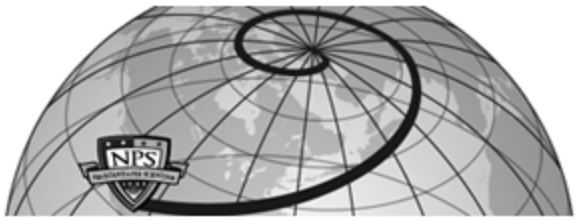


NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

SMART VOYAGE PLANNING MODEL SENSITIVITY ANALYSIS USING OCEAN AND ATMOSPHERIC MODELS INCLUDING ENSEMBLE METHODS

by

Scott E. Miller

September 2012

Thesis Co-Advisors: Peter C. Chu
James A. Hansen

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2012	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Smart Voyage Planning Model Sensitivity Analysis Using Ocean and Atmospheric Models Including Ensemble Methods		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Scott E. Miller			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) Kevin Lacroix, CNMOC 1002 Balch Blvd Stennis Space Center, MS 39529		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number _____N/A_____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

Smart Voyage Planning (SVP) has been identified as a key technology for the US Navy, capable of assisting with the fleet energy saving goals of Secretary of the Navy (SECNAV) and the Chief of Naval Operations (CNO). Commercial SVP tools use weather, waves, and specific ship platform characteristic data to develop optimal transit routes that save on the order of 5% in fuel expenditures. Sensitivity analysis was conducted utilizing a Naval Research Laboratory (NRL) industry standard SVP model. Model inputs used for sensitivity included ensemble techniques, ship class specific characteristics, and simulated enhanced environmental model outputs. Variances in predicted route costs compared to route costs using actual analysis environmental data and Great Circle baseline references were studied. Significant efforts focused on developing analysis tools to determine how uncertainty and sensitivity could be communicated. The analysis identified significant SVP model sensitivities to: geographic location; direction; seasonal synoptic weather; hull/propulsion type and condition; route length; specific model improvements; and ensemble post-processing methods. An SVP trial was also conducted at sea onboard USS PRINCETON (CG-59) with goals of Concept of Operations (CONOPS) development and determining types of operations that could affect a combatant vessel's ability to execute SVP routes. Important lessons learned, best practices, and recommendations were identified during this operational trial.

14. SUBJECT TERMS Smart Voyage Planning, Sensitivity Analysis, Ensembles, Environmental Models, CONOPS, Decision Aid.			**15. NUMBER OF PAGES** 202
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

SMART VOYAGE PLANNING MODEL SENSITIVITY ANALYSIS USING OCEAN AND ATMOSPHERIC MODELS INCLUDING ENSEMBLE METHODS

Scott E. Miller
Lieutenant Commander, United States Navy
B.S., University of South Carolina, 2000
B.S., Southern Illinois University, 2001
M.S., Old Dominion University, 2008

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN METEOROLOGY AND PHYSICAL OCEANOGRAPHY

from the

NAVAL POSTGRADUATE SCHOOL
September 2012

Author: Scott E. Miller

Approved by: Peter C. Chu
Thesis Co-Advisor

James A. Hansen
Thesis Co-Advisor

Mary L. Batteen
Chair, Department of Oceanography

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Smart Voyage Planning (SVP) has been identified as a key technology for the US Navy, capable of assisting with the fleet energy saving goals of Secretary of the Navy (SECNAV) and the Chief of Naval Operations (CNO). Commercial SVP tools use weather, waves, and specific ship platform characteristic data to develop optimal transit routes that save on the order of 5% in fuel expenditures. Sensitivity analysis was conducted utilizing a Naval Research Laboratory (NRL) industry standard SVP model. Model inputs used for sensitivity included ensemble techniques, ship class specific characteristics, and simulated enhanced environmental model outputs. Variances in predicted route costs compared to route costs using actual analysis environmental data and Great Circle baseline references were studied. Significant efforts focused on developing analysis tools to determine how uncertainty and sensitivity could be communicated. The analysis identified significant SVP model sensitivities to: geographic location; direction; seasonal synoptic weather; hull/propulsion type and condition; route length; specific model improvements; and ensemble post-processing methods. An SVP trial was also conducted at sea onboard USS PRINCETON (CG-59) with goals of Concept of Operations (CONOPS) development and determining types of operations that could affect a combatant vessel's ability to execute SVP routes. Important lessons learned, best practices, and recommendations were identified during this operational trial.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

3DVAR	Three-Dimensional Variational
4DVAR	Four-Dimensional Variational
AIRS	Atmospheric Infrared Sounder
AMSU	Advanced Microwave Sounding Unit
AMV	Atmospheric Motion Vector
AOR	Area of Operation
API	Application Programming Interface
ASCAT	Advanced Scatterometer
ATCF	Automated Tropical Cyclone Forecasting System
AVHRR	Advanced Very High Resolution Radiometer
BASH	Bourne-again Shell
C3F	Commander, Third Fleet
CG	Guided Missile Cruiser
CNMOC	Commander, Naval Meteorology and Oceanography Command
CNO	Chief of Naval Operations
COAMPS	Coupled Ocean/Atmosphere Mesoscale Prediction System
COGENT	Common Geospatial Navigation Toolkit
CONOPS	Concept of Operations
COTS	Commercial Off-the-Shelf
DBDB-V	Digital Bathymetric Data Base Variable Resolution
DCAP	Data Collection and Analysis Plan
DDG	Guided Missile Destroyer
DMS	Defense Message System

DMSP	Defense Meteorological Satellite Program
DoD	Department of Defense
DON	Department of the Navy
DOO WX SVCS	Director of Operational Oceanography, Weather Services
DR	Dead Reckon
DTG	Day Time Group
ECDIS-N	Electronic Chart Display and Information System – Navy
ECMWF	European Center for Medium Range Weather Forecasts
EFAS	Ensemble Forecast Application System
ERS-2	European Remote-Sensing Satellite
FNMOC	Fleet Numerical Meteorology and Oceanography Center
FTP	File Transfer Protocol
FWC	Fleet Weather Center
GEO	Geostationary Earth Orbit
GPS	Global Positioning System
GRIB	Gridded Binary
IASI	Infrared Atmospheric Sounding Interferometer
IPT	Integrated Program Team
JMV	Joint METOC Viewer
JTWC	Joint Typhoon Warning Center
LEO	Low Earth Orbit
METCAST	Meteorological Broadcast
METOC	Meteorology and Oceanography
MHS	Microwave Humidity Sounder
MODAS	Modular Ocean Data Assimilation System

MODIS	Moderate Resolution Imaging Spectroradiometer
MOE	Measure of Effectiveness
MOVEREP	Movement Report
MSC	Military Sealift Command
NAVDAS-AR	NRL Atmospheric Variational Data Assimilation System-Accelerated Representer
NCEP	United States National Centers for Environmental Prediction
NCOM	Navy Operational Global Ocean Model
NMFC	Navy Maritime Forecast Center
NOAA	National Oceanic and Atmospheric Administration
NOGAPS	Navy Operational Global Atmospheric Prediction System
NOOC	Naval Oceanography Operations Command
OPDEMO	Operational Demonstration
OPORDER	Operational Order
NRL	Naval Research Laboratory
NWP	Numerical Weather Prediction
OAML	Oceanographic and Atmospheric Master Library
OPC	Ocean Prediction Center
OPTASK	Operational Tasking
OSFR	Optimal Ship Fuel Routing
OTSR	Optimum Track Ship Routing
PIM	Position of Intended Movement
POTS	Plain Old Telephone System
RAOB	Rawinsonde Observation
RASP	Replenishment at Sea Program

RIMPAC	Rim of the Pacific Naval Exercise
SECNAV	Secretary of the Navy
SMP	U.S Navy Ship Motions Program Ship Motions Program
SOA	Speed of Advance
SSMI	Special Sensor Microwave/Imager
SST	Sea Surface Temperature
STARS	Ship Tracking and Routing System
SVP	Smart Voyage Plan
SVPDA	Smart Voyage Planning Decision Aid
SWH	Significant Wave Height
T-AO	Fleet Replenishment Oiler; Operated by MSC
TCW	Tropical Cyclone Warning
TDA	Tactical Decision Aid
TFE	Task Force Energy
UCAR	University Corporation for Atmospheric Research
UNIX	Uniplexed Information and Computing System
UNREP	Underway Replenishment
URL	Uniform Resource Locator
VERES	Marintek's Vessel Responses
VMS	Voyage Management System
WAM	Wave Modeling Project
WAMDIG	Wave Model and Development Implementation Group
WW3	WAVEWATCH III
WX	Weather
XML	Extensible Markup Language

ACKNOWLEDGMENTS

First, I would like to thank Professor Peter C. Chu at NPS for his encouragement and patience while being a strong advocate for my research. I would also like to especially thank Dr. James A. Hansen of NRL Monterey for his clever insight, professionalism, and consistent availability in guiding me to completion. I owe a strong thanks to the entire NRL Monterey staff in supporting my efforts throughout my thesis research experiments over the past several months. A special mention must be made and additional strong thanks go to Tim Whitcomb for helping to solve those pesky PYTHON programming challenges. I must also send a special thanks to NAVSEA and the entire SVPDA IPT team for allowing me to assist with prototype development and testing at sea.

Finally, I would like to thank my family. To the love of my life, my wife Shanon, whose constant caring and love kept me healthy and motivated to complete this task; to my son, Shane, whose work ethic and success as a young adult enabled my full concentration on studies; to my parents, who have dedicated their lives to loving and supporting unconditionally their children. I am blessed with a wonderful family that has supported me throughout my entire endeavors thus far in life and Naval career.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Task Force Energy (TFE) has identified Smart Voyage Planning (SVP) as a key technology, capable of reducing the Navy's carbon footprint and accomplishing energy saving goals of the Secretary of Navy (SECNAV) and Chief of Naval Operations (CNO). Commercial SVP tools currently use weather, ocean waves and specific ship platform characteristic data to develop optimal transit routes that save on the order of 5% in fuel expenditures. Today's robust platform hydrodynamic models and environmental forecasts combined with improved algorithms enable fuel savings in addition to aiding in heavy weather avoidance. However, with enhanced model output, the improvement in the least cost route as compared to the best possible route using analysis environmental data in SVP models has not been thoroughly studied.

This thesis attempts to explore this area by using sensitivity analysis. Sensitivity analysis contributes to model development, model calibration, model validation, reliability, and robustness analysis, decision-making under uncertainty, quality-assurance, and model reduction. The sensitivity analysis studies were conducted utilizing a Naval Research Laboratory (NRL) industry standard SVP model. This model provides a route optimizing engine based on a Ship Tracking and Routing System (STARS) software package. The overarching goal of this thesis was to assess the impact/sensitivity of ocean and atmospheric modeling, ensembles, and specific ship class inputs on ship route optimizer output improvements.

A basic description of environmental model input details and additional SVP model inputs follows. Synoptic model input environmental data fields are obtained from Navy Operational Global Atmospheric Prediction System (NOGAPS), WAVEWATCH III (WW3), and Navy Operational Global Ocean Model (NCOM). Environmental model grids are generated once per synoptic (12-hour) cycle. The SVP model also uses climatology for input after the 240 hour TAU mark. Environmental climatology data fields were developed by the Fleet Numerical Meteorology and Oceanography Center (FNMOC) models department and use United States National Centers for Environmental Prediction (NCEP) re-analysis wind fields from the last 15 years. The FNMOC WW3

model is used to generate sea and swell data from the NCEP winds. Baseline forecast fields are saved at 6-hour intervals out to 240 hours. For tropical storms, wind radii data is parsed from tropical cyclone warnings. Bathymetry is extracted from Digital Bathymetric Data Base Variable Resolution (DBDBV) 2-minute resolution bathymetry, provided by Naval Research Laboratory, Stennis Space Station, Mississippi.

