Model Predictive Path Integral Approach for Trajectory Guidance

David Guerrero
Flight Test Engineer,
United States Air Force,
Robins AFB, GA

Dr. J.V.R. Prasad
Professor and Associate Director of VLRCOE,
Georgia Institute of Technology,
Atlanta, GA

Abstract

This paper presents the design and results of a computer model to evaluate the Model Predictive Path Integral Approach for vehicle trajectory generation under dynamic terminal constraints. The case study focuses on a flying vehicle final approach, descent, and landing phases onto a moving ship deck. The vehicle and ship are modeled with two degrees of freedom and under simple kinematics with constant commanded acceleration over a preselected time step. The vehicle motion is assumed to be point mass approximation with commanded acceleration along each axis with a first order dynamic model of vehicle acceleration. Ship motion is a priori and representative of a DDG-51 destroyer type ship with three possible sea states: low, medium, or high. The results show the model autonomously positions the vehicle in average within a one and a half feet three-dimensional box at the target landing location.

1 Introduction

Current work in autonomous vehicle trajectory generation reveals the computer intensive requirements for onboard computations and constant realtime knowledge of terminal constraints [1]. Alternative indirect methods for computing an optimal trajectory as the Model Predictive Path Integral (MPPI) Approach [2] intend to reduce onboard computing requirements and terminal constraint prediction computations. Some other differences between the MPPI and traditional methods as Differential Dynamic Programming (DDP) are described in [2].

The present work is a tool to evaluate the MPPI control algorithm. The computer model is created in Matlab and makes use of simple, first order vehicle dynamics with two dregrees of freedom (2 DoF), reference Appendix A for the complete Matlab code. The objective of the model is to assess the effectiveness of the MPPI method for the case of dynamic terminal constraints. The case study represents an autonomous flying vehicle during its approach and landing phases onto a moving ship deck.

The paper is organized into the overall model structure for the MPPI approach in section 2, details of ship motion model generation in section 3, flying vehicle model details in section 4, results of a total of 90 runs of the model in section 5, and conclusions are given in section 6.

2 Model Predictive Path Integral Approach

The model structure for the MPPI approach is described in this section. The vehicle and the ship each are modeled with 2 DoF and under simple kinematics with constant commanded acceleration over a preselected time step, Δt. The 2 DoF consist of rectilinear motion in the x-, and z-axis. The vehicle motion is assumed to be point mass approximation with commanded acceleration along each axis with a first order dynamic model of vehicle acceleration. For the purpose of creating a simple model, the dynamic model of the vehicle motion along the vertical axis is assumed to be of the form

$$\begin{bmatrix} \dot{z} \\ \dot{w} \\ \dot{a} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -\frac{1}{\tau} \end{bmatrix} \begin{bmatrix} z \\ w \\ a \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{a_c}{\tau} \end{bmatrix} \tag{1}$$

Initial vehicle commanded acceleration $a_{c_{initial}}(t)$ is calculated as an optimal solution (2) to a quadratic performance index (3) with the terminal constraint of matching the average position z_s and velocity w_s of the ship at a selected final time (4).

$$a_{c_{initial}}(t) = \min_{a_c}(J) \tag{2}$$

$$J = \int_{t0}^{tf} \frac{1}{2} a_c^2(t)dt \tag{3}$$

$$z_a(t_f) = z_{s\_average}(t_f), \quad w_a(t_f) = w_{s\_average}(t_f) \tag{4}$$

A set of multiple random trajectories is generated from the start of the vehicle approach to the selected landing time (t_f). A randomly generated differential acceleration command $da_{ci}(t)$ proportional to the optimal initial trajectory $a_{c_{initial}}(t)$ defines each subsequent random trajectory.

$$\pm da_{ci}(t) \rightarrow random, limited\ by\ vehicle\ physical\ characteristics$$

Therefore, the set of random trajectories are calculated from the initial optimal trajectory (5) and the set of random differential command accelerations (6).

$$a_c(t) = a_{c_{initial}}(t) \tag{5}$$

$$a_{ci}(t) = a_c(t) + da_{ci}(t) \tag{6}$$

The set of vehicle trajectories are used to update, at each iteration, the commanded acceleration $a_c(t)$ in (6) through the path integral approach. An optimal trajectory is computed from the set of random trajectories based on terminal constraints and vehicle path cost. The performance index described in (7) is used to perform an exponentially weighted average as shown in (8), thus obtaining an update for $a_c(t)$ described in (6). Theoretically, the process iterates between (6) to (8) for a converged update of $a_c(t)$, i.e. optimal $a_c(t)$ for values of ship position and velocity at the terminal time generated by the ship deck motion model.

$$J_i = \int_{t0}^{tf} \frac{1}{2} F\, dt + \left(z_a(t_f) - z_s(t_f)\right)^2 + \left(w_a(t_f) - w_s(t_f)\right)^2 \tag{7}$$

$$\mathrm{a}_{cj} = a_{c_{j-1}} + \frac{1}{M} \sum_{i=1}^{M} (d\, a_i) e^{-J_i}, \quad \text{M=Number of trajectories} \tag{8}$$

The function F (t) in (7) is a weight function used to trade between path cost and terminal cost. The ship final position and velocity in (7) are generated from the ship deck motion model described in section 3.

The command acceleration $a_c(t)$ resulting from (8) is used to integrate the vehicle dynamics (1) from current time t_o to $t_o+\Delta t$; where Δt is a preselected time increment, for example, 0.1 second. The new position and velocity of the vehicle at $t_o+\Delta t$ is used as the starting vehicle state and the process is repeated from (6) through (8) with the time horizon from $t_o+\Delta t$ to t_f.

The above process is repeated to obtain an optimized solution over the time horizon from $t_o+2\Delta t$ to t_f, and next from $t_o+3\Delta t$ to t_f, and so on. Each time the optimized control $a_c(t)$ is applied only over a time interval of Δt.

In summary, the MPPI model first generates a set of random trajectories based on the previous optimal trajectory, subsequently an exponentially weighted average based on the terminal constraint and path cost selects a new optimal trajectory, and finally the vehicle dynamics are used to generate the new vehicle state over a single time-step. The process repeats to move the vehicle through a model predictive optimal trajectory until reaching the landing time.

A flow diagram of the overall MPPI model structure is shown in Figure 1. The same model structure is used for generating the optimized trajectory in both x- and z-axes.

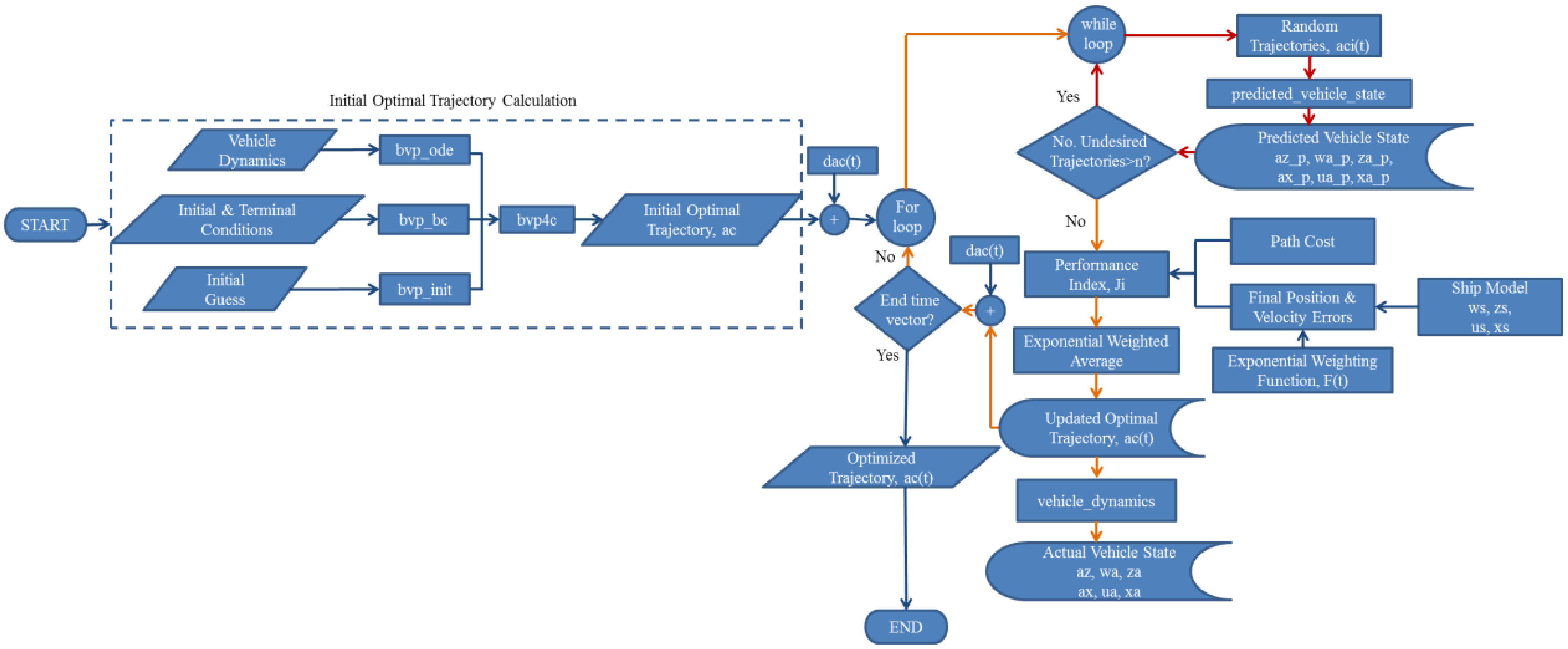
Initial Optimal Trajectory Calculation
START
Vehicle Dynamics
Initial & Terminal Conditions
Initial Guess
bvp_ode
bvp_bc
bvp_init
bvp4c
Initial Optimal Trajectory, ac
dac(t)
+
For loop
No
End time vector?
Yes
Optimized Trajectory, ac(t)
END
while loop
Random Trajectories, aci(t)
predicted_vehicle_state
Predicted Vehicle State az_p, wa_p, za_p, ax_p, ua_p, xa_p
Yes
No. Undesired Trajectories>n?
No
Path Cost
Performance Index, Ji
Final Position & Velocity Errors
Ship Model ws, zs, us, xs
Exponential Weighted Average
Exponential Weighting Function, F(t)
dac(t)
+
Updated Optimal Trajectory, ac(t)
vehicle_dynamics
Actual Vehicle State az, wa, za ax, ua, xa

Figure 1: MPPI Model Flow Diagram. The same model structure is used for both x- and z-axes.

