NTNU
Norwegian University of
Science and Technology

Waypoint-Following Guidance Based on Feasibility Algorithms

Tor Marius Jensen

Master of Science in Engineering Cybernetics
Submission date: June 2011
Supervisor: Tor Arne Johansen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

The main topic of the thesis is path planning for autonomous surface and ground vehicles. It is assumed that waypoints are given, and the problem is to find a suitable algorithm that provides a feasible path, given the vehicle dynamics and constraints.

1. Investigate algorithms that can create smooth, parameterized curves in $\mathbb{R}^2$, based on a predefined waypoint setup.
2. Investigate algorithms that analyse feasibility of paths with respect to vehicle dynamics.
3. Implement algorithms in MATLAB/Simulink and compare path tracking properties for different guidance algorithms.
4. Improve guidance strategy with feedforward from reference for waypoint-following vehicles.
5. Run numerical simulations in MATLAB/Simulink on Nomoto model and *Viknes 830* boat model to examine path feasibility.

Assignment given: 25th of January 2011
Supervisor: *Tor Arne Johansen*, ITK

Abstract

Unmanned vehicles have become more applicable due to extended research and more advanced technology through the last decades. The vehicles are versatile, hence studying ocean floor and reconnaissance and surveillance are popular areas of application. To perform unmanned operations, waypoint-following based on guidance, navigation and control systems are used in general. Unfortunately, not every waypoint setup guarantees a feasible trajectory as regards to the vehicle dynamics. Thus, algorithms analyzing the feasibility of the paths and suggesting solutions for infeasible paths have been proposed in this master's thesis.

Four different tangent vector calculation methods have been used to develop smooth, parameterized C^1-continuous curves as suitable paths between the successive waypoints. The appurtenant path curvature has been calculated and compared against the vehicle dynamics and constraints. Minimum vehicle turning radius is mapped directly to maximum trajectory curvature. When waypoint placement provides infeasible trajectories, three different algorithms of moving, adding or removing waypoints have been suggested to cope with the trajectory curvature.

For pure waypoint-following schemes without path generation, adapted velocity and acceptance circle radius have been applied. The adapted reference values are functions of the angle between the neighboring waypoints, and turns out to yield more efficient operations with respect to time and overshooting errors than with constant values. Employing path generation proves that the effect of preturns before approaching waypoints in addition to regulating velocity according to path curvature is beneficial to minimize deviation from waypoints. If, however, the trajectory is infeasible even after velocity adaption, one of the three path altering algorithms may be applied to provide a feasible path.

The simulations have been carried out in MATLAB/Simulink on two models; a first order Nomoto autopilot model and a more complex mathematical model of a *Viknes 830* unmanned surface vehicle. Despite simulating on boat models in $\mathbb{R}^2$, the path planning ideas and trajectory feasibility check can be employed on various types of unmanned vehicles, and -extended to $\mathbb{R}^3$.

Preface

The work presented in this thesis was carried out during spring 2011 at the *Department of Engineering Cybernetics*, NTNU, Trondheim, Norway. Writing the master's thesis concludes five years of higher education.

I would like to express my sincere gratitude to my supervisor Tor Arne Johansen for all his assistance and support throughout the work process. I would also like to give a special thanks to Vegard Evjen Hovstein, Eirik Evjen Hovstein and Arild Hepsø at *Maritime Robotics*; input on navigation systems and vehicle models from people that associate with guidance, navigation and control systems on a daily basis have been highly appreciated.

My interest-, competitive spirit- and urge for solving technical problems are credited to my parents; Åse and Trygve Jensen, and my siblings Inger Sofie Heggland and Lars Christian Jensen in particular.

I would also like to give thanks to some of my fellow student; Martin Hanger, Pål Skønberg Løvik, Andreas Jørgensen and Anders Freddy Johansen for insightful input, assistance and motivation on my work with my master's thesis. Finally, a special thanks goes to my girlfriend Helle Henriksveen for motivation and kind words when the work load felt staggering.

"I must have a prodigious quantity of mind; it takes me as much as a week sometimes to make it up." -Mark Twain

Table of Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 The Evolution of USVs . . . 2
- 1.3 Guidance, Navigation and Control . . . 3
- 1.4 Path Planning . . . 5
- 1.5 Path-Following . . . 6
- 1.6 Contributions . . . 6
- 1.7 Thesis Outline . . . 7

2 Interpolation **9**
- 2.1 Polynomial Interpolation . . . 9
- 2.2 Path Generation Based on Waypoints . . . 10
  - 2.2.1 Path Generation using Straight-Lines and Circles . . . 10
- 2.3 Spline . . . 10
- 2.4 Cubic Hermite Interpolation . . . 13

3 Guidance **17**
- 3.1 Pure Pursuit Guidance Algorithm . . . 17
- 3.2 LOS Guidance Algorithms . . . 18
  - 3.2.1 Enclosure-Based LOS Steering . . . 19
  - 3.2.2 Lookahead-Based LOS Steering . . . 21
- 3.3 WP Switching Algorithm . . . 23
- 3.4 Speed Profile . . . 23
  - 3.4.1 Speed Profile from Direct WP Layout . . . 24
  - 3.4.2 Acceptance Circle Radius Based on WP Layout . . . 26
  - 3.4.3 Curvature . . . 28
  - 3.4.4 Speed Profile Based on Path Curvature . . . 29
- 3.5 Feasible Path Selection . . . 31
- 3.6 Removing Waypoints Algorithm . . . 32
- 3.7 Inserting Waypoints Algorithm . . . 33
- 3.8 Relocating Waypoints Algorithm . . . 36

4 Models **39**
4.1 Choosing the Right Vehicle Model . . . 39
4.2 2nd Order Nomoto Model . . . 40
4.3 1st Order Nomoto Model . . . 40
4.4 Viknes 830 . . . 41
4.4.1 Heading Actuators . . . 41
4.5 Mathematical Model . . . 42
4.5.1 Actuators . . . 43
4.5.2 Environmental Aspects . . . 43
4.6 Heading Reference Model . . . 44

5 Control **49**
5.1 PID-Control . . . 49

6 Implementation **51**
6.1 Waypoint Generator . . . 51
6.2 Simulator Structure . . . 52

7 Results and Discussion **57**
7.1 Removing Waypoints . . . 57
7.2 Inserting Waypoints . . . 66
7.3 Relocating Waypoints . . . 74
7.4 Case Study . . . 82
7.4.1 Comparison of PP- and LOS-Based Guidance . . . 82
7.4.2 Validation of Nomoto Model . . . 86
7.4.3 Adapted Acceptance Circle Radius . . . 90
7.4.4 The Effect of Different Vehicle Velocity . . . 92
7.4.5 Adapted Vehicle Velocity and Acceptance Circle Radius 94
7.4.6 Path Generation with Cubic Splines . . . 96
7.4.7 Path-Following with Adapted Vehicle Velocity . . . 98
7.4.8 Path-Following with Regenerated Path . . . 101
7.4.9 Path-Following with Weather Disturbances . . . 103

8 Conclusion and Future Work **105**
8.1 Conclusion . . . 105
8.2 Future Work . . . 106

Bibliography **109**

A Continuity **113**

B Digital Appendix **115**

List of Figures

1.1 The USV *Mariner*. Courtesy of Maritime Robotics. 3
1.2 GNC block diagram. Adapted from [1]. 4
1.3 Comparison of feasible paths 5

2.1 Example of Cubic Spline based on four control points 12
2.2 Comparison of vector tangent calculation methods 16

3.1 Pure Pursuit guidance principle 18
3.2 Enclosure-based guidance principle. Adapted from [1] 20
3.3 Lookahead-based guidance principle. Adapted from [1] 22
3.4 Reference speed for $\alpha \in [-180°, 180°]$ 25
3.5 Waypoint layout resulting in three different α values 26
3.6 Acceptance circle radius based on α 27
3.7 Curvature for an arbitrary path 29
3.8 Reference velocity based on original curvature 30
3.9 Two ways of adding waypoints to decrease curvature 35
3.10 Curvature values before and after adding WPs 35
3.11 Principle of relocation of WPs 36
3.12 Comparison of curvature values for relocating WP algorithm . 37

4.1 Viknes 830. Courtesy of Viknes Båt og Service AS 42
4.2 Heading reference models. Adapted from [1] 47

6.1 Example of how waypoints can be set graphically 52
6.2 Overview of the simulator with the Nomoto model 54
6.3 Overview of the simulator with the 6 DOF Viknes boat model 55

7.1 Tangent method 1 and 2: Paths with removed WPs. $\kappa_v = 1.0$ 60
7.2 Tangent method 3 and 4: Paths with removed WPs. $\kappa_v = 1.0$ 61
7.3 Removing WPs: Path curvature values for $\kappa_v = 1.0$ 62
7.4 Tangent method 1 and 2: Paths with removed WPs. $\kappa_v = 0.4$ 63
7.5 Tangent method 3 and 4: Paths with removed WPs. $\kappa_v = 0.4$ 64
7.6 Removing WPs: Path curvature values for $\kappa_{vehicle} = 0.4$ 65

7.7 Tangent method 1 and 2: Paths with inserted WPs. $\kappa_v = 1.0$. 69
7.8 Tangent method 3 and 4: Paths with inserted WPs. $\kappa_v = 1.0$. 70
7.9 Inserting WPs: Path curvature values for $\kappa_v = 1.0$ 71
7.10 Tangent method 3 and 4: Paths with inserted WPs. $\kappa_v = 0.4$. 72
7.11 Inserting WPs: Path curvature values for $\kappa_v = 0.4$ 73
7.12 Tangent method 1 and 2: Paths with relocated WPs. $\kappa_v = 1.0$ 76
7.13 Tangent method 3 and 4: Paths with relocated WPs. $\kappa_v = 1.0$ 77
7.14 Relocating WPs: Path curvature values for $\kappa_v = 1.0$ 78
7.15 Tangent method 1 and 2: Paths with relocated WPs. $\kappa_v = 0.4$ 79
7.16 Tangent method 3 and 4: Paths with relocated WPs. $\kappa_v = 0.4$ 80
7.17 Relocating WPs: Path curvature values for $\kappa_v = 0.4$ 81
7.18 Comparison of different guidance strategies 84
7.19 Heading, heading reference and rudder values for PP guidance 85
7.20 Path comparison of Nomoto model and the Viknes 830 model 86
7.21 Comparison of minimum turning radius for boat models . . . 87
7.22 Comparison of velocity- and rudder values for boat models . . 88
7.23 Data from comparison of Nomoto and Viknes 89
7.24 Comparison of constant and adapted acceptance circle radius . 90
7.25 Trajectories for different vehicle velocities 92
7.26 Heading and velocity values for the trajectories in Figure 7.25 93
7.27 Comparison of trajectories with adapted radius and velocity . 94
7.28 Comparison of adapted and constant velocity/acceptance circle 95
7.29 Path-following vehicle trajectory with constant velocity 97
7.30 Path-following with constant velocity versus adapted velocity . 99
7.31 Comparison of constant and adapted velocity for path-following 100
7.32 Distinction in deviation for original and modified path 101
7.33 Vehicle trajectory with environmental disturbance 103
7.34 Heading, velocity and rudder data for trajectory w/disturbance 104

A.1 Orders of Continuity 114

Abbreviations

DOF	Degrees of Freedom
DP	Dynamic Positioning
GNC	Guidance, Navigation and Control
GPS	Global Positioning System
HSE	Health, Safety and Environment
LOS	Line-of-Sight
NED	North-East-Down
PID	Proportional–Integral–Derivative
PP	Pure Pursuit
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
USV	Unmanned Surface Vehicle
UUV	Unmanned Underwater Vehicle
UV	Unmanned Vehicle
WP	Waypoint

Chapter 1

Introduction

1.1 Motivation

Unmanned vehicles (UVs) have only been used for a limited number of application until relatively recently. However, through the last couple of decades, the technology and research on unmanned vehicles have put on speed. Unmanned vehicles can be divided into several subcategories depending on the area of application, that is; Unmanned Surface Vehicles (USVs), Unmanned Aerial Vehicles (UAVs), Unmanned Ground Vehicles (UGVs) and Unmanned Underwater Vehicles (UUVs).

Unmanned vehicles may be steered autonomously, semi-autonomously or manually. For completely autonomous UVs, the vehicles operate on their own. Semi-autonomous UVs have some intervention by humans, while manually steered unmanned vehicles are typically remote controlled by a human operator. Fully autonomous UVs are generally equipped with a Guidance, Navigation and Control-system (GNC-system). A guidance algorithm calculates the desired heading based on the current position captured by a navigation system, often a GPS receiver. A control system deals with the allocation of forces necessary to maintain heading and attitude. Unmanned vehicles have several advantages over manned vehicles for many reasons, especially when there are health, safety and environment (HSE) issues involved. In addition to potential hazards in inhospitable environment, there are also economical and efficiency aspect when arguing for the use of unmanned vehicles, thus the areas of application are many:

- Study ocean floor for mapping of the sea-bed.
- Reconnaissance and surveillance of areas.

- Reducing cost of employing drivers for public transportation.
- Logistic and transportation.
- Reduce lane width and safety margins for ground vehicles.

Although the range of applications are wide for unmanned vehicles, this thesis will principally deal with applications employing marine waypoint navigation, hence focusing on USVs.

1.2 The Evolution of USVs

Research and development on Unmanned Surface Vehicles advanced relatively unnoticed in the 1980s and early 1990s, while unmanned underwater vehicles were in the spotlight through those decades. Notwithstanding, USVs have been around since World War II when several radio controlled vehicles functioned [2]. The motivation for the enhancement in the research area on Unmanned Surface Vehicles is not only due to technological progress, but it is also highly justified by the US Navy's focus on anti-terrorism missions [3]. Because of the US Navy's interests in unmanned vehicles for reconnaissance and surveillance, USA has been one of the main contributors in the research field of USVs and this has resulted in, among other things, an USV test-bed designed by a group at the US Space and Naval Warfare Systems Center in San Diego [4].

Despite increased focus on USVs from the US Navy after successful missions in the second Gulf War, other countries expanded the research on Unmanned Surface Vehicles as well. *International Submarine Engineering* is a Canadian company that offers a kit transforming existing manned boats into USVs, and have successfully developed the semi-submersible vehicle *Dolphin MK* [5]. European researchers have also been highly involved in the research of USVs; e.g. the autonomous catamaran *CHARLIE* developed for collection of sea surface microlayer, made by CNR-ISSIA in Genova, Italy [6]. The University of Rostock has developed two USVs; *Measuring Dolphin* [7] and *Delfim* [8]. The latter is an ensemble with the UUV *INFANTE*. *Maritime Robotics*, a company located in Trondheim, Norway, has developed a high-speed USV called the *Mariner*. Its areas of application are intelligence, surveillance and reconnaissance, due to its video/infrared sensor equipment and maritime data gathering. For a thorough review of prototype USVs the

Figure 1.1: The USV *Mariner*

reader is accommodated to read the papers of Massimo Caccia: [3], [9].

Today, research and development on USVs have come a long way. As opposed to radio controlled unmanned vehicles from the 1940s, the technology now support fully autonomous USVs, and is making use of them all over the world. As an example; the US Navy operates numerous of USVs as target drones nowadays. To control the movement of the unmanned vehicles, guidance, navigation and control (GNC) systems applies.

1.3 Guidance, Navigation and Control

Guidance, Navigation and Control deals with designing systems for all kinds of vehicles to improve control. The use of GNC systems occur in autopilots, missiles, dynamic positioning (DP) among many other applications. In general, motion control systems are constructed as three interacting systems [1]. These three systems are the guidance, navigation and control systems. The guidance system provides a reference model with a calculated desired heading, the navigation system acquire position and attitude of the vehicle, while

the control system allocates thrust to the actuators to ensure that desired position and velocity is satisfied.

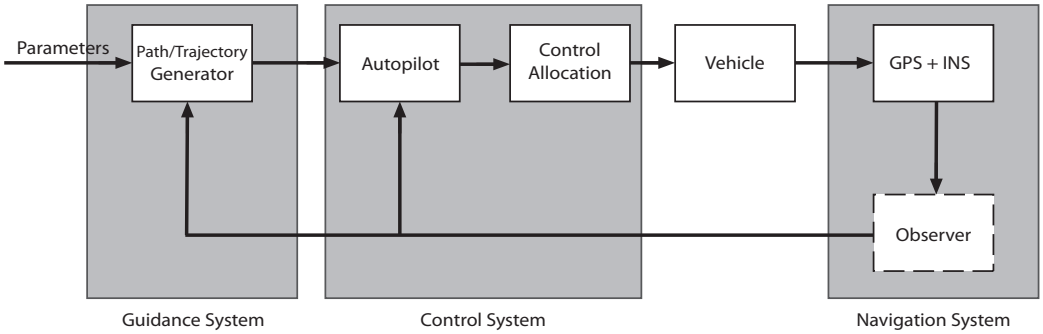

Figure 1.2: GNC block diagram

The *guidance system* keeps track of the desired heading angle that the object shall follow. The desired reference is continuously computed based on current position given by the navigation system and a target defined by the guidance algorithm. The desired heading is fed to the control system, so that the operator (human or autopilot) can follow the calculated result. Adding parameters as input to the guidance system, a path[1] or trajectory[2] can be generated by a computer. These parameters can describe craft dynamics, weather conditions, such as slippery road conditions if the GNC system is implemented in a land vehicle, or land curvature and obstacles that can have influence on a desired trajectory. The computation of the desired heading is in many cases an advanced optimization problem. However, depending on the application, simpler methods as different line-of-sight (LOS) methods may be applied [1].

The *navigation system* supplies the guidance and control systems with position and attitude of the vehicle. GPS data are the most common way of determining position, although internal navigation systems (INS) often aid the navigation in many applications.

The *control system* allocates force and torque to enforce the craft to satisfy the given control objective, for instance trajectory tracking/path-following and speed control. Flags from the guidance system and output from the

[1]Path: Pure geometric consideration. No temporal constraints, i.e. time-invariant.
[2]Trajectory: A path as a function of time; time-variant. Consider the dynamics.

navigation system are used for feedforward and feedback control respectively [1].

With the help of GNC, vehicles are able to function autonomously. For the sake of path-following schemes, the guidance systems have to put special emphasis on path planning to prepare a feasible trajectory. It follows that the vehicle dynamics and environments must be taken into consideration.

1.4 Path Planning

Planning is an important part of the guidance scheme. In this paper the emphasis is on path planning. There may be difficulties in the form of vehicle dynamics and attitude or obstacles that prevent the vehicle from moving between waypoints. Hence, a path planning algorithm has to evaluate such obstacles as parameters to create a feasible path.

The goal is usually a feasible path from A to B. An optimal solution with respect to fuel consumption, time, traveling distance or vehicle dynamics is frequently preferred. In such cases, one have to generate paths on the basis of saturating elements in the dynamics and possibly minimize a cost function subject to constraints.

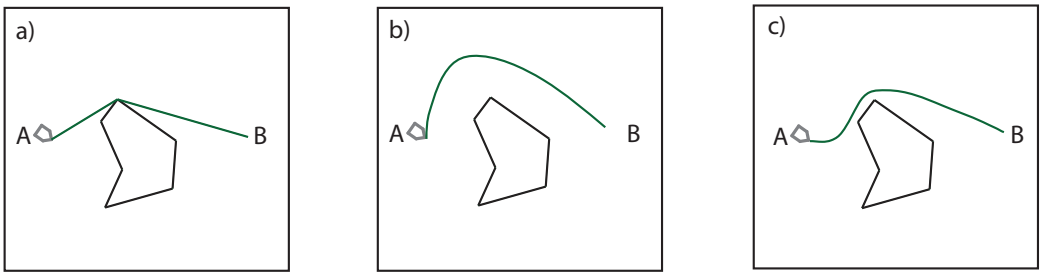

Figure 1.3: a) Optimal path with respect to shortest distance b) Feasible path ignoring vehicle dynamics c) Feasible path

Path generation is application dependent. Occasionally the goal is to have the path intersecting all waypoints, while in other applications passing the waypoint (WP) within a given error distance is sufficient. Because of this, different path generation algorithms appear and will become apparent throughout the paper.

1.5 Path-Following

When vehicles are motivated to follow a predefined waypoint setup, there are many ways to do so. A 3 DOF nonlinear controller for path-following is suggested by Fossen et al. [10]. Skjetne et al. [11] have put forward the path-following problem as a maneuvering problem separated into two tasks; geometric- and speed assignment. Simulations proved good results for the geometric task, provided a recursive design procedure. An update law was constructed to satisfy the speed assignment, and indicated satisfactory control of the vehicle velocity. However, the desired speed was set directly by the operator and not as a function of the path. Another way of accurate path-following is suggested by Nelson et al. [12]: A vector field control law is developed for various waypoint setups for air vehicles.

A review of guidance laws has been put forward by W. Naeem et al. [13] stating that LOS guidance is the key element in all guidance systems. At the MIT Autonomous Vehicles Laboratory, a waypoint-following fuzzy guidance controller [14] for a small autonomous boat proved to perform well, even on complex paths and with the presence of large disturbances. Desired heading was specified in each waypoint, in addition to the geographical coordinates.

J. Osbourne and R. Rysdyk [15] have considered waypoint guidance for UAVs in wind. Based on a lookup table, a proximity distance at each waypoint is retrieved. Thus the UAV yields smooth convergence to each new course without over- over or undershooting. Some of the same principles may be used in waypoint guidance for unmanned surface vehicles.

1.6 Contributions

In this thesis, four different ways of creating splines have been employed to recreate possible paths for path-following unmanned vehicles. The curvature along the entire path is calculated and compared with the maneuvering properties of the vehicle. Minimum vehicle turning radius is mapped directly to maximum trajectory curvature, thus assigning a curvature threshold that the path can not exceed.

A feasibility check of the path versus the vehicle trajectory is performed to determine a possibly path change. Reducing the vehicle velocity increases the turning properties, thus speed reduction as a function of path curvature/waypoint layout is proposed to maintain a feasible trajectory. If the

path is yet defined infeasible, three algorithms for moving, adding or removing waypoints have been suggested to modify the original path to comply with the vehicle dynamics.

Functions for calculating acceptance circle radius and velocities based on the angle between the previous, current and next waypoint are also proposed.

In addition, a simple graphical interface has been developed in MATLAB to simplify positioning of waypoints for simulations.

1.7 Thesis Outline

The thesis is structured as follows:

- Chapter 2 explains interpolation methods and how to create different type of smooth paths.
- Chapter 3 presents waypoint-following guidance algorithms. In addition, functions for calculating speed and acceptance circle radius with respect to waypoint layout and/or curvature are derived. A feasibility check and appurtenant algorithms coping with infeasible paths are suggested at the end of the chapter.
- Chapter 4 discusses the Nomoto- and *Viknes 830* models. A heading reference model is discussed as well.
- Chapter 5 summarizes PID-control briefly.
- Chapter 6 demonstrates the implementation and the simulator structure.
- Chapter 7 presents the results of the algorithms analyzing a waypoint setup. Simulation results for a case study are also presented.
- Chapter 8 concludes the thesis and suggests possible future work.

Chapter 2

Interpolation

Polynomial curve fitting is the technique of constructing and fitting a curve, a polynomial, to data points. Useful applications of curve fitting are for instance to visualize data trends or, as will become evident in this chapter; to create smooth paths between control points.

2.1 Polynomial Interpolation

Interpolation is one kind of curve fitting technique, and it is in principle a method of estimating the value of a function in between a discrete set of data points. As opposed to other curve fitting methods, polynomial interpolation requires an exact fit between the polynomial and the data points. This can be guaranteed if the polynomial $p(x)$ has n distinct x-values and the degree of the polynomial is at least $n-1$ [16]. However, it may be desirable to have a polynomial order higher than $n-1$ to adapt the derivative in the control points. For paths developed by interpolation methods, this implies specifying a heading in the waypoint which may be essential to guarantee path feasibility with respect to the vehicle dynamics. Extrapolation is, in contrast to interpolation, used to estimate the values outside the set of discrete data points.

Linear interpolation is the simplest kind of interpolation. Using higher order polynomials, consequently more data points to represent a function, does not always ensure better fit between interpolated polynomial and original function as Runge's Phenomenon proves [17]. Even though interpolation that ensures exact match exists, it is sometimes preferable to find an approximate fit to ignore noisy data and outliers, or to ensure that the curvature is not too large.

2.2 Path Generation Based on Waypoints

Routes of marine crafts are often represented by waypoints. This is usually a vector or a database of n waypoints that the craft consecutively follows. The waypoints are given as coordinates in $\mathbb{R}^2$ or $\mathbb{R}^3$, expressed as (x_k, y_k) and (x_k, y_k, z_k) respectively - where $k \in \{1, \ldots, n\}$. However, each waypoint can also hold other properties, such as speed and acceptance radius [1]. Hence, a waypoint vector may be expressed as

$$\mathbf{WP} = \begin{bmatrix} WP_1 \\ WP_2 \\ \vdots \\ WP_n \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & z_1 & \cdots & U_1 & R_1 \\ x_2 & y_2 & z_2 & \cdots & U_2 & R_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_n & y_n & z_n & \cdots & U_n & R_n \end{bmatrix} \tag{2.1}$$

where x_k, y_k, z_k are the waypoint coordinates, U_k is the speed and R_k the acceptance radius at waypoint k. It may also include other properties such as attitude (heading), time, course and task related information.

2.2.1 Path Generation using Straight-Lines and Circles

Straight lines and circle arcs is one way to generate paths and connect waypoints [1]. Alternatively, different interpolation strategies can be used to make a path. Using interpolation often results in a smoother path, but on the other hand the computation of the path is often more complex. For the interpolation approach, there will not occur a jump in yaw rate transitioning between the smooth line and the circle arcs, if the path has continuous curvature. Despite the convenience and simplicity of using straight lines and circle arcs to generate paths, the additional flexibility of an interpolating polynomial arises some interesting properties that make it favorable to use in path generation.

2.3 Spline

A spline curve is a function of piecewise polynomials. Usually a spline curve is composed of lower order polynomials that are joined together in knots or control points. Requesting equal value and slope of incoming and outgoing curve in the control point enforces C^0 and C^1 continuity (see Appendix

A), respectively, which is requisite to obtain a smooth path intersecting every control point in succession. This technique makes it possible to create interpolating spline functions of low degree, without encountering Runge's Phenomenon [17]. Opposite to curve fitting with least square solutions where the solution is approximating curves, splines ensure the curve to pass through every control point.

Third order polynomials, on the form $y(x) = a + bx + cx^2 + dx^3$, are a popular choice for creating splines. If you are to find a solution for a path that passes through four control points with ascending x-values , this can be done by solving the linear system $M\boldsymbol{a} = \boldsymbol{y}$, where M is the Vandermonde matrix with inserted x-values and $\boldsymbol{y}$ is the corresponding y-values. Hence, M is given by an $(n+1) \times (n+1)$ matrix

$$M = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^n \\ 1 & x_2 & x_2^2 & \dots & x_2^n \\ 1 & x_3 & x_3^2 & \dots & x_3^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n+1} & x_{n+1}^2 & \dots & x_{n+1}^n \end{bmatrix} \tag{2.2}$$

To have a spline curve running smoothly through all the control points, the properties of C^0, C^1 and possibly C^2 continuity have to be fulfilled. One way to solve this is described in [18] and is also repeated here:

1. To begin with, let a curve segment between the control points p_i and p_{i+1} be denoted by the function $f_i(x)$. To enforce C^0 continuity every curve segment has to pass through its control points, thus $f_i(x_i) = y_i$ and $f_i(x_{i+1}) = y_{i+1}$.

$$a_i + b_i x_i + c_i x_i^2 + c_i x_i^3 = y_i \tag{2.3}$$
$$a_i + b_i x_{i+1} + c_i x_{i+1}^2 + c_i x_{i+1}^3 = y_{i+1} \tag{2.4}$$

2. C^0 continuity does not imply C^1 continuity, therefore equations for the slope at the control points have to be obtained explicitly. The derivative at these points have to be the same for both the incoming and the outgoing curve segment, hence $f_i'(x_{i+1}) = f_{i+1}'(x_{i+1})$. The derivatives yield the following results

$$b_i + 2c_i x_{i+1} + 3d_i x_{i+1}^2 - b_{i+1} - 2c_{i+1} x_{i+1} - 3d_{i+1} x_{i+1}^2 = 0 \tag{2.5}$$

3. The same principle is used to obtain that the curvature is continuous, ergo C^2 continuous. $f_i''(x_{i+1}) = f_{i+1}''(x_{i+1})$ This results in the fourth equation for the linear system

$$2c_i + 6d_i x_{i+1} - 2c_{i+1} - 6d_{i+1}x_{i+1} = 0 \tag{2.6}$$

4. To complete the linear system, slopes at the start and end point must be provided. These slopes, s_0 and s_n, may be given by the current altitude so that $f_0'(x_0) = s_0$ and $f_{n-1}'(x_n) = s_n$:

$$b_0 + c_0 x_0 + 2d_0 + x_0^2 = s_0 \tag{2.7}$$
$$b_{n-1} + c_{n-1}x_n + 2d_{n-1} + x_n^2 = s_n \tag{2.8}$$

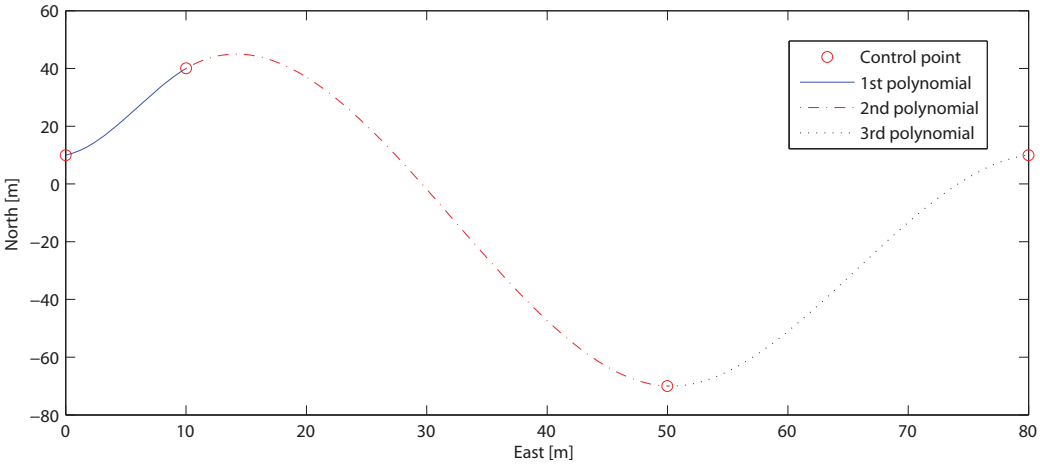

Figure 2.1: Example of Cubic Spline based on four control points

Following this approach for successive control points in ascending order solves the linear system $\mathbf{a} = M^{-1}\mathbf{y}$, so that the coefficients for the cubic curve segments can be found. However, the suggested solution above is only valid for ascending or descending values of x.

To cope with the fact that the solution above only is valid for monotonic increasing or decreasing x-values, each curve segment in 2.1 can be parameterized with a parameter $t \in [0, 1]$, as described in detail in [18]. To create spline curves for arbitrary control points, consequently for curves in two or three dimensions, one *have* to parameterize the equations. It follows that the x- and y-coordinates have to be functions of a new parameter t, thus

$$x = f(t) \tag{2.9}$$
$$y = g(t) \tag{2.10}$$

This results in curves created from two cubic curves that are function of t. Employing the same principle thought of having equal values and slopes in the control points for the arbitrary, parameterized space curves makes it possible to create smooth splines in higher dimensions as well. Expanding the linear system for the two-dimensional case results in solving two linear systems instead of one. The procedure and setup are the same, apart from solving for both parameters $\mathbf{a_x}$ and $\mathbf{a_y}$. The linear equations become:

$$M_x \boldsymbol{a_x} = \boldsymbol{x} \Leftrightarrow \boldsymbol{a_x} = M_x^{-1}\boldsymbol{x} \tag{2.11}$$
$$M_y \boldsymbol{a_y} = \boldsymbol{y} \Leftrightarrow \boldsymbol{a_y} = M_y^{-1}\boldsymbol{y} \tag{2.12}$$

2.4 Cubic Hermite Interpolation

There are several different kind of splines. A cubic hermite spline is one type where the curve segments consists of cubic parameterized polynomials in the interval $t \in [0, 1]$. Hermite interpolation differs from cubic splines in how the end points are handled [1]. While many other cubic splines require continuous curvature (C^2 continuity) at the control points, Hermite splines settles with C^1 continuity. The matrices are obtained as explained by D. House [18] for two control points p_i and p_{i+1}:

$$x(t) = a_x + b_x t + c_x t^2 + d_x t^3 \tag{2.13}$$
$$y(t) = a_y + b_y t + c_y t^2 + d_y t^3 \tag{2.14}$$

To solve the linear system and force C^0- and C^1-continuity through the control points, the following equations are employed:

$$x(0) = a_x = p_{i,x} \tag{2.15}$$
$$y(0) = a_y = p_{i,y} \tag{2.16}$$
$$x(1) = a_x + b_x + c_x + d_x = p_{i+1,x} \tag{2.17}$$
$$y(1) = a_y + b_y + c_y + d_y = p_{i+1,y} \tag{2.18}$$
$$x'(0) = b_x = p'_{i,x} \tag{2.19}$$
$$y'(0) = b_y = p'_{i,y} \tag{2.20}$$
$$x'(1) = b_x + 2c_x + 3d_x = p'_{i+1,x} \tag{2.21}$$
$$y'(1) = b_y + 2c_y + 3d_y = p'_{i+1,y} \tag{2.22}$$

As matrices, this can be expressed as

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} a_x & a_y \\ b_x & b_y \\ c_x & c_y \\ d_x & d_y \end{bmatrix} = \begin{bmatrix} p_{i,x} & p_{i,y} \\ p_{i+1,x} & p_{i+1,y} \\ p'_{i,x} & p'_{i,y} \\ p'_{i+1,x} & p'_{i+1,y} \end{bmatrix} \tag{2.23}$$

The leftmost matrix is constant and will not change depending on user input. The matrix on the right-hand side varies with the configuration and the different control points. The preferable slopes, $\boldsymbol{m}_i = [p'_{i,x},\ p'_{i,y}]^\top$ can be defined in various ways and will be discussed in the next section. This denotes that the only way the user can define the shape of the curves between the control points is by defining the slopes.

There are a few different ways of determining the slopes at the control points. If the waypoint vectors have information about the desirable attitude or heading at each WP, this can be used as a basis for the slope. Another quite intuitive way of determining a reasonable slope in a control point to obtain a smooth spline, is to use the position of the previous- and subsequent point, $\boldsymbol{p}_{i-1}$ and $\boldsymbol{p}_{i+1}$ respectively [19]. Using the local definition of the tangent $\boldsymbol{m}_i$ in the control point $\boldsymbol{p}_i$ yields:

$$\boldsymbol{m}_i = \frac{\boldsymbol{u}_i - \boldsymbol{u}_{i-1}}{2} \tag{2.24}$$

where $\boldsymbol{u}_j$ is defined as

$$\boldsymbol{u}_j = \frac{\boldsymbol{p}_{j+1} - \boldsymbol{p}_j}{\|\boldsymbol{p}_{j+1} - \boldsymbol{p}_j\|} \tag{2.25}$$

Consequently, $\boldsymbol{m}_i$ is the tangent vector defined by the unit vectors $\boldsymbol{u}_{j-1}$ and $\boldsymbol{u}_j$, parallel to the line segments in and out of the control point, respectively. The unit vector of the first and last control point ($\boldsymbol{u}_0$ and ($\boldsymbol{u}_n$) respectively) have to be taken care of individually. The magnitude of the tangent vectors $\boldsymbol{m}_i$ should be chosen so that it increases according to the distance between the control points [19]. There are several ways of choosing tangent vectors, and a few of them are listed below:

Method 1

$$\boldsymbol{m}_i = \frac{\boldsymbol{p}_{i+1} - \boldsymbol{p}_{i-1}}{2} \tag{2.26}$$

Method 2

$$\boldsymbol{m}_i = \frac{\|\boldsymbol{p}_{i+1} - \boldsymbol{p}_i\|(\boldsymbol{p}_{i+1} - \boldsymbol{p}_{i-1})}{\|\boldsymbol{p}_{i+1} - \boldsymbol{p}_{i-1}\|} \tag{2.27}$$

Method 3

$$\boldsymbol{m}_i = \frac{\|\boldsymbol{p}_{i+1} - \boldsymbol{p}_i\|(\|\boldsymbol{p}_{i+1} - \boldsymbol{p}_i\|\boldsymbol{u}_{i-1} + \|\boldsymbol{p}_i - \boldsymbol{p}_{i-1}\|\boldsymbol{u}_i)}{\|\boldsymbol{p}_{i+1} - \boldsymbol{p}_{i-1}\|} \tag{2.28}$$

Method 4

$$\boldsymbol{m}_i = \|\boldsymbol{p}_{i+1} - \boldsymbol{p}_i\|\boldsymbol{u}_{i-1} + \|\boldsymbol{p}_i - \boldsymbol{p}_{i-1}\|\boldsymbol{u}_i \tag{2.29}$$

In all cases above, the special case for the start and end point has to be treated individually. Equation (2.26) is known as the Catmull-Rom tangent and is a popular choice of tangent estimator in many applications [20]. However, other tangent estimators may yield different characteristic that is more suitable to the current objective.

