**Using Newton’s Method to Match Power Supply to Demand**

EMTH 171

Case Study 1

Logan Lee, [lpl29@uclive.ac.nz](mailto:lpl29@uclive.ac.nz)

Samuel Thornbury, [swt39@uclive.ac.nz](mailto:swt39@uclive.ac.nz)

20th September 2019

# **Exercise 1:**

## **Introduction:**

A sled is being hauled up an inclined slope (Figure 1). A cable attached to the sled is reeled in by a pulley system driven by an electric motor, through a gearbox.

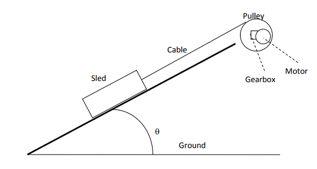


Figure 1: Sled haulage system

The power output of the electric motor ( *in kW).*

Where is the angular acceleration in , is a rotational constant in and is the angular velocity is . The motor gave a maximum power output just under at The gearbox of most basic AC electric motors spin much faster than required for a pulley system. Hence the output must be geared down to reel in the sled. These type of motors generally function most efficiently, and give their greatest output, at speeds between 1000-4000rpm.

The tangential speed of the winch drum ()

Where is the radius of the pulley in , is the electric motor’s angular velocity in and rgb is the gearbox ratio. is the speed in which the sled moves along the rail.

Power demand ( in order to push the sled up the slope.

Where is the tangential speed in . Gravity is = and with the coefficient of friction being , this must be overcome in order for the sled to start moving. The angle θ in of the slope affects both and .

=

When  and are equal the sled will be moving at a constant velocity.

Rearranged equation 4. with system parameters

This meant that the only unknown value was the velocity of the sled. When set to 0, and are equal meaning the system is in equilibrium. Hence why the equation can be solved to find the velocity of the sled.

This type of engineering problem can be solved using Newton’s Method, iterating from an initial guess for a root of a function (equation 5), to get an improved approximated value for the roots of the function.

## **Problem Analysis:**

Newton’s Method is an algorithm that approximates the roots of a given function. The method is iterative therefore produces more accurate approximations the more iterations performed. The method uses the formula

MATLAB was used to apply Newton’s Method on sled haulage problem, as it is important to have an initial guess close to the actual roots an initial guess value, given as . The values the algorithm produces are the intersection of the x-axis and the tangent of the previous value. Newton’s Method breaks once the values are within a given tolerance, which in this case was , thus meaning for the iterations to stop, the value calculated before must be within of the value calculated before. The code used to perform the method are in the appendix.

Figure 2 shows the sled power demanded and motor power output versus sled velocity. The points at which the Motor Power and Power Demand lines overlap in the graph is where they are equal, these points gave us the two roots possible by Newton’s method, which depends on the chosen initial value. When the initial guess is too low, Newton’s method will produce an approximation velocity of 0 .

Figure 3 shows the plot of velocity after each iteration as it converges towards the final value. Starting from an initial guess of 2.0 the final velocity of 2.6514 was reached after the 4th iteration. The 5th Iteration was carried out, however, it was within the given tolerance, meaning the process finished.

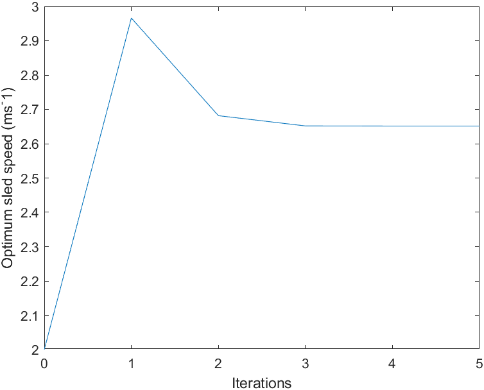


Figure 3: Plot of velocity after each iteration as it converges towards the final value