Commercial SVP packages make use of exact hull form, ship loading, and power curves. For this study, specific ship platform data and propulsion types are also used that model exact hull form, ship loading and power curves. SVP model sensitivity analysis was conducted utilizing these platform characteristics to determine the importance of these parameters for output routes. Also reviewed were model accuracy sensitivities and the effects various ensemble post-processing techniques.

Key research questions investigated during the SVP model experiments included:

- What were the conditions or requirements that impact the quality of input values?
- Which input values carry the highest sensitivity for the SVP system? What is the rank ordering of the sensitivity?
- What are the points of diminishing return? What accuracy is good enough?
- How should uncertainty and sensitivity be communicated? What effects does it have on routing improvements or degradation?
- With enhanced model output, what is the improvement in the least cost route as compared to the best possible route using actual analysis environmental data in SVP models?
- What effect do platform characteristics have on the outcome of the least cost route? (e.g. accurate hull modeling, platform loading, power curves)
- What additional incremental efforts are required to improve model outputs by x%?

This study evaluated opportunities for next generation modeling efforts by linking improved outputs to energy conservation/costs avoidance. By identifying the most important SVP model inputs, our limited resources can be directed towards improving critical environmental model outputs. Specific study benefits include:

- Possible fuel savings on the order of 5-8% (annual cost savings in the 10's of millions of dollars)
- Reduced CO2 emissions (contributing to the green fleet initiative)
- Enhanced model outputs to include greater accuracy and consistency
- Safer operation with improved severe weather avoidance and minimized loss of mission time
- Minimize vessel environmental related stress and maintenance requirements
- Validates importance of next-gen models and improved sensor capabilities
- Integrates and leverages the use of shore side state of the art models and shipboard Electronic Chart Display and Information System - Navy (ECDIS-N) navigation systems along with other related decision aids

Once specific environmental model outputs are identified, follow-on efforts can be made in improving air/ocean forecast skills and model outputs to enhance near term and medium term forecast skill. Methods such as ensembles and improved air/ocean coupled numerical models and/or increased resolution may be utilized. Use of assimilated forcing data from various sensors and climatology may also enhance model output and reduce variance. Utilizing targeted improved forecast weather, ocean wave and current accuracy, SVP models will ensure a least cost track, thereby maximizing fuel savings while sailing safe routes.

Sensitivity analysis areas of study are depicted in Figure 1. Specifically, weather, ocean wave, and current model outputs were used as inputs into the SVP model. 2010-2011 archived environmental data and realistic empirical data for vessel platform

characteristics were also used for the sensitivity studies. The following data fields were used for model input:

- Waves (period/swell/height)
- Winds
- Currents
- Platform hull forms
- Ship power curves and plant L/U
- Ship loading characteristics

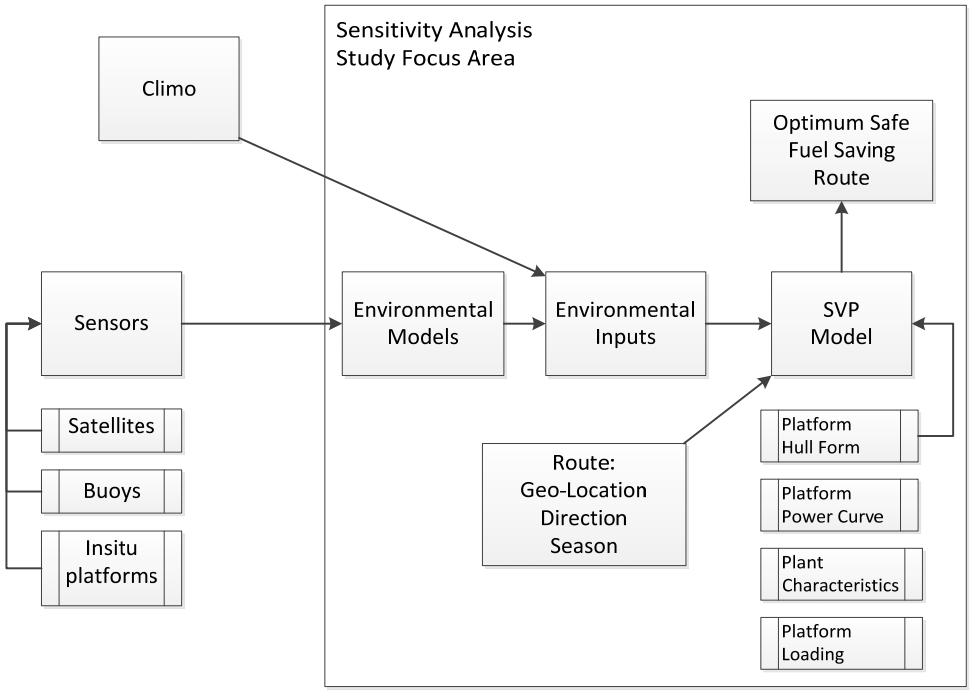


Figure 1. SVP model construct & thesis focus area

Sensitivity analysis was conducted at Naval Research Laboratory, Monterey. The sensitivity studies concentrated on SVP model output analysis of generated route costs with predicted environmental data sets compared with analysis environmental data sets in order to determine model skill variance effects. Results were correlated and analyzed to determine input variables that resulted in the highest sensitivity for SVP model outputs.

Custom computer scripting was written using the PYTHON programming language and Bourne-again Shell (BASH). This enabled sensitivity analysis of the complex SVP model outputs. Additional, computer scripting wrappers were utilized inside a UNIX operating system environment in order to support model input/output operations. A sensitivity analysis process diagram example is shown in Figure 2.

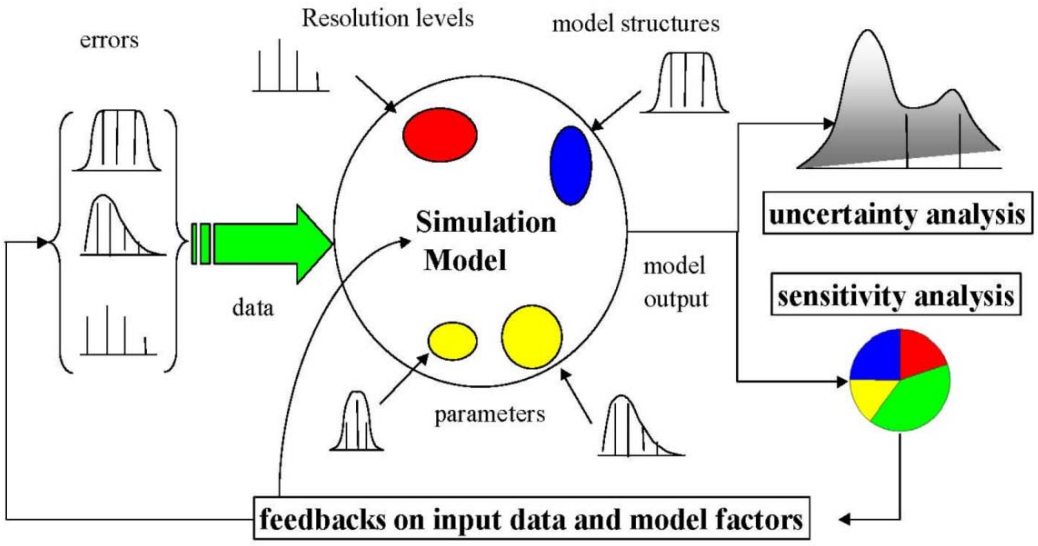


Figure 2. Sensitivity analyses process diagram (From Saltelli 2008)

The primary objective of the SVPDA model is to produce optimized ship routes and these optimized output tracks must meet the following goals:

- Find the most efficient (least power) path between two points. This route should be optimized over algorithms for finding the shortest route (e.g. great circle, rhumb line route, or coastal and direct inland) provided no safety re-route is required
- The most efficient route must meet proper safety constraints (bad weather avoidance/safe operating envelope)

A modified Dijkstra 3D linear programming open loop stochastic shortest path algorithm is used to solve the route problems. The SVP routing engine also incorporates numerical weather predictions into a shortest path network flow model in order to

provide optimal tracks around adverse weather while providing the best fuel efficiency. The network is constructed of nodes and arcs overlying discretized ocean grid points. Nodes represent latitude and longitude positions, while arcs connecting them represent navigable paths between nodes. Each node is populated with distance and time labels, wind and seas forecast predictions, and geographic accessibility. Arc costs are defined as the non-negative distances between nodes. Due to spatial grid resolution, it's expected that the SVP model will calculate the most efficient routes when calculating long (thousands of miles), open-water routes.

In summary, this thesis is a proof of concept to form the basis for sensitivity studies and possible additional testing in the future. It does not attempt to fully develop a fleet ready Optimum Track Ship Routing (OTSR) application, but rather investigates input parameter sensitivities compared to model output characteristics. In order to objectively assess the model, test cases using past NRL/FNMOC model and analysis data sets were used. The results of these tests indicated that the SVP model is very sensitive to various model inputs, but can optimize routes to save fuel, while maintaining the ship safe. The model avoids adverse weather and solves the least-time path to a destination. It calculates useful time, distance, and fuel consumption metrics in order to quantify routing decisions. A Concept of Operations (CONOPS) test at sea also demonstrated feasibility of the SVPDA model operational use on a naval combatant.

A. HISTORY

Development work at gaining greater efficiency in sea voyages in the area of data accumulation and climatology has a long history. Benjamin Franklin, as deputy postmaster general of the British Colonies in North America, produced a chart of the Gulf Stream from information supplied by masters of New England whaling ships. This first mapping of the Gulf Stream helped improve the mail packet service between the British Colonies and England. In some passages the sailing time was reduced by as much as 14 days over routes previously sailed. In the mid-19th century, Matthew Fontaine Maury compiled large amounts of atmospheric and oceanographic data from ships' log books. For the first time, climatology of ocean weather and currents of the world was available

to the mariner. This information was used by Maury to develop seasonally recommended routes for sailing ships and early steam powered vessels in the latter half of the 19th century. In many cases, Maury's charts were proved correct by the savings in transit time. Average transit time on the New York to California via Cape Horn route was reduced from 183 days to 139 days with the use of his recommended seasonal routes. In the 1950's the concept of ship weather routing was put into operation by several private meteorological groups and by the U.S. Navy (NIMA 2002).

By applying the available surface and upper air forecasts to transoceanic shipping, it was possible to effectively avoid heavy weather while generally sailing shorter routes than previously. The development of computers, the internet and communications technology has made weather routing available to nearly everyone afloat. Criteria for route selection reflect a balance between the captain's desired levels of speed, safety, comfort, and consideration of operations such as fleet maneuvers, fishing, towing, etc. Ship weather routing services are currently being offered by many nations. These nations include Japan, United Kingdom, Russia, Netherlands, Germany, and the United States. Several private firms also provide routing services to shipping industry clients. Additionally, numerous PC-based software applications have become available.

B. ATMOSPHERIC AND OCEANIC PHYSICAL PROCESSES

An understanding of the earth's global atmospheric and oceanic physical processes that affect ships at sea is important. With this knowledge, we can attempt to leverage natural phenomena in both the ocean and atmosphere to try and gain both efficiency and safety during open ocean transits.

1. Winds

The region of Earth receiving the Sun's direct rays is the equator. Here, air is heated and rises, leaving low pressure areas behind. Moving to about thirty degrees north and south of the equator, the warm air from the equator begins to cool and sink. Between thirty degrees latitude and the equator, most of the cooling sinking air moves back to the equator. The rest of the air flows toward the poles. The air movements toward the

equator are called trade winds where warm, steady breezes blow almost continuously. The Coriolis Effect then makes the trade winds appear to be curving to the west, whether they are traveling to the equator from the south or north. The trade winds coming from the south and the north meet near the equator. These converging trade winds produce general upward winds as they are heated, so there are no steady surface winds. This area of calm in the ocean is called the doldrums. Between thirty and sixty degrees latitude, the winds that move toward the poles appear to curve to the east. Because winds are named from the direction that they originate, these winds are called prevailing westerlies. Prevailing westerlies in the Northern Hemisphere are responsible for many of the weather movements across the United States and Canada. At about sixty degrees latitude in both hemispheres, the prevailing westerlies join with polar easterlies to reduce upward motion. The polar easterlies form when the atmosphere over the poles cools. This cool air then sinks and spreads over the surface. As the air flows away from the poles, it is turned to the west by the Coriolis Effect. Again, because these winds begin in the east, they are called easterlies. The various global changes in wind direction are illustrated in Figure 3.