3 Ship Deck Motion Model

Ship motion is representative of a destroyer type ship as the DDG-51. The US Navy Office of Naval Research (ONR), Naval Surface Warfare Center Carderock Division (NSWCCD) provided the statistical data required to generate the ship motion in the x- and z-axes. Ship motion in the z-axis (heave motion) is generated from random cyclic acceleration with mean and standard deviation of the selected sea state for the DDG-51 ship. The ship's forward velocity is selected from 4.8 knots (8.1 ft/sec), 19.8 knots (33.5 ft/sec), or 29.6 knots (49.98 ft/sec) depending on low, medium or high sea state, respectively. Surge motion is randomly generated with the standard deviation of the selected sea state. Table 1 shows statistical data for acceleration in the z-axis and velocity in the x-axis used to generate the ship motion model.

	Vertical Acceleration (ft/sec^2)			**Horizontal Velocity (ft/sec)**		
Sea State	**RMS**	**Mean**	**Max/Min**	**RMS**	**Mean**	**Max/Min**
Low	1.14	0.0	3.5/-4.2	0.18	8.1	8.8/7.5
Medium	2.43	0.0	8.3/-8.6	0.36	33.5	34.5/32.0
High	5.13	0.0	16.8/-19.2	0.73	50.0	51.7/46.1

Table 1: DDG-51 Combatant ship representative motion statistics, heave as dominant motion.

4 Vehicle Model

This section describes the details of the flying vehicle approach, descent, and landing. The model runs for an user-defined 20 seconds with 0.1 second time-steps. The vehicle approach starts 100 feet above the ship's datum plane at a pre-determined distance from the ship to execute a 3° glideslope. This version of the model includes motion in the vertical z-axis and horizontal x-axis. The vehicle y-axis motion is not included to simplify the model.

The vehicle dynamics are first order, point mass approximation in (1), described for the z-axis. Initial conditions of the vehicle state are arbitrary to provide a realistic approach. However, terminal conditions provide a successful landing at the end of the selected model time. Terminal conditions require matching positions and velocities for vehicle and ship.

Initial optimal solutions in each axis are obtained using the boundary value function in Matlab. The initial optimal solutions are calculated for a ship fixed to its x-y datum plane with no heave motion, and at a constant forward velocity with no surge motion. The model generates 100 random trajectories at each time-step starting with the previous optimal trajectory and adding a scaled, random, differential command acceleration. Each trajectory is assigned a performance index based on its final position and velocity with respect to the ship, i.e. its landing performance, and the path cost to generate that particular trajectory. The ship current vertical position is assumed to be its final position to calculate the z-axis performance index. The ship's final position used in the x-axis performance index is calculated based on the selected constant ship's forward velocity. The function F(t) used to trade between path cost and terminal cost is a logarithmically spaced vector created with logspace function in Matlab from 10^{-2} to $10^{0.5}$. This function exponentially favors terminal constraints rather than path cost as the vehicle approaches the ship deck. The user defines the number of random trajectories required to fall within a specified final position band with respect to the ship's final position. If this requirement is not met, a new set of random trajectories is generated by increasing the differential command acceleration in (6).

4.1 Vehicle Model Settings

The current vehicle model makes use of linear, point mass approximation vehicle dynamics. Once the user includes a different set of vehicle dynamics through the functions "predicted_vehicle_state" and "vehicle_dynamics" for both –x and –z axes, different settings in the code provide the model the required flexibility to achieve optimal results. The intended user-defined settings include:

1) Model total running time: **model_time**
2) Time step: **time_step**
3) Number of random trajectories, **tr**
4) Vehicle dynamics time constant: **tau**
5) Sea state: **low, medium, or high**
6) Initial differential command acceleration sensitivity: **r** and **rx**
7) Differential command acceleration sensitivity increment: **r_inc** and **rx_inc**
8) Maximum number of trajectories outside of acceptable final position band: **n**
9) Maximum number of cycles to generate minimum number of acceptable trajectories: **m**
10) Final position error acceptable range: **band_z** and **band_x**
11) Vehicle dynamics: **predicted_vehicle_state_z** and **_x**, **vehicle_dynamics_z**, and **_x**

Table 2 shows the settings used to obtain the results of section 5. It is recommended that every time the sea state is changed, or when new vehicle dynamics are used, the bottom settings in Table 2 are tuned for best landing performance results. Ship motion with higher standard deviation may require to increase the random initial differential command acceleration, r and rx, which provides the set of random trajectories more flexibility to deviate from the initial optimal trajectory. As the maximum number of undesirable trajectories, n, increases the code will run faster, but less trajectories could be used to calculate the vehicle command acceleration. The maximum number of cycles allowed to find the minimum required acceptable trajectories, m, is intended to allow the code to exit at a pre-determined condition even if the terminal constraints are not satisfied. Finally, it is important to define the best final position error range, band_z and band_x. A narrower range allows more precision on those trajectories used to calculate the commanded acceleration to the vehicle. However, it will eliminate a higher number of possible trajectories used in the calculation, and thus impacting the final optimal trajectory and its error in relation to ship's position and velocity.

As it is observed in Table 2, settings for the x-axis are more restrictive for all the sea states. Ship motion in the x-axis is more predictable with constant forward velocity and a lower standard deviation for surge motion. This allows to calculate the x-axis performance index with the predicted final ship position instead of using the current vertical ship position to calculate the z-axis performance index. Therefore the user may include less deviation, rx, from the initial optimal trajectory when generating the set of random trajectories for the x-axis.

	Sea State		
Setting	**Low**	**Medium**	**High**
model_time	20 seconds	20 seconds	20 seconds
time_step	0.1 seconds	0.1 seconds	0.1 seconds
random trajectories	100	100	100
tau	1	1	1
vehicle dynamics	Linear, point mass approximation	Linear, point mass approximation	Linear, point mass approximation
initial da$_c$ sensitivity	r=0.1, rx=0.05	r=0.1, rx=0.05	r=0.1, rx=0.02
da$_c$ sensitivity increment	r_inc=0.1, rx_inc=0.05	r_inc=0.1, rx_inc=0.05	r_inc=0.1, rx_inc=0.01
n	z-axis 40, x-axis 90	z-axis 40, x-axis 90	z-axis 40, x-axis 90
m	z-axis 10, x-axis 4	z-axis 10, x-axis 4	z-axis 10, x-axis 10
final position error range	band_z= ±5ft, band_x= ±5ft	band_z= ±5ft, band_x= ±5ft	band_z= ±4ft, band_x= ±3ft

Table 2: MPPI code settings for each sea state. Top settings are arbitrary to define the model. Bottom settings are tuned for best landing performance results.

5 Results

The MPPI model results for the initial conditions and terminal constraints described in sections 3 and 4 are discussed in this section. A simple statistical study was perfomed of 30 runs of the MPPI model at each sea state with the settings indicated in Table 2. The data for the 90 total runs is presented in Appendix B. The average values for landing performance indicators of the 30 runs at each sea state are shown in Table 3.

	Z-Axis			**X-Axis**			
Sea State	**Final Position Error (ft)**	**Final Velocity Error (ft/sec)**	**No. Acceptable Trajectories**	**Final Position Error (ft)**	**Final Velocity Error (ft/sec)**	**No. Acceptable Trajectories**	**Running Time (sec)**
Low	0.56	2.03	99.5	0.31	8.05	91.7	6.39
Medium	0.61	2.06	99.5	0.55	6.64	78.8	6.43
High	1.29	2.09	98.9	0.83	5.59	82.1	6.95

Table 3: Average values for a sample of 30 runs of the MPPI model for each sea state.

Vehicle landing performance under low and medium sea states results in the vehicle's final position within 1 foot from the target location on the ship deck in both the z- and x-axis. The final velocities of the vehicle and ship differ to a greater extent due to the fact that only a final position band is implemented in the code. Future versions of the code can add a velocity band of acceptable trajectories in an attempt to improve matching the final velocities.

The vehicle landing performance under high sea state results in the vehicle's final position within 1.3 feet from the target location on the ship deck on the z-axis, and less than a foot on the x-axis. The final velocity of the vehicle can again be improved to better match the ship's final velocity in both axes.

The average total number of acceptable trajectories used in the generation of the optimal trajectory is included in Table 3 to provide insight into the appropriateness of the selected settings. A low number of acceptable trajectories in relation to the user-defined random trajectories indicates the code settings could be modified to improve the model.

The running time of each code run is included for information purposes only and does not include graph generation time.

A graphical representation of the MPPI model results for one run follows. The results shown in Figures 2-5 correspond to the vehicle landing on a ship under medium sea state conditions. The approaches for a vehicle landing on a ship under low and high sea states are presented in Appendix C.

The vehicle command acceleration illustrated in Figure 2 represents the optimal trajectory generated by the MPPI approach. This resulting optimal trajectory is compared with the initial optimal trajectory calculated with the boundary value problem function with the ship under static terminal constraints, i.e. with no heave or surge motion.

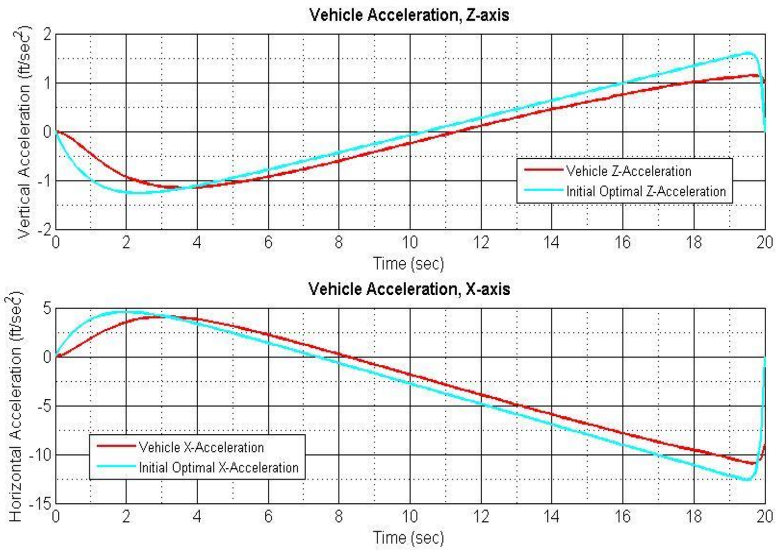

Figure 2: (Top) Vehicle command acceleration in the z-axis (optimal trajectory) versus initial optimal acceleration solution; (bottom) vehicle command acceleration in the x-axis versus initial optimal acceleration solution.