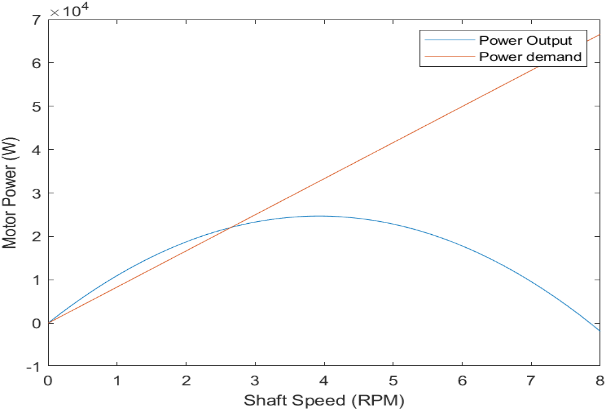


Figure 2: Sled power demanded and motor power output versus sled velocity

## **Conclusion:**

When is equal to the sled will be travelling at a constant velocity. Newton’s method was used with an initial guess of and tolerance of to approximate the convergence point. The algorithm took 4 iterations to be within tolerance and to calculate the correct value of . In this study two roots were found, the first root being 0 when the sled was stationary, the second root was when the sled was moving at a constant velocity, the function produced an answer of 2.. Hence Newton’s method performed as anticipated.

# **Exercise 2:**

## **Introduction:**

Exercise 2 considers a medium-sized family car with a petrol engine, the maximum velocity was found over various different slope angles and accelerations.

Power output in

Where () and are engine constants. V is the velocity of the car in , R is the radius of the wheel in metres. is the gearbox ratio and, is the final drive ratio.

Resistive forces are acting on the car, the quicker the car moves the greater the amount of power (watts) required to overcome these friction forces. The next four equations are the four main resistive forces.

Air resistance( )

Where is the drag coefficient, is the frontal area in , is the air density in and is the velocity in .

Rolling resistance ()

where is the rolling resistance coefficient is the gravitational acceleration in , is the mass of the car in is the slope of the road in and is the velocity of the car in .

## Gradient climbing ()

Where m is the mass of the car in is the gravitational acceleration in , is the slope of the road in and is the velocity in .

Rolling resistance ()

Where is the mass of the car in is the car’s acceleration in , and is the velocity in .

Equations 7 to 11 can be put into equation 12 to give function 13. Therefore the only unknown will be the car’s velocity. When function 13 is set to 0, the system is in equilibrium, so and become equal. Hence solving the equation will give the velocity of the car. Newton’s method was used once again to find the roots of equation 13.

- = 0

## **Problem Analysis:**

Looking at the graph shown in figure 4, it tends to represent a parabolic curve. On small engines used in cars, the curve is close to a quadratic. The graph shows that the parabolic path peaks at a power output of 100,000 W and an engine speed of about 450 rad/s. Engines are designed to have a certain stage in their cycle where they create their peak power, this is affected by many factors. Including cam profiling (intake and exhaust with valve timing), the engine control unit(spark and fuel injection timing), mass of air and fuel injected in each intake stroke, including the ignition time. When all these elements are functioning at their highest potential and in sync, the engine creates its maximum power. The engine is at its most efficient point when it is accelerating. Once the engine has passed its peak power output, the more resistive forces affect the motion of the car. This results in more power required to overcome these forces, causing the engines overall power output to decrease.

