ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

NONLINEAR MODEL BASED GUIDANCE WITH DEEP LEARNING BASED TARGET TRAJECTORY PREDICTION AGAINST AGILE ATTACK PATTERNS

M.Sc. THESIS

Abdullah Sadık SATIR

Department of Aeronautical and Astronautical Engineering

Aeronautical and Astronautical Engineering Programme

MARCH 2021

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

NONLINEAR MODEL BASED GUIDANCE WITH DEEP LEARNING BASED TARGET TRAJECTORY PREDICTION AGAINST AGILE ATTACK PATTERNS

M.Sc. THESIS

Abdullah Sadık SATIR
(511161164)

Department of Aeronautical and Astronautical Engineering

Aeronautical and Astronautical Engineering Programme

Thesis Advisor: Assoc. Prof. Dr. Nazım Kemal ÜRE

MARCH 2021

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ÇEVİK SALDIRI MANEVRALARINA KARŞI DERİN ÖĞRENME TABANLI TEHDİT YÖRÜNGE TAHMİNİ İLE DOĞRUSAL OLMAYAN MODEL ÖNGÖRÜLÜ KONTROL

YÜKSEK LİSANS TEZİ

Abdullah Sadık SATIR
(511161164)

Uçak ve Uzay Mühendisliği Anabilim Dalı

Uçak ve Uzay Mühendisliği Programı

Tez Danışmanı: Assoc. Prof. Dr. Nazım Kemal ÜRE

MART 2021

Abdullah Sadık SATIR, a M.Sc. student of ITU Graduate School student ID 511161164, successfully defended the thesis entitled "NONLINEAR MODEL BASED GUIDANCE WITH DEEP LEARNING BASED TARGET TRAJECTORY PREDICTION AGAINST AGILE ATTACK PATTERNS", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc. Prof. Dr. Nazım Kemal ÜRE** Istanbul Technical University	.....
Jury Members :	**Prof. Dr. Cengiz HACIZADE** Istanbul Technical University	.....
	Dr. Onur ALBOSTAN Turkish Aerospace Industries	.....

Date of Submission : 29 January 2021
Date of Defense : 26 March 2021

To my family,

FOREWORD

First of all, i would like to express my gratitude to my advisor Assoc. Prof. Dr Nazım Kemal ÜRE for providing me research environment and vision. I also thank for influencing me on researching.

I also would like to thank my family. My parents, Ömer SATIR and Melek SATIR always be supportive of doing research and made many sacrifices for me to provide an appropriate working environment. My brother, Muhammed Latif SATIR had also been a big support and patience with me. Words are not enough to simply define their worth.

March 2021 Abdullah Sadık SATIR

TABLE OF CONTENTS

Page

ABBREVIATIONS

AIR	**:** Average Absolute Input Rate
APN	**:** Augmented Proportional Navigation
DNN	**:** Deep Neural Network
EKF	**:** Extended Kalman Filter
GMPC	**:** Generalized Model Predictive Control
IMM	**:** Integrated Multi Model
IT	**:** Interception Time
LOS	**:** Line-Of-Sight
LSTM	**:** Long Short-Term Memory
MD	**:** Miss Distance
MLSTM	**:** Manoeuver Based Long Short-Term Memory
MPC	**:** Model Predictive Control
MSE	**:** Mean Square Error
NMPC	**:** Non-linear Model Predictive Control
NMPC-TAP	**:** Non-linear Model Predictive Control With Target Acceleration Prediction
NN	**:** Neural Network
PN	**:** Proportional Navigation
PPN	**:** Pure Proportional Navigation
RNN	**:** Recurrent Neural Network
TPN	**:** True Proportional Navigation
UKF	**:** Unscented Kalman Filter
QP	**:** Quadratic Programming

LIST OF TABLES

Page

LIST OF FIGURES

Page

NONLINEAR MODEL BASED GUIDANCE WITH DEEP LEARNING BASED TARGET TRAJECTORY PREDICTION AGAINST AGILE ATTACK PATTERNS

SUMMARY

Common guidance laws such as proportional navigation (PN) and augmented proportional navigation (APN) have either limited capability or unsatisfactory engagement performance against manoeuvring threats due to their reactive nature. This thesis presents a missile guidance law overcoming this issue by generating sub-optimal acceleration commands of the interceptor based on a recurrent neural network's acceleration predictions.

The proposed guidance law formulates the differential equations of the two-dimensional interceptor problem in the polar coordinate frame. The difference equations are derived from given differential equations via Euler's discretization. Then the derived difference equations are propagated along the prediction horizon to produce NMPC's matrices. Lastly, a cost function consisting of the weighted norm of the propagated states as well as inputs along a prediction horizon is derived. The interceptor inputs also are constrained. Thus, the interceptor problem has been converted to quadratic programming (QP) problem with the non-linear model predictive control (NMPC) method. A Long Short-Term Memory (LSTM) neural network has produced the necessary target acceleration prediction in this transformation. LSTM uses a position and velocity history to predict target accelerations along the prediction horizon in the QP problem's matrices. The LSTM is trained on a manoeuvre library. Manoeuvres, the combination of the agile two-dimensional manoeuvres such as level flight, coordinated-turn and weave, are generated for different execution speeds and oriented in various directions.

Moreover, three different kinds of neural network models, such as RNN, LSTM and MLSTM, are investigated according to acceleration prediction performance. The MLSTM yielding lower loss value has been chosen as the prediction scheme. MLSTM's prediction results show that it has capable of predicting acceleration within a small error and distinguishing manoeuvre transition. Simulation studies are conducted for interception of an approaching target to compare with common guidance laws and NMPC without target prediction. Results show that the proposed guidance algorithm is superior in terms of miss distance.

ÇEVİK SALDIRI MANEVRALARINA KARŞI DERİN ÖĞRENME TABANLI TEHDİT YÖRÜNGE TAHMİNİ İLE DOĞRUSAL OLMAYAN MODEL ÖNGÖRÜLÜ KONTROL

ÖZET

Tehdit önleme problemi savunma sistemleri için hayati önem taşımaktadır. Bu sebeple tehdidin bertaraf edilme olasılığını artırmak için çeşitli güdüm yasaları geliştirilmiştir. Güdüm yasalarının, tehdidi en düşük ıskalama mesafesi (miss distance) ile önleyebilmesi kullanılan algoritmaların başarısının en önemli göstergesidir. Böylece tehdit, durdurucu füzenin patlama alanı içerisinde kalarak bertaraf edilmiş olacaktır. PN gibi geleneksel güdüm yasaları temel dinamik denklemler ve geometrik ilişkilerden faydalanarak tehdidin engellenmesi için gerekli engelleyici füzenin ivmesinin üretilmesini sağlamaktadır. Fakat bu güdüm yasası hareketli tehditlere karşı yeterli performans gösterememektedir. Tehdidin hareket ettiği durum için önleme problemi doğrusallaştırılarak ikinci dereceden optimizasyon yöntemi ile analitik olarak çözülmesi sonucunda APN güdüm yasası elde edilmiştir. Bu güdüm yasasında tehdide ait ivmesinin PN güdüm yasasına eklenerek engelleme probleminin performansının hareketli tehditler için de iyileştirilebileceği gösterilmiştir.

Hareketli tehdidin gelecekte üreteceği manevraların tahmin edilmesi daha düşük ıskalama mesafesini oluşturulabileceğinin anlaşılmasıyla, farklı çalışmalarda tehdidin gelecekteki parametrelerinin öngörülmesiyle engelleme performansının iyileştirildiği yöntemler uygulanmıştır. Bu çalışmalarda model öngörülü kontrol yapısından esinlenilmiş veya doğrudan yapının kendisi kullanılmıştır. Tehdidin parametre öngörüsü, geçmiş radar verileri kullanarak kalman filtresi, fonksiyon uydurma gibi yaklaşımlarla gerçekleştirilebilmektedir. Kalman filtresi kullanılarka yapılan öngörüler, öngörü yapılacak sistemin karmaşıklığına göre iş yükü ve model karmaşıklığı artmaktadır. Öngörüsü yapımak istenen modellerin bu tür karmaşık yapıya sahip olması durumunda eğer elde fazlaca veride varsa, daha hızlı ve görece daha basit bir yapısı olan sinir ağlarının kullanılması tercih edilebilmektedir. Bu çerçevede literatürde de fazlaca zaman serisi problemlerinde kullanılan yineleyen sinir ağı (Recurrent Neural Network - RNN) yapıları iyi bir çözüm sunmaktadır. Zaman adımında uzak geçmişe bağımlılığın yüksek olduğu sistemlerde RNN yapılarının yetersiz kalması dolayısıyla LSTM sinir ağları kullanılmaya başlanmıştır. Bununla birlikte bizim problemde olduğu gibi manevra tabanlı bir öngörü prosedürünün olduğu durum için yapılan bir çalışmada manevra sınıflarınında sinir ağı yapısına dahil edilmesi yapılan öngörülerin iyileştirdiği gösterilmiştir.

Bu tezde, iki eksenli güdüm problemi lineer olmayan model öngörülü kontrol yöntemi (NMPC) kullanılarak ikinci dereceden programlama (QP) problemine dönüştürülmüştür. Long Short-Term Memory (LSTM), bu dönüşümde ihtiyaç duyulan tehdit ivmesi öngörülerini üretmiştir. Öngörülerde, hedefe ait geçmiş konum ve hız bilgisi kullanılırken, tehdidin öngörü ufku boyunca uygulayabileceği ivme girdileri kestirilmeye çalışmıştır.

Yukarıda belirtildiği üzere, bu çalışma NMPC ve LSTM yöntemlerinin birleştirilmesinden oluşmaktadır. İlk aşamada önleyici füze ivme dizisini bulan NMPC yöntemi şekillendirilmiştir. Bu amaçla iki boyutlu tehdit önleme probleminin dinamik modeli, polar koordinat sisteminde oluşturulmuştur. Elde edilen dinamik denklemler durum uzayında ifade edilerek matris formatına dönüştürülmüştür. Gelecek zaman tahminlerini oluşturmak için ihtiyaç duyulan denklemlerin üretilmesi amacıyla polar koordinat sistemindeki diferansiyel denklemler Euler yöntemi kullanılarak fark denklemlerine dönüştürülmüş ve şimdiki zamandaki durum değişkenleri bir önceki adımdaki durum değişkenleri cinsinden ifade edilmiştir. Bu denklemlerde durum uzayı değişkenleri istenilen zaman ufku boyunca birbirine bağlı olarak üretilerek matris şeklinde tanımlanarak standart öngörü formu denklemleri oluşturulmuştur. Buna ek olarak, denklemlerde kullanılan ivme girdileri bir önceki zamanda uygulanan ivme girdisinin farkı olacak şekilde tanımlanmıştır. Standard öngörü formundaki matrislerin hesaplanması için önleyici füzenin öngörü ufku boyunca ivme değerinin bilinmesi gerekmektedir. Bu problemin üstesinden gelebilmek için standart form denklemleri bir önceki adımda hesaplanan ivme girdileri cinsinden tanımlanmıştır. Ardından, optimizasyon için gerekli maliyet fonksiyonu durum uzayı değişkenlerinin ve her adımda uygulanan ivme değişiminin ağırlıklı karesi kullanılarak oluşturulmuştur. Önleyici füzenin ivme limitleri de probleme kısıt olarak eklenmiştir. Gerekli düzenlemeler yapıldıktan sonra NMPC problemi ikinci dereceden programlama problemi olarak tanımlanmıştır. Ayrıca tezde verilen bu yönetimin karşılaştırılacağı literatürde bulunan bir yöntemin ikinci dereceden programlama formatı verilmiştir. Bu yöntemde tehdit ivme tahminleri ikinci dereceden programlama problemine kısıt olarak verilmektedir.

İkinci aşamada ivme tahminlerinin alınacağı LSTM sinir ağının eğitilmesi ve bu eğitim için gerekli yörüngelerde oluşan verinin üretilmesi işlemleri bulunmaktadır. Gerekli verinin üretilebilmesi için düz uçuş, koordine dönüş ve weave manevralarını art arda belli olasılıklarla yaptırılarak bir manevra kütüphanesi oluşturulmuştur. Manevraların oluşturulması sırasında farklı manevra parametreleri kullanılırken her bir manevranın gerçekleştirildiği simülasyon farklı yönde ve farklı hızlarda başlatılmıştır. Bu süreçte, manevraların uygulanırken hava aracının konumu ve hızı x-y koordinat sisteminde kayıt edilmiştir. Ayrıca bu yörüngeyi gerçekleştirirken ki ivme değerleri kayıt altına alınmıştır. Kütüphanedeki manevralar arzu edilen öngörü formatına getirilebilmek için bir ön işlem sürecinden geçirilmiştir. Bu süreçte manevralar elli zaman adımlık parçalara ayrılıp her bir parça sinir ağına girdi olacak şekilde planlanırken her ellinci adımdan sonraki öngörü ufku boyunca ki ivme değeri sinir ağının tahmin edeceği ivme çıktısı olacak şekilde planlanmıştır. Üretilen veriler kullanılırken normalize edilerek sinir ağının belirli bir manevrayı yerine daha genel ama aynı karaktere sahip manevraları da öngörebilmesi hedeflenmiştir.

