MSc Project Report
School of Engineering and Computer Science
University of Hertfordshire

Modelling the guidance system of a RADAR missile attacking a linearly moving target using MATLAB/SIMULINK

Report by
Akin Kucukkurt

Supervisor
Jacob Morewood

Date
16/01/2023

1 DECLARATION STATEMENT

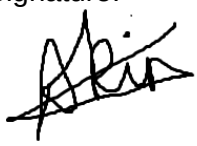
I have read the detailed guidelines to Students on academic offenses and misconduct information (Click this link to read the Assessment Offences and Academic Misconduct web page)

I have read the information about the Academic integrity, misconduct and plagiarism posted on the Canvas Project module page, attended the seminar and/or watched the recording delivered by the School Academic Integrity Officer (SAIO) - Dr Funlade Sunmola/Dr Muhammad Jamro.

I understand the University process of dealing with suspected cases of academic misconduct and the possible penalties.

Therefore, I certify that the work submitted is my own and that any material derived or quoted from the published or unpublished work of other persons has been duly acknowledged. (Ref. UPR AS/C/6.1, section 7 and UPR AS/C/5, section 3.6)

Signature:



Student name: Akin Kucukkurt
SRN: 16053406

UNIVERSITY OF HERTFORDSHIRE
SCHOOL OF PHYSICS, ENGINEERING AND COMPUTER SCIENCE

2 ABSTRACT

This is a report on the Masters project to model a RADAR missile guidance system attacking a linearly moving target. A base framework 2-dimensional guidance system was taken from MATHWORKS based on SIMULINK and modified to add RADAR MATLAB simulation that it lacked, dynamic missile thrust, dynamic mass, dynamic inertia and an easy-to-use way of allowing the user to modify and set initial conditions. The project conclusion was a successfully adequate simulation of a RADAR guidance system that can be understood by someone not familiar with the subject field and be used to learn about RADAR and missile guidance simulation.

3 ACKNOWLEDGEMENTS

I would like to thank Jacob Morewood for being my project supervisor and providing much needed guidance for the project.

4 TABLE OF CONTENTS

5 LIST OF FIGURES

1. Introduction

1.1 Background

The F-35 is the United States of America's latest generation of fighter aircraft which is curiously defined as primarily a stealth aircraft. Stealth is hardly the first word that comes to mind when one thinks of a fighter jet type aircraft and its classification warrants an explanation and is a glimpse into the future of warfare.

Another type of stealth vehicle many would be familiar with is the submarine, and it is here that one can find the fighting tactic of this new aircraft. Submarines are essentially built around signals and information, that piece of information being sound. It fights entirely by listening to sound, and sending out sound for information through a system known as SONAR (Sound Navigation and Ranging) which is the aquamarine environment equivalent of RADAR (Radio Detection and Ranging). It is the eyes and ears of the submarine. It picks up and processes sound echoes from its own SONAR and ambient sound from other SONAR systems or sounds like engine noises. This can be similarly applied to RADAR signals.

Finally, RADAR technology has been improved and condensed enough to essentially create a doctrine equivalent of a flying submarine, hence the stealth designation. RADAR has existed for decades and is simply described as sending an electromagnetic signal and reading the signal that bounces back after hitting a solid object. Its main competitor, at least in the context of the military, are infrared systems which works by locking on to high temperatures. RADAR is more complex and has more potential due to the nature of how it works. Infrared systems are easier to counter as you just need to modify temperature which can achieved through the use of burning hot flares which are combined with evasive manoeuvres to achieve a good chance at losing the missile's lock. You can try to do the same to a RADAR missile by dropping various bits of metal (chaff) from your aircraft but this has a much lower chance of success which can only go down as RADAR is developed further.

Hence it is the future of warfare and is the new doctrine of the fifth-generation fighter aircraft. No longer will there be visual contact in air engagements, instead battles will rely on RADAR, information and signal processing. Eventually as the technology gets more compact and advanced, other aspects of warfare which haven't already switched completely to RADAR will also no doubt start using RADAR where applicable.

It is for this reason that the aim of this project is modelling the guidance system of a RADAR missile attacking a linearly moving target using MATLAB/SIMULINK. SIMULINK is adept at modelling this kind of continuous state system.

1.2 Project plan and outline

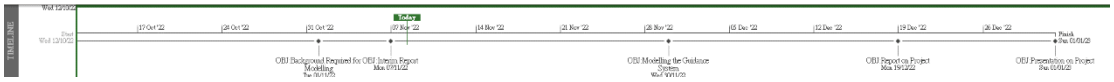


Figure 1 – Timeline of the Gantt chart

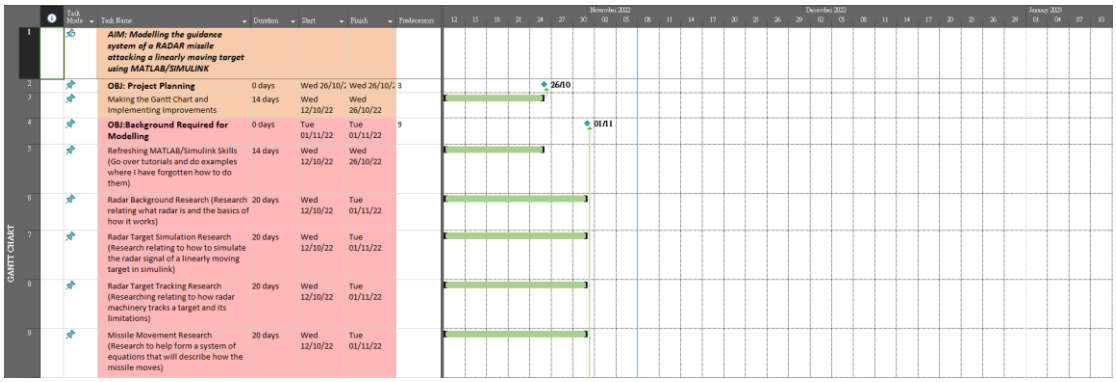


Figure 2 – The aim, first two objectives and relevant tasks of the Gantt chart

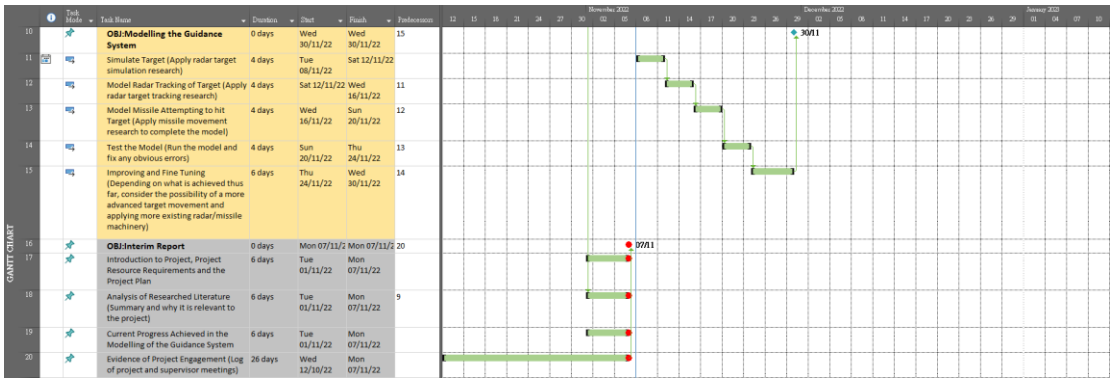


Figure 3 – The third and fourth objectives and relevant tasks of the Gantt chart

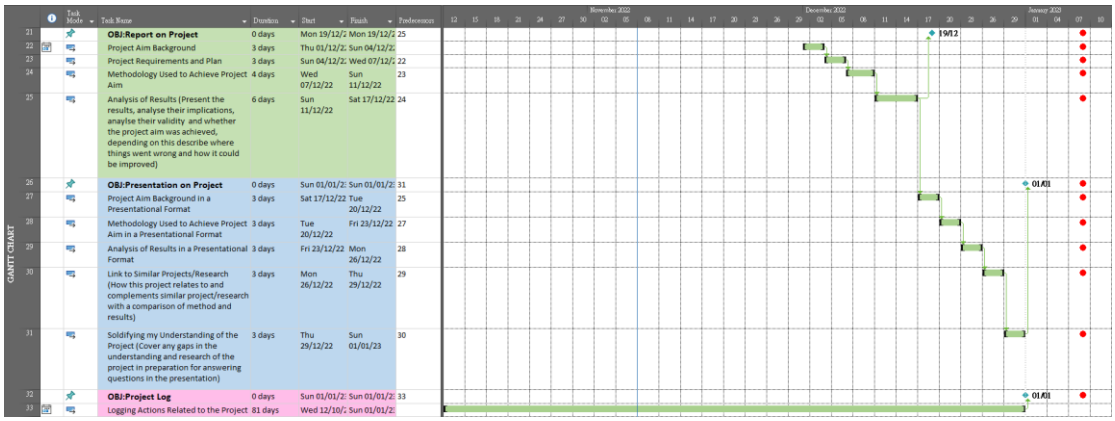


Figure 4 – The fifth, sixth and final objectives and relevant tasks of the Gantt chart

Figures 1 to 4 show the Gantt chart plan for the project which was created in Microsoft project using the University of Hertfordshire's computers. Objectives are outlined which are further

split into specific tasks. The basic required objectives for the module are included, making a Gantt chart, interim report, project report, project presentation and project log.

The background objective is about finding information on RADAR, missile systems and refreshing necessary MATLAB/SIMULINK knowledge so that I can understand the research practically and obviously to be able to actually model the guidance system. The actual modelling of the guidance system is broken down into a best estimate into the necessary individual tasks.

The chart gives extra room for delays by finishing about a week before the final deadline but this chart is meant to very adaptive as it can be expected that the days outlined for each task may not be completely accurate. Most likely some tasks will finish a bit earlier and some a bit later, these should mostly cancel each other out but the extra week is there if the balance tips towards delays or there is some other problem like illness. Constant checking in with the plan and updating it to account for circumstances is the key to making it work.

1.3 Ethical considerations

All software used is provided licensed by the University of Hertfordshire. The implementation of RADAR and this whole project is completely software orientated so there are few practical considerations for ethics. However, it must be said that this project is not built with any kind of security in mind like signal interference, hacking or signal jamming. Implementing this project directly into a real-life missile would therefore be a bad idea as it is missing any level of the military security technology that is classified and well outside of the requirements of this project. This project is mainly about providing a RADAR software project for learning, understanding and as an introduction point to RADAR. The ethics of the usage of deadly force is a concern for governments not physics students and is therefore not discussed here.

2. Literature Review

2.1 MATLAB missile guidance system example

MATHWORKS provides a fully functioning SIMULINK model of a missile guidance system as well as the relevant documentation explaining how it works[1]. It is however missing crucial aspects that would make it realistic as it models things like the aerodynamics of the missile and relevant physical equation quite well but has no simulation of a RADAR system at all.

This is perfect as adding that fully functioning RADAR system is the aim of this project and the existing model provided allows me to start with a working frame. This allows me to not have to model aspects of this whole system that are not inside the scope of this project as well as being very time intensive like target course prediction and missile aerodynamics.

Therefore, this is one of the most important sources in providing practical help for this project. The explanation of this system given in the documentation allowed me to understand the system so I would know how to modify it to make it adequately realistic. MATHWORKS is a major international company that provides high-end simulation software used in all corners of the world, therefore it can be trusted as a reliable source as they are judged internationally by all kinds of experts looking for mistakes in their newest work.

It is 2-dimensional but that is adequate for the scope of the project, it makes the problem much less complex and this is quite important as the project is too time limited to create a fully functioning 3-dimensional system. Especially considering that I started this project with extremely surface level RADAR knowledge and a full 3-dimensional implementation is complex enough to be an entire career as can be seen from ‘Missile Flight Simuation’ by J Strickland[2] .

It is good enough to explore the complexities of RADAR whilst also being able provide an understanding of RADAR implementation (provided by this project and report) that can be digested by those not familiar with the subject.

2.2 Simulating Test Signals for a Radar Receiver

‘Simulating Test Signals for a Radar Receiver’ is a piece of documentation that shows how MATLAB RADAR frameworks can be used as MATLAB code to simulate a RADAR system that provides a realistic range reading[3].

It is immensely useful for this project as it provides the simulation of a RADAR system that is implanted in the aforementioned guidance system example to achieve the aim of this project Modelling the guidance system of a RADAR missile attacking a linearly moving target using MATLAB/SIMULINK

whilst providing documentation that allowed me to understand how it works and its few limitations in simulating RADAR. It also gave me the best understanding of how RADAR works practically.

As mentioned before MATHWORKS can be considered a reliable source. It takes into account all relevant variables and conditions of a RADAR system and performs realistic signal processing to achieve a range reading. It is even designed to be used 3-dimensionally and works in 2-dimensions by setting the location of all objects in that third direction as zero.

2.3 Missile Flight Simulation 2nd edition

The simply titled “Missile Flight Simulation 2nd edition” by Jeffrey Strickland fulfils its purpose as a source that provides a complete quantitative and qualitative explanation of a missile, how it attacks a target, all the different types of missiles, guidance systems (including RADAR) and most importantly how all of this can be simulated[2].

From 1994 to the publish date of the book in 2015 Jeffrey Strickland has been working in the U.S defence industry in missile defence studies and missile flight simulations[2]. The U.S spends the most amount of money on its military in the world and is no doubt one of the nations with the most technologically advanced military system. This combined with 14 citations, the length of the book at over 600 pages and the fact that military speaking 2015 is very cutting edge, as technology research in that industry moves slowly with complete implementation of technologies taking a very long time, not the mention that the newest information is impossible to get as it is usually classified, this source is very useful.

It provided great help in being used as check for the whole system by comparing my project with the system and explanations shown in this book. From there I was able to realise some crucial missing aspects of my system and therefore add them. This meant the addition of allowing 2 motors to be simulated in the missile as a boost and sustain system. It provided the method used for simulating the change in the pressure component of the missile thrust, the change in mass as the missile burns fuel and the change in the moment of inertia of the missile. It also provided great help in understanding the background of missile guidance system simulation and achieving a more complete understanding of my own system.

2.4 Introduction to RADAR

With a large number of citations at 353 according to Google Scholar and being publicly available, “Introduction to RADAR” by Merrill I. Skolnik gives a mainly qualitative introduction into the concept of RADAR for those who are not familiar with the technology.

As it is a very old source, released in 1962 and the fact that it is concerned with simply giving the fundamentals of RADAR, this source did not provide much help for the project other than background reading. It was useful in giving a basic overview of a RADAR system with a system shown that involves a transmitter, duplexer, antenna, receiver, signal processing, data processing, displays, radar control and the radar waveform[4]. This had little impact on providing anything practical for the project.

2.5 Automotive radar target list simulation based on reflection center representation of objects

This source has its usefulness in showing what won’t be modelled in this project due to it being outside the scope of the project but is something that is necessary if one wanted to create a fully accurate RADAR system. It is therefore useful in creating a background understanding of what is going on in the field of RADAR but it is far too advanced to be used for the scope and limited time of this project. It shows how RADAR data can be generated that takes into the account the shape of the object that is being pinged by simplifying a car into reflection points and how this compares to real data[5].

The practical SIMULINK system in this project essentially treats the target as a point (with a stated RADAR cross section area), this simplification drastically reduces the complexity of the project whilst still giving a structurally sound RADAR model. This source shows possible future development of this project (especially if one wanted to develop a 3-dimensional upgrade to this project), and serves as introduction point for this missing feature.

2.6 Cognitive radar for target tracking using a software defined radar system

This source details the creation of a smart system that adjust a RADAR devices variables based on each signal received to track a target and improve the lock onto the target where possible[6]. The tracking part of this project is already built into the system example provided by MATLAB that is used as the root of this project. It is adequate at its job and the method inside this source is too advanced for the scope as well as the limited time of this project.

It is simply not necessary to implement this to achieve the project aim. It would be useful for someone building a fully accurate 3-dimensional RADAR system that would require this kind of advanced method.

Modelling the guidance system of a RADAR missile attacking a linearly moving target using MATLAB/SIMULINK

2.7 Modeling and simulation of a full coherent LFM pulse radar system based on Simulink

An already existing model of a RADAR system created in 2013 inside SIMULINK is qualitatively shown, the model involves a Linear Frequency Modulated signal generator, an echo block containing information on the echo signal, clutter, system noise, matched filter, moving target indication, moving target detection and constant false alarm ratio block which factors the system's noise into processing the signal[7].

Despite including a few quantitative elements, it lacks the information necessary to reproduce this model. Whilst this project also models a pulse RADAR, this source is focused on a 3-dimensional implementation. This means it includes more complex versions of elements existing in this project and newer elements like clutter simulation, filtering of said clutter and a coherent RADAR system where all the properties of the RADAR wave are known and are used to filter the signal which is not the case with this project.

It doesn't provide any practical help for this project other than as a comparison for the overall structure of this project's system (showing the difference of a more complex and realistic implementation of RADAR) and doesn't provide a good enough explanation and breakdown of the system inside the source that would be of use to anyone who isn't making a career in RADAR simulation. This is of course well outside the scope of this project and its limited time. It can be useful for anyone wanting to improve upon this project and move it to the third dimension.

3. Methodology

3.1 Overview of the existing Missile Guidance System Example

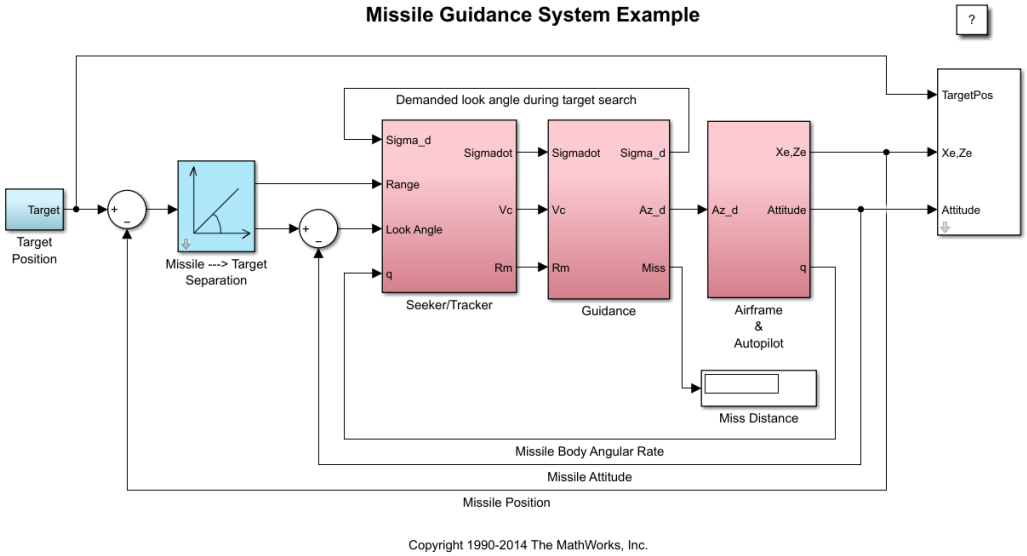


Figure 5 - Missile Guidance System SIMULINK example as provided by MATHWORKS [1]

In a real-life scenario, a missile knows where the target is by knowing the angle to the target (based on the angle at which the radar system is receiving echoes from the target) and the range of the target (by processing the signal echo and filtering to determine range). In figure 5 it can be seen that the system uses no radar system to find the range and simply uses the actual known values for the range which is completely unrealistic. The missile notes the angle and range of the target over time to predict the flight path of the target.

