EMTH171 Case Study 2

Electric Car Modeling Using Euler’s Method

|  |  |
| --- | --- |
| **Name 1: Azim Rushdi - 55522374** | **Email:** [**aar91@uclive.ac.nz**](mailto:aar91@uclive.ac.nz) |
| **Name 2: Jaime Sequeira - 42408823** | **Email:** [**jfs73@uclive.ac.nz**](mailto:jfs73@uclive.ac.nz) |

**Introduction**

Euler’s method is a great technique to estimate the solution of an ordinary differential equation. The method uses an iterative process that converges the variable when approaching the root of a function. From a differential equation such as *y’ = f(t,y),* Euler’s method can be implemented iteratively to solve it, and the result is expressed in the following equation:

|  |  |  |
| --- | --- | --- |
|  | *yi+1 =yi + f(ti ,yi)h* | **(1)** |

In the first part of this report, the distance that an electric car travels in the first 100 seconds is determined from a given equation of the angular acceleration of the motor of the car, . The differential equation is solved using the usual integration method and Euler’s method, and the results are compared.

is manipulated for the later part of the tasks to determine the current consumption of the electric car and its performance under different circumstances.

**Problem Analysis**

Task 0

In the first part of task 0, simple integration is required to determine the distance (s) that the electric car travels in the first 100 seconds. The angular acceleration of the motor of the car, is expressed in Equation 1 below:

|  |  |  |
| --- | --- | --- |
|  | = | **(2)** |

Where *Je*f is the effective moment of inertia of the car (*kgm2*). *Tm* is the torque of the motor . and are the angular speeds of the wheels and motor respectively. is the radius of the wheels of the car. is the low velocity drag force . is the wind resistance acting against the movement of the car where is the wind resistance drag coefficient and u is the velocity of the car . For this exercise, the wind resistance was ignored. Therefore

is the load force acting on the car due to gravity where M is the mass of the car (kg), g is the acceleration of the car vertically due to gravity and is the angle of the incline the car is on .

in equation above is the sprocket ratio which determines the value of *Tm.* This ratio can be rewritten as in equation (**3**):

|  |  |  |
| --- | --- | --- |
|  | = | **(3)** |

Where r1 is the radius of the motor sprocket (m) and r2 is the radius of the wheel sprocket (m).

Substituting equation (**3**) into equation (**2**) gives

|  |  |  |
| --- | --- | --- |
|  | = | **(4)** |

Integrating will give the angular velocity of the motor () in rad/s. The angular velocity can then be integrated to give the distance traveled by the car (rad) within the time range specified. This produces the following equation and the distance is converted from radians to meters:

|  |  |  |
| --- | --- | --- |
|  |  | **(5)** |

Task 1

This task required finding the current that would be required to be supplied to the motor in order keep the car travelling at a constant velocity of 100km/h taking wind resistance into account. This is achieved when the acceleration of the car is zero i.e.. Therefore, equation 4 can be manipulated to find the current required.

|  |  |  |
| --- | --- | --- |
|  |  | **(6)** |

This can be rearranged to form an equation where the current is the dependent variable after the motor torque is broken down into its components; current and the coefficient of the motor .

|  |  |  |
| --- | --- | --- |
|  |  | **(7)** |

In order to calculate the percentage of the battery remaining after traveling a certain time, the charge used need to be calculated.

|  |  |  |
| --- | --- | --- |
|  |  | **(8)** |

where Q is the charge used in coulombs, I is the current used in Amps and t is the time in seconds. To calculate the percentage used, the following formula is used:

|  |  |  |
| --- | --- | --- |
|  |  | **(9)** |

where is the total charge of the battery.

Calculating the remaining charge percentage is a straightforward calculation as follows:

|  |  |  |
| --- | --- | --- |
|  |  | **(10)** |

Task 2

Due to the tasks before requiring calculation of the acceleration, distance and velocity of the car each time, a function was created to avoid unnecessary repetition of code. The parameters of the function are current acceleration, current velocity, step size, current distance, angle of incline and coefficient of drag. This function’s outputs were the new acceleration, new velocity and new time as this needs to be updated in each iteration. These new values were then used for the next iteration in the function.

Task 2 required adapting this function so that the only parameters to the new function were the angle of the incline and the wind resistance coefficient. The output of the function was the distance traveled.

