ECEN 415 Assignment 4

Niels Clayton : 300437590

System Specifications:

- Achieve an altitude (x_1) of at least 1500m
- Achieve a vertical velocity (x_2) of as large as possible
- Achieve a downrange position (x_3) between 50m & 100m
- Achieve a downrange velocity (x_4) of 20m/s
- Achieve a pitch (x_5) of 10° & pitch rate (x_6) of 0°/s
- No fule remaining (x_7)

Changes to the model:

The initial mathematical model of the rocket only functioned correctly for positive inputs $\mathbf{u}$, limiting the rocket to upwards motion and clockwise rotation. This had the effect of 'refuelling' the rocket when negative inputs were used. Because of this, the fuel consumption equation was altered to allow for negative $\mathbf{u}_2$ values and anti-clockwise rotations. The final model equation was as follows:

$$x_7 = \frac{-1}{\eta} \cdot (u_1 + |u_2|)$$

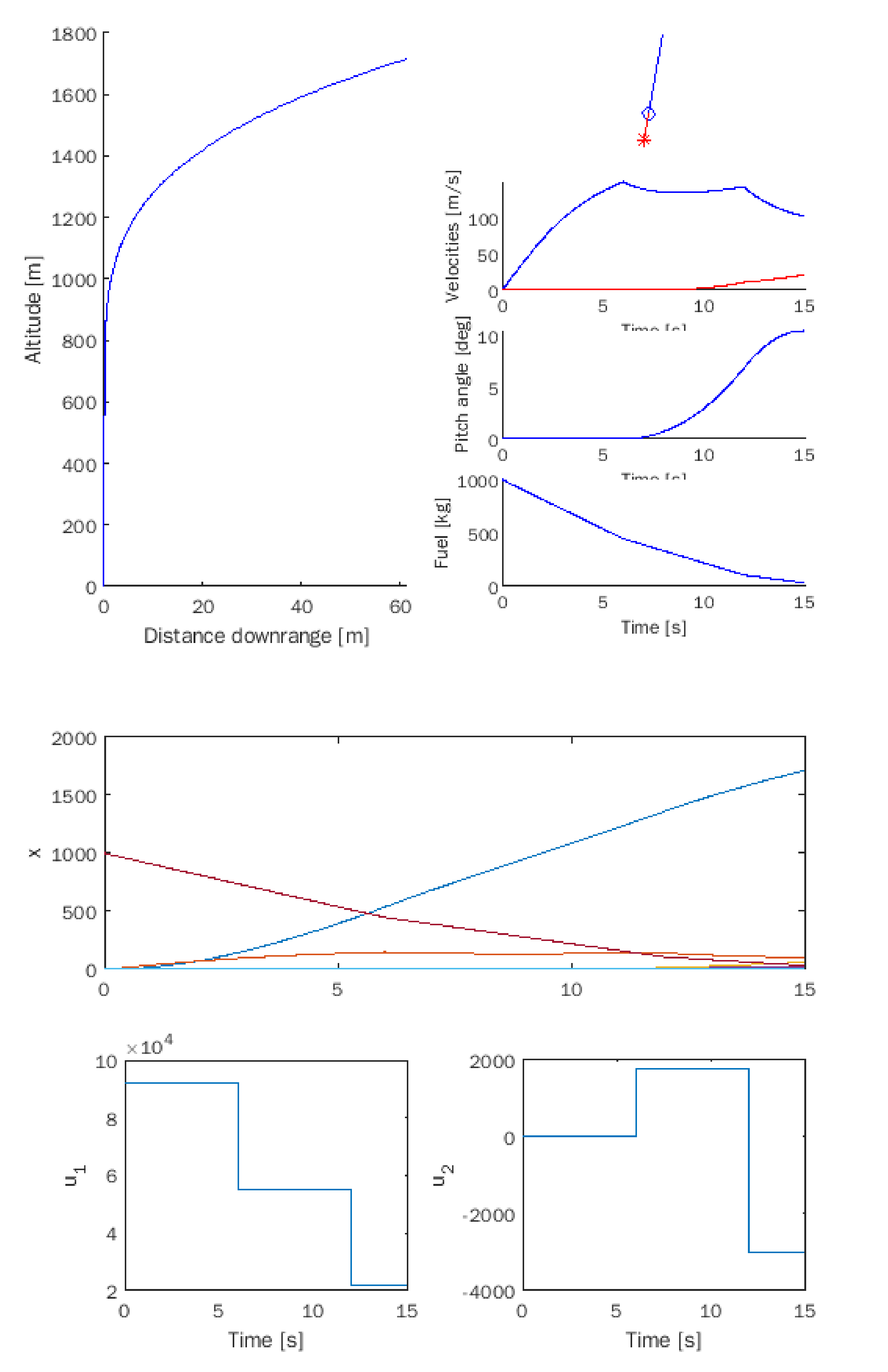
Approach

Open Loop Control - Static

Initially, to gain understanding of how the system responds to varying inputs, an 'open loop' controller was designed. This open loop controller worked by simply setting the two inputs u_1 and u_2 to predefined values at varying time steps. The basic code, as well as the performance of this controller can be seen below.

