Noah Teal, John Jeong, Tim Regrut
Professor Janiszewska
Executive Summary Lab 6
2 March 2016

Introduction

The objective of this lab was to develop a program to get the AEV from the maintenance station to the Grand Canyon pick up station and then back down to just past the curve. The purpose of this program was to demonstrate the student's understanding of the Arduino, find out which propeller speed would be the smartest choice, and make any changes if needed. The next task was to download the Arduino EEPROM data and convert the EEPROM data into Physical parameters. The student's following task was to write a script file in MATLAB to make the proper conversions. After the conversions were made, the students made a performance analysis and a graph to go with the data. This helped optimize the student's understanding of what happened in the lab and contribute to future decisions.

Experimental Setup

The AEV was carefully scrutinized for loose screws and components before testing. The connected battery was checked for charge to insure successful experimentation. The student opened the sketchbook file in the Arduino software and connected its laptop to the AEV with a mini USB cord.

Results

After the AEV performed the run, the EEPROM data was recorded in the arduino board. The data was converted into workable values in MATLAB, and were calculated for velocity, propulsion efficiency, and supplied power. A matlab function was used to plot the values, with which the run can be evaluated.

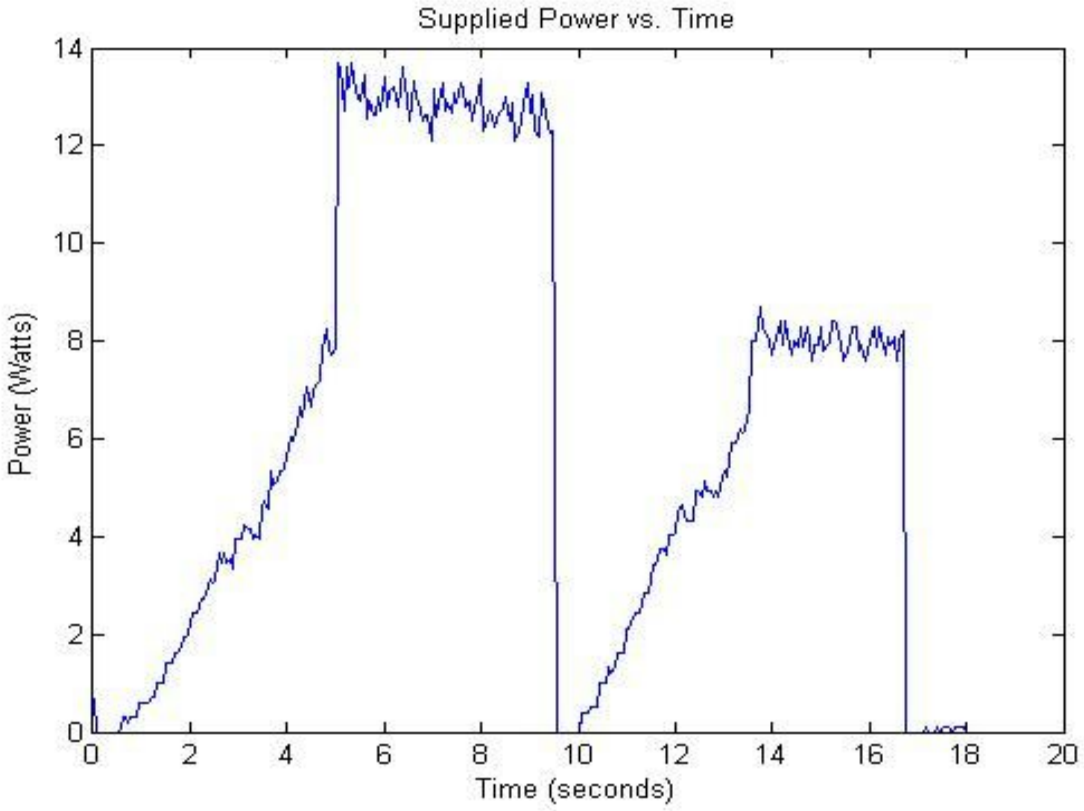

Figure 1. Supplied power (watts) in the AEV based on time elapsed in seconds

In Figure 1., the peaks/ranges for power (watts) with time (seconds) follows the supply of power that was written in code. During the run towards the top of the hill, the celerate function in Arduino increased supplied power from 0% to 38%. Keeping this high power for a planned interval of time, the AEV stopped immediately after a planned period of time. Figure 1. replicates this behavior during the time interval 0 to 10 seconds. After the 10 second point, the AEV ran backwards to the bottom of the hill, with a similar, yet less powerful function of code. Figure 1. replicates the similar, yet less powerful function, with a shorter version of the peak/range at the beginning. These two “back and forth” power peak/range can be compared for further efficiency analysis in uphill travel. A coexistence between supplied power usage and inertia can established during analysis of data.

