DG} c_2 \hat{i} + r_{DG} s_2 \hat{j}) \times m_2 (\ddot{x}_2 \hat{i} + \ddot{y}_2 \hat{j})$$
$$+ (r_{DE} c_3 \hat{i} + r_{DE} s_3 \hat{j}) \times m_3 (\ddot{x}_3 \hat{i} + \ddot{y}_3 \hat{j})$$

$$\Sigma M_D = I_1 \ddot{\theta}_1 - (m_1 (r_{CD} s_2 - r_{CF} s_1)) \ddot{x}_1 + (m_1 (r_{CD} c_2 - r_{CF} c_1)) \ddot{y}_1$$
$$+ I_2 \ddot{\theta}_2 - (m_2 (r_{DG} s_2)) \ddot{x}_2 + (m_2 (r_{DG} c_2)) \ddot{y}_2$$
$$- (m_3 r_{DE} s_3) \ddot{x}_3 + (m_3 r_{DE} c_3) \ddot{y}_3$$

FALLING - CONSTRAINTS

$\ddot{\theta}_1 = \ddot{\theta}_2 = \ddot{\theta}_3$ (SINGLE RIGID BODY)

SUBSTITUTING THOSE VARIABLES,

$$\vec{f} = M\ddot{\vec{x}}$$

$$= M(A\ddot{\vec{x}} + \vec{a})$$

$$\ddot{\vec{x}} = (MA)^{-1}(\vec{f} - M\vec{a})$$

WHERE

$$f = \begin{bmatrix} -(r_{CE}s_2 - r_{CF}c_1)m_1 g - (r_{DG}c_2)m_2 g - (r_{DE}c_3)m_3 g \\ 0 \\ 0 \end{bmatrix}$$

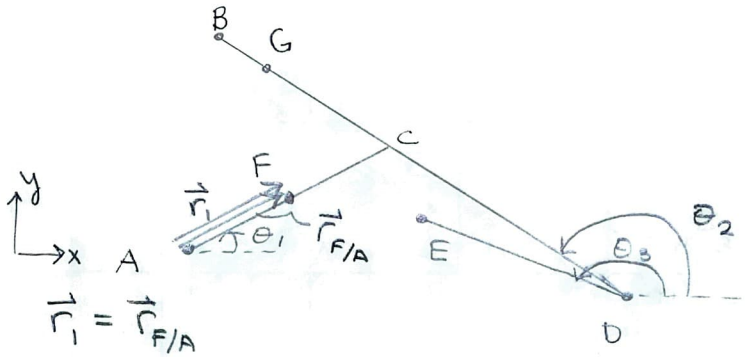
$$M = \begin{bmatrix} \bar{I}_1 & \bar{I}_2 & 0 & -m_1(r_{CD}s_2 - r_{CF}s_1) & m_1(r_{CD}c_2 - r_{CF}c_1) & -m_1 r_{DG}s_2 & m_2 r_{DG}c_2 & -m_3 r_{DE}s_3 & m_3 r_{DE}c_3 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

SOLVE FOR $\ddot{x}$.

REGIME 3 — SWINGING (AFTER IMPACT)

- POINT A FIXED ON GROUND

VECTOR LOOPS: LINK 1



$$\vec{r}_1 = \vec{r}_{F/A}$$

COMPONENTS:

X-DIR

$$x_1 = r_{AF} c_1$$

$$\dot{x}_1 = -r_{AF} s_1 \dot{\theta}_1$$

$$\ddot{x}_1 = -r_{AF} c_1 \dot{\theta}_1^2 - r_{AF} s_1 \ddot{\theta}_1$$

$$r_{AF} s_1 \ddot{\theta}_1 + \ddot{x}_1 = -r_{AF} c_1 \dot{\theta}_1^2$$

Y-DIR

$$y_1 = r_{AF} s_1$$

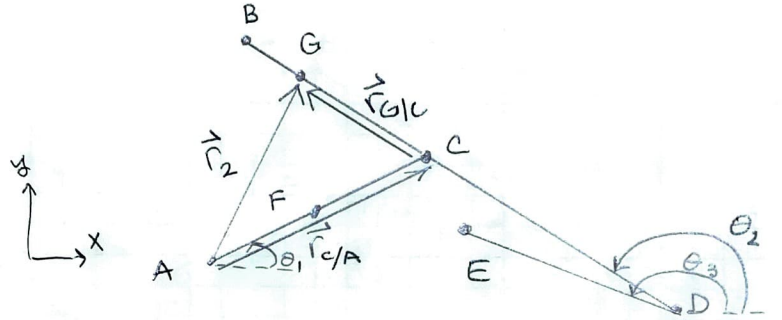
$$\dot{y}_1 = r_{AF} c_1 \dot{\theta}_1$$

$$\ddot{y}_1 = -r_{AF} s_1 \dot{\theta}_1^2 + r_{AF} c_1 \ddot{\theta}_1$$

$$-r_{AF} c_1 \ddot{\theta}_1 + \ddot{y}_1 = -r_{AF} s_1 \dot{\theta}_1^2$$

VECTOR LOOPS CONTD:

LINK 2



$$\vec{r}_2 = \vec{r}_{C/A} + \vec{r}_{G/C}$$

COMPONENTS:

X-DIR

$$x_2 = r_{AC} c_1 - r_{CG} s_2$$

$$\dot{x}_2 = -r_{AC} s_1 \dot{\theta}_1 - r_{CG} c_2 \dot{\theta}_2$$

$$\ddot{x}_2 = -r_{AC} c_1 \dot{\theta}_1^2 - r_{AC} s_1 \ddot{\theta}_1 + r_{CG} s_2 \dot{\theta}_2^2 - r_{CG} c_2 \ddot{\theta}_2$$

$$r_{AC} s_1 \ddot{\theta}_1 + r_{CG} c_2 \ddot{\theta}_2 + \ddot{x}_2 = -r_{AC} c_1 \dot{\theta}_1^2 + r_{CG} s_2 \dot{\theta}_2^2$$