Although the four different paths in Figure 2.2 achieve different characteristics in form of curvature and length, they all still fulfill the objective of the Hermite cubic spline. Ergo, they pass through every control point and are C^1 continuous. In this example, $\boldsymbol{m}_1$ and $\boldsymbol{m}_n$ are simply chosen as scaled unit vectors pointing from $\boldsymbol{p}_1$ and $\boldsymbol{p}_n$ to their neighboring control point.

After successfully generating a trajectory that intersects every waypoint, a GNC system that complies with the path-following properties can be developed. Guidance algorithms for heading control, such as LOS guidance are often a popular choice.

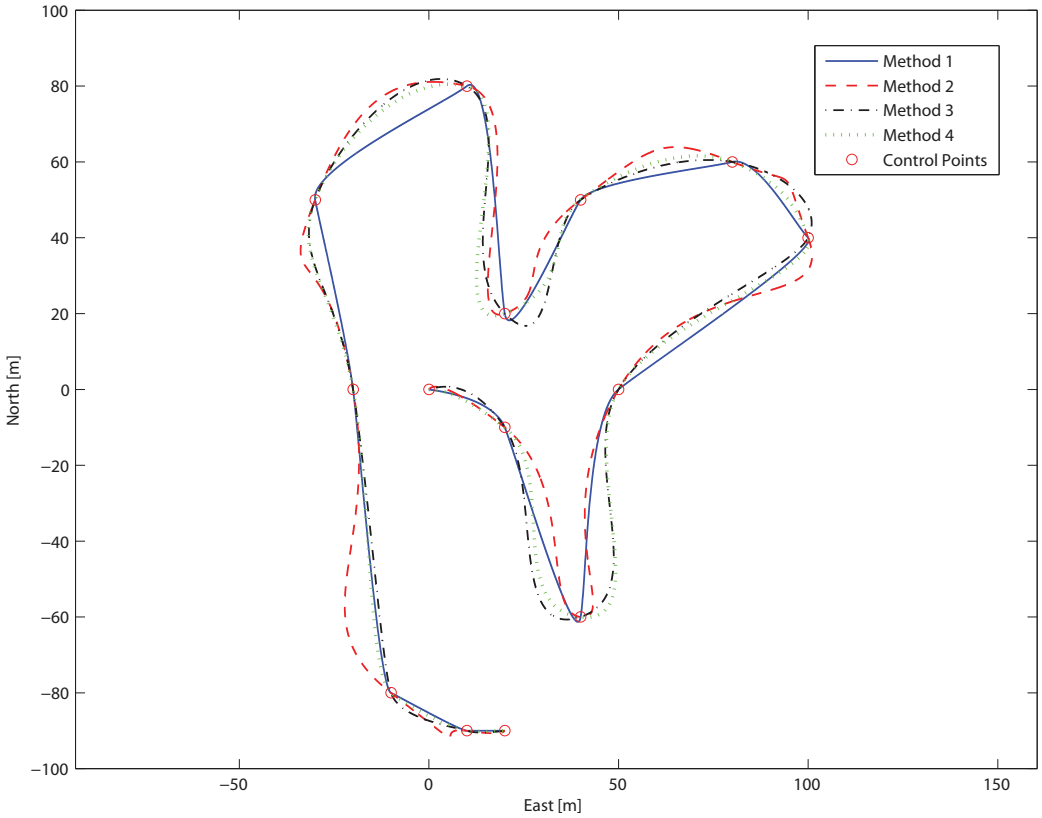

Figure 2.2: Paths generated with different tangent vectors in the control points

Chapter 3

Guidance

Guidance systems calculate the desired vehicle course based on waypoints and/or a path/trajectory. Vehicle dynamics and constraints may be counted for when deriving the course. In this chapter, a few different guidance algorithms are discussed. Functions calculating speed and circle of acceptance radius, path curvature calculation and algorithms for checking and possibly alter infeasible paths are presented as well. These paths are generated with the interpolation techniques described in the previous chapter.

3.1 Pure Pursuit Guidance Algorithm

Pure pursuit guidance only considers the target (waypoint) and the interceptor (vehicle) itself. The method is quite intuitive, whereas the heading angle is calculated only based on the current position of the craft and the next waypoint. This can be compared with the same approach a predator use to chase a prey where the approach usually results in a tale chase. The principle is explained in [1] and repeated here:

$$tan(\chi_d(t)) = \frac{y_k - y(t)}{x_k - x(t)} \tag{3.1}$$

where x_k and y_k are the positions of the next waypoint $\boldsymbol{p_k}$. The course angle χ_d can easily be calculated as

$$\chi_d(t) = atan2(y_k - y(t),\ x_k - x(t)) \tag{3.2}$$

atan2 is the four quadrant arctangent, thus $-\pi \leq atan2(y, x) \leq \pi$. To ensure that the course angle is continuous, mapping from $[-\pi,\ \pi] \rightarrow [-\infty,\ \infty]$ is

performed [21][1]. This discontinuity problem is solved by the current mapping for all three guidance algorithms.

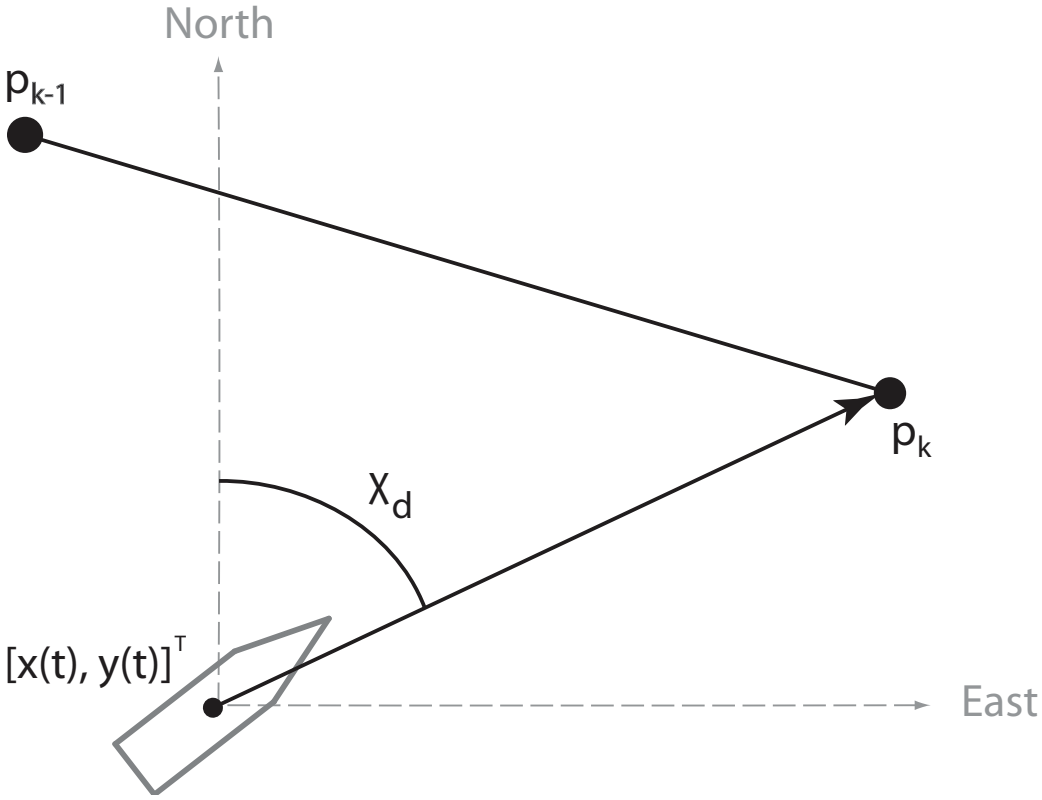

Figure 3.1: Pure Pursuit guidance principle

3.2 LOS Guidance Algorithms

The basic idea behind line-of-sight (LOS) guidance algorithms is to define a LOS setpoint on the straight line between two waypoints $\boldsymbol{p}_k$ and $\boldsymbol{p}_{k+1}$. The vector from the craft's current position to the setpoint is known as the LOS vector and its direction implicitly also becomes the course of the craft. The setpoint can be chosen in many ways. One method to set the waypoint is to introduce a radius, R, that encircles the craft. The setpoint is chosen in *one* of the *two* intersection points between the circle and the straight line

[1]The mapping described in [21] contains a minor error when mapping from second to fourth quadrant. This is however fixed in the latest version of MSS Toolbox.

between waypoint $\boldsymbol{p}_k$ and $\boldsymbol{p}_{k+1}$, depending on the waypoint orientation. Another method is to induct a lookahead distance Δ from from the vehicle's current position projected onto the line segment and a distance Δ ahead. Both methods assumes waypoint navigation and that the vehicle has passed the waypoint $\boldsymbol{p}_k$ and are heading for the waypoint $\boldsymbol{p}_{k+1}$. These approaches are explained in [1] and will be described in detail next:

Assuming the craft is at current position $\boldsymbol{p}(t) = [x(t),\ y(t)]^\top$ and the auxiliary setpoint is found at $[x_{los},\ y_{los}]^\top$. The trigonometric relations yields exactly as for pure pursuit guidance, although $\boldsymbol{p}_k$ is exchanged with $\boldsymbol{p}_{los}$:

$$tan(\chi_d(t)) = \frac{\Delta y(t)}{\Delta x(t)} = \frac{y_{los} - y(t)}{x_{los} - x(t)} \tag{3.3}$$

hence, employing the four quadrant inverse *atan2* finds the course angle χ_d

$$\chi_d(t) = atan2(y_{los} - y(t),\ x_{los} - x(t)) \tag{3.4}$$

Two different LOS guidance principles; *enclosure-based steering* and *lookahead-based steering* are described in [1] and will be reiterated in the next sections.

3.2.1 Enclosure-Based LOS Steering

Given a circle with radius $R > 0$ enclosing the craft's current position $[x(t),\ y(t)]^\top$. This circle will intersect the line between $\boldsymbol{p}_k$ and $\boldsymbol{p}_{k+1}$ at two points when R is chosen sufficiently large (see Figure 3.2) [1]. The main goal is to drive the cross-track error $e(t)$ to zero by driving the velocity towards the setpoint $[x_{los},\ y_{los}]^\top$. The coordinates of the LOS setpoint is unknown and have to be calculated based on the slope of the straight line and the circle equation. This will in fact evaluate to two solutions, however, the desired LOS point will be chosen based on the consecutive sequence of the waypoints.

$$(x_{los} - x(t))^2 + (y_{los} - y(t))^2 = R^2 \tag{3.5}$$

$$x_{los}^2 + x(t)^2 - 2x_{los}x(t) + y_{los}^2 + y(t)^2 - 2y_{los}y(t) = R^2 \tag{3.6}$$

$$\begin{aligned} tan(\alpha_k) &= \frac{y_{k+1} - y_k}{x_{k+1} - x_k} = \frac{\Delta y}{\Delta x} \\ &= \frac{y_{los} - y_k}{x_{los} - x_k} = constant \end{aligned} \tag{3.7}$$

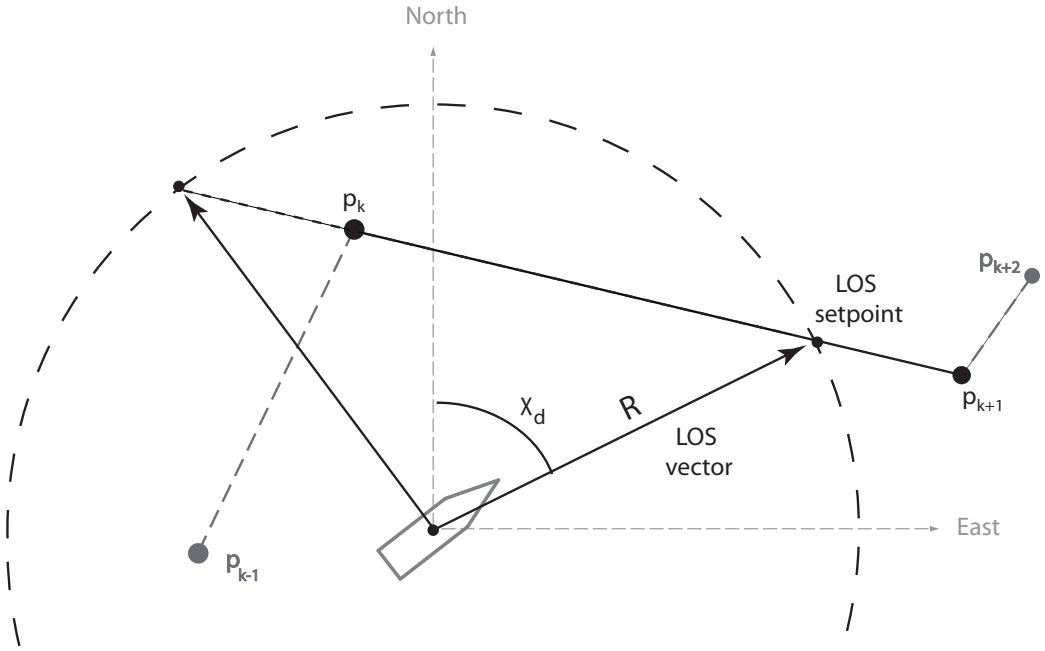

Figure 3.2: Enclosure-based guidance principle with radius R

Solving (3.7) with respect to y_{los} yields

$$y_{los} = \frac{\Delta y}{\Delta x}\Big(x_{los} - x_k\Big) + y_k \tag{3.8}$$

Inserting (3.8) into (3.6) gives

$$\begin{aligned} x_{los}^2 + x(t)^2 - 2x_{los}x(t) + \Big(\frac{\Delta y}{\Delta x}\Big(x_{los} - x_k\Big) + y_k\Big) + y(t)^2 \\ - 2\Big(\frac{\Delta y}{\Delta x}\Big(x_{los} - x_k\Big) + y_k\Big)y(t) = R^2 \end{aligned} \tag{3.9}$$

This can be rewritten into a simple second order equation

$$ax_{los}^2 + bx_{los} + c = 0 \tag{3.10}$$

where the coefficients a, b and c are expressed as

$$a = 1 + \left(\frac{\Delta y}{\Delta x}\right)^2 \tag{3.11}$$

$$b = -2x(t) - 2x_k\left(\frac{\Delta y}{\Delta x}\right)^2 + 2y_k\left(\frac{\Delta y}{\Delta x}\right) - 2y(t)\left(\frac{\Delta y}{\Delta x}\right) \tag{3.12}$$

$$c = x(t)^2 + y(t)^2 + x_k^2\left(\frac{\Delta y}{\Delta x}\right)^2 + y_k^2 - 2y_k x_k\left(\frac{\Delta y}{\Delta x}\right) + 2y(t)x_k\left(\frac{\Delta y}{\Delta x}\right) - 2y(t)y_k - R^2 \tag{3.13}$$

The solution x_{los} can easily be obtained as

$$x_{los} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{3.14}$$

The solution closest to the target waypoint should be chosen. As long as the vehicle has not passed the waypoint outside the acceptance circle, x_{los} should be picked accordingly:

$$x_{los} = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \qquad if\ \Delta x > 0 \tag{3.15}$$

$$x_{los} = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \qquad if\ \Delta x < 0 \tag{3.16}$$

Inserting this solution into (3.8), y_{los} is obtained. This solution is valid when $\Delta x \neq 0$. For the special case of $\Delta x = 0$, the solution is

$$x_{los} = x_k = x_{k+1} \tag{3.17}$$

$$y_{los} = y(t) + \sqrt{R^2 - (x_{los} - x(t))^2} \qquad if\ \Delta y > 0 \tag{3.18}$$

$$y_{los} = y(t) - \sqrt{R^2 - (x_{los} - x(t))^2} \qquad if\ \Delta y < 0 \tag{3.19}$$

When the solutions for x_{los} and y_{los} are obtained, equation (3.4) gives the desired heading, χ_d.

3.2.2 Lookahead-Based LOS Steering

For lookahead-based LOS, the desired course angle is separated into two parts [1]:

$$\chi_d(e) = \chi_p + \chi_r(e) \tag{3.20}$$

where

$$\chi_p = \alpha_k \tag{3.21}$$

$$\chi_r(e) = arctan\Big(\frac{-e(t)}{\Delta}\Big) \tag{3.22}$$

where (3.21) is the path-tangential angle and (3.22) is the course towards the LOS setpoint relative the lookahead-distance Δ and the cross-track error e (see Figure 3.3). The lookahead-distance Δ can be chosen arbitrary and affects the steering sensitivity when the saturating control law yields

$$\chi_r(e) = arctan(-K_p e(t)) \qquad K_p = \frac{1}{\Delta} \tag{3.23}$$

A shorter lookahead distance Δ results in more aggressive steering. The

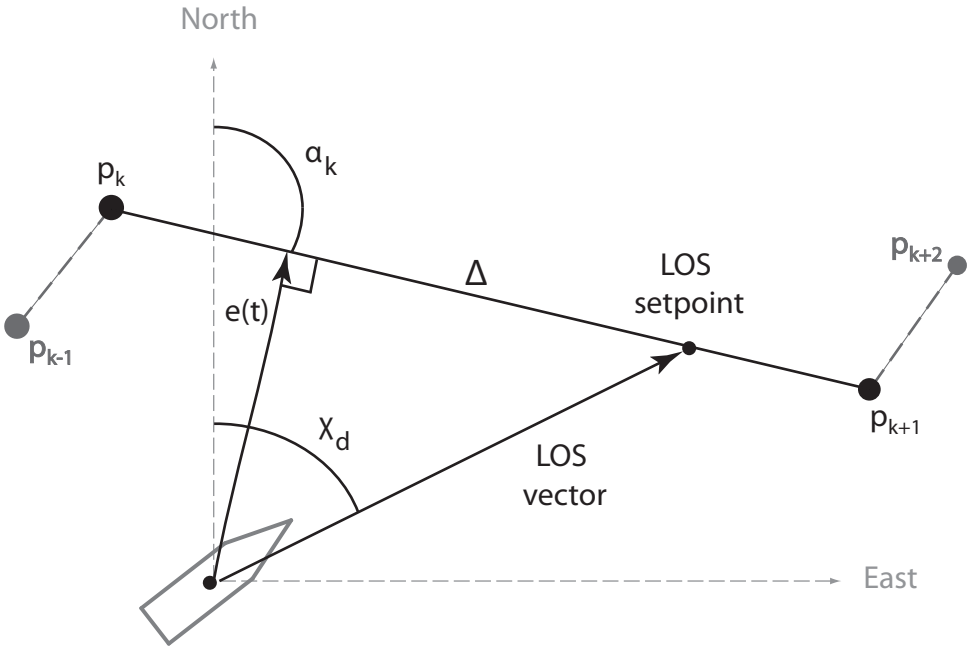

Figure 3.3: Lookahead-based guidance principle with lookahead distance Δ

parameter Δ should be selected with respect to the craft dynamics, speed

and application requirements.

Integral action could also be considered implemented to reduce potential steady-state error. Hence, the expression for χ_r becomes

$$\chi_r(e) = arctan(-K_p e(t) - K_i \int_0^\tau \epsilon(\tau) d\tau) \tag{3.24}$$

where $K_i > 0$ is the integral gain. However, integral action should be treated with carefulness to avoid wind-up and overshoots, therefore this should only be implemented when experiencing a steady-state error which may occur for crafts following a straight-line path in water with current or wind.

3.3 WP Switching Algorithm

For both pure pursuit guidance and LOS guidance algorithms, there have to be a way to switch between the different WPs. The most intuitive method is to assign a circle of acceptance to each of the waypoints. From a navigation system on board on the vehicle, the current position $\boldsymbol{p}(t) = [x(t),\ y(t)]^\top$ can at all time be obtained. Therefore, it is easy to check whether or not the craft is within the acceptance radius set at the current WP by obtaining the following formula [1]:

$$\Big(x_{k+1} - x(t))^2 + (y_{k+1} - y(t)\Big)^2 \leq R_{k+1}^2 \tag{3.25}$$

where the target WP is at $\boldsymbol{p}_{k+1} = [x_{k+1},\ y_{k+1}]^\top \in \mathbb{R}^2$ and $\boldsymbol{p}(t) = [x(t),\ y(t)]^\top \in \mathbb{R}^2$ denotes the position of the craft, at time t. The radius of acceptance can be set to a fixed number or chosen accordingly to the angle between WP $\boldsymbol{p}_{k-1}$, $\boldsymbol{p}_k$ and $\boldsymbol{p}_{k+1}$, which will be explained in detail later in this chapter. The WPs are ordered in a list, so that once you enter the acceptance circle of $\boldsymbol{p}_{k+1}$, the following WP, $\boldsymbol{p}_{k+2}$, in the list will be set as as the next target. Special consideration must be taken for the last WP, $\boldsymbol{p}_n$. Entering the acceptance circle of $\boldsymbol{p}_n$, it may be sensible to decelerate the vehicle and switch to manual control.

3.4 Speed Profile

Speed adaption is embedded in the path planning to ensure feasibility of path. To overcome sharp turns with high curvature the speed has to be accommodated, so that a feasible turning radius can be obtained. In manned ships

and cars, a mate adjusts the velocity due to visual impression and system of signs. For autonomous vehicles under a WP/path-following regime however, this speed adaption can be computed based on the waypoint placement. This means that the WPs or the entire path can be assigned a reference speed. When adjusting the speed in time before course-changing maneuvers, it may minimize cross-track error from the reference path and are therefore highly desirable.

When assigning a reference speed to each WP, a speed controller should be implemented in addition to a course controller to obtain desired speed along the path.

3.4.1 Speed Profile from Direct WP Layout

When employing regular waypoint guidance, where no preturn is introduced, either with pure pursuit guidance or line-of-sight guidance, it is preferable to alter the speed according to the path. For a straight line with consecutive WPs, it is not necessary to slow down when approaching a new WP. However, if the target WP at ($\boldsymbol{p}_k$) is straight north of the vehicle and the consecutive WP at ($\boldsymbol{p}_{k+1}$) is directly east of $\boldsymbol{p}_k$, it results in a sharp turn of 90°. This highly motivates that the speed should be reduced when approaching the turn. As a result of this, the following formula for reference speed in each WP is calculated. The desired speed is only a function of the angle between the previous WP $\boldsymbol{p}_{k-1}$, the next WP $\boldsymbol{p}_k$ and the consecutive waypoint $\boldsymbol{p}_{k+1}$. This angle is expressed as [22]:

$$\|\boldsymbol{u}_k \times \boldsymbol{u}_{k+1}\| = \|\boldsymbol{u}_k\| \, \|\boldsymbol{u}_{k+1}\| \sin(\alpha_k) \tag{3.26}$$

$$\boldsymbol{u}_k \cdot \boldsymbol{u}_{k+1} = \|\boldsymbol{u}_k\| \, \|\boldsymbol{u}_{k+1}\| \cos(\alpha_k) \tag{3.27}$$

$$\tan(\alpha_k) = \frac{\|\boldsymbol{u}_k \times \boldsymbol{u}_{k+1}\|}{\boldsymbol{u}_k \cdot \boldsymbol{u}_{k+1}} \tag{3.28}$$

$$\alpha_k = atan2\Big(\|\boldsymbol{u}_k \times \boldsymbol{u}_{k+1}\|, \; \boldsymbol{u}_k \cdot \boldsymbol{u}_{k+1}\Big) \tag{3.29}$$

where $\boldsymbol{u}_k = [x_k \; y_k]^\top$ and $\boldsymbol{u}_{k+1} = [(x_{k+1} - x_k), \; (y_{k+1} - y_k)]^\top$. `atan2` is used to ensure $\alpha \in [-\pi, \; \pi]$. Thus, the reference velocity at waypoint k can be chosen as

$$v_{ref}(\alpha_k) = v_{min} + \Big(v_{max} - v_{min}\Big) e^{-\frac{\alpha_k^2}{\sigma^2}} \tag{3.30}$$

where v_{min} is the lower bound of the velocity and should be chosen as the very slowest operating speed of the vehicle. v_{max} is the upper bound and should be the nominal velocity. α is the input angle defined in (3.29). The function guarantees symmetry about $\alpha = 0$ which is attractive since course-changing maneuvers should be independent of turning direction. The design parameter σ alters the shape/slope of the curve. The relationship between the vehicle speed and turning radius is not necessarily linear, therefore it is favorable that the Gaussian function (3.30) introduces a safety margin: This is achieved by decreasing the velocity further for sharper turning angles. The parameter σ must be chosen to take the vehicle dynamics into consideration, although $\sigma = 0.5$ can be seen as a rule of thumb. For $\sigma = 0.5$, the reference velocity is approximately the same for $|\alpha| \in [90°,\ 180°]$. Even lower σ-values should be set if course-changing maneuvering quality decreases drastically for increasing α.

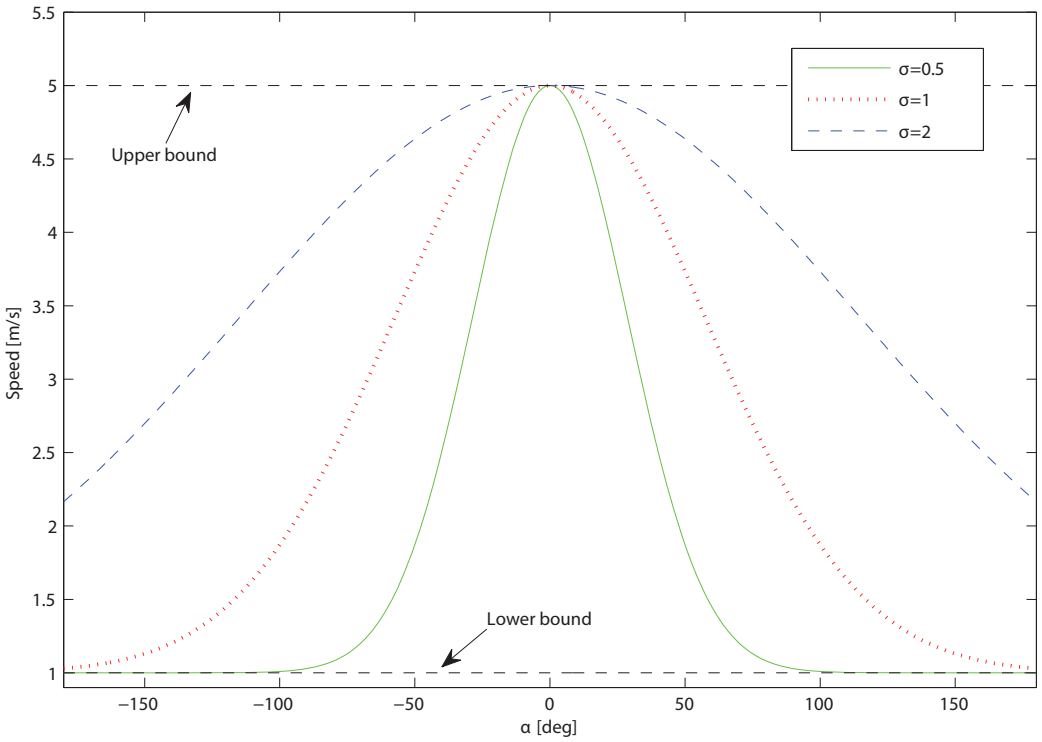

Figure 3.4: Reference speed $v_{ref}(\alpha)$ for $\alpha \in [-180°, 180°]$ for different σ-values

As seen in Figure 3.4, varying the parameter σ yields different reference velocity curves. The lower and upper velocity bounds are set to $v_{min} = 1\ m/s$

and $v_{max} = 5\ m/s$ respectively.

In Figure 3.5, σ is set to 0.5. This results in significantly different velocity outputs at waypoint $\boldsymbol{p}_2 = [50,\ 0]^\top$ for the three waypoint configurations. The desired WP speed is calculated to be $v^a_{ref} = 5.00\ m/s$, $v^b_{ref} = 1.03\ m/s$ and $v^c_{ref} = 1.00\ m/s$, for the configurations from left to right, respectively.

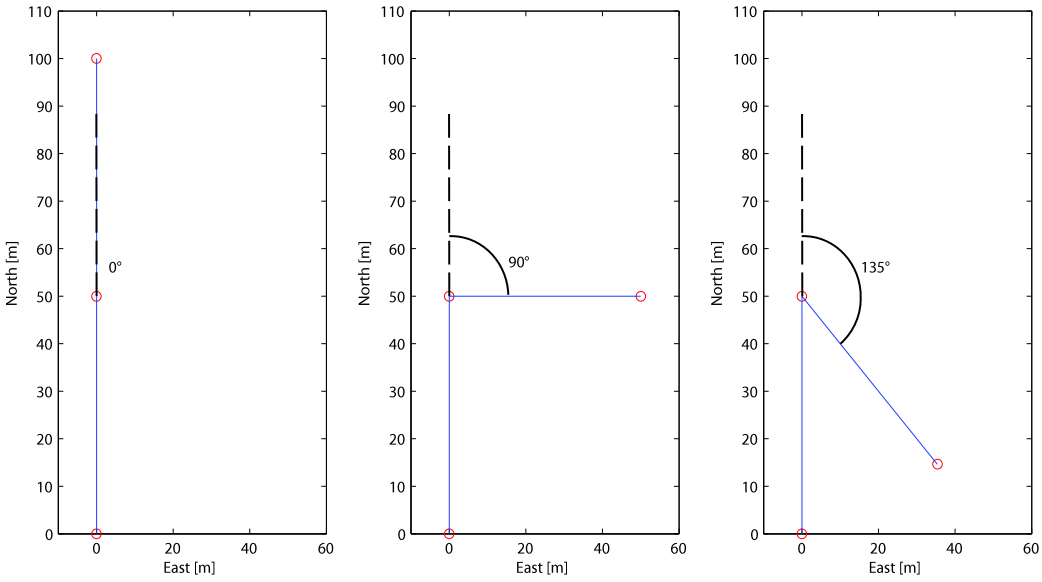

Figure 3.5: Three different α-angles: $\alpha_1 = 0°$, $\alpha_2 = 90°$ and $\alpha_3 = 135°$

The parameters v_{min}, v_{max} and σ are dependent of the vehicle dynamics and the area of application. Assigning different velocities at each WP can be an applicable solution when the WPs are equidistant distributed and have a smooth setup. However, if a straight-line path is sampled with a lot of noise and short waypoint intervals, the reference speed might become too low. A solution to this is to obtain a speed profile based on the curvature from the planned path intersecting the waypoints.

3.4.2 Acceptance Circle Radius Based on WP Layout

Every waypoint is encapsulated by a acceptance circle that can be considered as the border of the waypoint. When the vehicle is within this circle of acceptance, waypoint switching takes place. A new waypoint denotes a new heading. Thus, the acceptance circle effects the vehicle trajectory as regards to the timing of course-changing maneuvers. To avoid large overshoots

and passing far through the waypoint before turning because of slow vehicle dynamic, larger acceptance circle can in some applications be implemented based on the angle α (3.29) for the consecutive waypoints. For other applications, the radius is specified by the mission objective. An equation for calculating a desirable radius is:

$$R(\alpha) = R_{max} - \Big(R_{max} - R_{min}\Big)e^{-\frac{\alpha^2}{\sigma_R^2}} \tag{3.31}$$

where R_{max} is the desired maximum radius and R_{min} is the minimum possible radius. σ_R is a scaling factor that shapes the acceptance circle profile. The function is symmetric about $\alpha = 0$ and yields the same profitable properties as equation (3.30).

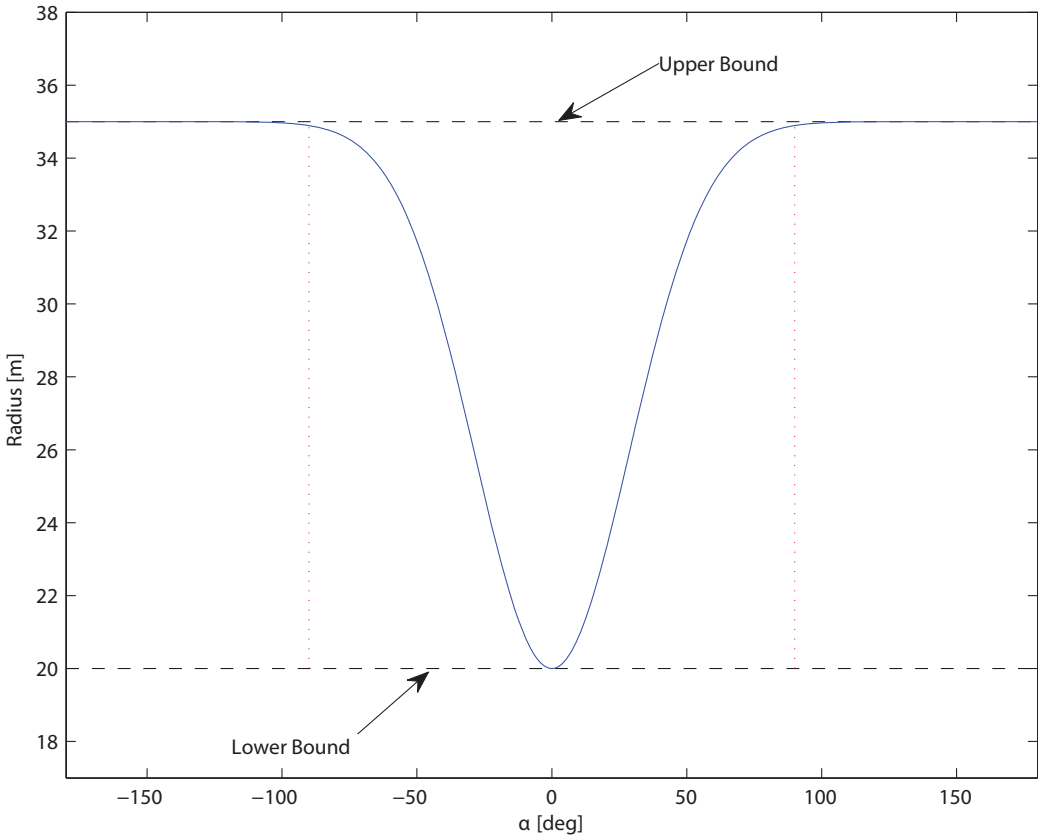

Figure 3.6: Acceptance circle radius based on α

As seen in Figure 3.6, the acceptance circle for a waypoint with $\alpha = 90°$, is $R \approx 35\ m = R_{max}$ with the given minimum radius bound $R_{min} = 20\ m$ and $\sigma_R = 0.5$.

3.4.3 Curvature

As explained in Chapter 2, there are several ways to create splines between waypoints. These splines are C^1 or C^2 continuous, depending on the interpolant method. Letting the spline represent the desired path opens for inserting more WPs on the trajectory between the old, original waypoints. Hence, the new WPs are distributed along the curve segment from $\boldsymbol{p}_k$ to $\boldsymbol{p}_{k+1}$, and this segment is parameterized and given by the expressions $x(t) = a_x + b_x t + c_x t^2 + d_x t^3$ and $y(t) = a_y + b_y t + c_y t^2 + d_y t^3$. Since there now is a 3rd order polynomial describing the curve, the curvature along this curve can be calculated.