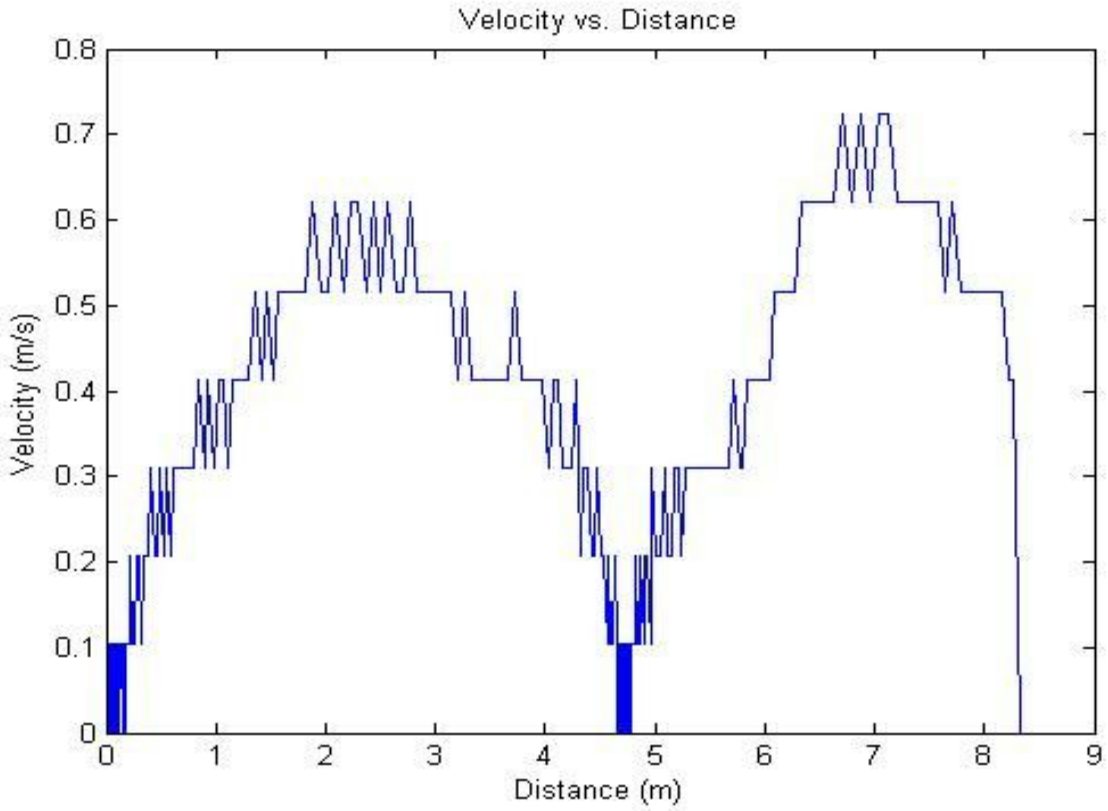

Figure 2. Velocity (in meters per second) of the AEV based on its distance traveled

Figure 2. helps aid the team's coding strategy because the velocity from the second peak, at approximately 7 meters which is when the AEV is going down the hill, is higher than the first peak which is at around 2 to 3 meters traveled. This is because the AEV is going down to the lower end of the track during that portion of the figure and is going faster than it needs to go to move. The team could lower the speed while the AEV is going down the hill to save energy.