$$y_2 = r_{AC} s_1 + r_{CG} c_2$$

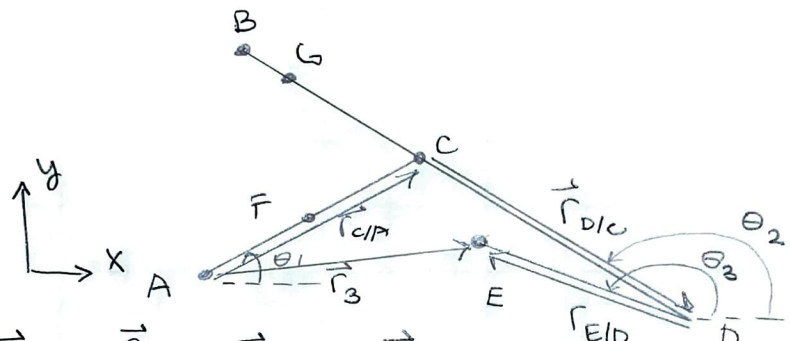
$$\dot{y}_2 = r_{AC} c_1 \dot{\theta}_1 - r_{CG} s_2 \dot{\theta}_2$$

$$\ddot{y}_2 = -r_{AC} s_1 \dot{\theta}_1^2 + r_{AC} c_1 \ddot{\theta}_1 - r_{CG} c_2 \dot{\theta}_2^2 - r_{CG} s_2 \ddot{\theta}_2$$

$$-r_{AC} c_1 \ddot{\theta}_1 + r_{CG} s_2 \ddot{\theta}_2 + \ddot{y}_2 = -r_{AC} s_1 \dot{\theta}_1^2 - r_{CG} c_2 \dot{\theta}_2^2$$

VECTOR LOOPS CONTD:

LINK 3



$$\vec{r}_3 = \vec{r}_{C/A} + \vec{r}_{D/C} + \vec{r}_{E/D}$$

COMPONENTS:

X-DIR

$$X_3 = r_{AC} c_1 + r_{CD} s_2 - r_{DE} s_3$$

$$\dot{X}_3 = -r_{AC} s_1 \dot{\theta}_1 + r_{CD} c_2 \dot{\theta}_2 - r_{DE} c_3 \dot{\theta}_3$$

$$\ddot{X}_3 = -r_{AC} c_1 \dot{\theta}_1^2 - r_{AC} s_1 \ddot{\theta}_1 - r_{CD} s_2 \dot{\theta}_2^2 + r_{CD} c_2 \ddot{\theta}_2 + r_{DE} s_3 \dot{\theta}_3^2 - r_{DE} c_3 \ddot{\theta}_3$$

$$r_{AC} s_1 \ddot{\theta}_1 - r_{CD} c_2 \ddot{\theta}_2 + r_{DE} c_3 \ddot{\theta}_3 + \ddot{X}_3 = -r_{AC} c_1 \dot{\theta}_1^2 - r_{CD} s_2 \dot{\theta}_2^2 + r_{DE} s_3 \dot{\theta}_3^2$$

Y-DIR

$$Y_3 = r_{AC} s_1 - r_{CD} c_2 + r_{DE} c_3$$

$$\dot{Y}_3 = r_{AC} c_1 \dot{\theta}_1 + r_{CD} s_2 \dot{\theta}_2 - r_{DE} s_3 \dot{\theta}_3$$

$$\ddot{Y}_3 = -r_{AC} s_1 \dot{\theta}_1^2 + r_{AC} c_1 \ddot{\theta}_1 + r_{CD} c_2 \dot{\theta}_2^2 + r_{CD} s_2 \ddot{\theta}_2 - r_{DE} c_3 \dot{\theta}_3^2 - r_{DE} s_3 \ddot{\theta}_3$$

$$-r_{AC} c_1 \ddot{\theta}_1 - r_{CD} s_2 \ddot{\theta}_2 + r_{DE} s_3 \ddot{\theta}_3 + \ddot{Y}_3 = -r_{AC} s_1 \dot{\theta}_1^2 + r_{CD} c_2 \dot{\theta}_2^2 - r_{DE} c_3 \dot{\theta}_3^2$$

MATRICIES AFTER IMPACT

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{x}_1 \\ \dot{y}_1 \\ \dot{x}_2 \\ \dot{y}_2 \\ \dot{x}_3 \\ \dot{y}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -r_{AF}s_1 & 0 & 0 \\ r_{AF}c_1 & 0 & 0 \\ -r_{AC}s_1 & -r_{CG}c_2 & 0 \\ r_{AC}c_1 & -r_{CG}s_2 & 0 \\ -r_{AC}s_1 & r_{CD}c_2 & -r_{DE}c_3 \\ r_{AC}c_1 & r_{CD}s_2 & -r_{DE}s_3 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}$$

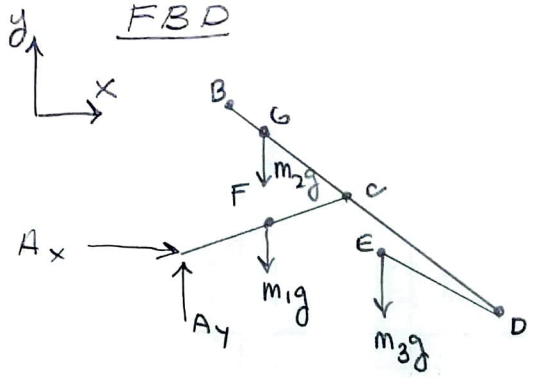
$(\dot{\vec{X}} = A\,\dot{\vec{x}})$

$$\begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \\ \ddot{x}_1 \\ \ddot{y}_1 \\ \ddot{x}_2 \\ \ddot{y}_2 \\ \ddot{x}_3 \\ \ddot{y}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -r_{AF}s_1 & 0 & 0 \\ r_{AF}c_1 & 0 & 0 \\ -r_{AC}s_1 & -r_{CG}c_2 & 0 \\ r_{AC}c_1 & -r_{CG}s_2 & 0 \\ -r_{AC}s_1 & r_{CD}c_2 & -r_{DE}c_3 \\ r_{AC}c_1 & r_{CD}s_2 & -r_{DE}s_3 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -r_{AF}c_1\dot{\theta}_1^2 \\ -r_{AF}s_1\dot{\theta}_1^2 \\ -r_{AC}c_1\dot{\theta}_1^2 + r_{CG}s_2\dot{\theta}_2^2 \\ -r_{AC}s_1\dot{\theta}_1^2 - r_{CG}c_2\dot{\theta}_2^2 \\ -r_{AC}c_1\dot{\theta}_1^2 - r_{CD}s_2\dot{\theta}_2^2 + r_{DE}s_3\dot{\theta}_3^2 \\ -r_{AC}s_1\dot{\theta}_1^2 + r_{CD}c_2\dot{\theta}_2^2 - r_{DE}c_3\dot{\theta}_3^2 \end{bmatrix}$$