The percentage improvement after these changes was calculated with the formula below:

|  |  |  |
| --- | --- | --- |
|  |  | **(11)** |

**Results and Discussion**

Task 0

In this task, the car is accelerated for 100 seconds with a current of 100 Amps and then the motor current is turned off. The car then coasts to a stop. The distance traveled in the first 100 seconds was calculated by hand to be 529.249925135 meters which was rounded to 529.2499 m. This was done by integrating the acceleration of the electric car twice with respect to time as stated in equation (**5**).

Equation (**4**) was integrated to calculate the angular speed of the motor of the car using Euler’s method. The angular speed of the motor was then converted into the linear speed of the car by rearrangement of equation (**3**):

=

Using the obtained , the distance traveled by the car at this point was measured using trapezoidal integration.

The distance that the electric car travels in the first 100 seconds obtained using this method was 529.2499 m with a step size of one second. The total distance that the car travel until it came to a stop after the motor was turned off was 1505.7 m.

Using a similar method as in part b, the distance traveled by the electric car until it stopped when taking wind resistance into account was 1278.3 m with a step size of one second. This value was 227.47 m lower than the distance obtained when the wind resistance was neglected. This occurred because the wind resistance is proportional to the square of velocity of the car. As the car sped up, there was more wind resistance acting on it and therefore the car traveled at a much lower distance over the same range of time.

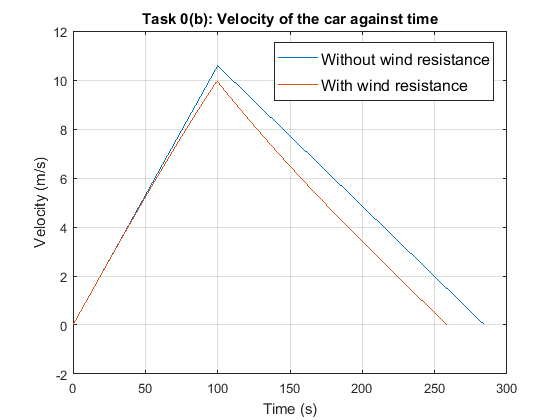
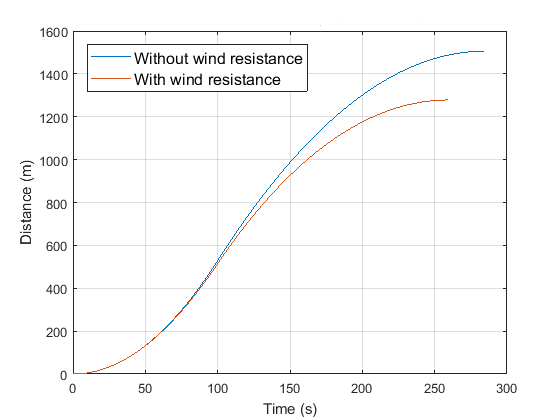


Figure 2: Distance of car against time with and without wind resistance.

Figure 1: Velocity of car against time with and without wind resistance.

The step size used in all these situations was one second as this produced a smooth curve. It gave a more accurate result than step sizes such as 50 and 10 without sacrificing compute timing. The table below compares the calculated distances traveled until the car stops using Euler’s method with different step sizes.

Table 1: Comparing step size with calculated distance with and without wind resistance

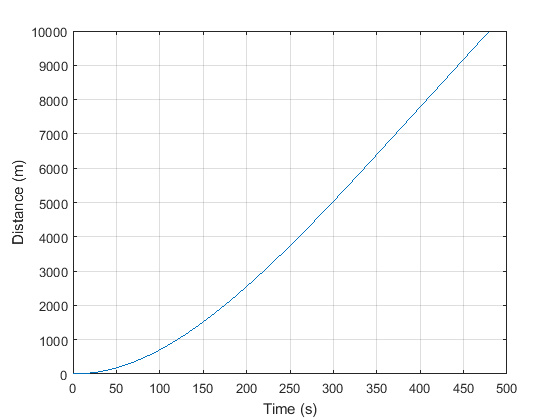
|  |  |  |
| --- | --- | --- |
| Step Size (s) | Total distance (No wind) (m) | Total distance (Wind) (m) |
| 50 | 1498.8 | 1234.9 |
| 10 | 1504.9 | 1282.1 |
| 1 | 1505.7 | 1278.3 |
| 0.1 | 1484.7 | 1258.2 |
| 0.01 | 1476.1 | 1256.1 |

As the step sizes decrease, the accuracy of the calculation improves as it gets closer to the true value of the distance. Euler’s method is only an estimate of an integral and is therefore not one hundred percent accurate.