The curvature of a curve in two-dimensional Eucledian space needs to be defined before proceeding with the speed profile generation. Intuitively, straight lines do not have curvature, while a circle has constant curvature everywhere on its path. A circle with large radius is less curved than one with a small radius, thus the conception of curvature can be thought of as rate of change of direction [22]. The curvature of circles becomes even more evident when defining the curvature κ as a function of the radius ρ:

$$\kappa = \frac{1}{\rho} \tag{3.32}$$

For two-dimensional smooth curves, there will always be an unique circle, called an osculating circle with radius ρ, that approximates the curve near a point p defined on the curve, as long as $\kappa \neq 0$; that is, the curve is not a straight line. If on the other hand the line is straight, it is presumed that the radius ρ of the osculating circle is infinite. For parameterized curves in $\mathbb{R}^2$, the derivatives of $x(t)$ and $y(t)$ will be investigated, so that a formula for the curvature κ can be obtained [22]:

$$\kappa = \frac{|x'y'' - x''y'|}{\left[(x')^2 + (y')^2\right]^{\frac{3}{2}}} \tag{3.33}$$

Since the curve is smooth, there are no discontinuities when $x'(t) = 0$ because $y'(t) \neq 0$ simultaneously. Looking closer at a cubic spline, the curvature for the entire curve can be calculated. The curvature for the example path in Figure 2.2 when method *3* for tangent calculation is employed, can be seen in Figure 3.7.

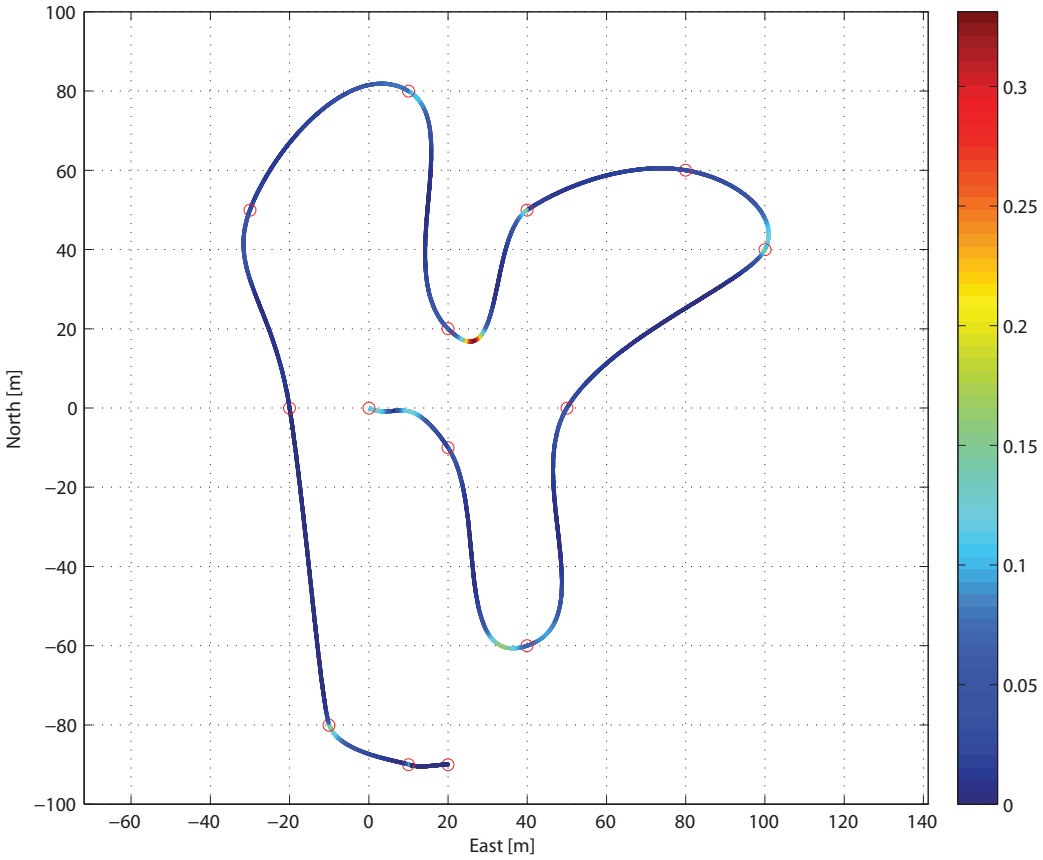

Figure 3.7: Curvature for an arbitrary path; (blue = low curvature) (red = high curvature)

The colorbar in Figure 3.7 spans from low curvature (dark blue) to high curvature (red). Since the interpolation method does not ensure C^2-continuity, the curvature at the waypoints is not continuous.

3.4.4 Speed Profile Based on Path Curvature

Despite steps in the curvature at the control points, a speed profile pursuant to the curvature can be calculated. Because of the C^2-discontinuity at the WPs, an averaging filter is implemented to smooth the curvature. The filtered curvature still contains the important information of the curvature; see Figure 3.8.

Discretization of the curvature values *may* be carried out to minimize wear on

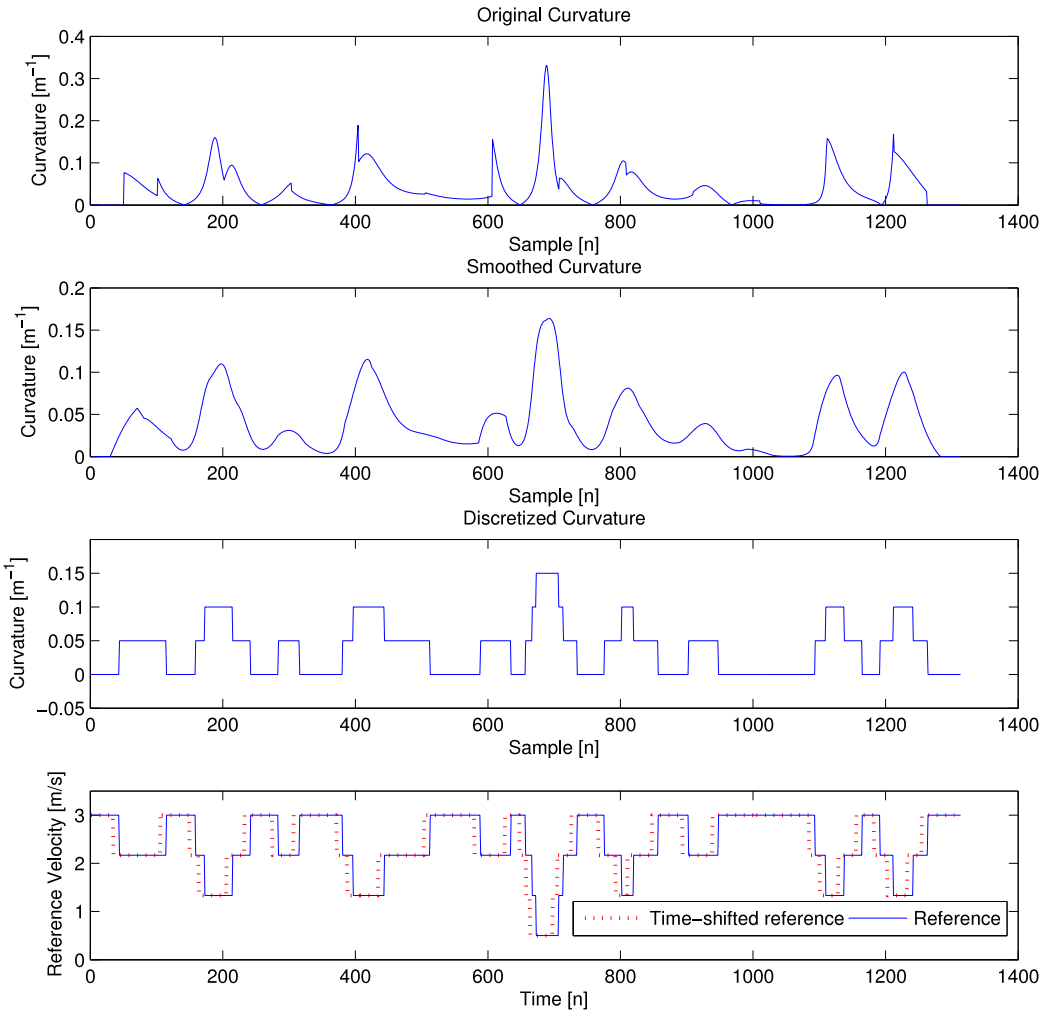

Figure 3.8: The three steps from original curvature to corresponding reference velocity

tear on the actuators, by limiting the number of velocity setpoints possible. In the end, the mapping from curvature to reference velocity is applied:

$$v_{ref} = v_{min} + \frac{max(\kappa) - \kappa_j}{max(\kappa)} \cdot (v_{max} - v_{min}) \tag{3.34}$$

where v_{min} is the minimum velocity applied at the point where maximum curvature takes place. v_{max} is maximum velocity, typical at straight-lines where the curvature is low. `max(`κ`)` is the maximum curvature value and κ_j

is the curvature at sample j.

Due to the time delay in most actuators, the reference velocity should be time-shifted forward to cope with the transition time it takes to attain the reference.

3.5 Feasible Path Selection

By introducing splines that intersects the WPs, the waypoint navigation problem is now treated as a path-following task. Since the curvature calculation for the generated spline can be implemented, an additional advantage arises. The curvature is a measure for how sharp the turns are. As a rule of thumb, the curve should not be sharper than the minimum turning radius of the vehicle, since that will cause an infeasible path. Proven that minimum turning radius increases dependent on the vehicle speed, there should be a speed reduction before approaching curves to meet the characteristics of a sharp bend. Hence, execution of a feasible turn is possible. If there is yet no feasible path after speed reduction, additional attention should be given. Possible solutions to cope with the infeasible paths are:

- Ignoring WPs that causes the infeasibility if the application allows it.
- Add additional waypoints to approach the infeasible turn with a different heading.
- Regenerate the path for the particular area by moving waypoints.

For all vehicles there is a minimum turning radius when applying maximum deflection so that the heading actuators saturates. For marine crafts this typically denotes maximum rudder deflection while it is represented by the maximum wheel positioning for ground vehicles. Based on the maximum deflection, every vehicle has a minimum turning radius that either can be calculated theoretically or found by experiments. By applying this system characteristic, a limit for maximum path curvature can be set, since the turning radius should not exceed the radius of the osculating circles along the path. The curvature of the circle path made by maximizing heading actuators is found from equation (3.32). This curvature value can be compared with the path curvature. Hence, the feasibility check yields:

$$\kappa_{vehicle} > \kappa_{path} \tag{3.35}$$

where $\kappa_{vehicle} = \frac{1}{\rho_{min}}$ and ρ_{min} is the minimum circle radius the vehicle can complete, constrained by its dynamics, and velocity in particular. $\boldsymbol{\kappa}$ is calculated from equation (3.33), and κ_{path} is the maximum value of $\boldsymbol{\kappa}$. It follows that the radius of a turn has to be less than the radius of any osculating circle along the path. Even though $\kappa_{vehicle}$ can be found theoretically from the specs of a vehicle, the effect of *slip* increases the turning radius. $\kappa_{vehicle}$ should therefore be chosen even smaller than $\frac{1}{\rho_{min}}$ in many cases. Slip is discussed in [23].

3.6 Removing Waypoints Algorithm

Removing waypoints to achieve a feasible path requires that the application allows this. For mapping of the sea-bed the WPs are set strategically to cover a specific area, thus any removing of waypoints spoil the intention of the assignment. However, in some special cases WP removal may suit the application if it is done with prudence and consideration. A set of WPs can represent a patrolling path for an USV or UAV. WPs may be eliminated if, for instance, they result in an infeasible path because of the operator's WP placement, or that the waypoints are sampled by a vehicle with divergence in the dynamics compared to the path-following unmanned vehicle.

The feasibility check (3.35) has to be carried out in order to investigate whether or not the path is feasible with respect to the vehicle dynamics. If it turns out that the path curvature κ_{path} is greater than $\kappa_{vehicle}$, an attempt to remove the WP previous to the infeasible section of the curve can be made. A new WP array of length $n-1$ holds the remaining WPs. A cubic spline is calculated based on the new remaining WPs. Again, the curvature check can be completed for this curve, thus resulting in an iterative algorithm; removing a waypoint every time $\kappa_{vehicle} > \kappa_{path}$ is not satisfied. Pseudocode for this is:

Algorithm 1 WP Removal

```
  n ← length(WP)
Require: n > 4
  κ_vehicle ← constant
  κ_path ← 2 · κ_vehicle
  while κ_path > κ_vehicle and n > 4 do
    κ ← generateSpline(WP)
    κ_path ← max(κ)
    if κ_path > κ_vehicle then
      WP ← removeWPfromAarray(critical WP)
      n ← length(WP)
    end if
  end while
```

`generateSpline` returns the curvature for the entire path in the vector $\boldsymbol{\kappa}$. The function `removeWPfromAarray` deletes the entry of *critical WP* from the WP array. The critical waypoint is the last WP passed before approaching the section of the curve where $\kappa_{path} > \kappa_{vehicle}$.

3.7 Inserting Waypoints Algorithm

If the original planned path is infeasible due to the curvature values, the critical areas where too high curvature occur should be emphasized and if possible altered in some way. As described in the previous section, one way is to remove waypoints. However, removing waypoints is not reasonable for all applications. A solution to the curvature problem is to add waypoints instead, to approach the consecutive WP with a different course. A prerequisite for adding WPs is that there is a usable area outside the path for such maneuvering. Generally, there is enough space for that kind of maneuvers on the open sea, although this may not be the case in e.g. harbors or in sheltered waters. Therefore, generated paths should be examined by the operator before employment.

There are several different possibilities for the WP placement. The number of inserted WPs yields different characteristics. In Figure 3.9, three WPs have been added to create a feasible path. Originally, the curvature was too high at the vertex. By inserting additional WPs, thus replanning the path, the curvature κ_{path} can be decreased to suit the vehicle dynamics and the minimum turning radius, $\rho_{min} = \frac{1}{\kappa_{vehicle}}$. The sequence of the new WPs is also important. In general, an inward curve should result in a crossing path when

adding WPs, while outward curves should not cross. A non crossing setup is applicable for inward curves as seen in Figure 3.9. Given a fictitious straight line between the two original WPs surrounding the critical curvature area; notice that to achieve the desirable decreased curvature effect, the inserted WPs must be placed on the same side of the line as the critical point (where the curvature is too high) itself. Pseudocode for WP insertion is presented next:

Algorithm 2 WP Insertion

$n \leftarrow$ `length`(WP)
$k \leftarrow 0$
$\kappa_{vehicle} \leftarrow constant$
$\kappa_{path} \leftarrow 2 \cdot \kappa_{vehicle}$
while $\kappa_{path} > \kappa_{vehicle}$ **and** $n > k$ **do**
  $\boldsymbol{\kappa} \leftarrow$ `generateSpline`(WP)
  $\kappa_{path} \leftarrow$ `max`$(\boldsymbol{\kappa})$
  if $\kappa_{path} > \kappa_{vehicle}$ **then**
    $NEW\_WPS \leftarrow$ `generateNewWPs`$(WP,\ critical\ WP)$
    $WP \leftarrow$ `addWPsToArray`(NEW_WPS)
  end if
  $k \leftarrow k + 1$
end while

`generateNewWPs` is the function that calculates the WP placement of the new waypoints. Different parameters can be set to define the orientation and layout of these WPs. The function `addWPsToArray` inserts the output of `generateNewWPs` into the original WP array subsequent to the critical WP.

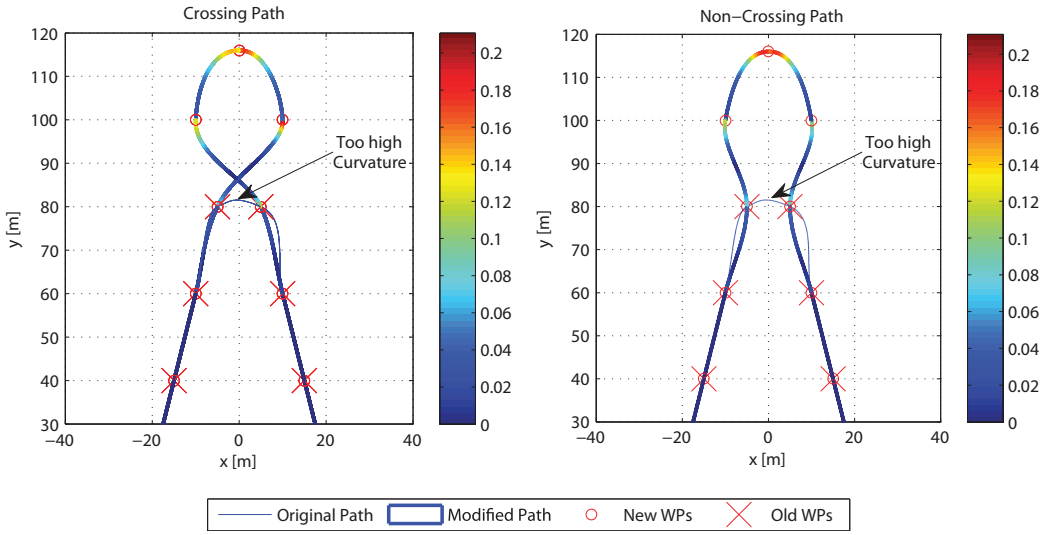

Figure 3.9: Two different ways of adding additional waypoints to decrease path curvature

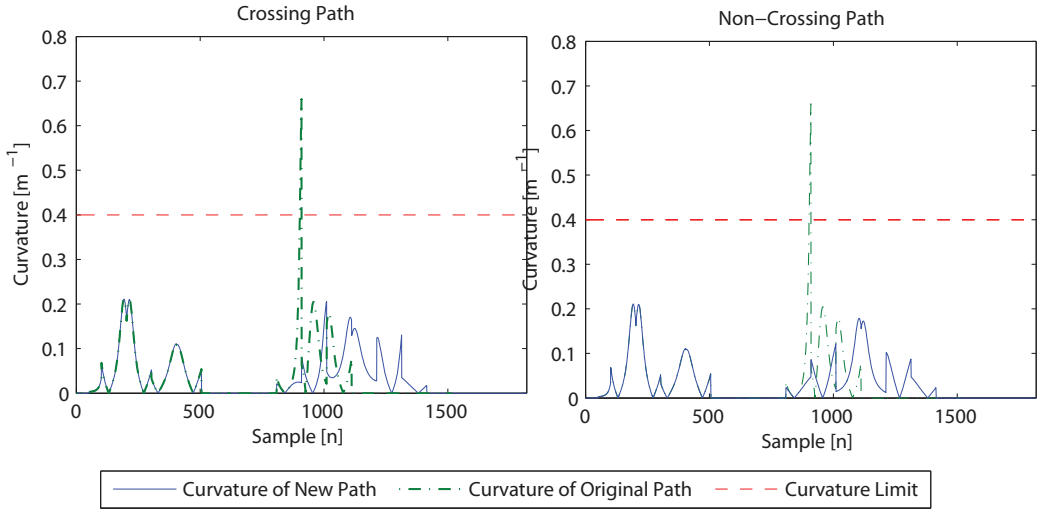

Figure 3.10: Curvature values for non-crossing and crossing WP insertion approach

3.8 Relocating Waypoints Algorithm

An algorithm that relocates waypoints does naturally lose the guarantee of intersecting every original WP. The fundamental principle of an approach that moves waypoints is to relocate an original waypoint that causes too high curvature to a better location with respect to curvature. Thus, a sharp turn can become less bendy and meet the requirements for maximum path curvature.

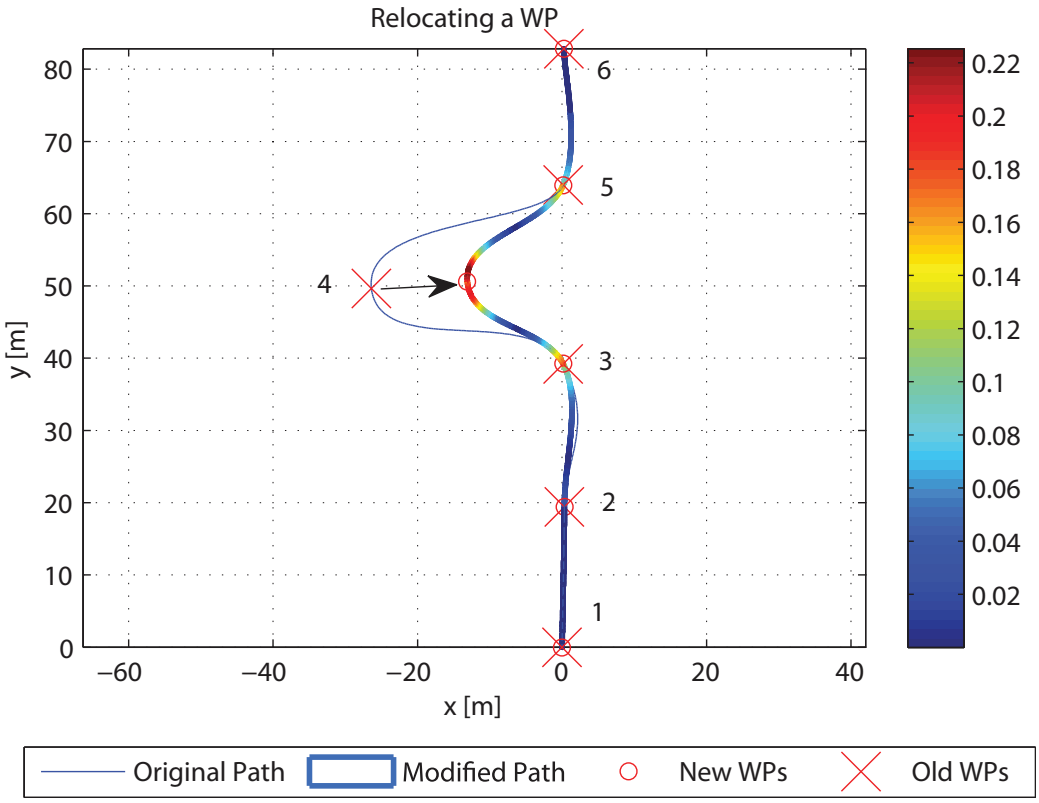

Figure 3.11: Principle of relocation of WPs that causes too high path curvature

Figure 3.11 visualizes the proposed idea of moving waypoints. Waypoint number 4 is causing too high path curvature for a vehicle with minimum turning radius $\rho_{min} = 4$, hence, $\kappa_{vehicle} = 0.25\ m^{-1}$. If it is required to intersect WP 4, adding WPs (as described in the previous section) is a solution. However, if it is sufficient to recreate a path with fairly similar design, the WP can be moved towards the midpoint of the straight line between way-

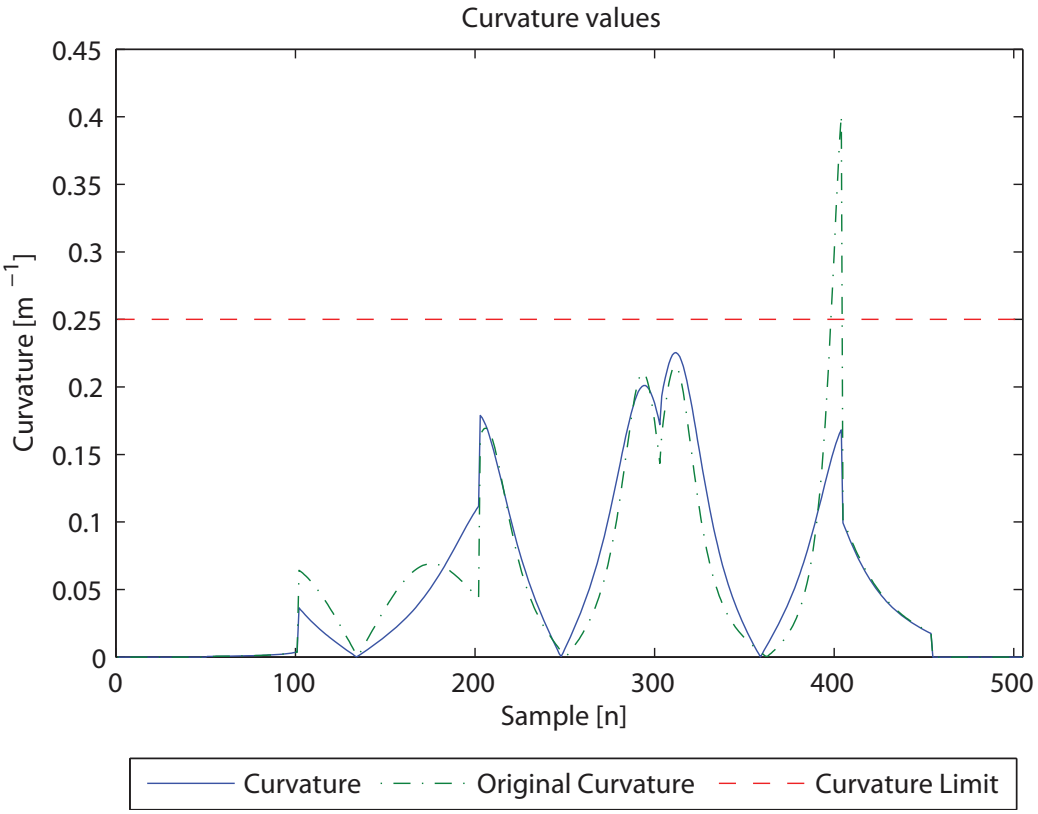

Figure 3.12: Comparison of curvature values for relocating WP algorithm

point 3 and 5. The distance of relocation of the WP towards the straight-line can be set by the operator, but in general it should not be more than half the distance from the midpoint to the relevant WP. At the same time, a smaller fraction may not yield sufficient curvature, but by re-iterating over the critical WP, it moves closer and closer to the line, thus decreasing the curvature. Important to notice is that moving waypoint 4 still affects the curvature for more than the range between waypoint 3 and 5; see Figure 3.12.

The following algorithm for relocating waypoints is proposed by the author:

Algorithm 3 Relocating WPs

$n \leftarrow$ `length`$(WP) \cdot iter\_constant$
$k \leftarrow 0$
$\kappa_{vehicle} \leftarrow constant$
$\kappa_{path} \leftarrow 2 \cdot \kappa_{vehicle}$
while $\kappa_{path} > \kappa_{vehicle}$ **and** $n > k$ **do**
  $\boldsymbol{\kappa} \leftarrow$ `generateSpline`(WP)
  $\kappa_{path} \leftarrow$ `max`$(\boldsymbol{\kappa})$
  if $\kappa_{path} > \kappa_{vehicle}$ **then**
    $RELOCATED\_WP \leftarrow$ `relocateWP`$(critical\ WP)$
    $WP \leftarrow$ `updateRelocatedWP`$(RELOCATED\_WP)$
  end if
  $k \leftarrow k + 1$
end while

`relocateWP` alters the placement of the critical WP: (WP_c). The distance moved is a fraction of the length from the relevant WP to the midpoint of the line between WP_{c-1} and WP_{c+1}. This fraction is a parameter set in the function `relocateWP` and may be decided by operator. A small fraction implies a small change in the WP placement. A rule of thumb is to set this parameter to $\frac{1}{4}$, thus moving the critical WP $\frac{1}{4}$ closer to the midpoint of the line. The number of maximum iterations made for the entire curve is resolved by *iter_constant*. Generally, the less distance a WP is moved, the more iteration should be carried out. Shorter relocating distance results in marginal change in curvature values, while greater distance yields unnecessary large safety margin of the curvature compared with $\kappa_{vehicle}$. The number of iterations is also a matter of computational power.

Chapter 4

Models

This chapter discusses different model types; both vehicle models and reference models. A short overview of Nomoto models and a more complex mathematical model of *Viknes 830* is carried out. A reference model considering limitation of the vehicle dynamics is discussed towards the end of the chapter.

4.1 Choosing the Right Vehicle Model

A mathematical vehicle model is a description of the system with mathematical equations. This makes it possible to look at the response of the vehicle excited by different input. Investigating the vehicle equations can also help understanding the system itself.

The model requirements with respect to complexity of the model are application dependent. Sometimes a model with considerable simplifications is sufficient for the controlling task. In other applications an accurate model is essential to capture the dynamic of the system. There is, however, a trade-off between the complexity and accuracy of the model that has to be taken into consideration. Nevertheless, a model of the system that describes its dynamics is essential to control it.

A complex vehicle model based on physical laws requires extensive knowledge of the phenomena involved and their mutual relations [24]. By neglecting details and focus on capturing the important dynamics, simpler mathematical models can be derived. Since simpler vehicle models may not perform as accurately as more complex models in comparison with the actual system, this leads to the design of more complex controllers to maintain performance.

A different approach is black box modeling based on input and output data. System identification makes it possible to fit model input and output data to measurements [23].

4.2 2nd Order Nomoto Model

Nomoto *et al.* [25] proposed a second order model for autopilot design from a linear maneuvering model by eliminating the sway velocity. The model became a simple transfer function from rudder angle δ to yaw rate r. Nomoto models are often preferred when a qualitative prediction capability is required from the model. For path-following properties and heading control, the Nomoto models are often sufficient when the feedback controller itself tolerates some modeling errors [26]. As stated in [1] the course transfer function for the second order Nomoto model is:

$$\frac{r}{\delta} = \frac{K(1 + T_3 s)}{(1 + T_1 s)(1 + T_2 s)} \tag{4.1}$$

Laplace transforming $\dot{\psi} = r$ and considering the transfer function from δ to ψ gives:

$$\frac{\psi}{\delta} = \frac{K(1 + T_3 s)}{s(1 + T_1 s)(1 + T_2 s)} \tag{4.2}$$

where T_1, T_2 and T_3 are the time constants and K is the gain. Given constant speed V, the vehicle velocity and position are found from the purely kinematic equations:

$$\dot{x} = V \cos\psi \tag{4.3}$$
$$\dot{y} = V \sin\psi \tag{4.4}$$

where the velocities $\dot{x}$ and $\dot{y}$ are defined in the NED coordinate system.

4.3 1st Order Nomoto Model

The second order Nomoto model can be simplified due to nearly cancellation of pole and zero in (4.1). The simple first order model becomes

$$\frac{r}{\delta} = \frac{K}{1+Ts} \tag{4.5}$$

and still noting $\dot{\psi} = r$, this can be rearranged as

$$\frac{\psi}{\delta} = \frac{K}{s(1+Ts)} \tag{4.6}$$

where $T = T_1 + T_2 - T_3$. This model is popular in autopilot design because of its combination of simplicity while still being reasonably accurate. However, it is important to notice that the model assumes constant thrust and small rudder angles. The model parameters change when altering the velocity. To comprise varying thrust a more complex model have to be employed.

4.4 Viknes 830

A mathematical model of a 27 feet leisure vessel of the type *Viknes 830* is derived in the master's thesis *Weather-Optimal Positioning Control for Underactuated USVs* of Øivind Kåre Kjerstad [27]. The vessel considered is owned by *Maritime Robotics*, and is equipped with an on-board computer and an extended sensor package so that it may operate as a versatile USV application development platform. Thus, full-scale experiments of for instance waypoint guidance can be carried out. This motivates simulating GNC systems in MATLAB/Simulink for the particular model.

The vessel net weight is approximately 3.3 metric tons and the length is nearly 8.5 meters. Due to a relatively large area of the vessel floating above the water-line, the Viknes 830 can be subject to considerable drift caused by wind forces.

4.4.1 Heading Actuators

The Viknes 830 heading actuators includes a rudder and a tunnel thruster. The rudder is installed directly behind the propeller and is controlled by a servo. Maximum rudder angle is constrained to $\pm 27°$. An electrical bow tunnel thruster can be applied for heading control at low velocities. However, this on-off thruster is superfluous for waypoint navigation when the velocity exceeds approximately 1 m/s. Thus, the mathematical model and simulator developed in [27] is adapted - the tunnel thruster is neglected.

Figure 4.1: Viknes 830

4.5 Mathematical Model

A full 6 DOF mathematical model of the *Viknes 830* boat is derived in [27]. The important findings and equations are reproduced in this section to get a better overall understanding of the model behavior in the simulation scenarios.

The position, orientation and velocity are given by the vectors η and ν in NED (North-East-Down) and BODY reference frames, respectively. The equations of motion (4.7) and (4.8) are derived from the seakeeping and maneuvering model.

$$\dot{\eta} = J(\eta)\nu + v_c \tag{4.7}$$

$$M\dot{\nu} + C_{RB}(\nu)\nu + C_A(\nu_r)\nu_r + D(\nu_r)\nu_r + \mu + G\eta = \tau_{waves} + \tau_{wind} + \tau \tag{4.8}$$

where the states are

$$\eta = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix} \qquad \nu = \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix} \tag{4.9}$$

The states of η are the vehicle position in addition to roll, pitch and yaw, respectively. For ν the states are defined as: Surge, sway, heave, roll rate, pitch rate and yaw rate.

The coefficients of equation (4.7) and (4.8) are briefly explained next (for thorough details see [27]). M is the mass matrix and includes both terms for the rigid physical structure and hydrodynamically added mass. C_{RB} and C_A are the Coriolis and centripetal matrices and contains nonlinear terms. Damping effects are gathered in the linear matrix D and the matrix $G > 0$ is a linear approximation of the restoring forces. μ deals with the fluid memory effects generated by the waves due to the motion of the vessel. The remaining terms on the right hand side of (4.8) are the driving forces of the system, where τ_{wind} and τ_{waves} are forces from the environment while τ are the control forces.

4.5.1 Actuators

The propulsion is generated by a diesel engine and the force acts only in 1 DOF - along the heading of the vessel. Additional forces in the 4 DOFs of surge, sway roll and yaw are generated by the rudder which consists of the rudder servo and the rudder itself. To generate force from the rudder, relative velocity must be present. Since the tunnel thruster is neglected for simulating scenarios in this thesis due to its inefficiency and superfluity at speed greater than 1 m/s, the vessel needs to have a minimum relative velocity to cope with maneuvering.

4.5.2 Environmental Aspects

The main contributors of environmental disturbances are wind, waves and water current. These phenomena generate forces that act on the vessel and should preferably be comprised in the mathematical model of the ship. Wind, waves and current can be modeled individually or as one combined environmental force. Notwithstanding, the forces from the environment should be

implemented to represent an overall adequate model.

Sea current is a continuous movement of water and arises from the effects of gravitational pull of the moon, temperature and salinity. Current has the effect of adding not-propulsion-generated velocity to the vessel, thus the term v_c occurs in the velocity equation (4.7). A constant current velocity leads to a heading dependent force that can be compensated for by integral action in the contingency of a PID-controller.

While sea current can be modeled as a constant force, waves on the other hand are considered to be irregular. Wave spectrum is commonly used to describe the energy distribution in the frequency domain on the surface of the ocean. This spectrum is location dependent due to disparities in the wave frequency distribution and a chosen spectrum is used to excite a response amplitude operator to generate either force or motion response [27]. The mean wave height of the one-third highest waves is called the significant wave height, and is an important parameter in the environment model for waves. The waves add randomness in time and space and are considered an important environmental disturbance. Every boatman knows the importance of the encountering angle of the waves, therefore the waves affect vessels differently depending on the boat attitude relative the waves.

Wind is usually modeled as a mean wind with random gust, although it consists of multiple components in general. The angle of attack and the shape of the vessel are significant factors when modeling the wind forces acting on the craft. The input to the wind model is the projected areas of the boat, the relative wind speed, the angle of attack and the wind coefficients that can be found experimentally. For a more thorough description see [27] and [1].