$(\ddot{\vec{X}} = A\,\ddot{\vec{x}} + \vec{a})$

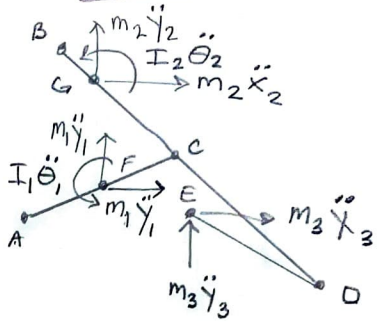
AFTER IMPACT KINETICS

FBD



=

KD



SUM MOMENTS: FBD

$$\Sigma \vec{M}_A = \vec{r}_{F/A} \times \vec{\omega}_1 + \vec{r}_{G/A} \times \vec{\omega}_2 + \vec{r}_{E/A} \times \vec{\omega}_3$$

$$\Sigma M_A \hat{k} = ((r_{AF} c_1 \hat{i}) + (r_{AF} s_1 \hat{j})) \times (-m_1 g \hat{j})$$
$$+ ((r_{AC} c_1 - r_{CG} s_2)\hat{i} + (r_{AC} s_1 + r_{CG} c_2)\hat{j}) \times (-m_2 g \hat{j})$$
$$+ ((r_{AC} c_1 + r_{CD} s_2 - r_{DE} s_2)\hat{i} + (r_{AC} s_1 - r_{CD} c_2 + r_{DE} c_2)\hat{j}) \times (-m_3 g \hat{j})$$

SUM MOMENTS: KD

$$\Sigma \vec{M}_A = \bar{\bar{I}}_1 \vec{\alpha}_1 + \vec{r}_{F/A} \times m_1 \vec{a}_F + \bar{\bar{I}}_2 \vec{\alpha}_2 + \vec{r}_{G/A} \times m_2 \vec{a}_G + \bar{\bar{I}}_3 \vec{\alpha}_3 + \vec{r}_{E/A} \times m_3 \vec{a}_E$$

$$\Sigma M_A \hat{k} = \bar{I}_1 \ddot{\theta}_1 \hat{k} + ((r_{AF} c_1)\hat{i} + (r_{AF} s_1)\hat{j}) \times m_1 (\ddot{x}_1 \hat{i} + \ddot{y}_1 \hat{j})$$
$$+ \bar{I}_2 \ddot{\theta}_2 \hat{k} + ((r_{AC} c_1 - r_{CG} s_2)\hat{i} + (r_{AC} s_1 + r_{CG} c_2)\hat{j}) \times m_2 (\ddot{x}_2 \hat{i} + \ddot{y}_2 \hat{j})$$
$$+ ((r_{AC} c_1 + r_{CD} s_2 - r_{DE} s_2)\hat{i} + (r_{AC} s_1 - r_{CD} c_2 + r_{DE} c_2)\hat{j}) \times m_3 (\ddot{x}_3 \hat{i} + \ddot{y}_3 \hat{j})$$

$$\Sigma M_A = \bar{I}_1 \ddot{\theta}_1 - (m_1 (r_{AF} s_1)) \ddot{x}_1 - (m_1 (r_{AF} c_1)) \ddot{y}_1$$
$$+ \bar{I}_2 \ddot{\theta}_2 - (m_2 (r_{AC} s_1 + r_{CG} c_2)) \ddot{x}_2 - (m_2 (r_{AC} c_1 - r_{CG} s_2)) \ddot{y}_2$$
$$+ \bar{I}_3 \ddot{\theta}_3 - (m_3 (r_{AC} s_1 - r_{CD} c_2 + r_{DE} c_2)) \ddot{x}_3 - (m_3 (r_{AC} c_1 + r_{CD} s_2 - r_{DE} s_2)) \ddot{y}_3$$

SWINGING - CONSTRAINTS

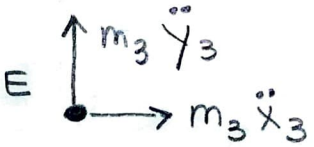
REGIME 3 ENDS AT THE RELEASE ANGLE, WHEN THE PROJECTILE BEGINS TO FLY. UNSURE OF HOW TO MODEL THIS STOP CONDITION.

REGIME 4 – FLYING

FBD

=

KD



$$\ddot{x}_3 = 0$$
$$\ddot{y}_3 = g$$

NEED PROJECTILE MOTION EQNS:

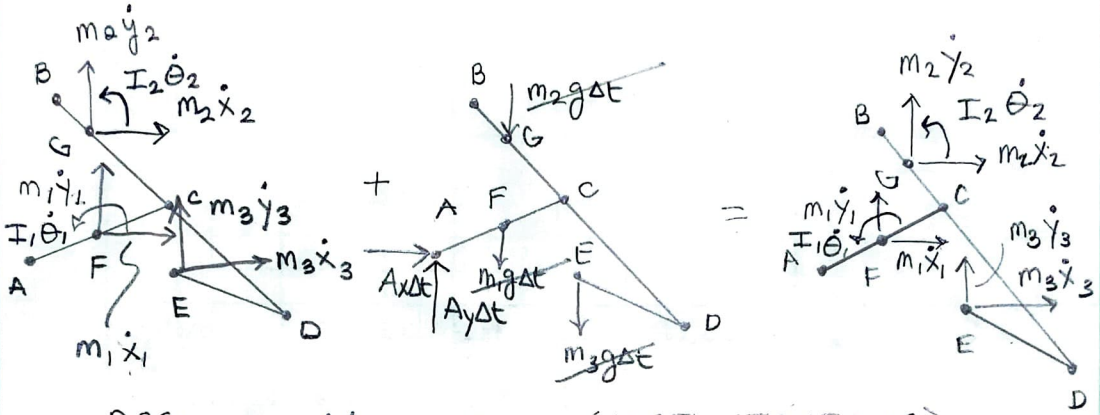
$$x_E = x_i + \dot{x}t + \tfrac{1}{2}(\ddot{x}t^2)$$
$$= x_i + \dot{x}t$$
$$y_E = y_i + \dot{y}t + \tfrac{1}{2}(\ddot{y}t^2)$$