The command acceleration shown in Figure 2 is integrated into the vehicle dynamics in (1) to calculate vehicle velocity and position along the optimal trajectory. Vehicle velocity is illustrated in Figure 3. The approach starts at a constant altitude with no velocity in the z-axis, and at the forward velocity required to reach to ship in the selected time, 95.4 ft/sec.

As shown in Table 3, final velocities for vehicle and ship differ to a greater extent than their final positions. The reason is that the code recalculates the whole set of random trajectories if a minimum number of trajectories do not fall within an acceptable final position error band. This process is not currently applied to match final vehicle and ship velocities.

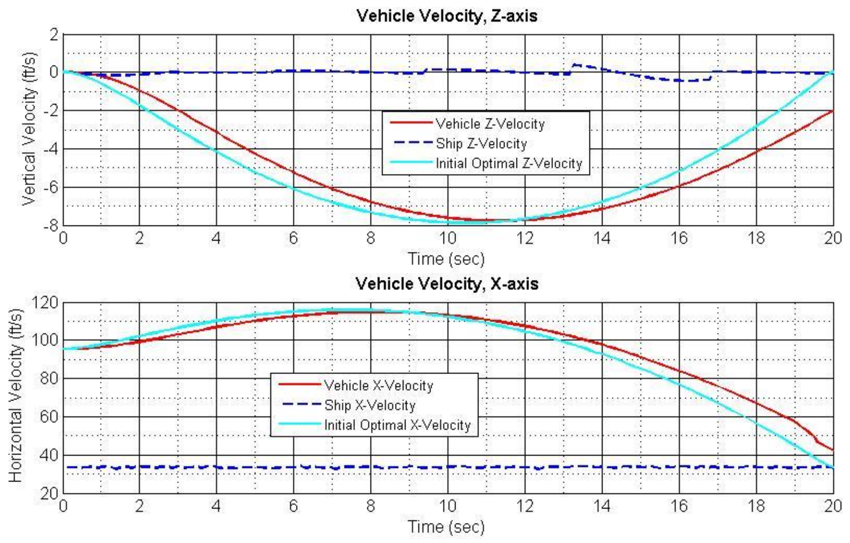

Figure 3: (Top) Vehicle velocity in the z-axis versus initial optimal solution and ship vertical velocity; (bottom) vehicle velocity in the x-axis versus initial optimal solution and ship horizontal velocity.

Figure 4 shows the position of the vehicle as it tracks the movement of the ship. The ship’s heave motion is almost indiscernable in top graph of Figure 4 due to the graph’s scale. A zoomed in view at the last second in the vehicle’s approach is shown in more detail in Figure 5. The ship’s forward velocity for the medium sea state shown in the graph in Figure 4 is 19.8 knots (33.5 ft/sec).

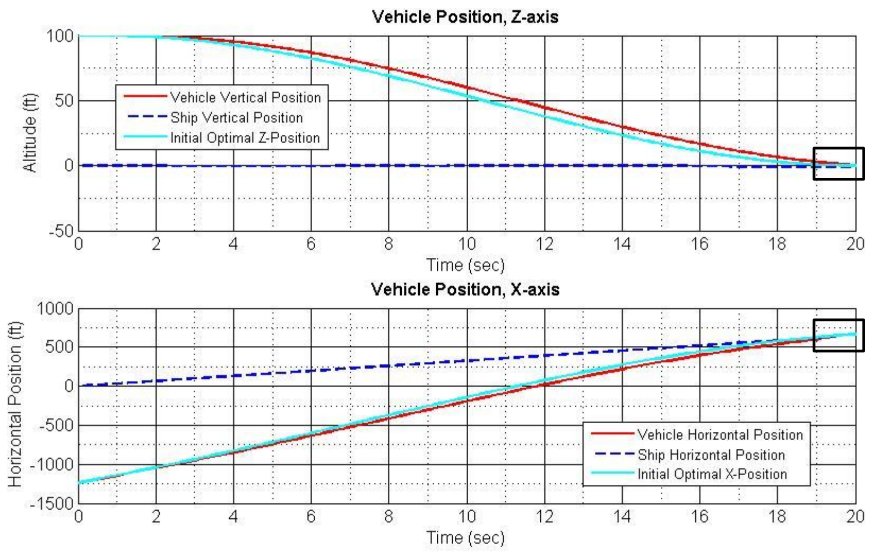

Figure 5: (Top) Vehicle position in the z-axis versus initial optimal solution and ship vertical position; (bottom) vehicle position in the x-axis versus initial optimal solution and ship horizontal position.

The top graph in Figure 5 shows the vehicle about one foot above the ship deck at the selected landing time. The ship itself is positioned about 0.7 feet below its initial datum plane due to its heave motion. The bottom graph in Figure 5 shows the final position in the x-axis with the vehicle about 3.0 feet ahead of the ship deck's target landing location.

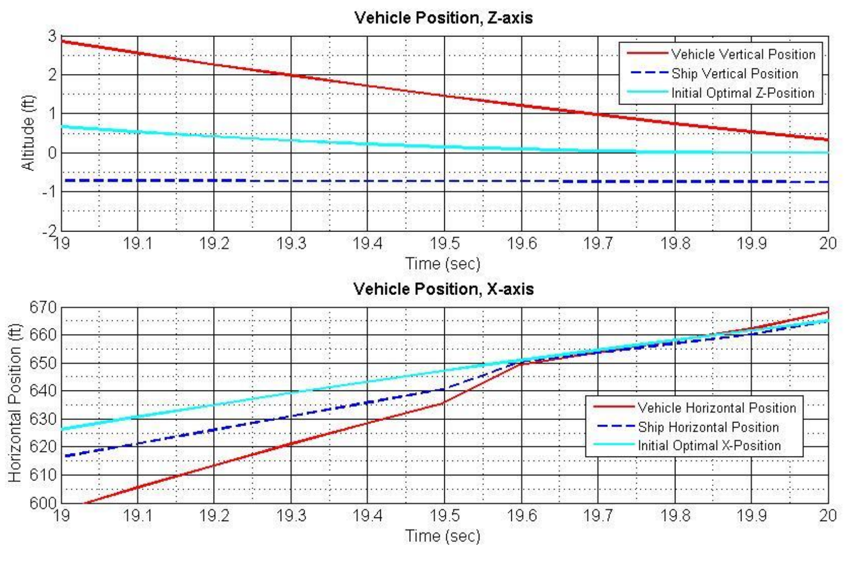

Figure 4: (Top) Last second in vehicle's landing showing vehicle position in the z-axis versus initial optimal solution and ship vertical position; (bottom) last second in vehicle's landing showing vehicle position in the x-axis versus initial optimal solution and ship horizontal solution.

The model for the MPPI approach achieves to optimize a trajectory for dynamic terminal constraints and autonomously positions the vehicle in average within a one and a half feet three-dimensional box at the target landing location.

6 Conclusions

A computer model in Matlab is presented for the MPPI approach for vehicle trajectory generation under dynamic terminal constraints. The case study focuses on a flying vehicle approach, descent, and landing onto a moving ship deck. The results show the MPPI model autonomously positions the vehicle in average within a one a half feet three-dimensional box at the target landing location. Future work includes adding the actual dynamics of a desired flying vehicle, including motion in the y-axis, and improving vehicle landing performance by reducing the position and velocity errors at landing.

Acknowledgments

We would like to thank the US Navy Office of Naval Research (ONR), Naval Surface Warfare Center Carderock Division (NSWCCD), and Mr. Alan Schwartz for providing the DDG-51 ship representative motion data.

References

[1] J. Horn, C. He, J. Tritschler, Et. Al. Autonomous Ship Approach and Landing using Dynamic Inversion Control with Deck Motion Prediction. *41st European Rotorcraft Forum*, 2015.

[2] G. Williams, A. Aldrich, and E. Theodorou. Model Predictive Path Integral Control using Covariance Variable Importance Sampling. *ArXiv*, 2015.