Sinir ağı mimarisi seçerken zaman serisi problemlerinde tercih edilen yineleyen sinir ağı (RNN) birimlerinden oluşan mimariler tercih edilmiştir. Literatürde bulunan mimarilerden bizim probleme en uygun olanı kodlayıcı-kod çözücü (encoder-decoder) mimarisidir. Çünkü bu mimaride girdi olarak verilen zaman adımı ile çıktı olarak alınacak zaman adımı sayısının aynı olması gerekmemektedir. Bu yapı farklı sinir birimleri kullanılarak ve yapısında değişiklikler yapılarak çeşitlendirilmiştir. Çeşitlendirilen bu yapılara uygulanan testlerde M-LSTM yapısının daha düşük hata ile ivme öngörülerini üretebildiği gözlemlenmiştir. NN tarafından yapılan ivme

öngörülerinin gerçekte olması gereken ivme değerine oldukça yakın olduğu ve manevra geçişlerinin oldukça iyi tahmin edilebildiği görülmüştür.

Arzu edilen güdüm yasasının performansı PN, APN ve NMPC gibi diğer güdüm yasalarıyla simülasyon ortamında karşılaştırılmıştır. Performansların test edilmesi için farklı G kuvvetlerinde sırayla koordine dönüş, düz gitme ve weave manevralarını birleşiminden oluşan bir manevra kullanılmıştır. Bu amaçla her bir güdüm yasası farklı G kuvvetlerinde gerçekleştirilen bu manevralar ile yüz Monte-Carlo simülasyonunda test edilmiştir. Elde edilen verilerde NMPC-TAP'ın daha düşü ıska mesafesine sahip olduğu gözlemlenmiştir. Ayrıca öngörü zamanının daha fazla artırılması ıska mesafeni daha da azalttığı gözlemlenmiştir. Ayrıca güdüm yasalarının ürettiği ivme değerlerine baktığımızda NMPC-TAP'ın diğer güdüm yasalarından bir miktar daha önce tehdidin davranışına karşılık verdiği görülmüştür.

1. INTRODUCTION

The interception problem is the most crucial subject in defence systems due to its vital significance. Different kinds of guidance algorithms have been developed to increase the killing probability of targets on the ground or air due to their importance. The guidance algorithms should be performed so that missiles able to intercept targets with small Miss Distance (MD). So, during interception, the target stays in the missile explosion area. Traditional algorithms utilize geometric parameters in the engagement geometry for handling such a killing capability. But today, demands on optimization-based guidance algorithms have increased due to the increasing computational power and their better interception performance. If the target has agile manoeuvre capability, both traditional and optimization-based guidance approaches that do not have any target prediction cannot achieve low Miss Distance (MD). Therefore, to develop a guidance algorithm that can intercept the targets with agile manoeuvre capability with lower Miss Distance (MD), predicting targets' future manoeuvres is crucial.

This thesis's primary purpose is to approach the problem in a model predictive control manner to create optimal control inputs of the missile with the help of the Neural Network acceleration predictor.

This first chapter starts with the problem statement section and continues with the literature survey and research summary section.

1.1 Problem Statement

In most of the case, if a target tries to enter the defended area, it does some confusing tricks for avoiding radar tracking target. These maneuvers affect the interceptor missile guidance performance if the targets' maneuvers are not considered. Some proportional navigation guidance (PNG) algorithm's equation variants involve the target's last known acceleration. So, interception performance is enhanced. Starting

from this point, where the target's last acceleration is used in the guidance equation, some guidance algorithms that use the target behaviour prediction method have been introduced in past years. Our work is inspired by this target behaviour prediction concept and uses it in the model predictive control (MPC) framework.

A point-mass model with acceleration inputs in 2D space represents the missile-target interception dynamics. In this thesis, guidance inputs are obtained for the MPC framework that needs system dynamics to be linearised along the prediction horizon. Because this interception problem is such a combination of two different systems (missile and target) that missile controlled by MPC, target controlled by enemy pilots along prediction horizon, this unknown target acceleration along the future horizon makes the new problem apparent. In this thesis, the prediction problem is solved with a Neural Network (NN) trained by many structured agile maneuvers.

Target's inputs prediction is obtained with Long Short Term Memory (LSTM), a Recurrent Neural Network (RNN) used in time series data prediction. LSTM is trained with the input of x-y position and velocity and output of lateral acceleration on a dataset that contains the different kinds of structured agile maneuvers. These maneuvers are the combination of coordinated turn, weave maneuver, and level flight. Thus, NMPC uses these inputs to obtain a suboptimal missile control sequence. All of these processes are illustrated in Figure-1.1. In the figure, LSTM Network takes the target's state history, which contains x-y position and velocities of target, produces acceleration perpendicular to the target's velocity along the prediction horizon. NMPC block takes these acceleration inputs then gives suboptimal control input sequence of interceptor missile along control horizon. In every simulation loop, only first inputs (a_M^1) depicted as the green arrow in the figure is applied to interceptor missile as input according to receding horizon procedure.

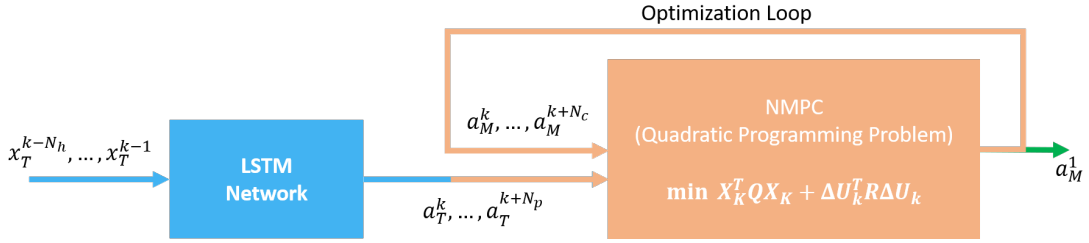

Figure 1.1: Algorithm block diagram.

1.2 Literature Survey

The most popular interceptor guidance law is Proportional Navigation (PN) because of its simplicity, effectiveness, and applicability. Thus, it has been widely studied and derived for strengthening weak points in the literature. It calculates the control input by using only geometric relations of engagement. It creates lateral acceleration that is proportional to the line-of-sight (LOS) angle rate, is also perpendicular to the line of sight [1]. The first published paper about PN is [2]. PN guidance methods can be classified according to the acceleration direction generated by PN. True proportional navigation (TPN) is one of the PN guidance classes that creates lateral acceleration normal to the line-of-sight (LOS). Because missile needs longitudinal acceleration for tracking guidance command, it is not applicable for a missile that does not have a longitudinal control system. Its performance investigation against a non-manoeuvring target was done in [3]. Another PN class that creates lateral acceleration normal to the interceptor velocity vector is Pure proportional navigation (PPN). Because it does not create longitudinal acceleration; hence, it does not force the interceptor to accelerate longitudinally. In [4], its performance is investigated in a non-linear framework. Because of forward acceleration, relatively large control effort requirements of TPN, PPN is the more practical guidance law [5]. Although it had been shown that PPN has adequate performance against a target with time-varying acceleration [6], optimal interception law had come up with APN against manoeuvring targets [7, 8].

APN guidance law contains the target acceleration information in the interception command. Therefore, guidance performance has been improved against specific manoeuvring targets. Although its performance improves against manoeuvring targets, its performance is still limited by non-agile manoeuvres. Several other guidance laws regarding the target acceleration have been derived to improve the interceptor's capability, aside from APN [9]. Some of them are based on optimal control theory such as GMPC [10], MPC and NMPC [11, 12]. In [11], non-linear model predictive control (NMPC) was studied for planer engagement scenario constrained by the bounded target acceleration. A neural network-based solution of quadratic programming (QP) formulation of the problem was solved for on-line implementation. In [12], a set of soft terminal constraint was added on QP to guarantee that interceptor get closer to the target. Besides optimization with bounded target acceleration, some methods

use target states themselves prediction or distribution prediction. [13, 14] are some of them which predict target position distribution . Model predictive guidance law was derived by inspiring from MPC for terminal phase guidance in [13, 14]. The problem cost function contains the sum of the probability density function of the estimated target position in the interceptor's effective area along its trajectory. They use an integrated multi model filter for estimating density function of target future position. We formulate the interceptor problem as an NMPC problem that can be solved by QP similar to [12]. However, acceleration prediction was used instead of constraining the target acceleration.

Various filter-based methods for estimating the necessary target states have been examined in the literature. The most known and used among them are the ones based on the Kalman filter. Its variations have been derived for adopting the method in different applications [15, 16]. Kalman filter is used in linear systems as, EKF, UKF and particle filter are used in a non-linear system. The particle filters differentiate from Kalman filter methods within that it doesn't have limitation imposed by gaussian assumption. Although all of these methods exist, these are not sufficient to use in the model representing the agile manoeuvring target because of its manoeuvring capabilities. If only a model cannot represent the target, interacting multiple model (IMM) filters are used to model different target manoeuvre behaviour, and switch between them [13, 14, 17–20]. Even so, they can switch among different models; classical methods can suffer from model uncertainty. Neural-network based approaches have strong capability of non-linear mapping as long as sufficient data provided [21–23]. The data gives the Neural Network (NN) to handle the capability of the model uncertainty. The data's that considered here consists of time-varying inputs and outputs. This kind of time series data is well fitted by Recurrent Neural Networks (RNN). Preliminary RNN architectures suffer from gradient vanishing and exploding problem. Long Short-Term Memory NN overcomes these problems [24]. Hence LSTM enables to fitting on time depended on long data. Different kind of areas make use of its advantage in time series fitting problem such as pedestrian trajectory prediction [25], vehicle trajectory prediction [26, 27], and next-word prediction in

sentence [28]. In our approach, we use encoder-decoder architecture to fitting data coming from structure attack simulations.

1.3 Objectives, Challenges and Solutions

In this section, sub-objectives that bring us to find the optimal input sequence coming up with small miss distance is defined. When achieving these sub-objectives, encountered challenges and the solutions of them coming through are briefly explained.

First, sub-objective is developing an MPC-based guidance algorithm that involves target acceleration command as input. The encountered challenges are listed as follows;

- Controllability of linearized engagement dynamics.
- Integrating interceptor acceleration to MPC problem.

Controllability of linearized engagement dynamics is increased with polar coordinate in engagement dynamics as done in [11, 12]. In the NMPC, the ΔU_k represent the difference of successive lateral acceleration input of interceptor. Because of the absence of desired interceptor acceleration in the optimization moment, the necessary ΔU_k is taken from last time. So, NMPC formulation obtained.

Second sub-objective is training an LSTM NN for prediction with high accuracy. The encountered challenges are listed as follows;

- Data generation.
- LSTM architecture selection.

Data is generated with random selected sequential manoeuvres involving weave, coordinated turn and straight flight. Encoder-decoder LSTM structure gives more accurate results after some trial-and-error for different architectures.

Structure of the thesis is as follows; the second chapter involves background information about engagement problem and nonlinear model predictive framework. The third chapter demonstrates the LSTM model used and training strategy. The last chapter is devoted to the simulation's results and conclusion.

2. GUIDANCE ALGORITHM AND CONSTRAINED OPTIMIZATION

This chapter is devoted to explaining engagement geometry and the non-linear model predictive control approach in guidance problems. The introduction already states that this thesis's primary objective is to develop a guidance algorithm that resulted in superior miss distance performance to the classical guidance laws.

The chapter starts with the description of engagement geometry in cartesian and polar coordinate. Next section explains both promised non-linear model predictive control and its variant in the literature.

2.1 Engagement Problem

This section expresses the two-dimensional engagement problem in both the cartesian and polar coordinate systems. Both represent the same problem, but they are useful when used for a specific purpose.

The engagement problem contains two apart systems controlled by distinct sources. They are a missile called interceptor, and a target can be anything attacking defended assets. The aim of the missile is intercepting the avoiding target when the target tries to confuse interceptor. Moreover, most of the case the avoiding target perform its manoeuvrer in the two-dimensional x-y plane. Due to this reason, the following subsections explain two separate systems' dynamics for the two-dimensional framework.

When describing system dynamics, following assumptions have been made for modelling the problem;

- Point mass model
- Engagement takes places in the two-dimensional space
- Constant velocities

- Only lateral acceleration input exists
- The acceleration inputs apply immediately

2.1.1 Cartesian coordinated system

All states in the interceptor model can be represented separately in the cartesian coordinates system. For this reason, it is useful for simulating engagement problems. Its parameters are also easily converted to the polar coordinate system.

Figure-2.1 depicts the two-dimensional geometry of the engagement problem. The engagement model contains two separate systems controlled by lateral acceleration input. In the figure, subscript **M** represents the states of missile (interceptor) as subscript **T** represents the target states.

