Power Equation Analysis of the Watt-II Six-Bar Linkage

Trevor J. Ladner

Purdue University, School of Mechanical Engineering

ME 45200, Fall 2021

1. INTRODUCTION

This report is an analysis of the energy and power contributions due to kinetic, gravitational potential, spring potential, and viscous damping effects on the Watt-II 6 Bar Linkage with a constant angular input of 25 rad/s. Gravity is acting in the negative-Y direction. There is a spring that connects pin O2 and point P and a viscous damper attached at point P parallel to the X-axis.

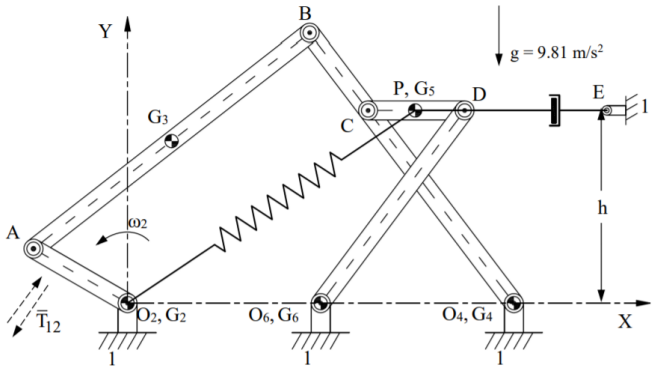

Figure 1. Diagram of the Watt-II 6-Bar linkage used for analysis of the mechanism.

The given lengths for each of the link are as follows: $O_2O_4 = 203.20$ mm, $O_2A = 57.15$ mm, $AB = 184.15$ mm, $O_4B = 177.80$ mm, $CD = 50.80$ mm, $O_6D = 127.00$ mm, $O_6O_4 = 101.60$ mm, $O_4C = 127.00$ mm, and $CP = PD = 25.40$ mm. The masses for the given links are $m_2 = 2$ kg, $m_3 = 5.5$ kg, $m_4 = 7.5$ kg, $m_5 = 1.5$ kg, and $m_6 = 6$ kg. The center of mass for links 2, 4, and 6 are at the pin connected to the ground. The center of mass for link 3 is at the midpoint. The center of mass for link 5 is at the coupler point P. The second moments of mass for each link are: $I_{G2} = 0.0067\ kgm^2$, $I_{G3} = 0.0433\ kgm^2$, $I_{G4} = 0.2426\ kgm^2$, $I_{G5} = 0.0009\ kgm^2$, and $I_{G6} = 0.0634\ kgm^2$. The linear spring has a spring rate of k = 5000 N/m and a free length $R_\circ = 150$ mm. The viscous damper has a damping coefficient of C = 350 Ns/m and is positioned at a height above the X-axis of 101.60 mm.

2. DELIVERABLE 2 RESULTS

2.1 Equivalent Mass Moment of Inertia

We begin our analysis by looking at the Kinetic Energy and its time derivative of the system as a whole. To do this, the equivalent mass moment of inertia is needed to be calculated for the mechanism. This can be done according to the following:

$$I_{eq} = \sum_{j=2}^{\infty} [m_j(X_{jg}'^2 + Y_{jg}'^2) + I_{jg}\theta'^2]$$

Which then can be expanded to match the characteristics of our system to be:

$$I_{eq} = I_{2g}\theta_2'^2 + m3(X_{3g}'^2 + Y_{3g}'^2) + I_{3g}\theta_3'^2 + I_{4g}\theta_4'^2 + m5(X_{5g}'^2 + Y_{5g}'^2) + I_{5g}\theta_5'^2 + I_{6g}\theta_6'^2$$

The results of which can be seen in the figure 2. Values of which can be seen in figure 5.

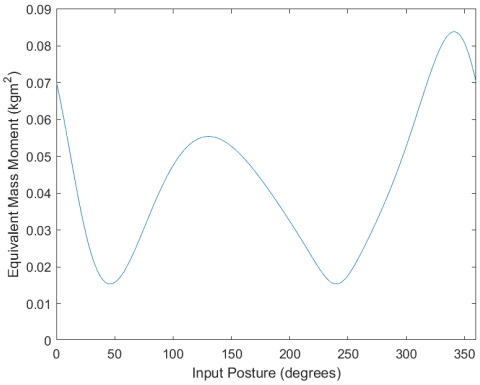

Figure 2. Resulting plot of equivalent mass moment of inertia vs link 2 input posture.

Using the equivalent mass moment of inertia, the values of kinetic energy and its time derivative can be found according to the following equations:

$$T(\theta_2) = \frac{1}{2} I_{eq} \dot{\psi}^2$$

$$\frac{dT}{dt}(\theta_2) = I_{eq}\dot{\psi}\ddot{\psi} + \Big[\sum_{j=2}^{\infty}(m_j(X_{jg}'X_{jg}'' + Y_{jg}'Y_{jg}'') + I_{jg}\theta'\theta''\Big]\dot{\psi}^3$$

In this expression for the time derivative of kinetic energy, the first component of it ends up equalling 0 due to the input angular velocity being a constant value. The results for kinetic energy and its time derivative can be seen below in figures 3 and 4. Numerical values for it can also be seen in figure 5.

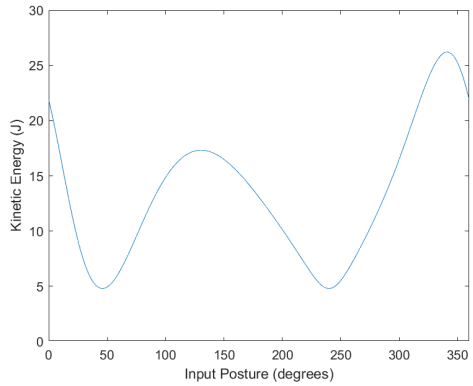

Figure 3. Resulting values of kinetic energy at every input posture.

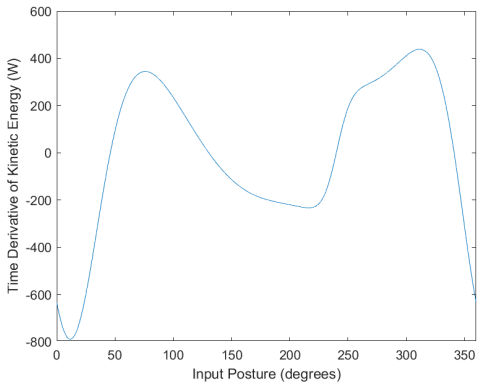

Figure 4. Resulting values of the time derivative of kinetic energy at every input posture.

Input (°)	I_{eq} (kgm^2)	T (J)	dT/dt (W)
0	0.0700938	21.9043	-633.0504
10	0.0538184	16.8183	-787.0235
20	0.0366829	11.4634	-710.2443
30	0.0233759	7.30496	-462.0541
40	0.0164172	5.13038	-161.9441
50	0.0157843	4.9326	92.39716
60	0.0198719	6.20998	257.7449
70	0.0266315	8.32235	333.8826
80	0.0342717	10.7099	340.5072
90	0.0415026	12.9695	300.9312
100	0.0475219	14.8506	234.9129
110	0.0519117	16.2224	157.0831
120	0.0545292	17.0404	77.63664
130	0.0554204	17.3189	3.468451
140	0.054757	17.1116	-60.96153
150	0.0527878	16.4962	-113.1754
160	0.0497964	15.5614	-152.5029
170	0.0460624	14.3945	-179.9535
180	0.0418263	13.0707	-197.9547
190	0.0372624	11.6445	-209.9335
200	0.0324624	10.1445	-219.7197
210	0.0274426	8.57581	-229.6319
220	0.0222575	6.95548	-231.0037
230	0.0175293	5.47792	-175.8839
240	0.0153613	4.80041	0.498507
250	0.0176545	5.51704	189.8122
260	0.0229617	7.17553	269.6288
270	0.029328	9.16499	297.9618
280	0.0362892	11.3404	327.055
290	0.0440261	13.7582	367.1859
300	0.0527256	16.4768	410.9672
310	0.0622687	19.459	438.0544
320	0.0719103	22.472	412.5016
330	0.0799611	24.9879	288.0992
340	0.0838138	26.1918	35.60761
350	0.0808291	25.2591	-310.9223
360	0.0700955	21.9049	-633.0676

Figure 5. Table of value results for equivalent mass moment of inertia, kinetic energy, and the time derivative of kinetic energy.

2.2 Kinematics of the Spring and Damper

To begin the analysis of the spring and the potential energy that results from it, the first determination needed is the first order kinematic coefficient of the spring. This is found using a standard approach for determining where R_s' is the coefficient of interest as seen below.

$$x : R_s cos(\theta_s) - R_{O2O6} cos(\theta_1) - R_6 cos(\theta_6) - R_{PD} cos(\theta_5) = 0$$
$$y : R_s sin(\theta_s) - R_{O2O6} sin(\theta_1) - R_6 sin(\theta_6) - R_{PD} sin(\theta_5) = 0$$
$$x' : R_s' cos(\theta_s) - R_s \theta_s' sin(\theta_s) = -(R_6 \theta_6' sin(\theta_6) + R_{PD} \theta_5' sin(\theta_5))$$
$$y' : R_s' sin(\theta_s) + R_s \theta_s' cos(\theta_s) = R_6 \theta_6' cos(\theta_6) + R_{PD} \theta_5' cos(\theta_5)$$
$$\begin{bmatrix} cos(\theta_s) & -R_s sin(\theta_s) \\ sin(\theta_s) & R_s cos(\theta_s) \end{bmatrix} \begin{bmatrix} R_s' \\ \theta_s' \end{bmatrix} = \begin{bmatrix} -(R_6 \theta_6' sin(\theta_6) + R_{PD} \theta_5' sin(\theta_5)) \\ R_6 \theta_6' cos(\theta_6) + R_{PD} \theta_5' cos(\theta_5) \end{bmatrix}$$

Solving this matrix found above for R_s' yields the first order kinematic coefficient for the spring. Figure 6 below shows the results from this. Figure 9 includes the resulting values in numerical form.

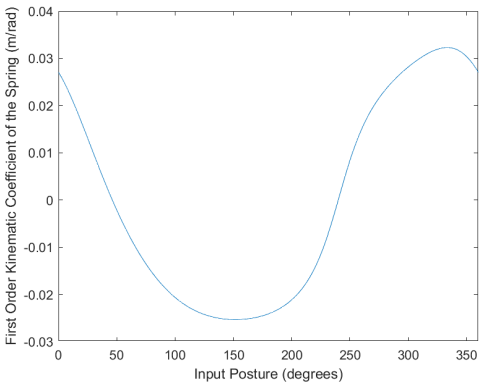

Figure 6. Resulting plot of first order kinematic coefficients for the spring at every input posture.

Once these kinematic coefficients are found, the values for the spring potential energy and its first time derivative can be found using the following expressions:

$$U_{sp}(\theta_2) = \frac{1}{2} k (R_s(\theta_2) - R_0)^2$$
$$\frac{dU_{sp}}{dt}(\theta_2) = k(R_s(\theta_2) - R_0) R_s' \dot{\psi}$$

The resulting plots for these values at every input can then be seen in figures 7 and 8. Resulting values can be found in figure 9 as well.

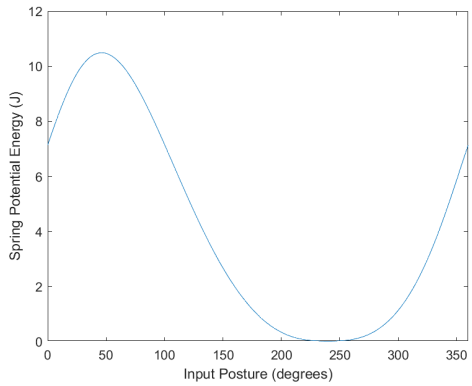

Figure 7. Plot of spring potential energy at every input posture.