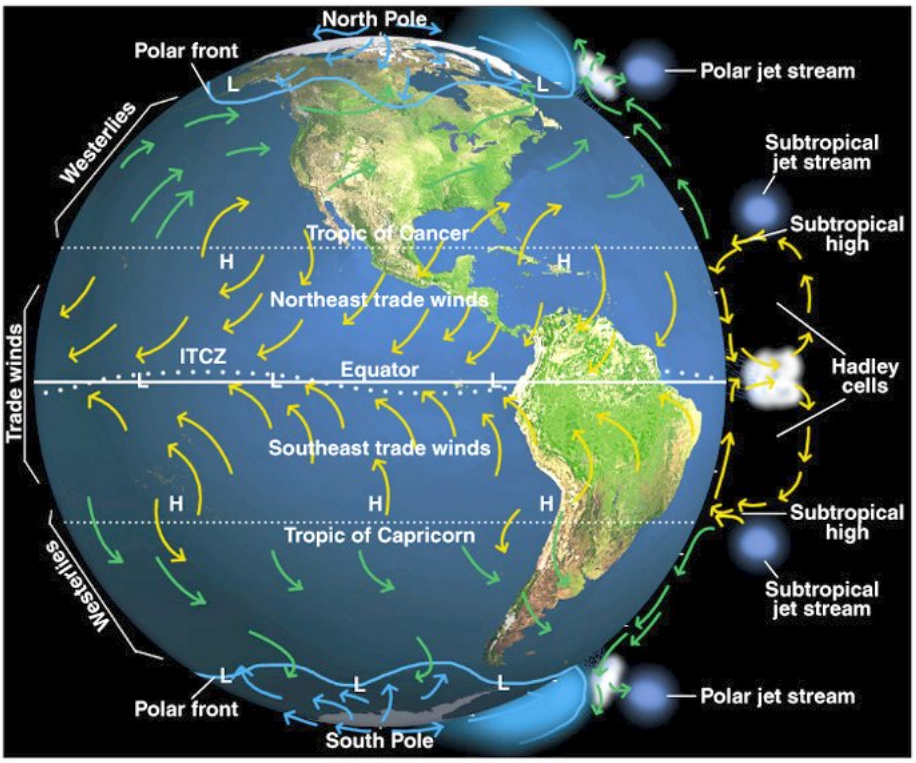


Figure 3. Global wind patterns (From Scripter 2012)

2. Waves

Wind waves are the waves at sea that are generated by local or distant winds. Waves generated locally are usually referred to as wind waves. Waves generated at distant locations in the past are referred to as swell. Wind waves range in wave height from negligible to 30m (100ft) or even higher, and in length (distance between consecutive waves) from centimeters to 1 km. Corresponding wave periods (i.e., the time it takes for two consecutive waves to pass a given location) range from less than 1 second to about 25s (Tolman 2000; Tolman 2007). Although wind wave conditions generally change slowly, no two consecutive waves are identical. Therefore the wave field is described with average measures for wave heights. Inside wave models, the wave field is not described with a single wave height, but with a wave spectrum, that describes the distribution of wave energy over wave directions and frequencies at a fixed location. The wind-driven wave regime occurs most frequently in the Southern Ocean, the Northern Hemisphere storm tracks, and in enclosed seas. In the Southern Ocean and Northern Hemisphere storm tracks, where wind speeds are high, the wind-driven wave regime occurs more than 10% of the time. The presence of fast-traveling swell combined with low wind speeds in the tropics and subtropics, between 40N and 40S, results in the wind-driven wave regime occurring less than 5% of the time. The exception is the northeast Indian Ocean, where the wind-driven wave regime occurs more than 20% of the time. The frequency of occurrence of wind-driven waves averaged from 1958–2001 is illustrated in Figure 4.

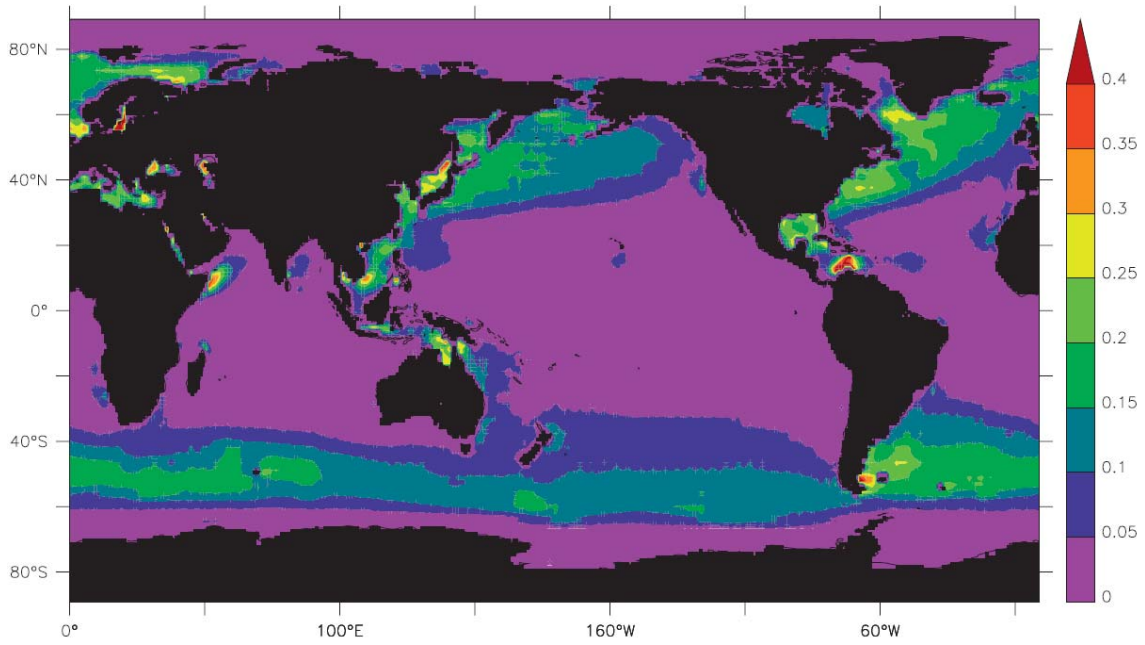


Figure 4. Wind-driven waves frequency of occurrence, calculated using the ERA-40 wind and wave data averaged over 1958–2001 (From Hanley 2010)

3. Currents

Ocean current predictions can now also be made, which plays a valuable role in predicting physical surface ocean features, such as eddies and western boundary currents. When we compare maps of wind and ocean currents at a global scale we see that they share most of the same features. The similarity arises from the fact that wind is the fundamental driver of surface ocean currents. A comparison of global ocean currents and surface winds is identified in Figure 5.

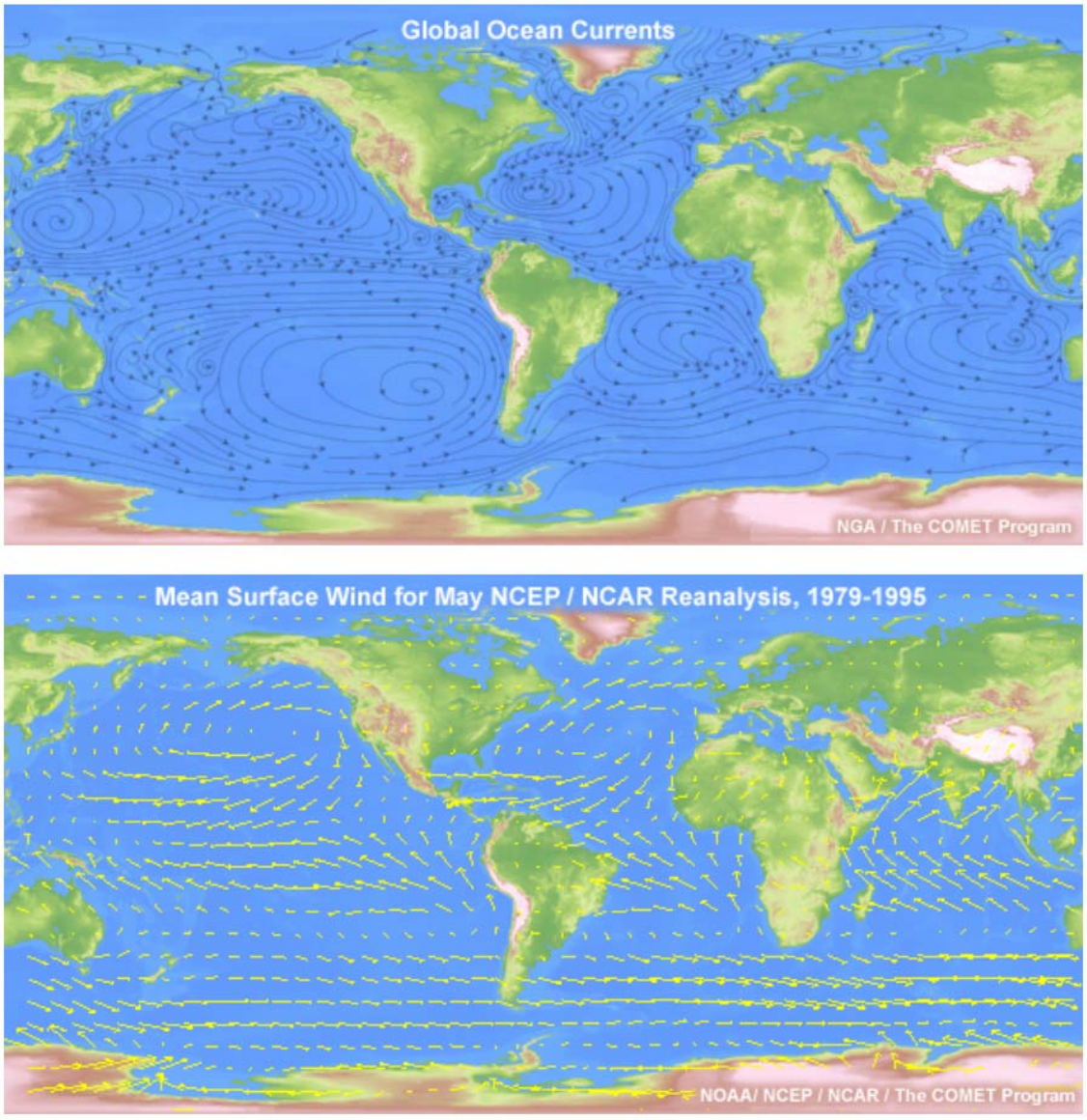


Figure 5. Global ocean currents and surface winds comparison (From UCAR 2012)

The global pattern of winds together with the Coriolis Effect and Ekman Transport produce large-scale currents in the global ocean environment. Note that gyres circulate clockwise in the northern Hemisphere and counter-clockwise in the Southern Hemisphere. These Gyres are characterized by circulation at the scale of the ocean basin. The influence of the Coriolis Effect on ocean currents increases with increasing latitude, so the equatorial currents are similar in each ocean basin, although their flow direction (east to west) is consistent with the sense of flow in the large-scale gyres within each

ocean basin. The variation of Coriolis forcing as a function of latitude has a pronounced effect on surface currents. Poleward currents on the Western side of each ocean basin are distinctly different from those on the Eastern side of the ocean basin. Western Boundary Currents are swift, narrow and deep relative to Eastern Boundary Currents, that are slower, broader and shallower than WBC's. Figure 6 identifies the locations and names of these global ocean features (Arthur 2011).