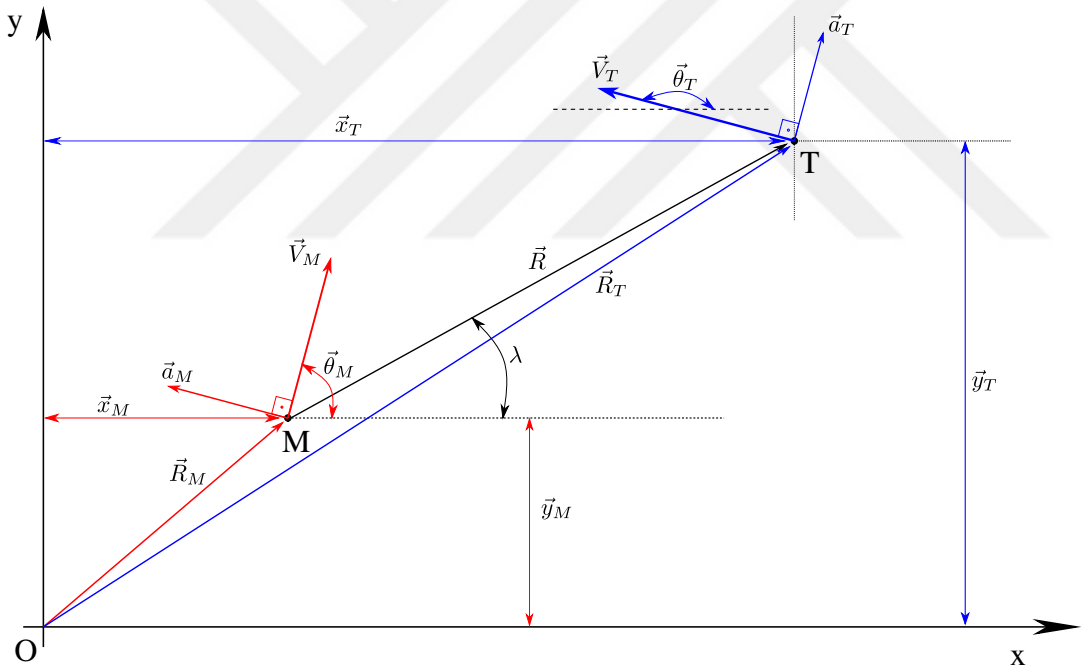

Figure 2.1: Engagement geometry in cartesian coordinate system.

Assumed that xOy axes define an inertial frame with unit vectors $\hat{i}$ and $\hat{j}$. x_M, y_M, x_T and y_T are respectively the x-y coordinates of missile and target. $\overrightarrow{R_M}$, $\overrightarrow{V_M}$, $\overrightarrow{R_T}$, and $\overrightarrow{V_T}$ are position and velocity vectors of the missile and target. λ represents the angle between line-of-sight (LOS) and the x axis. Dynamics of the missile-target engagement system can be defined in cartesian coordinate system with the help of state vector $X = \left[x_M, y_M, x_T, y_T, V_{M,x}, V_{M,y}, V_{T,x}, V_{T_y}\right]^T \in \Re^8$. Subscripts M and T correspond

to missile, target as, subscripts x and y correspond in x and y axis components. Equation 2.1 represents the general form of the non-linear engagement dynamics as a function of time, t, the state itself and the system input, u.

$$\dot{\mathbf{X}} = F(t, \mathbf{X}, u) \tag{2.1}$$

Equation 2.1 can also be written in open form.

$$\begin{bmatrix} \dot{x}_M \\ \dot{y}_M \\ \dot{x}_T \\ \dot{y}_T \\ \dot{V}_{M,x} \\ \dot{V}_{M,y} \\ \dot{V}_{T,x} \\ \dot{V}_{T,y} \end{bmatrix} = \begin{bmatrix} V_{M,x} \\ V_{M,y} \\ V_{T,x} \\ V_{T,y} \\ -a_M \sin\theta_M \\ a_M \cos\theta_M \\ -a_T \sin\theta_T \\ a_T \cos\theta_T \end{bmatrix} \tag{2.2}$$

where the dotted terms are the time derivatives, a_M, a_T are the norm of acceleration vector (perpendicular to the corresponding body's velocity), θ_M, θ_T are heading angles calculated from equation 2.3.

$$\begin{aligned} \theta_M &= \arctan \frac{V_{M,y}}{V_{M,x}} \\ \theta_T &= \arctan \frac{V_{T,y}}{V_{T,x}} \end{aligned} \tag{2.3}$$

Missile-target range $\overrightarrow{R}$ and missile-target closing velocity $\overrightarrow{V}$ can be written as in equation 2.4.

$$\begin{aligned} \overrightarrow{R} &= \overrightarrow{R_T} - \overrightarrow{R_M} = R_x\hat{\imath} + R_y\hat{\jmath} \\ \overrightarrow{V} &= \overrightarrow{V_T} - \overrightarrow{V_M} = V_x\hat{\imath} + V_y\hat{\jmath} \end{aligned} \tag{2.4}$$

Components of closing velocity and range are given in equations 2.5,2.6.

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \dot{V}_{T,x} - \dot{V}_{M,x} \\ \dot{V}_{T,y} - \dot{V}_{M,y} \end{bmatrix} \tag{2.5}$$

$$\begin{bmatrix} R_x \\ R_y \end{bmatrix} = \begin{bmatrix} \dot{R}_{T,x} - \dot{R}_{M,x} \\ \dot{R}_{T,y} - \dot{R}_{M,y} \end{bmatrix} \tag{2.6}$$

The system inputs are $u = [a_M,\ a_T]$ where a_M interceptor acceleration controlled by guidance algorithm and a_T is target acceleration controlled by opponent forces.

Open form state representation of the engagement model involves all possible states in two-dimensional space. This useful characteristic implies that the bodies' states can easily be tracked in a simulation environment constructed on the state-space model in a cartesian coordinate system.

2.1.2 Polar coordinated system

The polar coordinate system is represented by r distance to missile and angle λ between line-of-sight and x-axes are given in Figure-2.2. The state-space model given in equation 2.2 can be converted to the polar frame.

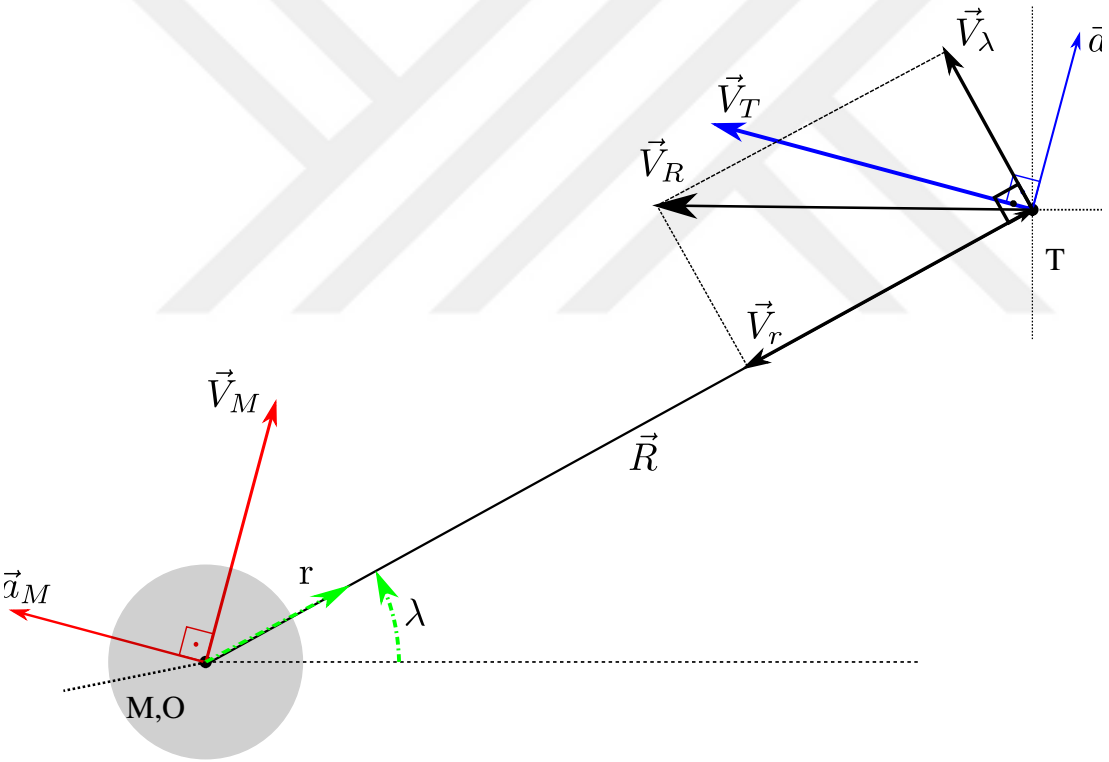

Figure 2.2: Engagement geometry in polar coordinate system.

2.7 gives the kinematic equations corresponding polar coordinate system.

$$
\begin{aligned}
\dot{r} &= V_T \cos(\lambda - \theta_T) - V_M \cos(\lambda - \theta_M) \\
r\dot{\lambda} &= -V_T \sin(\lambda - \theta_T) + V_M \sin(\lambda - \theta_M) \\
\dot{\theta_M} &= \frac{a_M}{V_M} \\
\dot{\theta_T} &= \frac{a_T}{V_T}
\end{aligned}
\tag{2.7}
$$

By assuming that the magnitude of missile and target velocities are constant, the equation 2.7 derived to time yields [29–31]

$$
\begin{aligned}
\dot{r} &= V_r \\
\dot{V}_r &= \frac{V_\lambda^2}{r} + a_{Tr} - a_M \sin(\lambda - \theta_M) \\
\dot{\lambda} &= \frac{V_\lambda}{r} \\
\dot{V}_\lambda &= -\frac{V_r V_\lambda}{r} + a_{T\lambda} - a_M \cos(\lambda - \theta_M)
\end{aligned}
\tag{2.8}
$$

where V_r,V_λ,a_{Tr},$a_{T\lambda}$ depicted in Figure-2.2 and defined as follows [31];

$$
\begin{aligned}
V_r &= V_T \cos(\theta_T - \lambda) - V_M \cos(\theta_M - \lambda) \\
V_\lambda &= V_T \sin(\theta_T - \lambda) - V_M \sin(\theta_M - \lambda) \\
a_{Tr} &= a_T \sin(\theta_T - \lambda) \\
a_{T\lambda} &= a_T \cos(\theta_T - \lambda)
\end{aligned}
$$

V_λ is a transversal component of relative velocity rotating with LOS, V_r is a component of relative velocity along LOS, a_{Tr} and $a_{T\lambda}$ can be described as the projection components of the target acceleration. Let us consider the state variable vector $x = [r, V_r, \lambda, V_\lambda]^T$ and rewrite equations 2.8 in the state space as follows;

$$
\dot{x} = f(x) + g(x)u + d(x)w, \tag{2.9}
$$

where $x \in \Re^4$ is a state vector, $w = [a_{Tr}, a_{T\lambda}]^T \in \Re^2$ is a vector which represents target acceleration in polar coordinates, and $u = a_M \in \Re$ is a control input. The control input is subjected to an symmetrical acceleration limits, which are expressed in equation 2.10.

$$
-u_{max} \leq u \leq u_{max}. \tag{2.10}
$$

As we can see in the next sections, the polar state representation is well suited for the linear engagement guidance problems. Although we solve the non-linear guidance problem, the NMPC model solves the problem after it is linearized.

2.2 Non-linear Model Predictive Control

State-space formulation of engagement dynamics has been derived beforehand. In this section, non-linear model predictive control (NMPC) will form the engagement problem to become solvable as linear quadratic programming in discrete-space. The inspired NMPC schema will also be explained briefly as well as promising Non-linear model predictive control with target acceleration prediction (NMPC-TAP).

2.2.1 Non-linear model predictive control with target acceleration prediction

As said before, NMPC will employ the polar state-space representation. The compact form of the kinematic equation of engagement has been given in equation 2.9 as;

$$\dot{x} = f(x) + g(x)u + d(x)w, \tag{2.11}$$

where,

$$f(x) = \begin{bmatrix} V_r \\ \frac{V_\lambda^2}{r} \\ \frac{V_r V_\lambda}{r} \\ -\frac{V_r V_\lambda}{r} \end{bmatrix}, \quad g(x) = \begin{bmatrix} 0 \\ \sin(\lambda - \theta_M) \\ 0 \\ -\cos(\lambda - \theta_M) \end{bmatrix}, \quad d(x) = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$u = [a_M] \in \Re, \quad w = \begin{bmatrix} a_{Tr} \\ a_{T\lambda} \end{bmatrix} \in \Re^2$$

MPC needs propagated states throughout the prediction horizon. These propagation equations can be derived from discrete-time domain. For this reason, the differential equation 2.11 should be written in the discrete domain. The difference equation easily be obtained by applying Euler discretization.

$$\dot{x} = f(x) + g(x)u + d(x)w \quad \textbf{(Differantial Equation)} \tag{2.12}$$

$$\frac{x_{k+1} - x_k}{\Delta(t)} = f(x_k) + g(x_k)u + d(x_k)w \quad \textbf{(Euler Discretization)} \tag{2.13}$$

$$x_{k+1} = x_k + \Delta(t) f(x_k) + \Delta(t) g(x_k)u + d(x_k)w \tag{2.14}$$

Thus, the difference equation is

$$x_{k+1} = f_d(x_k) + g_d(x_k)u_k + d_d(x_k)w_k \tag{2.15}$$

where,

$$f_d(x_k) = x_k + \Delta(t)f(x_k), \tag{2.16}$$

$$g_d(x_k) = \Delta(t)g(x_k), \tag{2.17}$$

$$d_d(x_k) = \Delta(t)d(x_k) \tag{2.18}$$

The reference predictive form can be expressed by using equation 2.15 as

$$\begin{aligned} x_{k+j|k} =& f_d(x_{k+j-1|k}) + \\ & g_d(x_{k+j-1|k})u_{k+j-1|k} + \\ & d_d(x_{k+j-1|k})w_{k+j-1|k} \end{aligned} \tag{2.19}$$

$$u_{k+j-1|k} = u_{k+j-2|k} + \Delta u_{k+j-1|k} \tag{2.20}$$

where subscript $k+j|k$ means that the current time-step is k and the distance from the current time-step is j.