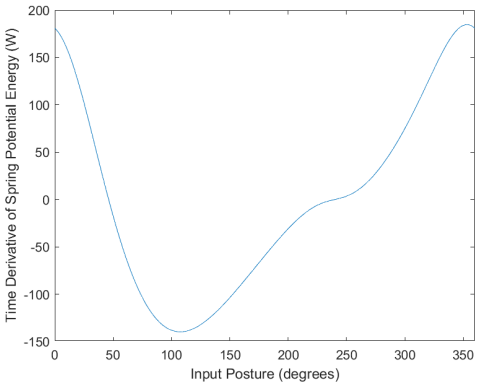

Figure 8. Plot of the first time derivative of spring potential energy at every input posture.

Input (°)	R_s' (m/rad)	U_{sp} (J)	dU_{sp}/dt (W)
0	0.0271079	7.13268	180.993246
10	0.0222496	8.33519	160.590511
20	0.0163014	9.33818	124.536179
30	0.0099077	10.0508	78.5257476
40	0.0036393	10.4274	29.379529
50	-0.0021338	10.467	-17.258857
60	-0.0072345	10.2008	-57.765161
70	-0.0116105	9.67906	-90.304461
80	-0.0152735	8.9599	-114.29583
90	-0.0182632	8.1026	-129.96609
100	-0.0206323	7.16292	-138.049
110	-0.0224403	6.19033	-139.58072
120	-0.0237511	5.22649	-135.7464
130	-0.0246304	4.30461	-127.75518
140	-0.0251408	3.44971	-116.73754
150	-0.0253338	2.6794	-103.67118
160	-0.0252391	2.0051	-89.347251
170	-0.0248531	1.43335	-74.387033
180	-0.0241277	0.96674	-59.307724
190	-0.0229578	0.60436	-44.618855
200	-0.021163	0.34146	-30.916369
210	-0.018445	0.16868	-18.938598
220	-0.0143185	0.07102	-9.5392725
230	-0.0081616	0.0278	-3.4021929
240	-8.29E-12	0.01692	-2.69E-09
250	0.0083517	0.02799	3.49326605
260	0.0149761	0.07323	10.1316472
270	0.0197355	0.17911	20.8810947
280	0.0232295	0.37382	35.5066333
290	0.0259452	0.68295	53.603446
300	0.0281648	1.12955	74.8341766
310	0.0300075	1.7342	98.7915742
320	0.0314345	2.51329	124.585674
330	0.0322318	3.47341	150.176575
340	0.0320271	4.60092	171.743445
350	0.0304031	5.84933	183.827366
360	0.0271082	7.13268	180.995351

Figure 9. Table of spring values.

To finish the analysis then of the potential energy of the system, the gravitational potential energy and its time derivative must be analysed. Expressions for this can be seen below:

$$U_g(\theta_2) = \sum_{j=2}^{\infty} m_j g Y_{jg}$$

$$\frac{dU_g}{dt}(\theta_2) = \sum_{j=2}^{\infty} m_j g Y'_{jg} \dot{\psi}$$

In this expression, the values for links 1, 2, 4, and 6 can be neglected as their center of mass is at a height of 0 m and their first order kinematic coefficient at each center of mass is 0 m/rad. The resulting plots of these values can be seen in figures 10 and 11 with figure 12 being a table of the calculated values.

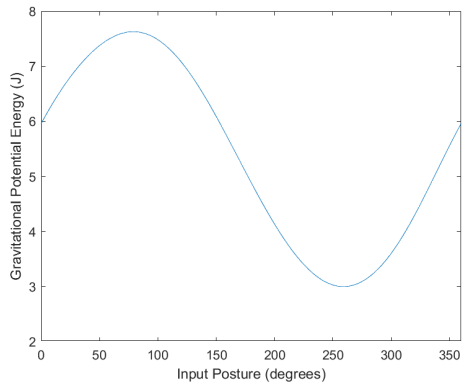

Figure 10. Plot of the gravitational potential energy values at every input posture.

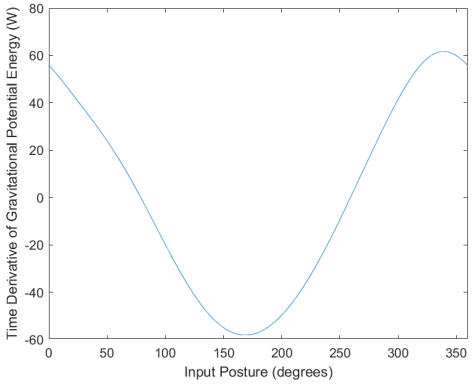

Figure 11. Plot of the first time derivative of gravitational potential energy at every input posture.

Input (°)	U_g (J)	dU_g/dt (W)
0	5.96115	55.840757
10	6.33173	50.155892
20	6.6601	43.871664
30	6.94404	37.460796
40	7.18286	30.91304
50	7.37475	23.969393
60	7.51602	16.380766
70	7.60173	8.0502753
80	7.62689	-0.9359506
90	7.58769	-10.338109
100	7.48237	-19.813002
110	7.31179	-28.965561
120	7.07962	-37.394154
130	6.79223	-44.727834
140	6.45838	-50.655357
150	6.08876	-54.945499
160	5.69534	-57.457454
170	5.29077	-58.140205
180	4.88775	-57.02121
190	4.4986	-54.18733
200	4.13486	-49.763388
210	3.80714	-43.894455
220	3.52499	-36.736939
230	3.2968	-28.462091
240	3.12971	-19.272015
250	3.02928	-9.4079513
260	2.99933	0.8789091
270	3.04199	11.356156
280	3.1578	21.792877
290	3.34555	31.918397
300	3.60191	41.378182
310	3.92065	49.696473
320	4.29175	56.271047
330	4.70075	60.448283
340	5.12898	61.721606
350	5.55556	60.012358
360	5.96115	55.840361

Figure 12. Table of gravitational values.

Next, an analysis of the viscous damper must be performed to get a final understanding of all energy and power effects on the system. With this analysis, one critical assumption must be made: The viscous damper being parallel to the X-axis has negligible motion in the Y-direction meaning that the first-order kinematic coefficient is equal and opposite to the first order kinematic coefficient in the X-direction for point P where it is attached. With this assumption in mind, we can write the expressions below to determine the energy and power effects it has on the system.

$$W_c(\theta_2) = cR_c'\dot{\psi}(X_{Pmax} - X_P(\theta_2))$$

$$\frac{dW_c}{dt}(\theta_2) = cR_c'^2\dot{\psi}^2$$

Resulting plots for the first order kinematic coefficient of the damper, the energy contribution of the damper, and its power contribution can be seen in figures 13, 14, and 15 respectively. Figure 16 is a table of the resulting values for each.

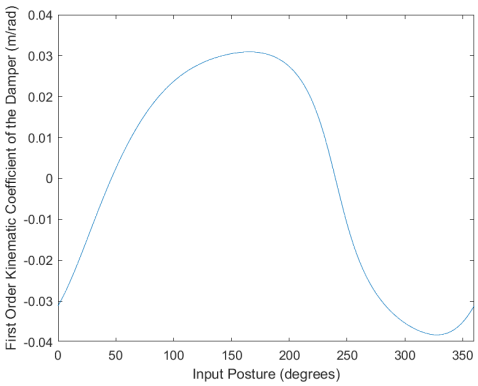

Figure 13. Plot of viscous damper first-order kinematic coefficients at every input angle.

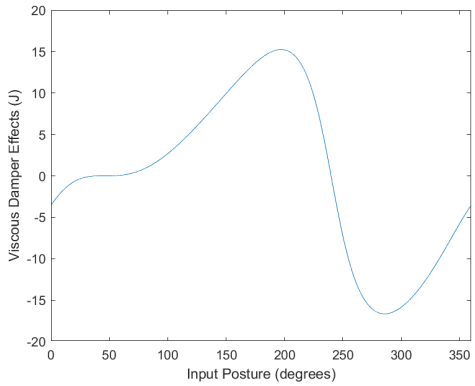

Figure 14. Plot of viscous damper energy effects at every input angle.

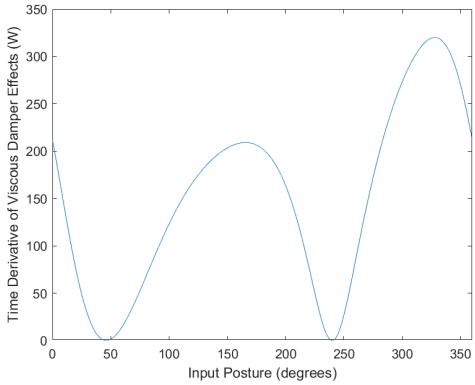

Figure 15. Plot of viscous damper power effects at every input angle.

Input (°)	R_c' (m/rad)	W_c (J)	dW_c/dt (W)
0	-0.0311537	-3.5474	212.307934
10	-0.0254913	-1.7952	142.145016
20	-0.0186488	-0.6828	76.0766649
30	-0.0113255	-0.1553	28.0582197
40	-0.0041583	-0.0081	3.78259323
50	0.0024381	0.00174	1.3002882
60	0.0082684	0.07437	14.9551417
70	0.0132773	0.33924	38.5628864
80	0.0174814	0.85898	66.849765
90	0.0209337	1.64458	95.8606686
100	0.0237092	2.67275	122.965057
110	0.0258962	3.90197	146.697
120	0.0275886	5.28526	166.497351
130	0.0288755	6.7781	182.392388
140	0.0298293	8.34027	194.640475
150	0.0304917	9.93104	203.381828
160	0.0308609	11.4983	208.336591
170	0.0308811	12.9632	208.609301
180	0.0304374	14.2042	202.657575
190	0.0293508	15.04	188.446982
200	0.0273606	15.2087	163.75691
210	0.024067	14.3279	126.704396
220	0.0188244	11.8291	77.5160816
230	0.0107911	7.02934	25.4728275
240	1.10E-11	7.25E-09	2.64E-17
250	-0.011042	-7.1913	26.6712093
260	-0.0196846	-12.351	84.7616378
270	-0.0257339	-15.247	144.863507
280	-0.0299872	-16.485	196.706789
290	-0.0330689	-16.583	239.214432
300	-0.0353346	-15.871	273.117178
310	-0.0369597	-14.557	298.816681
320	-0.0379706	-12.78	315.386103
330	-0.038228	-10.638	319.676838
340	-0.0374219	-8.246	306.336431
350	-0.0351558	-5.7911	270.359148
360	-0.0311541	-3.5474	212.313746

Figure 16. Table of viscous damper values.

2.3 The Power Equation and the Equation of Motion

Now that the contributions from every component has been determined, an analysis to gain a deeper understanding can be performed. To begin, the power equation - seen below - yields a value for the net power in the mechanism at every input posture. Then we are able to use this to find the torque seen in the expression below.

$$P(\theta_2) = \frac{dT}{dt}(\theta_2) + \frac{dU}{dt}(\theta_2) + \frac{dW_c}{dt}(\theta_2)$$

where:

$$\frac{dU}{dt}(\theta_2) = \frac{dU_g}{dt}(\theta_2) + \frac{dU_{sp}}{dt}(\theta_2)$$

and:

$$\overline{T_{12}} = \frac{P(\theta_2)}{\dot{\psi}}$$

Hand calculations for the motor torque input can be found in Appendix A. Using the relation found above, a plot for the value of torque can be seen in figure 17 and values for it can be seen in figure 18.