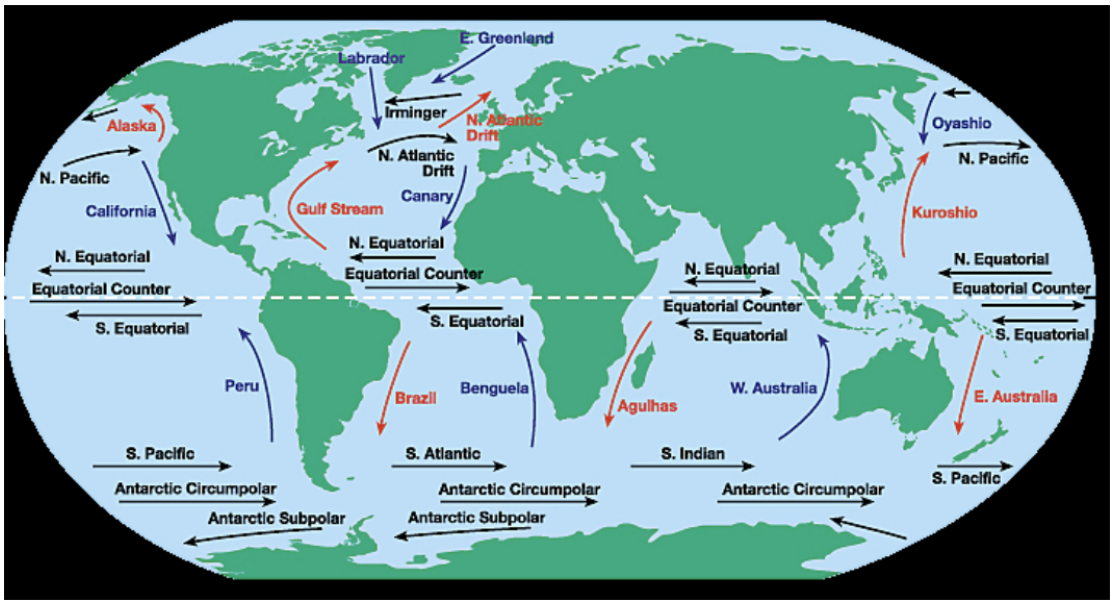


Figure 6. Global ocean gyres and currents (From Arthur 2011)

C. ENVIRONMENTAL EFFECTS ON SMART VOYAGE PLANNING

There are two very critical pieces that an Optimum Track Ship Routing (OTSR) algorithm needs to produce the safest and most fuel efficient route. The first is an accurate short and medium range weather prediction. The second is an accurate prediction of ship speed reacting to the wind/wave/current conditions to dead reckon the ship's accurate position. These two pieces are critical for the SVP model to calculate an optimum route since the predicted ship position must be time synchronized with the actual environmental weather (Chen 1998). Wind and waves are thought to be the key drivers with a +/- 15% effect on fuel usage. The atmospheric and ocean models used in this thesis are NOGAPS, WW3, and NCOM. These models are described in detail in the next chapter. The general effects of wind, waves, and current are described below.

1. Wind Speed and Direction

- First order effect (not distance)
- Non-linear
- Proportional to wind cubed above 35 kts (18 m/s) (wind causes ~1/3 of resistance below 35 kts, varies with relative direction)

2. Wave Amplitude, Period, and Direction

- First order effect (not distance)
- Positive or negative
- Critical when wavelength is ~ size of the ship
- Adds resistance and impacts propeller performance characteristics
- Drift and direction, especially in quartering seas

3. Currents

- +\- ~1/2 kt (.26 m/s) (straight forward effect)
- 1-2 kts (.51-1 m/s) in boundary currents (positive or negative)
- Varies with wind

D. STATUS OF OTSR

1. Commercial Routing Services

The ability to effectively advise ships to take advantage of favorable weather was previously hindered by forecast limitations and the lack of an effective communications system. However, the advent of extended range forecasting and the development of selective climatology, along with powerful computer modeling techniques, have made ship routing systems possible. The improvements in both atmospheric and ocean long range forecasting skill along with constant improvements in computing horsepower have

also made smart voyage planning decision aid tools practical. At present, satellite communications and computer web support make this service available to just about every vessel in near real time. Weather routing services come in the form of several recommendations and advisories. These include an initial route recommendation, surveillance, advisory, and diversion. Routing services are used to minimize weather impacts on shipping in order to maximize fuel savings, reduce damage to ship and cargo, and increase safety for crew. These metrics define optimality in ship routing. Currently, there are two general types of commercial routing services available. The first uses techniques similar to the Navy's OTSR system to forecast conditions and compute routing recommendations, that are then broadcast to the vessel. The second assembles and processes weather and sea condition data and transmits this to ships at sea for on-board processing and generation of route recommendations. The former system allows for greater computer power to be applied to the routing task because powerful computers are available ashore. The latter system allows greater flexibility to the ship's master or Commanding Officer in changing parameters, evaluating various scenarios, selecting routes, and displaying data (Chen 1998).

2. U.S. Navy OTSR

The United States Navy Meteorology and Oceanography (METOC) community is responsible for providing timely and accurate weather routing recommendations to U.S. Department of Defense (DoD) ships. With billion-dollar combatants, expensive cargos, and rising energy costs, every effort must be made to avoid or reduce the effects of adverse weather and sea states. Such conditions can cause damage, unnecessary stress, severe speed reductions, and lost time or money. However, with these no compromise constraints, there is also now a strong demand to save fuel. With global oil prices hovering around $100 a barrel, the US Navy's annual ship fuel costs now exceed 1 billion dollars a year among combatants and Military Sealift Command (MSC) ships.

The METOC community maintains ships outside of adverse weather by providing a routing advisory service called OTSR (CNMOC 2011). The CONOPS of current weather routing consists of Navy METOC Officers and Aerographer Mates (AG)

formulating optimum ship routes based on climatology, numerical weather forecasts, satellite products, and the individual ship's sailing capabilities. Commander, Naval Meteorology and Oceanography Command (CNMOC) is responsible for meeting the Navy's OTSR requirement. The METOC community maintains a policy that the safe operation of maritime forces through the avoidance of severe weather is vital. With limited resources challenging the routing services and rising costs impacting naval operations, optimal routing will become increasingly more important. Although the METOC community performs OTSR effectively, there are opportunities to automate the process and add tools that can quickly develop routes that may be more efficient. Manual methods used to formulate diversion are adequate to provide ships safe routes, but, if several optimal tracks based on shortest path, least time, least cost, or any combination of these factors can be analyzed, time and fuel savings may be possible at no additional risk to the vessel. The US Navy already produces high resolution environmental model outputs necessary for input into a Smart Voyage Planning Decision Aid (SVPDA). Fleet Numerical Meteorology and Oceanography Center (FNMOC) generates and distributes these numerical weather prediction models, satellite imagery, and other weather related products and services. Fleet Weather Centers (FWCs) currently utilize decision aids such as the Joint METOC Viewer (JMV) to develop safe routes. However, a tool such as SVPDA, that generates tracks based various inputs, can offer routing personnel the ability to analyze parameter sensitivities and alternative tracks in more detail than is currently possible with current methods.

This thesis identifies the advantages that an SVP decision aid can provide the fleet by using time-tested proven algorithms for solving shortest paths in conjunction with simple ship response functions to the environment. Some of the sensitivity analysis tools that I developed for this research may provide examples of additional possibilities.

E. OTSR CONOPS

Routing activities within the naval METOC community use several forms of formatted messages in order to perform OTSR effectively. This information is sent to the

routing activity from the requesting unit using the U.S. Navy Movement Report (MOVEREP) system (DON 1997) and the OTSR/Route Surveillance Request (CNMOC 2011).

The purpose of the movement report system is to gather and disseminate current location information to pertinent elements of the operational and administrative chain of commands regarding all U.S. Navy, Coast Guard, and MSC ships. Movement information is sent via a movement report message. This message contains information including: ship class, point of departure, estimated time of departure, position of intended movement track and times, destination and time of arrival, intended and maximum speed of advance, and highest operational limits for head, beam, and following seas, and wind velocities. Units requesting OTSR provide this information to the routing activity at least 72 hours prior to departure (CNMOC 2011). While sending just a movement report is acceptable, the unit requesting OTSR services may also send a Route Surveillance Request. Information contained in this message is more detailed, giving the routing activity better insight to the weather sensitivities and seaworthiness of the vessel. This information includes: loading characteristics, type of cargo loaded, deck loaded aircraft, material condition of ship, which may affect seaworthiness, and operations scheduled during a route.

The routing activity analyzes the movement report and surveillance request information along with climatology, winds and seas forecasts, satellite data, and other METOC products in order to make an initial route recommendation that is provided to the requesting unit 36 to 48 hours prior to departure. This recommendation provides the requesting unit with a route that minimizes the possibility of adverse weather during its transit. If route surveillance was requested, this is also the surveillance commencement, where the requesting unit's progress is monitored for adverse weather avoidance and ends when the requesting unit reaches its destination (CNMOC 2011).

Routing activities closely monitor elements of the ocean and atmosphere that may cause reduced speed, increased fuel consumption, ship damage, or personnel injury. Ocean and atmospheric elements of most concern are winds and seas. Ice and currents are considered under special transit circumstances, but rarely influence routing decisions.

Winds and seas are the most important because a route can be considered optimal if the effects of winds and seas can be minimized. If wind and sea conditions are expected to exceed the ship's specified limits at the time of its departure, a delay in port recommendation will be made. Routing activities will adjust the departure time at this phase of the surveillance to avoid potentially hazardous weather as there are generally very few routing options close to coastal regions (Montes 2005).

For surveillance, once underway, the requesting unit will transmit a daily OTSR report to the routing activity. This report includes the vessel's position, course, speed, weather observation, and current time of arrival. This information is crucial for the routing activity to properly monitor the vessel's transit progress. If the routing activity concludes that forecasted wind and sea conditions will exceed the ship's limit values during the surveillance, a route diversion will be recommended. A route diversion is an adjustment to the original transit track to avoid potentially hazardous weather and sea conditions forecasted to be encountered. While the divert track generally adds distance to the original track, this added cost is less expensive than the speed reduction, safety hazards, and added fuel consumption likely to occur while transiting adverse weather. However, certain weather systems may cause vessels to encounter a situation where time and maximum speed constraints do not allow a feasible divert solution. Concern for vessel safety and ship handling now become the primary consideration over efficient timely transit. Therefore, routing activities may recommend an adjustment in transit speed, where the vessel will increase or decrease speed in order to avoid the adverse weather with no change in track, or an evasion position in a general direction away from the adverse weather where the vessel can "wait out the storm." The routing activity will recommend a course and speed to regain original track once the hazardous weather conditions have abated. OTSR services terminate once the requesting unit has transmitted its arrival notice upon entering port. Since safety is paramount in these cases, there is little thought or capability to try to ensure the most efficient use of fuel (NIMA 2002).

To prove operational suitability for the SVPDA tool, a proof of concept test was conducted at sea, and identified CONOPS that could be used if SVPDA were implemented navy wide. Details and results of this test are discussed in Chapter IX.