MPC needs states to be defined along the prediction horizon. These states are obtained by propagating equation 2.19. The propagating states are put together into a matrix format, gives us a standard prediction form given in equation 2.23. Let us write the standard prediction form steadily by starting with the propagation equations until N_p-th time step.

$$\begin{aligned} x_{k+1|k} =& f_d(x_{k|k}) + g_d(x_{k|k})u_{k|k} + d_d(x_{k|k})w_{k|k} \\ x_{k+2|k} =& f_d(x_{k+1|k}) + g_d(x_{k+1|k})u_{k+1|k} + d_d(x_{k+1|k})w_{k+1|k} \\ x_{k+3|k} =& f_d(x_{k+2|k}) + g_d(x_{k+2|k})u_{k+2|k} + d_d(x_{k+2|k})w_{k+2|k} \\ & \vdots \\ x_{k+N_p|k} =& f_d(x_{k+N_p-1|k}) + g_d(x_{k+N_p-1|k})u_{k+N_p-1|k} + d_d(x_{k+N_p-1|k})w_{k+N_p-1|k} \end{aligned} \tag{2.21}$$

States and input vectors are selected along prediction horizon as equation 2.22.

$$X_k = \begin{bmatrix} x_{k+1|k} \\ x_{k+2|k} \\ \vdots \\ x_{k+N_p|k} \end{bmatrix}, U_k = \begin{bmatrix} u_{k|k} \\ u_{k+1|k} \\ \vdots \\ u_{k+N_c-1|k} \end{bmatrix}, \Delta U_k = \begin{bmatrix} \Delta u_{k|k} \\ \Delta u_{k+1|k} \\ \vdots \\ \Delta u_{k+N_c-1|k} \end{bmatrix} \tag{2.22}$$

Thus, the propagation equations 2.21 can be written in the standard prediction form.

$$X_k = F_k + G_k \Delta U_k + g_k + d_k \tag{2.23}$$

where,

$$F_k = \begin{bmatrix} f_d(x_{k|k}) \\ f_d(x_{k+1|k}) \\ \vdots \\ f_d(x_{k+N_p-1|k}) \end{bmatrix}, \; g_k = \begin{bmatrix} g_d(x_{k|k})u_{k-1|k} \\ g_d(x_{k+1|k})u_{k-1|k} \\ \vdots \\ g_d(x_{k+N_p-1|k})u_{k-1|k} \end{bmatrix}, \; d_k = \begin{bmatrix} d_d(x_{k|k})w_{k|k} \\ d_d(x_{k+1|k})w_{k+1|k} \\ \vdots \\ d_d(x_{k+N_p-1|k})w_{k+N_p-1|k} \end{bmatrix}$$

$$G_k = \begin{bmatrix} g_d(x_{k|k}) & 0 & 0 & \cdots & 0 \\ g_d(x_{k+1|k}) & g_d(x_{k+1|k}) & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ g_d(x_{k+N_p-1|k}) & g_d(x_{k+N_p-1|k}) & g_d(x_{k+N_p-1|k}) & g_d(x_{k+N_p-1|k}) & g_d(x_{k+N_p-1|k}) \end{bmatrix}$$

where $u(k-1|k)$ is the aplied last control input, N_p, N_c are the prediction and control horizons respectively, $X_k \in \Re^{n_x N_p x1}$, $F_k \in \Re^{n_x N_p x1}$, $G_k \in \Re^{n_x N_p x N_c}$, $\Delta U_k \in \Re^{N_c x1}$,$g_k \in \Re^{n_x N_p x1}$ and $D_k \in \Re^{n_x N_p x2}$. Here, n_x represents number of states.

Calculating the matrices F_k, G_k, g_k is challenging, since the ΔU_k is unknown until optimization problem is solved. The problem is tackled by utilizing ΔU_k belongs to previous time step to calculate these matrices.

$$\begin{aligned} \Delta u_{k|k} &\triangleq \Delta u_{(k-1)+1|k-1} \\ \Delta u_{k+1|k} &\triangleq \Delta u_{(k-1)+2|k-1} \\ &\vdots \\ \Delta u_{k+N_p-1|k} &\triangleq \Delta u_{(k-1)+N_p-1|k-1} \\ \Delta u_{k+N_p|k} &\triangleq 0 \end{aligned} \tag{2.24}$$

For the propagation equation 2.23, the quadratic cost function can be defined as;

$$J = X_k^T Q X_k + \Delta U_k^T R \Delta U_k \tag{2.25}$$

where $Q \in \Re^{n_x N_p x n_x N_p}$ and $R \in \Re^{N_c x N_c}$.

Then, the performance objective function can be rewritten by substituting equation (2.23) into (2.25)

$$
\begin{aligned}
J =& (F_k + G_k\Delta U_k + g_k + d_k)^T Q(F_k + G_k\Delta U_k + g_k + d_k) + \Delta U_k^T R\Delta U_k && (2.26)\\
=& F_k^T Q(F_k + G_k\Delta U_k + g_k + d_k)\\
&+ \Delta U_k^T Q(F_k + G_k\Delta U_k + g_k + d_k)\\
&+ g_k^T Q(F_k + G_k\Delta U_k + g_k + d_k)\\
&+ d_k^T Q(F_k + G_k\Delta U_k + g_k + d_k) + \Delta U_k^T R\Delta U_k\\
J =& (F_k + g_k + d_k)^T Q(F_k + g_k + d_k) && (2.27)\\
&+ (F_k + g_k + d_k)^T QG_k\Delta U_k && (2.28)\\
&+ (G_k\Delta U_k)^T Q(F_k + g_k + d_k) && (2.29)\\
&+ \Delta U_k^T Q(G_k^T Q + R)\Delta U_k && (2.30)
\end{aligned}
$$

Some mathematical tricks can simplify the cost function that are given in the equations 2.27-2.30. The first simplification is derived in equation 2.31 because equations 2.28 and 2.29 are transpose of each others. Twice one of them is equal to their total since the results of these equations are scalar. The matrix product 2.27 can be removed from the cost function as long as this product is calculated at the beginning of optimization.

$$
\begin{aligned}
2(F_k + g_k + d_k)^T QG_k =& (F_k + g_k + d_k)^T QG_k && (2.31)\\
&+ (G_k)^T Q(F_k + g_k + d_k)
\end{aligned}
$$

Eventually, the cost function takes the shape of equation 2.32.

$$
J = \Delta U_k^T Q(G_k^T Q + R)\Delta U_k + 2(F_k + g_k + d_k)^T QG_k\Delta U_k \tag{2.32}
$$

The limit on control input magnitude was defined in inequality 2.10. They take place in constraints of the quadratic programming (QP). The inputs and its rate is constrained symmetric minimum and maximum limits as:

$$
\begin{aligned}
U_{\min} \leq U_k \leq U_{\max}\\
U_{\min} \leq U_{k-1} + I_{lt}\Delta U_k \leq U_{\max}
\end{aligned} \tag{2.33}
$$

where, $U_{k-1} \in \Re^{N_c x 1}$ and $I_{lt} \in \Re^{N_c x N_c}$ are given as following

$$U_{k-1} = u(k-1|k)1_{N_c}$$
$$I_{lt} = \begin{bmatrix} I & 0 & \cdots & 0 \\ I & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ I & I & \cdots & I \end{bmatrix} \tag{2.34}$$

The increment on the control input constraint as follows

$$\Delta U_{\min} \leq \Delta U_k \leq \Delta U_{\max} \tag{2.35}$$

where $\Delta U_{\min}$,$\Delta U_{\max}$, $U_{\min}$, and $U_{\max}$ are the lower and upper bounds of the input vectors.

Finally, we can define the quadratic programming problem as;

$$\begin{aligned} \textbf{min} \quad & \Delta U_k^T W \Delta U_k + c^T \Delta U_k \\ \textbf{subject to} \quad & E\Delta U_k \leq b \end{aligned} \tag{2.36}$$

where

$$W = (G_k^T Q G_k + R),$$
$$C = 2(F_k + g_k + d_k)^T Q G_k$$
$$E = \begin{bmatrix} -I_{lt} \\ I_{lt} \\ -I_1 \\ I_1 \end{bmatrix} \quad I_1 = I_{N_c \times N_c}$$
$$b = \begin{bmatrix} -U_{min} + u(k-1|k)1_{N_c} \\ U_{max} + u(k-1|k)1_{N_c} \\ -\Delta U_{min} \\ \Delta U_{max} \end{bmatrix}$$

2.2.2 Non-linear model predictive control with unknown target acceleration

In the article [12], its authors define the quadratic programming by adding target acceleration as a constraint unless defined problem in the previous section. The input vector is constituted as in the form 2.37.

$$\begin{bmatrix} \Delta U_k \\ d_k \end{bmatrix} \tag{2.37}$$

By doing so, the QP problem is formulated as follows;

$$\begin{aligned} \min \quad & \begin{bmatrix} \Delta U_k \\ d_k \end{bmatrix}^T W \begin{bmatrix} \Delta U_k \\ d_k \end{bmatrix} + c^T \begin{bmatrix} \Delta U_k \\ d_k \end{bmatrix} \\ \text{subject to} \quad & E \begin{bmatrix} \Delta U_k \\ d_k \end{bmatrix} \leq b \end{aligned} \tag{2.38}$$

where

$$W = \begin{bmatrix} G_k^T Q G_k + R & G^T Q \\ Q G_k & Q \end{bmatrix}, \quad C = \begin{bmatrix} 2G_k^T Q(F_k + g_k) \\ 2Q(F_k + g_k) \end{bmatrix}$$

$$E = \begin{bmatrix} -I_{lt} & 0_1 \\ I_{lt} & 0_1 \\ -I_1 & 0_1 \\ I_1 & 0_1 \\ 0_2 & -I_2 \\ 0_2 & I_2 \end{bmatrix}, \qquad b = \begin{bmatrix} -U_{min} + u(k-1|k)1_{N_c} \\ U_{max} + u(k-1|k)1_{N_c} \\ -\Delta U_{min} \\ \Delta U_{max} \\ -d_{min} \\ d_{max} \end{bmatrix}$$

$$0_1 = 0_{N_p \times N_p n_x}, \qquad 0_2 = 0_{N_p n_x \times N_p}, \qquad I_2 = I_{N_p n_x}$$

3. DATA GENERATION, NETWORK DESIGN AND TRAINING

This chapter focuses on two topics which are data generation and Neural Network (NN) design. In the first part, the data generation process from manoeuvre selection to preprocessing is explained to generate data composed of target states and future accelerations. The second part determines the desired long short-term memory (LSTM) architecture, explains the training procedure and shows results.

3.1 Data Generation

To optimize the problem to be defined in the Chapter 2, a NN architecture predicting the target accelerations is required a specific dataset. This set should consist of target states as NN's inputs and missile's acceleration as NN's output. This kind of data mapping problem is called regression.

In this section, manoeuvres, used in data generation, are introduced and explained. In this context, it explains manoeuvres generation procedures with numerous combination of these. After data generation, it explains that data regulation to make data handleable for sequence-to-sequence LSTM network architecture.

3.1.1 Manoeuvres

Because they represent the real-world scenario, level-flight, coordinated-turn and weave manoeuvres are preferred to represent the agile characteristic of targets. An attacking target makes some manoeuvres for confusing radar when approaching an asset or escaping from the weapon system. Three manoeuvres combination can represent all of these intentions. When a target is escaping from radar detection zones, it tries to escape with the shortest trajectory, which can be explained as the first coordinated turn until radar system stays behind and then goes with level-flight. Other intentions also can be explained similarly.

A simulation environment which uses only the target's states of kinematic equations 2.2 has created the desired data. During the simulation, the desired manoeuvres described the following subsections define the input of kinematic equations.

3.1.1.1 Level-flight

In the level-flight, the target goes straight without warping its way. During manoeuvre, kinematic equation takes acceleration input a_T as zero. Figure-3.1 demonstrates the level-flight along 3.5 second with $100 m/s$ speed.