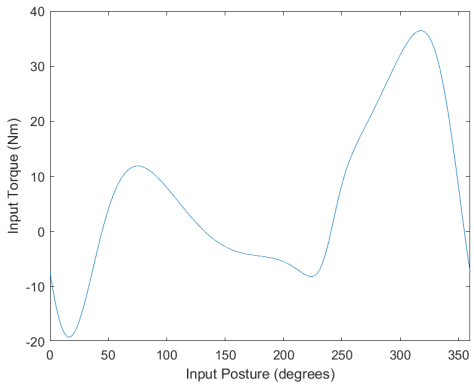

Figure 17. Plot of torque input on link 2.

Input (°)	Torque (Nm)
0	-7.35633727
10	-17.3652826
20	-18.6303906
30	-12.7203728
40	-3.91475566
50	4.016319299
60	9.252623876
70	11.60765216
80	11.68500782
90	10.25950645
100	8.000636939
110	5.409353516
120	2.839737257
130	0.535112978
140	-1.3485581
150	-2.73641136
160	-3.63884252
170	-4.15485943
180	-4.46504286
190	-4.81170729
200	-5.46570198
210	-6.63042063
220	-7.9905546
230	-7.29101534
240	-0.7509403
250	8.422749161
260	14.61603859
270	19.00250225
280	23.24245319
290	27.67688865
300	32.01186851
310	35.41436706
320	36.34977882
330	32.73603427
340	23.01636348
350	8.13106173
360	-7.35672401

Figure 18. Table of torque values.

The net power into the system can then be seen in figure 19.

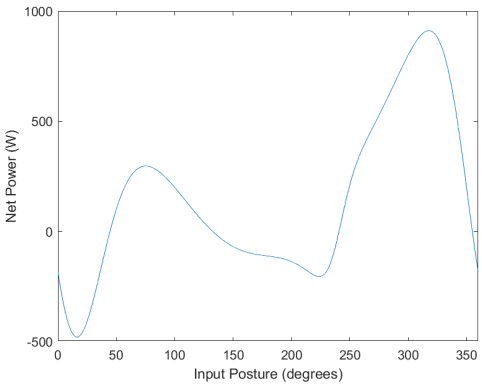

Figure 19. Net power on the mechanism.

From here, the contributions of each power component can be determined. To see this, two methods were used. The first method is the raw contribution relation that can be found using the general relation below. This method has a few issues with it - namely that the net power reaches a 0 value at certain input postures and therefore creates a limit on the contribution.

$$contribution = \frac{contributing factor}{net power}$$

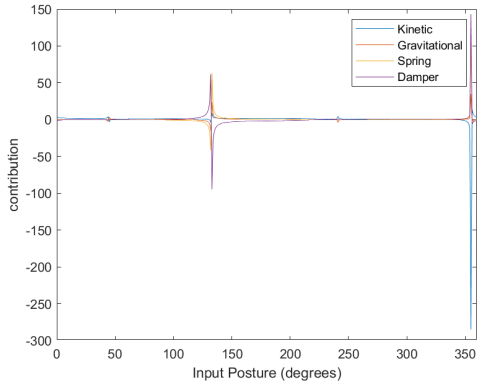

Figure 20. Contributions of each power component using the first method

The second method instead focuses on the absolute values. This method follows the general structure seen below. In this one, we are primarily comparing the size of each contribution by neglecting the sign for each.

$$contribution = \frac{|contributing factor|}{\sum |contributing factor|}$$

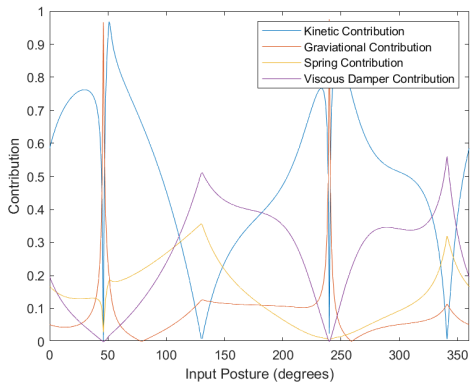

Figure 21. Contributions of each power component using the second method

Figure 22 is a breakdown of the net power and the contribution values for each method.

Input (°)	P (W)	Method 1				Method 2			
		dT/dt	dU_g/dt	dU_{sp}/dt	dW/dt	dT/dt	dU_g/dt	dU_{sp}/dt	dW/dt
0	-183.91	3.4422	-0.3036	-0.9841	-1.1544	0.58497	0.0516	0.16725	0.19618
10	-434.13	1.81287	-0.1155	-0.3699	-0.3274	0.69042	0.044	0.14088	0.1247
20	-465.76	1.52492	-0.0942	-0.2674	-0.1633	0.74392	0.04595	0.13044	0.07968
30	-318.01	1.45296	-0.1178	-0.2469	-0.0882	0.76234	0.06181	0.12956	0.04629
40	-97.869	1.6547	-0.3159	-0.3002	-0.0386	0.71651	0.13677	0.12999	0.01674
50	100.408	0.92022	0.23872	-0.1719	0.01295	0.92022	0.23872	0.17189	0.01295
60	231.316	1.11426	0.07082	-0.2497	0.06465	0.82062	0.05215	0.18392	0.04762
70	290.191	1.15056	0.02774	-0.3112	0.13289	0.73429	0.0177	0.1986	0.08481
80	292.125	1.16562	-0.0032	-0.3913	0.22884	0.65158	0.00179	0.21871	0.12792
90	256.488	1.17328	-0.0403	-0.5067	0.37374	0.56029	0.01925	0.24198	0.17848
100	200.016	1.17447	-0.0991	-0.6902	0.61478	0.45549	0.03842	0.26767	0.23842
110	135.234	1.16157	-0.2142	-1.0321	1.08477	0.33257	0.06133	0.29552	0.31058
120	70.9934	1.09357	-0.5267	-1.9121	2.34525	0.18606	0.08962	0.32532	0.39901
130	13.3778	0.25927	-3.3434	-9.5498	13.6339	0.00968	0.12482	0.35652	0.50899
140	-33.714	1.8082	1.5025	3.46259	-5.7733	0.14412	0.11975	0.27598	0.46015
150	-68.41	1.65436	0.80318	1.51543	-2.973	0.23818	0.11563	0.21818	0.42802
160	-90.971	1.67639	0.6316	0.98215	-2.2901	0.30041	0.11318	0.176	0.4104
170	-103.87	1.73246	0.55973	0.71614	-2.0083	0.34534	0.11157	0.14275	0.40033
180	-111.63	1.77337	0.51082	0.53131	-1.8155	0.38293	0.11031	0.11473	0.39203
190	-120.29	1.74519	0.45046	0.37092	-1.5666	0.42224	0.10899	0.08974	0.37903
200	-136.64	1.60799	0.36419	0.22626	-1.1984	0.47337	0.10721	0.06661	0.35281
210	-165.76	1.38532	0.26481	0.11425	-0.7644	0.54783	0.10472	0.04518	0.30227
220	-199.76	1.15638	0.1839	0.04775	-0.388	0.65109	0.10354	0.02689	0.21848
230	-182.28	0.96494	0.15615	0.01867	-0.1397	0.75415	0.12204	0.01459	0.10922
240	-18.774	-0.0266	1.02655	1.44E-10	########	0.02521	0.97479	1.36E-10	1.33E-18
250	210.569	0.90143	-0.0447	0.01659	0.12666	0.85348	0.0423	0.01571	0.11993
260	365.401	0.7379	0.00241	0.02773	0.23197	0.7379	0.00241	0.02773	0.23197
270	475.063	0.62721	0.0239	0.04395	0.30494	0.62721	0.0239	0.04395	0.30494
280	581.061	0.56286	0.03751	0.06111	0.33853	0.56286	0.03751	0.06111	0.33853
290	691.922	0.53068	0.04613	0.07747	0.34572	0.53068	0.04613	0.07747	0.34572
300	800.297	0.51352	0.0517	0.09351	0.34127	0.51352	0.0517	0.09351	0.34127
310	885.359	0.49478	0.05613	0.11158	0.33751	0.49478	0.05613	0.11158	0.33751
320	908.744	0.45392	0.06192	0.1371	0.34706	0.45392	0.06192	0.1371	0.34706
330	818.401	0.35203	0.07386	0.1835	0.39061	0.35203	0.07386	0.1835	0.39061
340	575.409	0.06188	0.10727	0.29847	0.53238	0.06188	0.10727	0.29847	0.53238
350	203.277	-1.5296	0.29523	0.90432	1.33001	0.37682	0.07273	0.22279	0.32766
360	-183.92	3.44212	-0.3036	-0.9841	-1.1544	0.58497	0.0516	0.16724	0.19618

Figure 22. Table of net power and contribution values using both methods.

2.4 Torque Analysis

Finally, we can compare the results of using the power equation to determine the torque to the values of torque found in deliverable 1. Figure 23 shows the results of both methods. and Figure 24 is the calculated values for these plots.

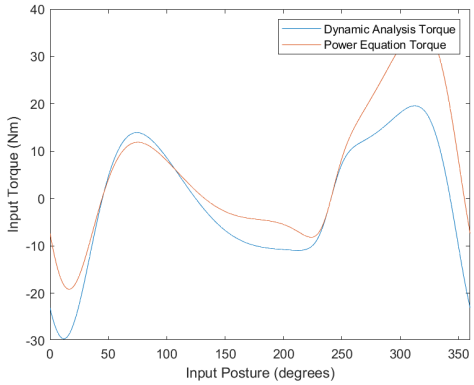

Figure 23. Comparison of the torque values calculated in deliverable 1 and deliverable 2.

Input (°)	Del. 1 Torque (Nm)	Del. 2 Torque (Nm)
0	-23.08833771	-7.356337269
10	-29.47417894	-17.36528262
20	-26.65447965	-18.63039057
30	-16.98349446	-12.7203728
40	-5.24115199	-3.914755659
50	4.654675092	4.016319299
60	10.96500891	9.252623876
70	13.67728984	11.60765216
80	13.58282277	11.68500782
90	11.62369646	10.25950645
100	8.603970413	8.000636939
110	5.124680311	5.409353516
120	1.609679138	2.839737257
130	-1.650393477	0.535112978
140	-4.464688553	-1.348558096
150	-6.724839242	-2.736411355
160	-8.398400412	-3.638842524
170	-9.523717343	-4.154859428
180	-10.19900637	-4.465042861
190	-10.56485973	-4.811707289
200	-10.77951875	-5.465701984
210	-10.94155955	-6.630420628
220	-10.71039525	-7.990554601
230	-8.174215952	-7.291015345
240	-0.750940748	-0.750940302
250	7.216523915	8.422749161
260	10.82122887	14.61603859
270	12.37343576	19.00250225
280	13.95425678	23.24245319
290	15.96425696	27.67688865
300	18.09376667	32.01186851
310	19.5099415	35.41436706
320	18.75081046	36.34977882
330	13.94183077	32.73603427
340	3.893192523	23.01636348
350	-10.03618765	8.13106173
360	-23.08865663	-7.356724015

Figure 24. Table of the calculated torque values from both deliverables.

Overall, the results from both methods are strikingly similar in general shape and magnitude. At multiple points, they even appear to be nearly equal. The slight differences are due to the addition of the spring and damper onto the system adding extra components that were not accounted for in the dynamic force analysis. If the spring and damper effects were not included in the analysis, the calculated torques for the two methods are identical.