II. NAVY ENVIRONMENTAL MODELS AND DATA

A. NOGAPS

The Navy Operational Global Atmospheric Prediction System (NOGAPS) is the Department of Defense's (DoD's) high resolution global numerical weather prediction (NWP) system model. Its development and operation is a joint activity of the Naval Research Laboratory (NRL) and the Navy's Fleet Numerical Meteorology and Oceanography Center (FNMOC). The NOGAPS forecast model is a global model that is spectral in the horizontal and energy-conserving finite difference (sigma coordinate) in the vertical. The model top pressure is set at 0.04 hPa; however, the first velocity and temperature level is approximately 0.07 hPa. The variables used in dynamic formulations are vorticity and divergence, virtual potential temperature, specific humidity, surface pressure, skin temperature, and ground wetness. NOGAPS is also a primary tropical cyclone forecast tool for forecasters at the Joint Typhoon Warning Center (JTWC). For its data assimilation, NOGAPS adopted the NRL Atmospheric Variational Data Assimilation System-Accelerated Representer (NAVDAS-AR) in September 2009. This four-dimensional variational (4DVAR) analysis scheme replaced the 3DVAR system first implemented operationally at FNMOC in 2003. The analysis is performed on the Gaussian grid of the T319L42 global spectral model. The corresponding horizontal resolution is approximately 70 km. Besides using conventional observations (surface, rawinsonde, pibal, and aircraft), the analysis makes heavy use of various forms of satellite-derived observations. The analysis employs both direct radiance (brightness temperature) and derived soundings from National Oceanic and Atmospheric Administration (NOAA) and Defense Meteorological Satellite Program (DMSP) polar-orbiting satellite instruments (AMSUA, AIRS, IASI, SSMI/S, MHS). Additional soundings are derived via GPS-radio occultation measurements. Surface marine wind speeds are assimilated using several different scatterometers (ASCAT, ERS-2, WindSat, SSMI) while winds aloft are estimated from atmospheric motion vector (AMV) measurements using water vapor, infrared, and visible satellite imagery (Geostationary, MODIS, AVHRR, and LEO/GEO) (UCAR 2012).

The physics package includes:

- Bulk-Richardson number-dependent vertical mixing patterned after European Center for Medium Range Weather Forecasts' (ECMWF) vertical mixing parameterization
- A time-implicit Louis surface flux parameterization
- Gravity wave drag
- Shallow cumulus mixing of moisture, temperature, and winds
- Emanuel cumulus parameterization
- Convective and stratiform cloud parameterization
- Harshvardhan solar and longwave radiation
- Semi-implicit treatment of gravity wave propagation and Robert time filtering

NOGAPS includes components for data quality control, data assimilation, model initialization, and model forecasts. While the nature of the components has changed considerably over the years, NOGAPS has been and remains the central engine that is the heart of the Navy's environmental prediction capability. NOGAPS provides forcing fields for mesoscale weather prediction; tropical cyclone prediction; aerosol prediction; ocean, wave, and ice prediction; and aircraft and ship routing applications. NOGAPS also forms the backbone of the Navy's ensemble prediction system, which provides global forecasts out to 10 days.

A common metric used by operational forecast centers to measure global model forecast skill is the anomaly correlation (a measure of skill of 500-hPa geopotential height forecasts); a correlation of 0.6 typically represents a forecast that has some practical skill. The NOGAPS model annual skill improvements are illustrated in Figure 7. While some years are clearly more predictable than others, the NOGAPS forecasts show an overall upward trend in skill at all forecast times, with the most dramatic improvements seen in the 5-day forecasts (NRLMRY 2012).

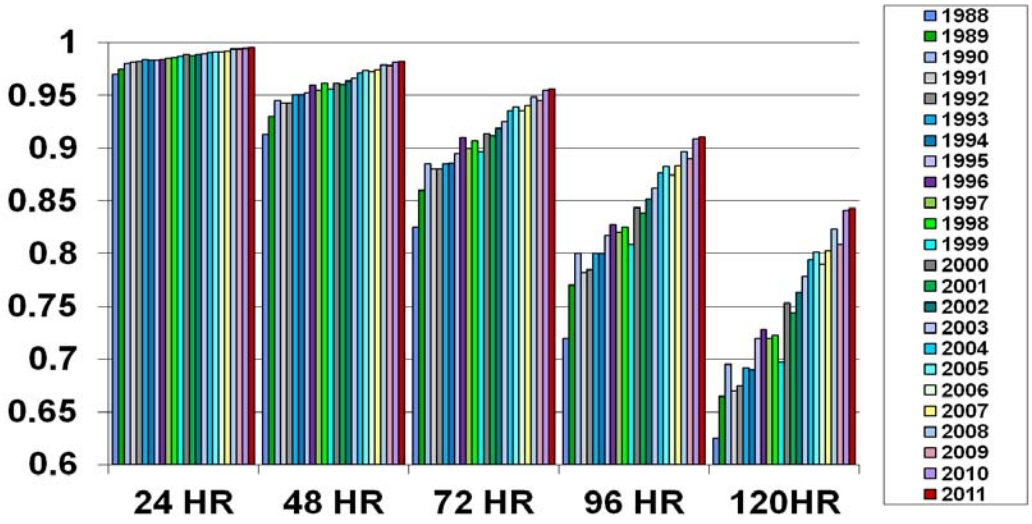


Figure 7. History of NOGAPS skill, northern hemisphere 500 mb heights anomaly correlation (From NRLMRY 2012)

The following are the SVPDA model input parameters from NOGAPS:

- Parameters: Wind speed and direction - calculated from wind u/v components at 10M height above the surface. Units are m/sec.
- Frequency of Generation: Grids are generated for 00, and 12Z forecast cycles.
- Number of Forecasts per cycle: The wind u/v components are available at 3 hour forecast intervals, but only extracted at 6 hour forecast intervals. Forecasts are available through tau 240 (10 days).
- Size: 2 parameters x 2 cycles x 31 forecasts/cycle x 1.04 megabytes = 129 megabytes.
- Resolution: Data is available at 1 degree and 1/2 degree resolution. The 1/2 degree fields are used.

B. WAVEWATCH III

WAVEWATCH III (NOAA 2009) is a third generation wave model developed at National Oceanic and Atmospheric Administration (NOAA)/NCEP in the spirit of the

Wave Modeling Project (WAM) model (WAMDIG 1988; Komen 1994). It's a further development of the model WAVEWATCH, as developed at Delft University of Technology (Tolman 1989) and WAVEWATCH II, developed at NASA, Goddard Space Flight Center (Tolman 1992). WAVEWATCH III, however, differs from its predecessors in many important points such as the governing equations, the model structure, the numerical methods and the physical parameterizations. Furthermore, with model version 3.14, WAVEWATCH III evolved from a wave model into a wave modeling framework, which allows for easy development of additional physical and numerical approaches to wave modeling.

WAVEWATCH III solves the random phase spectral action density balance equation for wavenumber-direction spectra. The implicit assumption of this equation is that properties of medium (water depth and current) as well as the wave field itself vary on time and space scales that are much larger than the variation scales of a single wave. With version 3.14 some source term options for extremely shallow water (surf zone) have been included, as well as wetting and drying of grid points. Whereas the surf-zone physics implemented so far are still fairly rudimentary, it does imply that the wave model can now be applied to arbitrary shallow water.

Wave energy spectra are discretized using a constant directional increment (covering all directions), and a spatially varying wavenumber grid. The latter grid corresponds to an invariant logarithmic intrinsic frequency grid (Tolman 1998). Both a first order accurate and third order accurate numerical scheme is available to describe wave propagation (NWS 1995). The propagation scheme is selected at the compile level. The source terms are integrated in time using a dynamically adjusted time stepping algorithm, which concentrates computational efforts in conditions with rapid spectral changes (NOAA 2010). Figure 8 identifies WW3 model output depicting Pacific Ocean Significant Wave Height (SWH) in feet and directional vectors for 21 Nov. 2009.

WW3 uses statistical properties of waves to predict the sea state at a point, rather than trying to predict individual waves. The full sea state at any point over the ocean consists of the overlaying (or more technically, the superposition) of waves with different characteristics (wavelength and amplitude) arriving from all directions. Both the global

and regional WAVEWATCH III models predicts the energy spectrum of these waves over a range of discrete frequencies and directions, using what is known as a wave action density equation (UCAR 2012). The equation relates the frequency and direction of ocean wave energy to the sources and sinks of wave energy, (E). This equation can be written as follows:

$$\frac{\partial E}{\partial t} + \mathbf{c}_g \bullet \nabla E = S_{in} + S_{nl} + S_{ds} + S_{bot}$$

where

- S_{in} is the wind-wave interaction term, and represents the development and destruction of waves by the wind (can be source or sink, depending on wave direction relative to the wind)
- S_{nl} is a non-linear wave-wave interaction term, and represents the change in wave energy at different frequencies and moving in different directions, resulting from the interaction of waves of different frequencies and directions (can be a source or sink at a given frequency)
- S_{ds} is a dissipation or 'whitecapping' term, similar to the breaking of gravity waves in the atmosphere (a sink)
- S_{bot} is a wave-ocean bottom interaction term (for shallow water only), similar to friction (a sink)

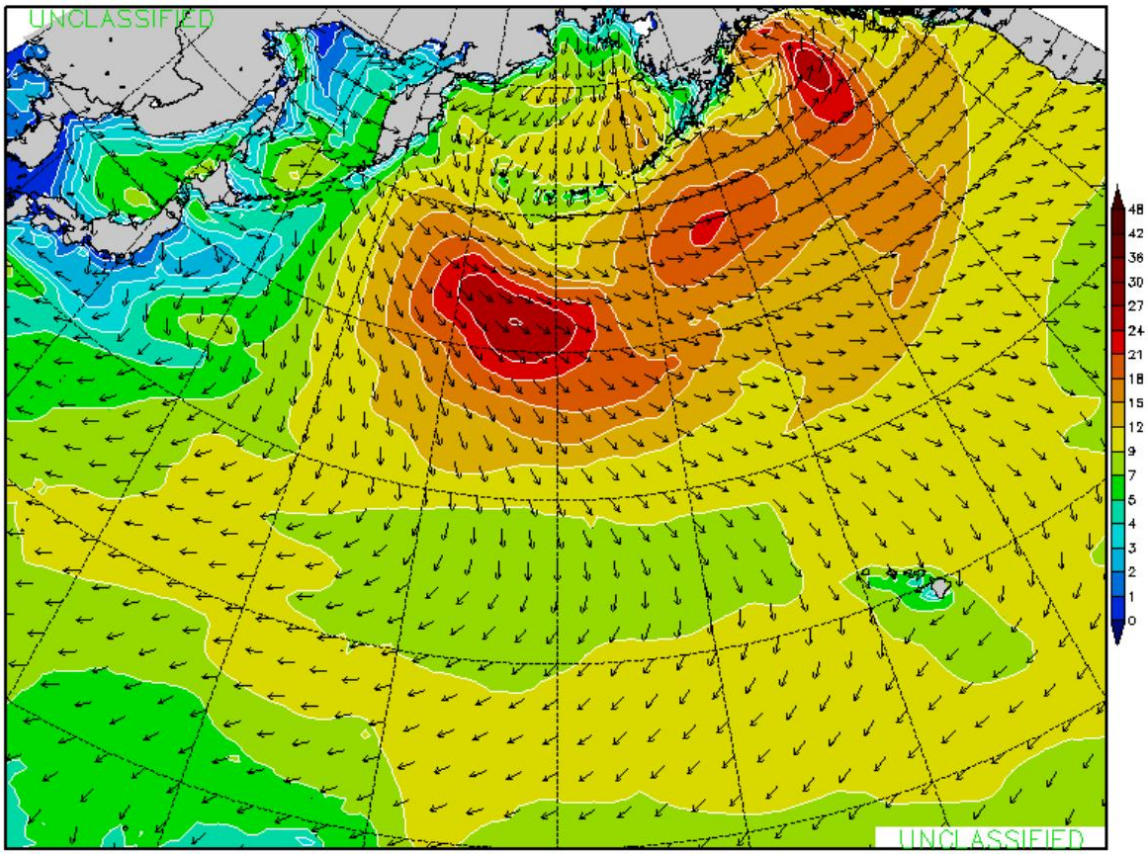


Figure 8. WW3 model output depicting Pacific Ocean Significant Wave Height (SWH) in feet and directional vectors for 21 Nov. 2009 (From NOAA 2010)

The following are the SVPDA model input parameters from WW3:

- Parameters: Sea Height (m), Period (s), Direction (deg) and Swell Height (m), Period (s), Direction (deg). Note that global data is generated using NOGAPS winds while data in the Mediterranean and other small water bodies is generated from COAMPS regional model winds.
- Frequency of Generation: Grids are available for 00Z and 12Z forecast cycles.
- Number of Forecasts per cycle: Fields are available at 6 hour forecast tau intervals through tau 240 (10 days).
- Size: 6 parameters x 2 cycles x 31 forecasts/cycle x 1.04 megabytes = 387 megabytes.