$$t = -\dot{y} \pm \sqrt{\frac{\dot{y}^2 - 2g y_i}{-g}}$$

REGIME 2 : IMPACT

IMPULSE MOMENTUM METHOD

NOTE : A IS FIXED POINT OF ROTATION,
D RELEASES FROM GROUND

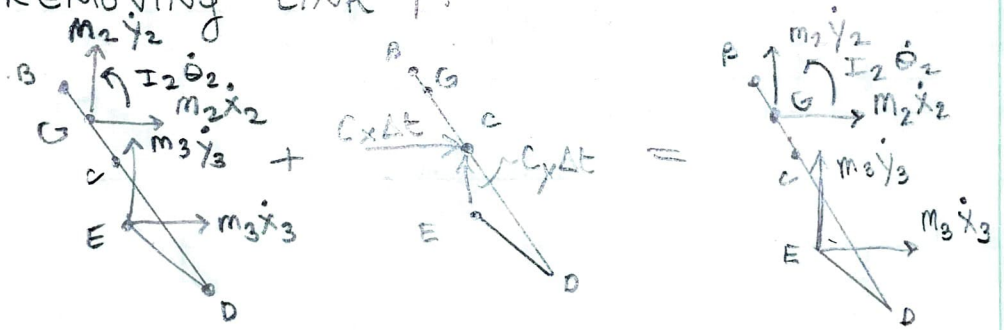


ASSUME $\Delta t \to 0$ (INSTANTANEOUS)

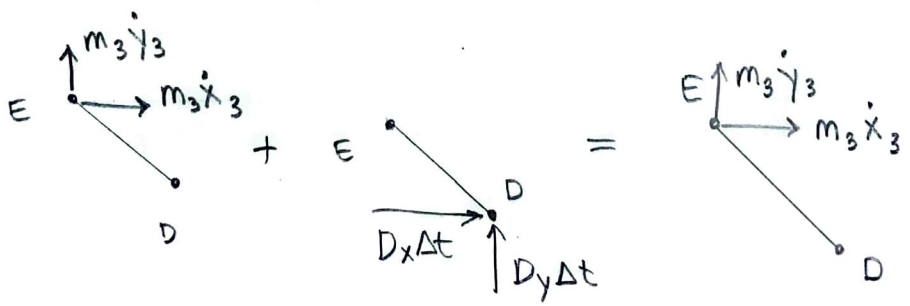
$$H_A + \int_0 \Sigma M_A \, dt = H_A'$$

$$H_A = H_A'$$

REMOVING LINK 1 :



REMOVE LINK 2 ALSO :



IMPACT

EQNS: $H_A = H_A'$ ← WHOLE MECHANISM

$H_C = H_C'$ ← LINK 2 AND 3

$H_D = H_D'$ ← LINK 3

KINEMATICS:

$$\vec{H} = M\dot{\vec{X}}$$

WHERE $H = \begin{bmatrix} H_A \\ H_C \\ H_D \end{bmatrix}$

M FOUND THROUGH ΣMOMENTS

WHOLE MECHANISM:

ΣM_A

LINKS 2 & 3:

ΣM_C

LINK 3:

ΣM_D

Other Required Files

Regime 1 Equations of Motion

```
%
% This function is responsible for determining the time rate-of-change of
% the state vector representing the angle of each link making up the
% walking arm trebuchet during Regime 1.
%
% Regime 1 is the falling regime in which the entire mechanism falls until
% point A impacts with the ground.
%
% This function takes in the following parameters
%   x           The state vector representing the angle of each link
%   x_dot       The velocity vector representing the angular velocity of
%               each link.
%   P           A MATLAB struct containing all of the constant parameters
%
% The function returns the following parameters
%   x_ddot      The acceleration vector representing the angular
%               acceleration of each link.
%   stp_cnd     The stop condition for the simulation. Turns true when
%               point A hits the ground.
%
function [x_ddot, stp_cnd] = reg_1_EOM(x, x_dot, P)

% First, all the angles in the state vector are used to define the sin and
% cos terms needed below.
c1 = cos(x(1)); s1 = sin(x(1)); thd1 = x_dot(1);
c2 = cos(x(2)); s2 = sin(x(2)); thd2 = x_dot(2);
c3 = cos(x(3)); s3 = sin(x(3)); thd3 = x_dot(3);

% Next, the matrix A is defined. This is the same A matrix used in Regime
% 2.
A = [   1           0           0
        0           1           0
        0           0           1
        P.r_CF*s1  -P.r_CD*s2   0
       -P.r_CF*c1   P.r_CD*c2   0
        0          -P.r_DG*s2   0
        0           P.r_DG*c2   0
        0           0          -P.r_DE*s3
        0           0           P.r_DE*c3 ];

% Then, the vector a is defined. This vector contains the left over terms
```

```
 % not accounted for by the A matrix.
 a = [   0
         0
         0
         P.r_CF*c1*thd1^2 - P.r_CD*c2*thd2^2
         P.r_CF*s1*thd1^2 - P.r_CD*s2*thd2^2
        -P.r_DG*c2*thd2^2
        -P.r_DG*s2*thd2^2
        -P.r_DE*c3*thd3^2
        -P.r_DE*s3*thd3^2 ];

 % Then, the matrix M is defined. This matrix combines all of the angular
 % and linear accelerations for the "I alpha + r x ma" side of Euler's law.
 M = zeros(3,9);
 % Row 1 (sum of moments about point D)
 M(1,1) =  P.I_1;
 M(1,2) =  P.I_2;
 M(1,4) = -P.m_1*(-P.r_CF*s1 + P.r_CD*s2);
 M(1,5) =  P.m_1*(-P.r_CF*c1 + P.r_CD*c2);
 M(1,6) = -P.m_2*(P.r_DG*s2);
 M(1,7) =  P.m_2*(P.r_DG*c2);
 M(1,8) = -P.m_3*(P.r_DE*s3);
 M(1,9) =  P.m_3*(P.r_DE*c3);
 % Row 2 (alpha1 = alpha2)
 M(2,1) =  1;
 M(2,2) = -1;
 % Row 3 (alpha1 = alpha3)
 M(3,1) =  1;
 M(3,3) = -1;

 % Then, the vector f is defined. This vector represents all the external
 % forces causing moments during the falling regime.
 f = [   -P.m_1*P.g*(-P.r_CF*c1 + P.r_CD*c2) - P.m_2*P.g*(P.r_DG*c2) -
P.m_3*P.g*(P.r_DE*c3)
          0
          0 ];

 % Finally, the acceleration vector is found
 x_ddot = (M*A)\(f - M*a);

 % The stop condition becomes true when the vertical displacement of point A
 % becomes negative indicating it has hit the ground.
 stp_cnd = (P.r_CD*s2-P.r_AC*s1) <= 0;
```