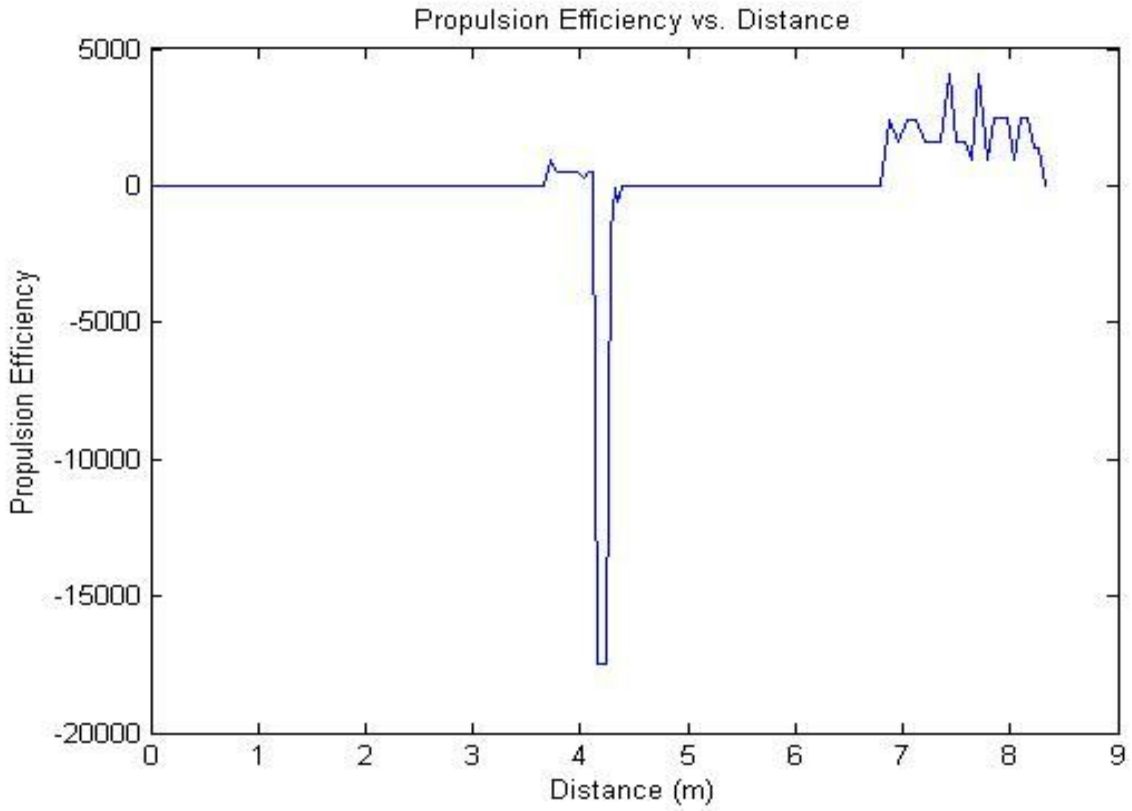

Figure 3. propulsion efficiency of the AEV based on its distance traveled

Figure 3. helps the team's coding strategy because the only time the AEV loses a significant amount of energy is when it is climbing up the hill and the team could increase the speed of the AEV when it is on that part of the track to insure that the AEV makes it up the hill every single time. After the AEV makes it up the hill the team's coding strategy could be the same as before, when the AEV went up the hill which is a steady, moderate speed.