4.6 Heading Reference Model

To avoid steps for the desired heading calculated from the guidance system when switching between waypoints, a heading reference model should be implemented. A smooth reference can be obtained with a simple third-order filter [1] on the form:

$$\frac{\psi_d}{\psi_r}(s) = \frac{\omega_n^3}{(s+\omega_n)(s^2+2\zeta\omega_n s+\omega_n^2)} \tag{4.10}$$

where ω_n is the natural frequency and ζ the relative damping ratio. ψ_r is the reference heading, e.g. calculated from a guidance system, and ψ_d is the desired state. With a third-order filter on that form, it should be noted that the final value theorem yields

$$\lim_{t\to\infty} \psi_d(t) = \psi_r \tag{4.11}$$

In addition, for steps in ψ_r the first and second order derivatives of ψ_d are still smooth and bounded. Using a reference model for smoothing the trajectory of the desired heading is advantageous so that the error between the desired and actual heading is kept small. It is important that the closed-loop bandwidth never get exceeded by the cut-off frequency of the reference model to assure that the craft tracks the desired state. In other words, the eigenvalues of the desired states should be chosen such that the dynamics of the reference model is slower than the actual vehicle dynamics.

Because of dynamic restriction in the servo that actuates the rudder, or limitation in the rudder dynamics itself, it is often advantageous to saturate the yaw rate and yaw acceleration. Saturation can be obtained by implementing a saturating element in the reference model, such that $|r_d| \leq r_{max}$ and $|a_d| \leq a_{max}$ [1] [28]. In general, it is important to create a reference model that takes the physical limitations of the craft into account.

A third-order filter, as denoted in equation (4.10), can be transformed into a state space realization by applying the transformation rules from transfer functions to controllable canonical form [29]. With that, saturation on yaw rate $r_d = \dot{\psi}_d$ and yaw acceleration $a_d = \ddot{\psi}_d$ are induced so that the course-changing maneuvering can be carried out as a three phase procedure; with acceleration, steady turning and deceleration. These saturation elements have several advantages: In addition to the smooth three-phase maneuvering, the controller will more likely keep the error between desired and actual heading small since the dynamics of the craft are faster than the reference model when r_{max} is picked sufficiently small.

The state space model on controllable canonical form of the strictly proper transfer function (4.10) with saturation elements on yaw rate and yaw acceleration yields:

$\dot{\ddot{\psi}}_d = \dot{r}_d = a_d$, so the states is defined as: $\boldsymbol{\psi} = [\psi_1\ \psi_2\ \psi_3]^\top = [a_d\ r_d\ \psi_d]^\top$.

$$\begin{bmatrix} \dot{\psi}_1 \\ \dot{\psi}_2 \\ \dot{\psi}_3 \end{bmatrix} = \begin{bmatrix} -(2\zeta+1)\omega_n & -(2\zeta+1)\omega_n^2 & -\omega_n^3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u(t) \tag{4.12}$$

$$y = \begin{bmatrix} 0 & 0 & \omega_n^3 \end{bmatrix} \boldsymbol{\psi} \tag{4.13}$$

where the input $u(t) = \psi_r$. Proper saturation can now be placed on the states r_d and a_d.

In Figure 4.2, saturation elements of $r_{max} = 10\ deg/s$ and $a_{max} = 10\ deg/s^2$ are selected. $\omega_n = 1$ and $\zeta = 1$. The model experience a step of 90° in the heading reference at $t = 3\ s$, and as a result of the rate saturation, the desired heading ψ_d reaches the reference heading ψ_r later than without constraints on yaw rate and yaw acceleration. Nevertheless, such a restriction is vital to avoid too large errors between reference- and actual heading since there are physical limitations in all steering mechanisms - e.g. rudders.

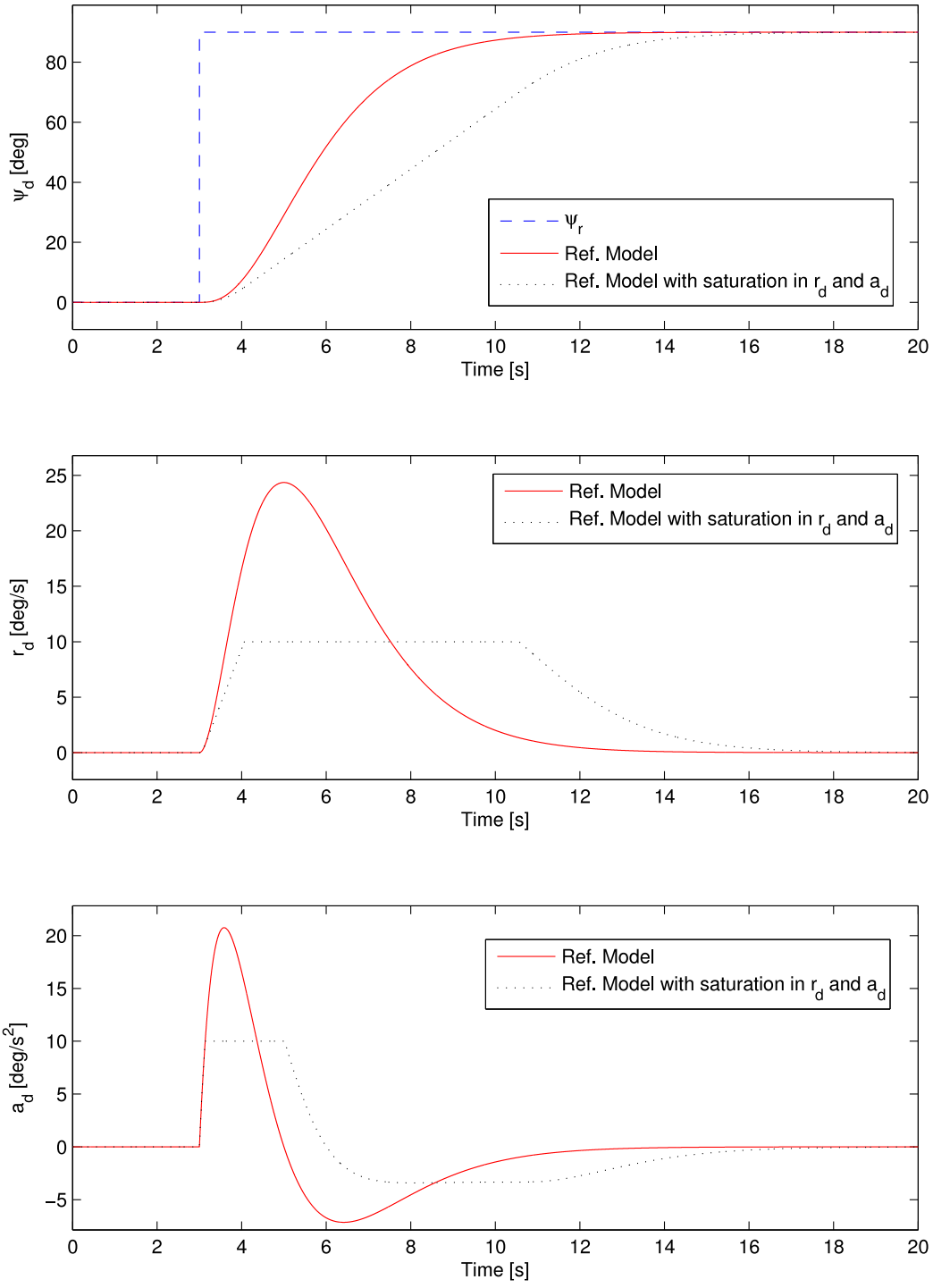

Figure 4.2: Reference models with and without saturation in r_d

Chapter 5

Control

In a GNC system the controller(s) task is to maintain or satisfy the control objectives. In the case of path-following based on waypoints, the controller is responsible for keeping up with the desired heading and speed. There are plenty of suitable controllers for path-following, but there is always a trade-off between complexity and performance. The controller of the USV *Viknes 830* by *Maritime Robotics* is a simple PID-controller, therefore a PID is the controller discussed in this chapter. The PID-controller is the most commonly used feedback controller and can by sensible tuning, described at the end of this chapter, achieve good control.

5.1 PID-Control

Conventional crafts are often equipped with main propellers for controlling surge and a rudder for controlling yaw. If no tunnel thrusters or actuators controlling sway are available, the vehicles are underactuated for 3 DOF tracking control. The output space of conventional waypoint guidance systems are usually reduced from 3 DOF to 2 DOF, only emphasizing surge and yaw [30].

To minimize the cross-track error e, a plain PID-controller is derived. Desired position is obtained by forcing the position p to converge to p_d and χ to reference course χ_d:

$$\lim_{t\to\infty}[p - p_d] = 0 \qquad \lim_{t\to\infty}[\chi - \chi_d] = 0 \tag{5.1}$$

The yaw moment output from the control system for the course control is allocated to the actuators. For smaller crafts this is usually distributed to

the rudder while for larger ships actuators as azimuth thrusters and tunnel thrusters are commonly used in addition to the rudder. However, for the dimensions of leisure size boats like *Viknes 830* operating at low/medium to high cruising velocity PID-control of only the rudder is sufficient.

The PID-controller is defined as [1] :

$$u(t) = K_p\epsilon(t) + K_i \int_0^\tau \epsilon(\tau)\mathrm{d}\tau + K_d\dot{\epsilon(t)} \tag{5.2}$$

where $\epsilon(t)$ is the error between the reference and the current state. K_* are the PID gains. The calculated controller outputs are the moments τ_{rudder} and propulsion $\tau_{propulsion}$ forcing the craft to converge to the LOS setpoint by affecting the rudder and main propellers respectively.

One way to set adequate regulator gains K_p, K_d and K_i is to compute them as function of the design parameters ω_n and ζ and the parameters of the 1st order Nomoto model: K and T. The relative damping ratio is dependent on the amount of overshoot the system tolerates. Typical values are $\zeta \in [0.8,\ 1]$. The natural frequency ω_n is expressed as:

$$\omega_n = \frac{1}{\sqrt{1 - 2\zeta^2 + \sqrt{4\zeta^4 - 4\zeta^2 + 2}}} \cdot \omega_b \tag{5.3}$$

where ω_b is the closed-loop-bandwidth of the regulated system. ω_b should be chosen between the bandwidth of the Nomoto model and the rudder bandwidth [1].

- The proportional gain K_p is set to:

$$K_p = \frac{T\omega_n^2}{K} \tag{5.4}$$

- Computing the D-gain, K_d:

$$K_d = \frac{2T\zeta\omega_n - 1}{K} \tag{5.5}$$

- A rule of thumb is to set the integral gain 10 times slower than the natural frequency ω_n.

$$K_i = \frac{\omega_n K_p}{10} \tag{5.6}$$

it follows that the gains K_p, K_d, $K_i > 0$.

Chapter 6

Implementation

To examine the theory of path generation and guidance algorithms a simulator can be used. The MATLAB/Simulink framework is a good working environment for simulations of control systems. In that context, the Marine System Simulator (MSS) Toolbox with basic libraries for GNC is employed. The boat model of *Viknes 830* [27] developed by Kåre Øivind Kjerstad is also developed in MATLAB/Simulink, thus it became an inevitable choice. The simulator made by Kjerstad is adopted and modified to fulfill the goals of waypoint path-following. In addition to the 6 DOF Viknes boat model, a Nomoto model was implemented to decrease simulation time, yet retain sufficient results compared to the more complex model. The simulator structures and implementation will become apparent throughout this chapter.

6.1 Waypoint Generator

To avoid tedious work of manually typing the waypoints, a simple waypoint generator making it possible to set the WPs graphically was developed. Assuming initial vehicle position to be in the origin $\boldsymbol{p}_0 = [x_0,\ y_0]^\top$, consecutive mouse click adds new waypoints to the WP vector. The WPs are ascendingly numbered with straight lines between the consecutive WPs for operator's legibility. This feature may become highly desirable when adding a scaled map as a background image, thus improve visualization of path and trajectory with respect to the topography. However, adding the feature of having a map as background image is left out for future work.

For the case of direct WP following without path generation from cubic splines, setting waypoints and calculating desired speed and acceptance radius at each WP are done simultaneously. These parameters are added to

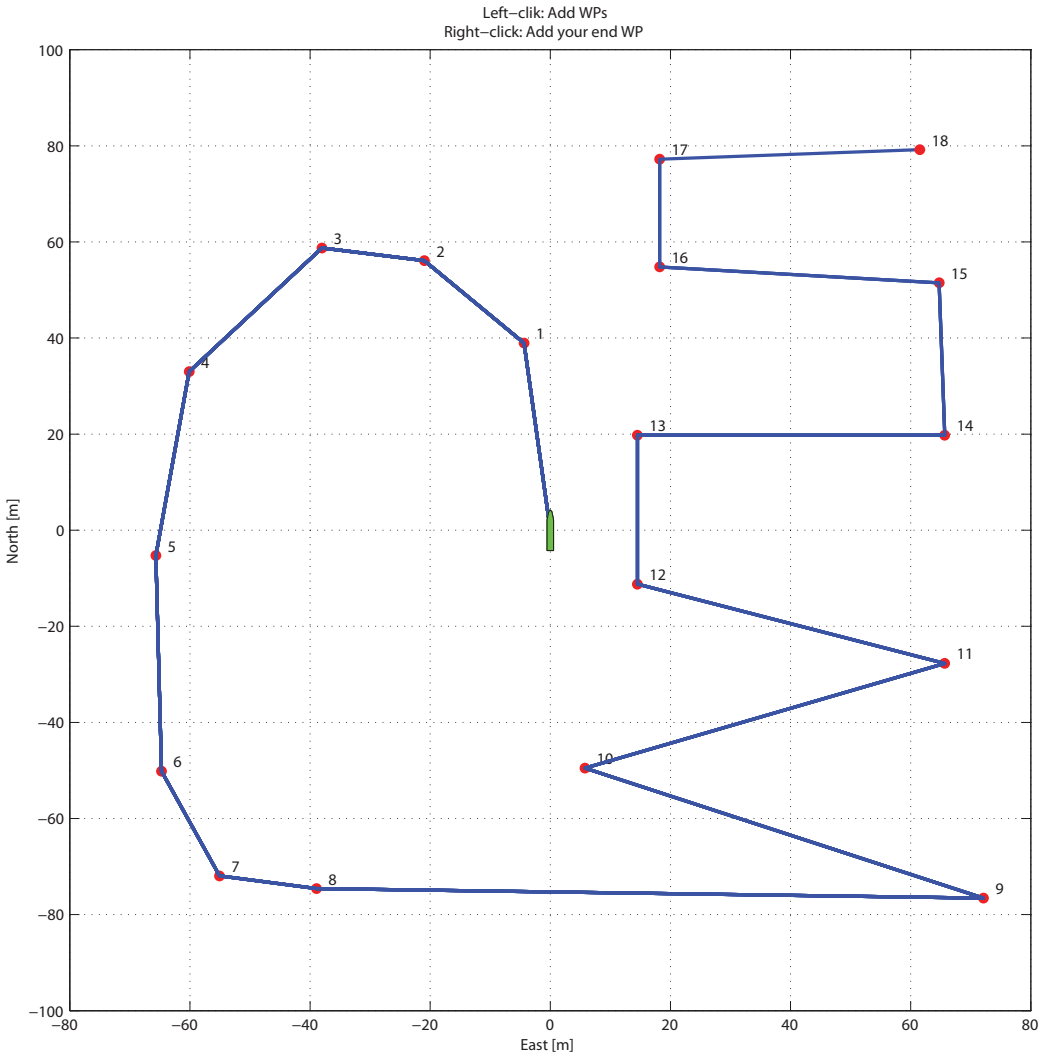

Figure 6.1: Example of how waypoints can be set graphically

the WP vector. Predefined fixed WP paths, e.g. zig-zag, pulse or spirals can be chosen manually for comparison of guidance algorithms or controller parameters.

6.2 Simulator Structure

The simulator structure is divided into three main categories; the guidance systems, controllers and models. The WP vector is input to the guidance system. This vector can either be made in the graphical waypoint generator

interface, or be one of the predefined WP setups. If path-following is employed, the WP vector is input to the path generator that generates a path according to the parameters set (tangent vector calculation method and vehicle constraints). There is a distinction in the structures of the simulators, depending on whether the Nomoto model or the Viknes boat model is applied. For the Nomoto model simulator, a preset constant vehicle speed is selected. Therefore, only the heading is output from the guidance system. A heading reference model saturates yaw rate and yaw acceleration to minimize the controller error. The Nomoto model takes the rudder deflection allocated by the controller as its input. Output from the Nomoto model is the vehicle heading and position.

Basically, the same structure applies for the 6 DOF Viknes boat model simulator as well. However, machinery and the actuators are modeled in detail, and environmental disturbances are added. For a thorough explanation on this the reader is encouraged to see the master's thesis of Kjerstad [27].

Waypoints can be set as a predefined setup or generated with the graphical WP generator. The waypoints may be fed to the path generator if a path-following task shall be performed. A few different guidance algorithms are implemented in the guidance block. The WP switcher that handles the check if the vehicle position is within the acceptance radius and switches waypoint accordingly is also implemented there (see Figure 6.2 and 6.3).

The path plotter was originally developed for student assignments in the course *TTK 4190 Guidance and Control* at the Norwegian University of Science and Technology. The plotter is, however, modified and adopted to appropriate dimensions pursuant to *Viknes 830*.

For future developers' convenience, the MATLAB function `boat_animation.m`[1] produces a movie of the vessel run. It aids the developer with a visual representation of the vessel response, attitude and velocity along the trajectory, and can be used as a coherent tool to the position- and attitude data for a better overall understanding of the vessel motion.

[1]The m-file can be found in the digital appendix

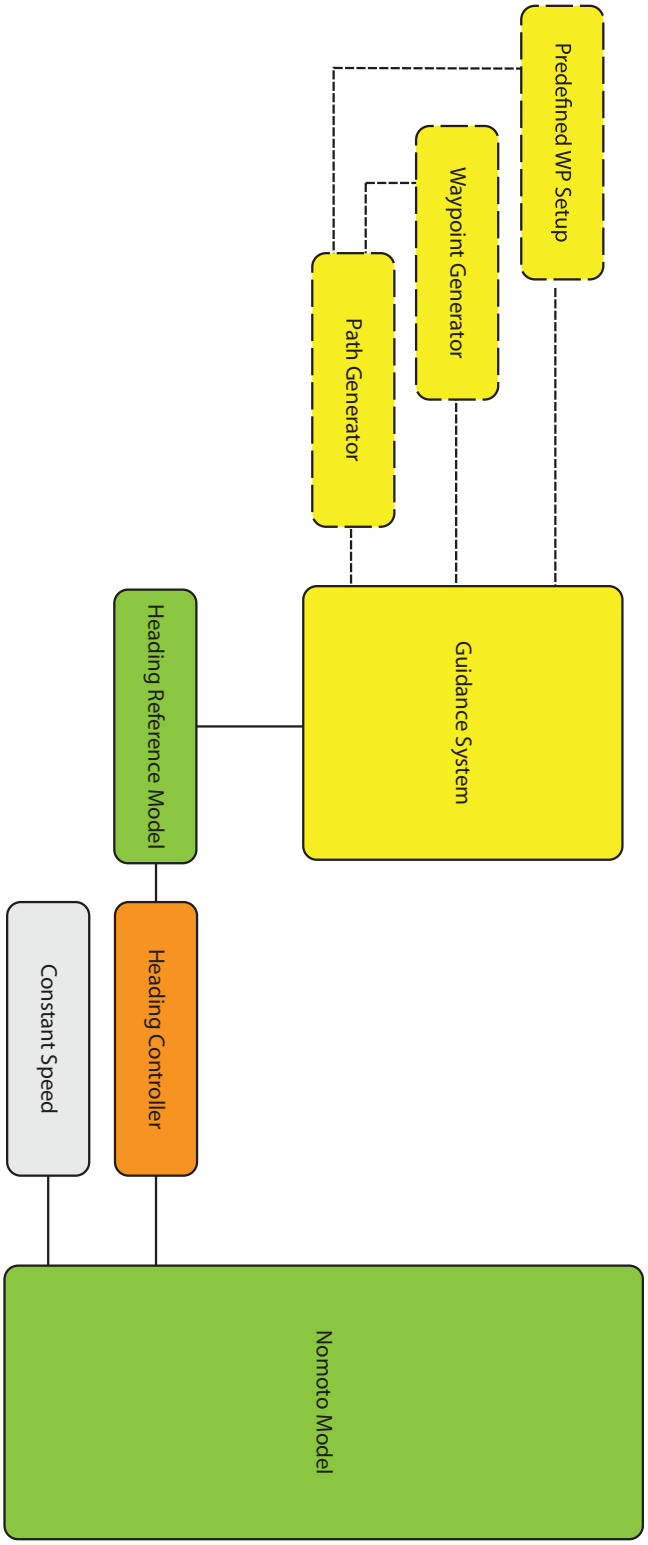

Figure 6.2: Overview of the simulator with the Nomoto model implemented

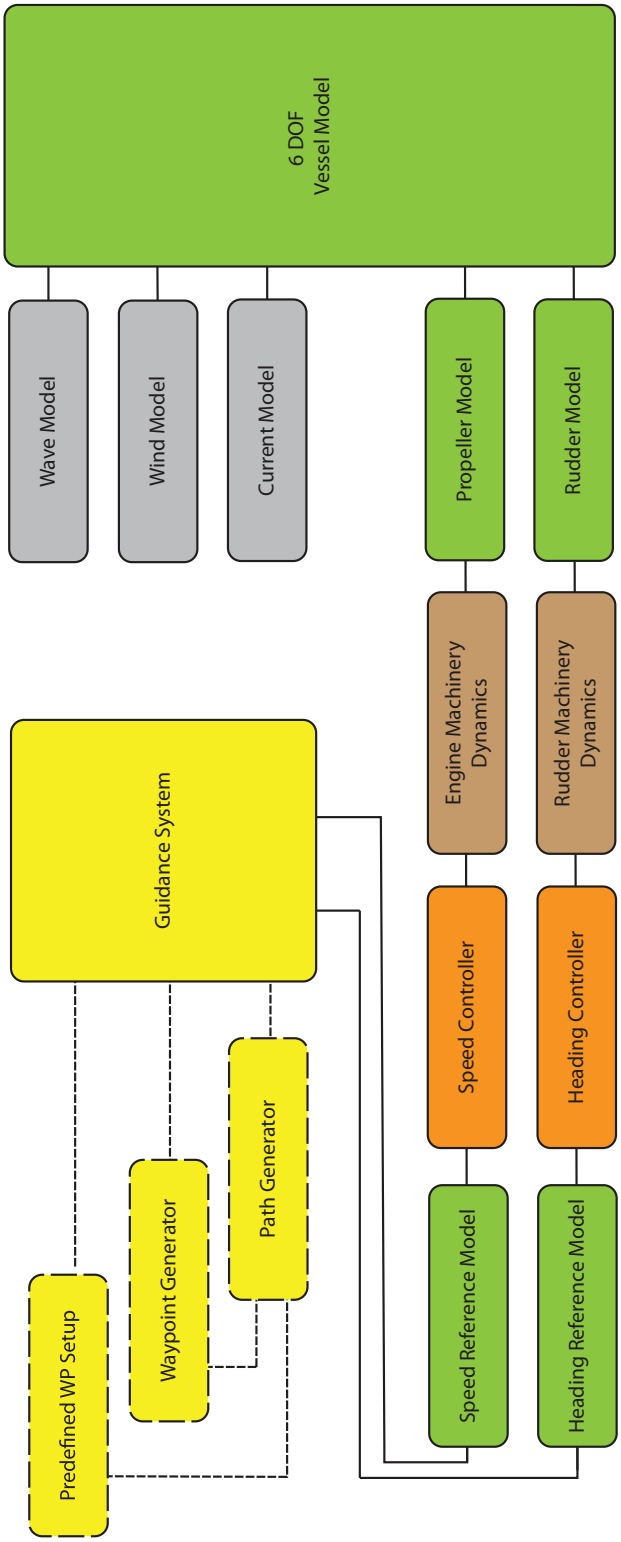

Figure 6.3: Overview of the simulator with the 6 DOF Viknes boat model implemented

Chapter 7

Results and Discussion

This chapter presents the results of the algorithms for removing, inserting and relocating waypoints for infeasible paths. A case study with simulations of direct waypoint-following and path-following, examining the theory put forward in earlier chapters, is carried out.

Discussion of the results for the path modifying algorithms is done prior to the respective plots.

7.1 Removing Waypoints

The maximum curvature of the spline created with method[1] *1* (Catmull-Rom tangents) is in general higher than the three other tangent calculation methods. This results in a relatively straight-on course from one waypoint to another, and because of this the turns often become sharp. High path curvature is unfavorable if the turning radius of the vehicle is high. On the other hand, if the vehicle dynamics support challenging turns, paths from Catmull-Rom tangents are shorter since the preturn aspect palls.

Given an arbitrarily produced path with 38 waypoints set by the graphical waypoint generator interface, experiments on waypoint removals based on path curvature could be carried out. Splines were calculated with the four different tangent methods described in chapter 2. A fictitious vehicle with curvature limit $\kappa_{vehicle} = 1.0\ m^{-1}$ at lowest possible traveling speed was proposed. Thus, decreasing the velocity to obtain lower $\kappa_{vehicle}$ became impracticable. The minimum turning radius yields: $\rho_{min} = \frac{1}{\kappa_{vehicle}} = 1.0\ m$.

[1]The vector tangent calculation methods (2.26)-(2.29) suggested in Chapter 2 are examined for all three algorithms and referred to as: Method *1-4*.

According to the WP removal algorithm, waypoints were eliminated if $\kappa_{path} > \kappa_{vehicle}$.

For tangent calculation method *1* and *2*, a significant number of waypoints have been removed; 32 and 26 respectively. This is apparently too many to recreate a fairly good approximation of the original path. Despite the removal of 26/38 waypoints, tangent method *2* recreates the first 2/3 of the path quite well. The last 1/3 of the path however is nearly ignored because of exceeding curvature values, κ_{path}. The path created with Catmul-Rum tangents is far from feasible with respect to restoration of the original path, and should be rejected immediately. It can be seen that the paths calculated from the mentioned methods are not satisfactory. Nevertheless, the remaining new paths do not exceed the maximum value of $\kappa_{vehicle}$; see Figure 7.3.

Method *3* and *4* yield more satisfactorily results; removing 2 and none WPs, respectively, while still complying with the requirement of $\kappa_{path} > \kappa_{vehicle}$. Method *3* removes two WPs at the bendy part towards the end of the path. The selection of whether these WPs can be ignored or not is up to the operator to determine, but it is often defined by the area of application. Important to notice is that after removing two waypoints for method *3*, the maximum curvature is lower than for method *4* which is unaltered; see Figure 7.2.

A more proper value of $\kappa_{vehicle}$ with respect to small maritime vessels is $\kappa_{vehicle} = 0.4\ m^{-1}$. Thus, $\rho_{min} = \frac{1}{\kappa_{vehicle}} = 2.5\ m$. The path of tangent method *1* exceeds the minimum number of WPs. The WP remover algorithm terminates at 4 remaining WPs and the remaining path is dissatisfactory in several ways, including too high path curvature that results in an infeasible path. Method *2* which indicated good results on the first 2/3 of the path in the former case, is now excluding WPs halfway through the path. In spite of meeting the requirements of κ_{path}, the new path omit WPs that causes the path to be infeasible with respect to recreate a tolerably similar path. The path from method *2* has the lowest κ_{path} of the 4 methods, but with that many WPs removed the path should yet be rejected.

The WP remover algorithm leaves the path from method *3* unchanged, as expected. Method *4* however now results in the removal of 4 waypoints. Despite this reduction in WPs, maximum curvature for method *4* is still higher than method *3*.

In the case of $\kappa_{vehicle} = 0.4\ m^{-1}$ a reasonable path would be the one calculated from method *3*, with only 2 eliminated waypoints. The overall trajectory is

fairly recreated despite the removal of a few WPs. The path generation with method *3* and method *4* yield generally better result (lower path curvature) and is usually preferred for path generation. Notwithstanding, Catmull-Rom tangents can be a good choice if the maneuverability is high and shortest distance is emphasized.

It is worth noticing that method *4* avoided removing any waypoints for $\kappa_{vehicle} = 1\ m^{-1}$ while the same method removed 4 waypoints for $\kappa_{vehicle} = 0.4\ m^{-1}$. Comparing this result with method *3* that removed 2 waypoints for both curvature limits detects an interesting fact: Which tangent calculation methods that works the best is dependent of the waypoint layout and the curvature limit. Thus, a combination of the different tangent calculation methods should preferable be carried out to minimize overall path curvature. This is however left out for future work.

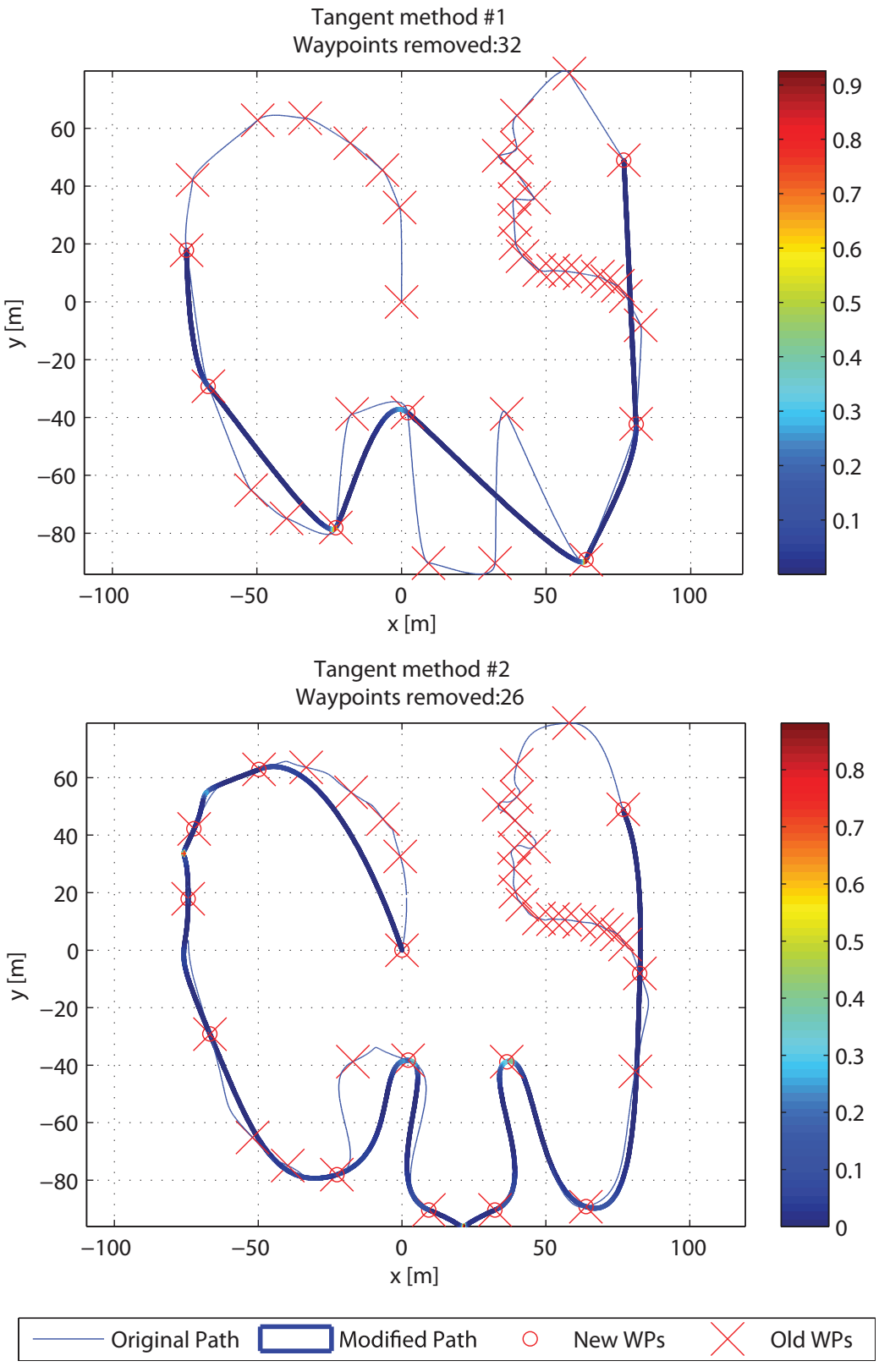

Figure 7.1: Paths with removed WPs and tangent calculation method 1 and 2. $\kappa_{vehicle} = 1.0\ m^{-1}$

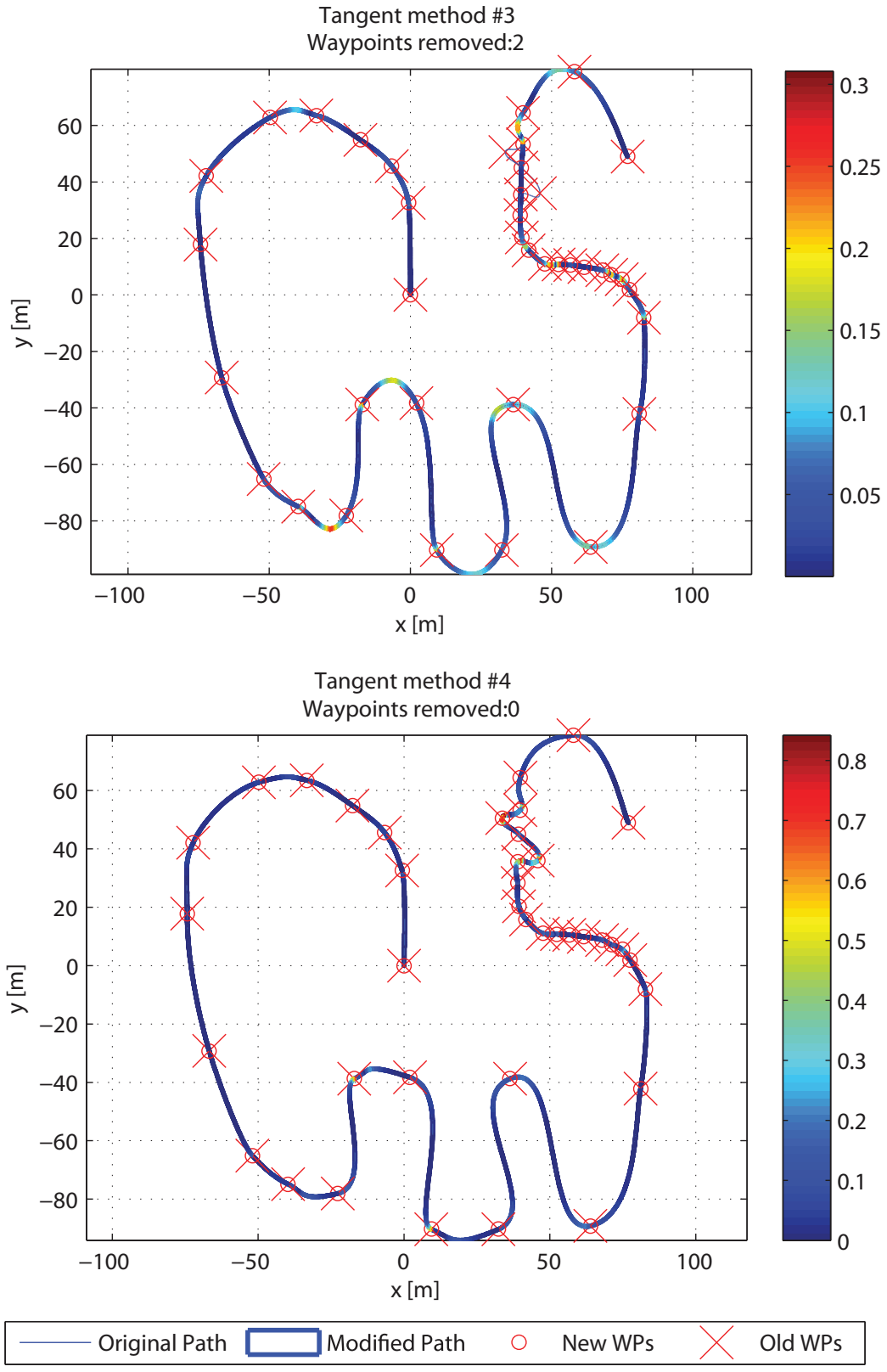

Figure 7.2: Paths with removed WPs and tangent calculation method 3 and 4. $\kappa_{vehicle} = 1.0\ m^{-1}$