- Resolution: Data is available at 1 degree and 1/2 degree resolution. The 1/2 degree fields are used.

C. GLOBAL NCOM

The U.S. Navy Operational Global Ocean Model (NCOM) is a 3-dimensional current model data, developed by the Naval Research Laboratory (Barron 2004; Barron 2006) and maintained by the Naval Oceanographic Office, and is used as the basis for the Ocean Prediction Center (OPC) Global Ocean SST & Currents Forecast. NCOM is a free-surface, primitive-equation model based primarily on two other models; the Princeton Ocean Model and the Sigma/Z-level Model (Martin 2000). In its global configuration, NCOM implements a curvilinear horizontal grid designed to maintain a grid-cell horizontal aspect ratio near (Rhodes 2002). Horizontal resolution varies from 19.5 km near the equator to 8 km or finer in the Arctic, with mid-latitude resolution of about 1/8° latitude (~14 km). Horizontal resolution has been sacrificed to allow increased vertical resolution. To improve the detail of upper-ocean dynamics, a maximum 1-m upper level thickness in a hybrid sigma/z vertical configuration with 19 terrain-following sigma-levels in the upper 137 m over 21 fixed-thickness z-levels extending to a maximum depth of 5500 m is used. Model depth and coastline are based on a global 2-minute bathymetry produced at the Naval Research Laboratory (NRL). The present daily model run consists of a 72-hour hindcast to assimilate fields that include recent observations, and a 72-hour forecast. Longer forecasts are currently being evaluated, but are not yet operational (NOAA OPC 2011). Global NCOM uses atmospheric forcing from the Navy Operational Global Atmospheric Prediction System, with latent and sensible heat fluxes calculated internally using NCOM SST. Data assimilation is based on global profiles of temperature and salinity derived using operational sea-surface fields and in situ data within the Modular Ocean Data Assimilation System (MODAS) (Fox 2002). Figure 9 depicts how Global NCOM assimilates satellite measurements of sea surface temperature and elevation to produce forecasts of temperature, salinity, elevation and currents that support Navy operations. The only NCOM environmental model input into the SVPDA model is ocean currents.

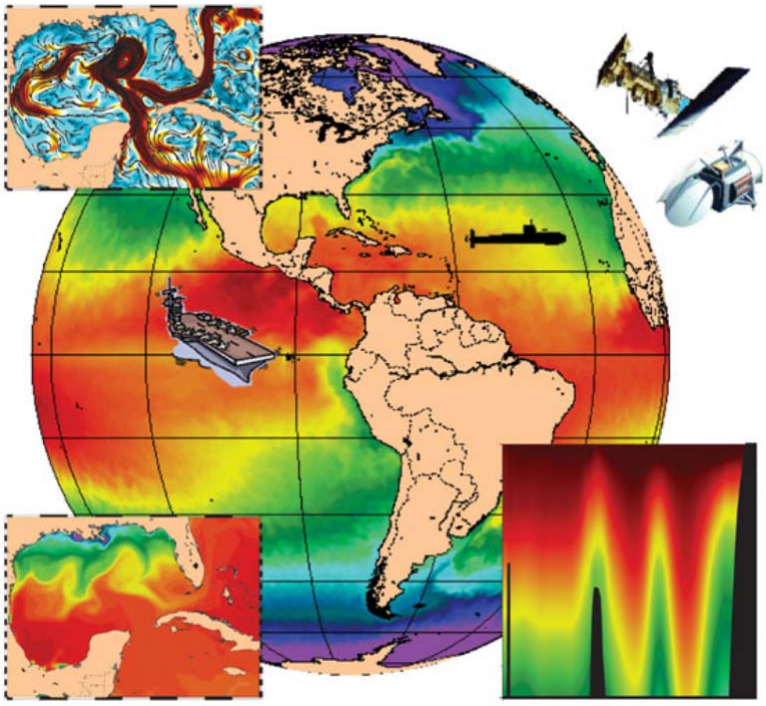


Figure 9. Global NCOM assimilates satellite measurements of sea surface temp. and elevation to produce forecasts of temperature, salinity, elevation and currents that support Navy operations (From NRL Stennis 2006)

The following are the SVPDA model input parameters from NCOM:

- Parameters: Current Speed and direction - calculated from current u/v components. Units are in m/sec.
- Frequency of Generation: Grids are available once per day for 00Z forecast cycle.
- Number of Forecasts per cycle: The current u/v components are available at 6 hour forecast tau intervals. Forecasts are available through tau 72 (3 days).
- Size: 2 parameters x 1 cycle x 13 forecasts/cycle x 1.04 megabytes = 27 megabytes.
- Resolution: Data is available at 1 degree and 1/2 degree resolution. The 1/2 degree fields are used.

D. CLIMATOLOGICAL DATA

Daily climatology files are available at four synoptic times per day for winds and seas. This climatology was generated at FNMOC by downloading NCEP re-analysis wind fields at 1-degree resolution at 4 times per day frequency for the last 15 years. The WW3 Hindcast program was run on each data set to generate the corresponding seas. The years were averaged on the synoptic times for both winds and seas to create the climatology dataset. The parameters are the same as for the NOGAPS winds and WW3 sea/swell data.

E. BATHYMETRY

The OTSR bathymetry data was extracted from the Oceanographic and Atmospheric Master Library (OAML) DBDB-V 2-minute bathymetry provided by the Naval Research Laboratory, Stennis Space Center, MS. The Application Programming Interface (API) provided with the database was used to break the data into 45-degree squares. The data points for the 45-degree squares were then converted to land/sea values based on a 12-meter depth being the cutoff for navigable waters. As depicted in Figure 10, the 1-digit character codes are: 0=open ocean, 1=shallow (< 12 m), 2=land, 3=missing.

Each value indicates the shallowest depth value for the entire square. This leads to many locations showing as land or shallow water that are actually navigable.

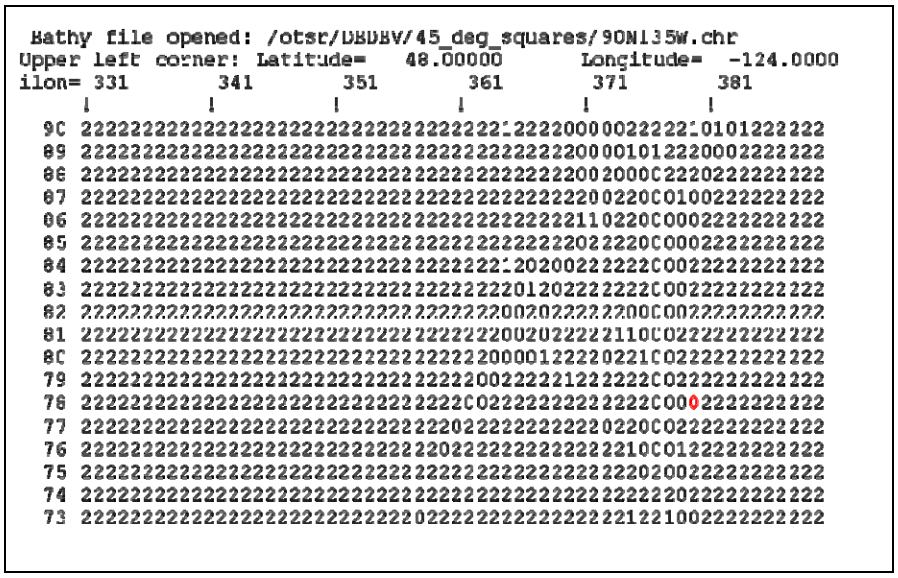


Figure 10. 1-digit character codes signifying bathymetry near Seattle, WA

F. TROPICAL CYCLONE TRACKS

The Tropical Cyclone Warning (.tcw) files from METCAST/JMV are used as input for the Tropical Cyclone Tracks. These files contain alerts and warnings from Defense Message System (DMS) message traffic of types Automated Tropical Cyclone Forecasting System (ATCF) and Rawinsonde Observation (RAOB). The formats used are Navy formats Automated Tropical Cyclone Forecasting System (ATCF), ATCG, ATCN and ATCP. FNMOC scripts that process the TCW files for situational awareness products also parse the lines from each TCW entry that define the storm radii and save them in a format that is easily readable by the Ship Tracking and Routing System (STARS) software.

G. TROPICAL CYCLONE RADIUS

The radius information is extracted from TCW files during processing of data for the situational awareness products. Radius information is stored in the /otsr/storms/[dtg] directory in files with the same names as the .tcw file but with a .wrn extension. Because the current set of fields may be in use while the next set is being constructed, a file is

created in the /otsr/storms base directory called current_dtg, containing the name of the current dtg directory name to use for tropical cyclone processing.

THIS PAGE INTENTIONALLY LEFT BLANK

III. SHIP ROUTE ENGINE

A. ROUTE ENGINE DESIGN

The Ship Tracking and Routing System (STARS) is a ship route optimization suite of software provided for research by FNMOC and modified for use in the NRL, Monterey computing environment. The model outputs an optimum route that is defined as the route that completes the voyage within time limits and with the least amount of fuel expended while keeping the vessel from exceeding wind and sea limits specified by competent authority. STARS utilizes a Dijkstra exhaustive search algorithm for minimum cost as follows:

- Creates a 3-D point grid (latitude, longitude, time to point) based on user-specified departure/arrival locations and times and min/max ship speed
- Grid aligns with the Great Circle route and internally defined grid spacing or utilizes manually inputted upper and lower boundary points
- Exhaustively searches all feasible routes (i.e. all possible forward-traversing connections between grid nodes that meet the constraints)
- At each geographic grid location a range of times to location are considered examining both the minimum and maximum ship speeds from all points at the previous stage

Note: Manually inputted upper and lower boundary points were used for this research with a bounded grid resolution of 300x60 nautical miles (the first value is the spacing along the direction of the route and the second value is the spacing across the route).

METOC inputs of winds and seas provide information to the algorithm that calculates the horsepower (HP) or fuel expended over a number of test routes. The algorithm then selects the optimized route (courses and speeds) to sail that will use the least (HP/fuel) and avoid weather limits that the user specifies. As previously mentioned in the model section, input parameters are: swell direction, swell height, swell period,

surface (10 meter) wind speed, surface (10 meter) wind direction, wind wave direction, wind wave height, wind wave period, surface current speed, and surface current direction. Vessel wind and sea limits used were 35 knots (18 m/s) and 12 foot (3.66 m), in accordance with US Navy standing Operational Order (OPORDER) requirements. The horsepower or fuel expended over the route is directly related to relative winds and seas that resist the forward advance of the ship and the distance/time that the engine runs. The greater the amplitude of the relative wind and relative seas, the greater the resistance and the longer the engine runs, and the greater the HP that is expended. In the relative wind and seas calculations, the ship's route (course and speed) are important. Wind, waves, and currents from aft-of-beam can actually reduce the resistance relative to calm water and can be used to assist in route optimization, thereby reducing fuel consumption. This type of situation is typically called "fair winds and following seas."