Appendix A – Matlab Code

```
%%Model Predictive Path Integral Approach for Trajectory Guidance
%%This model generates an optimal trajectory solution of an autonomous
%%aircraft landing on a ship deck. The model consists of the flying vehicle
%%and the ship each modeled with two degrees of freedom (2 DoF) and under
%%simple kinematics with constant commanded acceleration over a preselected
%%time step, delta_t. The 2 DoF of the vehicle consists of rectilinear
%%motion in the x- and z-axes. The vehicle motion is assumed to be point
%%mass approximation with commanded acceleration along each axis with a first
%%order dynamics model of vehicle acceleration.
tic
clc
clear all
close all

%%MODEL PARAMETERS SELECTION
model_time=20;  %Enter time for vehicle to land in seconds (total simulation time)
time_step=0.1;  %Enter time step in seconds
tr=100;         %Enter number of random trajectories
tau=1;          %Time constant of the first order dynamics for vehicle acceleration
                %Tau appears in bvp_ode function as well, make sure to change there
band_z=5;       %final z-position error range, ft
band_x=5;       %final x-position error range, ft
r_inc=0.1;      %dacz sensitivity increment, percentage of optimal trajectory
rx_inc=0.05;    %dacx sensitivity increment, percentage of optimal trajectory
%us=8.14;        %Ship's forward velocity (ft/sec), LOW level sea state *Update bvp_bcx
function*
us=33.49;        %Ship's forward velocity (ft/sec), MEDIUM level sea state *Update
bvp_bcx function*
%us=49.98;        %Ship's forward velocity (ft/sec), HIGH level sea state *Update bvp_bcx
function*

%%SEA STATE SELECTION
%sd=1.14;   %Standard deviation for ship's z-acceleration (ft/sec^2), LOW level sea state
sd=2.43; %Standard deviation for ship's z-acceleration (ft/sec^2), MEDIUM level sea state
%sd=5.13;  %Standard deviation for ship's z-acceleration (ft/sec^2), HIGH level sea state
%sdx=0.181;   %Standard deviation for ship's x-velocity (ft/sec^2), LOW level sea state
sdx=0.363;  %Standard deviation for ship's x-velocity (ft/sec^2), MEDIUM level sea state
%sdx=0.726;   %Standard deviation for ship's x-velocity (ft/sec^2), HIGH level sea state

%BVP Solution legend
%z(1)=z(t)  Vehicle vertical position
%z(2)=w(t)  Vehicle vertical velocity
%z(3)=az(t) Vehicle vertical acceleration
%z(4)=p1    costate 1
%z(5)=p2    costate 2
%z(6)=p3    costate 3

%x(1)=x(t)  Vehicle forward position
%x(2)=u(t)  Vehicle forward velocity
%x(3)=ax(t) Vehicle forward acceleration
%x(4)=p1    costate 1
%x(5)=p2    costate 2
%x(6)=p3    costate 3

%%OPTIMAL SOLUTION, SHIP AT CONSTANT X-VELOCITY AND Z=0
%Solve bvp for ac_init
solinitz=bvpinit(linspace(0,model_time,model_time/time_step),@bvp_init);
```

```
solinitx=bvpinit(linspace(0,model_time,model_time/time_step),@bvp_initx);
options=bvpset('Stats','on','RelTol',1e-1);
solz=bvp4c(@bvp_ode,@bvp_bc,solinitz,options);
solx=bvp4c(@bvp_odex,@bvp_bcx,solinitx,options);
tz=solz.x;              %time vector, z-axis
tx=solx.x;              %time vector, x-axis
z=solz.y;               %solution to boundary value problem, z-axis
x=solx.y;               %solution to boundary value problem, x-axis
dtz=tz(end)/length(tz);    %time step (sec), z-axis
dtx=tx(end)/length(tx);    %time step (sec), x-axis

%%VARIABLE INITIALIZATION
acz=ones(tr,1)*z(3,:); %initial optimal commanded acceleration, z-axis
acx=ones(tr,1)*x(3,:); %initial optimal commanded acceleration, x-axis
az=z(3,:);%initial vehicle acceleration, z-axis
ax=x(3,:);%initial vehicle acceleration, x-axis
wa=z(2,:);%initial vehicle acceleration, z-axis
ua=x(2,:);%initial vehicle acceleration, x-axis
za=z(1,:);%initial vehicle acceleration, z-axis
xa=x(1,:);%initial vehicle acceleration, x-axis
azi_p=zeros(size(acz));   %Predicted vehicle acceleration from all trajectories, z-axis
axi_p=zeros(size(acx));   %Predicted vehicle acceleration from all trajectories, x-axis
wa_p=ones(tr,1)*z(2,:);   %Predicted velocity, z-axis
za_p=ones(tr,1)*z(1,:);   %Predicted position, z-axis
ua_p=ones(tr,1)*x(2,:);   %Predicted velocity, x-axis
xa_p=ones(tr,1)*x(1,:);   %Predicted position, x-axis
pathcostz=zeros(size(acz));%Initial path cost, z-axis
pathcostx=zeros(size(acx));%Initial path cost, x-axis
Jzi=ones(size(acz));      %Performance index, z-axis
Jxi=ones(size(acx));      %Performance index, x-axis
Jzi_mod=ones(size(acz));  %Normalized by logarithmic scale (10^-3 to 10^1), z-axis
Jxi_mod=ones(size(acx));  %Normalized by logarithmic scale (10^-3 to 10^1), x-axis
dazi_e=zeros(size(acz));  %Exponential weighing factor, z-axis
daxi_e=zeros(size(acx));  %Exponential weighing factor, x-axis
azs=zeros(1,size(acz,2)); %Placeholder for ship's acceleration, z-axis
ws=zeros(1,size(acz,2)); %Placeholder for ship's velocity, z-axis
zs=zeros(1,size(acz,2)); %Placeholder for ship's position, z-axis
xs=zeros(1,size(acx,2)); %Placeholder for ship's position, x-axis
pz=zeros(1,size(acz,2)); %Placeholder number of usable trajectories
px=zeros(1,size(acx,2)); %Placeholder number of usable trajectories

%%SHIP'S VERTICAL MOTION MODEL
rng('shuffle');
s=1;                    %The following while loop calculates the ship's heave motion.
                        %Ship acceleration is based on random sine wave with "sd"
                        %standard deviation and "f" frequency with period
                        %between 4.5 - 8 seconds
while s<size(acz,2)
    A=sd*randn(1);            %Normally distributed amplitude
    f=0.09722*rand(1)+0.125;  %Frequency range of 4.5-8 sec.
    T_2=round((1/(dtz*f))/2);  %Half Period (index)
    hp=(s-1)+T_2;
    if hp<size(acz,2)
        for s=s:hp
           azs(s)=A*sin(2*pi()*f*tz(s)); %Calculates peak
        end
    else
        hp=size(acz,2);
         for s=s:hp
           azs(s)=A*sin(2*pi()*f*tz(s)); %Calculates peak
         end
    end
    A=sd*randn(1);            %Draws new normally distributed random amplitude
    ep=s+T_2;
    if ep<size(acz,2)
        for s=s+1:ep
           azs(s)=A*sin(2*pi()*f*tz(s)); %Calculates valley
        end
    else
        ep=size(acz,2);
        for s=s+1:ep
           azs(s)=A*sin(2*pi()*f*tz(s)); %Calculates valley
        end
    end
end

for s=2:size(acz,2)           %Calculates ship's vertical velocity and position
    ws(s)=azs(s)*dtz;
    zs(s)=zs(s-1)+ws(s)*dtz;
end
%%SHIP'S HORIZONTAL MOTION MODEL
us=sdx*randn(1,size(acx,2))+us*ones(1,size(acx,2)); %Ship's forward velocity with surge
motion
for s=2:size(acx,2)           %Calculates ship's x-position
    xs(s)=xs(s-1)+us(s)*dtx;
end
%%SENSITIVITY SELECTION FOR TRAJECTORY GENERATION
r=0.1*ones(1,size(acz,2)); %Sensitivity of differential commanded z-acceleration, starts
at 10% of initial optimal acceleration
rx=0.05*ones(1,size(acx,2));%Sensitivity of differential commanded x-acceleration, starts
at 2% of initial optimal acceleration
F=logspace(-2,.5,model_time/time_step);  %weight used to trade between path cost and
terminal cost

%%PATH INTEGRAL MODEL
%%Z-AXIS
for j=2:size(acz,2)
    n=tr;       %Counter of trajectories with final z-position outside of specified band
in relation to ship deck
    m=1;         %Counter for number of cycles in while loop
    while (n>40 && m<10)                    %Limits trajectories outside of an specified
band to 40/100 and 7 cycles
        amax=r(j)*abs(acz);                %Vehicle max differential acceleration ft/s^2
        dacz=2*amax.*rand(size(acz))-amax; %Creates random differential accelerations at
each node
        aci=acz+dacz;                         %Set of multiple trajectories
        [azi_p wa_p
za_p]=predicted_vehicle_state_z(aci,acz,azi_p,dtz,tau,wa_p,za_p,j);%Calculates predicted
vehicle state based on vehicle dynamics
        n=0;%Starts counter for trajectories with final z-position outside of specified
band
        for l=1:size(acz,1)   %Calculates the performance index for each trajectory
            pc=0;                           %Resets path cost
            for k=j:size(acz,2)
                pc=pc+0.5*dtz*aci(l,k)^2;    %Calculates path cost for each trajectory
            end
            pathcostz(l,j)=pc;
            if abs(za_p(l,end)-zs(j))>band_z%Terminal position of acceptable trajectories
is +-4.0 ft
```

```
                Jzi(l,j)=0;                    %from ship's final position
                n=n+1;
            else
                Jzi(l,j)=pathcostz(l,j)+F(j)*((wa_p(l,end)-ws(j))^2+(za_p(l,end)-
zs(j))^2); %Performance Index
            end
        end
        r(j)=r(j)+r_inc;                       %Increments sensitivity
        m=m+1;
    end
    pz(j)=size(acz,1)-n;                       %Number of usable trajectories
    Jzi_mod(:,j) = indexnorm2(Jzi(:,j),pz(j),tr);  %Normalizes the performance index to
give more weight to lowest cost trajectory
    for l=1:size(acz,1)                       %Performs exponential weighting of all
trajectories
        if Jzi_mod(l,j)==0
            for k=j:size(acz,2)
                dazi_e(l,k)=dacz(l,k)*0;
            end
        else
            for k=j:size(acz,2)
                dazi_e(l,k)=dacz(l,k)*exp(-Jzi_mod(l,j));
            end
        end
    end
    if n==tr                                  %Avoids dividing by 0
        for k=j:size(acz,2)                   %Uses previous optimal trajectory
            acz(:,k)=acz(1,k);
        end
    else
        for k=j:size(acz,2)                     %Calculates best trajectory at this time
step
            acz(:,k)=acz(1,k)+ sum(dazi_e(:,k))/(size(acz,1)-n);
        end
    end
    [az(j) wa(j) za(j)]=vehicle_dynamics_z(az,acz,dtz,tau,wa,za,j);%Calculates actual
vehicle state based on vehicle dynamics
end
%%PATH INTEGRAL MODEL
%%X-AXIS
for j=2:size(acx,2)
    n=tr;       %Counter of trajectories with final x-position outside of specified band
in relation to ship's landing position
    m=1;        %Counter for number of cycles in while loop
    while (n>90 && m<10)                        %Limits trajectories outside of an specified
band to 90/100 and 9 cycles
        axmax=rx(j)*abs(acx);                   %Vehicle max differential acceleration
ft/s^2
        dacx=2*axmax.*rand(size(acx))-axmax;    %Creates random differential accelerations
at each node
        acxi=acx+dacx;                           %Set of multiple trajectories
        [axi_p ua_p
xa_p]=predicted_vehicle_state_x(acxi,acx,axi_p,dtx,tau,ua_p,xa_p,j);%Calculates predicted
vehicle state based on vehicle dynamics
        n=0;%Starts counter for trajectories with final x-position outside of specified