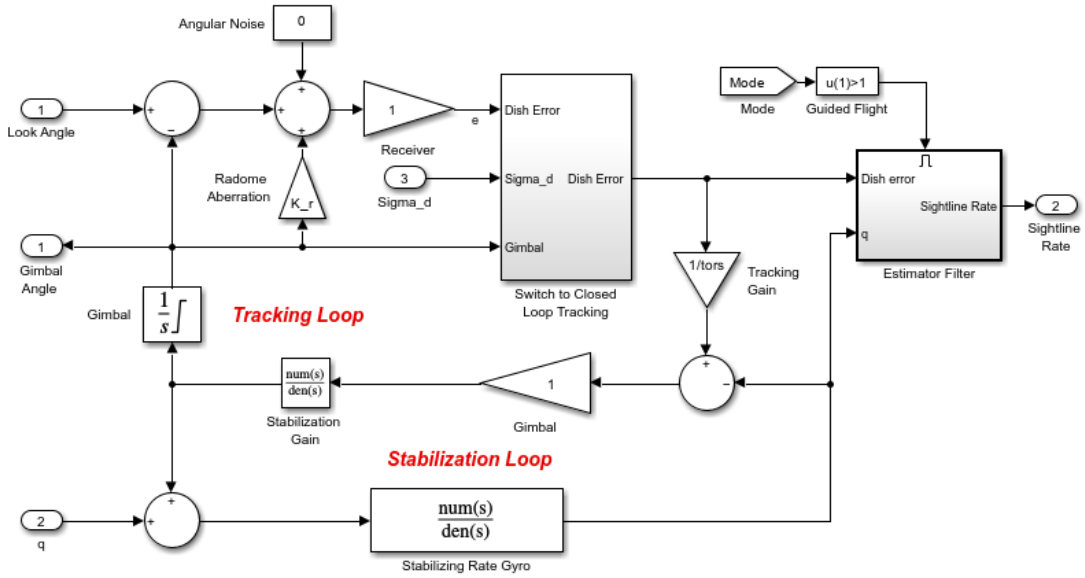


Figure 6 - Radar seeker dish angle system in the Missile Guidance System Example by MATHWORKS [1]

However, the angling of the seeker dish is modelled adequately, as it seeks and takes time to move as well as introducing radome error caused by the protective covering over the seeker. It even allows you add angular noise if you wish. This is visible in figure 6.

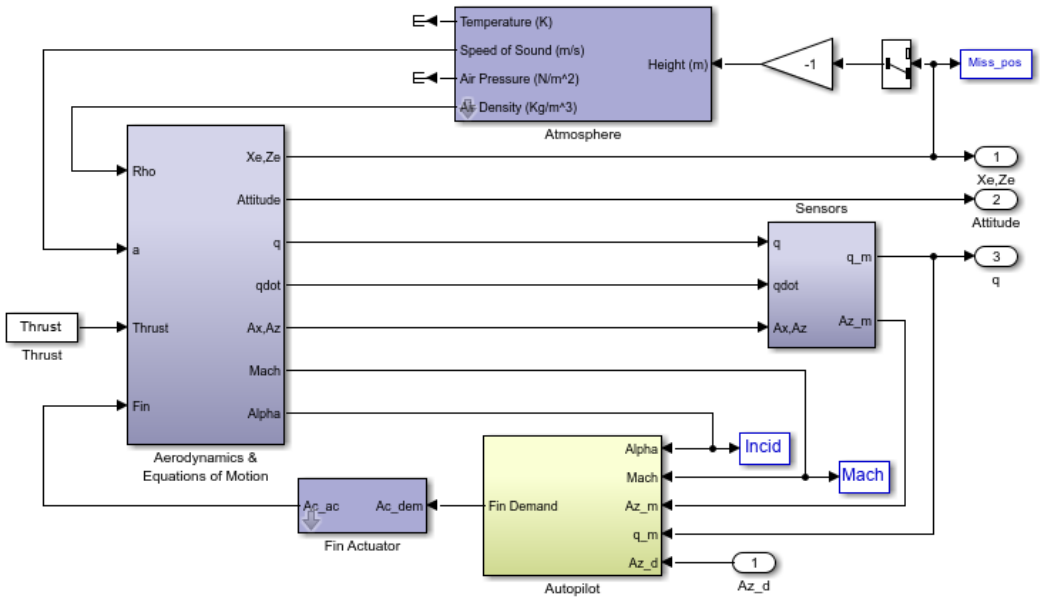


Figure 7 - Airframe and Autopilot system in the MATHWORKS example [1]

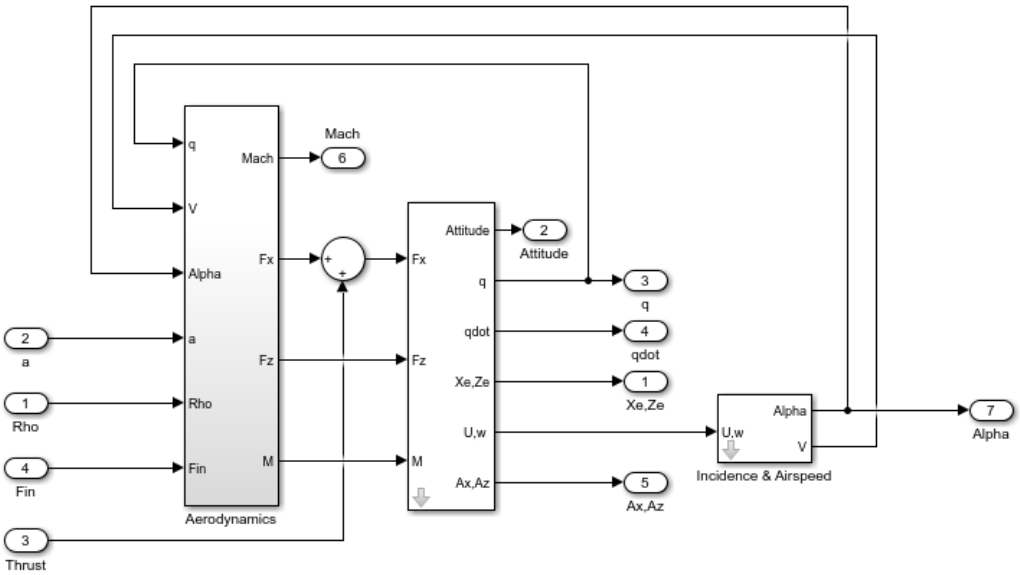


Figure 8 - Aerodynamics and Equations of Motion system in the MATHWORKS example [1]

The target moves in a straight line with a constant velocity, the missile starts at 984 m/s and has a constant thrust with a sufficient model for the aerodynamics and physics of the missile (including the fins). This can be seen in figures 7 and 8.

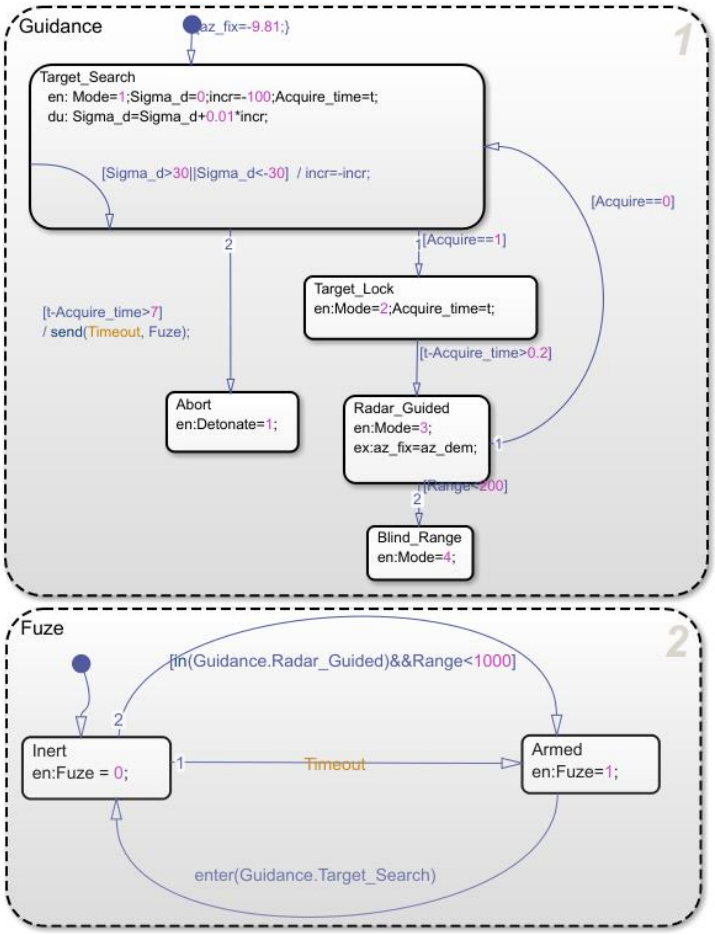


Figure 9 - Guidance processor system in MATHWORKS example [1]

From figure 9 it can be seen that the guidance system starts in a state of searching for the target in a 30 to -30 degree angle range then it has 2 possible next states, one in which after 7 seconds of no target lock the missile aborts and explodes, in the other one the target falls within the beamwidth of the seeker and the target is acquired.

A delay of 0.2 seconds is introduced so that the missile can predict the future position of the target which becomes the direction of flight. Whilst reducing this delay has no impact on the original example, once an actual RADAR system is introduced this and the missile starts at a standstill, this becomes significant.

In the next state the missile autopilots towards the target, once it is within 100 meters of the target it is considered as too close for RADAR so it continues on its course blindly. A constant check is run such that once the missile is within 1000 meters it calculates when it will hit the target and arms itself for that many seconds.

3.2 Results of the existing Missile Guidance System Example

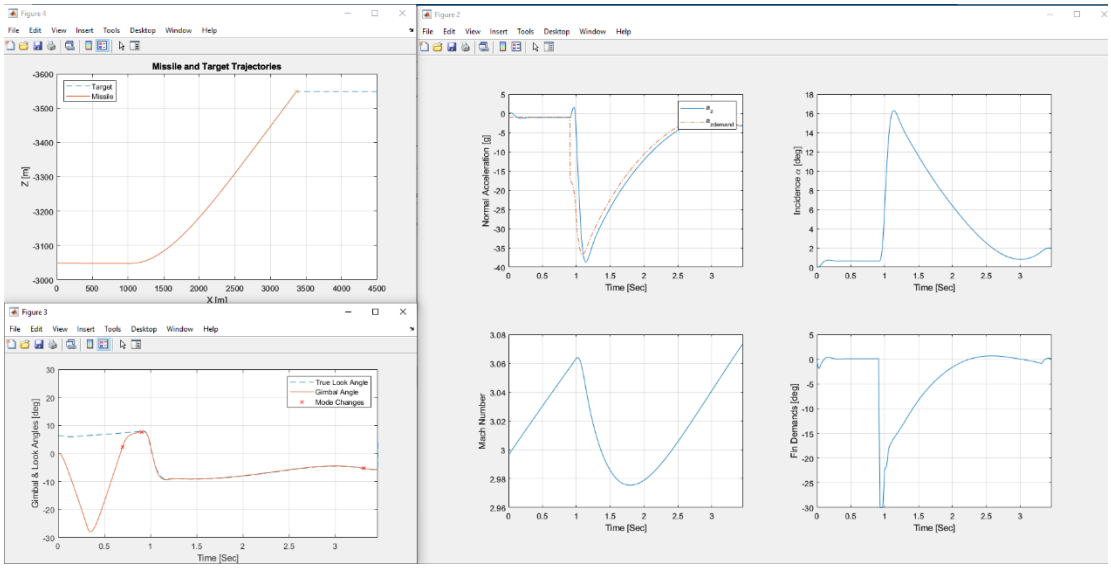


Figure 10 - Result graphs from the original Missile Guidance System MATHWORKS example [1]

Analysing the results from the MATHWORKS example in figure 10 it can be seen that the missile follows a very smooth path from straight on to curving at the target with a linear increase in speed that does experience sufficiently realistic aerodynamic drag but then accelerates linearly once again.

The demands of acceleration on the missile are a smooth line and curve, whilst the missile should experience a somewhat smooth actual acceleration as it moves aerodynamically, the demands should be constantly changing as the RADAR system updates constantly with an imperfect range reading. This is no doubt due the usage of the actual range between the missile and target. This also similarly applies to the fin demands on the missile's fins.

The seeker gimbals from 0 to -30 to 30 degrees until its beamwidth puts itself within the range of the true look angle which is the actual angular location of the target, this causes a change of mode in the guidance system as the target is acquired, the seeker attempts to look directly at the target and a target flight path prediction is calculated within the standard delay of 0.2 seconds after which the missile goes into homing mode (the mode changes again) and heads towards the target. This in effect means that even at infinite range, the target would be acquired so long as the seeker's gimbal angle is close to the true look angle. In reality you need to be both looking at the target and have a clear radar signal to have acquired the target.

3.3 Limitation and plans for improvement of the existing Missile Guidance System Example

The limitations dictate what will be added in this project. Firstly, an actual RADAR system that runs to find a real-life accurate value for the range. Secondly a target acquired check that uses a RADAR system along with the existing beamwidth check to confirm if the target is acquired.

Thirdly, a system for dynamically calculating thrust as altitude changes which has an effect on the pressure component of thrust, dynamic mass (as fuel is lost) and therefore dynamic moment of inertia (the missile becoming more agile as it loses mass).

The final change is to be able to modify and test all sufficient relevant initial conditions without having to change the model workspace values inside the model explorer along with annotations stating the full name of each initial condition for ease of use. These initial conditions include things like the physical properties of the radar, target/missile initial location/velocity, seeker search angle and missile properties.

4. Development stages of the improved system

4.1 The RADAR ranging system (first stage)

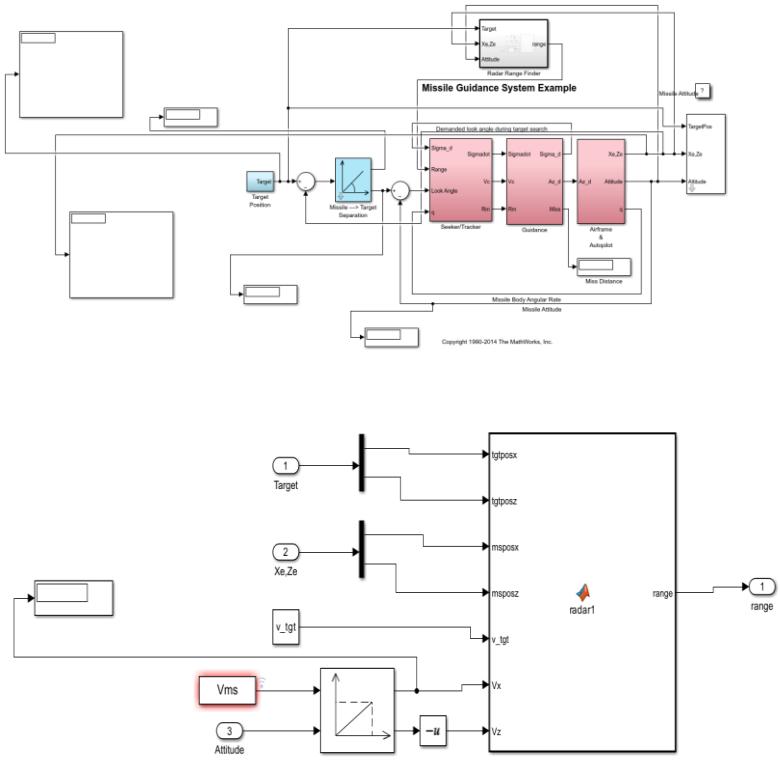


Figure 11 - Overview of the first major development of the Missile Guidance System

The first stage of this project was to add a RADAR ranging system using the existing MATLAB RADAR code. As can be seen from figure 11, the target/missile velocity, position and angular difference is fed into a MATLAB function block which simulates RADAR to find a realistic range reading.

```
function range  = radar1(tgtposx,tgtposz,msposx,msposz,v_tgt,Vx,Vz)

coder.extrinsic("radarfunc");


range=ones(1,1);
range=radarfunc(tgtposx,tgtposz,msposx,msposz,v_tgt,Vx,Vz);

end
```

Figure 12 - Code screenshot of the MATLAB function block

This code inside the MATLAB function block can be seen in figure 12 in which the RADAR function is run extrinsically and the output range value has its type defined. Running it extrinsically allows it to run in a complete MATLAB environment as SIMULINK runs in a

different environment and complex MATLAB code isn't able to run directly inside it. This is also why the output range value has to have its type defined which is an array with just 1 value inside it, being the range at the certain time that the code is run.

The RADAR function went through modifications and to achieve a full understanding of it, the final implementation of it needs to be shown.

4.2 The RADAR ranging system in the final stage of the project

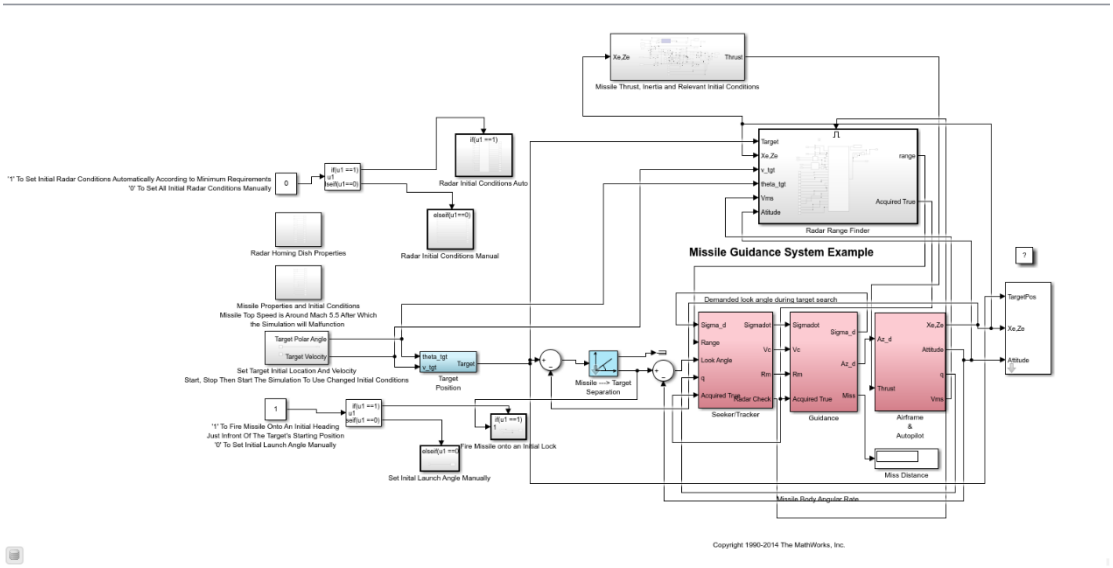


Figure 13 - Overview of the final product of the project

Looking at just the relevant RADAR development in figure 13, there is the ability to set every relevant initial condition by manually entering every physical property of the RADAR or entering some manual values and some desired outcomes which runs code that calculates the physical properties that would be necessary to achieve that outcome and stores these values.