Regime 1 Output Equations

```
%
% This function is responsible for converting the state variables in x to
% the x- and y-coordinates for each point in the mechanism.
%
% This function takes in the following parameters
%   x           The state vector representing the angle of each link
%   P           A MATLAB struct containing all of the constant parameters
%
% The function returns the following parameters
%   y           The output vector containing the x- and y- coordinates for
%               each point in the mechanism
%
function y  = reg_1_output(x, P)

y = zeros(14,1);

c1 = cos(x(1)); s1 = sin(x(1));
c2 = cos(x(2)); s2 = sin(x(2));
c3 = cos(x(3)); s3 = sin(x(3));

% Point D is the origin
y(7)  = 0;
y(8)  = 0;

% Find point B from point D
y(3)  = y(7) + P.r_BD*c2;
y(4)  = y(8) + P.r_BD*s2;

% Find point C from point D
y(5)  = y(7) + P.r_CD*c2;
y(6)  = y(8) + P.r_CD*s2;

% Find point A from point C
y(1)  = y(5) - P.r_AC*c1;
y(2)  = y(6) - P.r_AC*s1;

% Find point E from point D
y(9)  = y(7) + P.r_DE*c3;
y(10) = y(8) + P.r_DE*s3;

% Find point F from point C
y(11) = y(5) - P.r_CF*c1;
y(12) = y(6) - P.r_CF*s1;

% Find point G
```

```
y(13) = y(7) + P.r_DG*c2;
y(14) = y(8) + P.r_DG*s2;
```

Walking Arm Trebuchet - Regime 2 Impact Analysis

Original Author: Charlie Refvem

Last Modified: 05/04/2020

Description:

This function analyzes the impact of the walking arm trebuchet with the ground. The function relates the state of the system just before impact (JBI) to the state of the system just after impact, (JAI). This relationship is determined by considering the principle of impulse and momentum during the impact when point A hits the ground.

Three seperate equations will be required to relate the three angular velocities just after impact to the three angular velocities just before impact. These three equations are found by stragetically using the principle of impulse and moment for three different subsets of the mechanism.

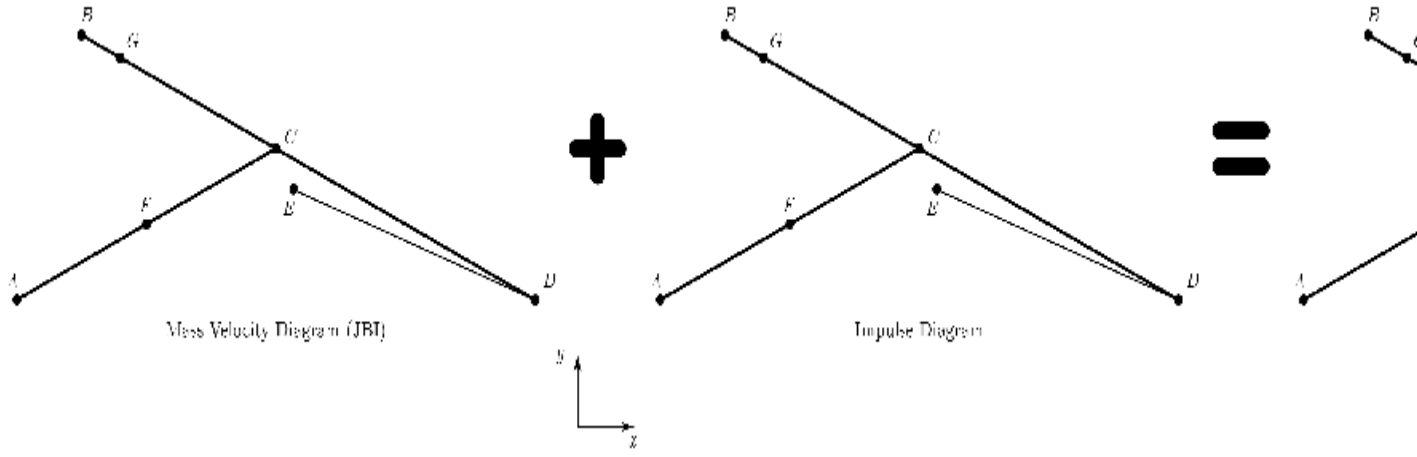
Equation 1

Equation 1 is found by applying the principle of impulse and momentum to the entire mechanism and then summing moments about the contact point A. This will avoid reaction forces at point A from showing up in Equation 1.

System: Link 1, Link 2, and Link 3

JBI: Before impact, point D is in contact with the ground and remains fixed.

JAI: After impact, point A is in contact with the ground and remains fixed.



$$\vec{H}_{A(JBI)} + \int_{JBI}^{JAI} M_A \, dt = \vec{H}_{A(JAI)}$$

$$\vec{H}_{A(JBI)} = \vec{H}_{A(JAI)} \quad (1)$$

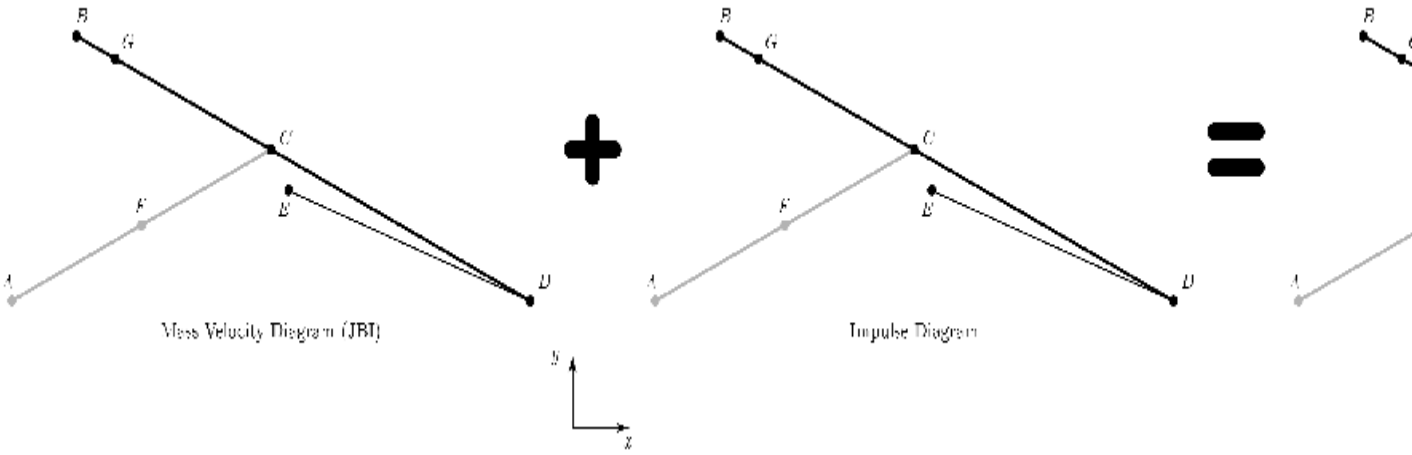
Equation 2

Equation 2 is found by applying the principle of impulse and momentum to just links 2 and 3 and then summing moments about point C. This will avoid reaction forces at point C from showing up in Equation 2.