band
        for l=1:size(acx,1)       %Calculates the performance index for each trajectory
            pc=0;                              %Resets path cost
            for k=j:size(acx,2)
                pc=pc+0.5*dtx*acxi(l,k)^2;    %Calculates path cost for each trajectory
            end
            pathcostx(l,j)=pc;
            if abs(xa_p(l,end)-xs(end))>band_x   %Terminal position of acceptable
trajectories is +-3.0 ft
                Jxi(l,j)=0;                      %from ship's final position
                n=n+1;
            else
                Jxi(l,j)=pathcostx(l,j)+F(j)*((ua_p(l,end)-us(end))^2+(xa_p(l,end)-
xs(end))^2); %Performance Index
            end
        end
        rx(j)=rx(j)+rx_inc;                          %Increments sensitivity
        m=m+1;
    end
    px(j)=size(acx,1)-n;                             %Number of usable trajectories
    Jxi_mod(:,j) = indexnorm2(Jxi(:,j),px(j),tr);  %Normalizes the performance index to
give more weight to lowest cost trajectory
    for l=1:size(acx,1)                          %Performs exponential weighting of all
trajectories
        if Jxi_mod(l,j)==0
            for k=j:size(acx,2)
                daxi_e(l,k)=dacx(l,k)*0;
            end
        else
            for k=j:size(acx,2)
                daxi_e(l,k)=dacx(l,k)*exp(-Jxi_mod(l,j));
            end
        end
    end
    if n==tr                                     %Avoids dividing by 0
        for k=j:size(acx,2)                       %Uses previous optimal trajectory
            acx(:,k)=acx(1,k);
        end
    else
        for k=j:size(acx,2)                        %Calculates best trajectory at this time
step
            acx(:,k)=acx(1,k)+ sum(daxi_e(:,k))/(size(acx,1)-n);
        end
    end
    [ax(j) ua(j) xa(j)]=vehicle_dynamics_x(ax,acx,dtx,tau,ua,xa,j);%Calculates actual
vehicle state based on vehicle dynamics
end
toc
figure (1)
subplot(2,1,1);
hold on
plot(tz,za,'r','LineWidth',1.5)
plot(tz,zs,'--b','LineWidth',1.5)
plot(tz,z(1,:),'c','LineWidth',1.5)
hold off
grid on
grid minor
set(gca,'fontsize',12)
ax1=gca;
set(ax1,'GridLineStyle','-');
set(ax1,'MinorGridLineStyle',':');
title('Vehicle Position, Z-axis','FontSize',12,'FontWeight','Bold');
xlabel('Time (sec)','FontSize',12);
ylabel('Altitude (ft)','FontSize',12);
```

```
leg=legend('Vehicle Vertical Position','Ship Vertical Position','Initial Optimal Z-
Position','location','best');
set(leg,'FontSize',10);
subplot(2,1,2);
hold on
plot(tx,xa,'r','LineWidth',1.5)
plot(tx,xs,'--b','LineWidth',1.5)
plot(tx,x(1,:),'c','LineWidth',1.5)
hold off
grid on
grid minor
set(gca,'fontsize',12)
ax1=gca;
set(ax1,'GridLineStyle','-');
set(ax1,'MinorGridLineStyle',':');
title('Vehicle Position, X-axis','FontSize',12,'FontWeight','Bold');
xlabel('Time (sec)','FontSize',12);
ylabel('Horizontal Position (ft)','FontSize',12);
leg=legend('Vehicle Horizontal Position','Ship Horizontal Position','Initial Optimal X-
Position','location','best');
set(leg,'FontSize',10);

figure (2)
subplot(2,1,1);
hold on
plot(tz,wa,'r','LineWidth',1.5)
plot(tz,ws,'--b','LineWidth',1.5)
plot(tz,z(2,:),'c','LineWidth',1.5)
hold off
grid on
grid minor
set(gca,'fontsize',12)
ax2=gca;
set(ax2,'GridLineStyle','-');
set(ax2,'MinorGridLineStyle',':');
title('Vehicle Velocity, Z-axis','FontSize',12,'FontWeight','Bold');
xlabel('Time (sec)','FontSize',12);
ylabel('Vertical Velocity (ft/s)','FontSize',12);
leg=legend('Vehicle Z-Velocity','Ship Z-Velocity','Initial Optimal Z-
Velocity','location','best');
set(leg,'FontSize',10);
subplot(2,1,2);
hold on
plot(tx,ua,'r','LineWidth',1.5)
plot(tx,us,'--b','LineWidth',1.5)
plot(tx,x(2,:),'c','LineWidth',1.5)
hold off
grid on
grid minor
set(gca,'fontsize',12)
ax2=gca;
set(ax2,'GridLineStyle','-');
set(ax2,'MinorGridLineStyle',':');
title('Vehicle Velocity, X-axis','FontSize',12,'FontWeight','Bold');
xlabel('Time (sec)','FontSize',12);
ylabel('Horizontal Velocity (ft/s)','FontSize',12);
leg=legend('Vehicle X-Velocity','Ship X-Velocity','Initial Optimal X-
Velocity','location','best');
set(leg,'FontSize',10);
```

```
figure (3)
subplot(2,1,1);
hold on
plot(tz,az,'r','LineWidth',1.5)
plot(tz,z(3,:),'c','LineWidth',1.5)
hold off
grid on
grid minor
set(gca,'fontsize',12)
ax3=gca;
set(ax3,'GridLineStyle','-');
set(ax3,'MinorGridLineStyle',':');
title('Vehicle Acceleration, Z-axis','FontSize',12,'FontWeight','Bold');
xlabel('Time (sec)','FontSize',12);
ylabel('Vertical Acceleration (ft/sec^2)','FontSize',12);
leg=legend('Vehicle Z-Acceleration','Initial Optimal Z-Acceleration','location','best');
set(leg,'FontSize',10);
subplot(2,1,2);
hold on
plot(tx,ax,'r','LineWidth',1.5)
plot(tx,x(3,:),'c','LineWidth',1.5)
hold off
grid on
grid minor
set(gca,'fontsize',12)
ax3=gca;
set(ax3,'GridLineStyle','-');
set(ax3,'MinorGridLineStyle',':');
title('Vehicle Acceleration, X-axis','FontSize',12,'FontWeight','Bold');
xlabel('Time (sec)','FontSize',12);
ylabel('Horizontal Acceleration (ft/sec^2)','FontSize',12);
leg=legend('Vehicle X-Acceleration','Initial Optimal X-Acceleration','location','best');
set(leg,'FontSize',10);

figure (4)
plot(tz,azs,'r','LineWidth',1.5)
grid on
grid minor
set(gca,'fontsize',12)
ax4=gca;
set(ax4,'GridLineStyle','-');
set(ax4,'MinorGridLineStyle',':');
title('Ship Vertical Acceleration','FontSize',12,'FontWeight','Bold');
xlabel('Time (sec)','FontSize',12);
ylabel('Vertical Acceleration (ft/sec^2)','FontSize',12);

figure (5)
subplot(2,1,1);
hold on
plot(tz,za,'r','LineWidth',1.5)
plot(tz,za_p(5,:),'--g','LineWidth',1.5)
plot(tz,za_p(17,:),':m','LineWidth',1.5)
plot(tz,za_p(56,:),':g','LineWidth',1.5)
plot(tz,za_p(79,:),':k','LineWidth',1.5)
plot(tz,za_p(99,:),'--m','LineWidth',1.5)
plot(tz,zs,'--b','LineWidth',1.5)
plot(tz,z(1,:),'c','LineWidth',1.5)
hold off
grid on
grid minor
set(gca,'fontsize',12)
ax1=gca;
set(ax1,'GridLineStyle','-');
set(ax1,'MinorGridLineStyle',':');
title('Vehicle Position, Z-axis','FontSize',12,'FontWeight','Bold');
xlabel('Time (sec)','FontSize',12);
ylabel('Altitude (ft)','FontSize',12);
leg=legend('Vehicle Vertical Position','Predicted Trajectory','Predicted
Trajectory','Predicted Trajectory','Predicted Trajectory','Predicted Trajectory','Ship
Vertical Position','Initial Optimal Z-Position','location','best');
set(leg,'FontSize',10);
subplot(2,1,2);
hold on
plot(tx,xa,'r','LineWidth',1.5)
plot(tx,xa_p(5,:),'--g','LineWidth',1.5)
plot(tx,xa_p(17,:),'--m','LineWidth',1.5)
plot(tx,xa_p(56,:),':g','LineWidth',1.5)
plot(tx,xa_p(79,:),':k','LineWidth',1.5)
plot(tx,xa_p(99,:),':m','LineWidth',1.5)
plot(tx,xs,'--b','LineWidth',1.5)
plot(tx,x(1,:),'c','LineWidth',1.5)
hold off
grid on
grid minor
set(gca,'fontsize',12)
ax1=gca;
set(ax1,'GridLineStyle','-');
set(ax1,'MinorGridLineStyle',':');
title('Vehicle Position, X-axis','FontSize',12,'FontWeight','Bold');
xlabel('Time (sec)','FontSize',12);
ylabel('Horizontal Position (ft)','FontSize',12);
leg=legend('Vehicle Horizontal Position','Predicted Trajectory','Predicted
Trajectory','Predicted Trajectory','Predicted Trajectory','Predicted Trajectory','Ship
Horizontal Position','Initial Optimal X-Position','location','best');
set(leg,'FontSize',10);
toc
%Parametric Study
ze=za(end)-zs(end)
we=wa(end)-ws(end)
pzmean=mean(pz)
xe=xa(end)-xs(end)
ue=ua(end)-us(end)
pxmean=mean(px)

function [azi_p wa_p za_p]=predicted_vehicle_state_z(aci,acz,azi_p,dtz,tau,wa_p,za_p,j)
%Calculates vehicle predicted state based on vehicle dynamics
for k=j:size(acz,2)
    for l=1:size(acz,1)
        azi_p(l,k)=azi_p(l,k-1)+((dtz/tau)*(aci(l,k)-azi_p(l,k-1)));%Vehicle acceleration
from all trajectories
        wa_p(l,k)=wa_p(l,k-1)+azi_p(l,k)*dtz;                          %Calculates velocity
for all trajectories
        za_p(l,k)=za_p(l,k-1)+wa_p(l,k)*dtz;   %Calculates position for all trajectories
    end
end
end

function [axi_p ua_p xa_p]=predicted_vehicle_state_x(acxi,acx,axi_p,dtx,tau,ua_p,xa_p,j)
%Calculates vehicle predicted state based on vehicle dynamics
for k=j:size(acx,2)
```

```
    for l=1:size(acx,1)
        axi_p(l,k)=axi_p(l,k-1)+((dtx/tau)*(acxi(l,k)-axi_p(l,k-1)));%Vehicle
acceleration from all trajectories
        ua_p(l,k)=ua_p(l,k-1)+axi_p(l,k)*dtx; %Calculates velocity for all trajectories
        xa_p(l,k)=xa_p(l,k-1)+ua_p(l,k)*dtx;  %Calculates position for all trajectories
    end
end
end

function [az wa za]=vehicle_dynamics_z(az,acz,dtz,tau,wa,za,j)
%Calculates actual vehicle state based on vehicle dynamics
az(1,j)=az(1,j-1)+((dtz/tau)*(acz(1,j)-az(1,j-1)));%Updates current vehicle acceleration
wa(1,j)=wa(1,j-1)+az(1,j)*dtz;            %Updates current vehicle velocity based on best
trajectory
za(1,j)=za(1,j-1)+wa(1,j)*dtz;          %Updates current vehicle position based on best
trajectory
az=az(1,j);
wa=wa(1,j);
za=za(1,j);
end

function [ax ua xa]=vehicle_dynamics_x(ax,acx,dtx,tau,ua,xa,j)
%Calculates actual vehicle state based on vehicle dynamics
ax(1,j)=ax(1,j-1)+((dtx/tau)*(acx(1,j)-ax(1,j-1)));%Updates vehicle x-acceleration
ua(1,j)=ua(1,j-1)+ax(1,j)*dtx;                   %Updates vehicle x-velocity based on
best trajectory
xa(1,j)=xa(1,j-1)+ua(1,j)*dtx;                 %Updates vehicle x-position based on best
trajectory
ax=ax(1,j);
ua=ua(1,j);
xa=xa(1,j);
end

function res = bvp_bc(ya,yb)
res=[ya(1)-100 ya(2)-0 ya(3)-0 yb(1)-0 yb(2)-0 yb(3)-0];
end

function res = bvp_bcx(xa,xb)
%res=[xa(1)+1745.2 xa(2)-95.4 xa(3)-0 xb(1)-162.8 xb(2)-8.14 xb(3)-0]; %LOW level sea
state
res=[xa(1)+1238.2 xa(2)-95.4 xa(3)-0 xb(1)-669.8 xb(2)-33.49 xb(3)-0]; %MEDIUM level sea
state
%res=[xa(1)+908.4 xa(2)-95.4 xa(3)-0 xb(1)-999.6 xb(2)-49.98 xb(3)-0]; %HIGH level sea
state
end

function v = bvp_init(t)
v=[-5*t+100 -1/10*(t-10)^2+10 1.5*sin(pi()*t/10) 1 1 1];
end

function v = bvp_initx(t)
%v=[95.4*t-1745.2 -4.363*t+95.4 0 1 1 1]; %LOW level sea state
v=[95.4*t-1238.2 -4.363*t+95.4 0 1 1 1]; %MEDIUM level sea state
%v=[95.4*t-908.4 -4.363*t+95.4 0 1 1 1]; %HIGH level sea state
end

function dydt = bvp_ode(t,y)
dt=0.1;
tau=1;
```

```
ac=-y(6)/dt;
dydt=[y(2) y(3) -(1/tau)*y(3)+(1/tau)*ac 0 -y(4) -y(5)+y(6)/dt];
end

function dxdt = bvp_odex(t,x)
dt=0.1;
tau=1;
ac=-x(6)/dt;
dxdt=[x(2) x(3) -(1/tau)*x(3)+(1/tau)*ac 0 -x(4) -x(5)+x(6)/dt];
end

function [Ji_mod] = indexnorm2(Ji,p)
[J,i]=sort(Ji);
if p<100
    logscale=[zeros(1,100-p),logspace(-3,1,p)];
else
    logscale=logspace(-3,1,p);
end
Ji_mod=J.*logscale';
Ji_mod=[i,Ji_mod];
Ji_mod=sortrows(Ji_mod,1);
Ji_mod=Ji_mod(:,2);
end
```