For example, entering the desired maximum range of the RADAR leads to it calculating how powerful the RADAR would have to be to achieve this maximum range. As stated in figure 13 you change the shown value to '0' to use manual values and '1' to use automatic values.

Looking at 13, it can be seen that in the final version I made it so that the variables that were taken in the first iteration in figure 11 now all come from direct connections to the SIMULINK subsystems rather than having some of them be stored in the workspace and called from there. This was done to make the system easier to digest as you can follow where these

values directly came from and only the initial condition values are stored and called from the workspace.

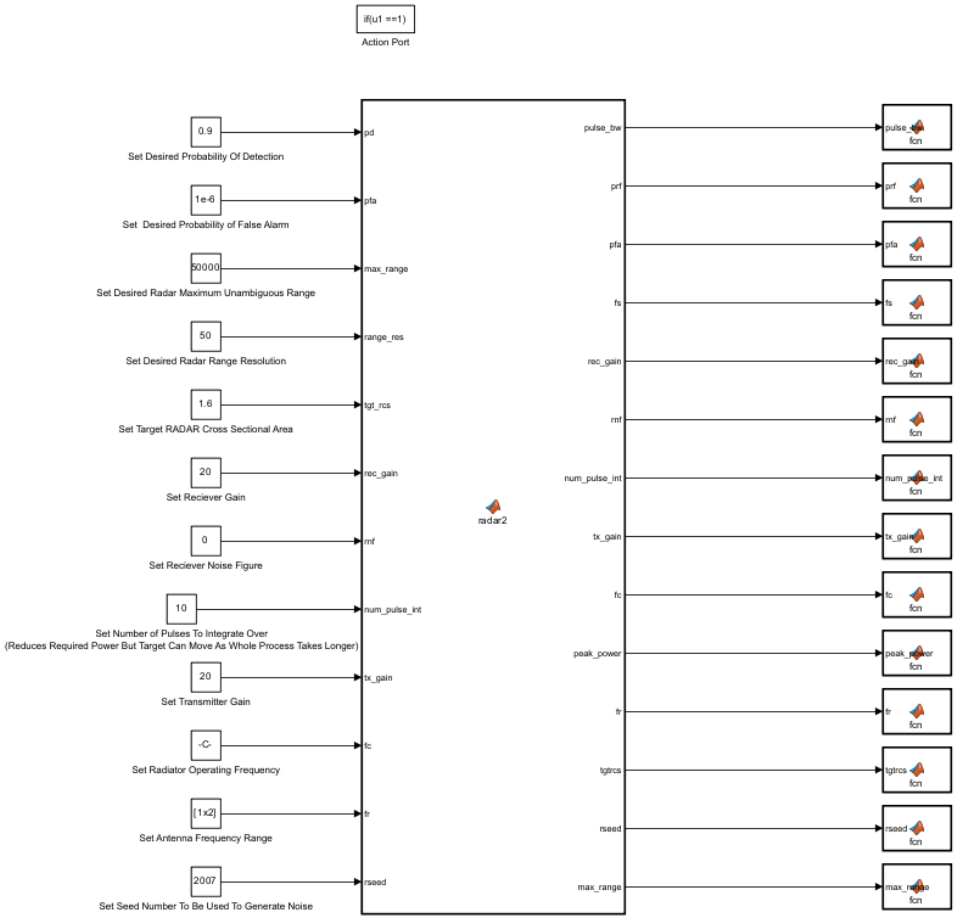


Figure 14 - Overview of the system block of the automatic RADAR initial conditions

The automatic RADAR initial conditions can be seen in figure 14 in which the full name and explanations of each initial condition is stated at the inputs. Manual values are passed straight into the system and straight out of the system with no changes, the automatic values are:

Desired probability of detection (of a target)

Desired probability of false alarm

Desired RADAR maximum unambiguous range

Desired RADAR range resolution (minimum distance difference that the RADAR can detect)

These are used to calculate:

RADAR pulse bandwidth

RADAR pulse repetition frequency

RADAR receiver sample rate

RADAR peak power

The code which calculates these values can be seen in figure 15.

```
function
[pulse_bw,prf,pfa,fs,rec_gain,rnf,num_pulse_int,tx_gain,fc,peak_power,fr,tg
trcs,rseed,max_range] =
radarcalc(pd,pfa,max_range,range_res,tgt_rcs,rec_gain,rnf,num_pulse_int,tx_
gain,fc,fr,rseed)
    tic
    prop_speed = physconst('LightSpeed');
    pulse_bw = prop_speed/(2*range_res);
    pulse_width = 1/pulse_bw;
    prf = prop_speed/(2*max_range);
    pfa=pfa
    fs = 2*pulse_bw;
    num_pulse_int=num_pulse_int
    snr_min = albersheim(pd, pfa, num_pulse_int);
    tx_gain=tx_gain
    fc=fc
    lambda = prop_speed/fc;
    peak_power = ((4*pi)^3*noisepow(1/pulse_width)*max_range^4*...
        db2pow(snr_min))/(db2pow(2*tx_gain)*tgt_rcs*lambda^2);
    fr=fr
    tgtrcs=tgt_rcs
    rseed=rseed
```

Figure 15 - MATLAB code for automatic initial conditions inside the MATLAB function block

The calculations in figure 15 were taken from the original RADAR function provided by MATHWORKS and these specific calculations were taken out to create the two possible options of automatic and manual.

```
function fcn(pulse_bw)
    coder.extrinsic("assignin");
    coder.extrinsic("get_param");
    bswks = get_param('aero_guidance_w_radar','ModelWorkspace');
    assignin(bswks,'pulse_bw',pulse_bw);
end
```

Figure 16 - Example of the MATLAB code used to store output variables

The outputs are stored in the local SIMULINK workspace under their variable names via MATLAB code seen in figure 16 and it is important to note that it is dependent on the file name being what is stated so any change to the file name means you would have to go into each of these output MATLAB function blocks and change the name.

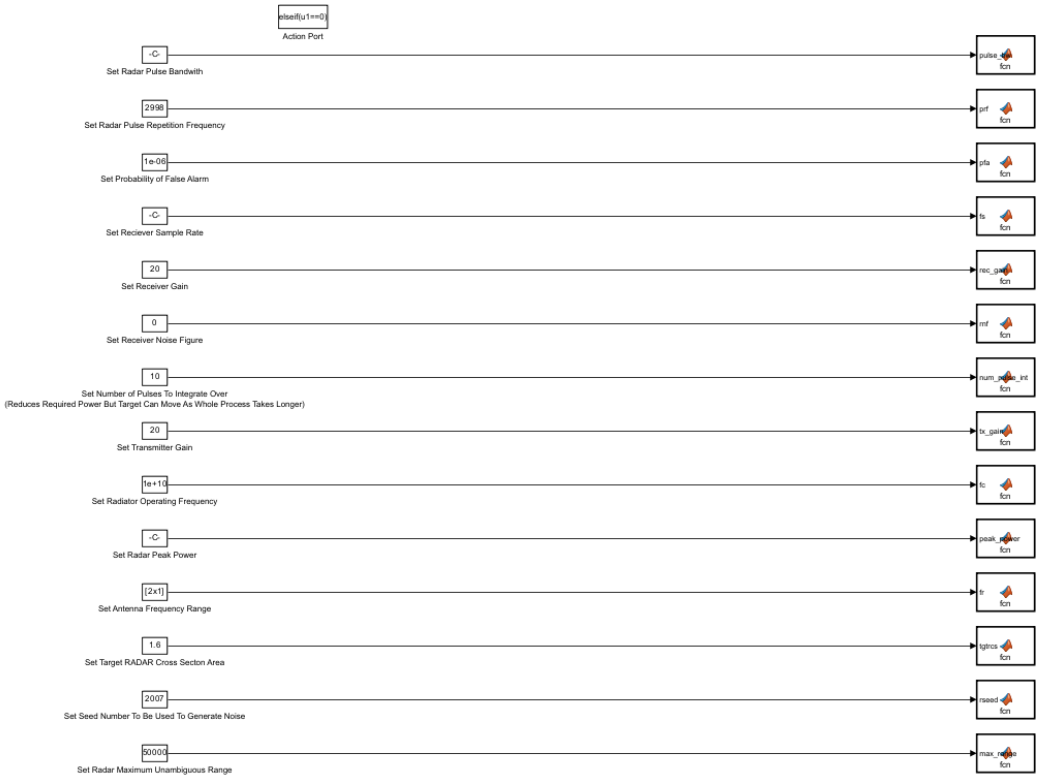


Figure 17 - Manual RADAR initial conditions subsystem

The manual subsystem simply stores the given initial conditions inside the base workspace as shown in figure 16, the values can't be passed directly into the RADAR subsystem because there are two possible versions of the variables and it isn't possible to have both of them as a potential input for the RADAR system which means you would have to duplicate the RADAR subsystem and rather than doing that, I decided to store them in the base workspace instead. The manual subsystem can be seen in figure 17.

4.3 The RADAR ranging target lock check in the final stage of the project

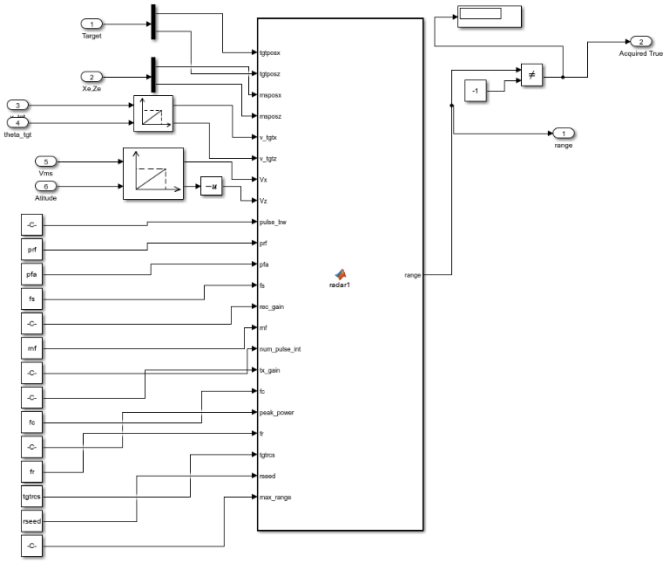


Figure 18 - RADAR subsystem in the final iteration of the project

In figure 18 it can been seen that there is a new addition, apart from all the initial conditions, this was part of the second major stage of development in which the aforementioned RADAR initial conditions were added and the overall system was modified so that there is now a check that once the target is within the beamwidth, the RADAR subsystem is initiated which then returns a -1 value for range if there is no valid value for the range and this is used as the check for whether the target is acquired.

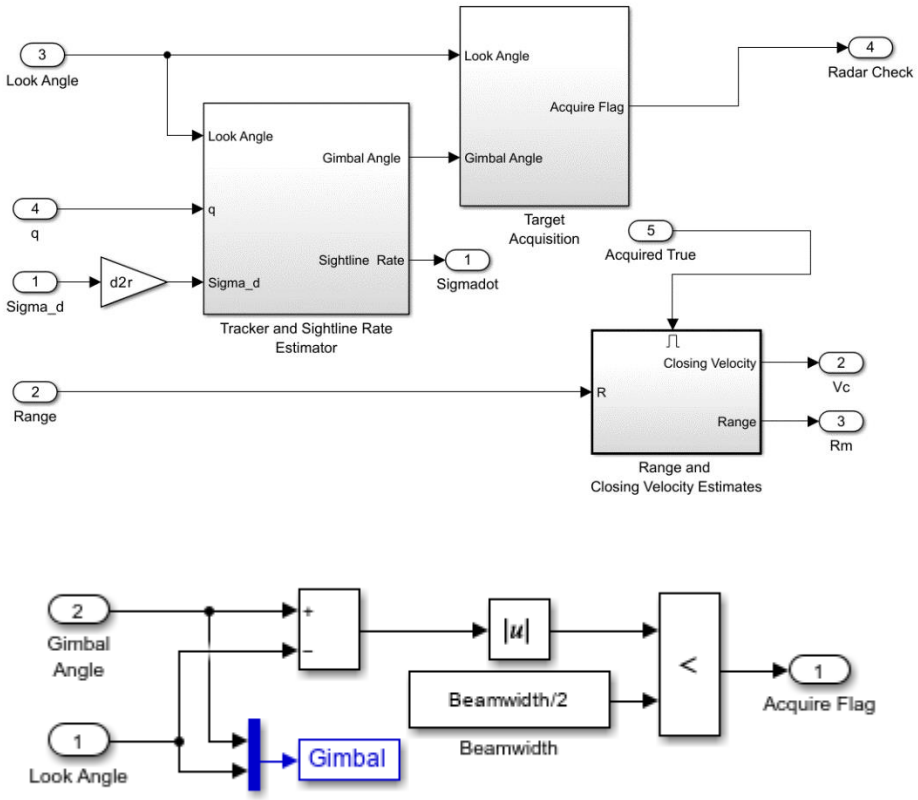


Figure 19- Seeker/Tracker subsystem in the final iteration of the project

From figure 19 it can be seen how the system checks if the target is within the beamwidth and activates the ‘Radar Check’ exit port if this is the case which connects to the ‘Radar Range Finder’ subsystem in figure 18 into an action port which then activates that subsystem to see if there is a valid range value. From figure 13 the acquired status is sent out from one of the output ports of the RADAR subsystem which then feeds into the Seeker/Tracker subsystem and the ‘Guidance’ subsystem to change the mode of the whole system to locking onto the target as seen on the guidance processor in figure 9 as well as activating the range value and closing velocity estimates as seen in figure 19. In the original example, just the beamwidth acquire flag fed straight into these positions so there is now both a beamwidth and a RADAR range value check for target acquisition.

4.4 The RADAR ranging function in the final stage of the project

Appendix B contains the code used to simulate RADAR to find a realistic range reading, the exact explanation of it can be found on the MATLAB documentation [3]. It models a monostatic RADAR system meaning the dish both sends a signal and has a receiver for the returning signal as opposed to a bistatic RADAR system in which there are there are separate dishes, one sending a signal and one receiving which makes it more accurate but the project is about modelling the singular RADAR system inside the missile. A non-coherent source is assumed which means no wave properties of the signal is used for processing the signal echo which would make the RADAR system more accurate but also increase computational time and isn't necessary to have a realistic enough Radar system.

There have been significant changes to the original code. First of all, the part of the code that outputted graphs have been removed as naturally we are only interested in the range reading and SIMULINK doesn't run this properly with those parts of the code still there. The parts of the code that found ideal conditions from desired properties have been commented out because they are now used in the automatic RADAR initial conditions subsystem as previously discussed. There is a list of all the input variables defined as themselves which are commented out, they are there so that you can uncomment them and see those values outputted on the SIMULINK diagnostic window as the system runs if you wish to keep a track of them over time. I used them to make sure the code was running correctly whilst it was being developed.

The next major change is that all RADAR properties have been turned into variables that are taken as inputs to the function.

The final change is at the end of appendix B in which an if statement ensures that if the RADAR range reading is 'null' or in this case 'empty', the range is outputted as '-1' which is part of the checking system as discussed earlier. Otherwise, the range value that is found is outputted as normal. Lastly there is the addition of 'tic' and 'tock' to the start and end of the code which just allows you see how long it takes for this specific piece of code to run in the diagnostic windows of SIMULINK when the system is running, this was also added to the automatic RADAR initial conditions function as well.

4.5 Initial Conditions of the RADAR system, missile launch angle and target lock improvements (second stage)

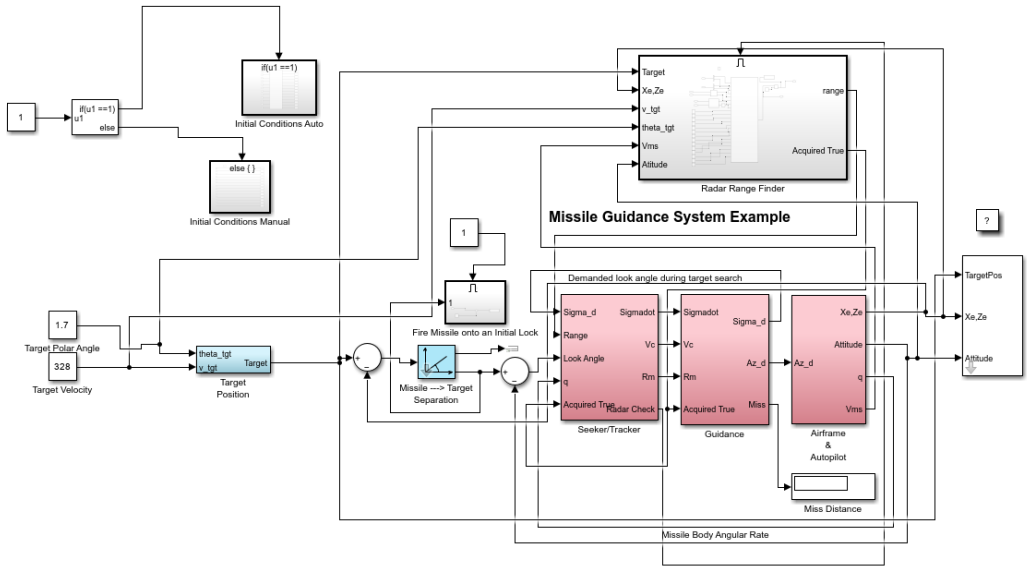


Figure 20 - Overview of the second major development stage of the project

From figure 20 it can be seen how the RADAR system and its initial conditions were first implemented. As well as the aforementioned RADAR ranging and beamwidth check for target acquisition.

There is also the addition of an on and off system for firing the missile onto an initial lock. This simply runs a subsystem that sets the initial angle of the missile as the actual angle between the missile and the target plus a certain number of radians. This value can be changed inside the subsystem.

The file with the some of the new initial workspace variables was added by loading the file through the preload property of the model which was already there but pointed to the workspace variable .m file provided by MATHWORKS.