3. CONCLUSION

Overall, this method provides a very rational methodology for determining the torque input when you have a complex mechanism like the one that was being analyzed. From this, we are able to see that the average contribution of kinetic energy is 51.35%, gravitational potential energy is 8.8%, spring potential energy is 14.89%, and viscous damping effects is 26.16%. This means that kinetic energy has by far the largest contribution to the net power on the system, while the others have relatively smaller contributions and don't effect the mechanism net power as greatly.

The primary concern with this mechanism is the magnitude of forces generated can be rather high at times. With this comes an increased likely hood in the links breaking due to the stresses on each of them. This can be avoided by choosing a material that allows for a safety factor of at least 1.5. Another concern that arises through

this design is the magnitude of torque. While finding a motor that should have no issue providing enough torque to meet the input needs, many motors will not be able to provide adequate torque to the system if the material needs to change to ensure that the links do not shear. If this recommended change is made, the masses will likely increase causing the torque required to also increase. This means careful attention must be paid to the torque output abilities of the chosen driving motor.

In summary, the Watt-II six bar linkage provides a fairly easy way to produce a straight line motion of a point when little needed input. However, there are careful considerations that must be made when determining the application that can greatly chan ge the results and cause other design changes to occur as well.

4. APPENDIX

Appendix A: Hand Calculations of Motor Torque

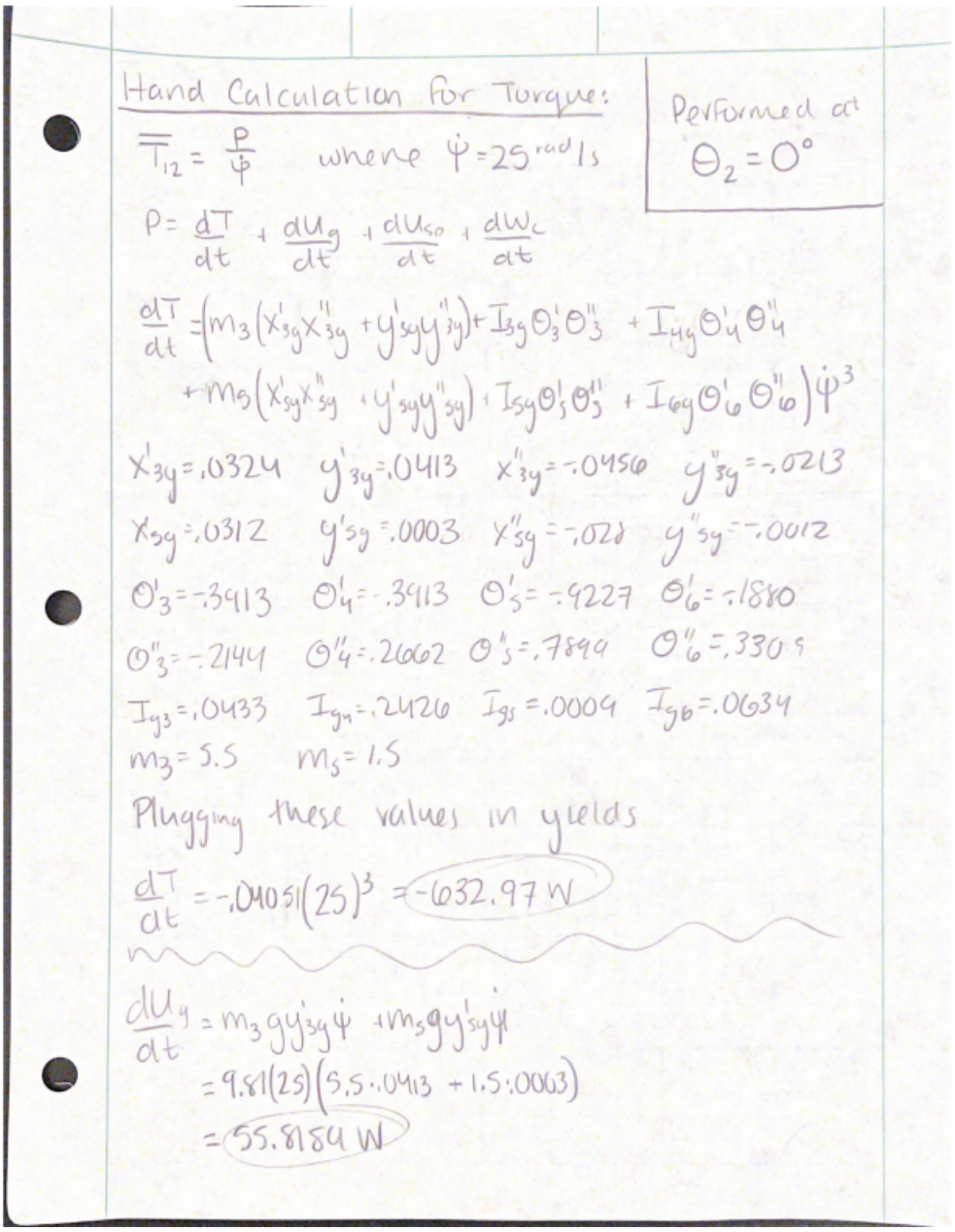

Figure 25.

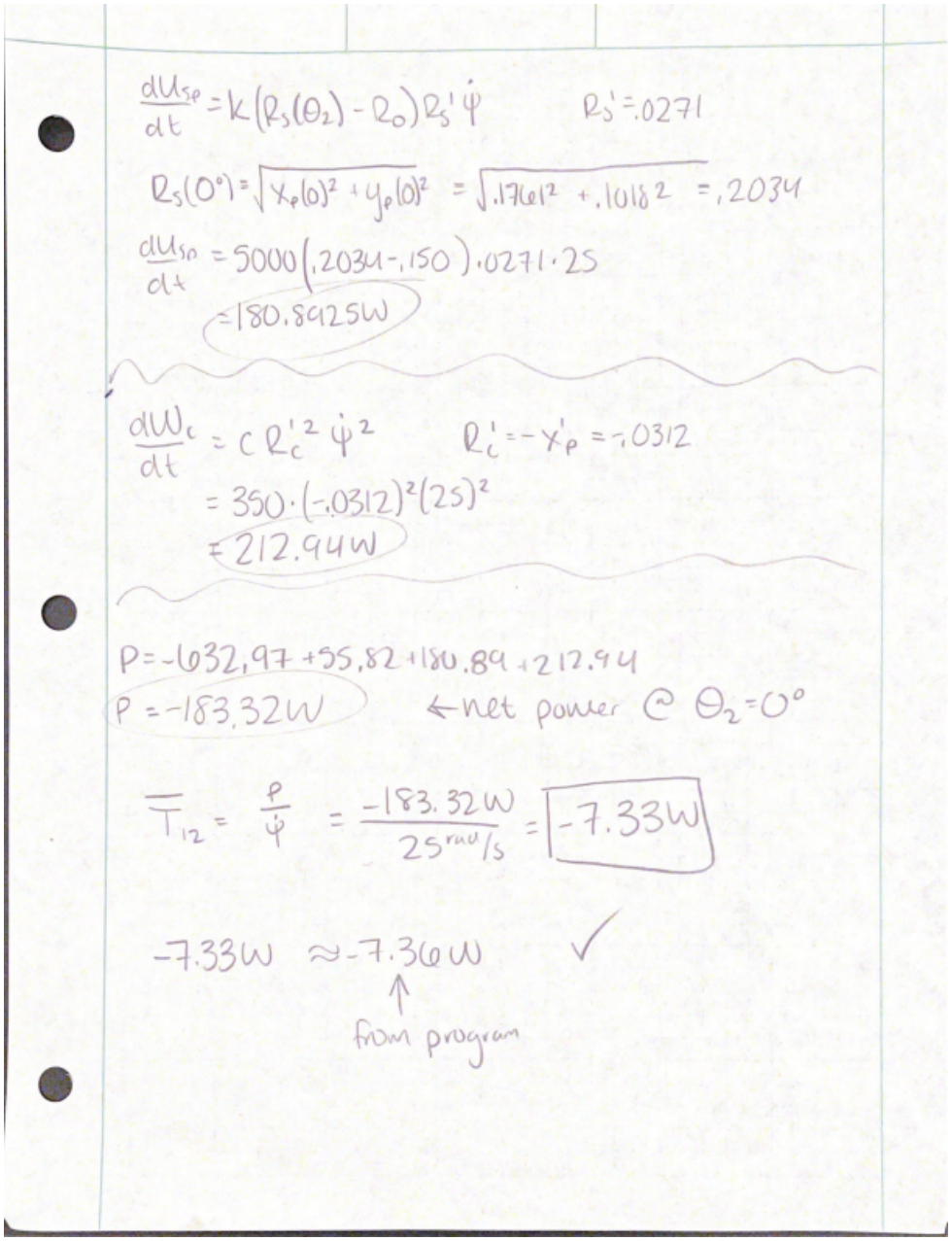

Figure 26.