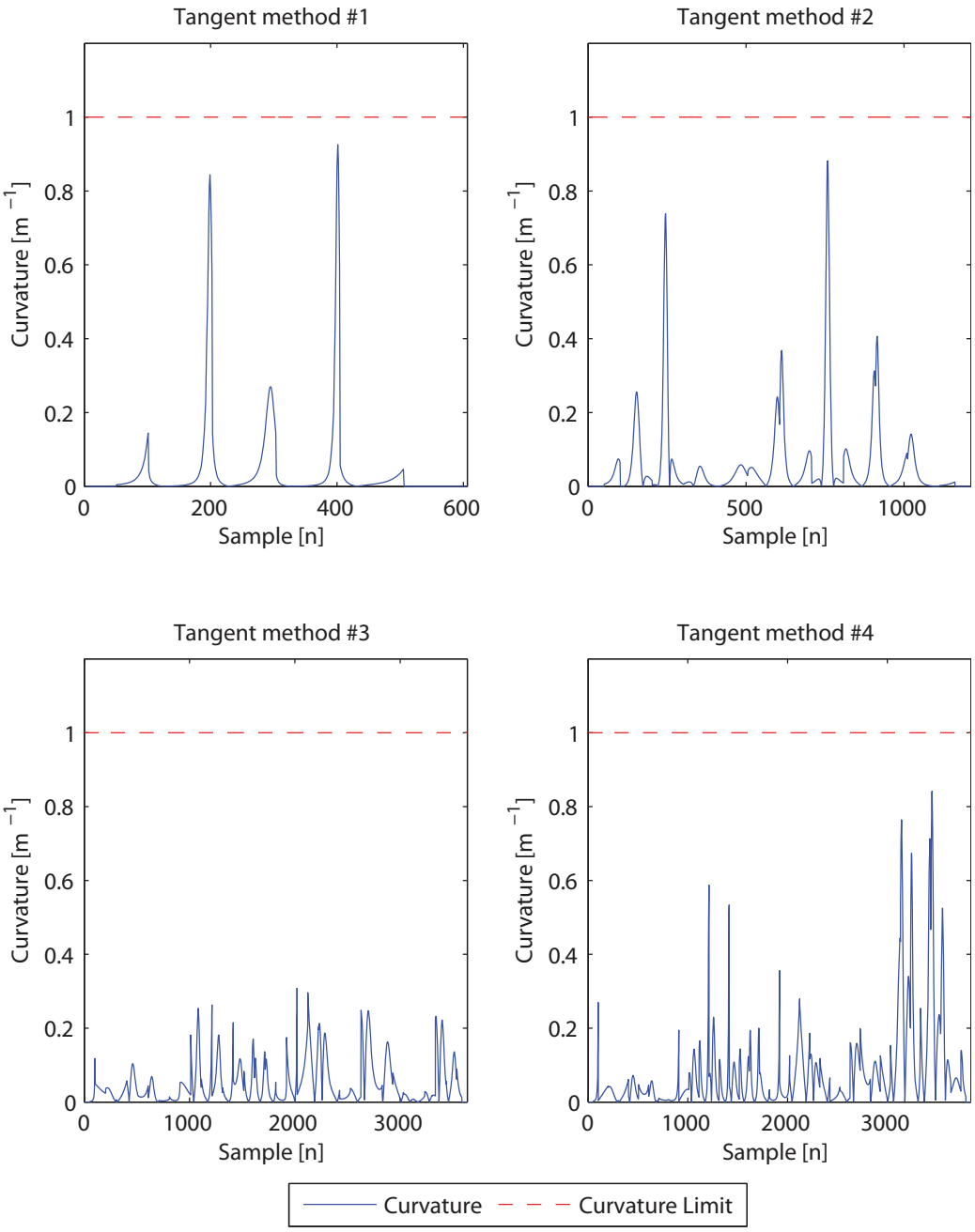

Figure 7.3: Path curvature values for removing WP algorithm for the four different vector tangent calculation methods for $\kappa_{vehicle} = 1.0\ m^{-1}$

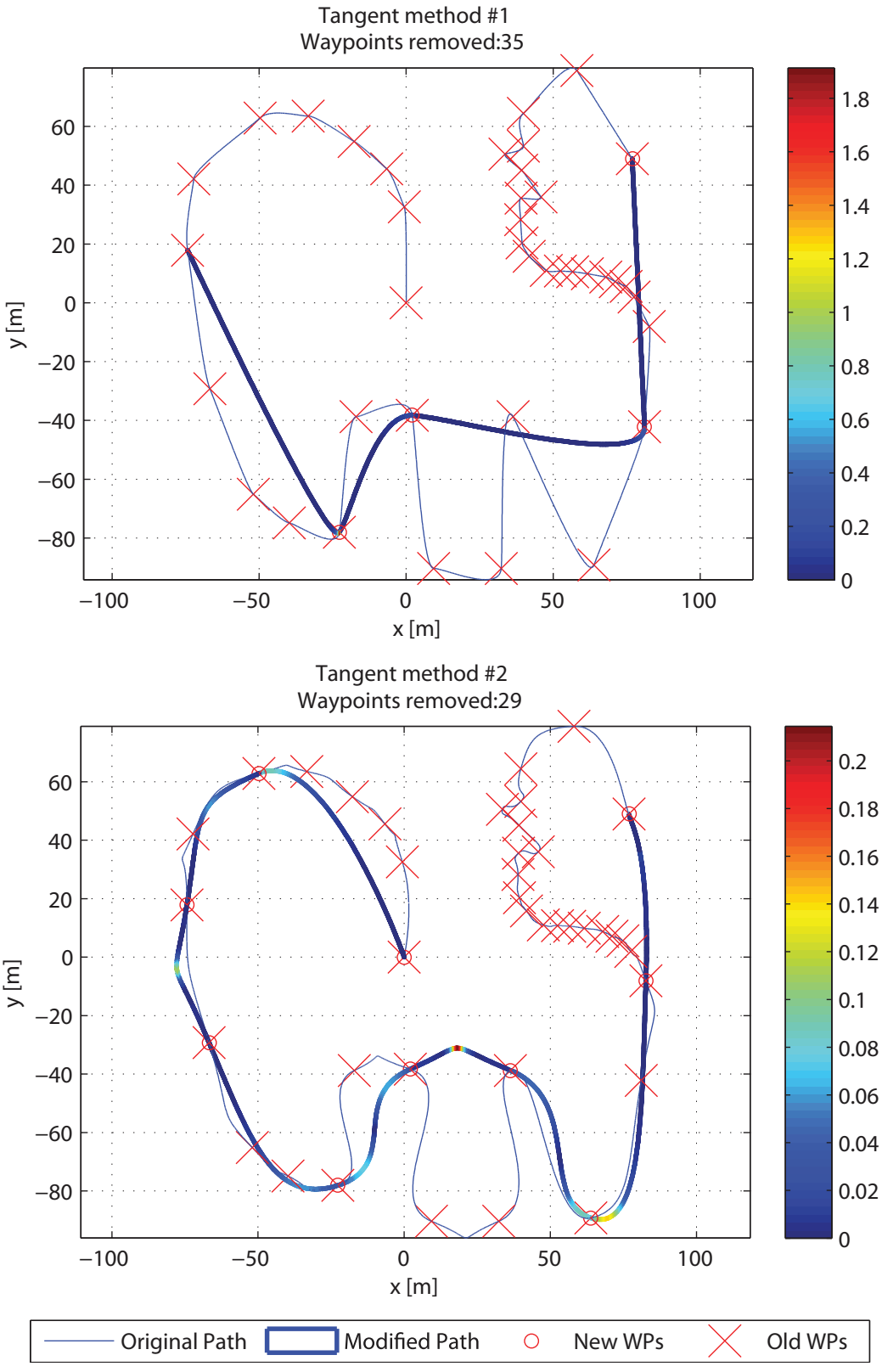

Figure 7.4: Paths with removed WPs and tangent calculation method 1 and 2. $\kappa_{vehicle} = 0.4\ m^{-1}$

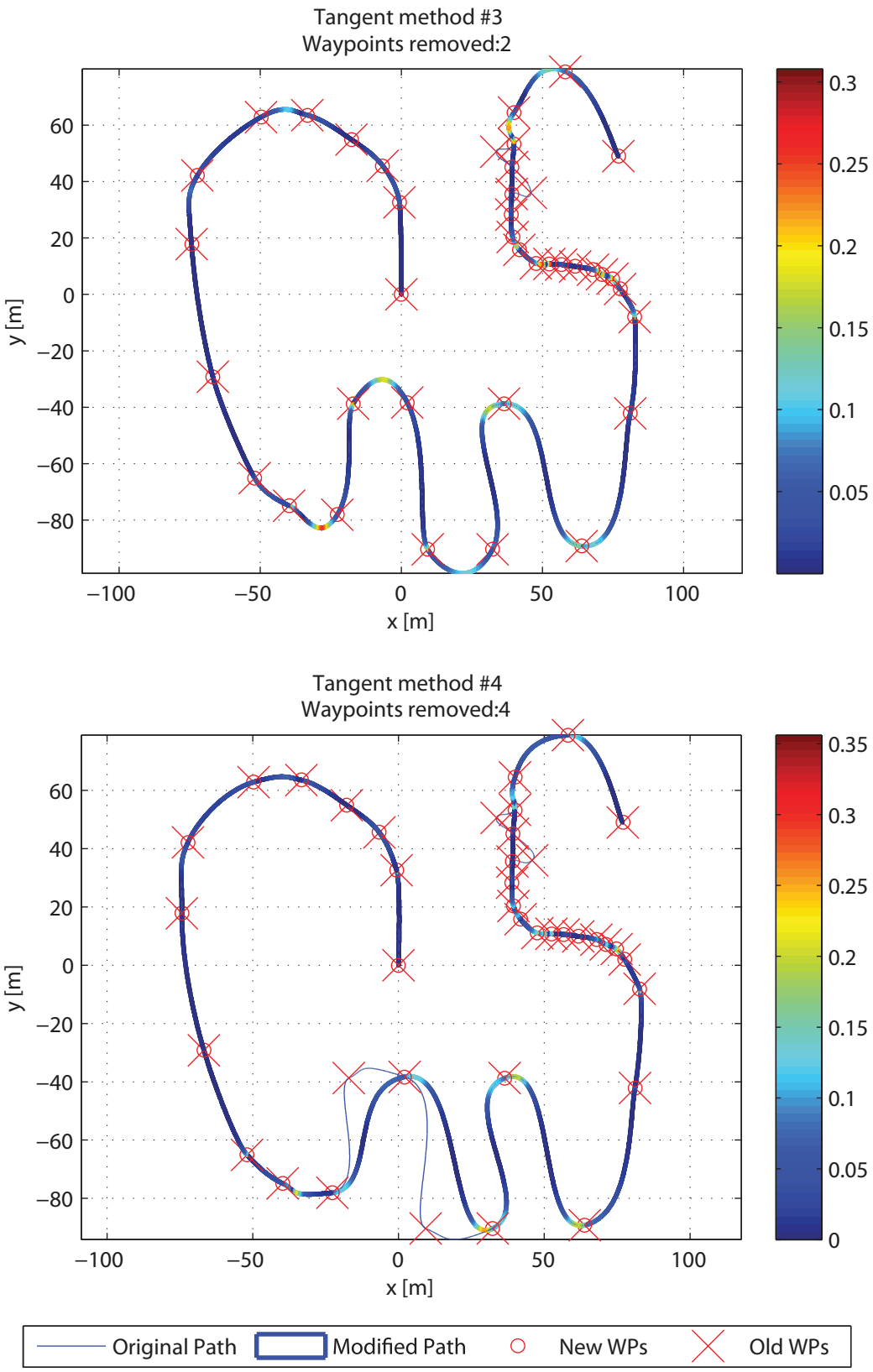

Figure 7.5: Paths with removed WPs and tangent calculation method 3 and 4. $\kappa_{vehicle} = 0.4\ m^{-1}$

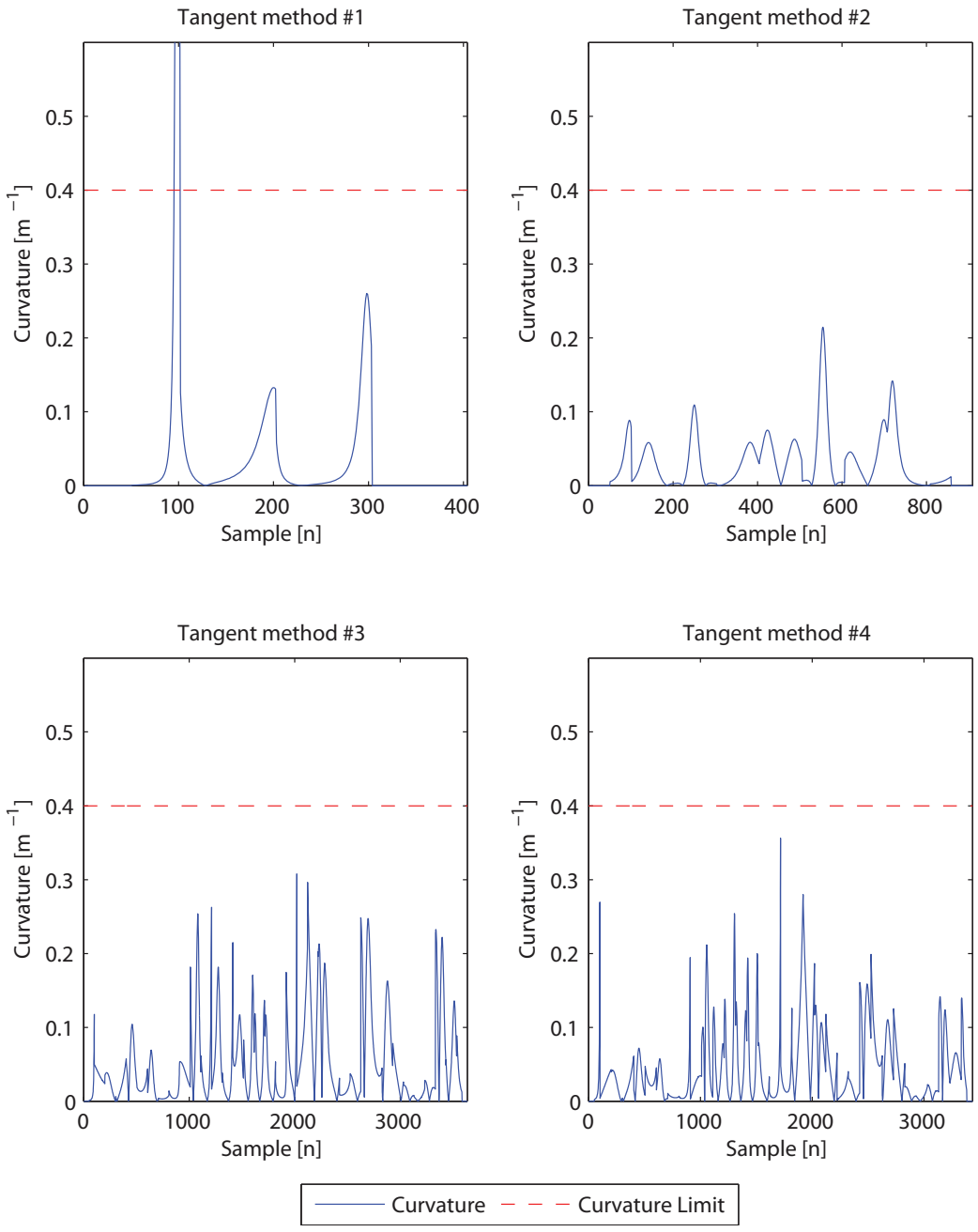

Figure 7.6: Path curvature values for removing WP algorithm for the four different vector tangent calculation methods for $\kappa_{vehicle} = 0.4\ m^{-1}$

7.2 Inserting Waypoints

By removing waypoints, the guarantee of making a feasible path that intersects every original waypoint disappear. On the other hand, inserting waypoints when the path curvature does not comply with the vehicle dynamics, makes it possible to re-plan the path to meet the curvature requirement. The algorithm of WP insertions is described in Chapter 3. This algorithm is tested for the identical WP setup as for WP removal.

As in the experiment with removing waypoints, the vehicle curvature limit $\kappa_{vehicle}$ is set to 1 m^{-1}. Three new WPs are inserted between two original waypoints if the curvature of the path κ_{path} is greater than $\kappa_{vehicle}$. The added WPs form a non-crossing arc.

For tangent method *1* (Catmull-Rom), too large curvature values were encountered at the end of the original path, where there are two sharp turns. The problem with Catmull-Rom tangents is that since their magnitude is low, the preturn characteristic is not present in the same way as for the other tangent calculation methods. Thus, only three inserted waypoints with the orientation given in this test do not result in a new path with curvature lower than the threshold of $\kappa_{vehicle} = 1.0\ m^{-1}$. As can be seen in Figure 7.9 for tangent method *1*, the curvature of the path created by the inserted waypoints is at $\kappa_{path} \approx 2.75\ m^{-1}$. The placement of the inserted WPs are obviously of importance, and for Catmull-Rom splines even more WPs should have been added to create a smoother turning circle. A minimum requirement is that the curvature of the new turning circle is lower than the threshold of the vehicle; if not, this approach is useless since the algorithm will keep iterating over the entire curvature and add new WPs that keeps resulting in a path with $\kappa_{path} > \kappa_{vehicle}$. After $n - 1$ iterations, where n is the length of the original WP array, the algorithm terminates. Owing to the fact that inserting WPs should decrease the curvature between two WPs where the original path curvature exceeds the curvature limit, a worst-case scenario will be that the original path curvature is too high between every two consecutive WPs, therefore adding WPs $n - 1$ times means that the curvature between every consecutive WPs has been investigated. Evidently, Catmull-Rom tangents do not yield satisfactory results for the configuration of the inserted WPs.

In general, tangent method *2* results in a curvature of $\kappa_{vehicle} \approx 0.45\ m^{-1}$ for the three newly added WPs, although, this curvature depends on the position of the original waypoints. For the critical area at the end of the path, the somewhat similar problem as for method *1* occurs. New waypoints

are inserted, although the new path becomes very bendy and inefficient. A problem for method *2* is the tangent magnitude scaling when changing from long to short distance between the WPs. As a result of this, the path obtains unfortunate high curvature where e.g. method *3* and *4* yields satisfactorily results. Method *2* indicates poor performance when inserting WPs.

Even though both method *1* and *2* fail in this attempt of inserting WPs, they could benefit from a different WP configuration. In addition to adding more WPs, the result is altered with different WP placement. Important to notice is that a different placement of the inserted WPs would have yielded a different results. This goes for both the number of WPs added and the distance from the fictive straight-line between the added waypoints.

The modified path for method *3* and *4* achieves better results. Naturally, method *4* does not imply any changes of the path since $\kappa_{vehicle} = 1.0\ m^{-1}$. and $\kappa_{path} < \kappa_{vehicle}$ for the entire path. Method *3* however inserts WPs according to the WP insertion algorithm with non-crossing arc. It is demonstrated that the new path has decreased curvature and therefore comply with the curvature path threshold.

Since method *1* and *2* failed for $\kappa_{vehicle} = 1.0\ m^{-1}$, there will be no amendment in the results for $\kappa_{vehicle} = 0.4\ m^{-1}$. As for the removing WP algorithm with the identical WP setup as in this experiment, there was no change in the paths for method *3* when modifying the vehicle curvature limit from $1\ m^{-1}$ to $0.4\ m^{-1}$. This is given by the fact the path curvature values for method *3* is beneath the new $\kappa_{vehicle}$ threshold (see Figure 7.9).

Method *4* adds 18 WPs to the original path to keep up with the vehicle curvature requirement. Thus, 6 critical areas have been altered. A modification of the path can easily be observed in Figure 7.10 at the 14'th WP (where WP_1 is at the origin). The path at this section, where κ_{path} originally is too high, is now altered to make a smoother turn with lower curvature. Even though the curvature threshold is met, a shorter turn could have been carried out, thus the placement of new WPs is not efficient with respect to e.g. fuel efficiency. This specific section however demonstrates a good application of WP insertion; resulting in a smooth directed turn with decreased curvature.

To achieve even better results, more emphasis should be put into the WP placement when inserting new WPs. In that case, both the number of WPs, the WP placement and the orientation (see Figure 3.9) should be taken into account. For the case of removing WPs, method *3* performs most satisfac-

tory. Despite the possibility of generating a feasible path by adding WPs, an unfortunate effect of crossing back and forth over a relatively straight line can occur, if the WP configuration is set poorly and wrong tangent calculation method is employed. If a zig-zag run takes place (as for method *2* in Figure 7.7), the path should be optimized with respect to desired heading when approaching a WP. Such optimization could result in e.g. a wider turn to get back on the original path with the original desired heading, and is accommodated in further work on this subject.

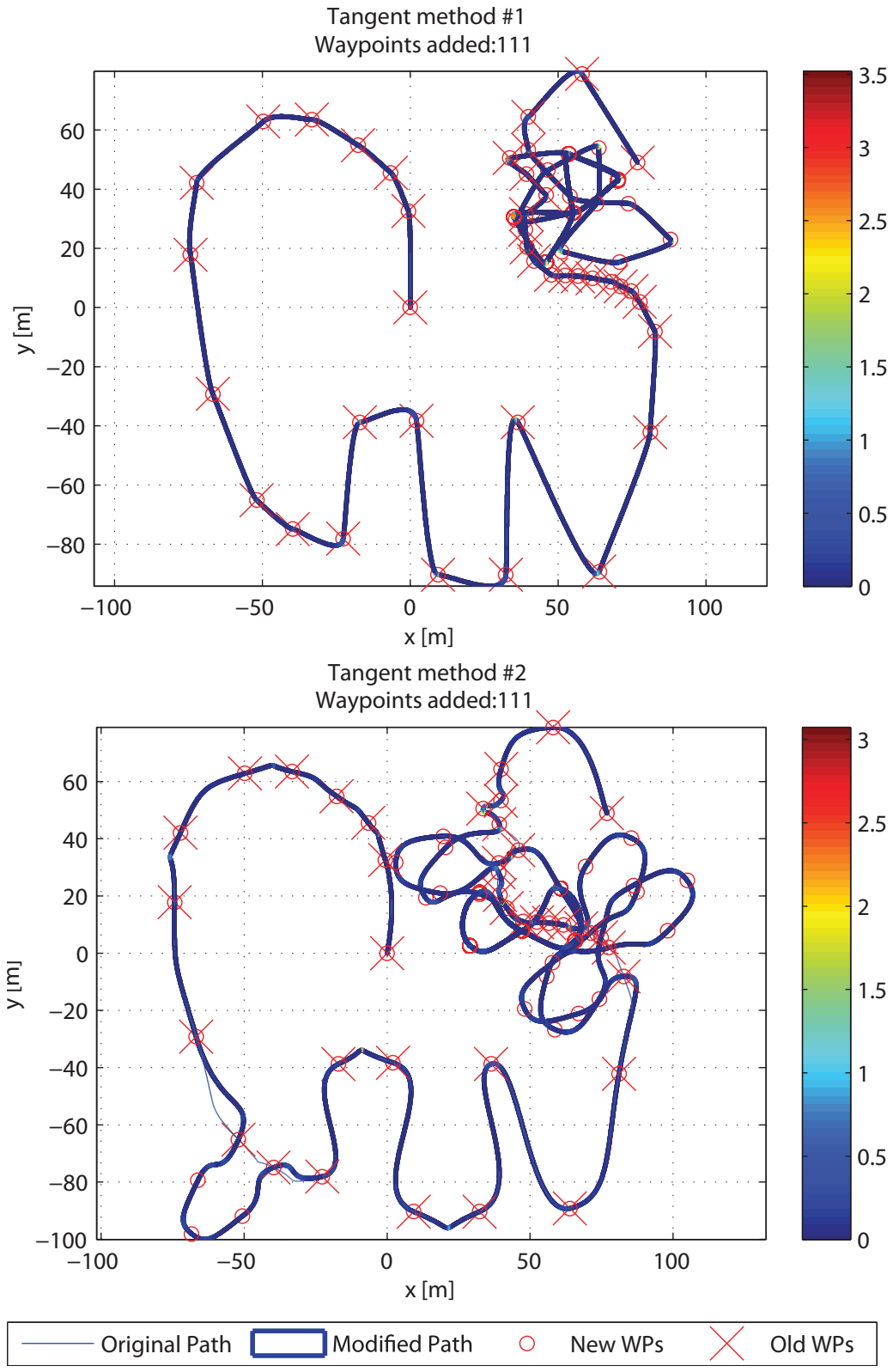

Figure 7.7: Paths with inserted WPs and tangent calculation method 1 and 2. $\kappa_{vehicle} = 1.0\ m^{-1}$

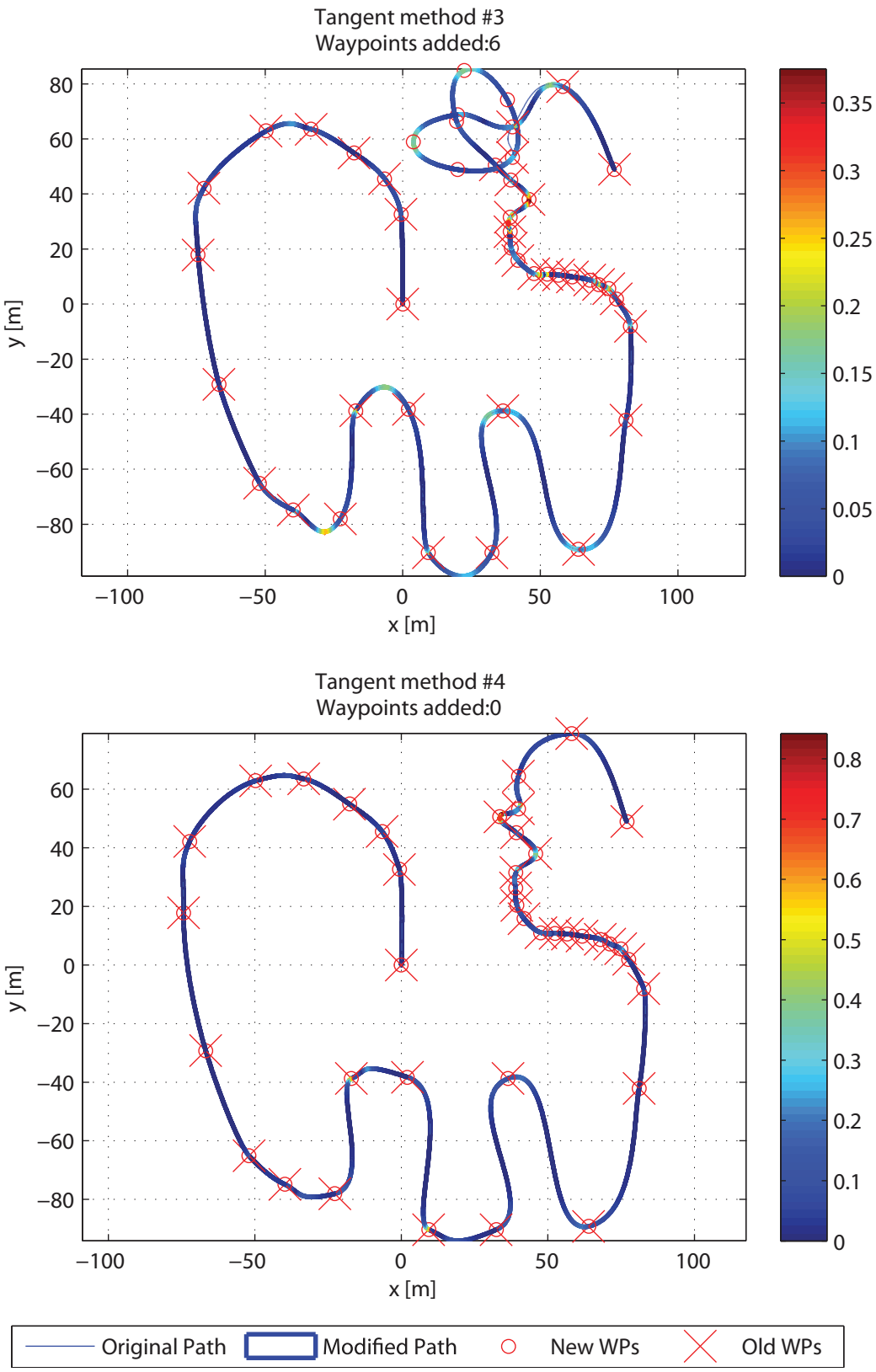

Figure 7.8: Paths with inserted WPs and tangent calculation method 3 and 4. $\kappa_{vehicle} = 1.0\ m^{-1}$

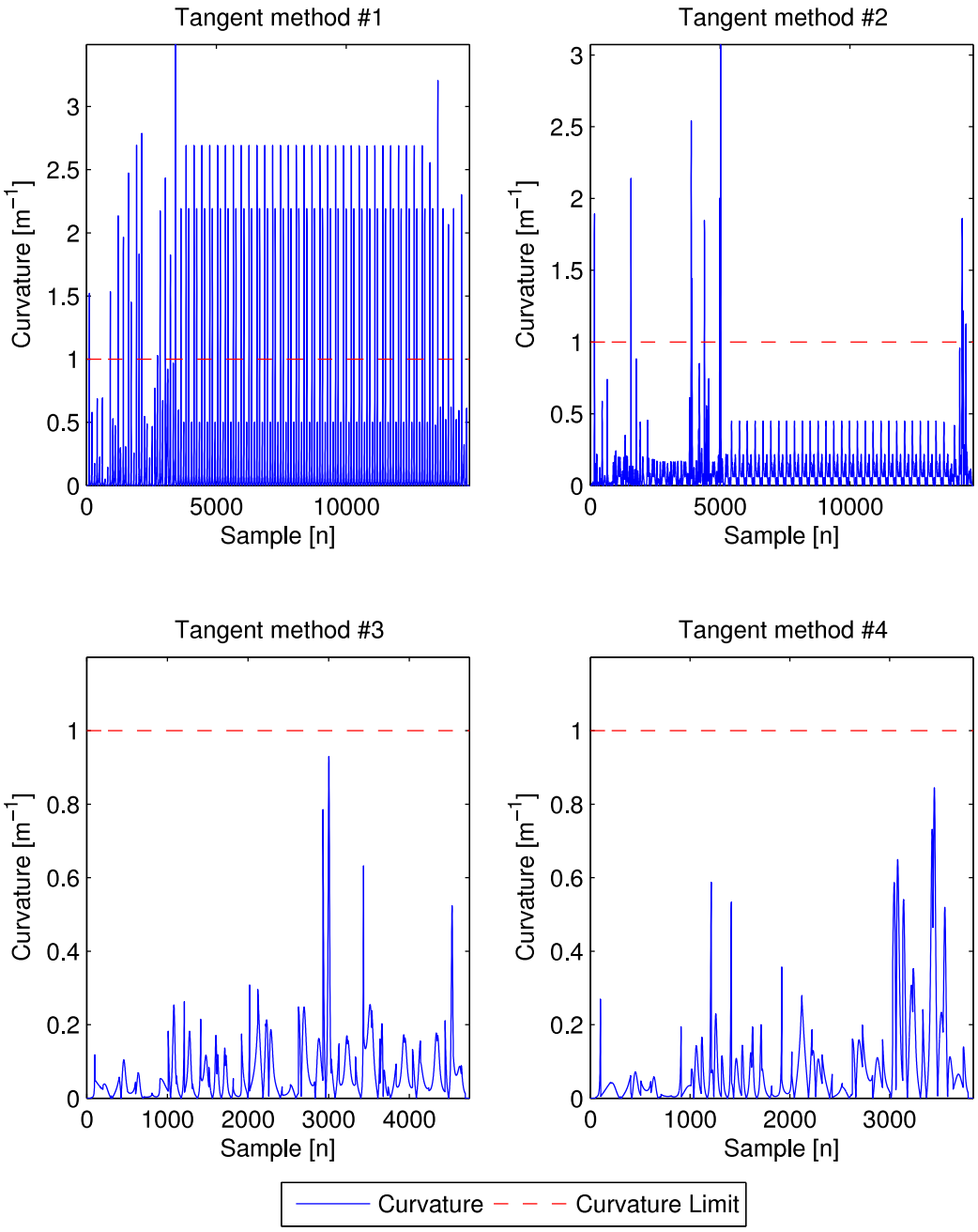

Figure 7.9: Path curvature values for WP insertion algorithm for the four different vector tangent calculation methods for $\kappa_{vehicle} = 1.0\ m^{-1}$

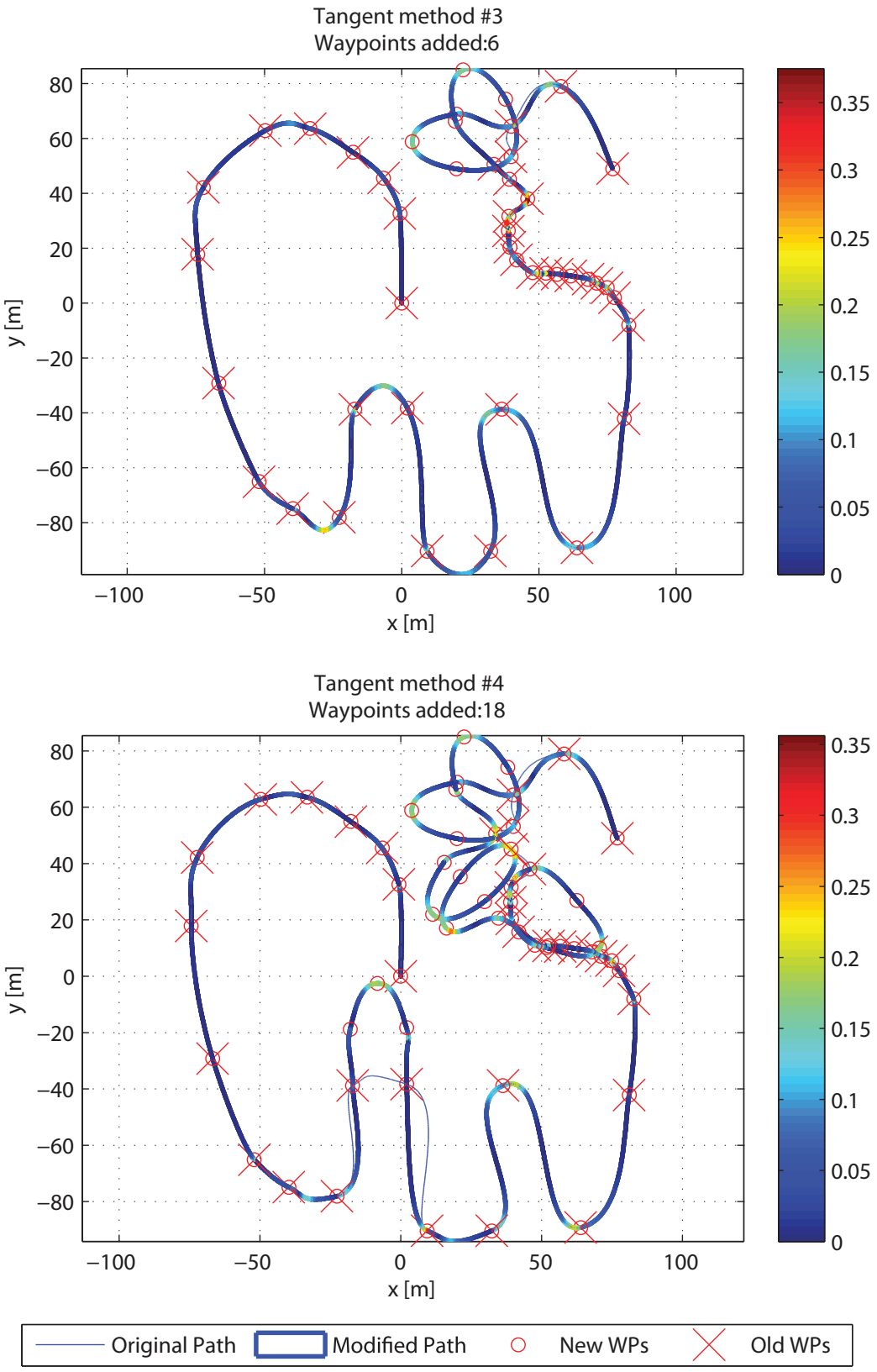

Figure 7.10: Paths with inserted WPs and tangent calculation method 3 and 4. $\kappa_{vehicle} = 0.4\ m^{-1}$

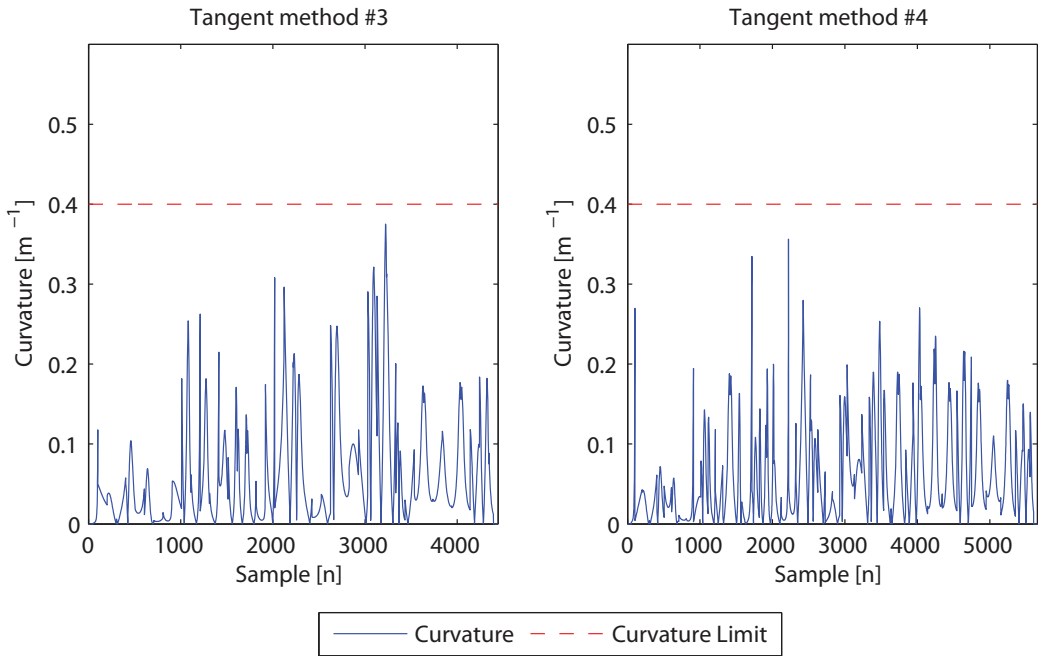

Figure 7.11: Path curvature values for WP insertion algorithm for tangent calculation method 3 and 4 for $\kappa_{vehicle} = 0.4\ m^{-1}$

7.3 Relocating Waypoints

The results of algorithms that remove and insert waypoints have been demonstrated in the previous section. A third method of developing a feasible path is suggested by relocating the original waypoints. The algorithm is explained in Chapter 3. The WP setup and curvature limits is identical to the preceding test cases. The relocating distance from the original WP towards the straight-line between the surrounding WPs is 0.75.