This controller provided us with a baseline performance for the rocket to achieve. The current states of the rocket ($\mathbf{x}$) at each controller time-step were also recorded and saved for use in further controllers as possible target locations for these controllers to achieve. These states were saved as the matrix x_{target} and saved in "x_target.mat".

```
if t < 6
    u1 = 92000;
    u2 = 0;
elseif t < 12
    u1 = 55000;
    u2 = 1790;
else
    u1 = 22000;
    u2 = -3000;
end
```



	altitude	vel_vert	distance	vel_horiz	pitch	pitch_rate	fuel
1	1.7137e+03	101.5699	61.3932	20.7220	10.4151	-0.0728	32.2600

Open Loop Control - Dynamic

Next, a more dynamic open loop controller was attempted. This controller linearises the non-linear rocket model using Jacobians to calculate Df_x and Df_u. Using these Jacobians, the system is linearised at each time step, substituting in the current rocket state and inputs to generate the state space matrices $\mathbf{A}$ and $\mathbf{B}$.

Using these matrices, the equation for the next required u to achieve a given x_{target} was derived from the following equations:

$$\dot{x} = \mathbf{A} \cdot x + \mathbf{B} \cdot \ u$$

$$x_{\text{target}} = \dot{x} + x$$

If we rearrange these we get the following equation for the required u:

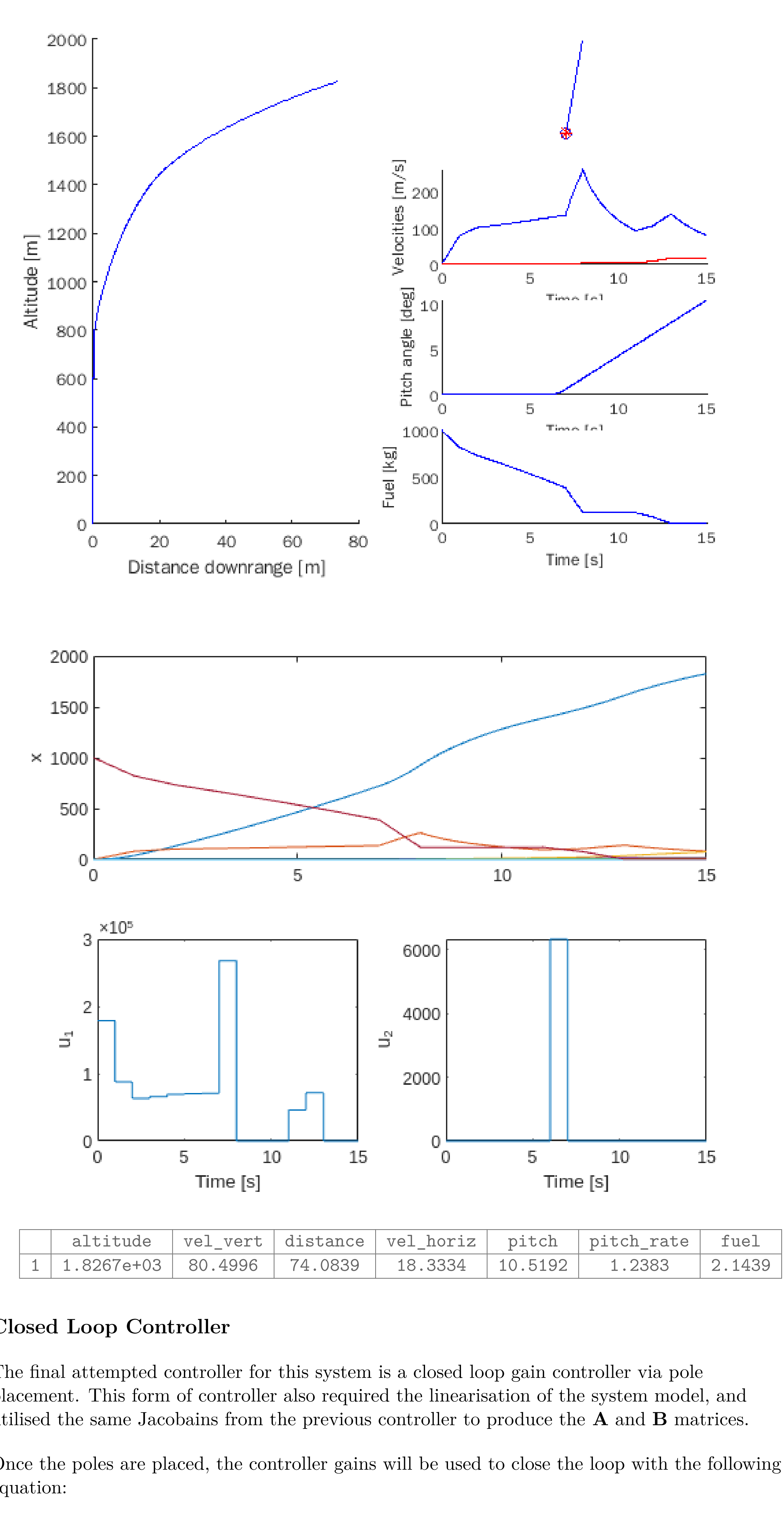
$$u = \mathbf{B}^{-1} \left(x_{\text{target}} - x - \mathbf{A} \cdot x \right)$$