Appendix B – Model Results for 90 Total Runs

	LOW Sea State									
Run	**Pos Error (ze)**	**Abs Pos Error (ze)**	**Vel Error (we)**	**Abs Vel Error (we)**	**Average Usable Trajectories**	**Pos Error (xe)**	**Abs Pos Error (xe)**	**Vel Error (ue)**	**Average Usable Trajectories**	**Running Time**
1	0.3249	0.3249	-2.0391	2.0391	99.49	-0.0526	0.0526	8.0844	86.4672	7.313
2	0.8547	0.8547	-2.1499	2.1499	99.5	0.4902	0.4902	7.9885	93.6642	6.29
3	0.8362	0.8362	-2.0285	2.0285	99.495	0.3227	0.3227	8.015	93.0365	6.342
4	0.576	0.576	-1.99	1.99	99.5	0.2737	0.2737	8.1107	92.1533	6.37
5	0.2753	0.2753	-2.1335	2.1335	99.49	0.3342	0.3342	8.173	90.8613	6.548
6	0.4042	0.4042	-2.0481	2.0481	99.5	0.0145	0.0145	7.7877	89.6496	6.524
7	0.7288	0.7288	-1.8655	1.8655	99.5	0.4337	0.4337	7.9378	93.5474	6.32
8	0.5014	0.5014	-2.1109	2.1109	99.495	0.2129	0.2129	8.3668	91.6204	6.523
9	0.8129	0.8129	-2.164	2.164	99.49	0.2121	0.2121	7.9343	92.4234	6.307
10	0.5369	0.5369	-1.9853	1.9853	99.5	0.7395	0.7395	8.2413	95.3285	6.278
11	0.5672	0.5672	-2.0045	2.0045	99.5	0.1277	0.1277	7.7408	91.0949	6.388
12	0.5816	0.5816	-2.1062	2.1062	99.495	0.057	0.057	8.1301	89.8175	6.319
13	0.893	0.893	-2.186	2.186	99.495	0.4831	0.4831	8.2254	93.2263	6.3
14	0.6122	0.6122	-2.0057	2.0057	99.49	0.1401	0.1401	7.8459	91.0219	6.413
15	0.4312	0.4312	-2.0507	2.0507	99.5	0.3382	0.3382	7.9425	92.927	6.315
16	0.6208	0.6208	-2.0155	2.0155	99.495	0.2295	0.2295	8.2305	91.7883	6.359
17	0.3683	0.3683	-1.8591	1.8591	99.5	0.5843	0.5843	8.0785	94.365	6.319
18	0.3044	0.3044	-1.9702	1.9702	99.49	0.4663	0.4663	7.9325	93.4672	6.283
19	0.5673	0.5673	-2.0694	2.0694	99.5	0.2027	0.2027	8.1501	91.3869	6.375
20	0.952	0.952	-2.0136	2.0136	99.49	0.1636	0.1636	7.9655	88.8978	6.213
21	0.5321	0.5321	-2.0641	2.0641	99.495	0.1023	0.1023	7.947	89.8613	6.369
22	0.3956	0.3956	-2.0242	2.0242	99.5	0.2042	0.2042	7.8442	89.1241	6.358
23	0.8629	0.8629	-1.931	1.931	99.495	0.1737	0.1737	8.2662	90.1679	6.406
24	0.7871	0.7871	-2.0307	2.0307	99.5	0.5116	0.5116	8.0065	94.6058	6.309
25	0.5189	0.5189	-2.0619	2.0619	99.5	0.6954	0.6954	8.4806	95.7007	6.359
26	-0.0424	0.0424	-2.0024	2.0024	99.495	0.3331	0.3331	7.854	92.7664	6.506
27	0.6224	0.6224	-1.9587	1.9587	99.49	0.4597	0.4597	8.1992	95.1606	6.295
28	0.4162	0.4162	-2.0693	2.0693	99.495	0.3548	0.3548	7.9667	93.6788	6.308
29	0.4707	0.4707	-2.1393	2.1393	99.495	-0.1877	0.1877	8.0405	83.6131	6.283
30	0.2597	0.2597	-1.73	1.73	99.5	0.2657	0.2657	8.0508	90.635	6.443
AVERAGE	**0.55241667**	**0.55524333**	**-2.02691**	**2.02691**	**99.496**	**0.28954**	**0.30556**	**8.05123333**	**91.7352767**	**6.39116667**