Task 1

In this task, the car was accelerated with a constant current supply of 100 Amps until it reached 100 km/h. Once the speed was reached, the current supply was readjusted to keep the car travelling at this speed constantly until the car had traveled 10 kilometers. Using equation (**7**), the current required to keep the car at a constant velocity of 100km/h was calculated to be 114.8818 Amps.

At this velocity, the acceleration from the motor is equal to the deceleration acting on the car caused by drag, wheel rolling resistance and other forces.

Euler’s method was used to find the time required to travel 10km. The time required was calculated to be 480s. The percentage of the battery that remains at this point was calculated using equation (**10**) and the result was 91.4174%.

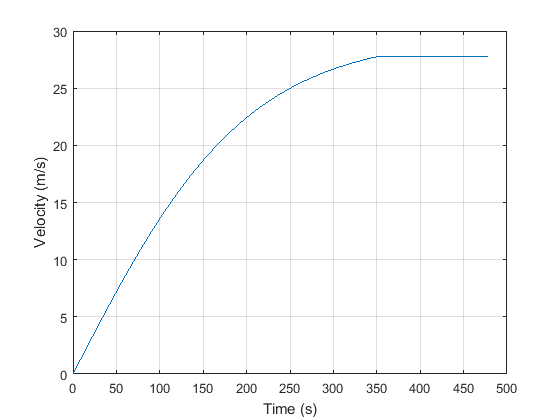


Figure 3: Distance of the car against time with custom method of control.

Figure 4: Velocity of the car against time using custom method of control.

Task 2

This task required designing a function that calculated the total distance traveled by a car. However, this function also had to handle changes to the angle of the incline and changes to the drag coefficient. Therefore, the parameters for the function were (angle of incline) and the coefficient of drag.

The function created was named *distanceFunction*. This function was designed to simulate the car accelerating with a current of 650 Amps for thirty seconds. It then travels at the speed it reaches constantly until the battery is completely empty.

Two for-loops were used with this function to compare the distances traveled with and without changes to the wind resistance coefficient and changes in the incline. The result of the each scenario is displayed in Figure (**5**) below.

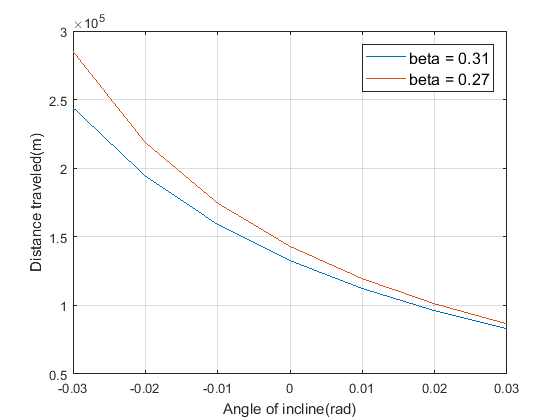


Figure 5: Distance of the car against angle of incline with and without the new beta co-efficient.

From this figure, it is clear that the total distance the electric car travel decreases as the angle of incline, increases. This is because as gets bigger, the car will have to overcome more drag thus consuming more energy as it moves.

By lowering the wind resistance and by using equation (**11**), the car was calculated to have improved in range performance by a percentage of 23.1128 %.

By lowering the wind resistance, there is less force acting on the car and therefore a smaller deceleration acting against the acceleration of the motor. By decreasing the angle of incline, the car has to use less power to accelerate up the incline and therefore less power is used and the car travels a longer distance.

**Conclusions**

The aim of this case study was to use Euler’s method to find the distances traveled by an electric car under different conditions.

In Task 0, the electric car traveled a distance of 529.2499m without wind resistance and 514.4902m with wind resistance in one hundred seconds. The car also traveled a distance of 1505.7m without wind resistance and 1278.3m with wind resistance before coming to a stop after the motor was switched off. These were calculated with Euler’s method and a step size of one second. By reducing the wind resistance, the car is able to travel a longer distance as there is less deceleration acting against it by the wind.

In the next task, it was found using Euler’s method that the car required 114.8818 Amps to drive at a constant velocity of 100km/h. The time taken for the car to travel a distance of 10km was 480s with 91.4174% of the battery charge remaining after the journey.

The last task required creating a function called *distanceFunction* which calculated the total distance traveled with changes to the angle of incline and wind resistance coefficient. The mean percentage range performance improvement was calculated to be 23.1128%.