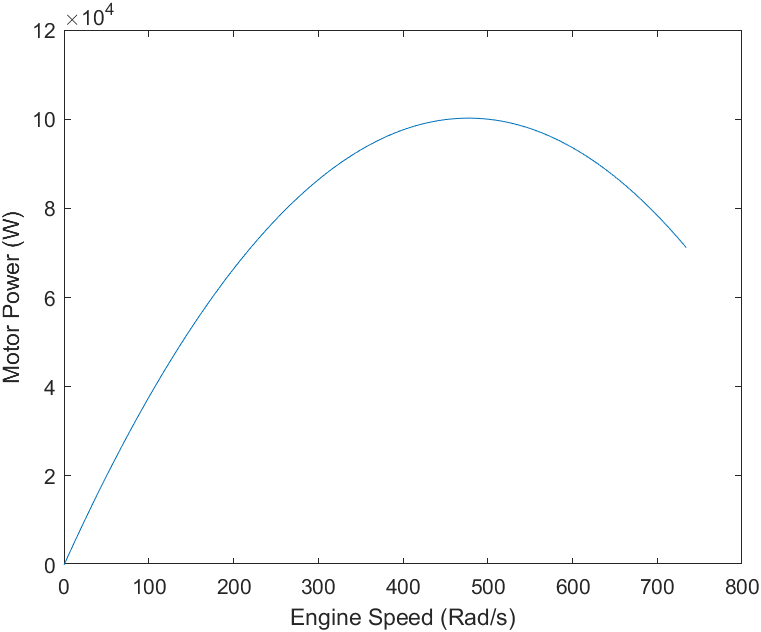


Figure 4: Power output of a petrol engine as function of speed

## **Case 1:**

Case one consisted of a incline and acceleration of . Newton’s Methods was used with an initial guess of and after 2 iterations the optimum velocity was calculated to be . The optimum velocity is where the engine output power converges with the road speed as shown in figure 5. As mentioned above it is important to have an initial guess that is reasonably close to a value of a root, this is because if the value is too low or too high Newton’s Method will approximate the wrong root. For example in figure 6 and 7, the approximation heads to the correct value of while in figure 8 since the initial guess is too low the approximated root is , this being MATLAB’s approximation of zero.

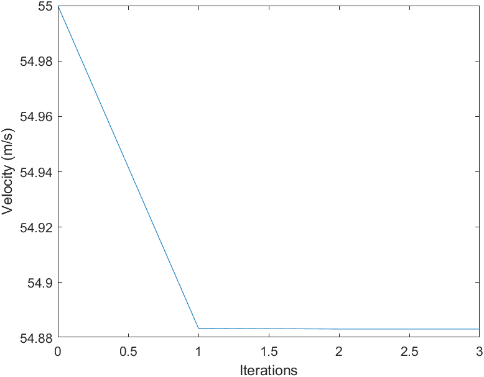


Figure 6: Newton's Method, Car velocity versus number of iterations with inital guess at 55 for case 1

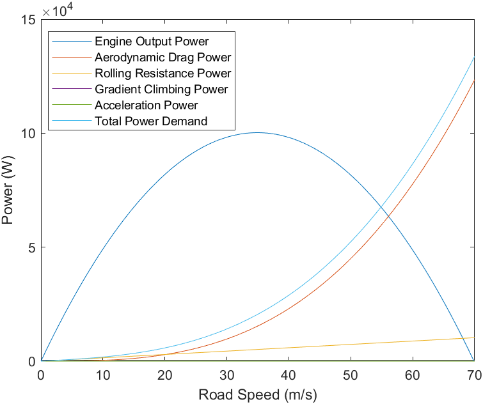


Figure 5: Engine output and power demands versus road speed for case 1

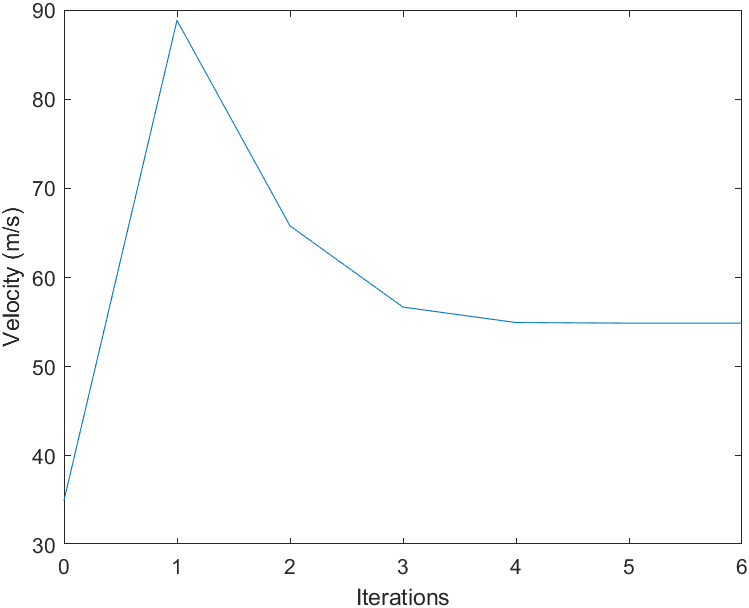


Figure 7: Newton's Method, Car velocity versus number of iterations with inital guess at 35 for case 1