Appendix B: MatLab Script

```
% This code was created by Trevor Ladner for use on Project 1 and 2 in ME
% 452 at Purdue University.
%
% If you are reading this, I just wanted to let you know that I have really
% enjoyed working on project 1 and 2 in this class. It has been a large
% amount of work, but I have really enjoyed the undertaking. Thank you for
% a great class!
%
% December 6, 2021

clc

% UNIVERSAL CONSTANTS
G = 9.81;

R1 = .2032;
R2 = .05715;
R3 = .18415;
R4 = .1778;
R5 = .0508;
R6 = .127;
R11 = .1016;
R44 = .127;
RCP = .0254;
R55 = RCP;

m2 = 2;
m3 = 5.5;
m4 = 7.5;
m5 = 1.5;
m6 = 6;

ig2 = 0.0067;
ig3 = 0.0433;
ig4 = 0.2426;
ig5 = 0.0009;
ig6 = 0.0634;

% INITIALIZING ARRAYS
a = zeros(5,361);
alpha1 = zeros(5,361);
alpha2 = zeros(5,361);
alpha25s = zeros(5,37);
alpha50s = zeros(5,37);
as = zeros(5,37);
CC = zeros(2,361);
CCs = zeros(2,37);
coef = zeros(4,361);
coef2 = zeros(4,361);
coef2s = zeros(4,37);
coefP = zeros(5,361);
coefPs = zeros(5,37);
coefS = zeros(4,37);
Det = zeros(2,361);
detS = zeros(2,37);
P = zeros(3,361);
Ps = zeros(3,37);
```

```
rho = zeros(2,361);
rhos = zeros(2,37);
RpP = zeros(1,361);
theta = zeros(6,361);
thetaS = zeros(6,37);
un = zeros(3,361);
uns = zeros(3,37);
ut = zeros(3,361);
uts = zeros(3,37);
v = zeros(5,361);
vel25S = zeros(4,37);
vel50S = zeros(4,37);
vs = zeros(5,37);
variables = zeros(16,361);
coefG3 = zeros(5,361);
accelX = zeros(2,361);
accelY = zeros(2,361);
accelX2 = zeros(4,361);
accelY2 = zeros(4,361);
results = zeros(16,361);
results50 = zeros(16,361);
staticResults = zeros(16,361);
part2 = zeros(13,361);
part2s = zeros(37,13);
resultss = zeros(37,16);
staticResultss = zeros(37,16);
part5 = zeros(2,361);
part5s = zeros(37,2);
i3 = zeros(1,361);
i4 = zeros(1,361);
i5 = zeros(1,361);
i6 = zeros(1,361);
ieq = zeros(2,361);
B = zeros(2,361);
b3 = zeros(1,361);
b4 = zeros(1,361);
b5 = zeros(1,361);
b6 = zeros(1,361);
dT = zeros(1,361);
T = zeros(1,361);
yg3 = zeros(1,361);
Ug3 = zeros(1,361);
Ug5 = zeros(1,361);
Ug = zeros(1,361);
dUg3 = zeros(1,361);
dUg5 = zeros(1,361);
dUg = zeros(1,361);
sprL = zeros(1,361);
delL = zeros(1,361);
Usp = zeros(1,361);
thetaSprRad = zeros(1,361);
thetaSpr = zeros(1,361);
coefR = zeros(1,361);
dUsp = zeros(1,361);
U = zeros(1,361);
dU = zeros(1,361);
W = zeros(1,361);
```

```
dW = zeros(1,361);
power = zeros(1,361);
Tpower = zeros(1,361);
kinCon = zeros(1,361);
gravCon = zeros(1,361);
sprCon = zeros(1,361);
damCon = zeros(1,361);
torque = zeros(1,361);
p2d2p1 = zeros(37,4);
Usps = zeros(37,4);
Ugs = zeros(37,3);
Ws = zeros(37,4);
torques = zeros(37,2);
kinCon2 = zeros(1,361);
gravCon2 = zeros(1,361);
sprCon2 = zeros(1,361);
damCon2 = zeros(1,361);
cont = zeros(37,10);
compare = zeros(37,3);

% PROJECT 1 DELIVERABLE 1
% Calculations for angles
j = 0;
resolution = .01*pi/180;
theta3s = 60*pi/180;
theta4s = 110*pi/180;
theta5s = -40*pi/180;
theta6s = 45*pi/180;

for theta2 = 0:pi/180:2*pi
    j = j+1;
    i = 0;
    dt_3 = 1;
    dt_4 = 1;
    while (abs(dt_3) > resolution) || (abs(dt_4) > resolution)
        i = i+1;
        errx = (R2 * cos(theta2)) + (R3 * cos(theta3s)) - (R4 * cos(theta4s))
- R1;
        erry = (R2 * sin(theta2)) + (R3 * sin(theta3s)) - (R4 *
sin(theta4s));
        derrxdt3 = -R3 * sin(theta3s);
        derrxdt4 = R4 * sin(theta4s);
        derrydt3 = R3 * cos(theta3s);
        derrydt4 = -R4 * cos(theta4s);
        D = (derrxdt3 * derrydt4) - (derrxdt4 * derrydt3);
        dt_3 = ((errx * -derrydt4) + (erry * derrxdt4))/D;
        dt_4 = ((erry * -derrxdt3) + (errx * derrydt3))/D;
        theta3s = theta3s + dt_3;
        theta4s = theta4s + dt_4;
        if i > 10
            break
        end
    end
    theta(1,j) = 0.00;
    theta(2,j) = theta2;
    theta(3,j) = theta3s;
    theta(4,j) = theta4s;
```

```
    Det(1,j) = D;
end

for m = 1:1:361
    i = 0;
    dt_5 = 1;
    dt_6 = 1;
    while (abs(dt_5) > resolution) || (abs(dt_6) > resolution)
        i = i+1;
        errx2 = (R6 * cos(theta6s)) - (R5 * cos(theta5s)) - (R44 *
cos(theta(4,m))) - R11;
        erry2 = (R6 * sin(theta6s)) - (R5 * sin(theta5s)) - (R44 *
sin(theta(4,m)));
        derrxdt5 = R5 * sin(theta5s);
        derrxdt6 = -R6 * sin(theta6s);
        derrydt5 = -R5 * cos(theta5s);
        derrydt6 = R6 * cos(theta6s);
        D2 = (derrxdt5 * derrydt6) - (derrxdt6 * derrydt5);
        dt_5 = ((-errx2 * derrydt6) - (-erry2 * derrxdt6))/D2;
        dt_6 = ((-erry2 * derrxdt5) - (-errx2 * derrydt5))/D2;
        theta5s = theta5s + dt_5;
        theta6s = theta6s + dt_6;
        if i > 10
            break
        end
    end
    theta(5,m) = theta5s;
    theta(6,m) = theta6s;
    Det(2,m) = D2;
end
thetaRad = theta;
theta = theta.*180./pi;
theta = round(theta,2);

% Swing Angle
maxt = max(theta(4,:));
mint = min(theta(4,:));

swing = maxt - mint;
% fprintf('The Swing Angle is %.2f degrees', swing);

% plots of angles and determinants
figure(1)
plot(theta(2,:),theta(3,:),theta(2,:),theta(4,:),theta(2,:),theta(5,:),theta(
2,:),theta(6,:));
xlabel('Input Posture (degrees)');
ylabel('Link Posture (degrees)');
xlim([0 360]);
title('Link Postures based on Input');
legend('Link 3', 'Link 4', 'Link 5', 'Link 6')

figure(2)
plot(theta(2,:),Det(1,:),theta(2,:),Det(2,:));
xlabel('Input Posture (degrees)');
ylabel('Determinant (mm^2)');
xlim([0 360]);
title('Vector Loop Determinants');
```

```
legend('VLE 1', 'VLE 2')

% Kinematic Coefficients
for n = 1:1:361
    tp3 = (R4 * R2 * sin(thetaRad(4,n) - thetaRad(2,n))) / Det(1,n);
    tp4 = (R3 * R2 * sin(thetaRad(3,n) - thetaRad(2,n))) / Det(1,n);
    coef(1,n) = tp3;
    coef(2,n) = tp4;
end

for o = 1:1:361
    tp5 = coef(2,o) * (R6 * R44 * sin(thetaRad(6,o) - thetaRad(4,o))) /
Det(2,o);
    tp6 = coef(2,o) * (R5 * R44 * sin(thetaRad(5,o) - thetaRad(4,o))) /
Det(2,o);
    coef(3,o) = tp5;
    coef(4,o) = tp6;
end

% kincoef plot
figure(3)
plot(theta(2,:),coef(1,:),theta(2,:),coef(2,:),theta(2,:),coef(3,:),theta(2,:
),coef(4,:));
xlabel('Input Posture (degrees)');
ylabel('Kinematic Coefficients');
xlim([0 360]);
title('First Order Kinematics Coefficients based on Input');
legend('Link 3','Link 4','Link 5','Link 6')

% velocity calculations and plots
vel1 = coef .* 25;
vel2 = coef .* 50;

figure(4)
plot(theta(2,:),vel1(1,:),theta(2,:),vel1(2,:),theta(2,:),vel1(3,:),theta(2,:
),vel1(4,:))
xlabel('Input Posture (degrees)');
ylabel('Angular Velocity (rad/s)');
xlim([0 360]);
title('Angluar Velocities with Input of 25 rad/s');
legend('Link 3','Link 4','Link 5','Link 6')

figure(5)
plot(theta(2,:),vel2(1,:),theta(2,:),vel2(2,:),theta(2,:),vel2(3,:),theta(2,:
),vel2(4,:))
xlabel('Input Posture (degrees)');
ylabel('Angular Velocity (rad/s)');
xlim([0 360]);
title('Angluar Velocities with Input of 50 rad/s');
legend('Link 3','Link 4','Link 5','Link 6')

% PROJECT 1 DELIVERABLE 2
% Second-Order Kin Coef
for r = 1:1:361
    D5 = R3 * R4 * sin(thetaRad(3,r) - thetaRad(4,r));
```

```
    tpp3 = ((-R2 * R4 * cos(thetaRad(4,r) - thetaRad(2,r))) - (R3 * R4 *
(coef(1,r)^2) * cos(thetaRad(4,r) - thetaRad(3,r))) + ((R4^2) *
(coef(2,r)^2))) / D5;
    tpp4 = ((-R2 * R3 * cos(thetaRad(3,r) - thetaRad(2,r))) + (R3 * R4 *
(coef(2,r)^2) * cos(thetaRad(3,r) - thetaRad(4,r))) - ((R3^2) *
(coef(1,r)^2))) / D5;
    coef2(1,r) = tpp3;
    coef2(2,r) = tpp4;
end

for s = 1:1:361
    D6 = R5 * R6 * sin(thetaRad(5,s) - thetaRad(6,s));
    tpp5 = ((-R44 * R6 * coef2(2,s) * sin(thetaRad(4,s) - thetaRad(6,s))) -
(R44 * R6 * (coef(2,s)^2) * cos(thetaRad(4,s) - thetaRad(6,s))) - (R5 * R6 *
(coef(3,s)^2) * cos(thetaRad(5,s) - thetaRad(6,s))) + ((R6^2) *
(coef(4,s)^2))) / D6;
    tpp6 = ((-R44 * R5 * coef2(2,s) * sin(thetaRad(4,s) - thetaRad(5,s))) -
(R44 * R5 * (coef(2,s)^2) * cos(thetaRad(4,s) - thetaRad(5,s))) + (R5 * R6 *
(coef(4,s)^2) * cos(thetaRad(6,s) - thetaRad(5,s))) - ((R5^2) *
(coef(3,s)^2))) / D6;
    coef2(3,s) = tpp5;
    coef2(4,s) = tpp6;
end

% Plots for Second Order Coefficients
figure(6)
plot(theta(2,:),coef2(1,:),theta(2,:),coef2(2,:),theta(2,:),coef2(3,:),theta(
2,:),coef2(4,:))
xlabel('Input Angle (degrees)')
ylabel('Second Order Kinematic Coefficient')