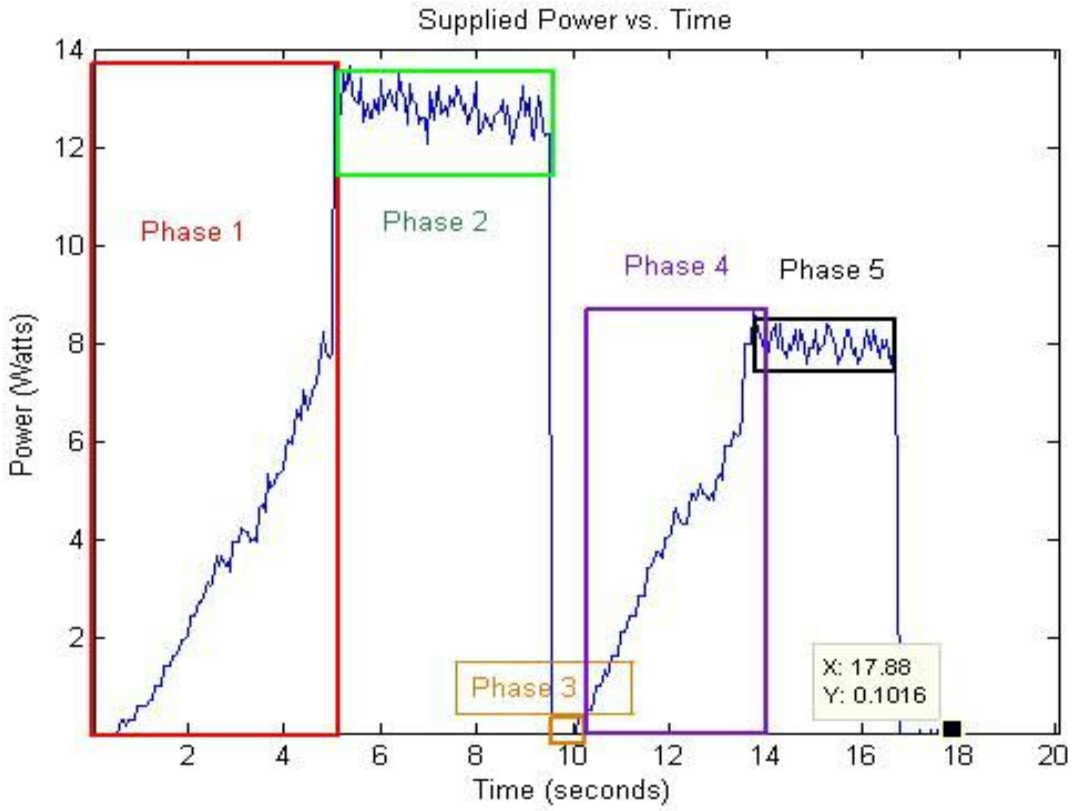

Figure 4. power (watts) vs. time elapsed in seconds, separated into phases pertaining to code

Figure 4. helps aid the team's coding strategy because it shows the energy used by the AEV. Since the AEV is traveling down the hill in phase 5 the speed could be lowered so that the AEV saves energy.

Phase	**Arduino Code**	**Distance (m)**	**Time(s)**	**Energy used in each phase (J)**
1	reverse(4); celerate(4,0,35, 5);	**0.1860**	**5.0430**	**8.0543**
2	motorSpeed(4,3 8); goToRelativePo sition(-280);	**3.4844**	**4.4400**	**78.4117**
3	motorSpeed(4,0);	**0.0868**	**0.1200**	**3.0004**

4	reverse(4); celerate(4,0,25, 4);	**0.6944**	**3.7200**	**63.8369**
5	motorSpeed(4,2 5); goToRelativePo sition(160);	**1.9840**	**2.9400**	**51.4063**
total energy				**204.7095**

Table 1. arduino code used, distance , time elapsed, total energy used in each phase joules, and the total energy used (joules)

Table 1. helps aid the team's coding strategy because it gives numerical values for each phase of the AEV. Phase 5 is when the AEV is going down the hill and it is the second highest amount of energy used. This will help aid the team's coding strategy because since the AEV is getting the benefit of gravity, the speed of the AEV can be lowered which will save energy and keep the AEV from going off the track.

During the lab when the AEV was traveling from the top of the hill to the bottom of the hill it was going past its starting point so the team had to run multiple trial runs to obtain a code that would allow the AEV to return back to its original position.

The next step to improving the AEV would be to lower the speed of the AEV in phase 5 (in table 1 and figure 4.), which is when the AEV goes down the hill. This would help the safety aspect of the AEV because if the AEV goes off the track the passengers would be injured in the simulation and the AEV could be damaged. Another part of the code that could be changed is in phase 4. The modification would be to increase the speed of the AEV so that it always makes it up the hill. This change would be made because in figure 3 the AEV loses propeller efficiency when it goes up the hill.

Arduino Code (w/ comments)

```
//up the hill
 reverse(4); (the propellers must be reversed in code so that the AEV can run forward)
 celerate(4,0,35,5);
 motorSpeed(4,38); (38% supplied power was the highest in this code)
 goToRelativePosition(-280);

 motorSpeed(4,0); (stop the propellers and let inertia carry the AEV)

 //down the hill
```

```
 reverse(4);
 celerate(4,0,25,4); (a lower supplied power was used for the downhill direction)
 motorSpeed(4,25);
 goToRelativePosition(160); (this used a mark of 160, so that the propellers can be shut off early)

 motorSpeed(4,0);
```