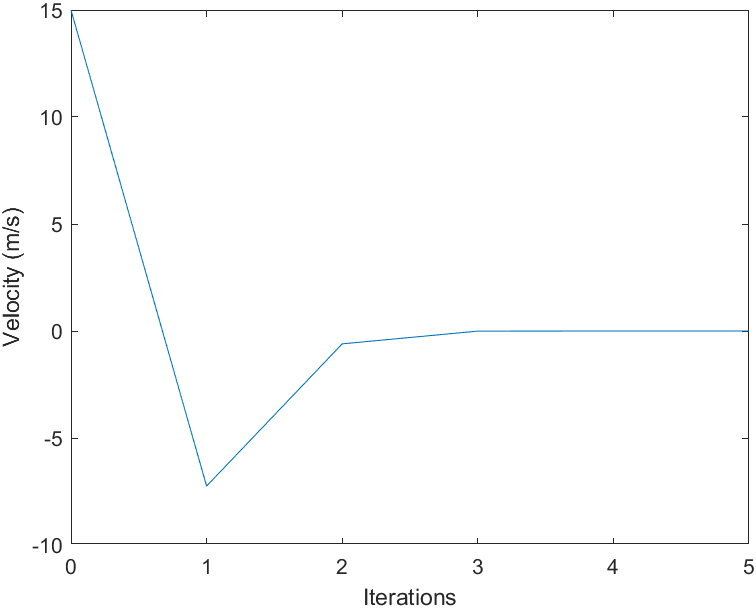


Figure 8: Newton's Method, Car velocity versus number of iterations with inital guess at 15 for case 1

## **Case 2:**

Case 2 consisted of a incline with no acceleration. Starting with an initial guess of Newton’s Method approximated the optimum velocity to be after 4 iterations as shown in figure 9 and 10. The optimum velocity is lower than in case 1 as the car is going up an incline thus working against the forces of gravity in addition to the rest of the resistive forces.