System: Link 2, and Link 3

JBI: Before impact, point D is in contact with the ground and remains fixed.

JAI: After impact, point A is in contact with the ground and remains fixed.



$$\vec{H}_{C(JBI)} + \int_{JBI}^{JAI} M_C \, dt = \vec{H}_{C(JAI)}$$

$$\vec{H}_{C(JBI)} = \vec{H}_{C(JAI)} \quad (2)$$

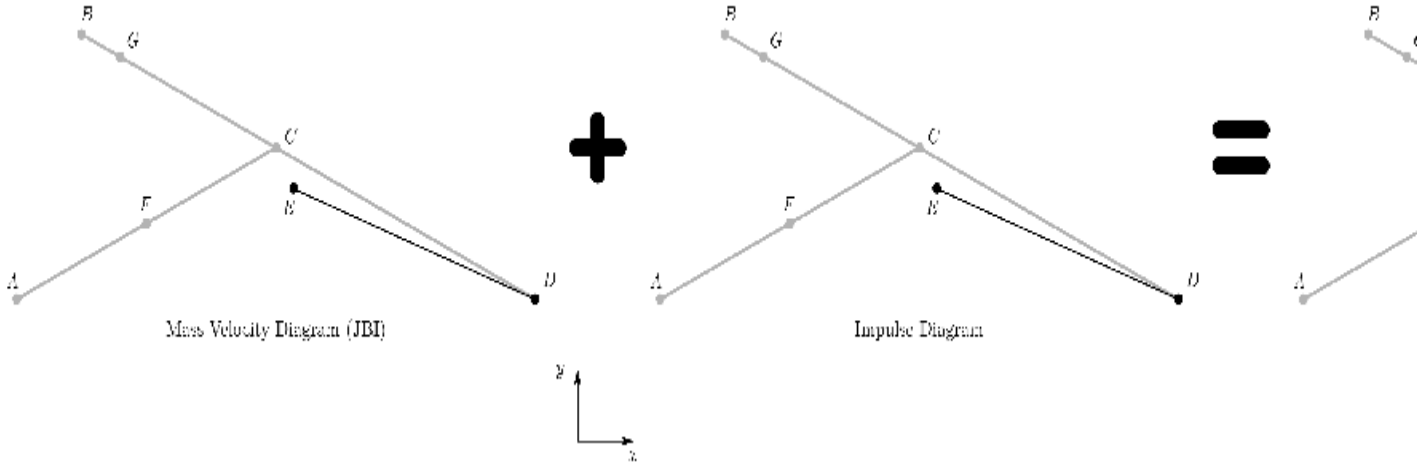
Equation 3

Equation 3 is found by applying the principle of impulse and momentum to just link 3 and then summing moments about point D. This will avoid reaction forces at point D from showing up in Equation 3.

System: Link 3

JBI: Before impact, point D is in contact with the ground and remains fixed.

JAI: After impact, point A is in contact with the ground and remains fixed.



$$\vec{H}_{D(JBI)} + \int_{JBI}^{JAI} M_D dt = \vec{H}_{D(JAI)}$$

$$\vec{H}_{D(JBI)} = \vec{H}_{D(JAI)} \quad (3)$$

Solution Approach

Setting up and solving the three equations derived above requires many more than three intermediate equations. First, the system kinematics must be analyzed, both before impact and after impact. Next, the moments must be calcualted using the principle of impulse and momentum. Fortunately, all of these relationships are linear and can be expressed as matrix equations that can be solved using MATLAB.

In the following analysis the derivative of the state vector will be referred to simply as the "velocity vector" and represented by the symbol $\dot{\mathbf{x}}$. An intermediate variable will be defined as $\dot{\boldsymbol{\chi}}$ and referred to as the "augmented velocity vector". The velocity vector, $\dot{\mathbf{x}}$, contains only the angular velocities for each link, but the augmented velocity vector, $\dot{\boldsymbol{\chi}}$, contains the linear velocities as well. Explicitly, $\dot{\mathbf{x}} = \begin{bmatrix} \dot{\theta}_1 & \dot{\theta}_2 & \dot{\theta}_3 \end{bmatrix}^\top$, and $\dot{\boldsymbol{\chi}} = \begin{bmatrix} \dot{\theta}_1 & \dot{\theta}_2 & \dot{\theta}_3 & \dot{x}_1 & \dot{y}_1 & \dot{x}_2 & \dot{y}_2 & \dot{x}_3 & \dot{y}_3 \end{bmatrix}^\top$.

The system kinematics can be analyzed using the vector loop method. Using the vector loop method, matrices A and B can be determined that will relate the velocity vector $\dot{\mathbf{x}}$ containing only angular velocites, to an augmented velocity vector $\dot{\boldsymbol{\chi}}$ containing the angular velocities and linear velocities for the JBI and JAI cases respectively.

Kinematics JBI

Before impact, point D is fixed in contact with the ground.

$$\dot{\boldsymbol{\chi}}_{JBI} = A\dot{\mathbf{x}}_{JBI}$$

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{x}_1 \\ \dot{y}_1 \\ \dot{x}_2 \\ \dot{y}_2 \\ \dot{x}_3 \\ \dot{y}_3 \end{bmatrix}_{JBI} = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \\ A_{41} & A_{42} & A_{43} \\ A_{51} & A_{52} & A_{53} \\ A_{61} & A_{62} & A_{63} \\ A_{71} & A_{72} & A_{73} \\ A_{81} & A_{82} & A_{83} \\ A_{91} & A_{92} & A_{93} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}_{JBI} \quad (4)$$

Kinematics JAI

After impact, point A is fixed in contact with the ground.

$$\dot{\boldsymbol{\chi}}_{JAI} = B\dot{\mathbf{x}}_{JAI}$$

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{x}_1 \\ \dot{y}_1 \\ \dot{x}_2 \\ \dot{y}_2 \\ \dot{x}_3 \\ \dot{y}_3 \end{bmatrix}_{JAI} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \\ B_{41} & B_{42} & B_{43} \\ B_{51} & B_{52} & B_{53} \\ B_{61} & B_{62} & B_{63} \\ B_{71} & B_{72} & B_{73} \\ B_{81} & B_{82} & B_{83} \\ B_{91} & B_{92} & B_{93} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix}_{JAI} \quad (5)$$