Using this equation, a controller was designed to take the linearised state space system as well as a target state, and provide the required inputs to achieve that state.

This posed the problem of requiring knowledge of how we want to system to respond, and what suitable target states are for the controller.

Initially, the system states of the static open loop controller were utilised as the target states for this controller. This however did not function as expected, and the final state of the system varied greatly from the targeted values. To resolve this, the target states for this controller were set to be a linear interpolation between some initial values and some final target state. This was found to be very difficult to tune, most likely due to the un-invertible nature of the 'thin' $\mathbf{B}$ matrix requiring the use of the pseudo-inverse approximation.

Despite this a controller was tuned to closely meets the specified requirements, and it's performance can be seen in the figures and tables below.



	altitude	vel_vert	distance	vel_horiz	pitch	pitch_rate	fuel
1	1.8267e+03	80.4996	74.0839	18.3334	10.5192	1.2383	2.1439

Closed Loop Controller

The final attempted controller for this system is a closed loop gain controller via pole placement. This form of controller also required the linearisation of the system model, and utilised the same Jacobains from the previous controller to produce the $\mathbf{A}$ and $\mathbf{B}$ matrices.

Once the poles are placed, the controller gains will be used to close the loop with the following equation:

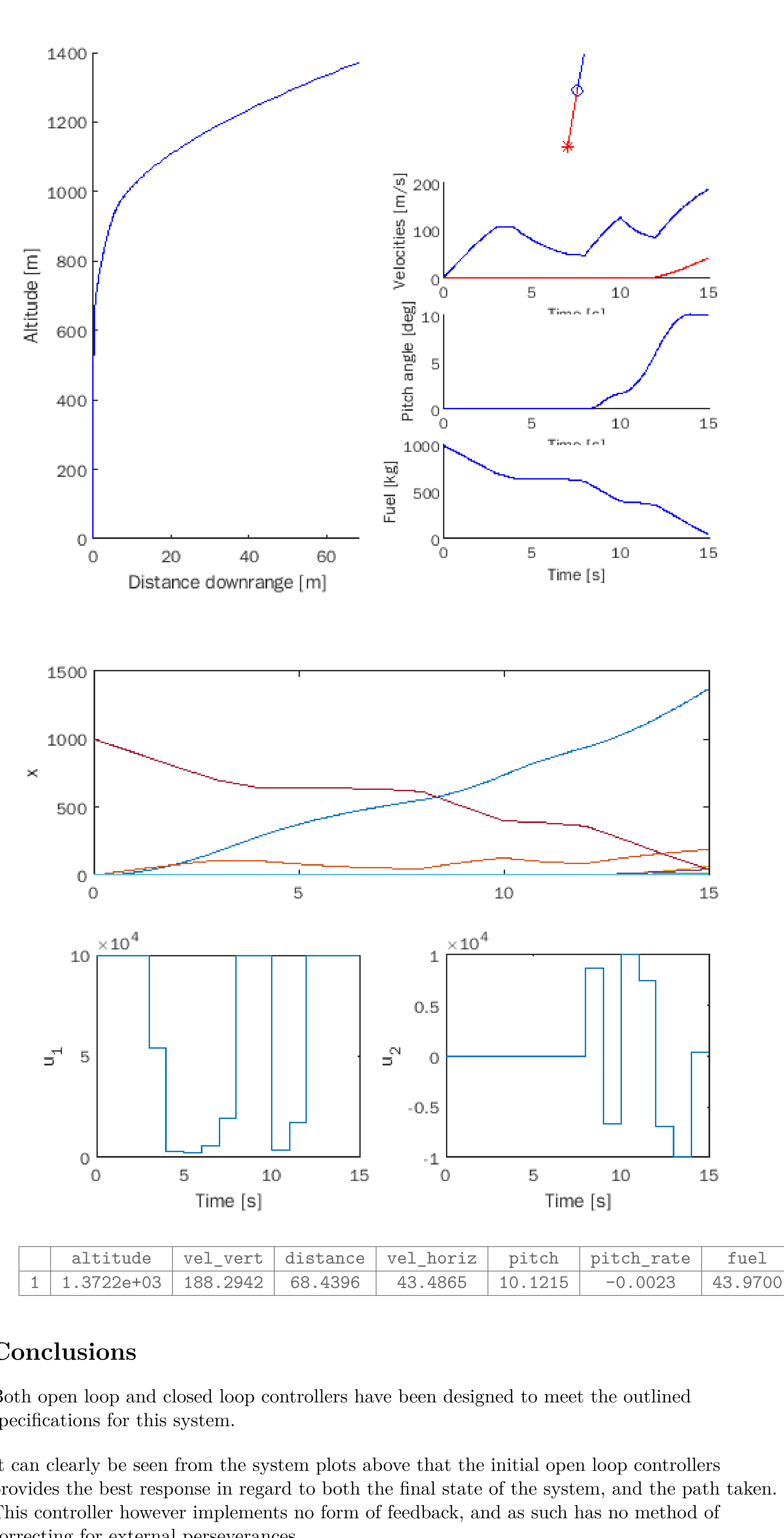
$$u = -\mathbf{K} \left(x - x_{\text{target}} \right)$$

The first challenge with this controller design was the placement of the system poles. As this system is not time invariant, the optimal pole locations and thereby controller gains are not constant. Due to this, the use of the matlab `'place()'` command was not feasible, as the optimal locations for all system poles at each given time-step would be very tedious to calculate.

To overcome this, a static linear quadratic regulator was used to calculate the controller gains at each time step. This controller required inputs $\mathbf{A}$ and $\mathbf{B}$ (the linearised system model) as well as $\mathbf{Q}$ and $\mathbf{R}$. Both the $\mathbf{Q}$ and $\mathbf{R}$ matrices sets bounds on the controller, specifying how much we wish the controller to be able to deviate from our target value, with larger values allowing for larger deviation.