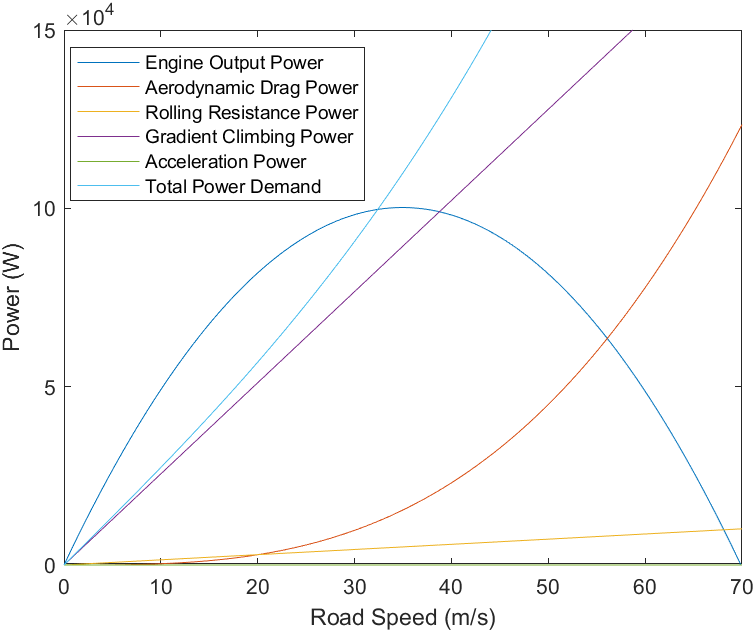


Figure 9: Engine output and power demands versus road speed for case 2

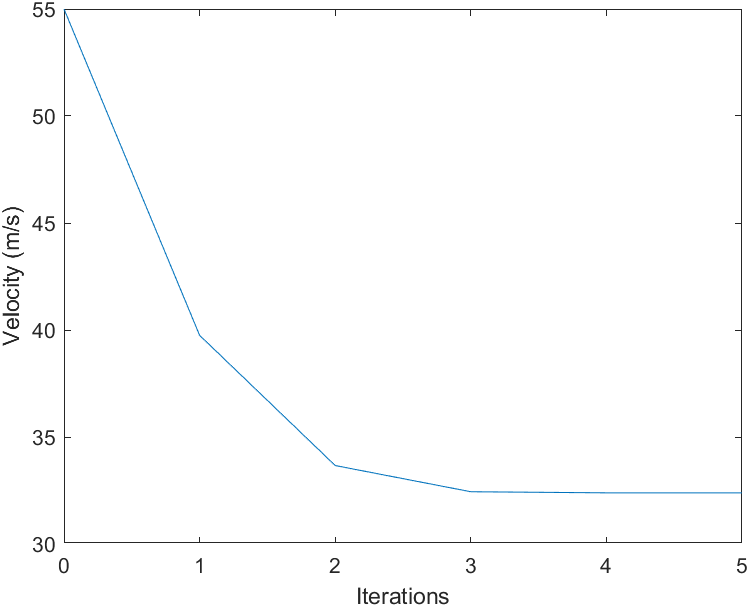


Figure 10: Newton's Method, Car velocity versus number of iterations with inital guess at 55 for case 2

## **Case 3:**

Case 3 consisted of a decline and no acceleration. Again with the initial guess being , Newton’s Method approximated the optimum car velocity to be after 4 iterations as shown in figure 11. As the car is going down a decline the optimum speed has increased as the force of gravity is aiding the car rather than resisting it, therefore, as seen in figure 12 gradient climbing power is negative and decreasing.

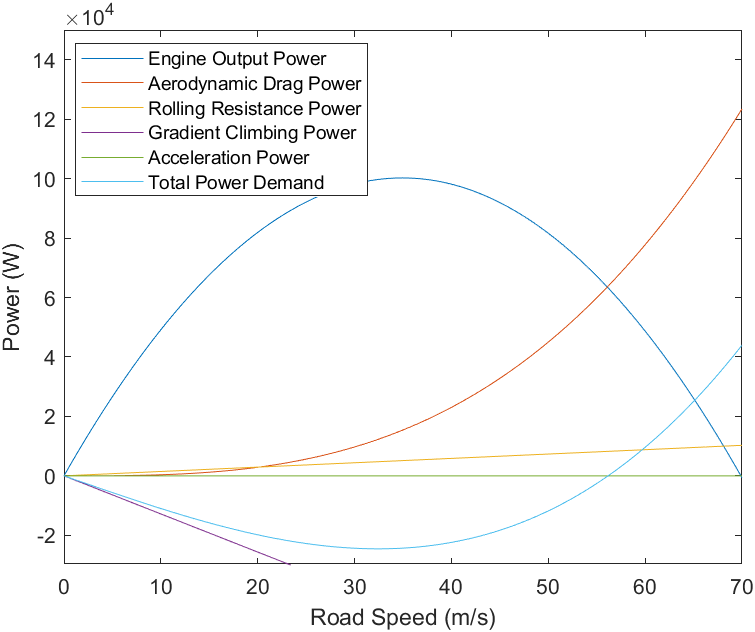


Figure 11: Engine output and power demands versus road speed for case 3

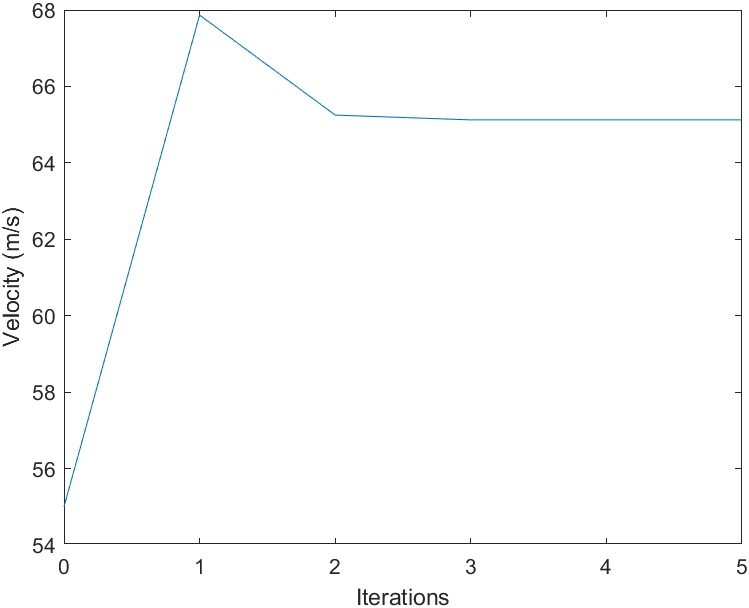


Figure 12: Newton's Method, Car velocity versus number of iterations with inital guess at 55 for case 3

## **Case 4:**

Case 4 consisted of a incline and a acceleration. The initial guess was and the algorithm calculated the optimum car velocity to be after 4 iterations as shown in figure 13. Since the car has a constant acceleration of the total power demand is significantly more than a car travelling at a constant speed thus as shown in figure 14 the optimum speed is much lower than in case 1.