Matlab Code (w/ comments)

```
clear,clc

(initialization of the EEPROM data)
data=xlsread('Moneyisthemotive.xlsx');
te=data(1:307,1);
ve=data(1:307,3);
ie=data(1:307,2);
marks=data(1:307,4);
pos=data(1:307,5);
pos=abs(pos);
m=.450;
d=3;

(initialization of newer data for lab 7)
t=te./1000;
v=(15.*ve)./1024;
I=(ie./1024)*(2.46)*(1/.185);
d=0.0124.*marks;
s=.0124.*pos;
p=v.*I;

(for loop code used incremental energy equation to create vector of the value)
ej=1:length(p);
for i=1:1:length(p)-1;
    ej(i)=((p(i)+p(i+1))/2)*(t(i+1)-t(i));
end

(similar to the last for loop, velocity vector is created)
v=1:length(d);
for i=1:1:length(d)-1;
    v(i)=((d(i+1)-d(i))/2)/(t(i+1)-t(i));
end
v(307)=0;
```

(kinetic energy vector is created)

```
ke=1:length(v);
for i=1:1:length(v);
   ke(i)=(1/2)*m*(v(i))^2;
end
ke(307)=0;
```

(rpm vector is created)

```
rpm=1:length(I);
for i=1:1:length(I);
   rpm(i)=-64.59*I(i)^2+1927.25*I(i)-84.58;
end
rpm(307)=0;
```

(propeller advance ratio)

```
J=1:length(rpm);
for i=1:1:length(rpm);
   J(i)=(v(i))/(rpm(i)/(60*3));
end
J(307)=0;
```

(propulsion efficiency)

```
n=1:length(J);
for i=1:1:length(J);
   n(i)=-454.37*J(i)^3+321.58*J(i)^2+22.603*J(i);
end
n(307)=0;
```

(below is the code to plot the created vectors by requirement)

```
figure(5)
plot(t,p)
title('Supplied Power vs. Time');
xlabel('Time (seconds)');
ylabel('Power (Watts)');

figure(2)
plot(d,v)
title('Velocity vs. Distance');
xlabel('Distance (m)');
ylabel('Velocity (m/s)');

figure(3)
```

```
plot(d,ke);
title('Kinetic Energy vs. Distance');
xlabel('Distance (m)');
ylabel('Kinetic Energy (joules)');

figure(4)
plot(d,n);
title('Propulsion Efficiency vs. Distance');
xlabel('Distance (m)');
ylabel('Propulsion Efficiency');


%Phase 1
xR1 = 5.043;
xL1 = 0;
iL1 = knnsearch(t,xL1);
iR1 = knnsearch(t,xR1);
E_phase_1 = ej(iL1:iR1);

ep=1:length(E_phase_1);
for i=1:length(E_phase_1)-1;
   ep(i)=((E_phase_1(i)+E_phase_1(i+1))/2)*(t(i+1)-t(i));
end
y1=sum(ep);

%Phase 2
xR2 = 9.483;
xL2 = 5.043;
iL2 = knnsearch(t,xL2);
iR2 = knnsearch(t,xR2);
E_phase_2 = ej(iL2:iR2);

ep=1:length(E_phase_2);
for i=1:length(E_phase_2)-1;
   ep(i)=((E_phase_2(i)+E_phase_2(i+1))/2)*(t(iL2-1+i)-t(iL2-2+i));
end
y2=sum(ep);

%Phase 3
xR3 = 10.02;
xL3 = 9.9;
iL3 = knnsearch(t,xL3);
iR3 = knnsearch(t,xR3);
```

```
E_phase_3 = ej(iL3:iR3);

ep=1:length(E_phase_3);
for i=1:length(E_phase_3)-1;
   ep(i)=((E_phase_3(i)+E_phase_3(i+1))/2)*(t(iL3-1+i)-t(iL3-2+i));
end
y3=sum(ep);

%Phase 4
xR4 = 13.72;
xL4 = 10.02;
iL4 = knnsearch(t,xL4);
iR4 = knnsearch(t,xR4);
E_phase_4 = ej(iL4:iR4);

ep=1:length(E_phase_4);
for i=1:length(E_phase_4)-1;
   ep(i)=((E_phase_4(i)+E_phase_4(i+1))/2)*(t(iL4-1+i)-t(iL4-2+i));
end
y4=sum(ep);

%Phase 5
xR5 = 16.68;
xL5 = 13.72;
iL5 = knnsearch(t,xL5);
iR5 = knnsearch(t,xL5);
E_phase_5 = ej(iL5:iR5);

ep=1:length(E_phase_5);
for i=1:length(E_phase_5)-1;
   ep(i)=((E_phase_5(i)+E_phase_5(i+1))/2)*(t(iL5-1+i)-t(iL5-2+i));
end
y5=sum(ep);
```