For this WP setup, relocating waypoints and creating a path with tangent calculation method *1* yield fairly good results compared to removing and adding WPs. κ_{path} does not exceed the curvature limit set to $\kappa_{vehicle} = 1.0\ m^{-1}$ (see Figure 7.14). The sharp turns are cut to meet the curvature prerequisite, thus the recreation of the entire path is hardly adequate. All the same, as far as the algorithms can be compared, this approach yields good results for Catmull-Rom tangents. Although, if intersection of all the original WPs is requested, it is given that relocating WPs fails.

Tangent method *2* is sensitive as regards to tangent magnitude when the distance between the consecutive waypoints is varying. The algorithm of relocating waypoints is therefore highly suited since the critical WP is pulled towards the midpoint of its two neighboring WPs, thus leveling the distance between the surrounding WPs. Even though nearly half of the WPs of the original path have been moved, the recreation of the initial path is good.

Tangent method *4* meets the path curvature requirements without relocating any WPs, while method *3* only results in a small perturbation of four of the last WPs in the original path. It is worth mentioning that the recreation of the initial path becomes somewhat more accurate for this case than for the method of removing the WPs completely. Based on these results, relocating the WPs is better suited for recreating the original path with satisfactorily path curvature for the given waypoint setup.

Results for $\kappa_{vehicle} = 0.4\ m^{-1}$ are discussed below. The relocating WP algorithm does not alter the path to comply with the path curvature for the Catmull-Rom tangents with a lower curvature limit, and so the resulting path is neither feasible, nor a good approximation of the original path.

As apposed to the cases of removing and inserting WPs, the method of relocating WP values yields adequate results for tangent method *2*. The characteristic turns halfway through the path are poorly imitated by the newly

generated path, but apart from that, the result of this tangent method is good.

Method *3* stays unchanged. Method *4* however, experiences small to medium perturbation in WP placement for the last and the central part of the path, respectively. Once again, the advantage of moving the WP a short distance becomes apparent; the characteristic bendy turns towards the end of the path are still present, while they are eliminated in the removing WP approach (see Figure 7.5).

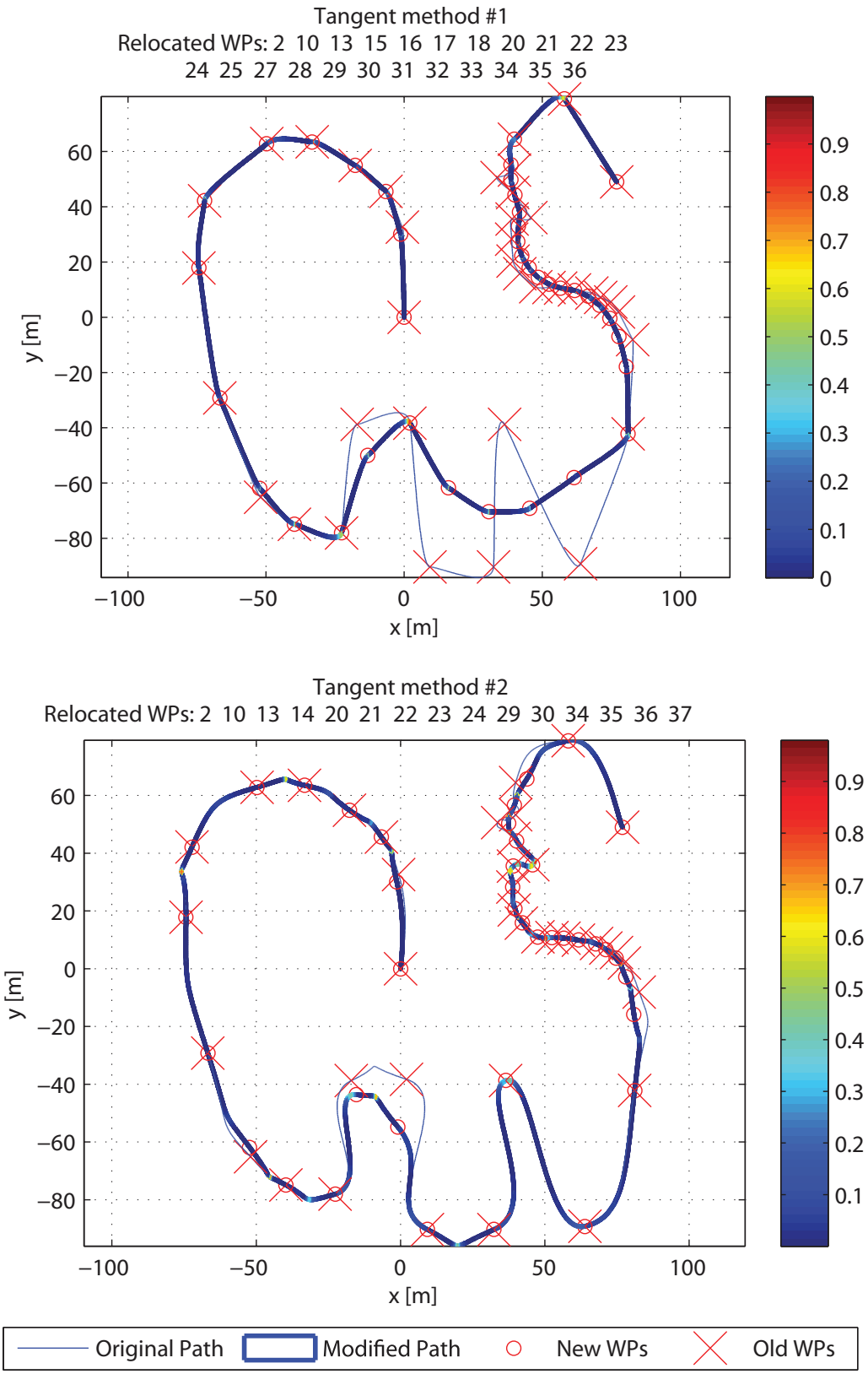

Figure 7.12: Paths with relocated WPs and tangent calculation method 1 and 2. $\kappa_{vehicle} = 1.0\ m^{-1}$

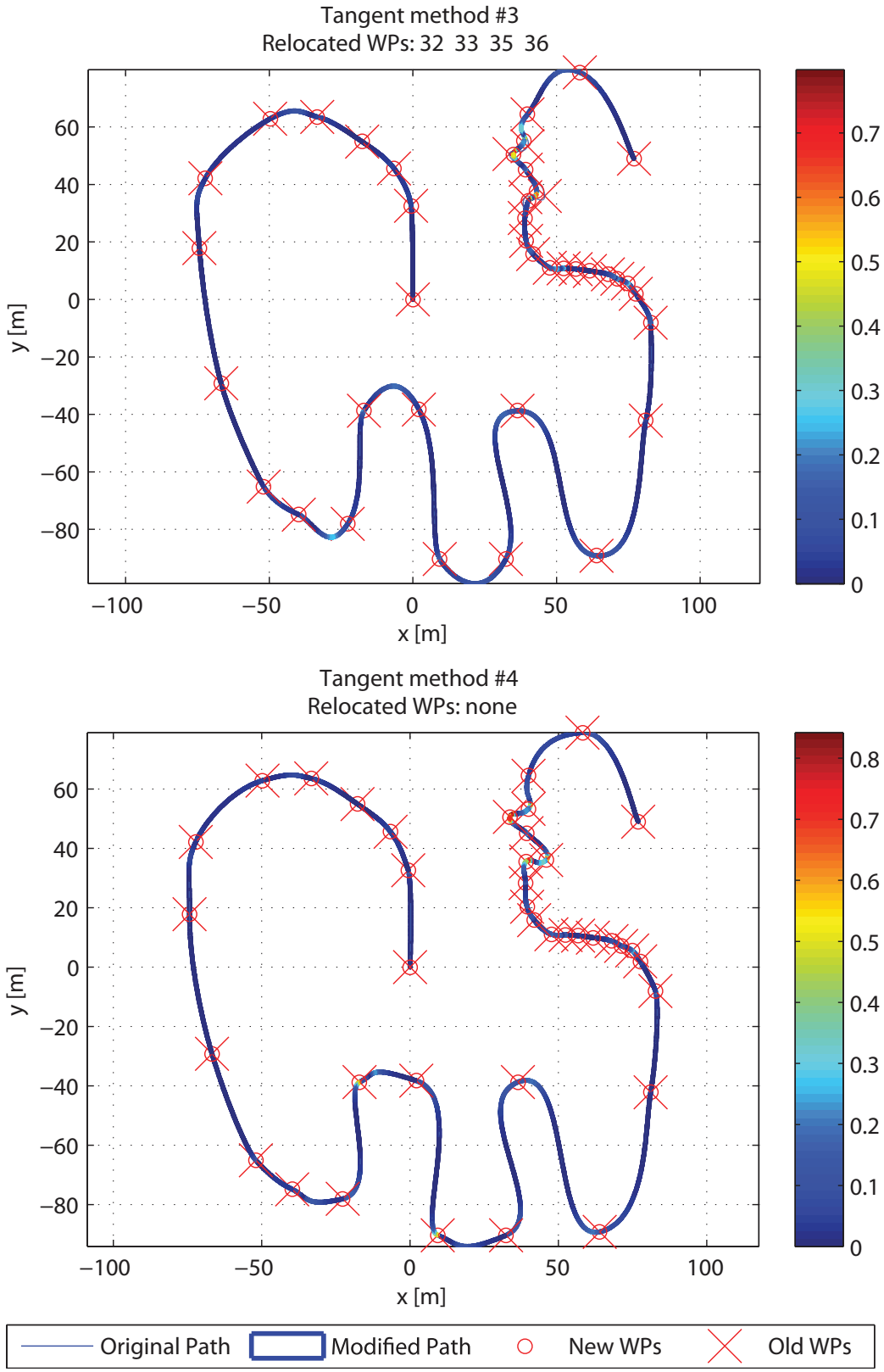

Figure 7.13: Paths with relocated WPs and tangent calculation method 3 and 4. $\kappa_{vehicle} = 1.0\ m^{-1}$

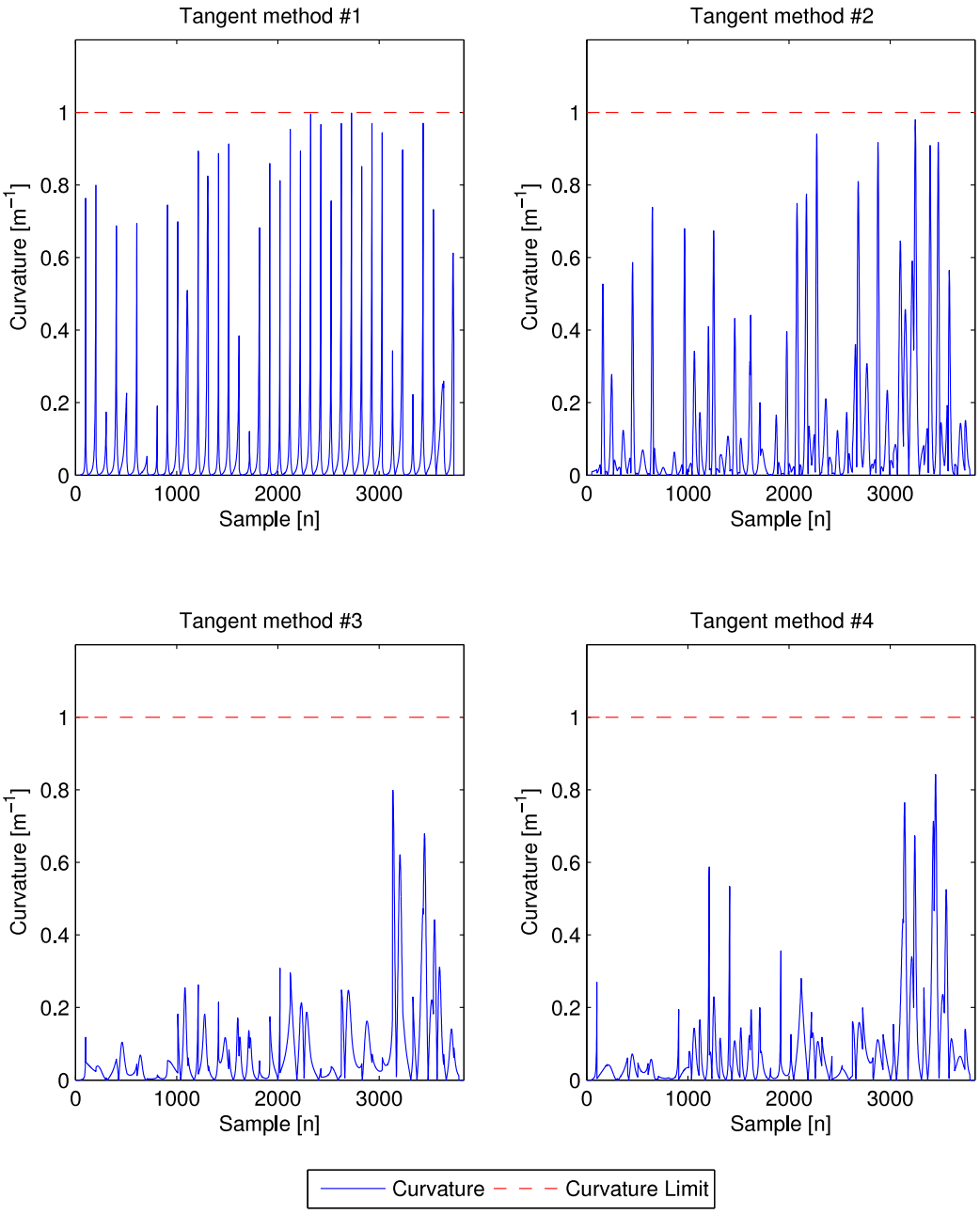

Figure 7.14: Path curvature values for Relocating WP algorithm for the four different vector tangent calculation methods for $\kappa_{vehicle} = 1.0\ m^{-1}$

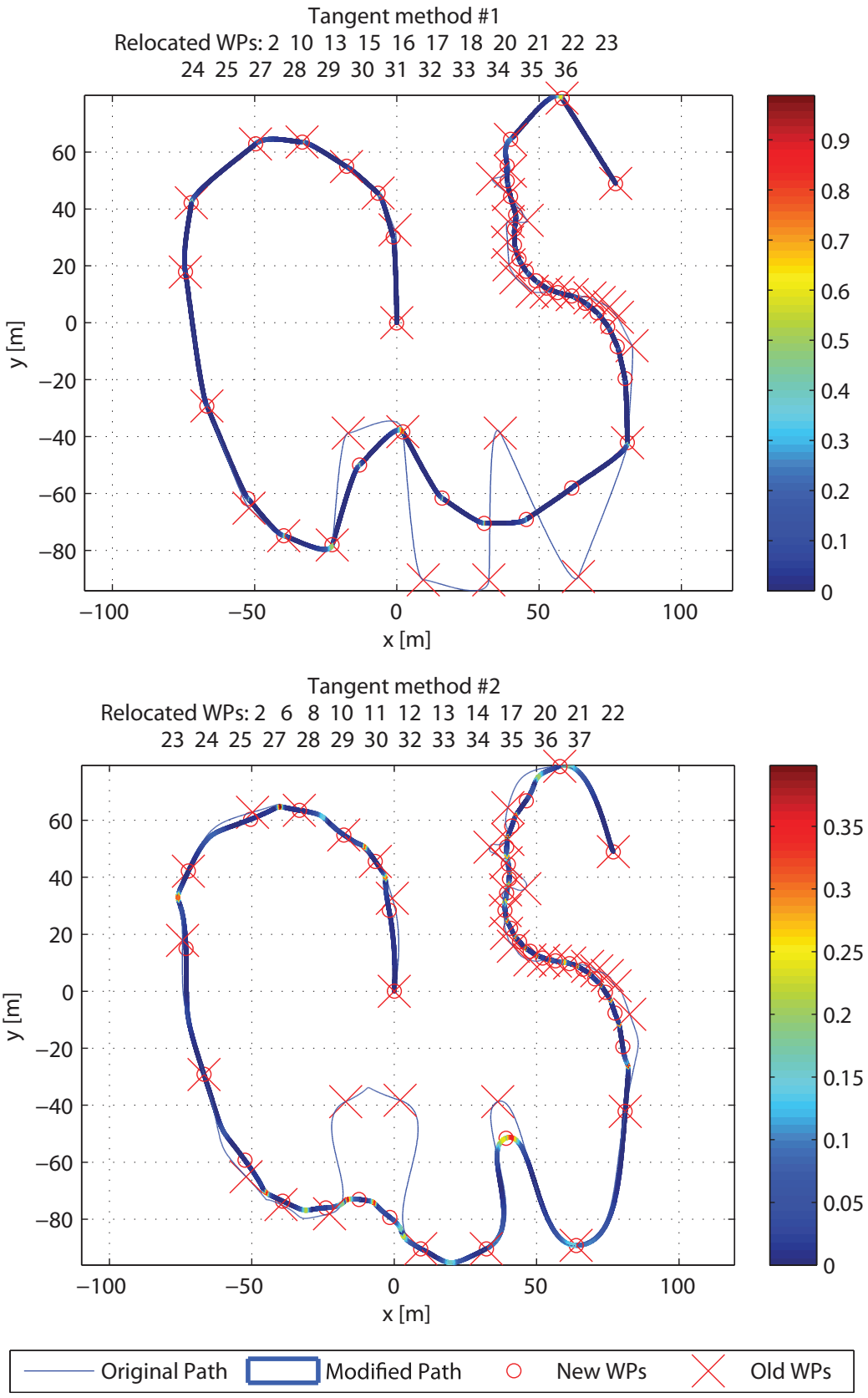

Figure 7.15: Paths with relocated WPs and tangent calculation method 1 and 2. $\kappa_{vehicle} = 0.4\ m^{-1}$

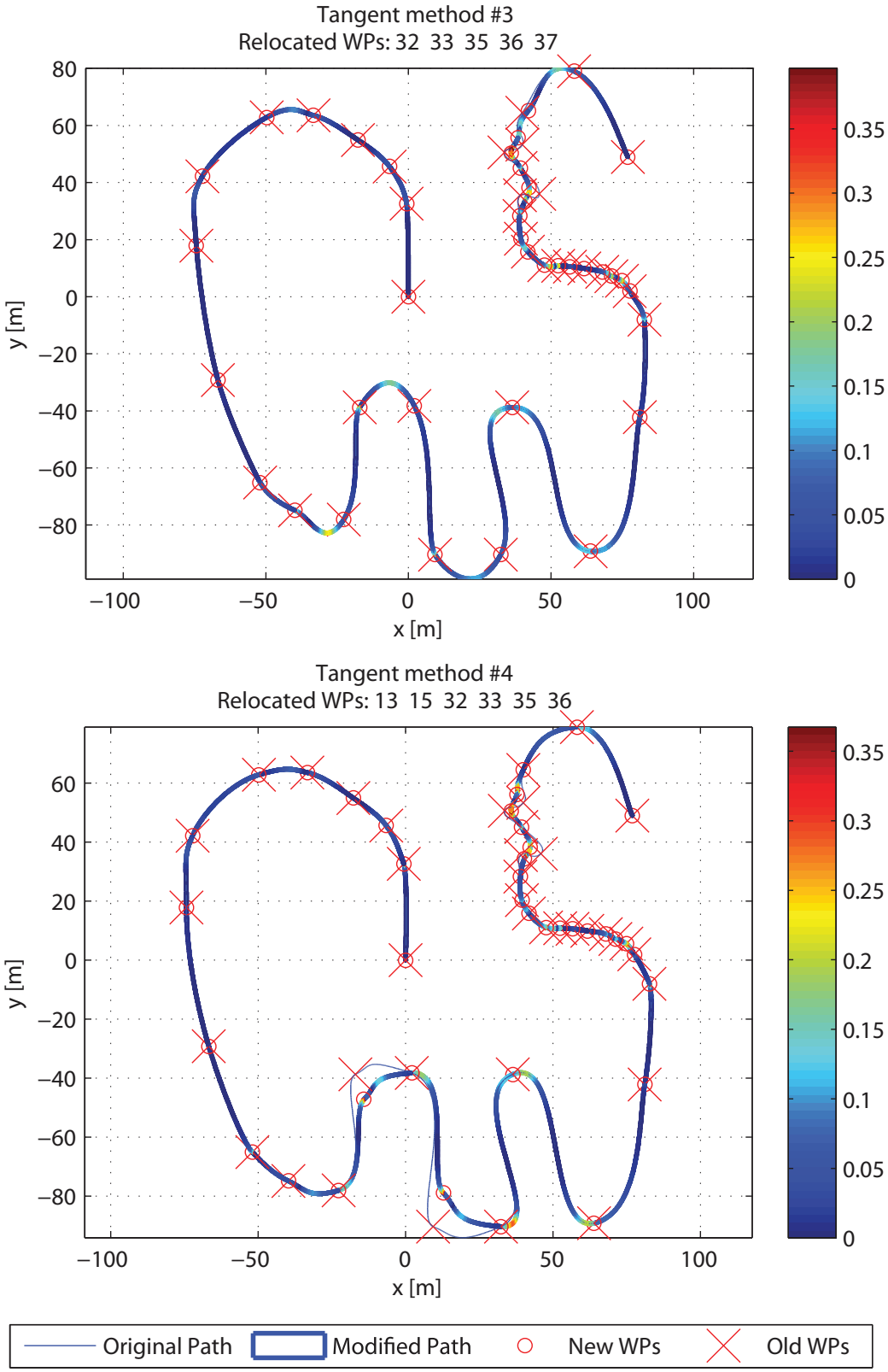

Figure 7.16: Paths with relocated WPs and tangent calculation method 3 and 4. $\kappa_{vehicle} = 0.4\ m^{-1}$

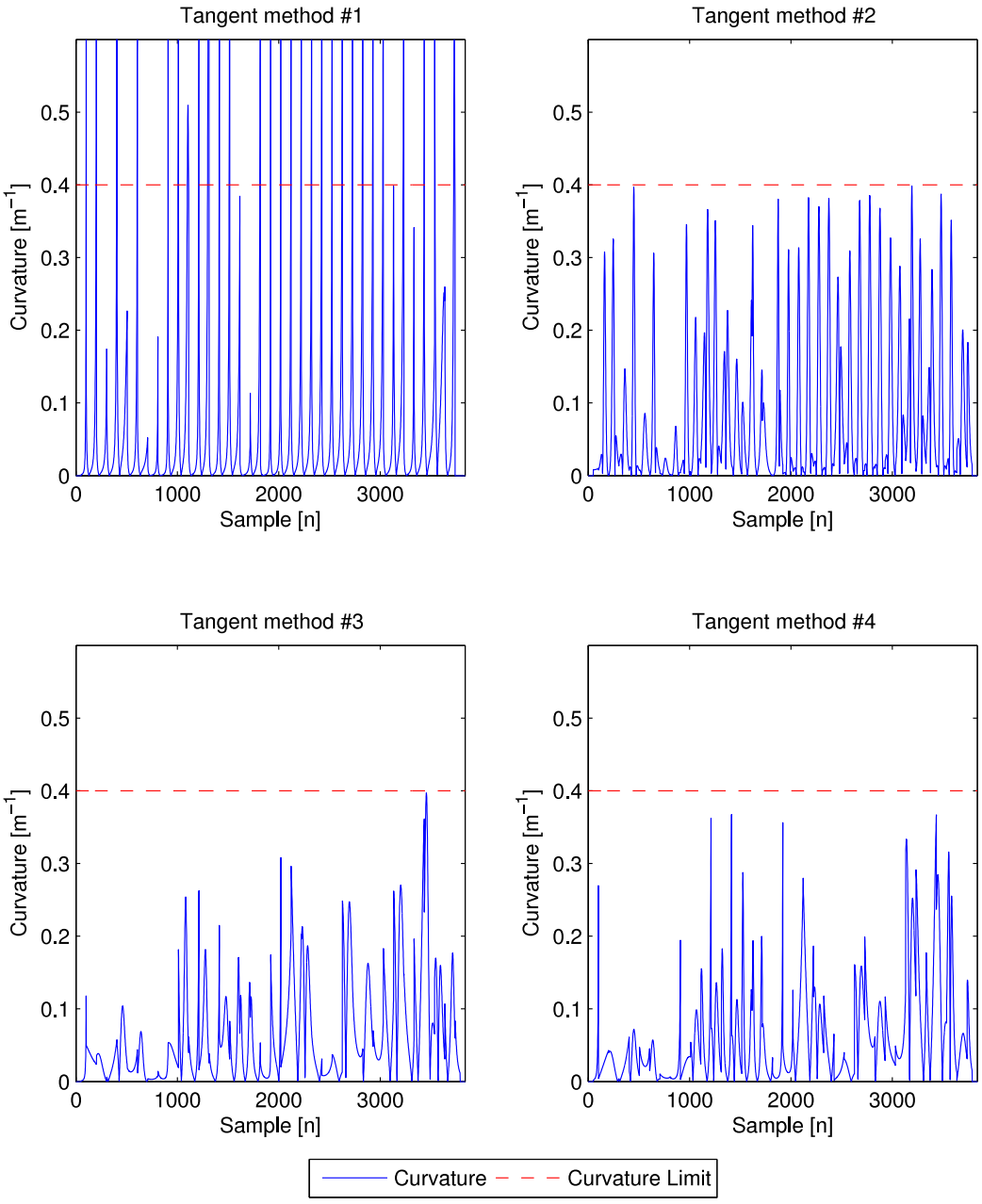

Figure 7.17: Path curvature values for Relocating WP algorithm for the four different vector tangent calculation methods for $\kappa_{vehicle} = 0.4\ m^{-1}$

7.4 Case Study

For every case study an identical WP setup have been employed. If not specified otherwise, the vehicle velocity is set to $v = 2.1\ m/s$ and the acceptance circle of the waypoints is $R = 20\ m$. The setup is created by the graphical WP generator and the particular WP placement is made on the basis of capturing several realistic cases, i.e. various distance between WPs, straight line following, and course-changing maneuvers; left and right turning, spanning from 0° to nearly 150° degrees. No weather disturbances are implemented unless it is specified. The vehicle starts at the origin of the map and travels the path according to the successive WPs. In total, there are 23 waypoints, and the current target WP switches to the consecutive WP once the vehicle enters the acceptance circle of the current target waypoint.

Path generation with vector tangent calculation method *3* is employed for the cases where paths are generated.

In principle, these calculations are supposed to be carried out offline. However, with sufficient computing power there is no reason for not calculating the paths online and optimize the trajectory as it goes.

7.4.1 Comparison of PP- and LOS-Based Guidance

The different guidance algorithms proposed in Chapter 3 are tested on a first order Nomoto model. Thus, the vehicle velocity is constant; $v_{nomoto} = 2.1\ m/s$. The parameter identification of the Nomoto model was found experimentally by Arild Hepsø, Svein Peder Berge and Vegard Evjen Hovstein *(Maritime Robotics)* during test-runs on Trondheimsfjorden in 2008. The time constant turned out to be $T_{nomoto} = 2.5\ s$ and the gain was set to $K_{nomoto} = 0.7328\ s^{-1}$. Both the lookahead- and enclosure-based guidance strategy use lookahead value and enclosure radius of 30 m in the following example.

The controller gains were set to: $P_k = 0.5$ and $P_i = 0.0001$ and the parameters of the heading reference model were chosen to: $\omega_n = 1$, $\zeta = 0.85$, $r_{max} = 15 \cdot \pi/180\ rad/s$ and $a_{max} = 5 \cdot \pi/180\ rad/s^2$.
When investigating Figure 7.18, it becomes obvious how the LOS algorithms and PP guidance differ. While Line-of-Sight guidance aims for a target LOS setpoint somewhere along the line between two WPs, pure pursuit guidance takes the next WP directly in sight. A consequence of this characteristic is that the path from a pure pursuit scheme often becomes shorter than for

enclosure- and lookahead-based guidance. This is due to the fact that PP guidance eludes to touch the straight-line that the LOS guidance techniques converges to. However, since PP encounters the target WP with a slightly different heading than for the LOS guidance techniques, it can be both advantageous and disadvantageous with respect to the path to the subsequent waypoint.

The calculation of pure pursuit guidance is less complex than both the LOS-based guidance schemes. The different techniques should be utilized regarding the proper application. When it comes to choosing between lookahead- and enclosure-based guidance which performed nearly identical in the previous example, lookahead-based- is both valid for all cross-track-errors and are less computational intensive than enclosure-based guidance. It is important to choose a suitable lookahead distance to avoid too aggressive steering while at the same time not selecting a large distance so that the LOS setpoint is far behind the target WP. Lookahead distance of 30 m gave good results with the controller parameters and models given.

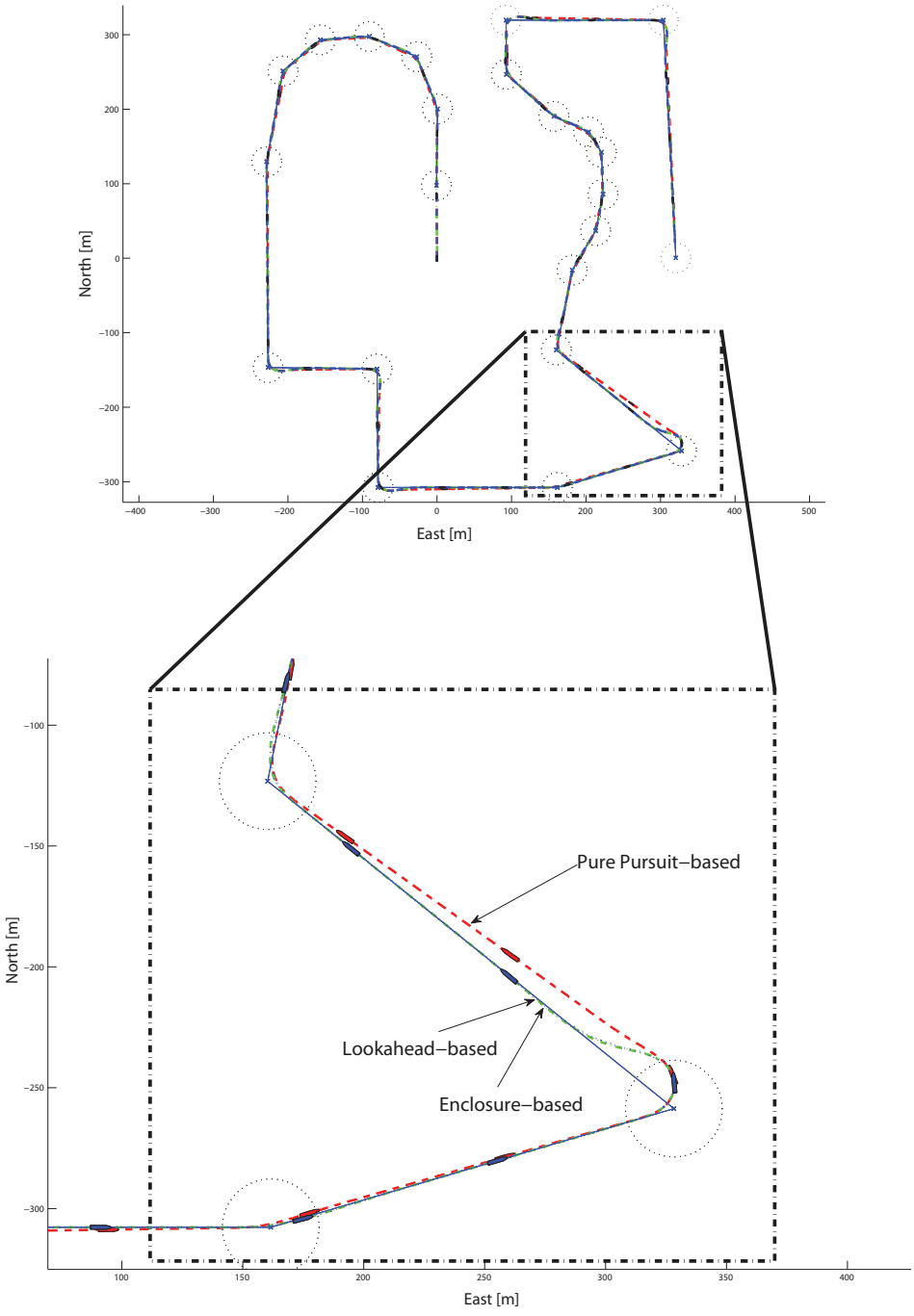

Figure 7.18: Comparison of pure pursuit-, lookahead- and enclosure-based guidance algorithms

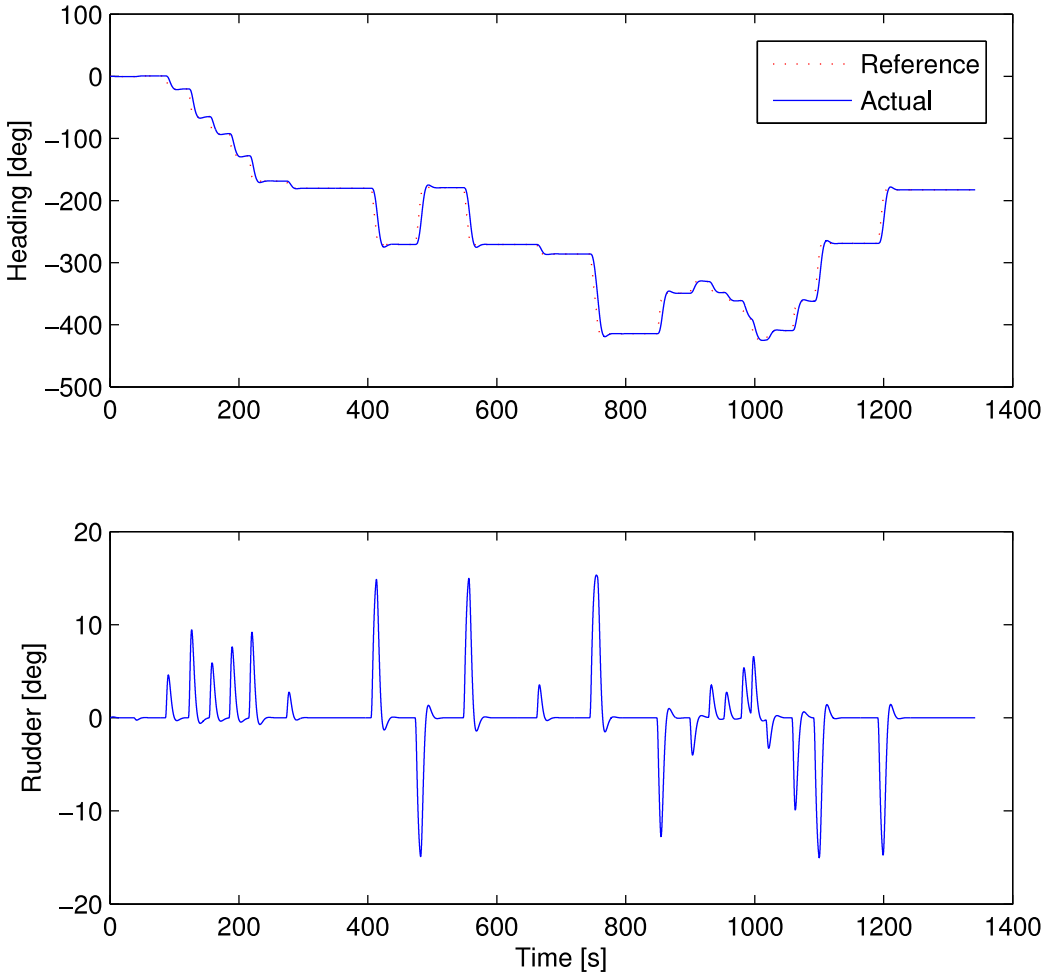

Figure 7.19: Heading, heading reference and rudder values for pure pursuit guidance

7.4.2 Validation of Nomoto Model

The parameters of the Nomoto model employed in the previous section were estimated by full-scale testing. The surge speed of the Nomoto model was assumed constant, so that the verification of the model is done with thrust equivalent to surge speed of 2.1 m/s. However, the validity of Nomoto models are limited to small rudder angles, therefore large rudder deflection will prove dissimilar response for the complex mathematical model of Viknes 830 [27] and the first order Nomoto model. Identical controller gains as for the comparison of pure pursuit and the LOS-guidance algorithms are used for the two models. The heading reference model is also identical to the previous case.