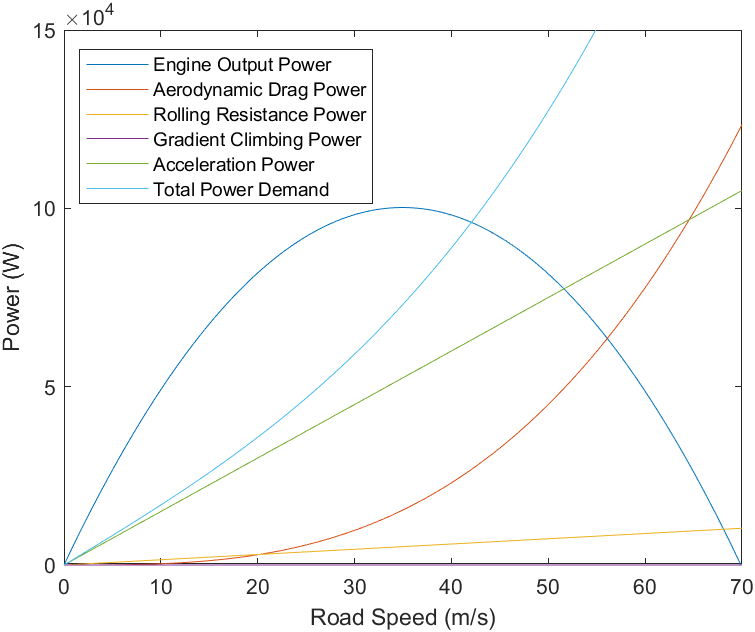


Figure 14: Engine output and power demands versus road speed for case 4

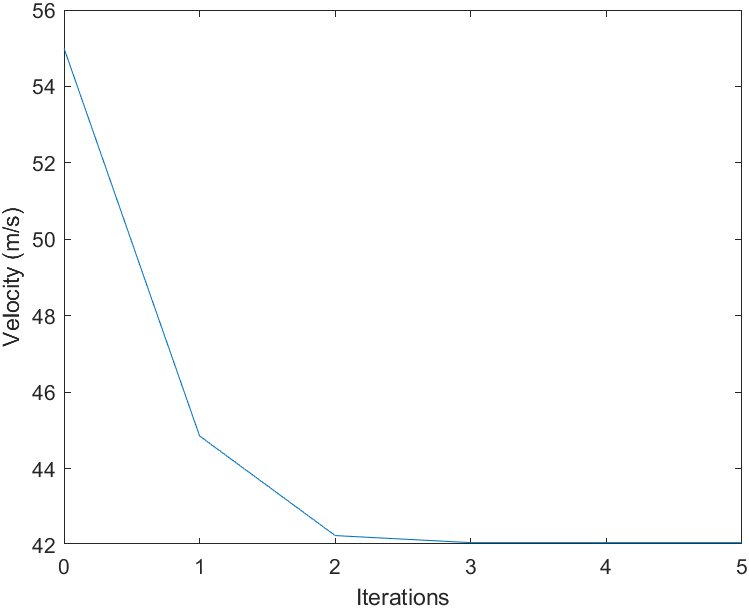


Figure 13: Newton's Method, Car velocity versus number of iterations with inital guess at 55 for case 4

## **Conclusion:**

After analysing the different conditions that the car was put under, including different slope angles and acceleration. Each case produced a different value, Case 1: , Case 2: , Case 3: , Case 4: . The intitial guess for each case was important in finding the correct root, if a poor initial guess was made (one that was closer to zero than to the final value), then Newton’s method would return an approximated value of zero. A reasonable initial guess being relatively close to the final value would give the correct root and would decrease the number of iterations needed to find the root. Giving 2 solutions to the problem, zero and the optimum car velocity number. This case study has shown how putting a car under different conditions changes the velocity of the vehicle. The lowest velocity was found at the speed of , for case 2, incline with no acceleration. This was expected as this was the steepest upwards slope out of all the cases, meaning it had the greatest gravity resitive force. The greatest velocity was a speed of for case 3, consisting of a decline and no acceleration. This was because the car was moving down a shallow slope meaning force due to gravity was causing the car to go faster. Case 4’s velocity was which is less than case 1’s velocity of , this is due to the car’s instantaneous acceleration of 1 (case 4), causing the engine to reach its maximum power output/velocity sooner. Hence why case 4’s maximum velocity is less than that of case 1.