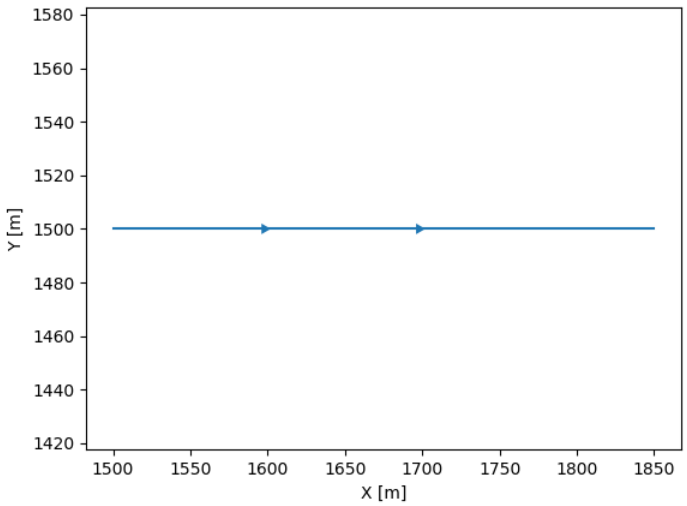

Figure 3.1: Level flight.

3.1.1.2 Coordinated turn

When target executes a coordinated turn, it takes acceleration input a_T in range [-25,25] representing g-force on the target. Figure-3.2 shows a trajectory of a target executing left coordinated-turn with 15G (15 g-force) lateral acceleration.

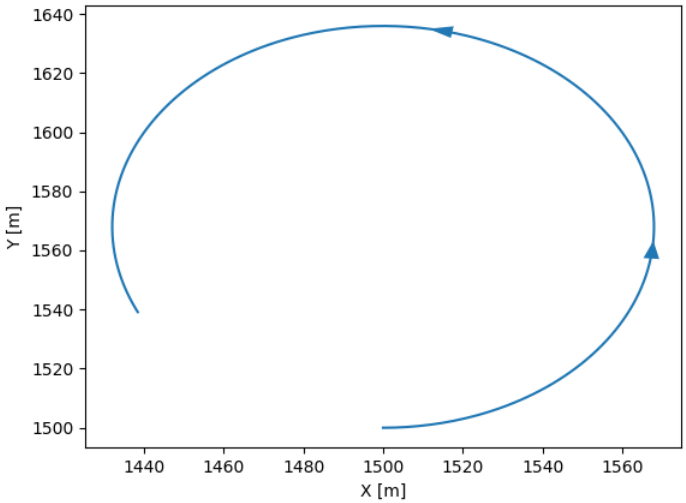

Figure 3.2: Coordinated-turn.

3.1.1.3 Weave manoeuvre

Weave manoeuvre consists of coordinated-turns commutated direction periodically. This manoeuvre can be diversified with its parameters which are acceleration in range [-25,25] G and period in range [1,8] seconds. In Figure-3.3, weave manoeuvre executed with 4G lateral acceleration and 3 second period is depicted.

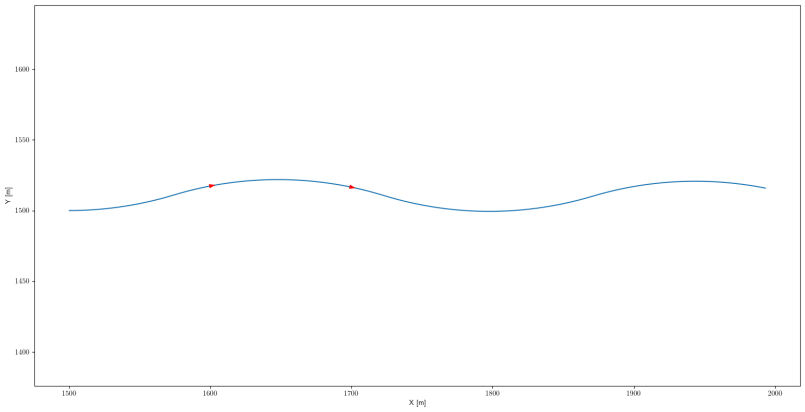

Figure 3.3: Weave manoeuvre.

3.1.2 Attack pattern generation

One of the main problems of generation attack pattern is that there is some ambiguity about what kind of attack pattern the target is making. For this reason, the data generator simulates a vast amount of different ordered manoeuvres for compensating as much as all possible attack pattern. All initial conditions and manoeuvres' parameters

are changed to cover all possible manoeuvre space in every simulation. Thus, a dataset has been produced, including all manoeuvre combinations for the whole range of their parameters. We expect that the acceleration of target performing defined manoeuvres with its parameters ranges should be predicted with high accuracy if we produced sufficient trajectories.

The simulation produces the position and velocity vectors with respect to the origin and acceleration inputs. The state vectors is;

$$\mathrm{X} = [\mathrm{x}_t, \mathrm{x}_{t-2}, \mathrm{x}_{t-3}, ..., \mathrm{x}_{t-n}]$$

where

$$\mathrm{x}_t = [x^t, y^t, v_x^t, v_y^t]$$

X vector contains from $t-n$ time t time position and velocity vectors, n is the size of observation history.

The labels of acceleration are;

$$\mathrm{A}_T = [a_{Tt}, a_{Tt-2}, a_{Tt-3}, ..., a_{Tt-n}]$$

where

a_{Tt} is lateral acceleration for t-th time step

Algorithm 1 reveals the data generation process to be conducted with time step 0.02 second along 30 second simulation time.

Algorithm 1 Attack pattern generation algorithm

Require: number_of_simulation, manoeuvre_probablities = [0.33, 0.33, 0.33], simulation_time

iteration_simulation = 0
iter_simulation_time = 0
Δt = 0.02
sample `number_of_simulation` number of velocities from uniform distribution in the range of [100,300] m/s.
sample `number_of_simulation` number of direction angle from uniform distribution in the range of [0,360] degree.
repeat
  Use `iteration_simulation` th velocity and movement direction
  for t_{sim}=0 : $t_{sim} \leq$ `simulation_time` : $t_{sim} = +\Delta t$ **do**
    if The previous manoeuvre is finish **then**
      Sample manoeuvre from `manoeuvre_probablities`
      Also sample the manoeuvre parameters whose ranges defined in **Manoeuvres** section.
    end if
    Execute sampled manoeuvre.
  end for
  Log all states (X) and executed acceleration (A_T)
until `iteration_simulation >= number_of_simulation`

Figure-3.4 and Figure-3.5 show the trajectories generated by algorithm-1. We can see manoeuvres how to be executed in Figure-3.4 whereas, Figure-3.5 is depicted the expanded trajectories initialized with various speed and direction.

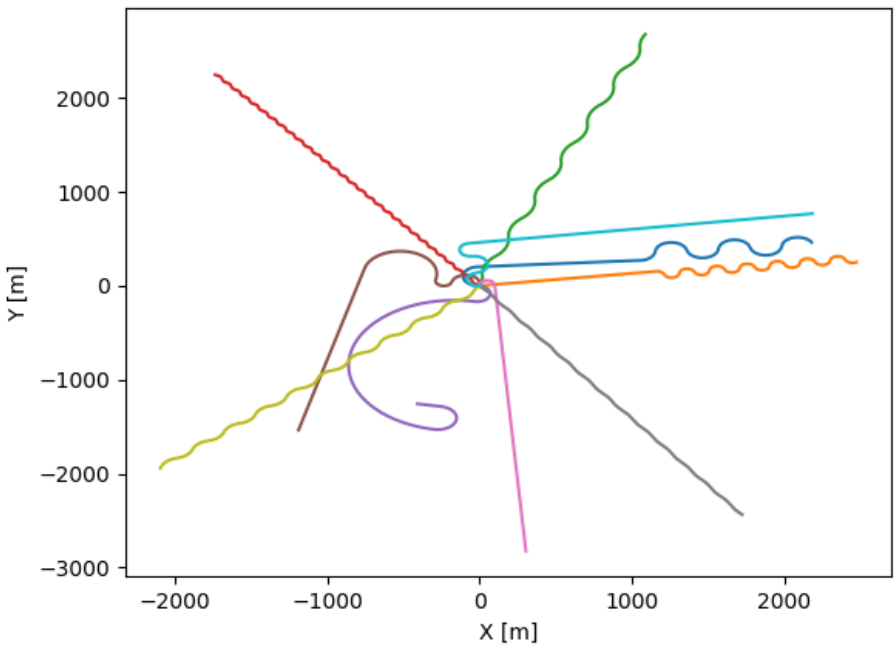

Figure 3.4: 10 generated trajectories.

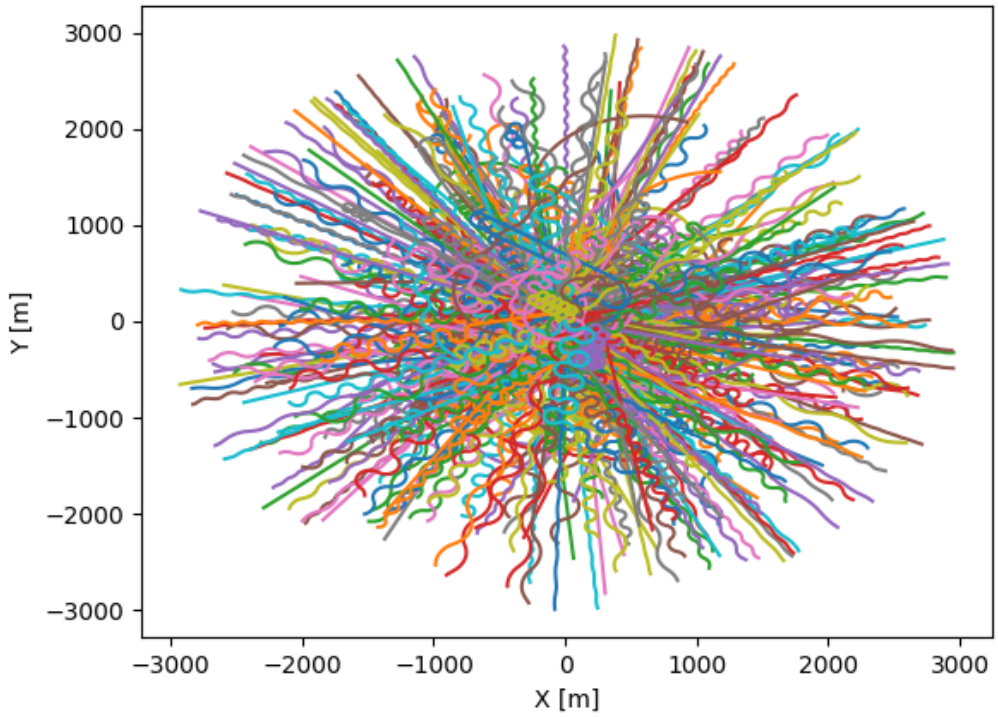

Figure 3.5: 500 generated trajectories.

3.1.3 Preprocessing

In the previous section, generated data has a shape of $[1500, 500 * 360, 5]$. The first dimension represents the number of simulation time steps calculated with simulation time 30 second over time-step 0.02 second. As we said before, trajectories should be generated for every initial direction. So the second dimension has 500*360 elements representing the total number of trajectories, 500 simulations for each degree of 360 degrees. The last dimension contains four target states with extra acceleration applied at the time.

A neural network needs a particular data format to improve learning performance or deducing the wright relationship between states and the acceleration label. As we said beforehand, this acceleration prediction problem is a sequence-to-sequence regression problem. In the literature, this kind of data fitting problem has been handled by a long short-term memory (LSTM) neural network, which will be explained in the next section. Sequence-to-sequence LSTM architecture uses states history as input and then produces the accelerations for more than one future time step. For these reasons, data is labelled as shown in Figure-3.6. When the black line represents the simulation flow

starting from left to right, the subscripts [t-lb, t-lb+1, ..., t-3, t-2, t-1, t, t+1, t+2, t+3, ..., t+N_p-2, t+N_p-1, t+N_p] mean time steps. Here, the **lb** is abbreviation of "look back" to express the amount of previous time steps given to LSTM NN. N_p is prediction horizon as we said beforehand. In the figure, blue box is put on the state x_t and input a_t at current time step. The red box takes in the past states for LSTM input when green box takes in acceleration commands during prediction horizon as LSTM outputs. In other word the red boxed states and the green boxed inputs are a sample of new dataset. All generated trajectories organized in same manner by current time step sliding one step over time zone for a new sample.