title('Second Order Kinematic Coefficients')
xlim([0 360])
legend('Link 3','Link 4','Link 5','Link 6')

% Position of Point P
for t = 1:1:361
    Px = R1 + (R44 * cos(thetaRad(4,t))) + (RCP * cos(thetaRad(5,t)));
    Py = (R44 * sin(thetaRad(4,t))) + (RCP * sin(thetaRad(5,t)));
    P(1,t) = Px;
    P(2,t) = Py;
    P(3,t) = theta(2,t);
end

figure(7)
plot(P(1,:),P(2,:))
xlabel('x (mm)')
ylabel('y (mm)')
title('Position of Point P')
xlim([0 190])
ylim([0 120])

% Displacements
maxPx = max(P(1,:));
minPx = min(P(1,:));
maxPy = max(P(2,:));
minPy = min(P(2,:));
dispPx = maxPx - minPx;
```

```
dispPy = maxPy - minPy;

% Max and Min Displacement Angles
for u = 1:1:361
    if P(1,u) == maxPx
        maxPxT = P(3,u);
    end
    if P(1,u) == minPx
        minPxT = P(3,u);
    end
    if P(2,u) == maxPy
        maxPyT = P(3,u);
    end
    if P(2,u) == minPy
        minPyT = P(3,u);
    end
end

fprintf('\nThe displacement of P in the x direction is %.2f mm\n', dispPx)
fprintf('The displacement of P in the y direction is %.2f mm\n', dispPy)
fprintf('The input posture for maximum x position is %.2f degrees\n', maxPxT)
fprintf('The input posture for minimum x position is %.2f degrees\n', minPxT)
fprintf('The input posture for maximum y position is %.2f degrees\n', maxPyT)
fprintf('The input posture for minimum y position is %.2f degrees\n', minPyT)

% Point P Kinematic Coefficients
for vv = 1:1:361
    xp = (-R44 * coef(2,vv) * sin(thetaRad(4,vv))) - (R55 * coef(3,vv) *
sin(thetaRad(5,vv)));
    yp = (R44 * coef(2,vv) * cos(thetaRad(4,vv))) + (R55 * coef(3,vv) *
cos(thetaRad(5,vv)));
    xpp = (-R44 * coef2(2,vv) * sin(thetaRad(4,vv))) - (R44 * (coef(2,vv)^2)
* cos(thetaRad(4,vv))) - (R55 * coef2(3,vv) * sin(thetaRad(5,vv))) - (R55 *
(coef(3,vv)^2) * cos(thetaRad(5,vv)));
    ypp = (R44 * coef2(2,vv) * cos(thetaRad(4,vv))) - (R44 * (coef(2,vv)^2) *
sin(thetaRad(4,vv))) + (R55 * coef2(3,vv) * cos(thetaRad(5,vv))) - (R55 *
(coef(3,vv)^2) * sin(thetaRad(5,vv)));
    coefP(1,vv) = xp;
    coefP(2,vv) = yp;
    coefP(3,vv) = xpp;
    coefP(4,vv) = ypp;
    coefP(5,vv) = theta(2,vv);
end

figure(8)
plot(coefP(5,:),coefP(1,:),coefP(5,:),coefP(2,:))
xlabel('Input Posture (degrees)')
ylabel('Coefficient')
legend('X Prime', 'Y Prime')
xlim([0 360])
title('First Order Kinematic Coefficients for Point P')

figure(9)
plot(coefP(5,:),coefP(3,:),coefP(5,:),coefP(4,:))
xlabel('Input Posture (degrees)')
ylabel('Coefficient')
legend('X PPrime', 'Y PPrime')
```

```
xlim([0 360])
title('Second Order Kinematic Coefficients for Point P')

% Unit Normal and Tangent Vectors
for w = 1:1:361
    RpP(w) = sqrt((coefP(1,w)^2) + (coefP(2,w)^2));
    utx = coefP(1,w) / RpP(w);
    uty = coefP(2,w) / RpP(w);
    unx = -coefP(2,w) / RpP(w);
    uny = coefP(1,w) / RpP(w);
    rho(2,w) = (RpP(w)^3) / ((coefP(1,w) * coefP(4,w)) - (coefP(2,w) *
coefP(3,w)));
    rho(1,w) = theta(2,w);
    ut(1,w) = theta(2,w);
    ut(2,w) = utx;
    ut(3,w) = uty;
    un(1,w) = theta(2,w);
    un(2,w) = unx;
    un(3,w) = uny;
end

% Plot Rho
figure(10)
plot(rho(1,:),rho(2,:))
xlabel('Input Posture (degrees)')
ylabel('Radius of Curvature (mm)')
title('Radius of Curvature for Point P')

% Calculate and plot center of curfacture
for b = 1:1:361
    CC(1,b) = P(1,b) + (rho(2,b) * (-coefP(2,b) / RpP(b)));
    CC(2,b) = P(2,b) + (rho(2,b) * (coef(1,b) / RpP(b)));
end

figure(11)
plot(CC(1,:),CC(2,:))
xlabel('x position (mm)')
ylabel('y position (mm)')
title('Center of Curvature Position')

% Angluar Acceleration at 25 m/s
for x = 1:1:361
    a3 = coef2(1,x) * (25^2);
    a4 = coef2(2,x) * (25^2);
    a5 = coef2(3,x) * (25^2);
    a6 = coef2(4,x) * (25^2);
    alpha1(1,x) = theta(2,x);
    alpha1(2,x) = a3;
    alpha1(3,x) = a4;
    alpha1(4,x) = a5;
    alpha1(5,x) = a6;
end

figure(12)
plot(alpha1(1,:),alpha1(2,:),alpha1(1,:),alpha1(3,:),alpha1(1,:),alpha1(4,:),
alpha1(1,:),alpha1(5,:))
xlabel('Input Posture (degrees)')
```

```
ylabel('Angluar Acceleration (rad/s^2)')
legend('Link 3','Link 4','Link 5','Link 6')
xlim([0 360])
title('Angular Accelerations at an input of 25 m/s')

% Angular Accelerations at 50 m/s
for y = 1:1:361
    a3 = coef2(1,y) * (50^2);
    a4 = coef2(2,y) * (50^2);
    a5 = coef2(3,y) * (50^2);
    a6 = coef2(4,y) * (50^2);
    alpha2(1,y) = theta(2,y);
    alpha2(2,y) = a3;
    alpha2(3,y) = a4;
    alpha2(4,y) = a5;
    alpha2(5,y) = a6;
end

figure(13)
plot(alpha2(1,:),alpha2(2,:),alpha2(1,:),alpha2(3,:),alpha2(1,:),alpha2(4,:),
alpha2(1,:),alpha2(5,:))
xlabel('Input Posture (degrees)')
ylabel('Angluar Acceleration (rad/s^2)')
legend('Link 3','Link 4','Link 5','Link 6')
xlim([0 360])
title('Angular Accelerations at an input of 50 m/s')

% velocity and acceleration of point p at 25
for z = 1:1:361
    vpx = coefP(1,z) * 25;
    vpy = coefP(2,z) * 25;
    apx = coefP(3,z) * (25^2);
    apy = coefP(4,z) * (25^2);
    v(1,z) = theta(2,z);
    v(2,z) = vpx;
    v(3,z) = vpy;
    a(1,z) = theta(2,z);
    a(2,z) = apx;
    a(3,z) = apy;
end

for ii = 1:1:361
    vpx = coefP(1,ii) * 50;
    vpy = coefP(2,ii) * 50;
    apx = coefP(3,ii) * (50^2);
    apy = coefP(4,ii) * (50^2);
    v(4,ii) = vpx;
    v(5,ii) = vpy;
    a(4,ii) = apx;
    a(5,ii) = apy;
end

% figures for velocities and accels of P
figure(14)
plot(v(1,:),v(2,:),v(1,:),v(3,:),v(1,:),v(4,:),v(1,:),v(5,:))
xlabel('Input Posture')
ylabel('velocity (mm/s)')
```

```
legend('x at 25','y at 25','x at 50','y at 50')
title('Point P x and y Velocities')

figure(15)
plot(a(1,:),a(2,:),a(1,:),a(3,:),a(1,:),a(4,:),a(1,:),a(5,:))
xlabel('Input Posture')
ylabel('acceleration (mm/s)')
legend('x at 25','y at 25','x at 50','y at 50')
title('Point P x and y Accelerations')

% exporting to csv
q = 1;
for p = 1:10:361
    coefS(:,q) = coef(:,p);
    detS(:,q) = Det(:,p);
    thetaS(:,q) = theta(:,p);
    vel25S(:,q) = vel1(:,p);
    vel50S(:,q) = vel2(:,p);
    coef2s(:,q) = coef2(:,p);
    alpha25s(:,q) = alpha1(:,p);
    alpha50s(:,q) = alpha2(:,p);
    Ps(:,q) = P(:,p);
    coefPs(:,q) = coefP(:,p);
    uts(:,q) = ut(:,p);
    uns(:,q) = un(:,p);
    rhos(:,q) = rho(:,p);
    CCs(:,q) = CC(:,p);
    vs(:,q) = v(:,p);
    as(:,q) = a(:,p);
    q = q+1;
end

writematrix(coefS, 'coef.csv');
writematrix(detS, 'det.csv');
writematrix(thetaS, 'theta.csv');
writematrix(vel25S, 'vel25.csv');
writematrix(vel50S, 'vel50.csv');
writematrix(coef2s, 'coef2.csv');
writematrix(alpha25s, 'accel25.csv');
writematrix(alpha50s, 'accel50.csv');
writematrix(Ps, 'P.csv');
writematrix(coefPs, 'coefP.csv');
writematrix(uts, 'ut.csv');
writematrix(uns, 'un.csv');
writematrix(rhos, 'rho.csv');
writematrix(CCs, 'CC.csv');
writematrix(vs, 'vP.csv');
writematrix(as, 'aP.csv');

% PROJECT 2 DELIVERABLE 2

% Find kinematic coefficients of point G3
for kk = 1:361
    g3xp = (-R2 * sin(thetaRad(2,kk))) - ((R3 / 2) * coef(1,kk) *
sin(thetaRad(3,kk)));
    g3yp = (R2 * cos(thetaRad(2,kk))) + ((R3 / 2) * coef(1,kk) *
cos(thetaRad(3,kk)));
```

```
    g3xpp = (-R2 * cos(thetaRad(2,kk))) - ((R3 / 2) * coef2(1,kk) *
sin(thetaRad(3,kk))) - ((R3/2) * (coef(1,kk)^2) * cos(thetaRad(3,kk)));
    g3ypp = (-R2 * sin(thetaRad(2,kk))) + ((R3 / 2) * coef2(1,kk) *
cos(thetaRad(3,kk))) - ((R3/2) * (coef(1,kk)^2) * sin(thetaRad(3,kk)));
    coefG3(1,kk) = g3xp;
    coefG3(2,kk) = g3yp;
    coefG3(3,kk) = g3xpp;
    coefG3(4,kk) = g3ypp;
end

% Find all needed accelerations
for ll = 1:361
    ag3x = coefG3(3,ll) * 25 * 25;
    ag3y = coefG3(4,ll) * 25 * 25;
    ag5x = a(2,ll);
    ag5y = a(3,ll);

    accelX(1,ll) = ag3x;
    accelX(2,ll) = ag5x;
    accelY(1,ll) = ag3y;
    accelY(2,ll) = ag5y;
end

% Dynamic Analysis
for jj = 1:361
    AA = 0;
    AB = m2 * G;
    AC = 0;
    AD = m3 * accelX(1,jj);
    AE = m3 * (accelY(1,jj) + G);
    AF = (ig3 * alpha1(2,jj)) + ((R3 * m3 /
2)*((cos(thetaRad(3,jj))*accelY(1,jj)) - (sin(thetaRad(3,jj))*accelX(1,jj)) +
(G*cos(thetaRad(3,jj)))));
    AG = 0;
    AH = m4 * G;
    AI = ig4 * alpha1(3,jj);
    AJ = m5 * accelX(2,jj);
    AK = m5 * (G + accelY(2,jj));
    AL = (ig5 * alpha1(4,jj)) + ((R5 * m5 /
2)*((cos(thetaRad(5,jj))*accelY(2,jj)) - (sin(thetaRad(5,jj))*accelX(2,jj)) +