Conclusion

During this lab, the team developed a program that would successfully and efficiently transport the AEV from the maintenance station, up to the Grand Canyon pick up station, and back down to the curve. After a couple tests, the errors in the program were fixed, and the trip became successful. The team found that the most efficient method of travel was to accelerate quickly to gain enough speed to get up the hill, while letting the momentum of the AEV push it further towards the end. Once the AEV reached the top, the direction was reversed, and a similar

process took place. By converting the data into tables and graphs, the team learned that the AEV was traveling too fast and using too much energy as it went down the hill.

Acknowledgments

The team would like to thank the Ohio State University, Their instructor, Professor Janiszewska, and the engineering department for giving the team the tools and resources necessary to complete this lab.

Noah Teal

Sample Calculations (in phase 2):

```
t(32)=te(32)./1000;
```

t(32) = 3173ms / 1000 = **3.173s**

```
v(32)=(15.*ve(32))./1024;
```

v(32) = (15*549 EEPROMv)/1024 = **8.0420v**

```
i(32)=(ie(32)./1024)*(2.46)*(1/.185);
```

i(32) = (52 EEPROMcurrent /1024)*(2.46v)*(1amp/.185v) = **0.6753 c**

```
d(32)=0.0124.*marks(32);
```

d(32) = 0.0124*15 marks = **0.1860 m**

```
s(32)=.0124.*pos(32);
```

s(32)= .0124 * 15 wheel counts = **0.1860 pos**

```
p(32)=v(32).*i(32);
```

p(32) = 8.0420v * 0.6753c = **5.4304 w**

```
ej(32)=((p(32)+p(33))/2)*(t(33)-t(32));
```

ej(32)= ((5.4304 w + 5.4304)/2)* (3.2750s-3.1730s) = **0.5539 j**

John Jeong

Sample Calculations:

*data in phase 7

```
t(140)=te(140)./1000;
```

t(140) = 14223ms / 1000 = **14.2233s**

```
v(140)=(15.*ve(140))./1024;
```

v(140) = (15*549 EEPROMv)/1024 = **7.9541v**

```
i(140)=(ie(140)./1024)*(2.46)*(1/.185);
```

i(140) = (82 EEPROMcurrent /1024)*(2.46v)*(1amp/.185v) = **1.0648 c**

```
d(140)=0.0124.*marks(140);
```

d(140) = 0.0124*394 marks = **4.8856 m**

```
s(140)=.0124.*pos(140);
```

s(140)= .0124 * 394 wheel counts = **4.8856 pos**

```
p(140)=v(140).*i(140);
```

p(140) = 7.9541v * 1.0648c = **8.4697 w**

```
ej(140)=((p(140)+p(141))/2)*(t(141)-t(140)));
```

ej(140)= ((8.4697 w + 8.2631 w)/2)* (14.3320s-14.3250s) = **0.8534 j**

Tim Regrut

ENGR 1181.02

Sample Calculations:

PHASE 1 CALCULATION

```
t(5)=te(5)./1000;
```

t(5) = 103ms / 1000 = **0.1030s**

```
v(5)=(15.*ve(5))./1024;
```

v(5) = (15*560 EEPROMv)/1024 = **8.2031v**

```
i(5)=(ie(5)./1024)*(2.46)*(1/.185);
```

i(5) = (0 EEPROMcurrent /1024)*(2.46v)*(1amp/.185v) = **0 c**

```
d(5)=0.0124.*marks(5);
```

d(5) = 0.0124* 0 marks = **0 m**

```
s(5)=.0124.*pos(5);
```

s(5)= .0124 * 0 wheel counts = **0 pos**

```
p(5)=v(5).*i(5);
```

p(5) = 8.2031v * 0 c = **0 w**

```
ej(5)=((p(5)+p(6))/2)*(t(6)-t(5)));
```

ej(5)= ((0.4246 w + 0.4246 w)/2)* (0.5120s-0.4100s) = **0.0433 j**