	MEDIUM Sea State									
Run	**Pos Error (ze)**	**Abs Pos Error (ze)**	**Vel Error (we)**	**Abs Vel Error (we)**	**Average Usable Trajectories**	**Pos Error (xe)**	**Abs Pos Error (xe)**	**Vel Error (ue)**	**Average Usable Trajectories**	**Running Time**
1	0.98	0.98	-1.9988	1.9988	99.495	0.1832	0.1832	6.6095	83.3212	6.902
2	0.2347	0.2347	-2.3354	2.3354	99.5	0.1967	0.1967	7.372	82.3723	6.663
3	-0.2178	0.2178	-1.9446	1.9446	99.5	0.1896	0.1896	6.5058	82.5766	6.454
4	1.7319	1.7319	-1.9586	1.9586	99.5	0.396	0.396	6.7613	85.7372	6.26
5	0.3284	0.3284	-2.176	2.176	99.5	0.2738	0.2738	7.3226	82.2555	6.384
6	0.1244	0.1244	-2.6727	2.6727	99.495	0.2258	0.2258	6.2185	81.292	6.381
7	-0.3941	0.3941	-2.2038	2.2038	99.5	0.3304	0.3304	6.6382	82.4526	6.205
8	-0.6702	0.6702	-1.774	1.774	99.5	0.2555	0.2555	6.836	82.2263	6.271
9	-0.2308	0.2308	-1.8726	1.8726	99.495	0.2858	0.2858	6.888	84.927	6.297
10	0.1026	0.1026	-2.1621	2.1621	99.5	0.3943	0.3943	6.9133	84.9781	6.286
11	1.0155	1.0155	-2.1114	2.1114	99.5	0.3115	0.3115	6.4987	83.5109	6.391
12	0.984	0.984	-1.9068	1.9068	99.49	0.2861	0.2861	6.7267	84.5839	6.437
13	-0.6695	0.6695	-1.9352	1.9352	99.495	-7.8246	7.8246	4.8136	0.4672	7.435
14	0.0888	0.0888	-1.998	1.998	99.48	0.2634	0.2634	6.6602	84.219	6.339
15	0.5317	0.5317	-1.7728	1.7728	99.5	0.3391	0.3391	6.7269	85.1022	6.164
16	1.055	1.055	-1.8035	1.8035	99.5	0.2597	0.2597	6.7812	83.8248	6.507
17	-0.3642	0.3642	-2.3904	2.3904	99.5	0.3047	0.3047	6.6838	83.7445	6.289
18	-0.2858	0.2858	-2.1591	2.1591	99.5	0.2934	0.2934	6.7221	82.2774	6.383
19	0.587	0.587	-1.8947	1.8947	99.49	0.1969	0.1969	7.0546	83.5036	6.244
20	-0.3129	0.3129	-1.9527	1.9527	99.495	0.4097	0.4097	6.8883	86.4307	6.285
21	-0.0438	0.0438	-2.1642	2.1642	99.5	-0.7598	0.7598	6.8219	61.1533	6.602
22	0.2518	0.2518	-2.1445	2.1445	99.495	-0.0201	0.0201	6.5811	77.0511	6.571
23	-0.0935	0.0935	-1.7702	1.7702	99.5	0.1649	0.1649	6.5396	81.6058	6.515
24	1.0199	1.0199	-2.1503	2.1503	99.5	0.3829	0.3829	6.5327	83.1314	6.44
25	1.5549	1.5549	-2.041	2.041	99.495	0.2677	0.2677	7.3223	81.4307	6.311
26	0.7908	0.7908	-1.9005	1.9005	99.495	0.3219	0.3219	6.3189	81.6642	6.25
27	0.6179	0.6179	-2.0499	2.0499	99.495	-0.7552	0.7552	6.4875	62.0511	6.793
28	2.3972	2.3972	-2.4049	2.4049	99.495	0.1658	0.1658	6.4413	83.2409	6.312
29	0.3806	0.3806	-2.1946	2.1946	99.5	0.3882	0.3882	6.0154	83.0949	6.235
30	0.3865	0.3865	-2.0507	2.0507	99.5	0.1621	0.1621	6.6175	80.8686	6.189
AVERAGE	**0.39603333**	**0.61487333**	**-2.06313333**	**2.06313333**	**99.497**	**-0.07035333**	**0.55362667**	**6.64331667**	**78.8365**	**6.4265**

	HIGH Sea State									
Run	**Pos Error (ze)**	**Abs Pos Error (ze)**	**Vel Error (we)**	**Abs Vel Error (we)**	**Average Usable Trajectories**	**Pos Error (xe)**	**Abs Pos Error (xe)**	**Vel Error (ue)**	**Average Usable Trajectories**	**Running Time**
1	-0.1599	0.1599	-2.2249	2.2249	99.495	0.5768	0.5768	5.7745	85.8029	6.771
2	-0.3164	0.3164	-1.9364	1.9364	99.5	0.5324	0.5324	5.1301	86.0876	6.535
3	-0.0745	0.0745	-2.0614	2.0614	99.485	0.4847	0.4847	6.028	87.5766	6.774
4	-1.4751	1.4751	-2.7095	2.7095	99.5	0.3677	0.3677	6.1729	85.4307	6.833
5	1.3543	1.3543	-1.6775	1.6775	99.495	0.6558	0.6558	4.8737	85.9927	6.51
6	-0.874	0.874	-1.9468	1.9468	99.5	0.5196	0.5196	6.0046	84.1022	6.863
7	-0.9853	0.9853	-1.8783	1.8783	99.5	0.3678	0.3678	5.2502	85.1314	7.001
8	-0.4523	0.4523	-2.3379	2.3379	99.3	0.5463	0.5463	5.1981	86.6204	6.8
9	-0.2381	0.2381	-2.6273	2.6273	99.39	0.5057	0.5057	5.865	84.3212	6.997
10	1.2368	1.2368	-1.9429	1.9429	99.45	0.5701	0.5701	5.4696	86.0949	6.804
11	1.0793	1.0793	-1.8172	1.8172	98.925	0.297	0.297	5.5812	84.5036	6.965
12	1.4518	1.4518	-1.2604	1.2604	96.805	0.6595	0.6595	6.0324	87.6204	6.691
13	2.2882	2.2882	-2.0353	2.0353	98.845	0.5314	0.5314	5.5693	86.8467	6.84
14	-0.9205	0.9205	-3.0549	3.0549	99.34	0.1824	0.1824	7.2093	82.1314	6.96
15	-1.7086	1.7086	-0.25	0.25	98.045	0.4268	0.4268	5.4507	85.2701	6.383
16	0.2402	0.2402	-1.9183	1.9183	99.13	0.4808	0.4808	5.1303	83.9708	6.914
17	2.4957	2.4957	-1.7339	1.7339	99.105	-10.8185	10.8185	3.7606	0.0511	12.132
18	2.06	2.06	-2.6851	2.6851	97.845	0.3909	0.3909	5.9247	83.2628	7.04
19	-3.2747	3.2747	-2.5004	2.5004	98.84	0.5255	0.5255	5.4347	90.2628	6.521
20	2.1497	2.1497	-1.6784	1.6784	98.335	0.4461	0.4461	5.5621	84.6058	6.719
21	0.4906	0.4906	-1.9762	1.9762	99.125	0.6819	0.6819	5.6137	91.5401	6.71
22	1.7645	1.7645	-2.1068	2.1068	98.27	-0.6753	0.6753	5.9016	60.2555	7.354
23	3.5001	3.5001	-2.0871	2.0871	96.865	0.4396	0.4396	4.9799	85.0146	6.746
24	0.6466	0.6466	-2.3978	2.3978	99.23	0.4929	0.4929	5.6534	88.4964	6.491
25	1.6677	1.6677	-2.141	2.141	99.33	0.5888	0.5888	5.8764	87.4745	6.364
26	0.6011	0.6011	-2.1981	2.1981	99.095	0.5466	0.5466	4.0088	85.5255	6.904
27	2.782	2.782	-2.7462	2.7462	97.61	0.3951	0.3951	6.1324	83.5547	7.017
28	0.3404	0.3404	-1.8963	1.8963	99.335	0.1654	0.1654	7.2817	83.0949	6.871
29	0.6877	0.6877	-2.1515	2.1515	99.415	0.3845	0.3845	6.0855	86.5182	6.499
30	-1.4716	1.4716	-2.729	2.729	99.45	0.6236	0.6236	4.6575	86.7372	6.59
AVERAGE	**0.49619**	**1.29292333**	**-2.09022667**	**2.09022667**	**98.9185**	**0.06306333**	**0.82931667**	**5.58709667**	**82.1299233**	**6.9533**

Appendix C – Low Sea State Results

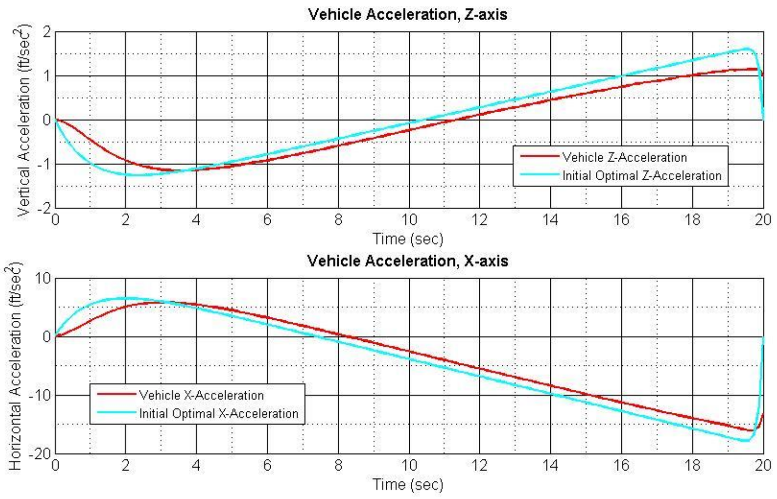

Figure 6: (Top) Vehicle command acceleration in the z-axis (optimal trajectory) versus initial optimal acceleration solution; (bottom) vehicle command acceleration in the x-axis versus initial optimal acceleration solution.

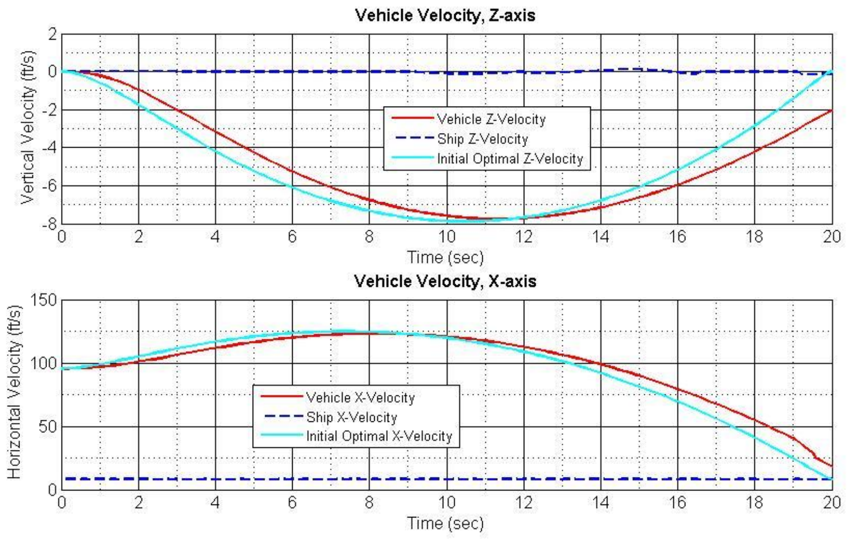

Figure 7: (Top) Vehicle velocity in the z-axis versus initial optimal solution and ship vertical velocity; (bottom) vehicle velocity in the x-axis versus initial optimal solution and ship horizontal velocity.