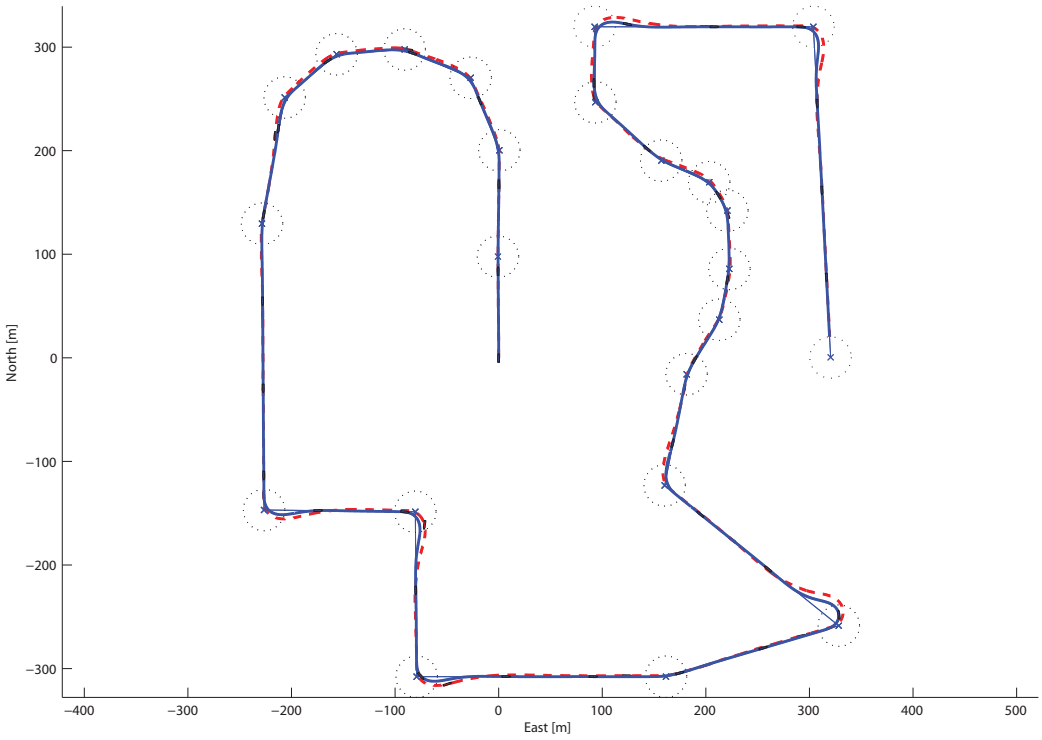

Figure 7.20: Path comparison of first order Nomoto model (blue/solid) and the Viknes 830 model (red/dashed)

It becomes apparent that the heading response of the two models differs. A faster response is achieved for the Nomoto model, thus the radius of the acceptance circles can be set smaller while still avoiding overshoot onto the

straight-line from the current WP to the next one, compared to the Viknes 830 model. There is a time constant $T_{rmm} = 0.5s$ in the rudder machinery model that causes a heading response delay. However, the distinction of the minimum turning radius ρ becomes the most noticeable factor when setting the rudder angle to a maximum of 27° (see Figure 7.21).

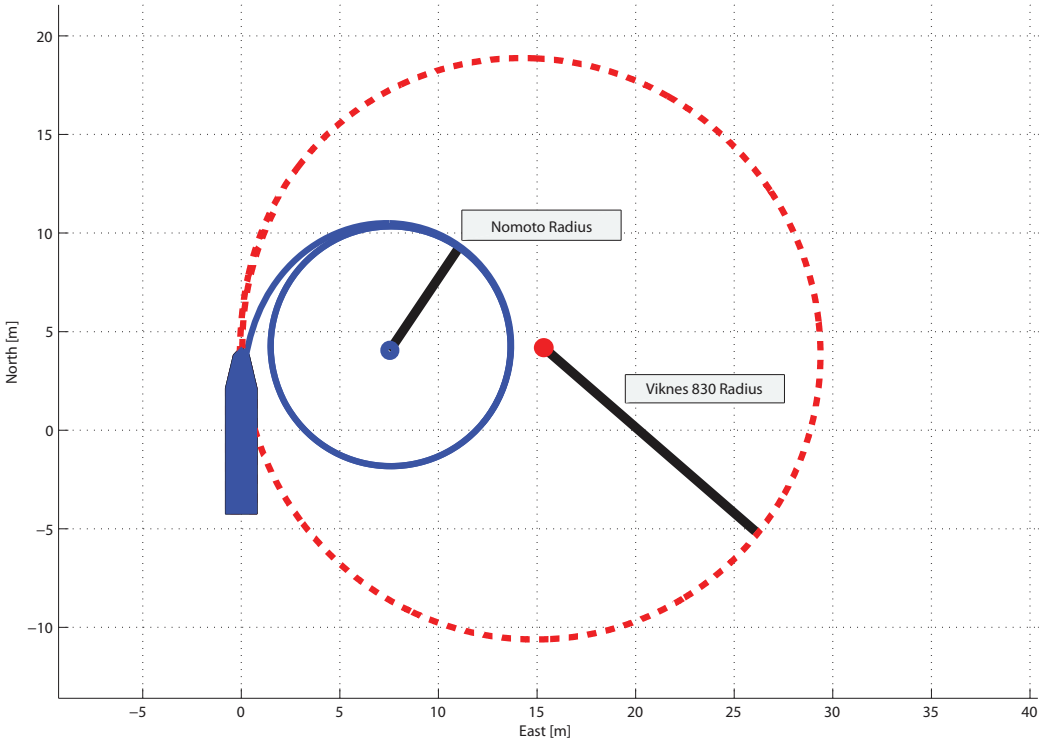

Figure 7.21: Comparing minimum turning radius for first order Nomoto- (blue/dashed) and Viknes 830 model (red/solid)

Examining Figure 7.21, the minimum radius for the Nomoto- and Viknes 830 model are found to be $\rho_{nomoto} \approx 6\ m$ and $\rho_{viknes} \approx 14.5\ m$. This is a significant difference, therefore it is not expected that the Viknes 830 model manage the same maneuvering as the Nomoto model for sharp turns. It is evident that the Nomoto model deviates from the mathematical model of Viknes 830 when it comes to large rudder angles, which were assumed in advance.

The first order Nomoto model is not adequate to make up for a complex model when varying velocity and large rudder angles applies. By utilizing gain scheduling, the Nomoto model could have been modified to take varying

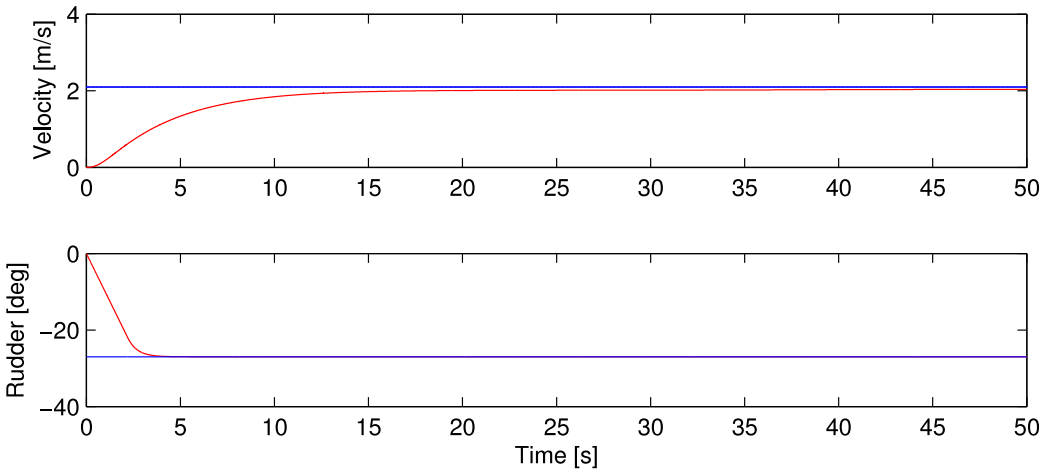

Figure 7.22: Velocity and rudder values for Nomoto- (blue) and Viknes 830 (red) model with maximum rudder deflection

velocities into consideration, although the maneuvering characteristic was significantly different for $v = 2.1\ m/s$. It is also possible that a second order Nomoto model would have given better results for higher rudder deflection as well [24]. Nevertheless, the first order Nomoto model is sufficient to simulate different guidance schemes with low computational power, hence resulting in fast simulation time while at the same time capturing the principles of the different guidance techniques quite well.

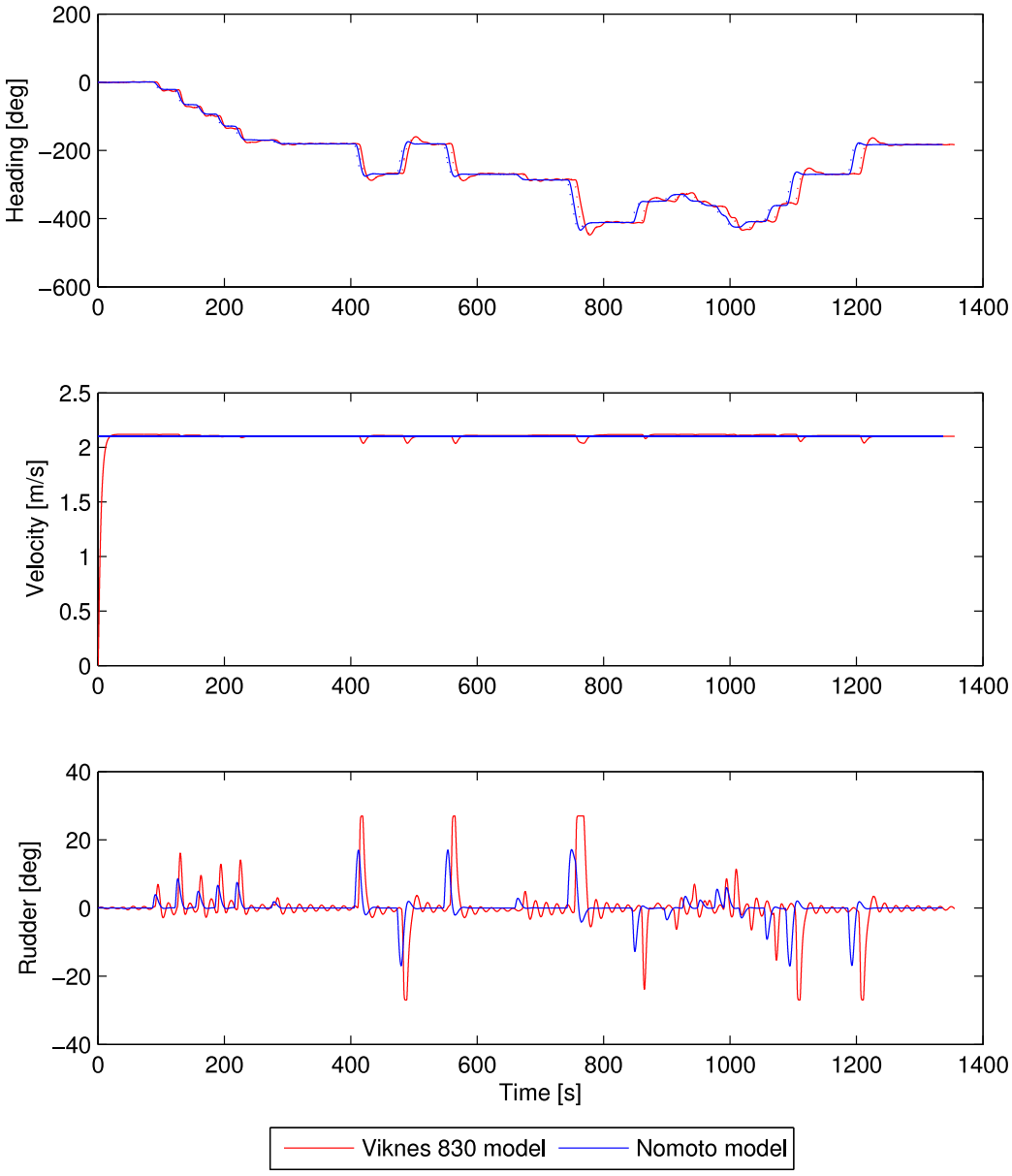

Figure 7.23: Heading, velocity and rudder data from the trajectories of the validation of Nomoto (blue) and Viknes 830 (red)

7.4.3 Adapted Acceptance Circle Radius

For every succeeding test case, the mathematical model of Viknes 830 is utilized. The control parameters, reference model parameters and vehicle velocity are identical to the former section when comparing the Nomoto- and Viknes 830 model, if not specified otherwise.

To avoid overshooting when making a sharp turn of e.g. 90°, a function (3.31) that takes the angle between the previous, current and next WP as an input and calculates a desired acceptance circle radius is employed. The parameters for the following case are: $R_{max} = 45\ m$, $R_{min} = 20\ m$ and $\sigma_R = 1.2$. The constant acceptance circle radius is set to $R_{constant} = 20\ m$.

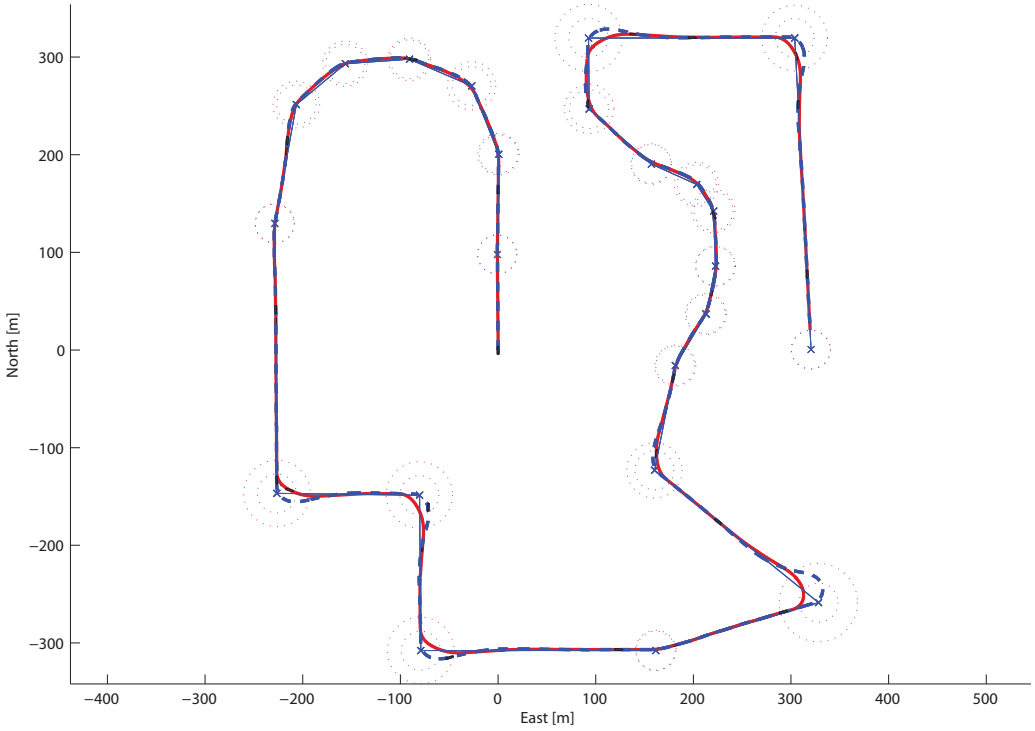

Figure 7.24: The vehicle trajectories for constant (blue/dashed) and varying (red/solid) acceptance circle radius

By inspection of Figure 7.24, one can clearly see the advantage of having adapted acceptance circle radius with respect to meet the straight-line to the next WP as early as possible. However, larger radius can cause greater deviation to the target WP since turning, and WP switching, occurs earlier than for smaller circles. The circle size should be adapted to the vehicle dynamics

and maneuvering ability. It is preferable to keep the deviation between the trajectory and the original WP low. Thus, smaller acceptance circle radius is chosen when course-changing maneuvers are minor. The radius should be defined to minimize the overshoot, analogous to a critical damped step response.

Adapting the circle of acceptance radius can be seen as a feed-forward effect, since it takes the future path layout into consideration.

7.4.4 The Effect of Different Vehicle Velocity

For the cases discussed so far, the vehicle velocity has been constant. When the velocity increases, it is expected that the turning radius is increased. Thus, a former feasible path can become infeasible due to the altered turning characteristic when increasing the vehicle velocity.

The velocities are set to $v_1 = 2.1\ m/s$ and $v_2 = 4.0\ m/s$ for the following test.

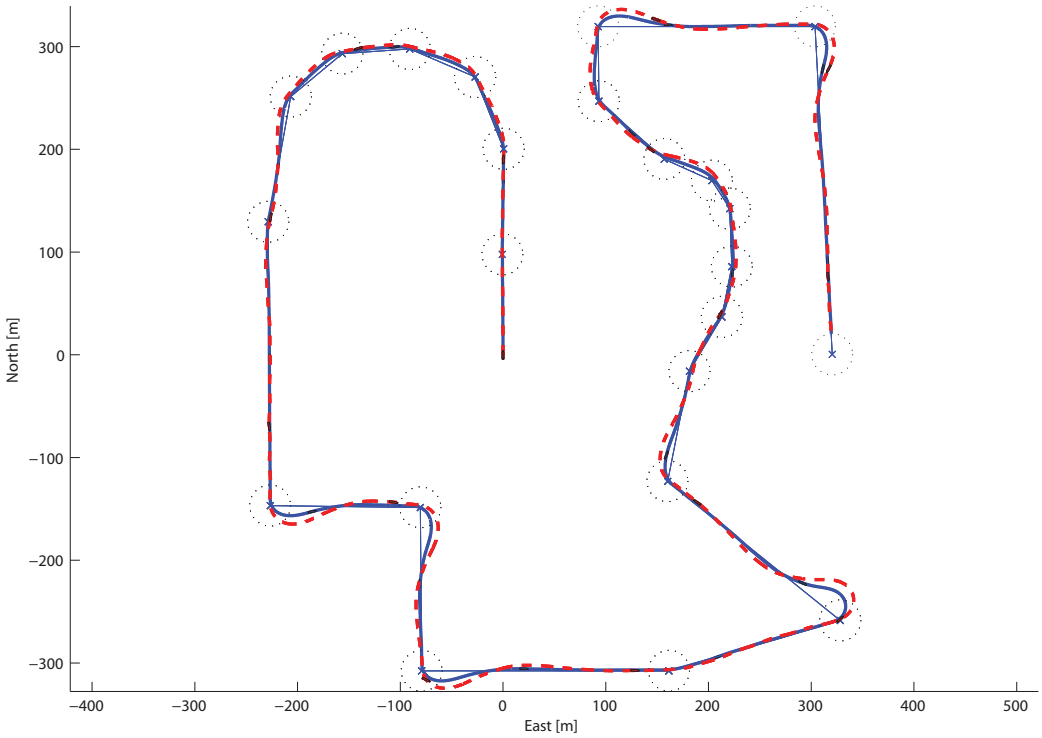

Figure 7.25: Trajectories for vehicle velocities of $v_1 = 2.1\ m/s$ (blue/sold) and $v_2 = 4.0\ m/s$ (red/dashed)

In Figure 7.25 it becomes clear how increased velocity results in wider turns. With nearly doubled velocity, the running time is naturally halved, but with the drawback of greater deviation between new trajectory and original path. However, straight-line following is approximately the same. This motivates decreasing the speed only in the waypoints. One way to cope with the overshoot is to increase the acceptance circle radius, but other actions can be made to enhance performance as will become apparent in the next section.

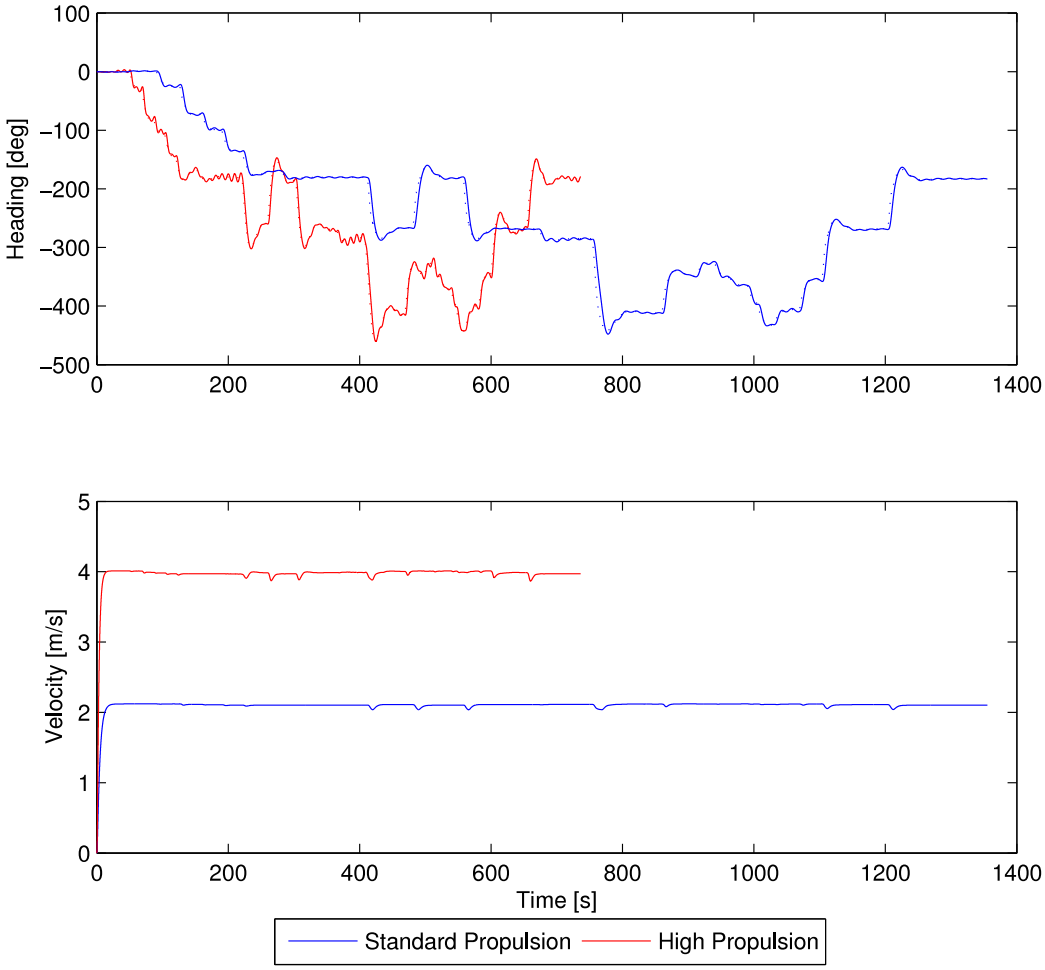

Figure 7.26: Heading and velocity values for the trajectories in Figure 7.25 with corresponding color code

7.4.5 Adapted Vehicle Velocity and Acceptance Circle Radius

It is now demonstrated how increased velocity shorten the running time while still resulting in a fairly good trajectory. Enhanced circle of acceptance radius proved it possible to avoid overshooting when entering a sharp turn. By introducing both these techniques, it will become evident that a good, feasible trajectory can be developed where the vehicle adapts velocity and WP radius to the waypoint layout of the path.

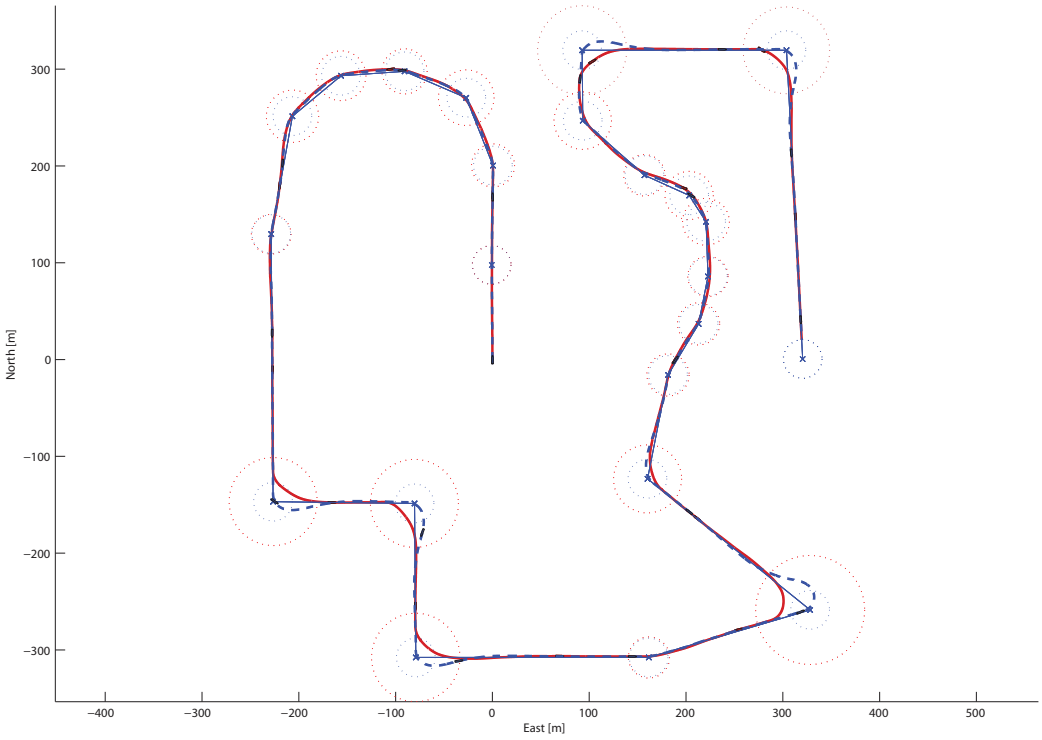

Figure 7.27: Trajectory with adapted velocity and circle of acceptance radius (red/soid) versus constant values (blue/dashed)

A PI speed controller is implemented with $K_{p,surge} = 40$ and $K_{i,surge} = 4$. The desired speed at the WPs is calculated according to (3.30), where $v_{max} = 3.1\ m/s$, $v_{min} = 1.0\ m/s$ and $\sigma_v = 1$. Outside the WPs (at the straight-lines), the nominal velocity was set to $v_{nominal} = 2.5\ m/s$. The acceptance circle radius is found from equation (3.31) with $R_{max} = 60\ m$, $R_{min} = 20\ m$ and $\sigma_R = 1.2$.

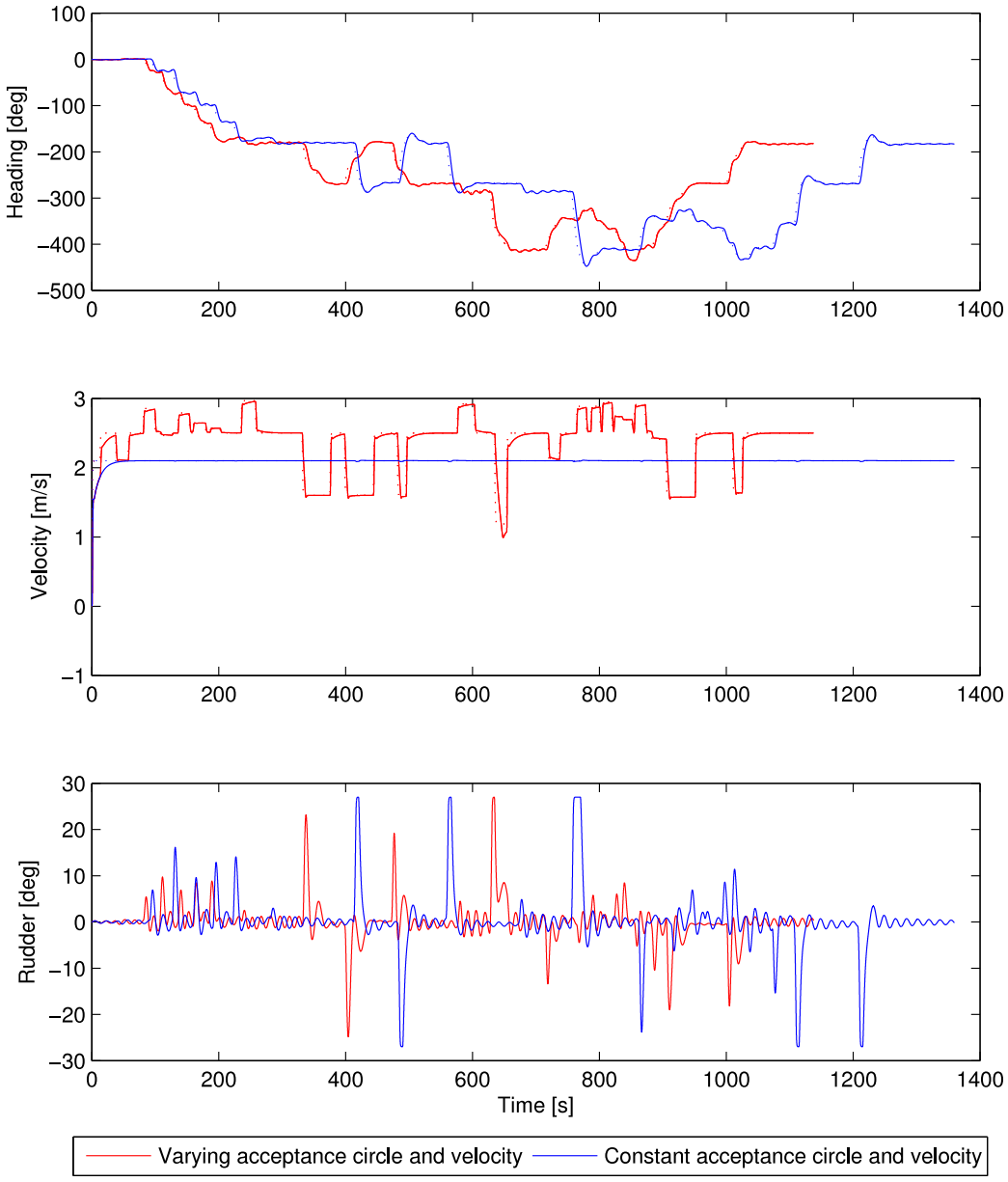

Figure 7.28: Heading, velocity and rudder angle for trajectories with adapted and constant velocity and acceptance circle

Setting both the WP radius and desired velocity in each waypoint is preferable to obtain a smooth trajectory with a good running time. The advantages can be seen in the straight-line following properties of Figure 7.27 and the running time (see Figure 7.28). Despite good performance in the current

test, this method does cause the path to deviate from the WP setpoints for sharp turns. To create trajectories that comprehend preturn and better WP intersection properties, a cubic spline can be derived in advance. This will become evident in the next section.

7.4.6 Path Generation with Cubic Splines

Using an interpolation technique, additional waypoints can be inserted between the original 23 waypoints, so that the vehicle will encounter the original target WP with a different heading. In other words, the spline is constructed in such way that it prepares the vehicle for the successive waypoint. This effect is called a preturn; where the vehicle prepares for the WP layout by encountering the target WP with less angle with respect to the consecutive waypoint.

The path generated in this example is calculated by tangent method *3*; equation (2.28). The guidance algorithm is lookahead-based LOS with lookahead distance $\Delta = 30\ m$ and the velocity is constant, $v = 2.1\ m/s$.

Figure 7.29 indicates low deviation between the generated path and the vehicle trajectory. The generated path is colored with respect to its curvature, where red means relative high curvature and blue indicates low curvature. Generally, the error between the generated path and the vehicle trajectory is small, but the error is, as expected, greatest where the curvature is high.