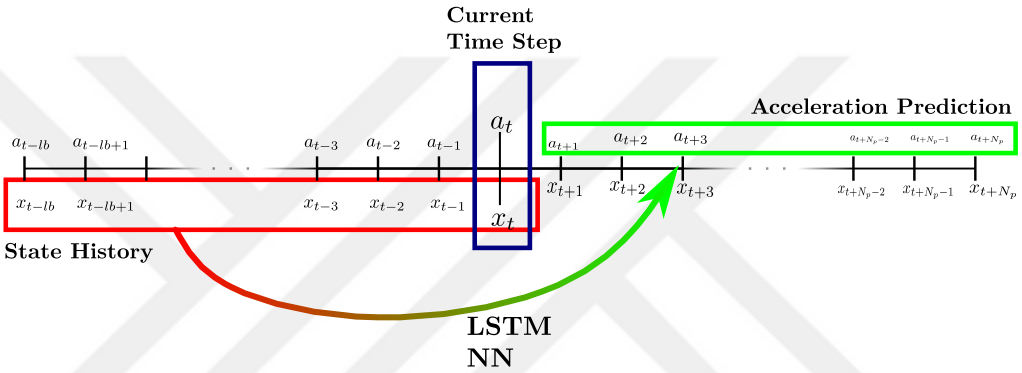

Figure 3.6: Reshaping inputs and its labels.

The next step of preprocessing is centring the initial position of samples by subtracting all time steps' x and y values from first time step. Therefore, all samples given to LSTM start from same position $(0, 0)$ and also the training stability is ensured.

The other steps of preprocessing are normalizing and adding noise to data. States and labels are normalized between each other. Adding zero-mean Gaussian noise with a 0.01 standard deviation increases the stability of LSTM result for incorrect inputs. Figure-3.7 depicts some normalized and noisy trajectories.

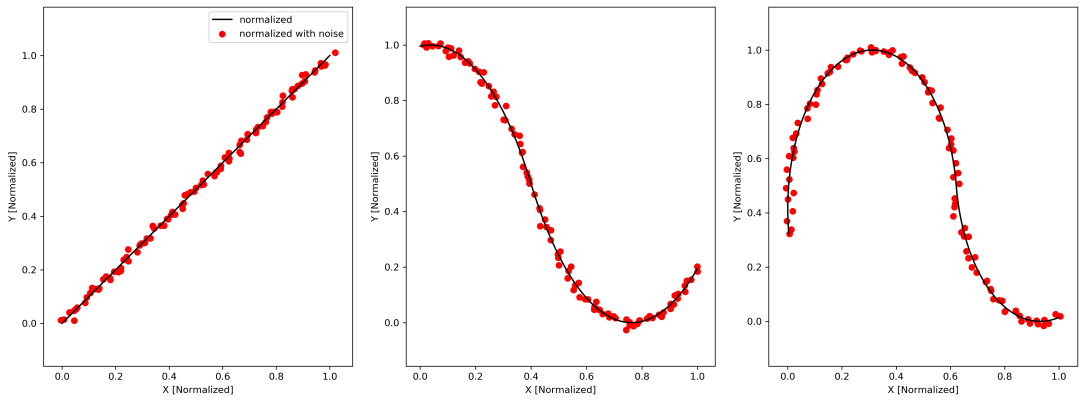

Figure 3.7: Normalized and noisy samples.

3.2 Deep Neural Network Design

This section explains the advantages of naive LSTM architecture against dense layer and recurrent neural networks. It continues with the explanation of encoder-decoder LSTM architecture, manoeuvre based LSTM (MLSTM) and it also explains why we choose MLSTM architecture. Then the chosen architecture's training period is given. In the end, the section concluded with the loss value of predictions and discussion about its performance.

3.2.1 Sequential data fitting problem

A dense neural network (NN) is doing a good job when data is stationary, not changing with time. Except that some curious studies, because of the challenging process of dense NNs' training and fitting it is not preferring for time-series data fitting problem. So, variant of recurrent neural networks are developed and widely used in that field.

3.2.1.1 Recurrent neural network (RNN)

A recurrent layer is added to a dense neural network for transferring the influence of earlier steps to the current step. This layer is called a hidden layer. Figure-3.8 shows the folded and unfolded RNN. Equation 3.1 gives input-output relationship as mathematically. Although its learning performance is better than the dense NN, It can be difficult to train RNNs to solve problems that require learning long-term temporal

dependencies.

$$\begin{aligned} h_t &= f_w(h_{t-1}, x_t) \\ &= tanh(w_{hh}h_{t-1} + W_{xh}x_t) \\ y_t &= W_{hy}h_t \end{aligned} \tag{3.1}$$

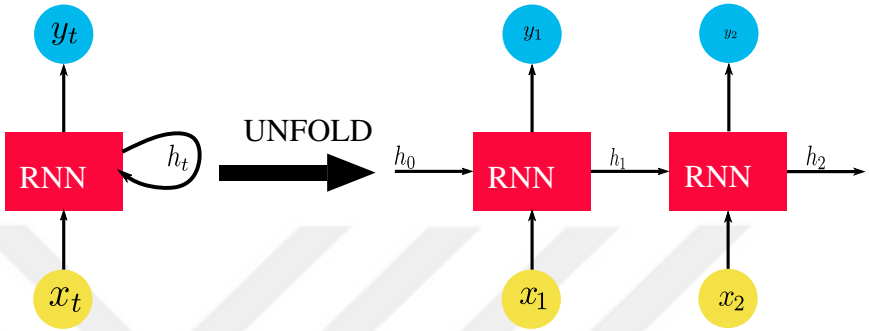

Figure 3.8: RNN data flow.

3.2.1.2 Long short-term memory (LSTM)

Because the RNN suffers from connecting the information for long sequential data, researchers introduced LSTM. Then, it has become famous for learning the connection between long-term temporal depended data. In Figure-3.9 data flow of LSTM is depicted.

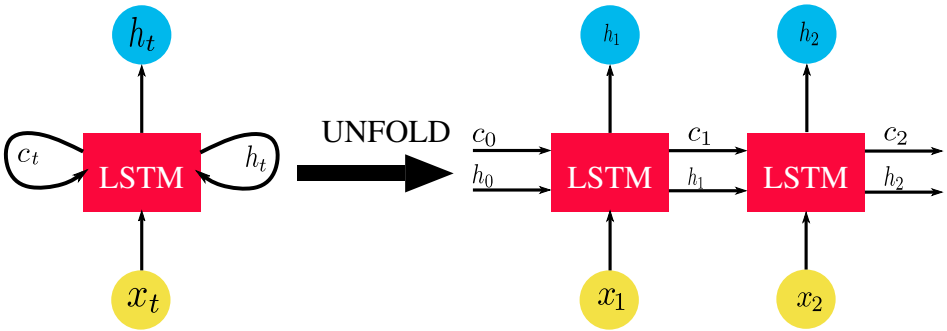

Figure 3.9: LSTM data flow.

LSTM's cell state c_t brings information coming from previous steps. Some operations add information to the cell or take out from the cell. These special operations are called gate. These gates are "forget gate" deciding whether to erase the cell, "input gate" whether to add to the cell, "gate gate" specifying how much to add to the cell, "output gate" specifying how much to exhibit the cell. The equation 3.2 shows the necessary mathematical operations for these gates.

$$\begin{aligned}
\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} &= \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \\
c_t &= f \odot c_{t-1} + i \odot g \\
h_t &= o \odot \tanh(c_t)
\end{aligned} \tag{3.2}$$

where, σ is the sigmoid function, W is NN's weights, $\odot$ is the element-wise product. Figure-3.10 depicts its interior architecture. There are three inputs in the architecture: x_t is the state, h_{t-1} is the hidden states to be initialized randomly and lastly c_{t-1} is the cell bringing information from beginning to end. The hidden state combines with the state, and then it is multiplied by NN weights (W). The multiplication's output pass through the gates. The output of the forget gate decides how much knowledge flow throughout LSTM by element-wisely multiplication with cell state c_{t-1}. The summation of the reduced cell state and the other knowledge coming from element-wisely multiplication of the input (i) and gate(g) create the next time step cell states c_t. The element-wise multiplication of the knowledge representing previous sequence history (c_t) and the knowledge representing the previous state (o) generates the next hidden state (h). The LSTM layers' outputs are taken from directly hidden states (h_1, h_2), as shown with blue circle in Figure-3.9.. All of these data flow ensures the LSTM to learn sequential data better than the other NN architecture.

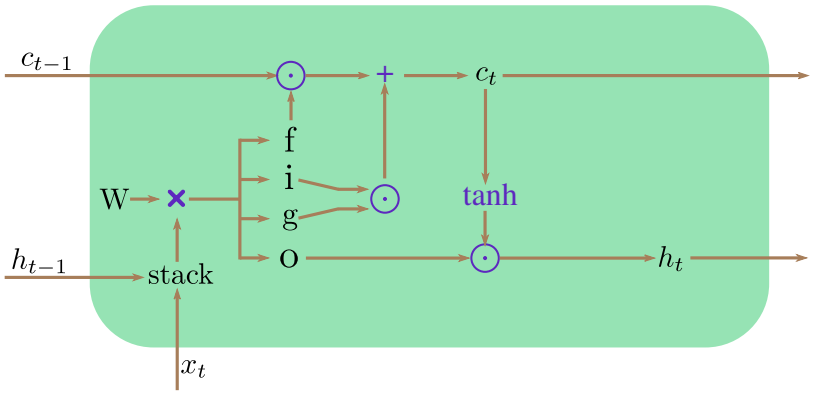

Figure 3.10: LSTM interior structure.

3.2.2 Selected LSTM architectures

Before we decide which architecture to use, we test several of them such that simple Recurrent Neural Network (SimpleRNN), encoder-decoder (LSTM), manoeuvre based LSTM (MLSTM). Simple RNN is ordered RNN architecture as illustrated in Figure-3.8.

3.2.2.1 Encoder-decoder LSTM

As you can see in Figure-3.9, classical LSTM has to employ same number of inputs (x_1, x_2) and outputs (h_1, h_2). As a solution to the fixed input-output size issue, researchers suggest encoder-decoder LSTM architecture. In Figure-3.11, when encoder takes input the temporal states x_1, x_2, it gives hidden sates h_2 as output. The decoder uses the encoder's outputs to produce a different number of outputs from encoders. Due to this structure, it is not any longer necessary to choose the same input-output size. In our work, we exploit the property of encoder-decoder structure since we use long states history for feeding NN, unlike we expect short time prediction.

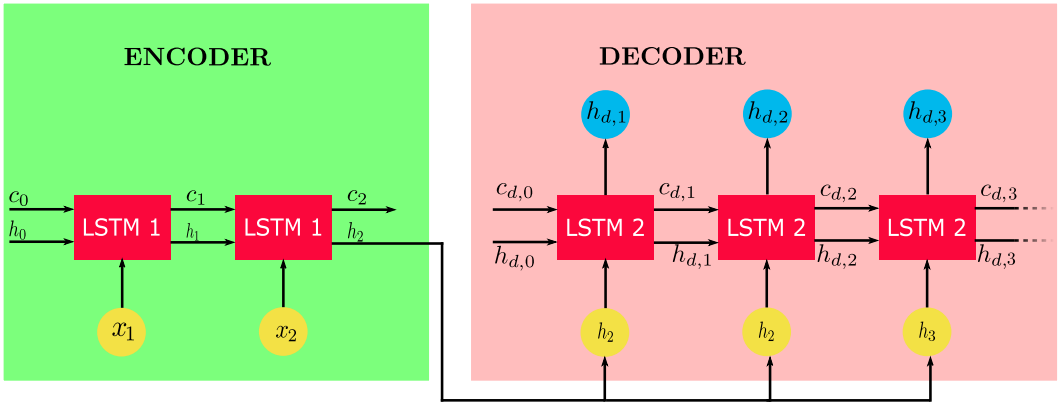

Figure 3.11: Encooder-Decooder LSTM.

3.2.2.2 Manoeuvre based LSTM (MLSTM)

In our work, we use a variant of MLSTM given in [27], because its authors predict probabilistic position distribution of a vehicle despite we only need an acceleration prediction. Our MLSTM variant is given in Figure-3.12. This structure incorporates a manoeuvre classifier into encoder with multi decoder architecture. In our version, as you can see in Figure-3.12, a trained manoeuvre classifier LSTM branch is directly concatenated with encoder output. Then three split LSTM decoder predict accelerations with additional dense layer.