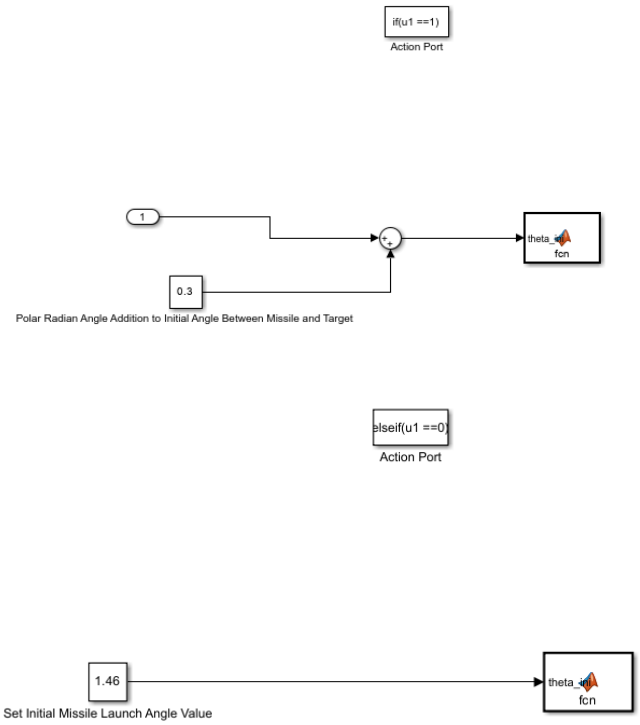


Figure 21 - Subsystems for the initial angle of the missile as seen on the final iteration

It can be seen how this angular system is implanted though the overall system in figure 13 and the specific subsystems in figure 21. Overall, it is a basic system for setting the initial angle of the missile.

4.6 Dynamic missile thrust, mass, inertia and all relevant initial conditions for the system (third and final stage)

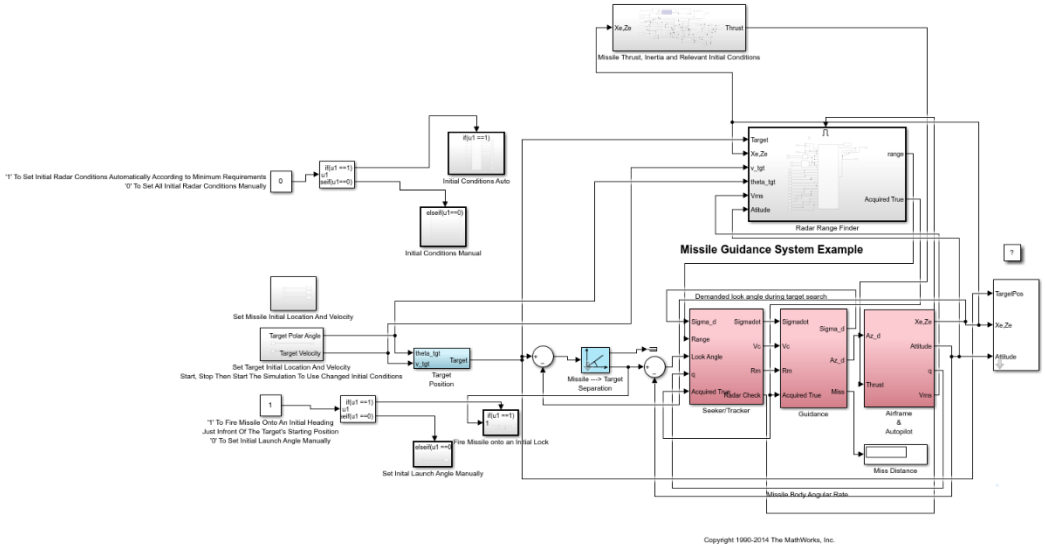


Figure 22 - Overview of the third development stage of the project

From figure 22 it can be seen that some relevant initial conditions were added as subsystems and the missile launch angle was split into the two subsystems as discussed just above. Also, a subsystem for dynamic missile thrust, inertia, mass was added which includes some required initial conditions to calculate these values inside of it.

The difference between this and the final iteration of the project seen in figure 13 is minor as just more initial conditions are added to the system and the new dynamic missile subsystem was tidied up to make it more digestible and as such the third and final stages are joined together as one section in this report. More information on the development stages is available inside the logbook on appendix A.

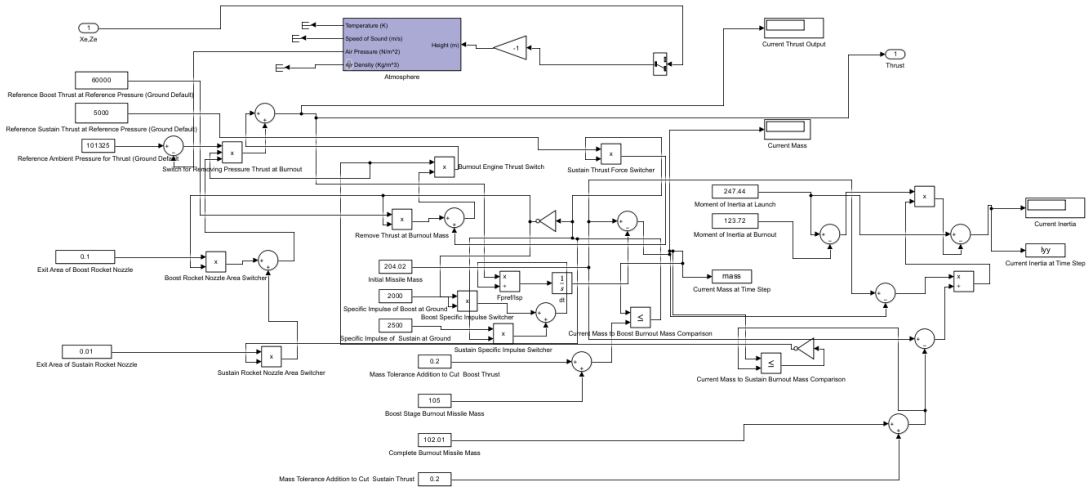


Figure 23 - The dynamic missile subsystem in the final iteration of the project

The subsystem in figure 23 models the change in thrust as altitude increase and therefore pressure changes, it models the change in mass as the missile expends the fuel for its motor and it models the change in the moment of inertia of the missile (the manoeuvrability of the missile) as it loses mass.

It also allows the modelling of 2 rocket motors as outlined in 'Missile Flight Simulation' [2] in which there is a powerful first motor with high thrust that gets the missile up to high velocity and a second much weaker motor with low thrust that activates after the first motor burns out which is used to allow the missile to maintain its high velocity. Whilst this is the more common configuration of a missile, you can use a simple boost glide system with 1 motor in which once the first motor burns out, the missile glides to the target with no thrust. This is done by simply setting the boost stage burnout mass and the complete burnout mass as equal to each other.

The thrust is modelled by the following equation[2]:

$$Fp = Fpref + (Pref - Pa)Ae$$

Where Fp is equal to total thrust, $Fpref$ is equal to the thrust at the reference pressure (ground pressure in this case), $Pref$ is equal to the reference pressure (ground pressure in this case), Pa is equal to the pressure at the current altitude of the missile and Ae is equal to the exit area of the rocket nozzle.

The mass is modelled by the following equation[2]:

$$m(t + \Delta t) = m(t) - Fpref * \Delta t/Isp$$

Where $m(t + \Delta t)$ is equal to the mass in the next time step, $m(t)$ is equal to the mass at the current time step, $Fpref$ is equal to the reference thrust (this is the total thrust from the previous equation) and Isp is equal to the specific impulse (this is the amount of thrust provided by the rocket motor for every unit of fuel consumption). The source uses an addition

rather than a minus which would just increase the mass over time which makes little sense fro this project so a minus is used and the terms after the minus are a calculation of how many kilograms of fuel have been burned in a time step.

The moment of inertia is modelled by the following equation[2]:

$$I = (Io - Ibo)(\frac{mo - m}{mo - mbo})$$

Where I is equal to the moment of inertia of the missile, Io is equal to the moment of inertia of the missile at launch, Ibo is equal to the moment of inertia at burnout, m is equal to the current missile mass, mo is the missile mass at launch and mbo is the missile mass at burnout. As the missile loses mass, it becomes more manoeuvrable which means it requires less force to change its direction. The source adds a multiplication of Io at the beginning (just in front of the bracket) but this just produces a wrong result and doesn't seem to follow the physics of this calculation where we assume location of the centre of mass varies linearly with time.

The switching between the two rocket motors is controlled by comparing the current mass with the input burnout mass for that stage. The mass tolerance for that stage is added to the input burnout mass when comparing the values where the result is either 1 or 0. This value is then put through the various switchers which multiply the given value by 1 to keep it on or 0 to turn it off. These are labelled on figure 23. The value for the air pressure at the current height is brought in from standard atmosphere model inside SIMULINK. The current thrust that is calculated is outputted into the overall system and the moment of inertia and mass is stored inside a variable that was already in use by the original example to calculate the physics of the missile.

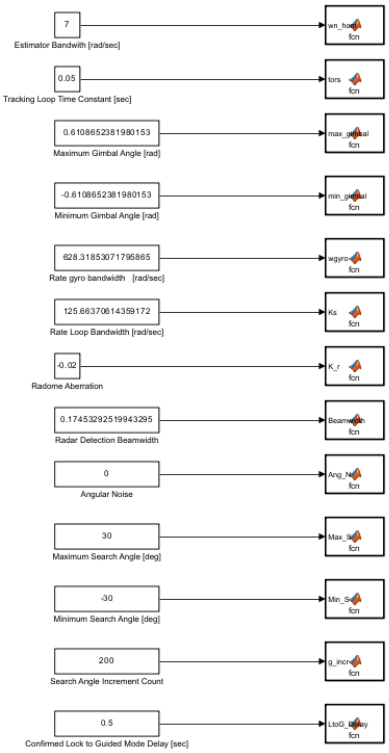


Figure 24 - RADAR homing dish properties subsystem of the final iteration of the project

Figure 24 shows the subsystem that allows the user to change initial conditions pertaining to the properties of the RADAR homing dish inside the system. The modifications that were required to allows some of these initial conditions to be changed can be seen in the guidance processor of figure 25 where variables were added and the blind flying distance (where the missile is considering too close for RADAR range readings so the missile switches to flying blindly straight) was changed from a very high 200 metres to a more reasonable 10 metres.

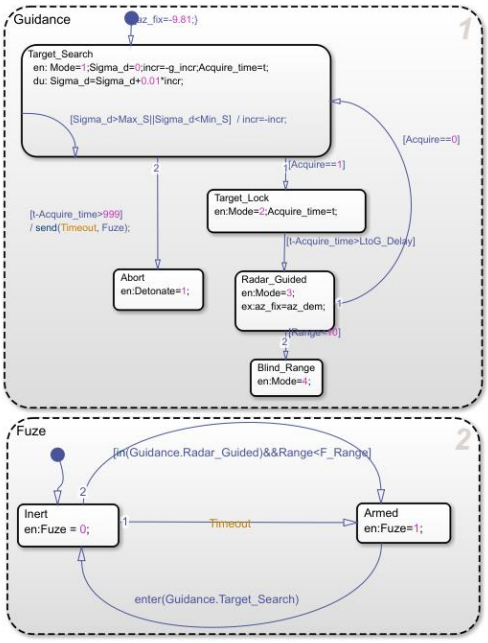


Figure 25 - Guidance processor of the final iteration of the project

Modelling the guidance system of a RADAR missile attacking a linearly moving target using

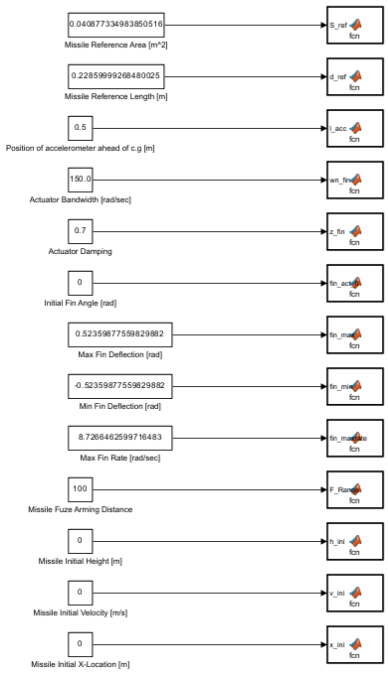


Figure 26 - Missile properties and initial conditions subsystem of the final iteration of the project

Figure 26 shows the subsystem that allows the user to change initial conditions of the missile as well as some of its properties. The model provided by MATLAB uses lookup tables to model aerodynamics and its values seem to only go up to 5.5 Mach speed after which the model breaks down and the missile moves erratically. Real life anti-air missiles are just about capped at this speed so this was not changed and it is outside the scope of the aims of this project. The user needs to make sure the missile does not go past this velocity.

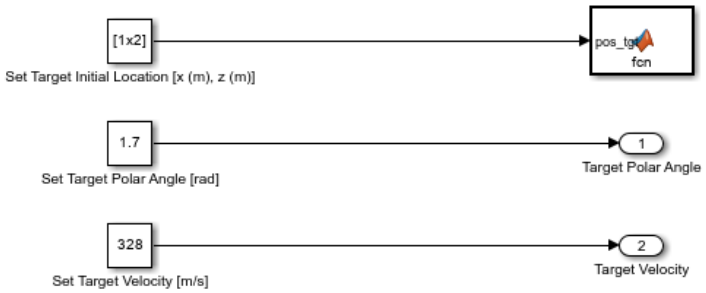


Figure 27 - Set target initial location and velocity subsystem of the final iteration of the project

Figure 27 shows the subsystem that allows the user to change the initial location, velocity and polar angle of the target. The usage of a MATLAB function to store these values in the workspace means the user has to start, stop and then start the simulation to apply new initial conditions. There isn't any way around this and after much research in trying to find a way, I decided to just mention that slight blocker rather than spend more of the limited time of the project trying to solve a quality-of-life issue that has no bearing on the outcome of the project.

5. Results and analysis

5.1 Initial conditions

These results were obtained by running the project with mostly default initial conditions that were already embedded inside the SIMULINK example and the MATLAB radar code. Their values were copied as the default for their respective initial condition variable. The manual RADAR conditions were used along with the launch of the missile just ahead of the target. These values can almost all be seen by the preceding figures. There are far too many to test out every single initial condition separately for its own effect on the results.

All the values inside the 'missile thrust, inertia and relevant initial conditions' subsystem were made up by looking at sensible ranges for each and using values that would provide the most variability in the results. In addition, the values were chosen to be in the realm of reality as well as to allow the missile to use both rocket motors and accelerate to a sensible speed. The missile itself was set to start at the 0,0 position with no initial velocity and the target was placed slightly further away on the horizontal axis at 4500 metres but the same distance on the vertical axis at -3547.99. The target was also set to go in an upward angle whilst going towards the missile to provide a harder to hit target.

5.2 Results and analysis of normal acceleration and acceleration demands

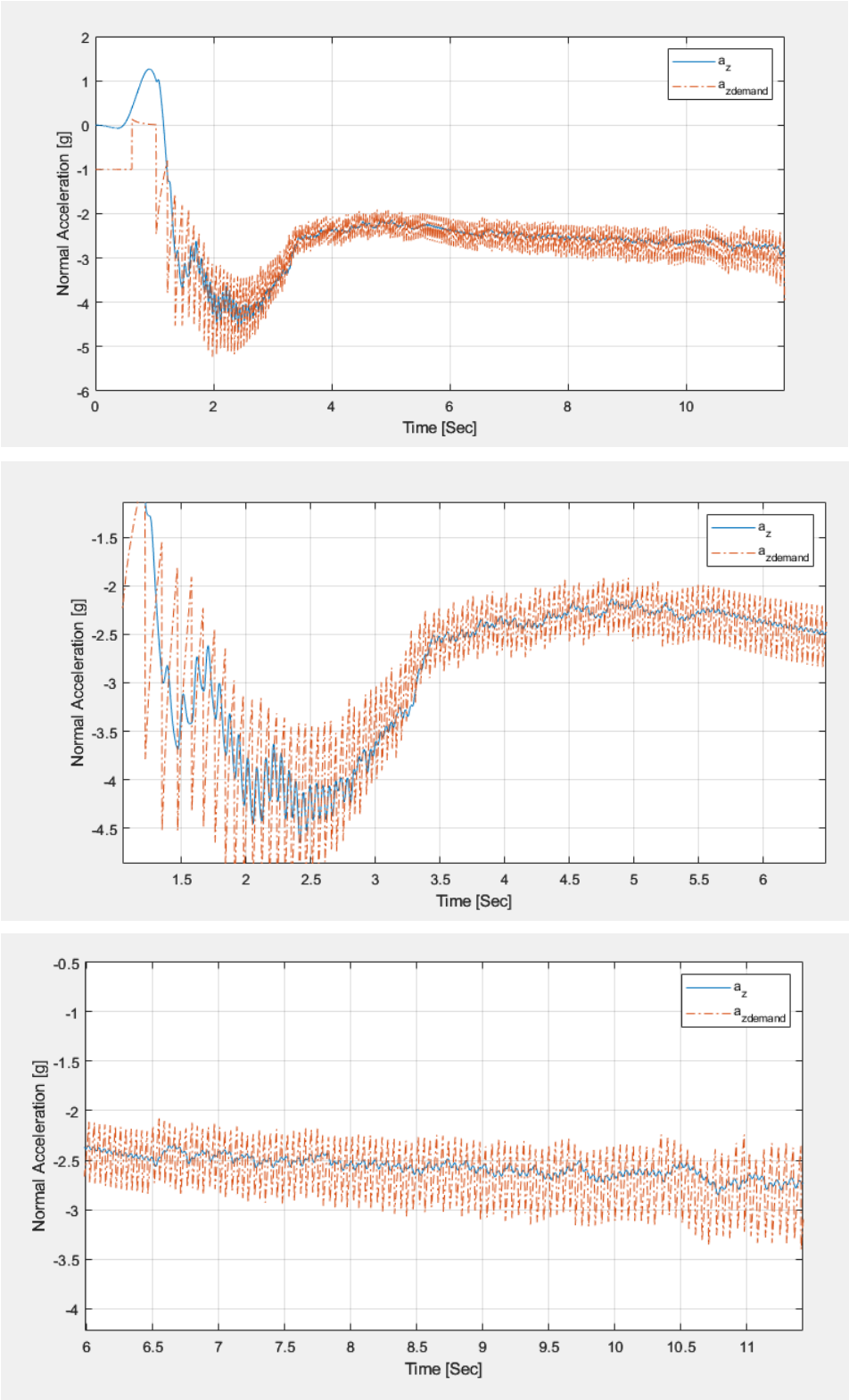


Figure 28 - Result graph for normal acceleration and acceleration demand with zoomed in sections

Looking at figure 28 we can see how the additions to this project have led to a more realistic acceleration profile for the missile. The demand is constantly going above and below the actual acceleration as the RADAR system constantly pings the target and receives a range reading with limited precision. The actual acceleration is no longer a smooth line but instead a bumpy road as it constantly adjusts for the varying demand. The missile experiences a sort of simple harmonic motion as it gets its initial lock which then dampens out as the missile stabilises over time onto a direct course for the target.