Summation of Moments

With the augmented velocity vectors defined, before and after impact, the angular momentum can be determined. Consider a vector $\mathbf{H}$, which contains the values of angular momentum that are conserved during impact. That is:

$$\mathbf{H} = \begin{bmatrix} H_A \\ H_C \\ H_D \end{bmatrix} \begin{matrix} \text{(Link 1, Link 2, and Link 3)} \\ \text{(Link 2 and Link 3)} \\ \text{(Link 3)} \end{matrix}$$

The vector $\mathbf{H}$ can be related to the augmented velocity vector using another matrix M.

$$\mathbf{H} = M\dot{\boldsymbol{\chi}}$$

$$\begin{bmatrix} H_A \\ H_B \\ H_C \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} & M_{15} & M_{16} & M_{17} & M_{18} & M_{19} \\ M_{21} & M_{22} & M_{23} & M_{24} & M_{25} & M_{26} & M_{27} & M_{28} & M_{29} \\ M_{31} & M_{32} & M_{33} & M_{34} & M_{35} & M_{36} & M_{37} & M_{38} & M_{39} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{x}_1 \\ \dot{y}_1 \\ \dot{x}_2 \\ \dot{y}_2 \\ \dot{x}_3 \\ \dot{y}_3 \end{bmatrix}_{JAI} \quad (6)$$

Equation 1 through Equation 6 can be combined and solved to relate the velocity vector after impact, $\dot{\mathbf{x}}_{JAI}$, to the velocity vector before impact, $\dot{\mathbf{x}}_{JBI}$.

$$\begin{aligned} \mathbf{H}_{JBI} &= \mathbf{H}_{JAI} \\ M\,\dot{\boldsymbol{\chi}}_{JBI} &= M\,\dot{\boldsymbol{\chi}}_{JAI} \\ M\,B\dot{\mathbf{x}}_{JBI} &= M\,A\dot{\mathbf{x}}_{JAI} \\ \dot{\mathbf{x}}_{JAI} &= (M\,B)^{-1}M\,A\dot{\mathbf{x}}_{JBI} \quad (7) \end{aligned}$$

So, to find $\dot{\mathbf{x}}_{JAI}$, all that needs to be done is to define the matrices A, B, and M and then to solve Equation 7 using MATLAB . None of the equations need to be solved by hand.

Matlab Implementation

This function takes in the following parameters

x_JBI	The state vector just before impact
x_dot_JBI	The velocity vector just before impact
P	A MATLAB struct containing all of the constant parameters

The function returns the following parameters

x_JAI	The state vector just after impact
x_dot_JAI	The velocity vector just after impact

```
function [x_jai, x_dot_jai] = jbi2jai(x_jbi, x_dot_jbi, P)
```

First, all the angles in the state vector are used to define the sin and cos terms used in A, B, and M.

```
th1 = x_jbi(1); s1 = sin(th1);  c1 = cos(th1);
th2 = x_jbi(2); s2 = sin(th2);  c2 = cos(th2);
th3 = x_jbi(3); s3 = sin(th3);  c3 = cos(th3);
```

Next, the A matrix is defined.

```
A = [   1           0           0
        0           1           0
        0           0           1
        P.r_CF*s1  -P.r_CD*s2   0
       -P.r_CF*c1   P.r_CD*c2   0
        0          -P.r_DG*s2   0
        0           P.r_DG*c2   0
        0           0          -P.r_DE*s3
        0           0           P.r_DE*c3 ];
```

Then, the B matrix is defined.

```
B = [   1           0           0
        0           1           0
        0           0           1
        -P.r_AF*s1  0           0
        P.r_AF*s1   0           0
        -P.r_AC*s1  -P.r_CG*c2  0
        P.r_AC*c1   -P.r_CG*s2  0
        -P.r_AC*s1  P.r_CD*c2   -P.r_DE*c3
        P.r_AC*c1   P.r_CD*s2   -P.r_DE*s3 ];
```

The definition for the M matrix does not fit as well on screen so it will be filled in entry by entry.

```
M = zeros(3,9);
% Row 1 (moments about A for link 1, link 2, and link 3)
M(1,1) =  1;                                            % <------
% Row 2 (moments about C for link 2 and link 3)
M(2,2) =  1;                                            % <------
% Row 3 (moments about D for link 3)
M(3,3) =  1;                                            % <------
```

Finally, the state after impact is related to the state before impact. Because the impact occurs instantaneously, the state vector does not change.

```
x_jai     = x_jbi;
```

However, the velocity vector does change according to Equation 7 as defined above.

```
x_dot_jai = (M*B)\M*A*x_dot_jbi;
end
```