The SVPDA tool uses a cost function that is a nonlinear combination of the model parameters and depends on one or more independent variables. The ship route engine has the overall objective of minimizing fuel consumption within environmental and geographic constraints. The optimizer calls the cost evaluation function many times, so it must be computationally efficient. This incorporates a regression-based approach that is fast, but only approximately represented by:

f(Vship, wndwvht, wndwvper, uvwndwv, swlht, swlper, uvswl, uvcur, uvwind)

where

- f is the cost function (minimize cost within specified constraints)
- Vship is the cost derived from the ship's hull friction and powerplant fuel use (from CLASS.dat modeling data-store files)
- wndwvht is cost of the ocean wind wave height (from WW3)
- wndwvper is cost from the ocean wind wave period (from WW3)
- uvwndwv is cost from the ocean wind waves in the u and v directions (from WW3)

- swlht is cost from the ocean swell height (from WW3)
- swlper is cost from the ocean swell period (from WW3)
- uvswl is cost from the ocean swell in the u and v directions (from WW3)
- uvcur is cost from the ocean current in the u and v directions (from NCOM)
- uvwind is cost from 10 m winds in the u and v directions (from NOGAPS)

Figure 11 identifies the general flow of processing for the SVPDA system.

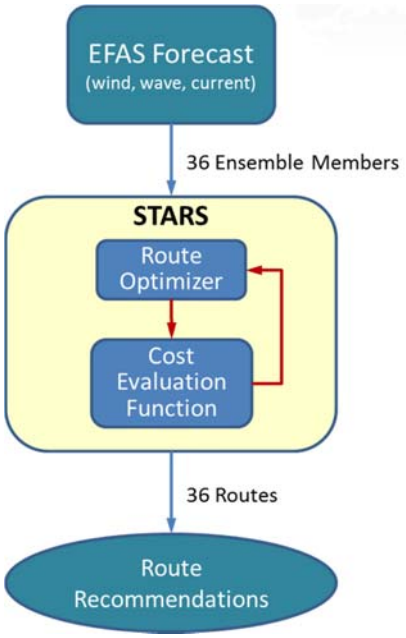


Figure 11. Graphical depiction of the basic routing engine process flow

1. Factors Affecting Fuel Consumption

- Drag due to waves (sea and swell)
- Added drag due to wind
- Effects of sea surface currents
- Hull/Propeller fouling condition

- Reduced propulsive efficiency
- Ships service loads
- Plant operation mode (e.g., full plant, trail shaft)
- Propeller pitch control system

2. Voyage Constraints

- Maximum allowable wave height in head, beam, and following seas
- Maximum allowable true and relative wind speeds
- Tropical cyclone avoidance limits for 35 & 50 kt (18 & 25.72 m/s) wind circles
- Land mass and shallow water avoidance
- Arrival time window

3. Cost Implementation Strategy

Long running calculations (e.g., added wave resistance) are cached in a data-store and interpolated at the routing engine run-time. Short running cost calculations are performed in-line. Estimated added resistance in waves is done using first order principle computational tools such as the U.S Navy Ship Motions Program (SMP) and Marintek's Vessel Responses (VERES) program. All above costs are then combined to determine a total fuel consumption rate in the cost evaluator as identified in Figure 12.

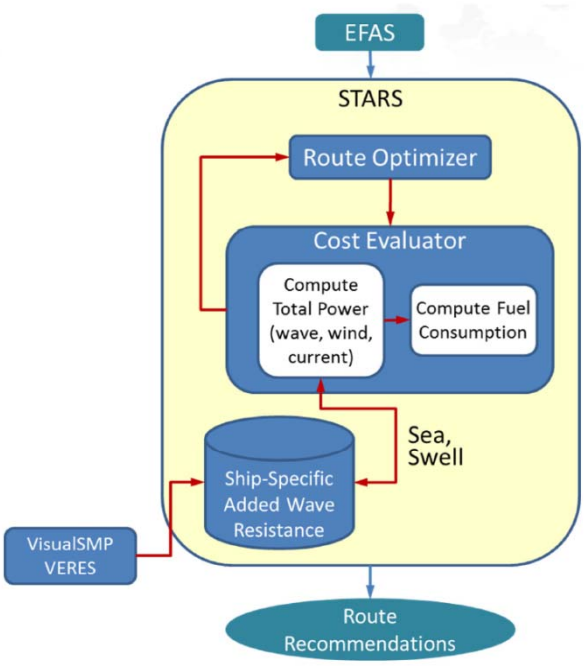


Figure 12. Graphical depiction of routing engine with improved cost predictors

B. SHIP ROUTE ENGINE PROCESSING

There are three primary processing options when running the SVP software:

- WEAX - Given waypoints, returns a route with intermediate synoptic reporting points interspersed. Reports weather and any warnings along the route. Useful for determining the actual cost of a route when run in an analysis forecast environment or for MOVEREPS. (Note: no route optimization is performed with this option)
- SPEEDX - Given waypoints, adjusts speeds between points to avoid bad weather. Useful for divert routing. (Note: only speed optimization is performed with this option)
- ROUTEX - Given start and end points or boundary limits, constructs grids within those limits and determines the optimum route within the grid based on fuel efficiency and weather/bathymetry limits. (Note: both route speed and track optimization are performed with this option)

1. WEAX Routing Option

The WEAX option is used to generate ship tracks from MOVREPs received from the fleet. MOVREPs define the track as a series of waypoints consisting of latitude/longitude location, Date-Time Group (DTG) at the point; and, optionally, course, speed and navigation type (DON 1997).

For each MOVREP waypoint the ship route engine calculates a basic route based on the navigation type:

- Great Circle – Uses a great circle calculation between the current waypoint and the next to determine distance, the DTGs at the two waypoints to determine time, and calculates speed.
- Rhumb Line – Uses a rhumb line calculation between the current waypoint and the next to determine distance, the DTGs at the two waypoints to determine time, and calculates the speed.
- Coastal and Direct Inland – Uses a shortest route algorithm to calculate the shortest route between the current waypoint and the next that, as much as possible, avoids land and shallow water. The results are smoothed and inserted in the waypoint list as new "land avoidance" points. The total distance (rhumb line) between the inserted points, and the DTGs at the start and end points are used to calculate the speed. Details of the algorithms for finding the shortest route are provided below.

The entire route is sailed in simulation and for each point:

- Calculate intermediate synoptic reporting points.
- For each point get synoptic or climatological environmental data plus bathymetry.
- Compute relative power-hours based on winds, waves and currents for ship speed.
- Save data values and mark any environmental or land limits that were exceeded.

- Save time, distance, power at that point and cumulative to that point along the route.
- The route data is then written as XML.

The purpose of the algorithm for finding the shortest route is to provide a reasonable estimate of coastal routing locations for subsequent WEAX product generation. A route should be generated that avoids land yet takes the shortest route available.

- If the track is very short (less than 30 nm), or a rhumb line route is entirely over open water, then a simple rhumb line is used. Otherwise, the following steps are taken to search for the shortest route.
- A dynamic ellipse of point values is constructed. The ellipse is oriented along a rhumb line from the start to the end point. The size varies with the total distance between the two points. Longer distances will have coarser grids to assist with computational savings.
- A series of circles around the start and end points allow for fine resolution searches for a valid open water path to and from ports that may be inland or protected by bays. The circles also allow for going away from the intended line of travel to ultimately find an open water path (say around an island or peninsula).
- The circle points are thinned during construction by only allowing those points that are over open or shallow water. Shallow water points are allowed because of the conservativeness of the bathymetry data. For example, the port location will be near or on land. The 2-minute bathymetry resolution translates very roughly to a 2 nm square. Any land or shallow water in the square will cause the whole square to be marked accordingly. Consequently, the actual start and end points of the track are allowed to be land values.

- A football shaped grid covers the bulk of the distance between the points. The football is automatically generated by computing tracks at right angles to the line of travel at regular intervals along the grid. The start and end grids are extended to form a box around the circles at the start and end points. This allows for further travel away from the intended line at a coarser resolution than the circles.
- The football grid points are thinned during construction by only allowing those points that are over open water. The grid is defined such that X is in the direction of the track and Y is at right angles to the track.
- An optimization search is performed in the direction of the track wherein the path to each grid point at location X from all of the grid points in the previous stage (X-1) is checked for bathymetry along the entire distance between the two points. The link to the best path to that grid point is chosen by these criteria:
- The least depth along the track must be shallow water except in the circles closest to the start and end of the tracks to allow approaching and leaving port.
- The end point bathymetry value is less than the bathymetry value of the previous best path end point, and the bathymetry value at the start of the track is the same or better than the start of the previous best path.
- Or the average depth along the track is less than the average depth of the previous best track and the start and end point bathy values are at least as good as those for the previous best path.
- Or the distance along the track is less than the distance of the previous best path and the average depth and depths at the start and end points are at least as good as those for the previous path.
- The track is traversed backward through the links to save the best path in common for subsequent processing.

- A smoother is applied to the track to eliminate jaggedness produced by the grid construction. A point is eliminated if a rhumb line path from the previous to the following point is completely over water.
- The remaining points are those added to avoid land. They are inserted in the original point list. The way point number value in the XML output is set to 0 to indicate that these points should be retained for the situational awareness products, but that they are not original MOVREP points.
- If no valid route is found, a rhumb line between the start and end points is returned and the problem is logged in the otsr.log file.

2. SPEEDX Routing Option

The SPEEDX option attempts to vary the arrival times at the waypoints along the track to avoid bad weather. The environmental data checking is not turned on yet in this option. This option can be utilized to support divert routes.

3. ROUTEX Routing Option

The ROUTEX option attempts to find the most efficient (least power) path between two points that also avoids land and bad weather. This option is used for route recommendations and route weather related divert calculations. Overview of the current route optimization algorithm:

- Construct a grid between the start and end points of the track. If the user has specified upper and lower boundary lines, these are used. Otherwise a football shaped grid is automatically generated.
- The grid not only has 2-dimensional latitude/longitude locations, but also a third dimension of time stages. That is, the ship can arrive at each location at a variable time.
- Using the maximum and minimum speeds for the ship specified by the user, the maximum and minimum times to each of the grid points from all

of the previous grid points are calculated. Minimum transit time (so far) is the minimum of:

- The minimum transit time found thus far; or
- Time it takes, traveling at maximum speed, to travel the distance between previous & current gridpoints in question, plus distance to get to previous gridpoint (since it's cumulative)

- Similar calculations are done for maximum time & minimum speed.
- Optimization loops (nested):
  - For each point in the X-direction (along the track)
  - For each point in the Y-direction (across the track) – y-state
  - For each time increment (delta = 1 hour) from minimum to maximum arrival time at the point
  - Compute time to travel from each previous y-state (point in last Y column) to the current point – looping over all possible arrival times from the previous state:
  - If there was a solution at the previous y-state
  - If the speed required is greater than the minimum speed, and less than the maximum speed
  - If no land was encountered for the track
  - If no environmental limits were exceeded for the track
  - Compute the total power (HP-hours or fuel) to get to the current point
  - If the total power is less than the total power to get to this point at this time from any of the previous y-state points, save it as the best solution for this y-state

- Loop through all the time solutions for the entire route, find the solution with an arrival time that does not exceed the estimated time of arrival that has the least power cost.
- If no valid route is found, return an error message and stop processing.
- If a valid route is found, the track is traversed backward through the links to save the best path in common for subsequent processing.
- Calls the WEAX routine to insert the intermediate synoptic reporting points and output the XML.

Some limitations of this algorithm:

- The grid is a fixed size designed for long (thousands of miles), open-water routes.
- There is no way to travel in a direction that is not directly from point of departure to point of arrival. For example, there is no way to purposely go around islands or peninsulas.
- The bathymetry is checked within the time loop, resulting in much longer run times than necessary.
- There is no route smoothing, so an artificial jaggedness is introduced to the final route that is caused by the grid resolution rather than any improvements in the environment.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. ENSEMBLE SYSTEM DESCRIPTION

Ensembles of meteorological and oceanographic (METOC) numerical forecast models are an excellent tool for quantifying the uncertainties in natural environments that impact tactical operations. Ensembles account for two sources of uncertainty in weather forecast models. The first is errors introduced by chaos or sensitivity dependence on the initial conditions. The second is errors introduced because of imperfections in the model, such as the finite grid spacing. The verified weather pattern should be consistent with ensemble spreads and the amount of spread should be related to the confidence of certain weather events occurring. Therefore ensembles can be a key in increasing forecast skill for better route predictions.