5.3 Results and analysis of fin demands

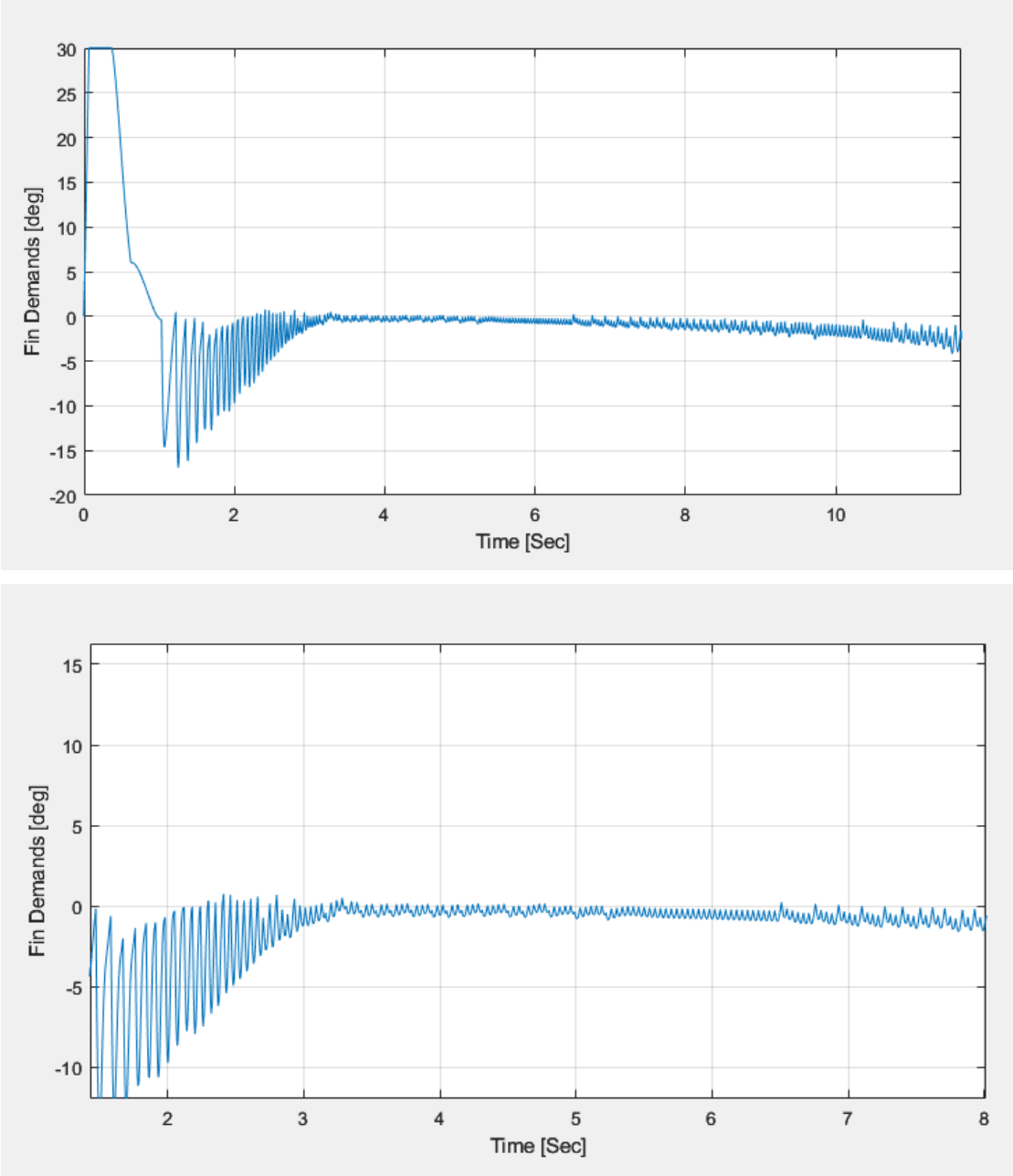


Figure 29 - Result graph for fin demands with a zoomed in section

Modelling the guidance system of a RADAR missile attacking a linearly moving target using

From figure 29 we can see the same result as figure 28 but it is represented as fin demands. A system that constantly has to adjust for the new non certain input. This is what a real profile of a missile would look like generally.

5.4 Results and analysis of incidence angle between the missile and the target

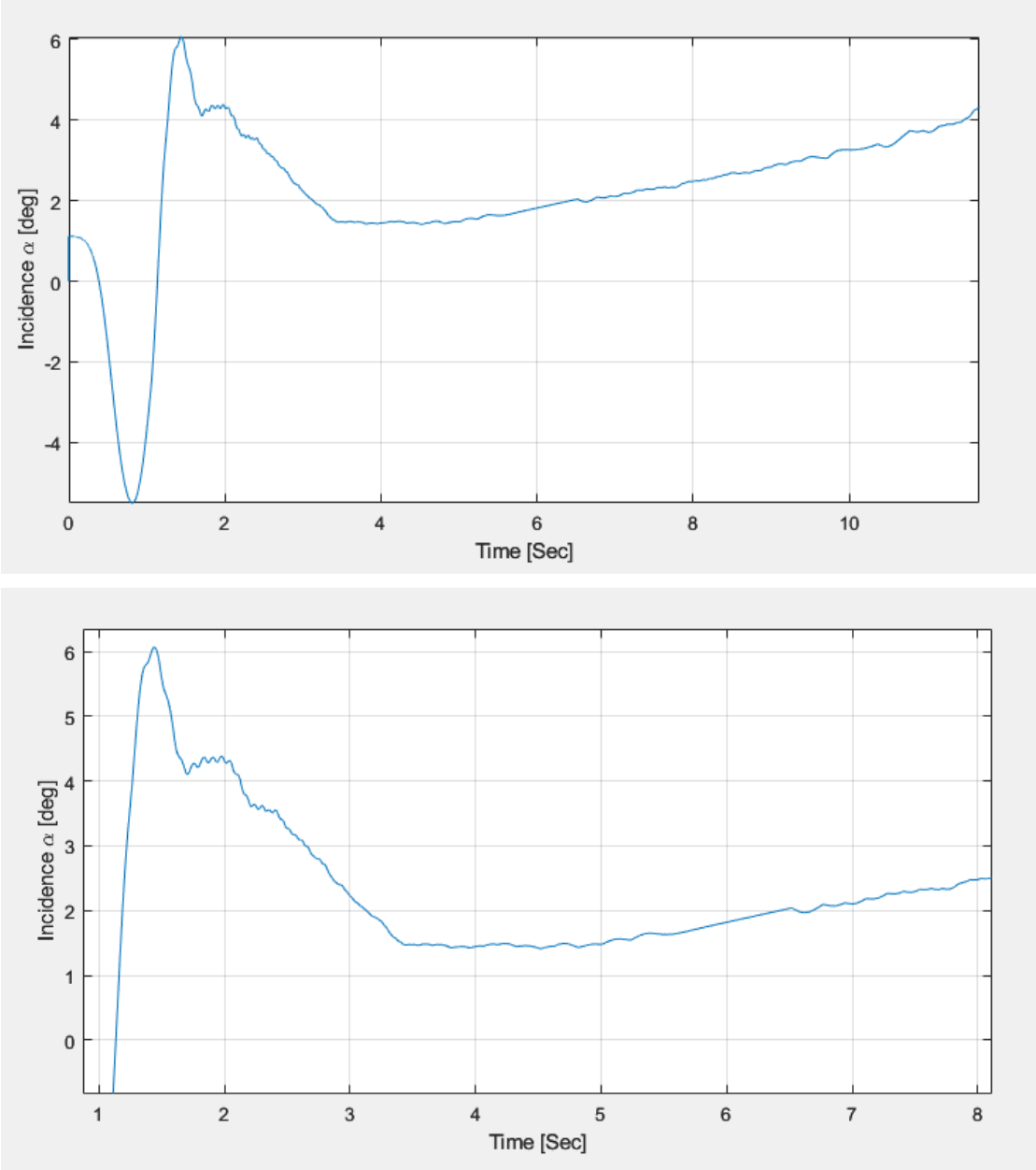


Figure 30 - Result graph for the incidence angle between the missile and the target along with a zoomed in section

Figure 30 is a representation of how the missile flew towards the target. There are the traces of the bumps from the previous results however, it is much smoother. This shows that the irregularities from the previous results are okay as the missile itself doesn't fly in a manner of oscillation but rather a realistic setting of some smooth sections and many minor corrections that do not detach it aerodynamically from airflow or cause significant drag. You can estimate this line as a smooth line which is the case with the original example as it is simpler.

Once again, this too is the kind of profile that you would expect to see from a real missile and it is a result of the modifications that were performed by this project.

5.5 Results and analysis of the Mach number profile of the missile

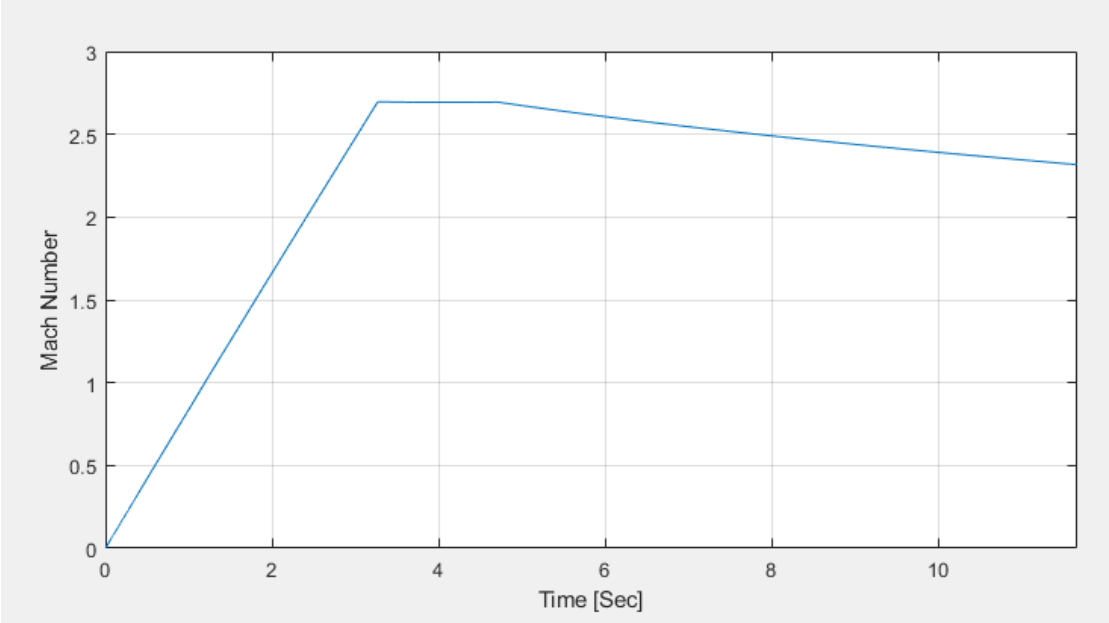


Figure 31 – Result graph of the Mach number profile (velocity)

From figure 31 we can see there is no aerodynamic drag curve from the missile turning to head towards the target. This is because it is started by being launched just ahead of the target angle as an initial lock, this is how anti-air missiles are used in real life, and we can see here why as the missile can achieve much higher speeds as it does not have to waste energy on doing a very high-speed turn.

The profile shown here matches perfectly with the boost sustain style missile profile shown in 'Missile Flight Simulation' [2]. The boost stage takes the missile up to high speed and then the sustain stage ensures that the velocity of the missile declines at a much lower rate and it is able to keep near its top boost velocity for the duration of the simulation. Also, a sustain motor can help manoeuvrability as when the fins cause a change in direction of the missile, the

thrust from the motor will force the missile onto that new course at a much faster rate that is also more stable. It operates as an extra control authority. Therefore, this is once again the kind of profile one would expect to see from a real missile.

5.6 Results and analysis of the missile and target trajectories

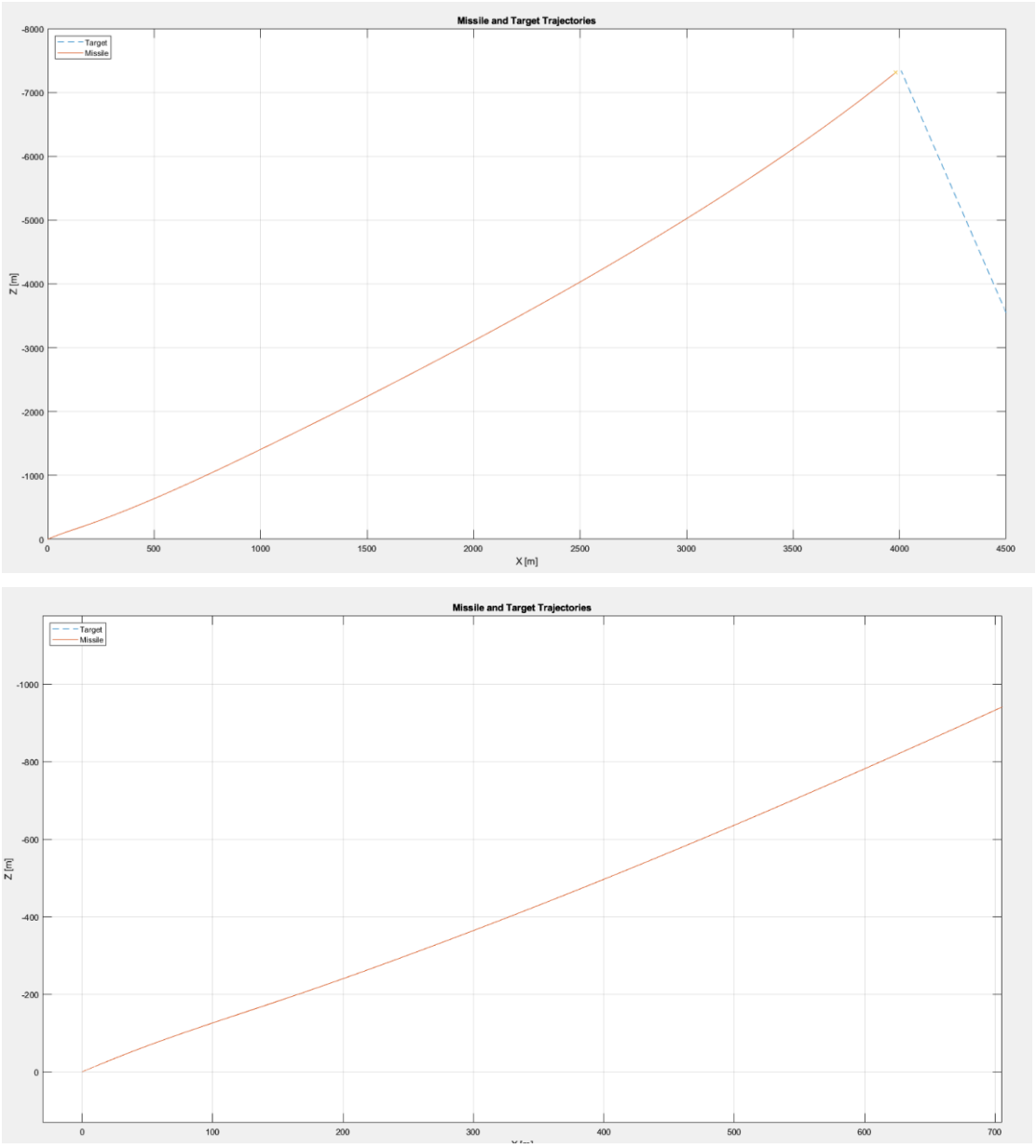


Figure 32 - Result graph for the missile and target trajectories of the missile and the target with a zoomed in section

The height is shown as negative in figure 32 due to the fact that negative height is considered as the upwards direction in MATLAB. As mentioned before here we can see that the oscillating fin and accelerating demands did not have a significant negative effect on the missile, they are all mostly smoothed out.

Corresponding to figure 30 we see there is a mostly smooth curve with hard to notice tiny corrections. It is mostly visible at the start when the missile is trying to head directly at the predicted path of the target after locking on. This could again be approximated as a smooth line as is the case with the original example. This new curve that is not perfectly direct is once again the exact kind of profile one would expect to see from a real missile.

5.7 Results and analysis of the gimbal, true look angle and mode changes of the missile

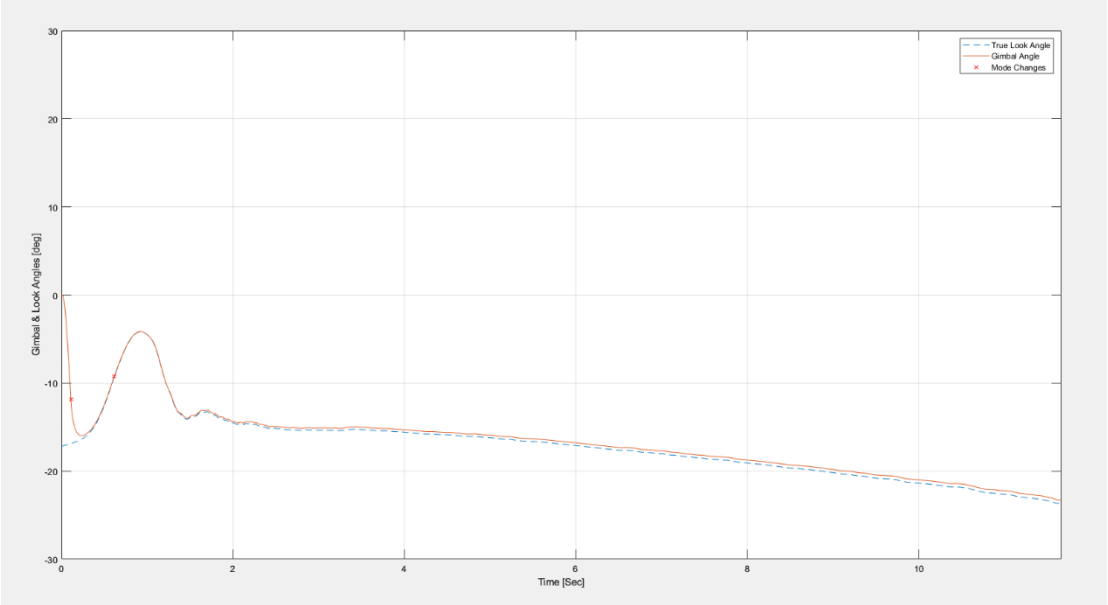


Figure 33 - Result graph of the gimbal, true look angle and mode changes of the missile

In figure 33 we can see the same kind of mode changes as the original where the missile confirms a lock and then gives itself time to calculate the trajectory of the target as well as point the RADAR head directly at the target.