Appendix

function [ omega,origVelocity,origDisplacement ] = calculateDataStep(omega,origVelocity,origDisplacement,tdelta,beta,current)

%calculateDataStep: This function is what we use to save retyping the same code over and over again

% It calculates the new velocities and outputs them to be used in the next

% iteration

% Initialise variables

g = 9.81; % Gravitational constant m/sec/sec

% Car constants

m = 760; % Mass in kg

Jmotor = 25.5; % Motor moment of intertia in kg.m^2

Jwheel = 10.0; % Drive wheel moment of intertia in kg.m^2

fd0 = 180; % Static (& low speed) drag force in N

km = 0.5;

rmotorS = 0.1; rwheelS = 0.15; rwheel = 0.25; % Radii in m

ratios = rmotorS / rwheelS \* rwheel; %Converts rad/s to mp/s

% Parameters of the road fof task 0 and 1

sina = sin (atan (-10 / 1000));

% Car parameterrs

Tm = current \* km; % Fixed motor torque in N.m while current is applied

Jef = Jmotor + (rmotorS / rwheelS)^2 \* (Jwheel + rwheel^2 \* m); % Effective moment of intertia in kg.m^2

fload = fd0 + m \* g \* sina;

% The calculations that are iteratively done each loop

dwdt = 1 / Jef \* (Tm - (ratios \* (fload + (beta \* origVelocity ^ 2))));

omegaNew = omega + tdelta \* dwdt;

velocityNew = omegaNew \* ratios;

displacementNew = origDisplacement + tdelta \* (origVelocity + velocityNew)/2;

omega = omegaNew;

origVelocity = velocityNew;

origDisplacement = displacementNew;

end

% EMTH171 Case study 2 Task 0 and 1

% This script uses calculateDataStep function to find distances traveled by car

clear,clc

%Initial conditions (standing start)

index = 1;

omega = 0; % Initial omega\_m ('om')

origVelocity = 0; % Speed along road in m/sec

origDisplacement = 0; % Distance along road from start in m

tdelta = 1; % Step size for Euler's method

tf = 100; % Time when the current is turned off

% Start task 0(b)

for t = 1:tdelta:tf

beta = 0; current = 100; % Set wind resistance and current

[omega,origVelocity,origDisplacement] = calculateDataStep(omega,origVelocity,origDisplacement,tdelta,beta,current);

vLinear(index) = origVelocity;

distanceArray(index) = origDisplacement;

index = index + 1;

end

%Continues the calculation until the car stops

while (origVelocity > 0)

current = 0; beta = 0;

[omega,origVelocity,origDisplacement] = calculateDataStep(omega,origVelocity,origDisplacement,tdelta,beta,current);

vLinear(index) = origVelocity;

distanceArray(index) = origDisplacement;

index = index + 1;

end

%----------Plotting-------%

figure(1)

plot ([0:tdelta:tdelta \* (index - 2)], vLinear)

title('Task 0 (b): Velocity of car(m/s) against time(s)')

xlabel('Time (s)'), ylabel('Velocity(m/s)')

grid on

figure(2)

plot ([0:tdelta:tdelta \* (index - 2)], distanceArray)

title('Task 0 (b): Distance traveled by car(m) against time(s)')

xlabel('Time(s)'), ylabel('Distance traveled(m)')

grid on

% Task 0(c) and reset all iterated values

omega = 0;

origVelocity = 0;

origDisplacement = 0;

index = 1;

beta = 0.31; % Co-efficient of drag

current = 100;

for t = 1:tdelta:tf

[omega,origVelocity,origDisplacement] = calculateDataStep(omega,origVelocity,origDisplacement,tdelta,beta,current);

vLinearWind(index) = origVelocity;

distanceArrayWind(index) = origDisplacement;

index = index + 1;

end

current = 0; % Current is switched off

while (origVelocity > 0)

[omega,origVelocity,origDisplacement] = calculateDataStep(omega,origVelocity,origDisplacement,tdelta,beta,current);

vLinearWind(index) = origVelocity;

distanceArrayWind(index) = origDisplacement;

index = index + 1;

end

%-----------Plotting---------%

figure(3)

plot ([0:tdelta:tdelta \* (index - 2)], vLinearWind)

title('Task 0 (c): Velocity of car in m/s')

xlabel('Time(s)'), ylabel('Velocity(m/s)')

grid on

figure(4)

plot ([0:tdelta:tdelta \* (index - 2)], distanceArrayWind)

title('Task 0 (c): Distance traveled in m')

xlabel('Time(s)'), ylabel('Distance traveled by car(m)')

grid on

% Start task 1 and reinitialise iterated values

omega = 0;

origVelocity = 0;

origDisplacement = 0;

index = 1;

beta = 0.31;

while origVelocity < (100 \* 1000 / 3600)

current = 100;