# **Appendix:**

**Exercise 1:**  
% EMTH 171, Case Study 1

% Exercise 1, Sled haulage problem

% Use Newton's methods to solve for "V"

% Logan Lee, Samuel Thornbury

clear

clc

close all

% Global Variables---------------------------------------------------------

alpha = 1; % constant of multiplication (W.S^2.rads^2)

beta = 100 \* pi; % constant of multiplication (rads^-1)

rGB = 20; % Gear ratio

R = 0.5; % Radius of drum(m)

m = 1000; % Total mass of sled (kg)

g = 9.81; % Gravity (ms^-2)

theta = pi/4; % Angle of ramp (degrees)

cF = 0.2; % Coefficient of friction

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Motor Power Ouput and Shaft Speed Plot

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%Plotting Preparation-----------------------------------------------------

P\_out = @(O) alpha\*(beta\*O - O^2); % Power output

P\_demand = @(v) (m\*g\*sin(theta)\*v) + (cF\*m\*g\*cos(theta)\*v); % Power demand

sledSpeed = [0:0.1:8];

P\_outArray = [];

P\_demandArray = [];

for ii = sledSpeed

P\_outArray = [P\_outArray, P\_out((ii \* rGB) / R)];

P\_demandArray = [P\_demandArray, P\_demand(ii)];

end

% Motor Output / Shaft Speed Plotting-------------------------------------

figure(1)

plot(sledSpeed, P\_outArray, sledSpeed, P\_demandArray);

%title("Motor Output Power Versus Shaft Speed");

legend("Power Output", "Power demand")

xlabel("Shaft Speed (RPM)");

ylabel("Motor Power (W)");

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Newton's Method

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Newtons Method Variables-------------------------------------------------

V = 2; % Initial Guess of velocity

N = 20; % Max number of iterations

tol = 0.0001; % Tolerance of V

resultArray = V; % begins the array of results with the initial guess

iterations = 0; % to calculate number of iterations performed in for loop

% Functions----------------------------------------------------------------

f = @(V) (((m\*g)\*sin(theta)\*V) + ((cF\*(m\*g))\*cos(theta)\*V)...

- alpha\*(beta\*((rGB\*V)/R) - (((rGB^2) \* V^2)/R^2)));

% Derviavtive of the above function

d = @(V) (((m\*g)\*sin(theta)) + ((cF\*(m\*g))\*cos(theta))...

- alpha\*(beta\*(rGB/R) - (((2\*(rGB^2))\*V)/R^2)));

% Newton's method for-loop-------------------------------------------------

for n = 1 : N

iterations = iterations + 1;

oldV = V;

V = (V - (f(V) ./ d(V)));

resultArray = [resultArray; V];

if (abs(f(V)) < tol) && (abs(oldV - V) < tol)

break

end

end

iterationsArray = [0: iterations];

%Plotting the results------------------------------------------------------

figure(2)

plot(iterationsArray, resultArray)

xlabel("Iterations")

ylabel("Optimum sled speed (ms^-1)")

**Exercise 2, Case 3:**  
% EMTH 171, Case Study 1

% Exercise 2, Power and speed of a car

% Use Newton's methods to solve for "V"

% Logan Lee and Samuel Thornbury

clear

clc

close all

% Global Variables---------------------------------------------------------

m = 1500; % mass of vehicle (kg)

cD = 0.30; % drag coefficient

A = 2.0; % frontal area (m^2)

CRR = 0.010; % Coefficient depending on wheel and ground properties

R = 0.205; % wheel radius (m)

alpha = 420; % (W.s / rad)

beta = 0.440; % (W.s^2 / rad^2)

dR = 3.5; % final drive ratio

rGB = 0.80; % gearbox ratio

g = 9.81; % gravity (m/s^2)

p = 1.2; % air density (kg / m^3)