However, we can see the exact kind of change as discussed before, where the tracking head is not able to keep perfectly on target as it has to adjust for the minor correctional movements of the missile. This could be approximated as a perfect smooth line as is the case in the original example. This is the exact kind of realistic resultant profile that was wanted in the aim of this project.

If the system loses the lock on the target, then it is capable of reasserting its lock as it sweeps for the target in its search angle. The original system was capable of this but now the lock is reliant on there being a valid RADAR range reading as well as the target being within the beamwidth.

5.8 System's time to run and an additional animation figure

The simulation runs much faster when it is searching for the target as the RADAR subsystem and code takes the most amount of time, around 0.03 seconds but runs every 0.01 seconds when there is lock and this discrepancy causes the slowdown. Nevertheless, the system takes about 35 or so seconds to run its 12 second simulation time. This factor of about three makes sense. The run time of the RADAR code can be reduced significantly by reducing the number of pulses that is used to integrate the RADAR signal (about 0.018 seconds when running with 2 pulses) but this will mean that you will need a much more powerful RADAR system as the detection of the target signal echo will be much harder.

The system also runs the animation figure that was included in the original example which shows a red painted missile flying towards the target in a 3-dimensional space. It can't be shown here but the longer run time of the new system means you can comfortably watch the missile fly a mostly smooth path with minor corrections towards the target. In the original example the simulation ran so fast that the animation figure would last about a second before it ended.

6. Conclusion

6.1 Summary of the result and achievement of the project aim

Overall, this project has achieved its stated aim and produced realistic results. It is a simulation that works whilst also providing all the relevant initial conditions in an easy to change manner with the meaning of their variable names stated. The system is 2-dimensional whilst also including simplifications but considering the limited time of this project and the fact that it required deep extensive research on the subject matter, this is perfectly adequate. The changes that were performed greatly deepened my knowledge and understanding of MATLAB, SIMULINK, RADAR simulation and missile guidance simulation.

The SIMULINK system and this report has been done in such a way as to allow other beginners to learn, understand and have a starting point for this subject field.

6.2 Project management review

From appendix A the old version of the Gantt chart used for this project can be seen and the last version of the Gantt chart used for this project can be seen in figures 1-4. Following feedback from the supervisor the Gantt chart was moved to a Microsoft project where features unique to the program were learnt and added to the Gantt chart. There were corrections of minor mistakes and the expansion of defined aims and tasks. Also, some missing tasks were added. Deadlines for tasks were moved to be more reasonable as recommended by the supervisor. Overall, this led to a more comprehensive Gantt chart with much more information, that is easier to modify with better presentation.

The tasks and their deadlines were followed quite well for the project and development completely finished on time whilst meeting the project aim. There was a delay in the write up of the report due to some personal circumstances but considering this is the last section of the Gantt chart and is only concerned with the write up of the project, the Gantt chart still fulfilled its purpose.

The Gantt chart was written before finding the existing MATHWORKS guidance system example and MATLAB RADAR framework and as a result some of the specific objectives were already completed upon doing so like target tracking. However, the Gantt chart was made with flexibility in mind and as such this was not a major issue for the development of the project. The supervisor and the weekly meetings served a great deal of help in deciding where the project was heading at that point in time and what actions needed to be taken next. The practical part of the project went perfectly to plan.

6.3 Recommendations for future improvements and limitations of the project

For future improvements this simulation can be moved to the third dimension. This not only presents the challenge of the new dimension but it adds the need to simulate clutter (terrain features) and add a system to filter out this new signal. These kinds of new very advanced additions were mentioned in the literature review section with advanced adaptive RADAR tracking and shape representation of target objects that is factored into the received signal, rather than treating the target as a point target. The RADAR MATLAB toolbox that was used in this project includes some necessities for the new dimension with clutter simulation[8], terrain simulation[9] and RADAR that searches for the target in the third dimension[10].

It should also be mentioned that MATLAB assumes perfect conditions with no wind, no material degradation, no fuel degradation, no simulation of temperature, the standard

atmosphere model and no part failures (like the motor failing to start up). Also, the rocket motors do not include their very short spool up times and the motors instead operate at 100% thrust immediately. As mentioned, there is no clutter simulation so the missile is likely to hit a terrain object if this system were translated into the 3-dimensions and used with no modifications.

Also, the RADAR system used here is monostatic. A future improvement could be a bistatic system where one RADAR systems send the signal and a separate RADAR system receives the signal which leads to a more accurate range reading but naturally is a more complex system.

The system is also non-coherent which means the properties of the RADAR wave that is used is treated as being unknown. A future improvement could be a coherent system that factors in known properties of the wave to decode the received signal echo. The system uses a rectangular waveform but MATHWORKS mentions that a chirp waveform could be used to improve results[3].

Lastly the target moves in a linear line with no changes in its velocity. A future improvement could include a full aerodynamic simulation of the target with its own propulsion and the ability of the target to detect that it is being locked onto by RADAR and therefore deploy counter measures and evasive manoeuvres.

6 REFERENCES

[1]“Designing a Guidance System in MATLAB and Simulink - MATLAB & Simulink - MathWorks United Kingdom.” https://uk.mathworks.com/help/releases/R2020b/simulink/slref/designing-a-guidance-system-in-matlab-and-simulink.html (accessed Nov. 02, 2022).

[2]J. Strickland, *Missile Flight Simulation*. 2015. Accessed: Oct. 21, 2022. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=VhD8CgAAQBAJ&oi=fnd&pg=PR17&dq=missile+AND+simulation&ots=FlwQmIrkSf&sig=XbyTOkznYTuLsYz6H5WKcSzUINk

[3]“Simulating Test Signals for a Radar Receiver - MATLAB & Simulink - MathWorks United Kingdom.” https://uk.mathworks.com/help/phased/ug/designing-a-basic-monostatic-pulse-radar.html (accessed Nov. 02, 2022).

[4]M. S.-R. handbook and undefined 1962, “Introduction to radar,” *helitavia.com*, Accessed: Oct. 13, 2022. [Online]. Available: https://helitavia.com/skolnik/Skolnik_chapter_1.pdf

[5]M. Bühren, B. Y.-Proc. Intern. W. on Intelligent, and undefined 2006, “Automotive radar target list simulation based on reflection center representation of objects,” *iss.uni-stuttgart.de*, Accessed: Oct. 14, 2022. [Online]. Available: https://www.iss.uni-stuttgart.de/forschung/publikationen/buehren_wit2006.pdf

[6]J. T. Johnson *et al.*, “Cognitive radar for target tracking using a software defined radar system,” *ieeexplore.ieee.org*, 2015, doi: 10.1109/RADAR.2015.7131213.

[7]W. Hao and H. Z. Zhao, “Modeling and simulation of a full coherent LFM pulse radar system based on Simulink,” *Proceedings of 2013 2nd International Conference on Measurement, Information and Control, ICMIC 2013*, vol. 1, pp. 495–499, 2013, doi: 10.1109/MIC.2013.6758012.

[8] “Introduction to Radar Scenario Clutter Simulation - MATLAB & Simulink - MathWorks United Kingdom.” https://uk.mathworks.com/help/radar/ug/introduction-to-radar-scenario-clutter-simulation.html (accessed Nov. 02, 2022).

[9]“Radar Performance Analysis over Terrain - MATLAB & Simulink - MathWorks United Kingdom.” https://uk.mathworks.com/help/radar/ug/radar-performance-analysis-over-terrain.html (accessed Nov. 02, 2022).

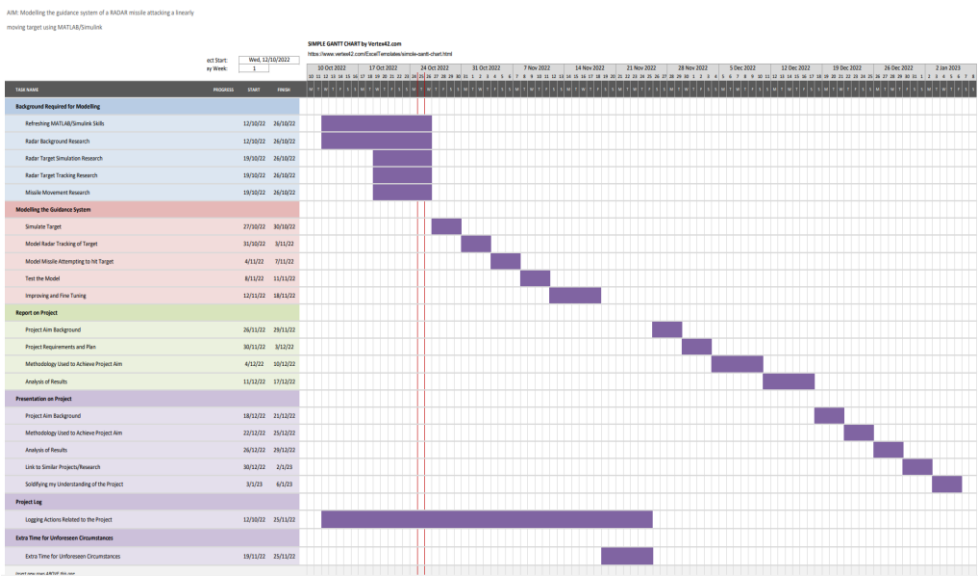
[10]“Radar Scenario Tutorial - MATLAB & Simulink - MathWorks United Kingdom.” https://uk.mathworks.com/help/radar/ug/radar-scenario-tutorial.html (accessed Nov. 02, 2022).

7 APPENDICES

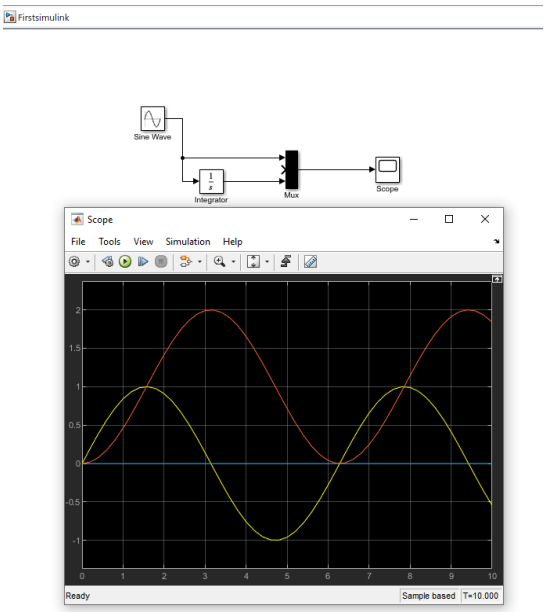
APPENDIX A: LOGBOOK

Name:	Akin Kucukkurt
Project title:	AIM: Modelling the guidance system of a RADAR missile attacking a linearly moving target using MATLAB/SIMULINK
Supervisor:	Jacob Morewood

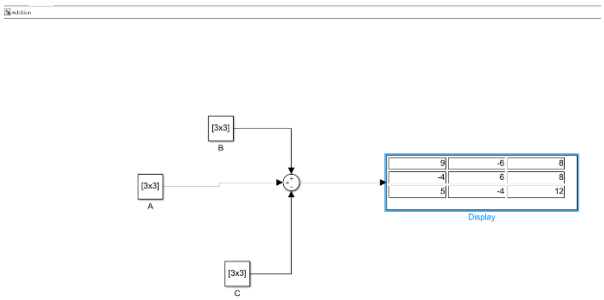
Student notes & date:	
Planned tasks and Work Undertaken:	First Msc project online seminar attended- Introduction for the module Second Msc project online seminar attended- Info given on the online library, academic support available, joined skillup module, installed and got Mendeley working. First Supervisor Meeting- Project intro, what is expected of me and requirements for a Gantt chart discussed, written notes as follows, “40 hours a week Aim for next week Objectives- Stepping stones to complete Aim Specific Measurable Achievable Realistic Time constrained Break down objective into Tasks Objective- Research Task- What are you researching Draft Gantt chart next week day by day Background research of some form Document what you do Logbook” Draft Gantt Chart made-



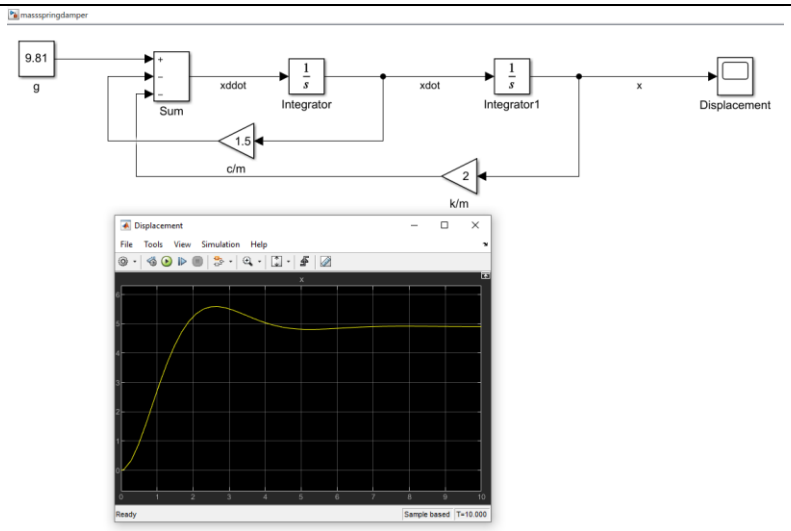
Some Simulink skills refreshed- basic integration



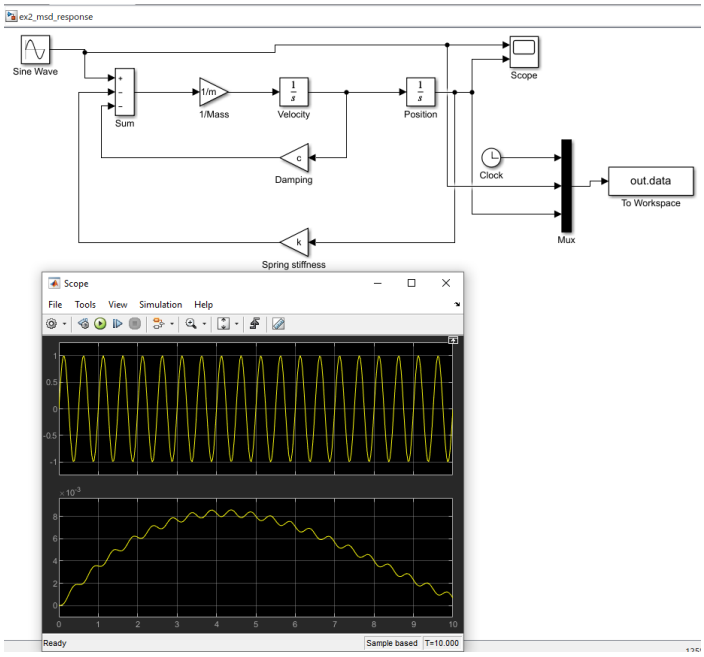
Basic addition



Mass spring damper system



Msd response



Research on basics of radar and radar target simulation-

Introdution of radar with a block diagram of the system, equations of radar, what information can be taken from a radar signal (Range, Radial Velocity, Angular Direciton, Size, Shape, other target measurements, various radar frequencies explained and radar nomenclature explained[4]

A system is shown for simulated data of radar targeting a car with images showing where the radar hits the car, which areas are visible, what the signal will look like with a comparison of simulated data versus actual data and their differences, resolution, visibility, antenna patterns, simulation structure and some information on tracking.[5]

	Third Msc project online seminar attended- Academic misconduct and plagiarism, clearly defined what is plagiarism, must reference and put into own academic terms. Consequences and how plagiarism reports, cases are handled shown. Second Supervisor Meeting- Draft Excel Gantt chart discussed, written notes as follows, “Be more specific on tasks, eg (Model missile- define test condition) Subheadings fine, generally fine Somethings are at the same time, somethings require other tasks first, put in order requirements. Investigate Microsoft Project as suggested by supervisor. Add some screenshots to logbook Leave space at the end of Gantt chart for leeway. Add a task for interim report. Could use Harvard or numeric based reference system” Research on radar target tracking and missile flight simulation- A software defined radar system is shown with the various equation governing that system stated and explained with a 3d diagram example, various detailed graphs and diagrams showing the data and signals processed by the system, range vs time with doppler effect, range track, velocity track, velocity standard deviation etc[6]. A book detailing various simulation diagrams of a missile flight simulation, diagram of systems inside a missile and their explanations, Infrared tracking is shown, reticule optical tracking, missile scanning pattern, radar tracking with various types of radar like pulse radar, continuous wave radar, pulse doppler radar and angle tracking methods, autopilot system shown, control system explained (lateral acceleration, canard control, tail control , wing control etc), warhead and fuse explained, propulsion systems explained , other more complicated systems that are outside the scope of the project as well as information on how to improve/optimise the system and finally the system of equations governing the missile is shown and explained [2]. Microsoft Project Import- Spent a significant amount of time importing the excel Gantt chart to Microsoft Project and figuring out how project works and adding improvements as mentioned by supervisor. Fourth Msc project online seminar attended- How to make a good project plan discussed, lots of repeated points already given to me by supervisor. Excel gantt chart shown but I’d already made mine and imported it to Microsoft project.

Copied out old handwritten log notes and logged unlogged activity to word logbook- 25/10/2022

Meeting-
Extend the project write up dead line- move to December and maybe put presentation at the end of December, extend the previous bits to cover the gap.
Edit the Gantt chart as things come up.
Write up about the Gantt chart -why, have you accounted for slipping and how for report. Mention previous Gantt chart. Microsoft project has more functions, mention this in report. Aims objectives, why you should be doing this project. Report on Simulink training. Write up the research for the report. For next week write up some of this stuff for the report.

30 credit is the same deadlines just a bit less in every way.