(G*cos(thetaRad(5,jj)))));
    AM = 0;
    AN = m6 * G;
    AO = ig6 * alpha1(5,jj);

    matrix(1,:) = [1,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,AA];
    matrix(2,:) = [0,1,0,-1,0,0,0,0,0,0,0,0,0,0,0,AB];
    matrix(3,:) = [0,0,(R2*sin(thetaRad(2,jj))),-
(R2*cos(thetaRad(2,jj))),0,0,0,0,0,0,0,0,0,0,1,AC];
    matrix(4,:) = [0,0,1,0,-1,0,0,0,0,0,0,0,0,0,0,AD];
    matrix(5,:) = [0,0,0,1,0,-1,0,0,0,0,0,0,0,0,0,AE];
    matrix(6,:) = [0,0,0,0,(R3*sin(thetaRad(3,jj))),(-
R3*cos(thetaRad(3,jj))),0,0,0,0,0,0,0,0,0,AF];
    matrix(7,:) = [0,0,0,0,1,0,-1,0,0,0,1,0,0,0,0,AG];
    matrix(8,:) = [0,0,0,0,0,1,0,-1,0,0,0,1,0,0,0,AH];
```

```
    matrix(9,:) = [0,0,0,0,(-
R4*sin(thetaRad(4,jj))),(R4*cos(thetaRad(4,jj))),(R44*sin(thetaRad(4,jj))),-
(R44*cos(thetaRad(4,jj))),0,0,0,0,0,0,0,AI];
    matrix(10,:) = [0,0,0,0,0,0,1,0,-1,0,0,0,0,0,0,AJ];
    matrix(11,:) = [0,0,0,0,0,0,0,1,0,-1,0,0,0,0,0,AK];
    matrix(12,:) = [0,0,0,0,0,0,0,0,(R5*sin(thetaRad(5,jj))),-
(R5*cos(thetaRad(5,jj))),0,0,0,0,0,AL];
    matrix(13,:) = [0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,AM];
    matrix(14,:) = [0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,AN];
    matrix(15,:) = [0,0,0,0,0,0,0,0,(-
R6*sin(thetaRad(6,jj))),(R6*cos(thetaRad(6,jj))),0,0,0,0,0,AO];
    matrix = rref(matrix);
    for mm = 1:15
        results(mm+1,jj) = matrix(mm,16);
    end
    results(1,jj) = theta(2,jj);
end

% Static Analysis
for jj = 1:361
    AA = 0;
    AB = m2 * G;
    AC = 0;
    AD = 0;
    AE = m3 * G;
    AF = (R3 * m3 / 2)*(G*cos(thetaRad(3,jj)));
    AG = 0;
    AH = m4 * G;
    AI = 0;
    AJ = 0;
    AK = 0;
    AL = (R5 * m5 / 2)*(G*cos(thetaRad(5,jj)));
    AM = 0;
    AN = m6 * G;
    AO = 0;

    matrix(1,:) = [1,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,AA];
    matrix(2,:) = [0,1,0,-1,0,0,0,0,0,0,0,0,0,0,0,AB];
    matrix(3,:) = [0,0,(R2*sin(thetaRad(2,jj))),-
(R2*cos(thetaRad(2,jj))),0,0,0,0,0,0,0,0,0,0,1,AC];
    matrix(4,:) = [0,0,1,0,-1,0,0,0,0,0,0,0,0,0,0,AD];
    matrix(5,:) = [0,0,0,1,0,-1,0,0,0,0,0,0,0,0,0,AE];
    matrix(6,:) = [0,0,0,0,(R3*sin(thetaRad(3,jj))),(-
R3*cos(thetaRad(3,jj))),0,0,0,0,0,0,0,0,0,AF];
    matrix(7,:) = [0,0,0,0,1,0,-1,0,0,0,1,0,0,0,0,AG];
    matrix(8,:) = [0,0,0,0,0,1,0,-1,0,0,0,1,0,0,0,AH];
    matrix(9,:) = [0,0,0,0,(-
R4*sin(thetaRad(4,jj))),(R4*cos(thetaRad(4,jj))),(R44*sin(thetaRad(4,jj))),-
(R44*cos(thetaRad(4,jj))),0,0,0,0,0,0,0,AI];
    matrix(10,:) = [0,0,0,0,0,0,1,0,-1,0,0,0,0,0,0,AJ];
    matrix(11,:) = [0,0,0,0,0,0,0,1,0,-1,0,0,0,0,0,AK];
    matrix(12,:) = [0,0,0,0,0,0,0,0,(R5*sin(thetaRad(5,jj))),-
(R5*cos(thetaRad(5,jj))),0,0,0,0,0,AL];
    matrix(13,:) = [0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,AM];
    matrix(14,:) = [0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,AN];
    matrix(15,:) = [0,0,0,0,0,0,0,0,(-
R6*sin(thetaRad(6,jj))),(R6*cos(thetaRad(6,jj))),0,0,0,0,0,AO];
```

```
    matrix = rref(matrix);
    for mm = 1:15
        staticResults(mm+1,jj) = matrix(mm,16);
    end
    staticResults(1,jj) = theta(2,jj);
end

figure(15)
plot(results(1,:),results(2,:),results(1,:),results(3,:))
xlabel('Input Posture (degrees)')
ylabel('Force (N)')
legend('F12x','F12y')
xlim([0 360])

figure(16)
plot(results(1,:),results(4,:),results(1,:),results(5,:))
xlabel('Input Posture (degrees)')
ylabel('Force (N)')
legend('F23x','F23y')
xlim([0 360])

figure(17)
plot(results(1,:),results(6,:),results(1,:),results(7,:))
xlabel('Input Posture (degrees)')
ylabel('Force (N)')
legend('F34x','F34y')
xlim([0 360])

figure(18)
plot(results(1,:),results(8,:),results(1,:),results(9,:))
xlabel('Input Posture (degrees)')
ylabel('Force (N)')
legend('F45x','F45y')
xlim([0 360])

figure(19)
plot(results(1,:),results(10,:),results(1,:),results(11,:))
xlabel('Input Posture (degrees)')
ylabel('Force (N)')
legend('F56x','F56y')
xlim([0 360])

figure(20)
plot(results(1,:),results(12,:),results(1,:),results(13,:))
xlabel('Input Posture (degrees)')
ylabel('Force (N)')
legend('F14x','F14y')
xlim([0 360])

figure(21)
plot(results(1,:),results(14,:),results(1,:),results(15,:))
xlabel('Input Posture (degrees)')
ylabel('Force (N)')
legend('F16x','F16y')
xlim([0 360])

figure(22)
```

```
plot(staticResults(1,:),staticResults(2,:),staticResults(1,:),staticResults(3
,:))
xlabel('Input Posture (degrees)')
ylabel('Force (N)')
legend('F12x','F12y')
xlim([0 360])

figure(23)
plot(staticResults(1,:),staticResults(4,:),staticResults(1,:),staticResults(5
,:))
xlabel('Input Posture (degrees)')
ylabel('Force (N)')
legend('F23x','F23y')
xlim([0 360])

figure(24)
plot(staticResults(1,:),staticResults(6,:),staticResults(1,:),staticResults(7
,:))
xlabel('Input Posture (degrees)')
ylabel('Force (N)')
legend('F34x','F34y')
xlim([0 360])

figure(25)
plot(staticResults(1,:),staticResults(8,:),staticResults(1,:),staticResults(9
,:))
xlabel('Input Posture (degrees)')
ylabel('Force (N)')
legend('F45x','F45y')
xlim([0 360])

figure(26)
plot(staticResults(1,:),staticResults(10,:),staticResults(1,:),staticResults(
11,:))
xlabel('Input Posture (degrees)')
ylabel('Force (N)')
legend('F56x','F56y')
xlim([0 360])

figure(27)
plot(staticResults(1,:),staticResults(12,:),staticResults(1,:),staticResults(
13,:))
xlabel('Input Posture (degrees)')
ylabel('Force (N)')
legend('F14x','F14y')
xlim([0 360])

figure(28)
plot(staticResults(1,:),staticResults(14,:),staticResults(1,:),staticResults(
15,:))
xlabel('Input Posture (degrees)')
ylabel('Force (N)')
legend('F16x','F16y')
xlim([0 360])

% Find all needed accelerations at 50 rad/s
for ll = 1:361
```

```
    ag3x = coefG3(3,ll) * 50 * 50;
    ag3y = coefG3(4,ll) * 50 * 50;
    ag5x = a(4,ll);
    ag5y = a(5,ll);

    accelX2(1,ll) = ag3x;
    accelX2(2,ll) = ag5x;
    accelY2(1,ll) = ag3y;
    accelY2(2,ll) = ag5y;

end

% Dynamic Analysis at 50 rad/s
for jj = 1:361
    AA = 0;
    AB = m2 * G;
    AC = 0;
    AD = m3 * accelX2(1,jj);
    AE = m3 * (accelY2(1,jj) + G);
    AF = (ig3 * alpha2(2,jj)) + ((R3 * m3 /
2)*((cos(thetaRad(3,jj))*accelY2(1,jj)) - (sin(thetaRad(3,jj))*accelX2(1,jj))
+ (G*cos(thetaRad(3,jj)))));
    AG = 0;
    AH = m4 * G;
    AI = ig4 * alpha2(3,jj);
    AJ = m5 * accelX2(2,jj);
    AK = m5 * (G + accelY2(2,jj));
    AL = (ig5 * alpha2(4,jj)) + ((R5 * m5 /
2)*((cos(thetaRad(5,jj))*accelY2(2,jj)) - (sin(thetaRad(5,jj))*accelX2(2,jj))
+ (G*cos(thetaRad(5,jj)))));
    AM = 0;
    AN = m6 * G;
    AO = ig6 * alpha2(5,jj);

    matrix(1,:) = [1,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,AA];
    matrix(2,:) = [0,1,0,-1,0,0,0,0,0,0,0,0,0,0,0,AB];
    matrix(3,:) = [0,0,(R2*sin(thetaRad(2,jj))),-
(R2*cos(thetaRad(2,jj))),0,0,0,0,0,0,0,0,0,0,1,AC];
    matrix(4,:) = [0,0,1,0,-1,0,0,0,0,0,0,0,0,0,0,AD];
    matrix(5,:) = [0,0,0,1,0,-1,0,0,0,0,0,0,0,0,0,AE];
    matrix(6,:) = [0,0,0,0,(R3*sin(thetaRad(3,jj))),(-
R3*cos(thetaRad(3,jj))),0,0,0,0,0,0,0,0,0,AF];
    matrix(7,:) = [0,0,0,0,1,0,-1,0,0,0,1,0,0,0,0,AG];
    matrix(8,:) = [0,0,0,0,0,1,0,-1,0,0,0,1,0,0,0,AH];
    matrix(9,:) = [0,0,0,0,(-
R4*sin(thetaRad(4,jj))),(R4*cos(thetaRad(4,jj))),(R44*sin(thetaRad(4,jj))),-
(R44*cos(thetaRad(4,jj))),0,0,0,0,0,0,0,AI];
    matrix(10,:) = [0,0,0,0,0,0,1,0,-1,0,0,0,0,0,0,AJ];
    matrix(11,:) = [0,0,0,0,0,0,0,1,0,-1,0,0,0,0,0,AK];
    matrix(12,:) = [0,0,0,0,0,0,0,0,(R5*sin(thetaRad(5,jj))),-
(R5*cos(thetaRad(5,jj))),0,0,0,0,0,AL];
    matrix(13,:) = [0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,AM];
    matrix(14,:) = [0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,AN];
    matrix(15,:) = [0,0,0,0,0,0,0,0,(-
R6*sin(thetaRad(6,jj))),(R6*cos(thetaRad(6,jj))),0,0,0,0,0,AO];
    matrix = rref(matrix);
    for mm = 1:15
```

```
        results50(mm+1,jj) = matrix(mm,16);
    end
    results50(1,jj) = theta(2,jj);
end

figure(29)
plot(results(1,:),results(16,:))
xlabel('Input Posture (degrees)')
ylabel('Torque (Nm)')

figure(30)
plot(results50(1,:),results50(16,:))
xlabel('Input Posture (degrees)')
ylabel('Torque (Nm)')

figure(31)
plot(results(1,:),results(16,:),results50(1,:),results50(16,:))
xlabel('Input Posture (degrees)')