Regime 3 equations of motion

```
%
% This function is responsible for determining the time rate-of-change of
% the state vector representing the angle of each link making up the
% walking arm trebuchet during Regime 3.
%
% Regime 3 is the swinging regime in which all three links move with
% independant velocities.
%
% This function takes in the following parameters
%   x           The state vector representing the angle of each link
%   x_dot       The velocity vector representing the angular velocity of
%               each link.
%   P           A MATLAB struct containing all of the constant parameters
%
% The function returns the following parameters
%   x_ddot      The acceleration vector representing the angular
%               acceleration of each link.
%   stp_cnd     The stop condition for the simulation. Turns true when
%               the projectile releases from the sling.
%
function [x_ddot, stp_cnd] = reg_3_EOM(x, x_dot, P)

% First, all the angles in the state vector are used to define the sin and
% cos terms needed below.
c1 = cos(x(1)); s1 = sin(x(1)); thd1 = x_dot(1);
c2 = cos(x(2)); s2 = sin(x(2)); thd2 = x_dot(2);
c3 = cos(x(3)); s3 = sin(x(3)); thd3 = x_dot(3);

% Next, the matrix B is defined. This is the same B matrix used in Regime
% 2.
B = [   1           0           0
        0           1           0
        0           0           1
        -P.r_AF*s1  0           0
        P.r_AF*s1   0           0
        -P.r_AC*s1  -P.r_CG*c2  0
        P.r_AC*c1   -P.r_CG*s2  0
        -P.r_AC*s1  P.r_CD*c2   -P.r_DE*c3
        P.r_AC*c1   P.r_CD*s2   -P.r_DE*s3 ];
```

```
% Then, the vector b is defined. This vector contains the left over terms
% not accounted for by the B matrix.
b = [   0
        0
        0
        -P.r_AF*c1*thd1^2
        -P.r_AF*s1*thd1^2
        -P.r_AC*c1*thd1^2 + P.r_CG*s2*thd2^2
        -P.r_AC*s1*thd1^2 - P.r_CG*c2*thd2^2
        -P.r_AC*c1*thd1^2 - P.r_CD*s2*thd2^2 + P.r_DE*s3*thd3^2
        -P.r_AC*s1*thd1^2 + P.r_CD*c2*thd2^2 - P.r_DE*c1*thd3^2];

% Then, the matrix M is defined. This matrix combines all of the angular
% and linear accelerations for the "I alpha + r x ma" side of Euler's law.
M = zeros(3,9);
% Row 1 (sum of moments about point A)
M(1,1) =  P.I_1;
M(1,2) =  P.I_2;
M(1,4) = -P.m_1*(-P.r_CF*s1 + P.r_CD*s2);
M(1,5) =  P.m_1*(-P.r_CF*c1 + P.r_CD*c2);
M(1,6) = -P.m_2*(P.r_DG*s2);
M(1,7) =  P.m_2*(P.r_DG*c2);
M(1,8) = -P.m_3*(P.r_DE*s3);
M(1,9) =  P.m_3*(P.r_DE*c3);
% Row 2 (alpha1 = alpha2)
M(2,1) =  1;
M(2,2) = -1;
% Row 3 (alpha1 = alpha3)
M(3,1) =  1;
M(3,3) = -1;
% Then, the vector g is defined. This vector represents all the external
% forces causing moments during the falling regime.
g = [   -P.m_1*P.g
        -P.m_2*P.g
        -P.m_3*P.g ];

% Finally, the acceleration vector is found
x_ddot = (M*B)\(g-M*b);

% The stop condition becomes true when the angle between link 2 and link 3
% exceeds the selected release angle.
stp_cnd = true;
```

Regime 3 Output Equations

```
%
```

```
% This function is responsible for converting the state variables in x to
% the x- and y-coordinates for each point in the mechanism.
%
% This function takes in the following parameters
%   x           The state vector representing the angle of each link
%   P           A MATLAB struct containing all of the constant parameters
%
% The function returns the following parameters
%   y           The output vector containing the x- and y- coordinates for
%               each point in the mechanism
%
function y  = reg_3_output(x, P)

y = zeros(14,1);

c1 = cos(x(1)); s1 = sin(x(1));
c2 = cos(x(2)); s2 = sin(x(2));
c3 = cos(x(3)); s3 = sin(x(3));

% Point A
y(1)  = 0;                                                  % <------
y(2)  = 0;                                                  % <------

% Point B
y(3)  = 0;
y(4)  = 0;

% Point C
y(5)  = 0;                                                  % <------
y(6)  = 0;                                                  % <------

% Point D
y(7)  = 0;                                                  % <------
y(8)  = 0;                                                  % <------

% Point E
y(9)  = 0;                                                  % <------
y(10) = 0;                                                  % <------

% Point F
y(11) = 0;                                                  % <------
y(12) = 0;                                                  % <------

% Point G
y(13) = 0;                                                  % <------
y(14) = 0;                                                  % <------
```

```
% Point D is the origin
y(7)  = 0;
y(8)  = 0;

%% Attempting to use coordinates from Reg 1

% Find point B from point D
y(3)  = y(7) + P.r_BD*c2;
y(4)  = y(8) + P.r_BD*s2;

% Find point C from point D
y(5)  = y(7) + P.r_CD*c2;
y(6)  = y(8) + P.r_CD*s2;

% Find point A from point C
y(1)  = y(5) - P.r_AC*c1;
y(2)  = y(6) - P.r_AC*s1;

% Find point E from point D
y(9)  = y(7) + P.r_DE*c3;
y(10) = y(8) + P.r_DE*s3;

% Find point F from point C
y(11) = y(5) - P.r_CF*c1;
y(12) = y(6) - P.r_CF*s1;

% Find point G
y(13) = y(7) + P.r_DG*c2;
y(14) = y(8) + P.r_DG*s2;
```

Regime 4 equations of motion

```
%
% This function is responsible for determining the time rate-of-change of
% the state vector representing the location of the projectile during
% Regime 4.
%
% Regime 4 is the flying regime in which the projectile is flying through
% the air.
%
% This function takes in the following parameters
%   x           The state vector representing the x and y displacement of
%               the projectile.
%   x_dot       The velocity vector representing the x and y velocity of
%               the projectile.
%   P           A MATLAB struct containing all of the constant parameters
```

```
%
% The function returns the following parameters
%   x_ddot      The acceleration vector representing the x and y
%               acceleration of the projectile.
%   stp_cnd     The stop condition for the simulation. Turns true when
%               the projectile hits the ground.
%
function [x_ddot, stp_cnd] = reg_4_EOM(x, x_dot, P)

% For this regime the acceleration equations are nearly trivial. The
% projectile is simply a particle acted upon by gravity alone.
x_ddot = [ 0
           P.g ];

% The stop condition becomes true when the vertical displacement of point E
% become negative indicating that the projectile has hit the ground.
stp_cnd = true;
%% Regime 4 output equations
%
% This function is responsible for converting the state variables in x to
% the x- and y-coordinates for each point in the mechanism.
%
% This function takes in the following parameters
%   x           The state vector representing the x- and y- coordinates of
%               the projectile.
%   P           A MATLAB struct containing all of the constant parameters
%
% The function returns the following parameters
%   y           The output vector containing the x- and y- coordinates for
%               each point in the mechanism
%
function y  = reg_4_output(x, P)

y = zeros(14,1);
% Please note I was unable to find the following values for simulation
% Point A
y(1)  = 0;                                                  % <------
y(2)  = 0;                                                  % <------

% Point B
y(3)  = 0;                                                  % <------
y(4)  = 0;                                                  % <------

% Point C
y(7)  = 0;                                                  % <------
y(8)  = 0;                                                  % <------
```

```
% Point D
y(5)  = 0;                                                  % <------
y(6)  = 0;                                                  % <------

% Point E
y(9)  = 0;                                                  % <------
y(10) = 0;                                                  % <------

% Point F
y(11) = 0;                                                  % <------
y(12) = 0;                                                  % <------

% Point G
y(13) = 0;                                                  % <------
y(14) = 0;                                                  % <------
```