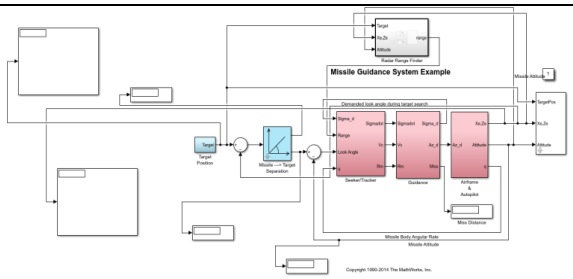
Implemented some Gantt chart improvements as discussed, the project now finishes at a more reasonable January first , if everything goes to plan.
26/10/2022

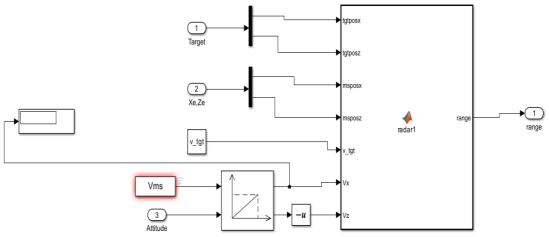
A lot of time spent trying to interpret to what seemed a bang on and useful piece of research titled “Modeling and simulation of a full coherent LFM pulse radar system based on Simulink” [7]. On initial reading it seemed plausible in laying down the blueprint for the modelling of a radar system in Simulink and such a purpose is stated in the research but, what is given in the paper is a overview of the system, and individual blocks are not explained well enough and the necessary information is not given that would allow one to recreate and run the system shown. This ends up with the research being qualitative and not what I needed which was quantative. As I didn’t understand and know the technical aspects of radar shown, I didn’t realise that fact. The worst part of not knowing something is not knowing how much you don’t know. 29/10/2022

Fifth Msc project online seminar attended-
Research means to collect, analyze and interpret information in order to increase understanding of project. Literature review means to evaluate material relevant to the subject to demonstrate subject knowledge and the project/authors position in the domain. It is not about summarising what is in the research, its always about making some sort of argument. Some help shown in how to search for research

	and what to evaluate when looking at research shown. Types of research shown and explained. 1/11/2022 Found that Simulink provides a radar tracking a linear moving plane model as well as a virtually complete model of a missile homing onto and attacking a target with radar. Whilst the missile model has good depth, and is adequately accurate enough, the radar model is extremely simplistic, and is extremely qualitative of the issue rather than quantative. Spent a lot time looking at both models and evaluating them to the research. The missing depth of the radar model I found in matlab documentation, which explains and show a proper model of radar in full complexity with a proper radar signal, proper clutter simulation proper techniques in how that signal is interpreted." Simulating Test Signals for a Radar Receiver" , "Introduction to Radar Scenario Clutter Simulation" ," Radar Scenario Tutorial" etc. These are the things that were not fully quantitatively shown in [7] Whilst this is matlab, it can certainly be ported into Simulink with blocks as you can run the matlab code as a function. I was able to understand how matlab integrates into simulink by looking at the first mentioned radar and missile model. It certainly is not very straight forward or intuitive. Matlab data generally has to processed and shown with matlab functions. Whilst I have matlab skills, when doing programming before the model that required it last year, I was taught Python, which I am intimately familiar with, even creating a machine learning algorithm to rate images of the sky for how much cloud coverage there is so that you can automatically tell whether the observatory telescopes will be useful on any given day for my Bsc project. Whilst matlab is similar to python, there are differences and it is not easy for me to fully understand matlab code as I am hardwired for python and its rules. It is quite hard to generate a matlab signal from a matlab function and simply tie it directly to a Simulink scope for example, Simulink doesn't accept things like matlab arrays and data types. Simulink has its usefulness in being a block diagram of how the matlab code works. The makes it much easier to debug and understand the code. As simply walls of code, even with comments is quite a challenge or someone to understand and interpret whereas if it is implemented into Simulink in blocks, then it is at least very easy to achieve a qualitative understanding what your code is and what it does.

	Now I have all the information necessary to actually make the model which will involve using the matlab documentation to create a Simulink radar system and then making changes to the existing Simulink missile model and implementing the created full radar to create the full model. 2/11/2022 Spent the remaining time till deadline writing up the report 7/11/2022 Analysing Simulating Test Signals for a Radar Receiver, a non-coherent detection scheme means we are assuming the wavelength echo changes signal properties each time (different frequency, wavelength and phase) , if otherwise we would use wave information in processing the signal which would be more computationally expensive. Since non-coherent is the worser case, assumable if the system works in non-coherent it can work in coherent as well so its not a big deal. Monostatic radar system meaning its one radar detector, if you had two towers at comparable length to the target range you could have a better bistatic system. “Simulating Test Signals for a Radar Receiver” will give me the range information Spent time understanding how the matlab code works, and trying to implement into Simulink. Achieved some basic implementation that doesn’t take into account target/sensor velocities. 09/11/2022 Sixth Msc project online seminar attended- Information, examples and requirements of references shown. Make sure to caption figures and mention them. Literature review in your own words. 15/11/2022 Achieved Successful implementation of radar function to find range values. Uses target/missile location, velocities and angular difference to simulate the radar. It is run as an extrinsic function with declared variable size. This is how you run matlab code in a matlab framework in Simulink.






```
function range  = radar1(tgtposx,tgtposz,msposx,msposz,v_tgt,Vx,Vz)

coder.extrinsic("radarfunc");


range=ones(1,1);
range=radarfunc(tgtposx,tgtposz,msposx,msposz,v_tgt,Vx,Vz);

end
```

Discussed this progress in meeting.

Meeting Notes:

Advances vs current system

Maybe find actual values for radar parameters

Screencap and the visual animation for the presentation

Is it becoming more realistic

If matlab updates their model maybe you can compare.

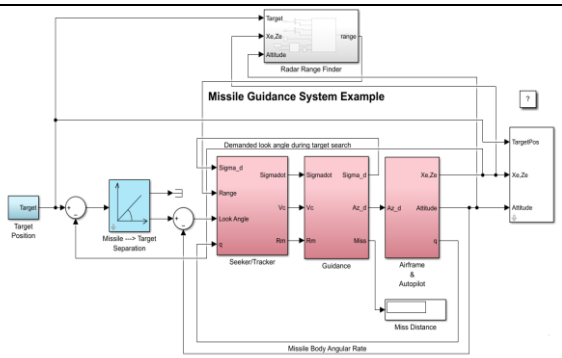
Running it as a separate matlab function block is fine.

Bistatic is a development that someone could do in the future, not in the scope of the project

See some of it written up.

17/11/2022

Wrote up some of the methodology of the report for the progress thus far. Tidied up the simulation a bit.



Discussed with Jacob Morewood and sent email with videos of the missile animation with various conditions.

Notes:

Methodology, limitations, what will you do, what is the aim

Next section what you did

Next Results

Next Analysis

Document like the person has no experience/knowledge of the field

Can you improve the constant tiny corrections (in hindsight this is just how the system should work, as the system constantly has to account for the range reading which only updates about every 25 metres and since the missile flies smoothly overall, there is no issue here.)

Contact the lrc about the borrowing of J. Strickland, *Missile Flight Simulation*. (done and book received.)

Do more write up.

Present in a clear manner.

Explain Graph and mode changes.

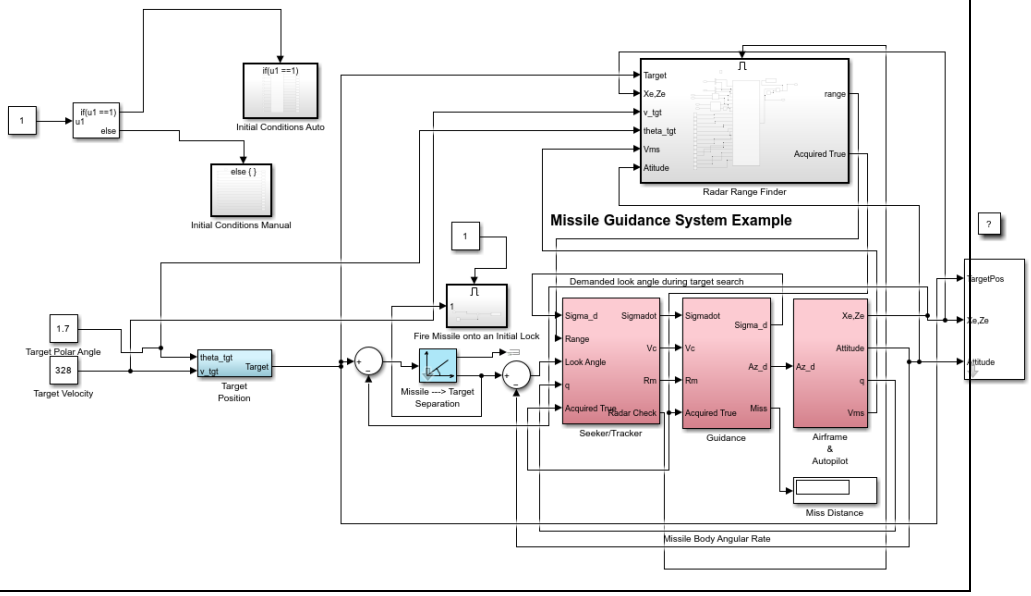
23/11/2022

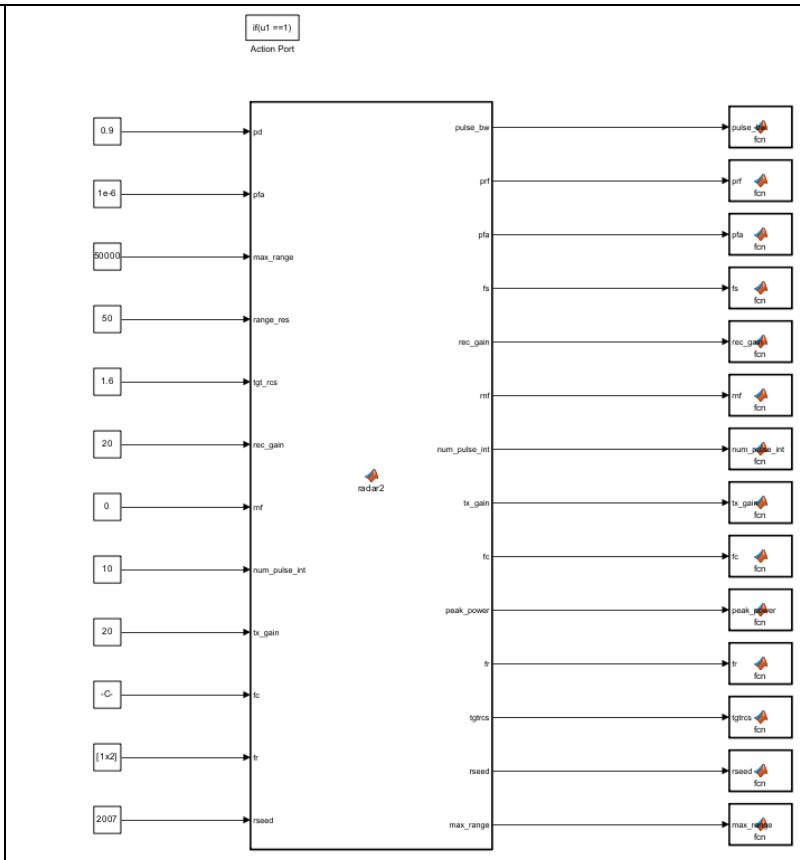
Focused entirely on simulation development.

Used the values straight from simulation as an input to the function block for the aforementioned radar parameters rather than storing them into the workspace and loading them from the workspace.

Added all initial conditions for radar including radar power, max range, target cross section etc. Can choose between defining every parameter or defining ideal parameters which then calculates the minimum strength the radar would have to be to get a reading and uses that. This is stored as a workspace variable through matlab function code as Simulink workspace variable function doesn't accept duplicates and it can't be fed directly into the radar function like the other variables

can as again there are two of them and you can't have two inputs of the same name into the function block even if only one of them is going to active at a time.
Missile is fired onto an initial lock (just starts the missile facing ahead on the trajectory of the target)
The system will now check if the target is within the radar beamwidth and if there is a valid range reading, otherwise will continue searching as opposed to the old system where if the target was within the beamwidth, then there was a lock which is obviously not as realistic.
Had to modify the matlab radar function provided by matlab to accept the ability to change all the parameters and accept not having a valid range reading (ie the radar has failed to acquire the target) by outputting -1 which is used as the check for if the radar has detected anything.

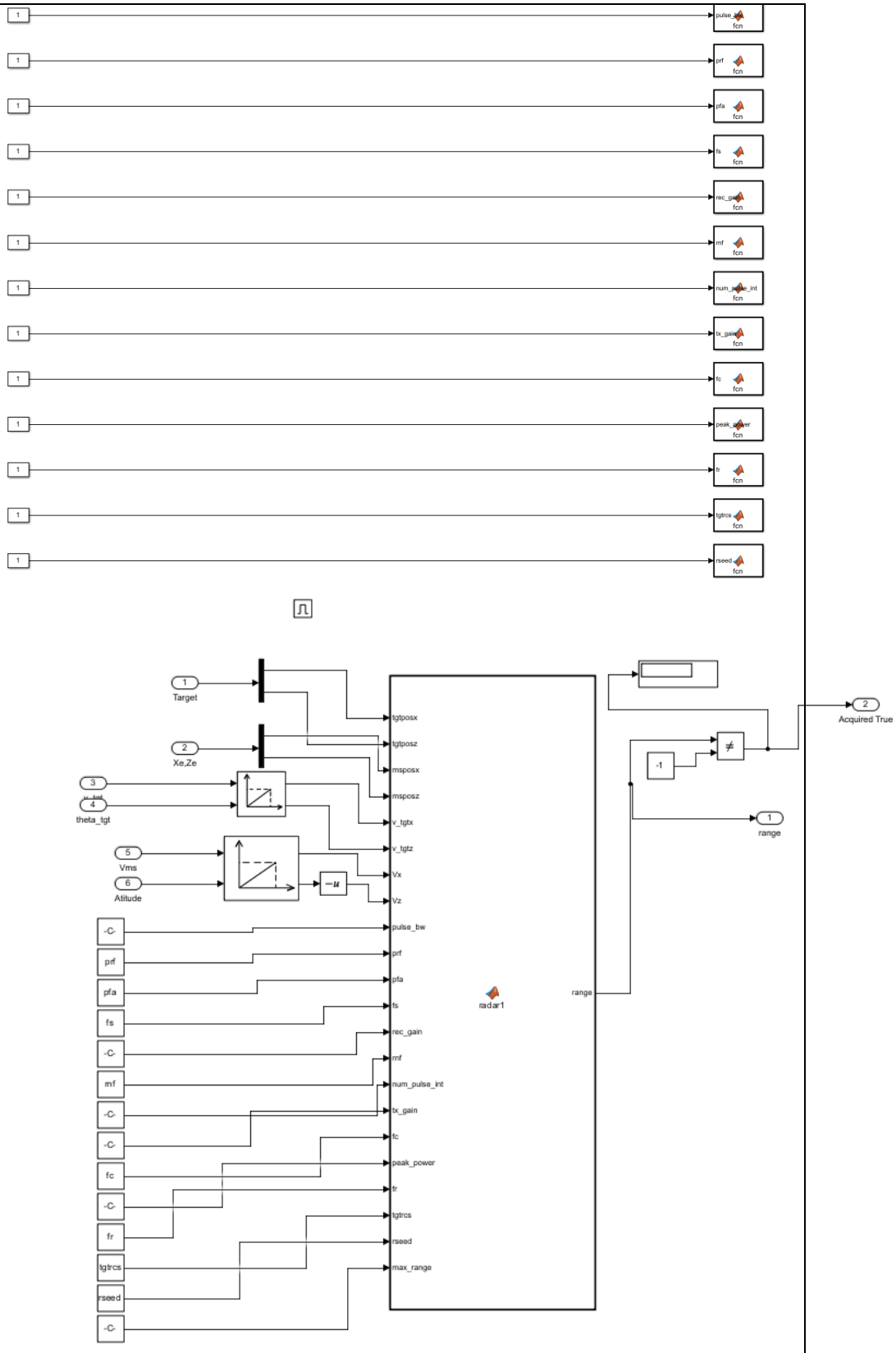




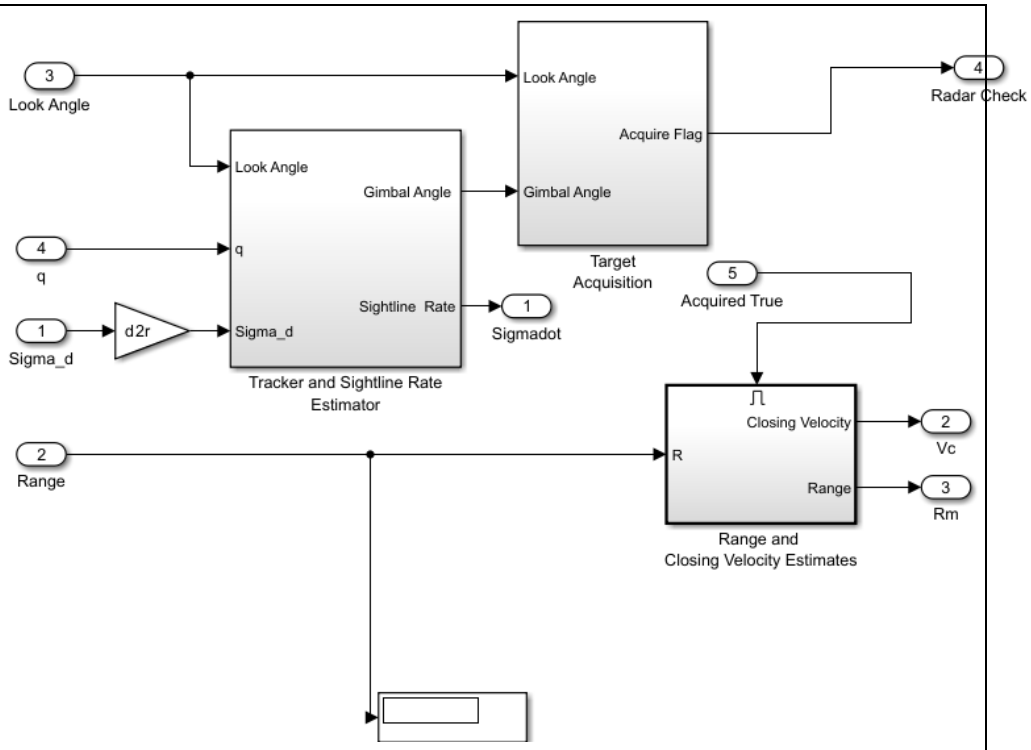

```
function [pulse_bw,prf,pfa,fs,rec_gain,rnf,num_pulse_int,tx_gain,fc,peak_power,fr,tgtrcs,rseed,max_range] = radarcalc(pd,pfa,max_range,ra
    tic
    prop_speed = physconst('LightSpeed');
    pulse_bw = prop_speed/(2*range_res);
    pulse_width = 1/pulse_bw;
    prf = prop_speed/(2*max_range);
    pfa=pfa
    fs = 2*pulse_bw;
    num_pulse_int=num_pulse_int
    snr_min = albersheim(pd, pfa, num_pulse_int);
    tx_gain=tx_gain
    fc=fc
    lambda = prop_speed/fc;
    peak_power = ((4*pi)^3*noisepow(1/pulse_width)*max_range^4*...
        db2pow(snr_min))/(db2pow(2*tx_gain)*tgt_rcs*lambda^2);
    fr=fr
    tgtrcs=tgt_rcs
    rseed=rseed



    toc
end
```