ylabel('Torque (Nm)')
legend('25 m/s','50 m/s')
xlim([0 360])

figure(32)
plot(staticResults(1,:),staticResults(16,:))
xlabel('Input Posture (degrees)')
ylabel('Torque (Nm)')
xlim([0 360])

% Converting to Tables
coefG3(5,:) = coefG3(4,:);
coefG3(4,:) = coefG3(3,:);
coefG3(3,:) = coefG3(2,:);
coefG3(2,:) = coefG3(1,:);
coefG3(1,:) = theta(2,:);

part2(1,:) = coefG3(1,:);
part2(2,:) = coefG3(2,:);
part2(3,:) = coefG3(3,:);
part2(4,:) = coefG3(4,:);
part2(5,:) = coefG3(5,:);
part2(6,:) = accelX(1,:);
part2(7,:) = accelY(1,:);
part2(8,:) = coefP(1,:);
part2(9,:) = coefP(2,:);
part2(10,:) = coefP(3,:);
part2(11,:) = coefP(4,:);
part2(12,:) = accelX(2,:);
part2(13,:) = accelY(2,:);

part5(1,:) = results50(1,:);
part5(2,:) = results50(16,:);

part2 = part2.';
results = results.';
staticResults = staticResults.';
part5 = part5.';

kk = 1;
```

```
for jk = 1:10:361
    part2s(kk,:) = part2(jk,:);
    resultss(kk,:) = results(jk,:);
    staticResultss(kk,:) = staticResults(jk,:);
    part5s(kk,:) = part5(jk,:);
    kk = kk + 1;
end

writematrix(part2s, 'part2.csv');
writematrix(resultss, 'part3.csv');
writematrix(staticResultss, 'part4.csv');
writematrix(part5s, 'part5.csv');

% PROJECT 2 DELIVERABLE 2

% Equivelant Mass Moment of Full Machine
for tt = 1:361
    i3(tt) = (m3*((coefG3(2,tt)^2) + (coefG3(3,tt)^2))) + (ig3 *
(coef(1,tt)^2));
    i4(tt) = ig4 * (coef(2,tt)^2);
    i5(tt) = (m5*((coefP(1,tt)^2) + (coefP(2,tt)^2))) + (ig5 *
(coef(3,tt)^2));
    i6(tt) = ig6 * (coef(4,tt)^2);
    ieq(1,tt) = theta(2,tt);
    ieq(2,tt) = i3(tt) + i4(tt) + i5(tt) + i6(tt) + ig2;
end

figure(33)
plot(ieq(1,:),ieq(2,:))
xlabel('Input Posture (degrees)')
ylabel('Equivalent Mass Moment (kgm^2)')
xlim([0 360])
ylim([0 0.09])

% Kinetic Energy
for zz = 1:361
    b3(zz) = (m3 * ((coefG3(2,zz)*coefG3(4,zz)) +
(coefG3(3,zz)*coefG3(5,zz)))) + (ig3 * coef(1,zz) * coef2(1,zz));
    b4(zz) = ig4 * coef(2,zz) * coef2(2,zz);
    b5(zz) = (m5 * (((coefP(1,zz)*coefP(3,zz)) + (coefP(2,zz)*coefP(4,zz)))))
+ (ig5 * coef(3,zz) * coef2(3,zz));
    b6(zz) = ig6 * coef(4,zz) * coef2(4,zz);
    B(1,zz) = theta(2,zz);
    B(2,zz) = b3(zz) + b4(zz) + b5(zz) + b6(zz);
end

for uu = 1:361
    T(uu) = .5 * 25 * 25 * ieq(2,uu);
    dT(uu) = B(2,uu) * (25^3);
end

figure(34)
plot(theta(2,:),T)
xlabel('Input Posture (degrees)')
ylabel('Kinetic Energy (J)')
xlim([0 360])

figure(35)
```

```
plot(theta(2,:),dT)
xlabel('Input Posture (degrees)')
ylabel('Time Derivative of Kinetic Energy (W)')
xlim([0 360])

% Gravitational Potential Energy
for yy = 1:361
    yg3(yy) = (R2 * sin(thetaRad(2,yy))) + ((R3 / 2) * sin(thetaRad(3,yy)));
end

for xx = 1:361
    Ug3(xx) = m3 * G * yg3(xx);
    Ug5(xx) = m5 * G * P(2,xx);
    Ug(xx) = Ug3(xx) + Ug5(xx);
    dUg3(xx) = m3 * G * 25 * coefG3(3,xx);
    dUg5(xx) = m5 * G * 25 * coefP(2,xx);
    dUg(xx) = dUg3(xx) + dUg5(xx);
end

figure(36)
plot(theta(2,:),Ug)
xlabel('Input Posture (degrees)')
ylabel('Gravitational Potential Energy (J)')
xlim([0 360])

figure(37)
plot(theta(2,:),dUg)
xlabel('Input Posture (degrees)')
ylabel('Time Derivative of Gravitational Potential Energy (W)')
xlim([0 360])

% Spring Potential Energy
for ww = 1:361
    sprL(ww) = (((P(1,ww)^2) + (P(2,ww)^2))^(1/2));
    delL(ww) = sprL(ww) - .150;
end

for rr = 1:361
    coefR(rr) = ((P(1,rr) * coefP(1,rr)) + (coefP(2,rr) * P(2,rr))) /
sprL(rr);
end

for vv = 1:361
    Usp(vv) = .5 * 5000 * delL(vv)^2;
    dUsp(vv) = 5000 * delL(vv) * coefR(vv) * 25;
end

figure(38)
plot(theta(2,:),coefR)
xlabel('Input Posture (degrees)')
ylabel('First Order Kinematic Coefficient of the Spring (m/rad)')
xlim([0 360])

figure(39)
plot(theta(2,:),Usp(:))
xlabel('Input Posture (degrees)')
ylabel('Spring Potential Energy (J)')
xlim([0 360])
```

```
figure(40)
plot(theta(2,:),dUsp(:))
xlabel('Input Posture (degrees)')
ylabel('Time Derivative of Spring Potential Energy (W)')
xlim([0 360])

% Total Potential Energy
for ss = 1:361
    U(ss) = Ug(ss) + Usp(ss);
    dU(ss) = dUg(ss) + dUsp(ss);
end

figure(41)
plot(theta(2,:), U(:))
xlabel('Input Posture (degrees)')
ylabel('Total Potential Energy (J)')
xlim([0 360])

figure(42)
plot(theta(2,:), dU(:))
xlabel('Input Posture (degrees)')
ylabel('Total Potential Power (W)')
xlim([0 360])

% Viscous Damper Effects
right = max(P(1,:));

for qq = 1:361
    W(qq) = 350 * -coefP(1,qq) * 25 * (right - P(1,qq));
    dW(qq) = 350 * (coefP(1,qq)^2) * (25^2);
end

figure(43)
plot(theta(2,:),W(:))
xlabel('Input Posture (degrees)')
ylabel('Viscous Damper Effects (J)')
xlim([0 360])

figure(44)
plot(theta(2,:),dW(:))
xlabel('Input Posture (degrees)')
ylabel('Time Derivative of Viscous Damper Effects (W)')
xlim([0 360])

% Power and Contributions
for oo = 1:361
    power(oo) = dT(oo) + dU(oo) + dW(oo);
    Tpower(oo) = abs(dT(oo)) + abs(dU(oo)) + abs(dW(oo));
end

for pp = 1:361
    kinCon(pp) = abs(dT(pp)) / Tpower(pp);
    gravCon(pp) = abs(dUg(pp)) / Tpower(pp);
    sprCon(pp) = abs(dUsp(pp)) / Tpower(pp);
    damCon(pp) = abs(dW(pp)) / Tpower(pp);
end
```

```
for ccc = 1:361
    kinCon2(ccc) = dT(ccc)/power(ccc);
    gravCon2(ccc) = dUg(ccc)/power(ccc);
    sprCon2(ccc) = dUsp(ccc)/power(ccc);
    damCon2(ccc) = dW(ccc)/power(ccc);
end

figure(45)
plot(theta(2,:),power(:))
xlabel('Input Posture (degrees)')
ylabel('Net Power (W)')
xlim([0 360])

figure(46)
plot(theta(2,:),kinCon(:),theta(2,:),gravCon(:),theta(2,:),sprCon(:),theta(2,
:),damCon(:))
xlabel('Input Posture (degrees)')
ylabel('Contribution')
legend('Kinetic Contribution','Graviational Contribution','Spring
Contribution','Viscous Damper Contribution')
xlim([0 360])

% Torque
for nn = 1:361
    torque(nn) = power(nn) / 25;
end

figure(47)
plot(theta(2,:),torque(:))
xlabel('Input Posture (degrees)')
ylabel('Input Torque (Nm)')
xlim([0 360])

figure(48)
plot(theta(2,:),results(:,16),theta(2,:),torque(:))
xlabel('Input Posture (degrees)')
ylabel('Input Torque (Nm)')
legend('Dynamic Analysis Torque','Power Equation Torque')
xlim([0 360])

figure(49)
plot(theta(2,:),-coefP(1,:))
xlabel('Input Posture (degrees)')
ylabel('First Order Kinematic Coefficient of the Damper (m/rad)')
xlim([0 360])

figure(50)
plot(theta(2,:),kinCon2(:),theta(2,:),gravCon2(:),theta(2,:),sprCon2(:),theta
(2,:),damCon2(:))
xlabel('Input Posture (degrees)')
ylabel('contribution')
legend('Kinetic','Gravitational','Spring','Damper')
xlim([0 360])

% Exporting to Tables
bbb = 1;
for aaa = 1:10:361
    p2d2p1(bbb,1) = ieq(1,aaa);
```

```
    p2d2p1(bbb,2) = ieq(2,aaa);
    p2d2p1(bbb,3) = T(aaa);
    p2d2p1(bbb,4) = dT(aaa);
    Usps(bbb,1) = theta(2,aaa);
    Usps(bbb,2) = coefR(aaa);
    Usps(bbb,3) = Usp(aaa);
    Usps(bbb,4) = dUsp(aaa);
    Ugs(bbb,1) = theta(2,aaa);
    Ugs(bbb,2) = Ug(aaa);
    Ugs(bbb,3) = dUg(aaa);
    Ws(bbb,1) = theta(2,aaa);
    Ws(bbb,2) = -coefP(1,aaa);
    Ws(bbb,3) = W(aaa);
    Ws(bbb,4) = dW(aaa);
    torques(bbb,1) = theta(2,aaa);
    torques(bbb,2) = torque(aaa);
    cont(bbb,1) = theta(2,aaa);
    cont(bbb,2) = power(aaa);
    cont(bbb,3) = kinCon2(aaa);
    cont(bbb,4) = gravCon2(aaa);
    cont(bbb,5) = sprCon2(aaa);
    cont(bbb,6) = damCon2(aaa);
    cont(bbb,7) = kinCon(aaa);
    cont(bbb,8) = gravCon(aaa);
    cont(bbb,9) = sprCon(aaa);
    cont(bbb,10) = damCon(aaa);
    compare(bbb,1) = theta(2,aaa);
    compare(bbb,2) = results(aaa,16);
    compare(bbb,3) = torque(aaa);
    bbb = bbb + 1;
end

writematrix(p2d2p1, 'p2d2p1.csv')
writematrix(Usps, 'Usps.csv')
writematrix(Ugs, 'Ugs.csv')
writematrix(Ws, 'Ws.csv')
writematrix(torques, 'torque.csv')
writematrix(cont, 'cont.csv')
writematrix(compare, 'comparevals.csv')

% General Contribution of each Using Method 2
kin = mean(kinCon)*100;
grav = mean(gravCon)*100;
spr = mean(sprCon)*100;
dam = mean(damCon)*100;

fprintf('The Average Contribution due to Kinetic Energy is %.2f %',kin)
fprintf('\nThe Average Contribution due to Graviational Potential Energy is
%.2f %',grav)
fprintf('\nThe Average Contribution due to Spring Potential Energy is %.2f
%',spr)
fprintf('\nThe Average Contribution due to Viscous Damping Effects is %.2f
%',dam)

% END OF CODE
```