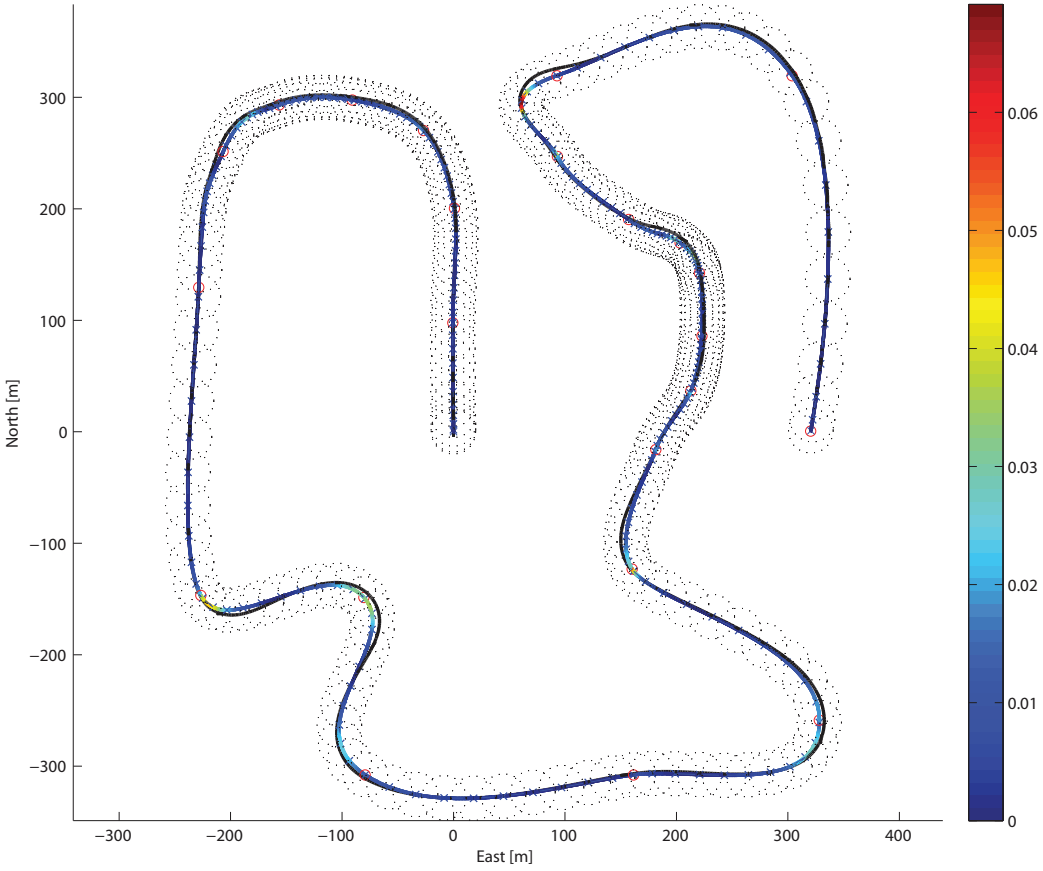

Figure 7.29: Path-following vehicle trajectory (black/solid) with constant velocity

7.4.7 Path-Following with Adapted Vehicle Velocity

Path-following can also be carried out with velocity adapted to the path layout; see equations (3.34). In this case, $v_{max} = 3\ m/s$ and $v_{min} = 0.5\ m/s$. The curvature is rounded to the nearest multiple of 0.01.

The same results occur now as for the adaption of velocity (and circle of acceptance radius) in section 7.4.5. Increasing the velocity where the curvature is low (corresponding to straight-lines section for the non-generated spline cases) and decreasing velocity where curvature is high (in turns), results in an efficient run with low deviation from the interpolated path. As opposed to the the methods without creating paths with the aid of splines, the trajectory now intersects every original WP within a small radius due to its preturn property and velocity adaption.

Despite good path-following properties, undesirable rudder oscillations arise (see Figure 7.31). There is a time delay from rudder input to course measurements; therefore oscillations may occur when applying too high controller proportional gain. Suggestion for solving this problem is to decrease controller gain or introduce a controller that handles the time delay better. This master's thesis does not focus on the controller, thus better path-following results may occur by re-tuning the PID-controller or introducing a different controller scheme. Feedforward in the sense of increasing radius of acceptance circle and decreasing velocity is not sufficient in this case.

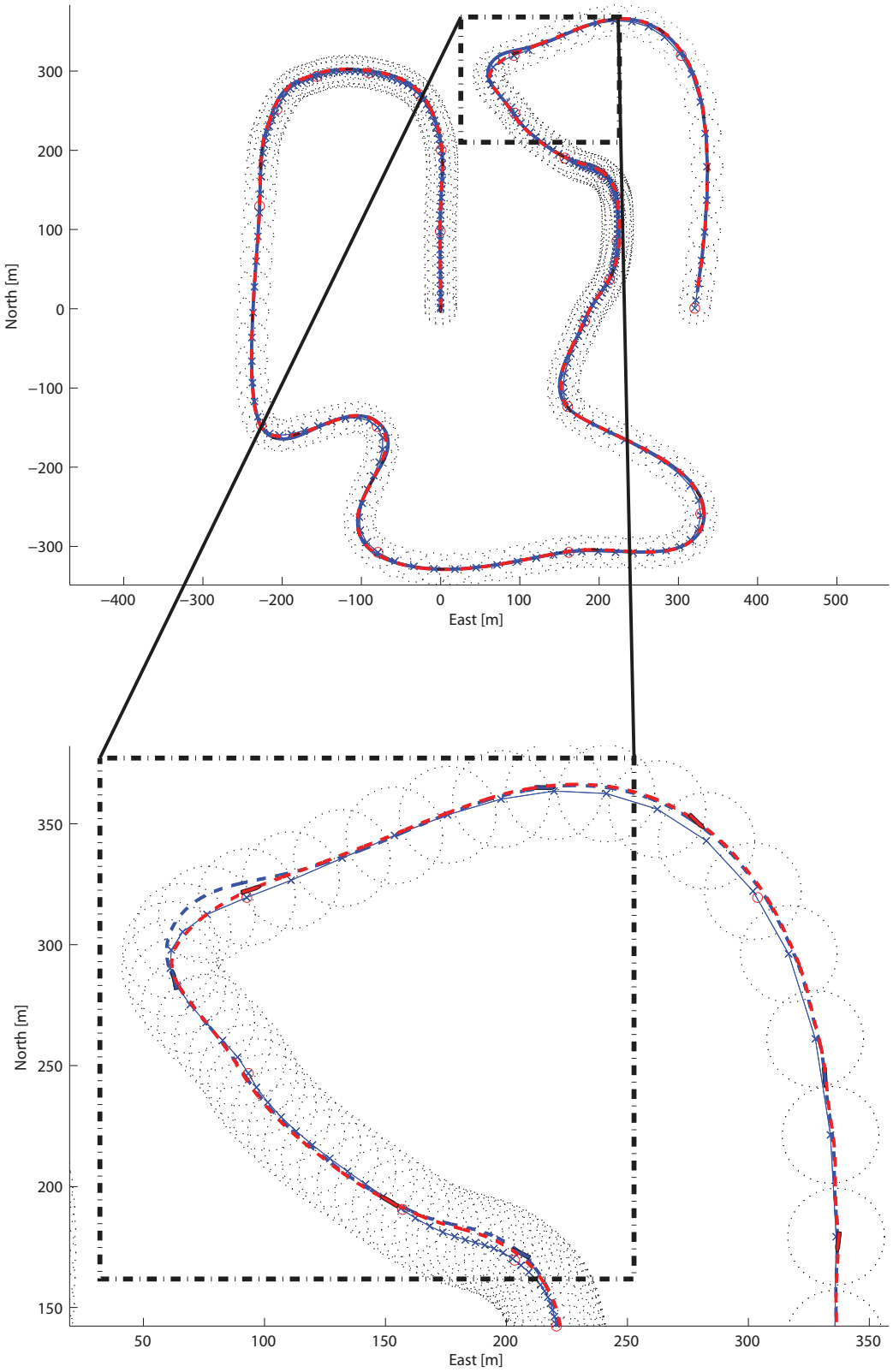

Figure 7.30: Path-following with constant velocity (blue) versus adapted velocity (red)

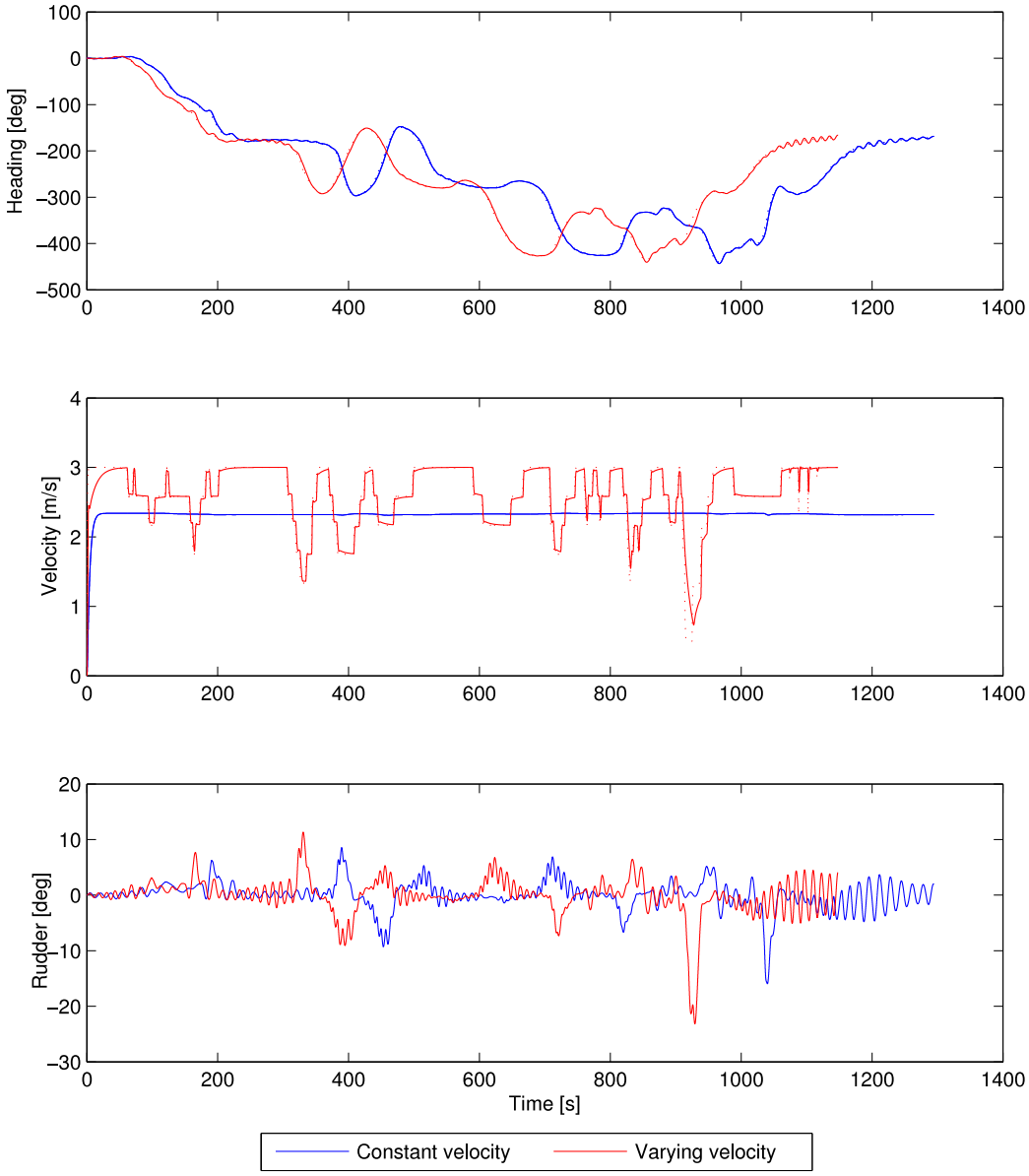

Figure 7.31: Heading, velocity and rudder angle values for path-following with constant and adapted velocity

7.4.8 Path-Following with Regenerated Path

The minimum turning radius ρ_{viknes} was calculated to 14.5 m for a vehicle velocity of $v = 2.1\ m/s$. Thus, the corresponding maximum curvature is: $\kappa_{vehicle} = \frac{1}{\rho_{viknes}} \approx 0.069\ m^{-1}$. Notice that the maximum path curvature κ_{path} barely exceeds this value. By regenerating the path to have maximum curvature beneath the threshold set by the vehicle, a consequence will be better path-following properties. Since the path curvature is close to the threshold already, it is expected that the original path-following properties are fairly good, which is also confirmed in the previous test case. However, modification of the path can prove slightly better path-following results.

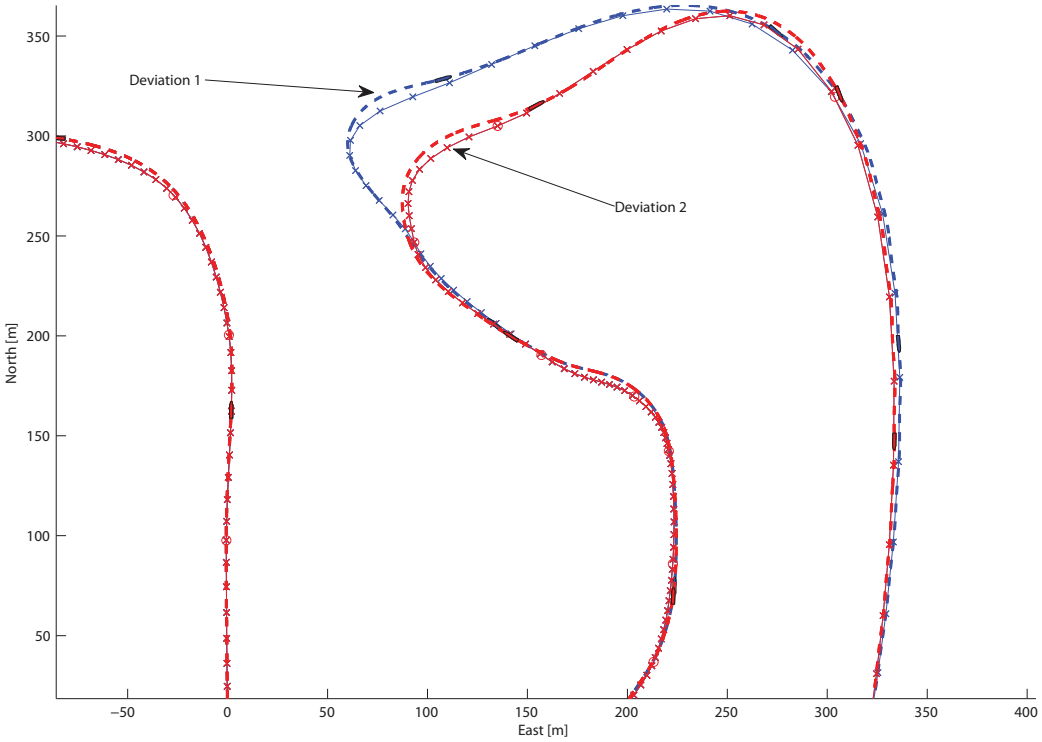

Figure 7.32: Distinction in deviation for original (blue) and modified path (red)

A modification of the path where the curvature was too high was carried out by the principle of moving WPs (see Figure 3.8). The relocation parameter was set to 2/5. The result yields some improvement with respect to deviation of calculated path, although the decrease in deviation was small. It is

worth noticing that with a path curvature so close to the curvature threshold of the vehicle, it might be more sensible to refrain from path modifications and cope with the minor error as can be seen as *Deviation 1* in Figure 7.32. Relocating a WP in this case does not improve the path-following property significantly (*Deviation 2*), thus the original path may be satisfactory.

Relocating WPs should only be carried out when the path is infeasible, since it alters the original WP setup; which has its layout for a reason. If, however, the path obtained from the WP placement is far from feasible with respect to the vehicle dynamics, action must be taken. Nevertheless, altering the path must be done with care, depending on the area of application. Before altering the path, minimum velocity in the critical areas should be tested, due to the change in the maneuvering characteristic at lower velocities. It follows that moving WPs uncritically before reducing the speed where the path curvature is too high is less than optimal action. The path curvature κ_{path} was close to $\kappa_{vehicle}$ for a vehicle velocity of $v = 2.1\ m/s$. As seen in 7.25 and well-known by empiricism, reducing the speed causes smaller turning radius. A good solution is therefore to exam the path for low vehicle velocity, and determine potential path altering based on the vehicle trajectories at low speed.

7.4.9 Path-Following with Weather Disturbances

When examining the path-following in more realistic conditions, simulations with wind-, water- and current disturbance were carried out. The environmental disturbance parameters were set to $v_{wind} = 2\ m/s$ with attack angle of 45° (directly north-east), $v_{current} = 0.05\ m/s$ at 90° and significant wave height was $0.8\ m$ at 45°.

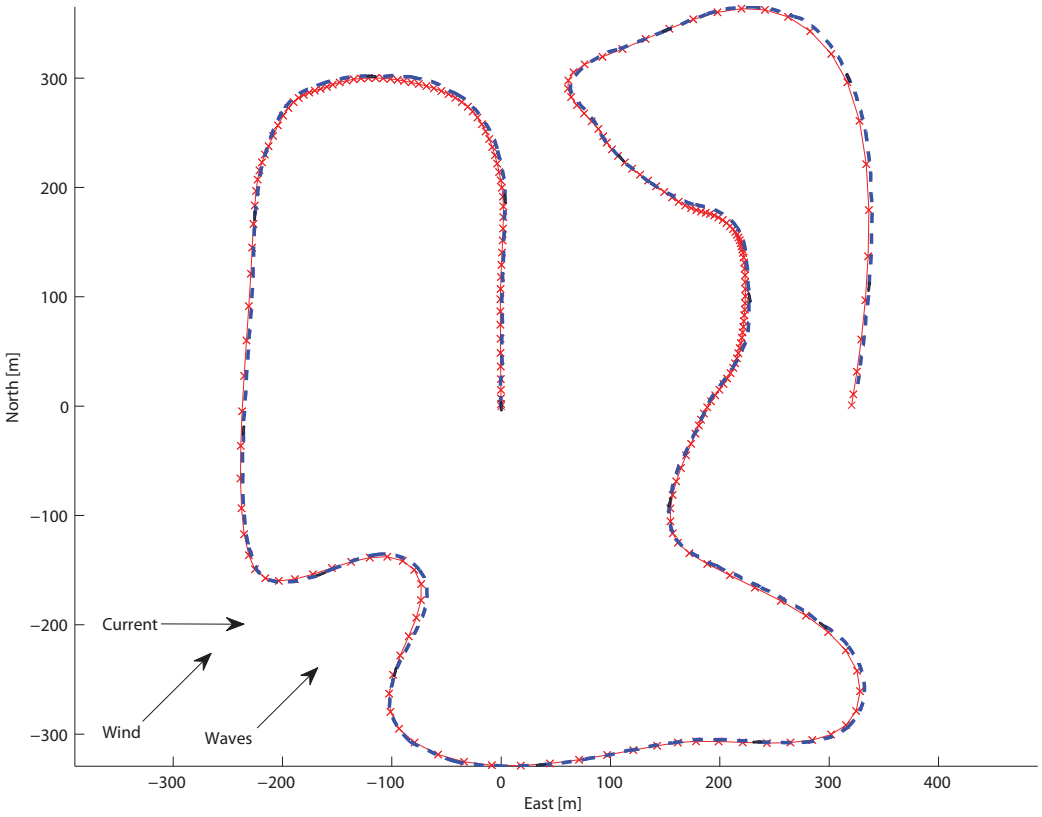

Figure 7.33: Vehicle trajectory including wind-, current- and wave disturbance

The vehicle proved good path-following properties in gentle breeze ($1.3 - 2.1\ m/s$ wind speed) with wave height according to the weather conditions. However, it is worth noticing that a small steady-state error arises owing to the wind and current disturbance. Tuning the controller, particularly increasing the integrator gain will cope with this problem and yields less deviation between vehicle trajectory and proposed path.

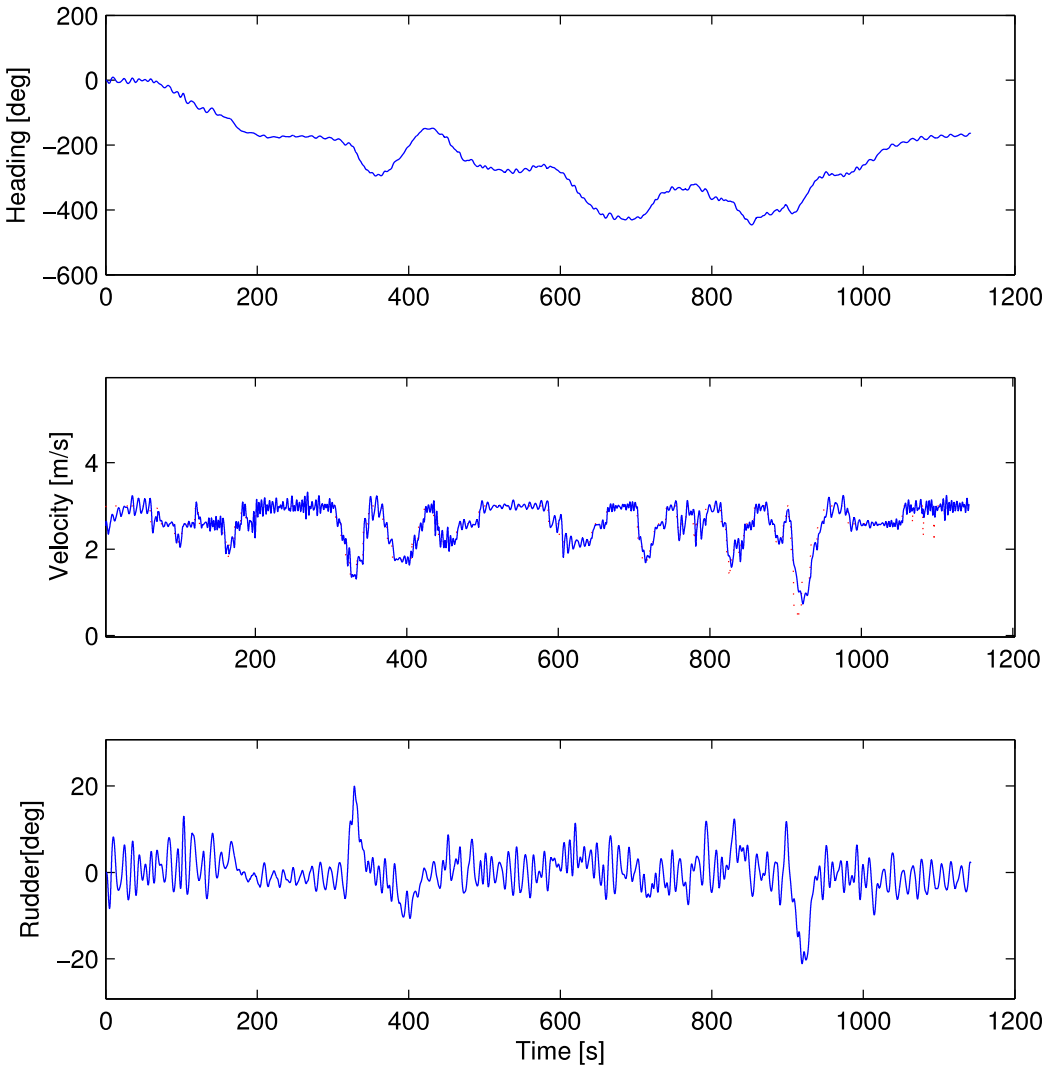

Figure 7.34: Heading, velocity and rudder angle for trajectory when including wind-, current- and wave disturbance

To avoid wear and tear, a wave filter should be implemented so that the rudder does not flap all along during the run. Nevertheless, disregarding this aspect, the vehicle proved good results even with varying weather disturbances.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

The main contribution to this thesis is algorithms for generating C^1-continuous paths based on a predefined waypoint setup and an appurtenant path feasibility check with respect to the dynamics of a vehicle.

It is demonstrated how four different tangent vector calculation methods can be used to create splines, with continuous first derivative through the interpolating points. In general, one tangent calculation method was superior the others, thus yield best results with respect to smooth curves for various waypoint layouts, compared to the other three methods.

By calculating the curvature along the entire path and comparing it with the vehicle dynamics, a path feasibility check was carried out. It is proven that speed- and acceptance circle adaption at each waypoint, or altering the original waypoint setup are suitable solutions to cope with the infeasibility problem for vehicles with trajectory curvature exceeding the path curvature limit. Since speed reduction enhances the turning property, it is preferable to decrease the velocity to a minimum at the critical path sections before possible modification of the waypoint setup have to be carried out.

Velocity- and acceptance circle radius functions based on the angle between the succeeding waypoints have been suggested to optimize driving with respect to path-following properties and time. A function for the vehicle velocity, taking path curvature into consideration is also proposed. Results of efficient path-following with small deviation between vehicle trajectory and path are apparent.

For paths failing the feasibility check, algorithms that remove, insert and relocate waypoints, thus decreasing the path curvature, have been suggested. These algorithms alter the original waypoint setup, and should not be used heedlessly.

Simulations have been carried out both on a simple first order Nomoto model and a more complex vehicle model of the Viknes 830. The limitation of the Nomoto model for large rudder deflection is demonstrated. Different guidance principles such as pure pursuit and line-of-sight guidance have been tested on the models and both methods function well for waypoint-following in any case.

8.2 Future Work

Although functional algorithms and good path-following properties were demonstrated in this master's thesis, there are several improvement areas that should be emphasized and devoted more time and research:

- Let the vector tangent calculation methods interact to create splines with reduced curvature, thus choosing the tangent method in each control point that yields minimized overall path curvature.
- Define the path generation problem as an optimization problem subject to minimized curvature.
- Improve the algorithm for adding waypoints to take the current waypoint placement and arrival course into consideration, thus defining whether crossing- or non-crossing added path should be generated. In addition, determine the number of added waypoints and their placement with respect to minimizing the curvature instead of inserting a predefined fixed waypoint setup.
- Resolve automatically if waypoint removal or -relocating yields best result with respect to curvature and original path restoration.
- Introduce filters to reduce wear and tear, especially for the rudder when waves are present.
- Improve controller scheme, e.g. with model predictive control.

In addition to the suggested improvements, full-scale testing on Trondheimsfjorden with *Maritime Robotics'* unmanned surface vehicle (*Viknes 830*) should be carried out to verify the case study simulations. Unfortunately, unanticipated problems with the power supply module in *Viknes 830* occurred which resulted in postponement of the full-scale testing until after submission date of this thesis.

Bibliography

[1] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons Ltd., 2011.

[2] S. Corfield and J. Young, "Unmanned surface vehicles-game changing technology for naval operations," *Advances in unmanned marine vehicles*, 2009.

[3] V. Bertram, "Unmanned surface vehicles–a survey," *Skibsteknisk Selskab, Copenhagen, Denmark*, 2008.

[4] J. Ebken, M. Bruch, J. Lum, and Space and naval warfare systems center San Diego CA., "Applying unmanned ground vehicle technologies to unmanned surface vehicles," in *Proc. SPIE*, vol. 5804, pp. 585–596, Citeseer, 2005.

[5] International Submarine Engineering Ltd., "Dolphin mk i semi-submersible auv." http://www.ise.bc.ca/Dolphin/ISE Semi-Submersible Dolphin.pdf (last visited 19th of June, 2011).

[6] M. Caccia, G. Bruzzone, and R. Bono, "Modelling and identification of the charlie2005 asc," in *Control and Automation, 2006. MED'06. 14th Mediterranean Conference on*, pp. 1–6, IEEE, 2006.

[7] J. Majohr and T. Buch, "Modelling, simulation and control of an autonomous surface marine vehicle for surveying applications measuring dolphin messin," *Advances in unmanned marine vehicles*, p. 329, 2006.

[8] A. Pascoal, P. Oliveira, C. Silvestre, L. Sebastião, M. Rufino, V. Barroso, J. Gomes, G. Ayela, P. Coince, M. Cardew, *et al.*, "Robotic ocean vehicles for marine science applications: the european asimov project," in *OCEANS 2000 MTS/IEEE Conference and Exhibition*, vol. 1, pp. 409–415, IEEE, 2000.

[9] M. Caccia, "Autonomous surface craft: prototypes and basic research issues," in *Control and Automation, 2006. MED'06. 14th Mediterranean Conference on*, pp. 1–6, IEEE, 2006.

[10] T. Fossen, M. Breivik, and R. Skjetne, "Line-of-sight path following of underactuated marine craft," in *Proceedings 2003 IFAC Conference on Maneuvering and Control of Marine Craft*.

[11] R. Skjetne, T. Fossen, and P. Kokotovic, "Output maneuvering for a class of nonlinear systems," in *Proc. 15th IFAC World Congress Automatic Control, Barcelona, Spain*, 2002.

[12] D. Nelson, D. Barber, T. McLain, and R. Beard, "Vector field path following for miniature air vehicles," *Robotics, IEEE Transactions on*, vol. 23, no. 3, pp. 519–529, 2007.

[13] W. Naeem, R. Sutton, S. Ahmad, and R. Burns, "A review of guidance laws applicable to unmanned underwater vehicles," *Journal of Navigation*, vol. 56, no. 01, pp. 15–29, 2003.

[14] T. Vaneck, "Fuzzy guidance controller for an autonomous boat," *Control Systems Magazine, IEEE*, vol. 17, no. 2, pp. 43–51, 1997.

[15] J. Osborne and R. Rysdyk, "Waypoint guidance for small UAVs in wind," *AIAA Infotech@ Aerospace*, pp. 1–12, 2005.

[16] J. Arvo, "Polynomial Interpolation and the Vandermonte Matrix," May 2009. `http://www.ics.uci.edu/~arvo/CS206/code/Vandermonde.pdf` (last visited 19th of June, 2011).

[17] C. Maes, "Runge's phenomenon from the wolfram demonstrations project." `http://demonstrations.wolfram.com/RungesPhenomenon/` (last visited 19th of June, 2011).

[18] D. H. House, "Spline curves," June 2010. `http://www.cs.clemson.edu/~dhouse/courses/405/notes/splines.pdf` (last visited 19th of June, 2011).

[19] G. Knott, *Interpolating cubic splines*. Birkhauser, 2000.

[20] Y. Wang, D. Shen, and E. Teoh, "Lane detection using catmull-rom spline," in *IEEE International Conference on Intelligent Vehicles*, pp. 51–57, 1998.

[21] M. Breivik, "Nonlinear maneuvering control of underactuated ships," *Master's thesis, NTNU, Trondheim, Norway*, 2003.

[22] C. Edwards and D. Penney, *Calculus*. Prentice Hall, 2002.

[23] T. M. Jensen, "Modeling Approaches for an Articulated Dumper Truck," 2010.

[24] J. Van Amerongen, "Adaptive steering of ships A model-reference approach to improved manoeuvring and economical course keeping," 1982.

[25] K. Nomoto, T. Taguchi, K. Honda, and S. Hirano, "On the steering qualities of ships," *International Shipbuilding Progress*, vol. 4, no. 3, pp. 354–370, 1957.

[26] C. Tzeng and J. Chen, "Fundamental properties of linear ship steering dynamic models," *Journal of Marine Science and Technology*, vol. 7, no. 2, pp. 79–88, 1999.

[27] K. Kjerstad, *Weather-Optimal Positioning Control for Underactuated USVs*. Tapir Uttrykk, 2010.

[28] J. Van Amerongen, "Adaptive steering of ships–A model reference approach," *Automatica*, vol. 20, no. 1, pp. 3–14, 1984.

[29] C. Chen, *Linear system theory and design*. Oxford University Press, Inc., 1998.

[30] A. Healey and D. Marco, "Slow speed flight control of autonomous underwater vehicles: Experimental results with NPS AUV II," in *Proc. of the 2nd Internat. Offshore and Polar Engineering Conf*, pp. 523–532, 1992.

Appendix A

Continuity

Continuity is a property that describes the smoothness of a path, curve or surface. A function $f(x)$ is continuous at a point a provided that $\lim_{x \to a} f(x)$ exists and that the limit in this point is $f(a)$ [22]. A function $f(x)$ is C^n continuous if $\frac{d^n f}{dt^n}$ is continuous. There are different orders of continuity:

- Discontinuous path
- C^0: The path is joined - no discontinuities
- C^1: The first derivative of the function is continuous
- C^2: The first and second derivatives of the function is continuous
- C^n: The first to n'th derivatives of the function is continuous

The different levels of continuity can be visualized in Figure A.1:

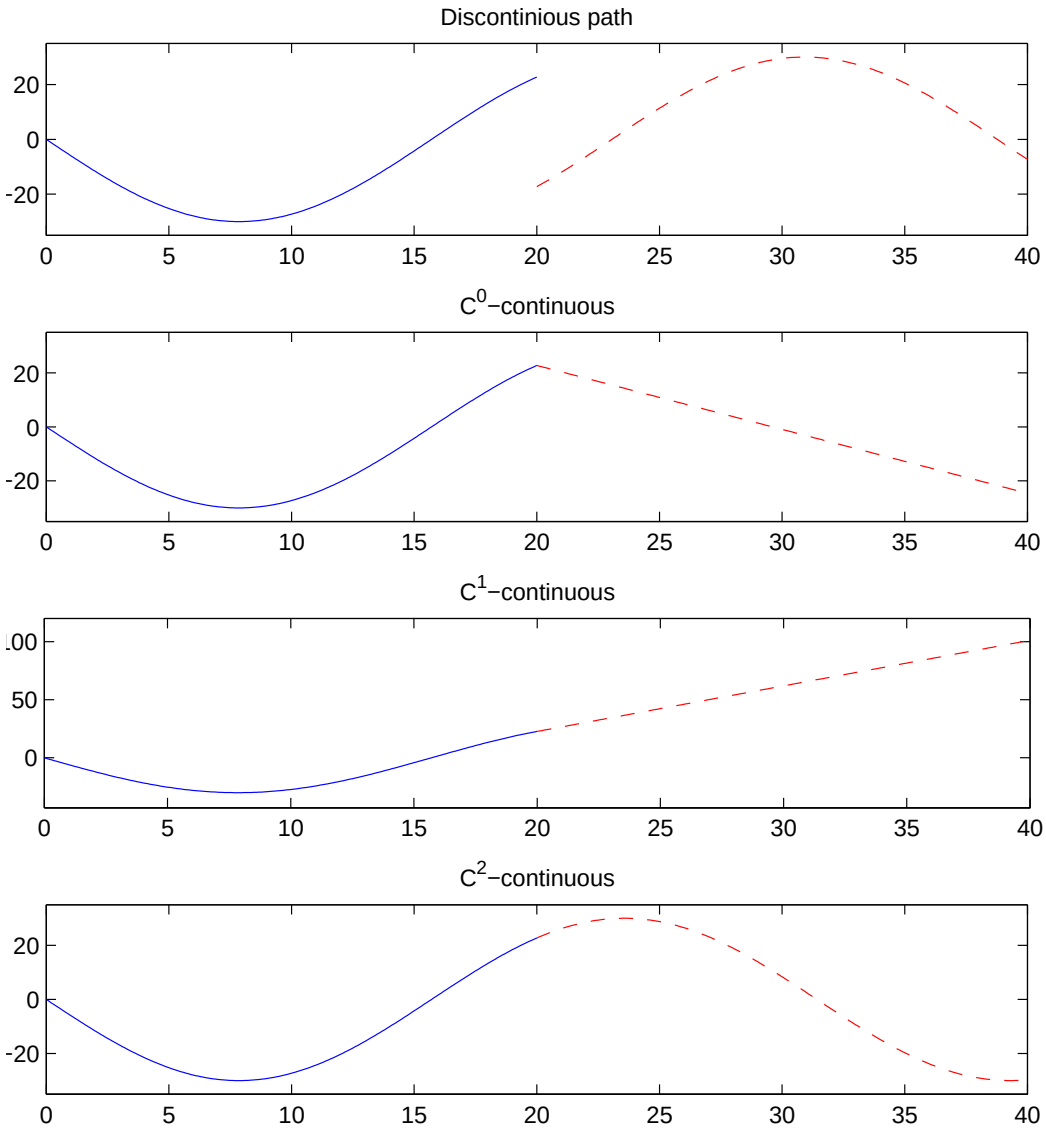

Figure A.1: Orders of Continuity

Appendix B

Digital Appendix

The digital appendix contains the Simulink models and the necessary MATLAB code to run simulations. Notice that the MSS Toolbox have to be downloaded and installed to run the simulations.

`INIT_NOMOTO.m` and `INIT_VIKNES.m` initialize the Simulink models *nomoto_model.mdl* and *VIknes_830_USV_model.mdl*. Controller-, reference model-, lookahead distance and other parameters are initialized in these m-files.

The *Misc* folder contains MATLAB code for several of the figures in the master's thesis. `boat_animation.m` creates an animation of the vehicle trajectory and is also located in the *Misc* folder.