K = R/(dR\*rGB);

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Motor Power Ouput and Shaft Speed Plot

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Plotting Preparation-----------------------------------------------------

P\_out = @(O) alpha\*O - beta\*O^2; % Power output

E\_speed = [0 : 734]; % Engine Speed

P\_outArray = [];

E\_speedArray = [];

for ii = [0 : 734]

P\_outArray = [P\_outArray, P\_out(ii)];

end

% Motor Output / Shaft Speed Plotting-------------------------------------

figure(1)

plot( E\_speed, P\_outArray);

%title("Power output of a petrol engine as a function of speed");

xlabel("Shaft Speed (Rad/s)");

ylabel("Motor Power (W)");

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Newton's Method

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Newton's Method Variables------------------------------------------------

V = 55; % Initial Guess of velocity

N = 20; % Max number of iterations

tol = 0.0001; % Tolerance of V

resultArray = V; % begins the array of results with the initial guess

iterations = 0; % to calculate number of iterations performed in for loop

% Case Variables-----------------------------------------------------------

a = 0; % Acceleration

theta = (-5 \* pi)/180; % Angle of gradient

% Functions----------------------------------------------------------------

f = @(V) cD\*A\*0.5\*p\*V^3 + m\*g\*sin(theta)\*V + CRR\*m\*g\*cos(theta)\*V...

+ m\*a\*V - (alpha\*V/K - beta\*V^2/K^2);

% Derivative of the above function

d = @(V) 1.5\*cD\*A\*p\*V^2 + m\*g\*sin(theta) + CRR\*m\*g\*cos(theta) + m\*a...

- (alpha/K - 2\*beta\*V/K^2);

% Newton's method for-loop-------------------------------------------------

for n = 1 : N

iterations = iterations + 1;

oldV = V;

V = (V - (f(V) ./ d(V)));

resultArray = [resultArray; V];

if (abs(f(V)) < tol) && (abs(oldV - V) < tol)

break

end

end

iterationsArray = [0: iterations];

% Plotting Newton's Method-------------------------------------------------

figure(2)

plot(iterationsArray, resultArray)

xlabel("Iterations")

ylabel("Velocity (m/s)")

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Case 3

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% Case Variables (as stated above)-----------------------------------------

a = 0; % Acceleration

theta = (-5 \* pi)/180; % Angle of gradient

% Plotting Preparation for Case--------------------------------------------

% Functions:

Peout = @(v) (alpha \* v)/K - (beta\*v^2)/K^2; % Power output

Pd = @(V) cD\*A\*0.5\*p\*V^3; % Power dissipated by aerodynamic drag

Prr = @(v) CRR\*m\*g\*cos(theta)\*v; % Power dissaipated by rolling resistance

Pgc = @(v) m\*g\*sin(theta)\*v; % Power consumed by climbing a gradient

Pa = @(v) m\*a\*v; % Power required to accelerate

Ptot = @(v) cD\*A\*0.5\*p\*v^3 + m\*g\*sin(theta)\*v + CRR\*m\*g\*cos(theta)\*v ...

+ m\*a\*v; % Total power demand

% Arrays

PeoutArray = [];

PdArray = [];

PrrArray = [];

PgcArray = [];

PaArray = [];

PtotArray = [];

xArray = [0 : 0.1 : 70];

for ii = [0 : 0.1 : 70]

PeoutArray = [PeoutArray, Peout(ii)];

PdArray = [PdArray, Pd(ii)];

PrrArray = [PrrArray, Prr(ii)];

PgcArray = [PgcArray, Pgc(ii)];

PaArray = [PaArray, Pa(ii)];

PtotArray = [PtotArray, Ptot(ii)];

end

% Plotting the results-----------------------------------------------------

figure(3)

plot(xArray,PeoutArray, xArray,PdArray, xArray, PrrArray, xArray, PgcArray, ...

xArray, PaArray, xArray, PtotArray);

%title("Power Demand Versus Engine Output Power")

legend("Engine Output Power", "Aerodynamic Drag Power", ...

"Rolling Resistance Power", "Gradient Climbing Power", ...

"Acceleration Power", "Total Power Demand")

xlabel("Road Speed (m/s)")

ylabel("Power (W)")

ylim([-3e4, 15e4]);

V % Display V