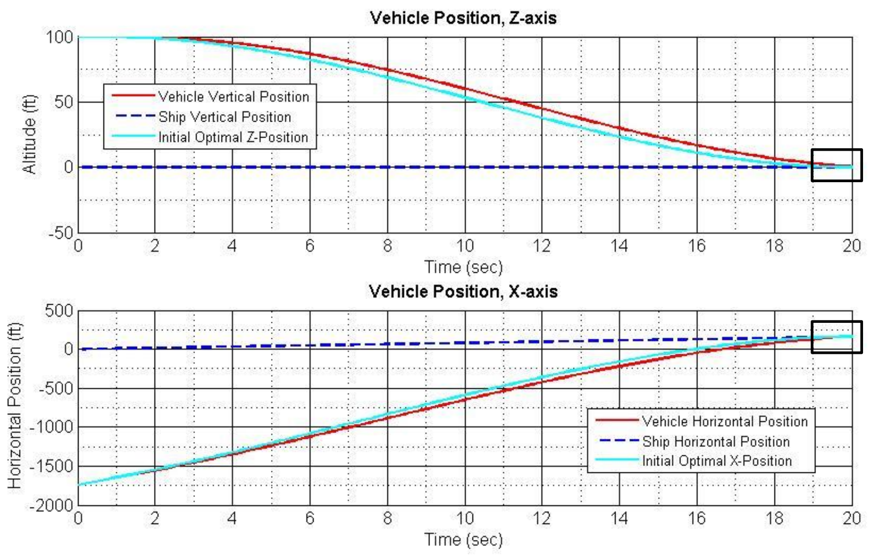

Figure 8: (Top) Vehicle position in the z-axis versus initial optimal solution and ship vertical position; (bottom) vehicle position in the x-axis versus initial optimal solution and ship horizontal position.

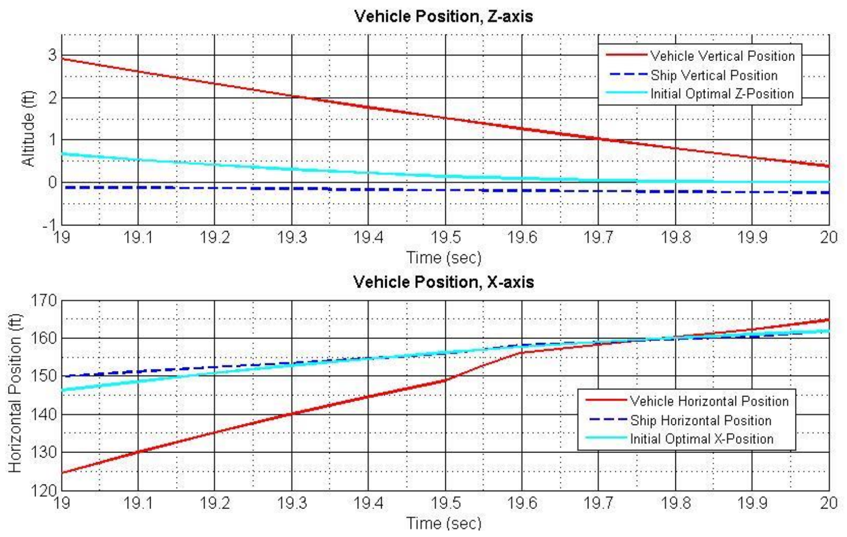

Figure 9: (Top) Last second in vehicle's landing showing vehicle position in the z-axis versus initial optimal solution and ship vertical position; (bottom) last second in vehicle's landing showing vehicle position in the x-axis versus initial optimal solution and ship horizontal position.

Appendix C –High Sea State Results

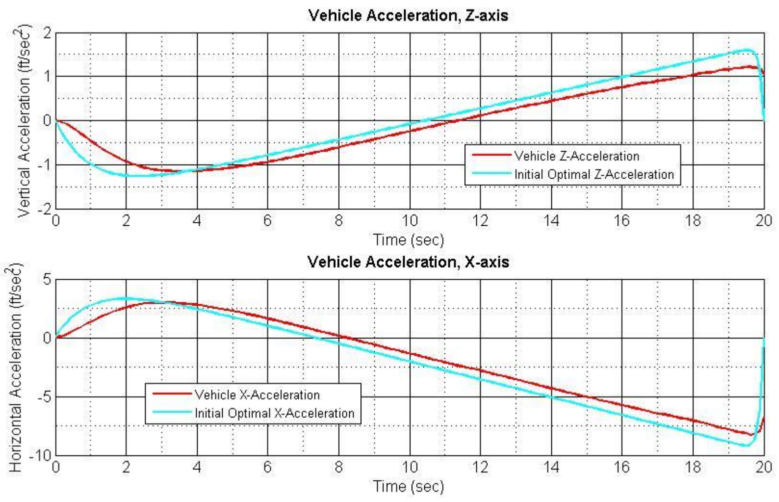

Figure 10: (Top) Vehicle command acceleration in the z-axis (optimal trajectory) versus initial optimal acceleration solution; (bottom) vehicle command acceleration in the x-axis versus initial optimal acceleration solution.

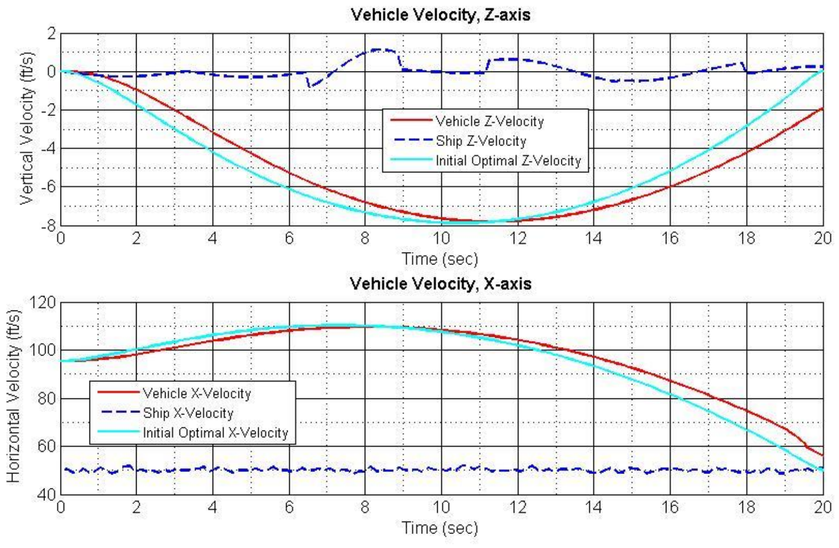

Figure 11: (Top) Vehicle velocity in the z-axis versus initial optimal solution and ship vertical velocity; (bottom) vehicle velocity in the x-axis versus initial optimal solution and ship horizontal velocity

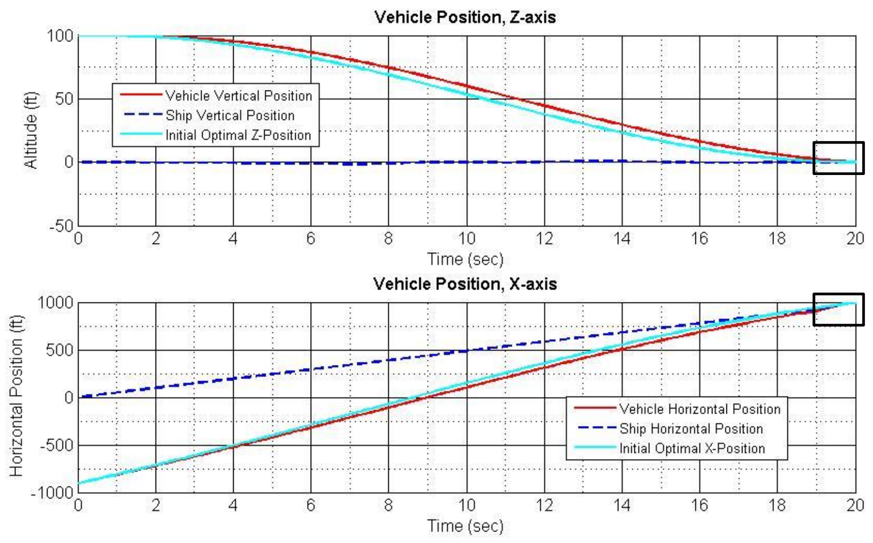

Figure 12: (Top) Vehicle position in the z-axis versus initial optimal solution and ship vertical position; (bottom) vehicle position in the x-axis versus initial optimal solution and ship horizontal position.

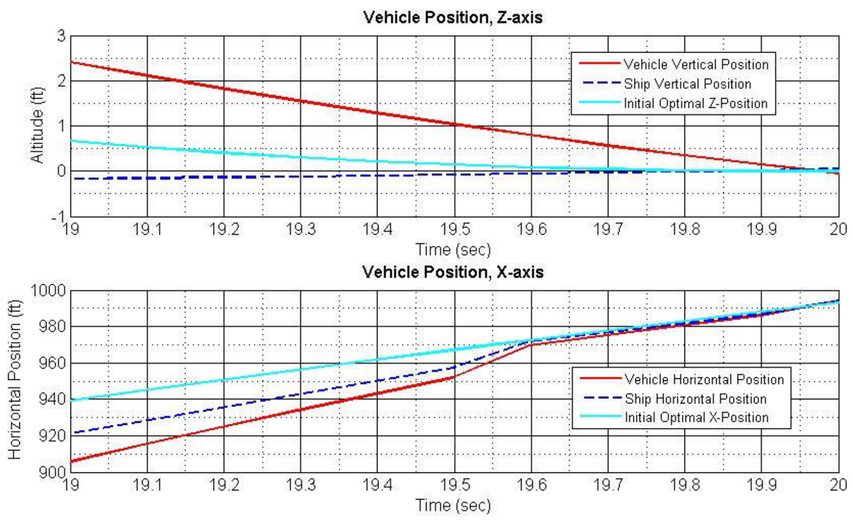

Figure 13: (Top) Last second in vehicle's landing showing vehicle position in the z-axis versus initial optimal solution and ship vertical position; (bottom) last second in vehicle's landing showing vehicle position in the x-axis versus initial optimal solution and ship horizontal position.