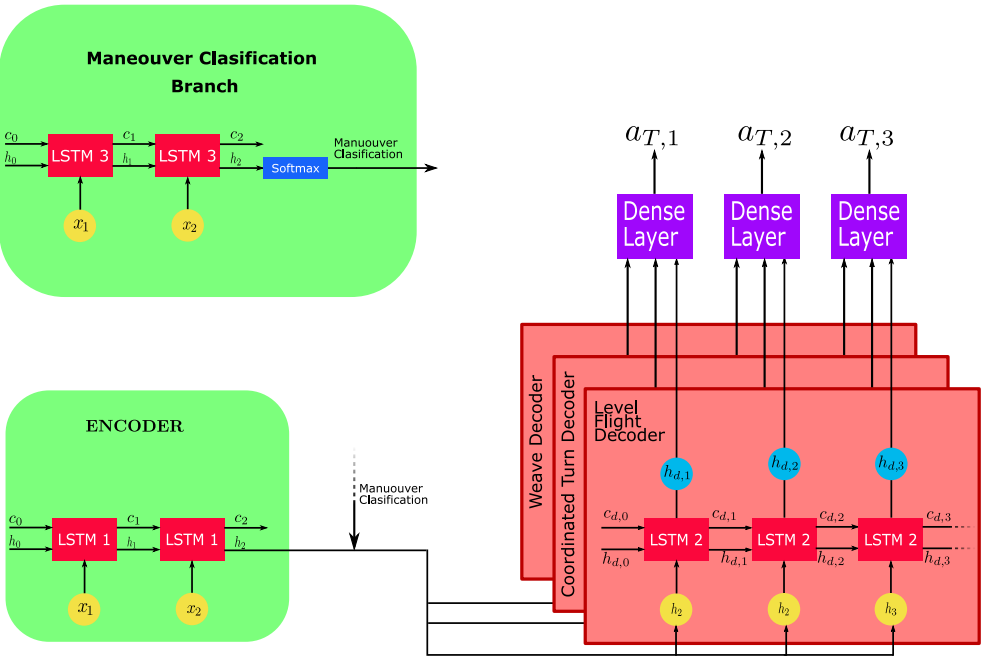

Figure 3.12: Our variant of manoeuvre based LSTM.

3.2.3 Training, Results and Comparison

The given deep neural network (DNN) model is trained in the Keras environment " [32]". After some manual tuning of model parameters, the obtained results show that the increasing number of neurons in a single layer can decrease the total model loss and enhance the model performance. However, adding an extra layer neither decreases the loss functions nor increases the training speed.

To avoid over-fitting and reduce noise-sensitivity, we added dropout to models at the amount of 20% for recurrent layers and 10% for dense layers. After several trials and errors, the results show that Adagrad optimization method yields faster lower loss.

The models' number of trainable parameters vary since their architecture differ. The comparison of the models has been made for three stages of neuron numbers: small, medium, and large. Each stage is valid for own model that is independent of other models' size and neuron numbers. Because all model contains the same encoder-decoder architecture, we can define a general stage selection strategy. For the small-sized models, encoder contains two layers with 64 and 32 units respectively from start to end. Then inverse ordered layers with units create a decoder. The large-sized models contain three layers with 128, 64, 32 units for the encoder and three layers with 32, 64, 128 units for decoder layers. According to our trials-errors for several models, the models can be improved to model the starting layer with 128 units after that no significant enhancement observed.

To ensure the models' effectiveness, we compare simple RNN, LSTM and M-LSTM models with various size. Table -3.1 gives the training and test results of these model. In the table, means square error (MSE) loss value is used for comparison criterion. Due to the fact that the accelerations label to be predicted in the range $[-25, 25]$ are normalized to $[0, 1]$ range, the results show the loss value for the comparison of real and predicted accelerations in the range $[0, 1]$.

Table 3.1: Training and test losses of the models.

	SimpleRNN		LSTM		M-LSTM	
Number of Parameters	Train	Test	Train	Test	Train	Test
Small	0.0304	0.0373	0.0121	0.0191	0.0175	0.0156
Medium	0.0258	0.0328	0.0169	0.0152	0.0158	0.0142
Large	0.0376	0.0386	0.0162	0.0096	0.0148	0.0091

According to this table, we can infer from lower test loss values that all models can generalize to manoeuvres not used in training. Also, LSTM and M-LSTM architectures' are outstanding to simple RNN since their loss value is almost half. Furthermore, MLSTM is a little bit better than the LSTM. Therefore we have used the more accurate model MLSTM.

Table 3.2 shows various acceleration prediction results. We can infer from these results that the model target acceleration prediction is highly accurate when considering its agile manoeuvres' test set. It should be mentioned that the model also has the capability of capturing switches among manoeuvre types and prediction of associated accelerations. This property ensures that our proposed method NMPC with target acceleration prediction has a significant advantage over alternatives.

Table 3.2: Example prediction results on test data.

	10 Horizon		10 Horizon		20 Horizon		20 Horizon	
Time Step	Real	Prediction	Real	Prediction	Real	Prediction	Real	Prediction
k	-16.21	-16.88	-24.67	-25	14.829	14.572	15.182	14.749
k+1	-16.21	-15.74	-24.67	-25	14.829	14.643	15.182	13.991
.	16.21	-8.67	-24.67	-25	.	.	.	.
.	16.21	12.18	-24.67	-11.08	.	.	15.182	12.97
.	16.21	19.25	24.67	20.57	.	.	15.182	9.227
.	16.21	17.84	24.67	25.11	14.829	13.624	15.182	-0.084
.	16.21	17.54	24.67	23.83	14.829	13.61	.	.
.	16.21	17.26	24.67	23.91	14.829	13.578	-15.182	-15.699
k+n	16.21	17.22	24.67	24.19	14.829	13.561	-15.182	-16.032

4. SIMULATION RESULTS

Simulation studies have been conducted to compare the result of promised algorithm NMPC-TAP with previously presented methods such as PN, APN and NMPC. Each guidance algorithms run for 100 Monte-Carlo simulations with 5% noise. The guidance algorithms' performances are evaluated for structural agile attacks. Table-4.1 shows shared simulation parameters for all attack case except that case-specific parameters are defined in the following figures.

Table 4.1: Simulation parameters.

Δt	0.02 [s]	d_{max}	$8g \times \Delta t[m/s^2]$
g	9.81 $[m/s^2]$	d_{min}	$-d_{max}$
u_{max}	25g	Δu_{max}	$0.025u_{max}$
u_{min}	$-u_{max}$	Δu_{min}	$-\Delta u_{max}$
q	[0,0,0,100]	R	I
$\theta_{M,0}$	0°	$\theta_{T,0}$	190°
$x_{M,0}$	0	$x_{T,0}$	1000
$y_{M,0}$	0	$y_{T,0}$	1000
V_M	150	V_T	100
$N' = 3$			

Here the parameters meaning that Δt is simulations time step, g is gravitational acceleration, $[u_{min}, u_{max}]$ are lateral acceleration limits, $[\Delta u_{min}, \Delta u_{max}]$ are lateral acceleration rate limits. θ, x and y are the initial value of heading angle, x and y positions respectively. N' is PN and APN gain. V also represents the constant velocity along simulations. Beside the simulation parameters, the optimization's parameter defined as $Q = diag([q, \dots q])$ and $q \in \Re^{n_x \times 1}$. The weights Q and R ensure the cost function in the equation 2.25 to penalize the V_λ and input rate.

An interception problem is simulated with target attacking combinations of three manoeuvres coordinated-turn, level-flight and weave. The target attack pattern is depicted in Figure-4.1. The manoeuvre is assumed that the target begins left

coordinated turn with 4G and 70° then makes 50 m level flight and weave manoeuvre with 8G and 4sec. period.

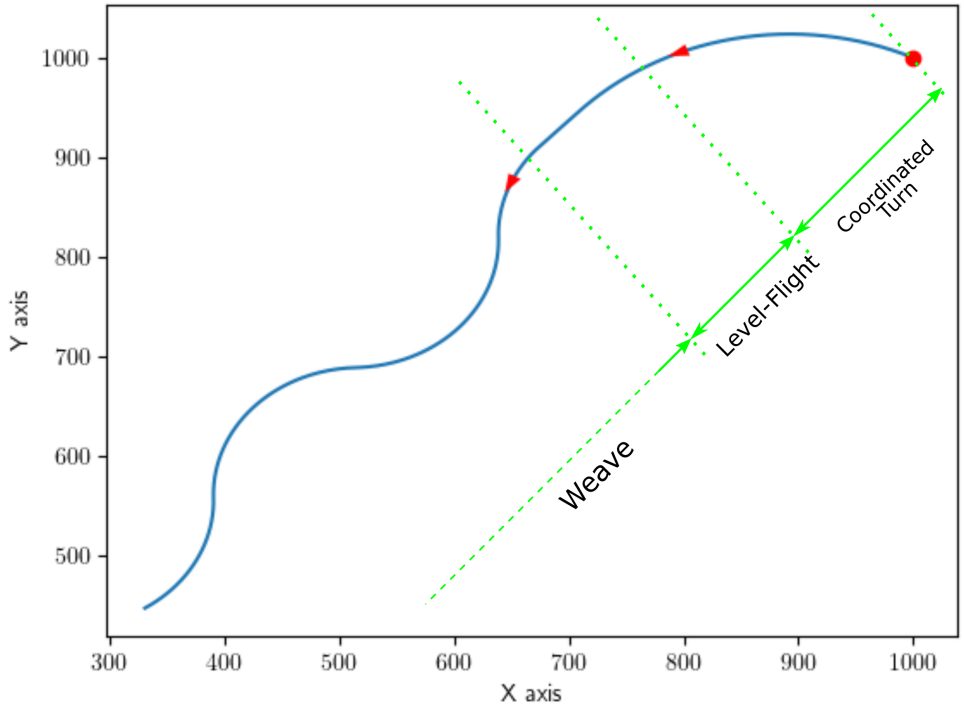

Figure 4.1: Structural agile attack.

The interception trajectories of all guidance methods against above agile manoeuvring target are depicted in Figure-4.2. We can infer from the trajectories that all guidance methods achieving target successfully although any methods cannot follow the target after interception point. Furthermore, Figure-4.3 shows that NMPC-TAP methods achieve the lowest miss distance value just before the interception of NMPC despite after interception of PN and APN laws.

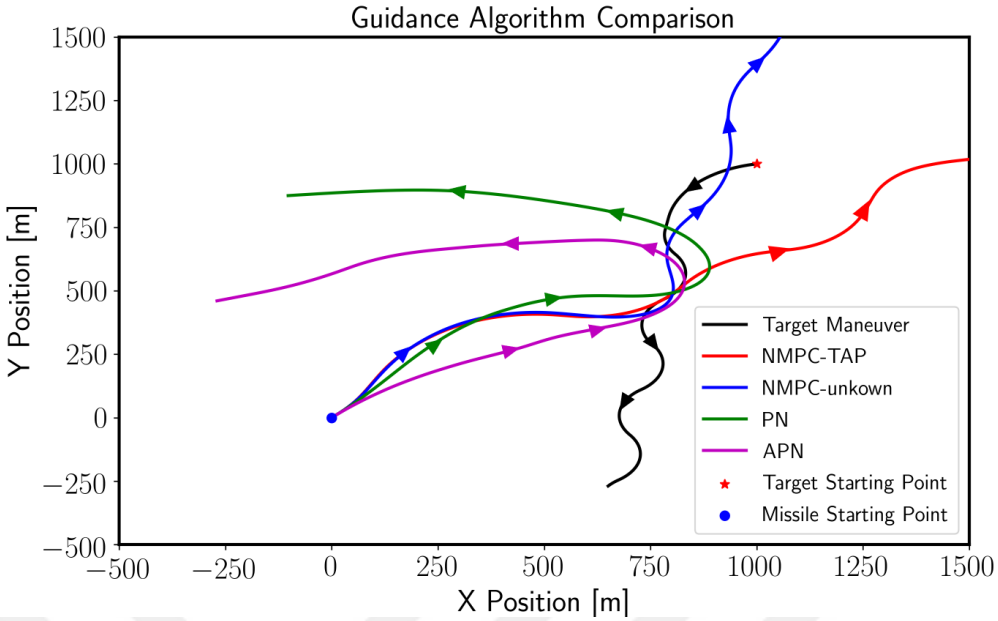

Figure 4.2: Missile trajectories generated by each guidance algorithms during missile-target interception.

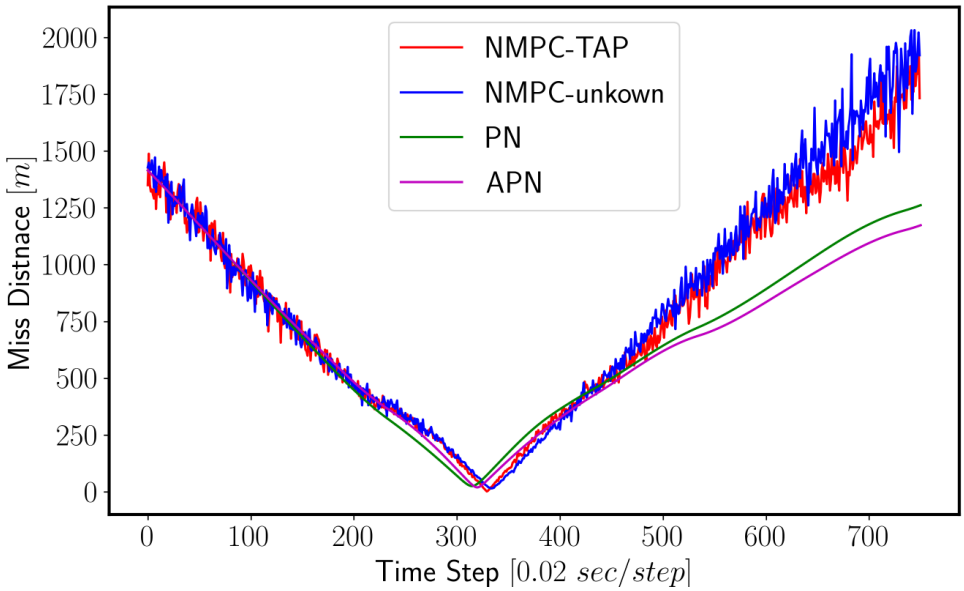

Figure 4.3: Miss distance values of guidance methods.