[omega,origVelocity,origDisplacement] = calculateDataStep(omega,origVelocity,origDisplacement,tdelta,beta,current);

vLinearWindNew(index) = origVelocity;

distanceArrayWindNew(index) = origDisplacement;

index = index + 1;

end

current = 114.8818; % Current required to maintain current origVelocity

while origDisplacement < 100

[omega,origVelocity,origDisplacement] = calculateDataStep(omega,origVelocity,origDisplacement,tdelta,beta,current);

vLinearWindNew(index) = origVelocity;

distanceArrayWindNew(index) = origDisplacement;

index = index + 1;

end

%----------Plotting-----------%

figure(5)

plot ([0:tdelta:tdelta \* (index - 2)], vLinearWindNew)

title('Task 0 (c): Velocity of car in m/s')

xlabel('Time(s)'), ylabel('Velocity(m/s)')

grid on

figure(6)

plot ([0:tdelta:tdelta \* (index - 2)], distanceArrayWindNew)

title('Task 0 (c): Distance traveled in m')

xlabel('Time(s)'), ylabel('Distance traveled by car(m)')

grid on

function [ origDisplacement ] = distanceFunction(beta, alpha)

%origDisplacement: This function takes a drag co-efficient and angle and

%outputs the total distance traveled by the car. Car details are given already

% Initialise variables

g = 9.81; % Gravitational constant m/sec/sec

% Car constants

m = 760; % Mass in kg

Jmotor = 25.5; % Motor moment of intertia in kg.m^2

Jwheel = 10.0; % Drive wheel moment of intertia in kg.m^2

fd0 = 180; % Static (& low speed) drag force in N

km = 0.5;

rmotorS = 0.1; rwheelS = 0.15; rwheel = 0.25; % Radii in m

% Parameters of the road for Task 0 and Simulation parameters

current = 650; % Fixed motor current in A

sina = sin (alpha); % sin(alpha) for 10 m down every km along

tdelta = 1.0; % Time step in seconds

totCap = 38 \* 5 \* 3600; % Total capacity of the battery

% Calculate derived quantities

Tm = current \* km; % Fixed motor torque in N.m while current is applied

Jef = Jmotor + (rmotorS / rwheelS)^2 \* (Jwheel + rwheel^2 \* m); % Effective moment of intertia in kg.m^2

ratios = rmotorS / rwheelS \* rwheel;

% Initialise iterated variables

omega = 0;

dt = 0;

origVelocity = 0;

dtcons = 1;

origDisplacement = 0;

r\_wind = 0;

index = 1;

while dt < 30

dwdt = 1 / Jef \* (Tm - (ratios \* (fd0 + (m \* g \* sina) + r\_wind)));

omegaNew = omega + tdelta \* dwdt; % New acceleration

velocityNew = omegaNew \* ratios; % New velocity

r\_wind =(beta \* velocityNew .^ 2); % New wind resistance

displacementNew = origDisplacement + tdelta \* (origVelocity + velocityNew)/2; % New distance

omega = omegaNew;

origVelocity = velocityNew; % Update arrays

origDisplacement = displacementNew; %

Varray(index) = origVelocity; %

Darray(index) = origDisplacement; %

index = index + 1;

dt = dt + 1;

end

usedC = current \* dt;

percentageLeft = usedC/totCap \* 100;

remainingCapPercentage = 100 - percentageLeft;

Irequired = (ratios \* (fd0 + m \* g \* sina + beta \* origVelocity.^2)) / km;

while remainingCapPercentage > 0

Tm = Irequired \* km;

dwdt = 1 / Jef \* (Tm - (ratios \* (fd0 + m \* g \* sina + (beta \* origVelocity .^ 2))));

omegaNew = omega + tdelta \* dwdt;

velocityNew = omegaNew \* ratios;

displacementNew = origDisplacement + tdelta \* (origVelocity + velocityNew)/2;

omega = omegaNew;

origVelocity = velocityNew;

origDisplacement = displacementNew;

usedC = usedC + Irequired \* dtcons;

percentageLeft = usedC/totCap \* 100;

remainingCapPercentage = 100 - percentageLeft;

Varray(index) = origVelocity;

Darray(index) = origDisplacement;

index = index + 1;

end

%----------Plotting----------%

figure(1)

plot (0:tdelta:tdelta \* (index - 2), Varray)

grid on

figure(2)

plot (0:tdelta:tdelta \* (index - 2), Darray)

grid on

end