To tune this controller, the selection of input and output bounds must be completed, as well as the selection of the target states for the controller to achieve. Due to the large number of variables to tune for this controller, it was decided that the system state matrix saved from the first controller would be used for the target values, to decrease the number of parameters. From here the bounds were tuned on a basis of which final states were deemed to be most important for the controller to achieve. In the case of this controller, emphasis was placed on both the pitch angle and pitch rate of the rocket, allowing for more deviation in the final state of the fuel.

The final tuned controller can be seen operating below. Due to the emphasis placed achieving the pitch angle and rate, it can be seen that the controller maintains this value. It can also be seen from that table below that the controller closely meets the majority of requirements outlined.



	altitude	vel_vert	distance	vel_horiz	pitch	pitch_rate	fuel
1	1.3722e+03	188.2942	68.4396	43.4865	10.1215	-0.0023	43.9700

Conclusions

Both open loop and closed loop controllers have been designed to meet the outlined specifications for this system.

It can clearly be seen from the system plots above that the initial open loop controllers provides the best response in regard to both the final state of the system, and the path taken. This controller however implements no form of feedback, and as such has no method of correcting for external perseverances.

Although both the dynamic open loop and closed loop controllers achieved the majority of system requirements, the tuning of the controllers was complex and tedious. However, the closed loop controller integrates feedback into the system and may allow for the rejection/correction of external errors.

Overall, of the three controllers designed and discussed, the static open loop controller provides the best response and easiest implementation for a system with no external forces, while the closed loop controller provides a more robust controller design.