Since the quadratic NMPC based approaches primarily aim to penalize V_λ, relative velocity component perpendicular to the LOS, let's look at the V_λ given in Figure-4.4. Although the V_λ values of NMPC-unknown follow closely to NMPC-TAP until the target-interceptor distance getting too short at about 250-th time step, NMPC-unknown

cannot hold V_λ value small as NMPC-TAP still ensures lower V_λ value. Other guidance law PN and APN already have big V_λ values at all the times of interception.

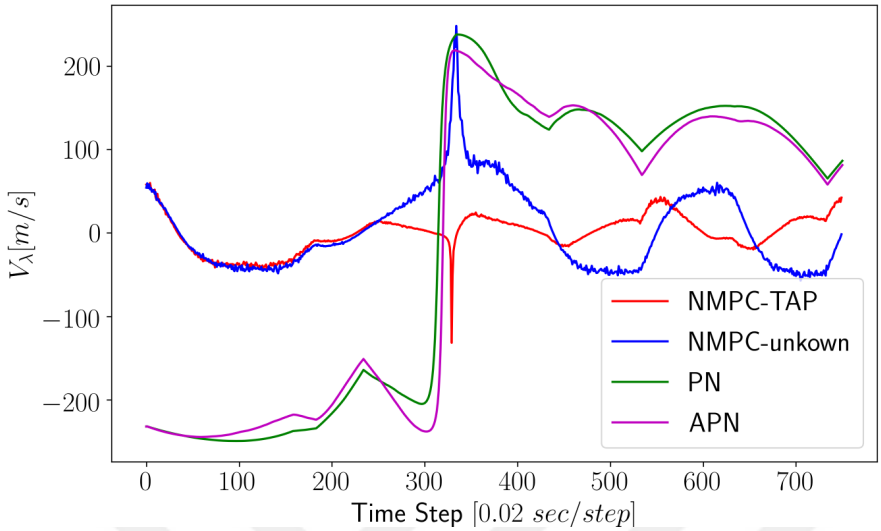

Figure 4.4: V_λ values during guidance.

Figure-4.5 shows the missile necessary acceleration inputs. The advantage of integrating target acceleration predictions in the optimization problem appears in the control input history. The predictions provide earlier direction change to NMPC-TAP than NMPC. Moreover inputs of both NMPC based methods stay in the acceleration limits.

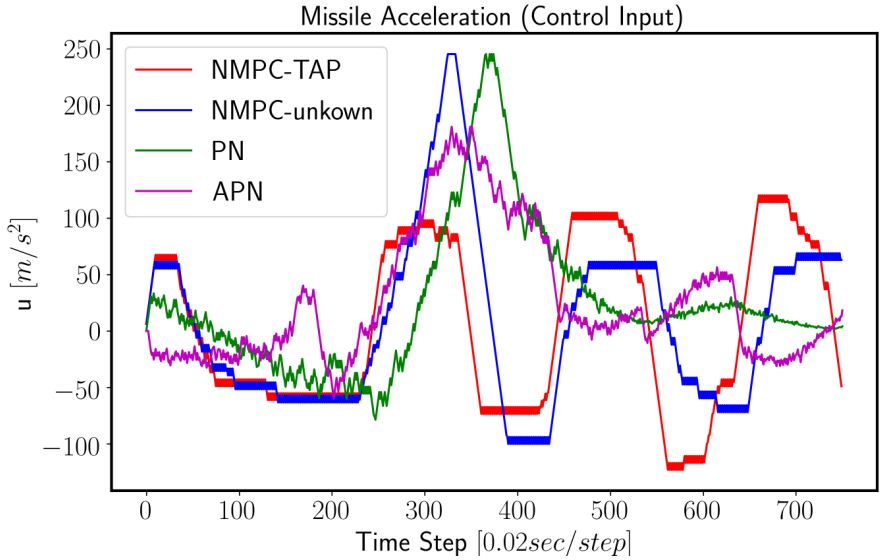

Figure 4.5: Missile acceleration values.

In order to increase the deep of the comparisons, 100 Monte-Carlo simulations were conducted for all guidance methods. The result of the simulation is compared according to miss distance mean and variance (MD_{mean}, MD_{STD}), interception time (IT) and average absolute input rates (AIR) and, besides NMPC based methods are simulated for different prediction horizons. During simulations, it is assumed that the prediction horizon is equal to the control horizon ($N_p = N_c$). Table-4.2 shows the simulations results. An increasing number of prediction time leads to a decrease in the MD_{mean} values NMPC based methods. Also, their miss distance means and standard deviations are lower than the PN laws. Because of the LSTM prediction, NMPC-TAP methods miss distance value is also lower than the NMPC as you can see in the table. When we look at the interception times, they are almost same of all algoritm if we don't take account of the maximum 0.13 second difference. According to the average absolute input rates, NMPC methods force more missile to accelerate.

Table 4.2: Results obtained by MC simulations.

	MD_{mean} (m)	MD_{STD} (m)	IT (s)	AIR (m/s^2)
APN	18.69	22.75	6.52	5.21
PN	26.378	7.72	6.52	5.3
Results for $N_p = 10$				
NMPC-TAP	7.2	12.91	6.54	5.64
NMPC	9.2	17.49	6.67	4.76
Results for $N_p = 30$				
NMPC-TAP	6.58	8.80	6.56	6.06
NMPC	11.22	6.19	6.57	6.06
Results for $N_p = 40$				
NMPC-TAP	3.57	2.70	6.62	6.025
NMPC	14.58	6.05	6.73	5.96

5. CONCLUSIONS

In this work, we combined NMPC with LSTM to get more precise guidance command for the two-dimensional interception problem. To able to do this, an NMPC model that involves target acceleration as a constraint into quadratic programming problem is converted to NMPC model with two inputs target and missile acceleration. The unknown target accelerations are predicted with the help of a variant of manoeuvre based LSTM. In this way, the desired NMPC model has been formulated.

On the LSTM training, we use manoeuvre combination of coordinated-turn, level-flight and weave. Manoeuvre starts a point goes with all directions of as in 360 degrees, so all approaching direction of target has been added to dataset. After that, dataset is preprocessed to compress the data values in the range of $[0, 1]$. The results of three DNN models show that MLSTM is the most accurate architecture among trained.

On the result, the promised NMPC-TAP method is compared to the other methods against target attacking with the agile structural manoeuvre. The NMPC-TAP's results ensure that the additional MLSTM predictions improve the performance of the NMPC.

REFERENCES

[1] **Zarchan, P.** (2012). *Tactical and strategic missile guidance*, American Institute of Aeronautics and Astronautics, Inc.

[2] **Yuan, L.C.L.** (1948). Homing and navigational courses of automatic target-seeking devices, *Journal of Applied Physics*, *19*(12), 1122–1128.

[3] **Guelman, M.** (1976). The closed-form solution of true proportional navigation, *IEEE Transactions on Aerospace and Electronic Systems*, (4), 472–482.

[4] **Guelman, M.** (1971). A qualitative study of proportional navigation, *IEEE Transactions on Aerospace and Electronic Systems*, (4), 637–643.

[5] **Shukla, U.S. and Mahapatra, P.R.** (1990). The proportional navigation dilemma-pure or true?, *IEEE Transactions on Aerospace and Electronic Systems*, *26*(2), 382–392.

[6] **Ghawghawe, S. and Ghose, D.** (1996). Pure proportional navigation against time-varying target manoeuvres, *IEEE Transactions on Aerospace and Electronic Systems*, *32*(4), 1336–1347.

[7] **Garber, V.** (1968). Optimum intercept laws for accelerating targets., *AIAA Journal*, *6*(11), 2196–2198.

[8] **Ghosh, S.**, **Ghose, D. and Raha, S.** (2014). Capturability of augmented pure proportional navigation guidance against time-varying target maneuvers, *Journal of Guidance, Control, and Dynamics*, *37*(5), 1446–1461.

[9] **Vathsal, S. and Sarkar, A.K.** (2005). Current trends in tactical missile guidance, *Defence Science Journal*, *55*(3), 265–280.

[10] **Clarke, D.W.**, **Mohtadi, C. and Tuffs, P.** (1987). Generalized predictive control—Part I. The basic algorithm, *Automatica*, *23*(2), 137–148.

[11] **Li, Z.**, **Xia, Y.**, **Su, C.Y.**, **Deng, J.**, **Fu, J. and He, W.** (2014). Missile guidance law based on robust model predictive control using neural-network optimization, *IEEE transactions on neural networks and learning systems*, *26*(8), 1803–1809.

[12] **Bhattacharjee, D.**, **Chakravarthy, A. and Subbarao, K.** (2020). Nonlinear Model Predictive Control based Missile Guidance for Target Interception, *AIAA Scitech 2020 Forum*, p.0865.

[13] **Best, R.A. and Norton, J.P.** (2000). Predictive Missile Guidance, *Journal of Guidance, Control, and Dynamics*.

[14] **Dionne, D.**, **Michalska, H. and Rabbath, C.A.** (2006). A predictive guidance law with uncertain information about the target state, *2006 American Control Conference*, IEEE, pp.6–pp.

[15] **Grewal, M.S. and Andrews, A.P.** (2014). *Kalman filtering: Theory and Practice with MATLAB*, John Wiley & Sons.

[16] **Url-1**, <https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python>, date retrieved : 18.12.2020.

[17] **Foo, P.H. and Ng, G.W.** (2007). Combining IMM method with particle filters for 3D maneuvering target tracking, *2007 10th International Conference on Information Fusion*, IEEE, pp.1–8.

[18] **Mazor, E.**, **Averbuch, A.**, **Bar-Shalom, Y. and Dayan, J.** (1998). Interacting multiple model methods in target tracking: a survey, *IEEE Transactions on aerospace and electronic systems*, *34*(1), 103–123.

[19] **Boers, Y. and Driessen, J.N.** (2003). Interacting multiple model particle filter, *IEE Proceedings-Radar, Sonar and Navigation*, *150*(5), 344–349.

[20] **Yuan, D.P. and Zheng, J.Y.** (2011). Interacting multiple model target tracking algorithm based on particle filtering, *Proceedings of 2011 IEEE CIE International Conference on Radar*, volume 2, IEEE, pp.1907–1910.

[21] **Sarkar, A.**, **Vathsal, S.**, **Sundaram, S. and Mukhopadhay, S.** (2005). Target acceleration estimation from radar position data using neural network, *Defence Science Journal*, *55*(3), 313.

[22] **Kim, B.**, **Kang, C.M.**, **Kim, J.**, **Lee, S.H.**, **Chung, C.C. and Choi, J.W.** (2017). Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network, *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, pp.399–404.

[23] **Gao, C.**, **Yan, J.**, **Zhou, S.**, **Varshney, P.K. and Liu, H.** (2019). Long short-term memory-based deep recurrent neural networks for target tracking, *Information Sciences*, *502*, 279–296.

[24] **Hochreiter, S. and Schmidhuber, J.** (1997). Long short-term memory, *Neural computation*, *9*(8), 1735–1780.

[25] **Zhang, P.**, **Ouyang, W.**, **Zhang, P.**, **Xue, J. and Zheng, N.** (2019). Sr-lstm: State refinement for lstm towards pedestrian trajectory prediction, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.12085–12094.

[26] **Park, S.H.**, **Kim, B.**, **Kang, C.M.**, **Chung, C.C. and Choi, J.W.** (2018). Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture, *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, pp.1672–1678.

[27] **Deo, N. and Trivedi, M.M.** (2018). Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms, *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, pp.1179–1184.

[28] **Sutskever, I.**, **Vinyals, O. and Le, Q.V.** (2014). Sequence to sequence learning with neural networks, *Advances in neural information processing systems*, *27*, 3104–3112.

[29] **Zhu, Z.**, **Xu, D.**, **Liu, J. and Xia, Y.** (2012). Missile guidance law based on extended state observer, *IEEE Transactions on Industrial Electronics*, *60*(12), 5882–5891.

[30] **Shtessel, Y.B.**, **Shkolnikov, I.A. and Levant, A.** (2009). Guidance and control of missile interceptor using second-order sliding modes, *IEEE Transactions on Aerospace and Electronic Systems*, *45*(1), 110–124.

[31] **Chai, R.**, **Savvaris, A. and Chai, S.** (2019). Integrated missile guidance and control using optimization-based predictive control, *Nonlinear Dynamics*, *96*(2), 997–1015.

[32] **Chollet, F.** *et al.*, (2015), Keras, `https://keras.io`.

CURRICULUM VITAE

Name Surname	:	Abdullah Sadık Satır

EDUCATION :

- **B.Sc.** : 2016, Kocaeli University, Faculty of Engineering, Mechanical Engineering
- **B.Sc.** : 2017, Kocaeli University, Faculty of Engineering, : Mechatronic Engineering