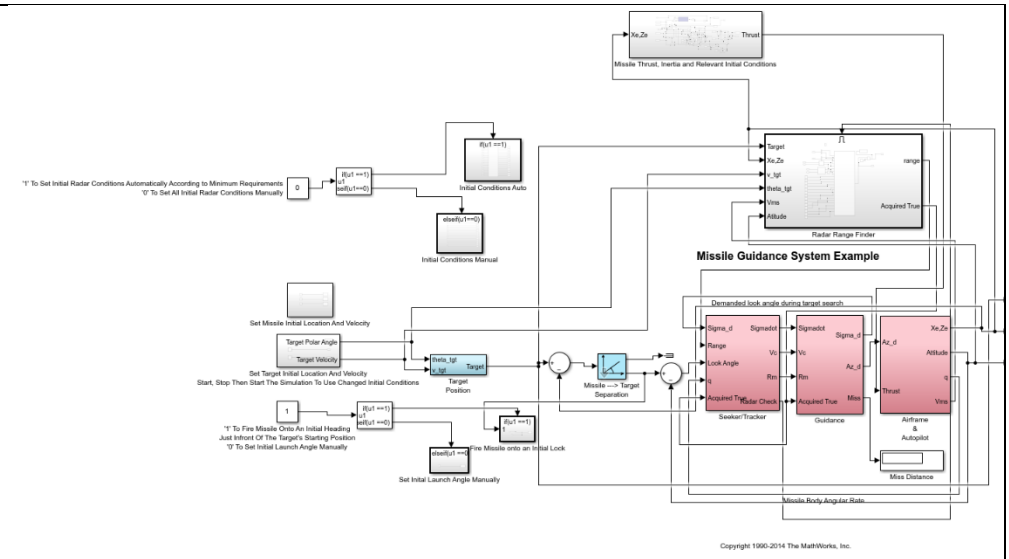
See Appendix B for the full code of the final iteration of the RADAR function

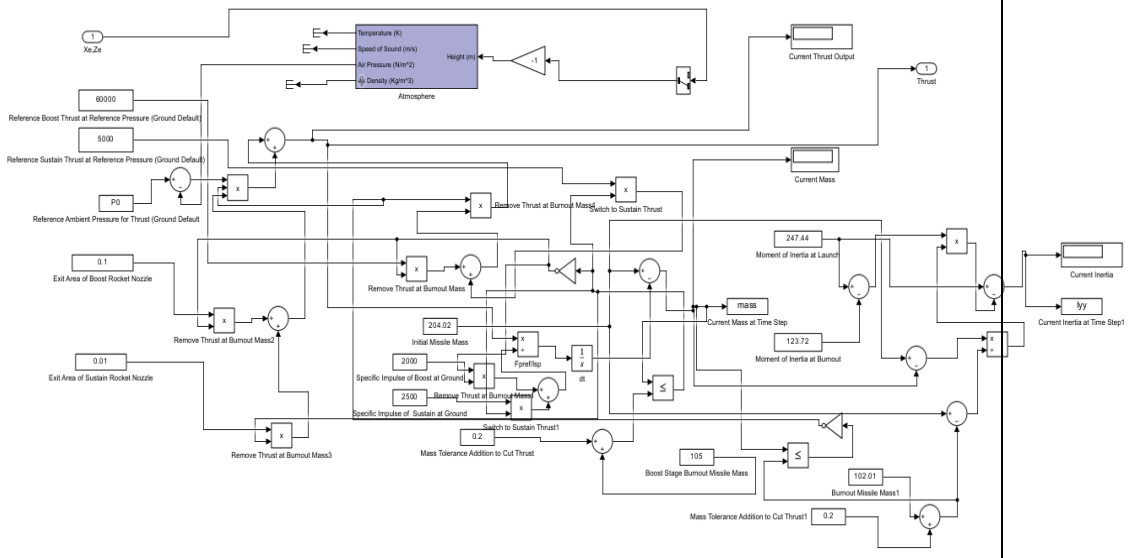


Emailed model and ran it on online matlab on Jacob's computer. Very happy with progress. Just keep going. After meeting researched and found out how to load in variables from .m file when the simulation is opened so you don't have to load into the workspace yourself manually. You modify the preload model property to run the specific .m file.

30/11/2022

Added some more initial conditions like missile initial position and velocity and target initial position and velocity.
Added a choice to launch missile at a custom angle or ahead of the target.
Most importantly used equations of J. Strickland, *Missile Flight Simulation* to model dynamic thrust as the missile goes higher in the atmosphere and dynamic mass as well as dynamic moment of inertia (manoeuvrability of the missile which gets more agile as it loses mass).



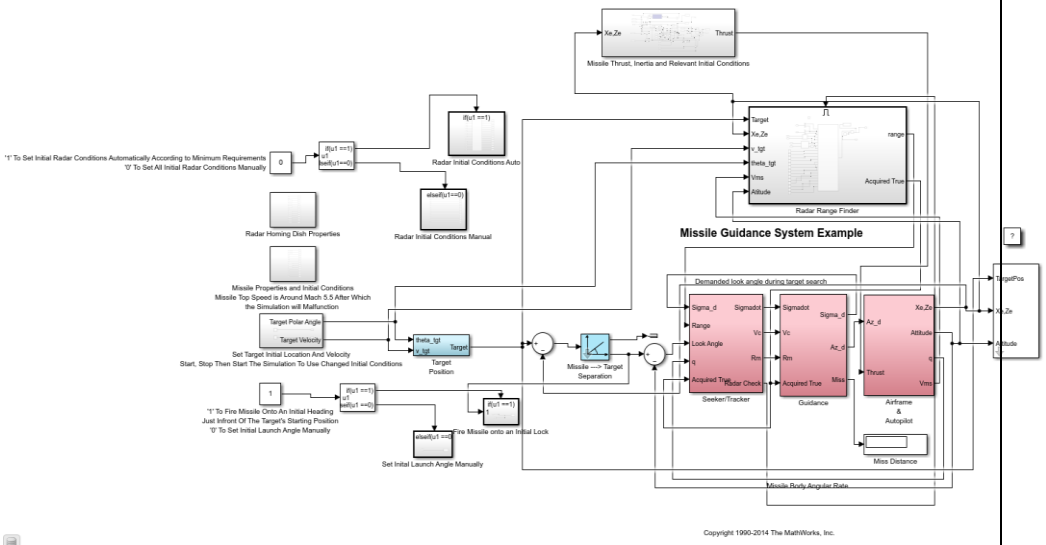


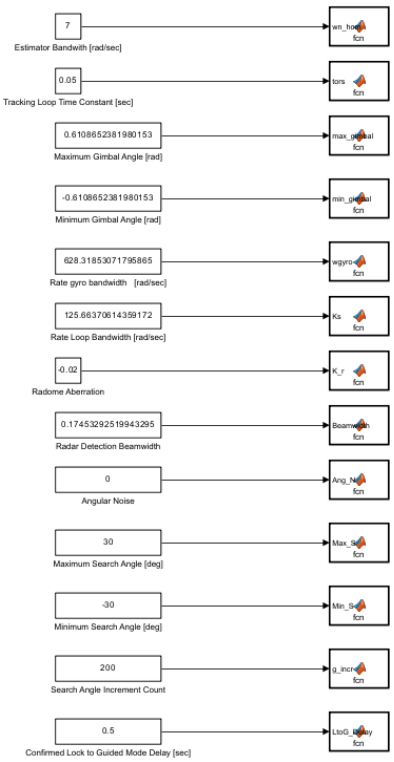
Jacob laptop broke down so couldn't run the code on his laptop but the mentioned progress was discussed, rated very good. Discussed that this is pretty much the end for the scope of the project, just said that want to add every single relevant initial condition for ease of usage and customisation.

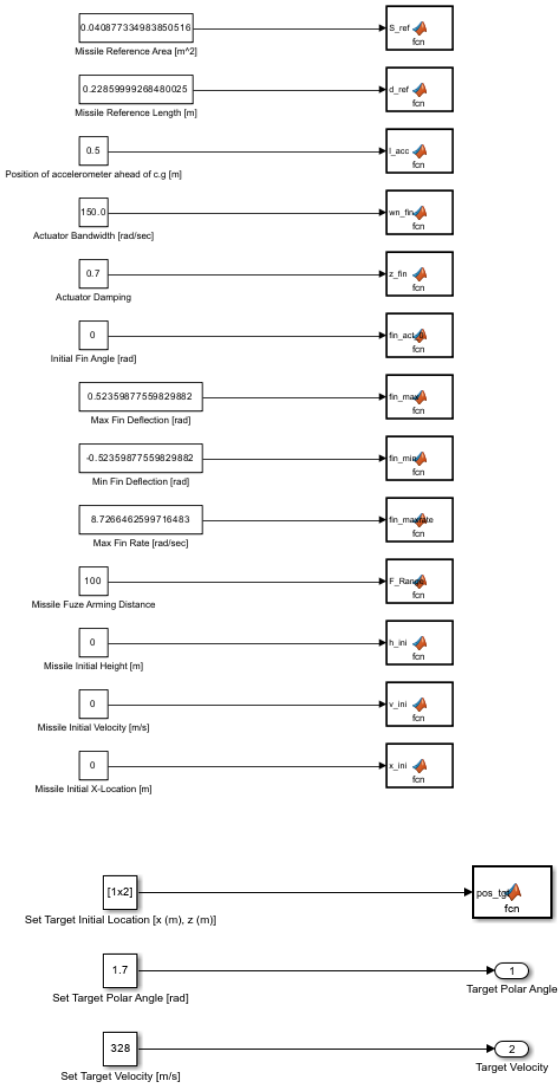
08/12/2022

Added every single relevant initial condition and ran the code on my computer through teams to show to Jacob the code working and discussed everything that has been done and showing how it works. Very happy with the result, now just the write up and the presentation remains. Possibility of finding real values for initial conditions and testing with those if deemed necessary. User will have to start stop start to apply any changed initial condition as they are stored in workspace variables though matlab code, no way around this but it is a very minor annoyance. Missile can't go faster than mach 5.5 as the tables used for the aerodynamic properties of the missile (as given by the standard matlab code) don't seem to

have values for the missile going faster than that. Considering that missiles in real life usually top out at that speed, this isn't an issue.







12/12/2022

[1] "Designing a Guidance System in MATLAB and Simulink - MATLAB & Simulink - MathWorks United Kingdom." https://uk.mathworks.com/help/releases/R2020b/simulink/slref/designing-a-guidance-system-in-matlab-and-simulink.html (accessed Nov. 02, 2022).

[2] J. Strickland, *Missile Flight Simulation*. 2015. Accessed: Oct. 21, 2022. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=VhD8CgAAQBAJ&oi=fnd&pg=PR17&dq=missile+AND+simulation&ots=FlwQmIrkSf&sig=XbyTOkznYTuLsYz6H5WKcSzUINk

[3] "Simulating Test Signals for a Radar Receiver - MATLAB & Simulink - MathWorks United Kingdom."

https://uk.mathworks.com/help/phased/ug/designing-a-basic-monostatic-pulse-radar.html (accessed Nov. 02, 2022).

[4] M. S.-R. handbook and undefined 1962, "Introduction to radar," *helitavia.com*, Accessed: Oct. 13, 2022. [Online]. Available: https://helitavia.com/skolnik/Skolnik_chapter_1.pdf

[5] M. Bühren, B. Y.-Proc. Intern. W. on Intelligent, and undefined 2006, "Automotive radar target list simulation based on reflection center representation of objects," *iss.uni-stuttgart.de*, Accessed: Oct. 14, 2022. [Online]. Available: https://www.iss.uni-stuttgart.de/forschung/publikationen/buehren_wit2006.pdf

[6] J. T. Johnson *et al.*, "Cognitive radar for target tracking using a software defined radar system," *ieeexplore.ieee.org*, 2015, doi: 10.1109/RADAR.2015.7131213.

[7] W. Hao and H. Z. Zhao, "Modeling and simulation of a full coherent LFM pulse radar system based on Simulink," *Proceedings of 2013 2nd International Conference on Measurement, Information and Control, ICMIC 2013*, vol. 1, pp. 495–499, 2013, doi: 10.1109/MIC.2013.6758012.

[8] "Introduction to Radar Scenario Clutter Simulation - MATLAB & Simulink - MathWorks United Kingdom." https://uk.mathworks.com/help/radar/ug/introduction-to-radar-scenario-clutter-simulation.html (accessed Nov. 02, 2022).

[9] "Radar Performance Analysis over Terrain - MATLAB & Simulink - MathWorks United Kingdom." https://uk.mathworks.com/help/radar/ug/radar-performance-analysis-over-terrain.html (accessed Nov. 02, 2022).

[10] "Radar Scenario Tutorial - MATLAB & Simulink - MathWorks United Kingdom." https://uk.mathworks.com/help/radar/ug/radar-scenario-tutorial.html (accessed Nov. 02, 2022).

APPENDIX B: Complete code of the final iteration of the RADAR function

```
function range =
radarfunc(tgtposx,tgtposz,msposx,msposz,v_tgtx,v_tgtz,Vx,Vz,pulse_bw,prf,pfa,fs,rec_gain,rnf,num_pulse_int,tx_gain,fc,peak_power,fr,tgtrcs,rseed,max_range)
tic

   %pd = 0.9;          % Probability of detection
```

```
%pfa = 1e-6;          % Probability of false alarm
%max_range = 10000;    % Maximum unambiguous range
%range_res = 25;       % Required range resolution
%tgt_rcs = 1;          % Required target radar cross section


prop_speed = physconst('LightSpeed'); %%  % Propagation speed
%pulse_bw = prop_speed/(2*range_res); %input   % Pulse bandwidth
pulse_width = 1/pulse_bw; %%              % Pulse width
%prf = prop_speed/(2*max_range); %input       % Pulse repetition frequency
%fs = 2*pulse_bw;   %input                   % Sampling rate
waveform = phased.RectangularWaveform(...
    'PulseWidth',1/pulse_bw,...
    'PRF',prf,...
    'SampleRate',fs);



%tgtposx=tgtposx
%tgtposz=tgtposz
%msposx=msposx
%msposz=msposz
%v_tgtx=v_tgtx
%v_tgtz=v_tgtz
%Vx=Vx
%Vz=Vz
%pulse_bw=pulse_bw
%prf=prf
%pfa=pfa
%fs=fs
%rec_gain=rec_gain
%rnf=rnf
%num_pulse_int=num_pulse_int
%tx_gain=tx_gain
%fc=fc
%peak_power=peak_power
%fr=fr
%tgtrcs=tgtrcs
%rseed=rseed
%max_range=max_range
```

```
noise_bw = pulse_bw; %%

receiver = phased.ReceiverPreamp(...
    'Gain',rec_gain,...                 %input
    'NoiseFigure',rnf,...           %input
    'SampleRate',fs,...
    'EnableInputPort',true);




%num_pulse_int = 10; %input

%snr_min = albersheim(pd, pfa, num_pulse_int);

%tx_gain = 20; %input

%fc = 10e9; %input
lambda = prop_speed/fc; %%

%peak_power = ((4*pi)^3*noisepow(1/pulse_width)*max_range^4*...
    %db2pow(snr_min))/(db2pow(2*tx_gain)*tgt_rcs*lambda^2);


transmitter = phased.Transmitter(...
    'Gain',tx_gain,...
    'PeakPower',peak_power,...
    'InUseOutputPort',true);
fr=fr.';
antenna = phased.IsotropicAntennaElement(...
    'FrequencyRange',fr);     %input ex [0 10]

sensormotion = phased.Platform(...
    'InitialPosition',[round(msposx); 0; round(msposz)],...
    'Velocity',[round(Vx); 0; round(Vz)]);
```

```
radiator = phased.Radiator(...
    'Sensor',antenna,...
    'OperatingFrequency',fc);

collector = phased.Collector(...
    'Sensor',antenna,...
    'OperatingFrequency',fc);



tgtpos = [round(tgtposx);0;round(tgtposz)];
tgtvel = [round(v_tgtx);round(v_tgtz);0];
tgtmotion = phased.Platform('InitialPosition',tgtpos,'Velocity',tgtvel);

%tgtrcs = 1.6; %input
target = phased.RadarTarget('MeanRCS',tgtrcs,'OperatingFrequency',fc);

channel = phased.FreeSpace(...
    'SampleRate',fs,...
    'TwoWayPropagation',true,...
    'OperatingFrequency',fc);

fast_time_grid = unigrid(0,1/fs,1/prf,'[)');
slow_time_grid = (0:num_pulse_int-1)/prf;



receiver.SeedSource = 'Property'; %%
receiver.Seed = rseed; %input ex 2007

% Pre-allocate array for improved processing speed
rxpulses = zeros(numel(fast_time_grid),num_pulse_int);

for m = 1:num_pulse_int

    % Update sensor and target positions
    [sensorpos,sensorvel] = sensormotion(1/prf);
    [tgtpos,tgtvel] = tgtmotion(1/prf);

    % Calculate the target angles as seen by the sensor
    [tgtrng,tgtang] = rangeangle(tgtpos,sensorpos);
```

```
        % Simulate propagation of pulse in direction of targets
        pulse = waveform();
        [txsig,txstatus] = transmitter(pulse);
        txsig = radiator(txsig,tgtang);

        txsig = channel(txsig,sensorpos,tgtpos,sensorvel,tgtvel);

        % Reflect pulse off of targets
        tgtsig = target(txsig);

        % Receive target returns at sensor
        rxsig = collector(tgtsig,tgtang);
        rxpulses(:,m) = receiver(rxsig,~(txstatus>0));
    end

    npower = noisepow(noise_bw,receiver.NoiseFigure,...
        receiver.ReferenceTemperature);
    threshold = npower * db2pow(npwgnthresh(pfa,num_pulse_int,'noncoherent'));




    matchingcoeff = getMatchedFilter(waveform);
    matchedfilter = phased.MatchedFilter(...
        'Coefficients',matchingcoeff,...
        'GainOutputPort',true);
    [rxpulses, mfgain] = matchedfilter(rxpulses);

    matchingdelay = size(matchingcoeff,1)-1;
    rxpulses = buffer(rxpulses(matchingdelay+1:end),size(rxpulses,1));

    threshold = threshold * db2pow(mfgain);



    range_gates = prop_speed*fast_time_grid/2;

    tvg = phased.TimeVaryingGain(...
        'RangeLoss',2*fspl(range_gates,lambda),...
        'ReferenceLoss',2*fspl(max_range,lambda));
```

```
    rxpulses = tvg(rxpulses);


    rxpulses = pulsint(rxpulses,'noncoherent');



    [~,range_detect] = findpeaks(rxpulses,'MinPeakHeight',sqrt(threshold));

    range = round(range_gates(range_detect));
    if isempty(range)
        range=-1

    else
        range=range(1)


toc
end
```