Model forecasts can be sensitive to the design of the model as well as to the initial conditions. Each model configuration approximates the actual behavior of the atmosphere differently, so this introduces another source of forecast uncertainty. We will never be able to construct an NWP model that includes the behavior of the atmosphere in every detail at infinitely high resolution. Even if we could create such a "perfect" NWP model, its forecast would eventually break down because of errors in initial conditions, although such a breakdown might take longer to happen. Due to the atmosphere's sensitive dependence on initial conditions, model initial conditions would need to be "perfect" for there to be any hope of making a perfect forecast. Unfortunately, the reality is that our observing and assimilation systems will never give us perfect initial conditions. However, we can apply our knowledge that the atmosphere is chaotic and highly sensitive to initial conditions to the forecast process (UCAR 2012).

A. U.S. NAVY ENSEMBLE FORECAST SYSTEM

The Fleet Numerical Meteorology and Oceanography Center (FNMOC) currently run NOGAPS in the FNMOC Global Ensemble Forecast System (GEFS). The initial conditions for the FNMOC GEFS are produced by the Navy Atmospheric Variational Data Assimilation System-Accelerated Representer (NAVDAS-AR), a 4-dimensional variational data assimilation system. The 42 level T319 spectral truncation analysis

produced by this system is used for the T319L42 control (deterministic) forecast, and also truncated to T159 and perturbed using the Ensemble Transform (ET) technique for the GEFS. The FNMOC GEFS consists of 80 NOGAPS perturbed members at T159L42 resolution run for 6-hour forecasts (used to produce the perturbations for the next cycle); 20 of these members continue the forecast out to 16 days. Output from the model runs are on one-degree by one-degree spherical grids. For production of probabilities and other statistics, the members also include one-degree grids from the deterministic NOGAPS 42-level T319 forecast and the T319L42 forecast lagged by 12 hours, for a total of 32 members. Fleet Numerical Meteorology and Oceanography Center also runs a 32 member wave model ensemble forced by winds from the NOGAPS Ensemble Forecast System. The wave model runs on a 1°x1° resolution global grid on a 12 hour update cycle. The WAVEWATCH III EFS members forecast out to 10 days (240 hours). Unlike the NOGAPS EFS, the "first guess" wave field is not perturbed; rather the variability among the members comes from the variability of the wind forcing.

B. ENSEMBLE POST PROCESSING

The Ensemble Forecast Application System (EFAS) post-processing calibration is initiated after the ensemble Gridded Binary (GRIB) files have been downloaded from the production facility. EFAS processing is controlled by a configuration file specifying the forecast parameters and data levels (e.g. sea level pressure, 10 meter wind speed and direction, 500 mb temperature, significant wave height, total sky cover, etc.) required from the numerical weather and/or ocean model ensembles. EFAS organizes the selected fields into its database and computes a bias-correction to the parameter across every ensemble member for each grid point.

The METOC ensemble that is downloaded from a central site or from multiple sites and averaged in the EFAS data base is referred to as the Raw Ensemble. The average of the raw ensemble members is called the Ensemble Average. The ensemble data calibration process is initiated by applying a bias-correction to every grid point and level, and at every forecast step, or TAU, that a result is needed. For the SVPDA model ensemble input, bias-correction was applied to the needed forecast parameters at 6-hour

intervals across the full set of available forecasts out to TAU 240 hours. For each forecast TAU (6hr, 12hr, 18hr… 240hr), and at every grid point each of the ensemble members, the forecast parameter value was subtracted from the verifying analysis value. The average difference across the ensemble members between the forecast and analysis at each grid point and TAU comprise the bias value at that grid point. For each forecast cycle (00Z forecast cycle and 12Z forecast cycle) a running mean of the last 30 days of 00Z and then 12Z forecast bias values was then computed at each TAU and grid point. Then the new 30-day running mean grid point bias-correction was applied to each ensemble member (the time span for the running mean is configurable). This data set is referred to as the bias-corrected ensemble. This process was repeated at every 12 hour forecast cycle after the raw ensemble is produced (Itri 2010).

The equation applied at every grid point is:

$$B = \frac{1}{M}\frac{1}{N}\left\{\sum_{m=1}^{M}\sum_{d=1}^{N}[P(i,j,k,t,m,d) - Pobs(i,j,k,t,d)]\right\}$$

where

- B is the bias-correction for a given forecast parameter at location i,j,k and TAU
- P*(i,j,k,t,m, d=0) = P(i,j,k,t,m,d = 0) - B
- P* is the unbiased forecast for each grid point (i,j,k), each TAU (t), and each ensemble member (m)
- P(i,j,k,t,m,d) is the raw forecast parameter for each point, TAU, and ensemble member on any given model run (d)
- N is the number of model runs (d) of history (i.e. if 30 days of 12Zforecasts; d = 30) stored for correcting the bias
- d = 0 is the current forecast time
- M is the number of ensemble members (m)

The next step in EFAS is to assemble the consensus forecast results of the parameters needed. As determined from an analysis, if a wind speed or wave height forecast is needed the information is extracted from the bias-corrected ensemble average. If a wind direction forecast is needed the information from the raw ensemble average is extracted. When these individual consensus forecasts are combined it is referred to as the hybrid forecast or the hybrid ensemble. This hybrid combination of forecast values can then be interfaced to the SVPDA in the same manner as provided by a deterministic forecast. When using hybrid consensus forecasts whose method varies by parameter and forecast TAU, it's important to remember that the resulting forecasts are not physically/meteorologically consistent because it is a statistical result.

The EFAS is interfaced to STARS to derive a 35 member (16 raw ensemble members, 16 bias-corrected members, 1 raw ensemble average, 1 bias-corrected average, and 1 hybrid) ensemble of ship routes optimized for minimum fuel burn and to avoid high winds and seas. Table 1 summarizes the ensemble forecast groups.

Ensemble	Description
Raw	16 raw members
Bias Corrected	16 bias corrected members, w/ 30 day running mean
Hybrid	1 member using raw wind direction and BC for other input parameters

Table 1. Various post processed ensemble types

Figure 13 identifies the environmental and ensemble inputs into STARS.

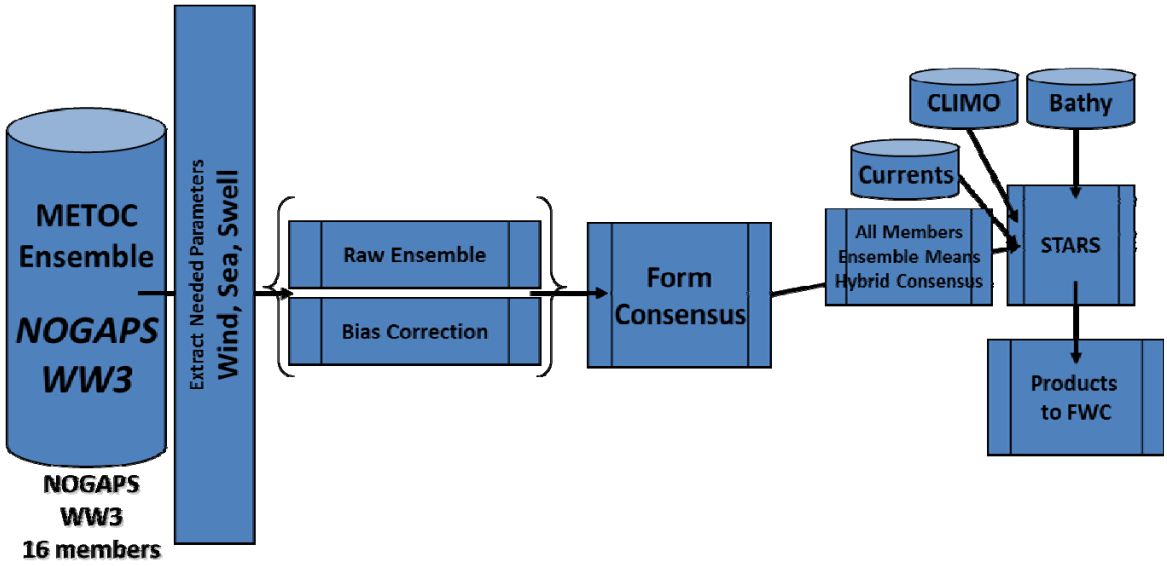


Figure 13. Environmental and ensemble STARS model inputs

THIS PAGE INTENTIONALLY LEFT BLANK

V. HULL FORM AND PROPULSION PLANT MODELS

A. SPECIFIC HULL CHARACTERISTICS

The version of STARS used for this research utilizes an implementation of higher fidelity, ship-specific cost evaluation functions including effects of wind, waves, and current related to specific ship hull shape and propulsion plants. Specific modeled ship class improvements include:

- Propeller and hull cleaning schedule
- Analytic estimates of added resistance in waves incorporating wave period into optimization objective evaluation
- Use of fuel consumed (instead of horsepower required) for the cost metric with inclusion of actual ship fuel consumption data
- Inclusion of calm water powering data from full-scale ship trials and model scale experiments
- Inclusion of fuel consumed by electric plant loads

1. Fuel Consumption Ship Speed Sensitivity

Fuel consumption is very sensitive to ship speed. Fuel consumption rate increases rapidly with ship speed, by approximately 1 to 4% per knot at moderate speeds and approximately 9% per knot at high speeds. Fuel consumption efficiency in gas turbine ships is also very sensitive to lower speeds. Therefore, optimizing the ship speed profile during transit can yield significant fuel savings. Realistic propulsion fuel curves are used for the various classes of ships. A typical gas turbine propulsion plant fuel curve is identified in Figure 14.

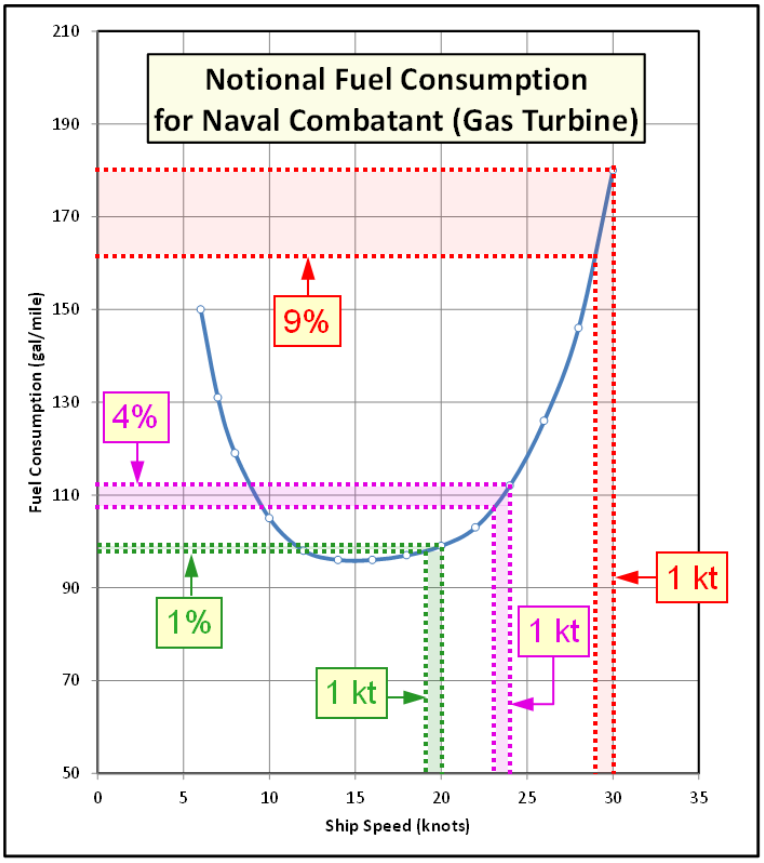


Figure 14. Typical gas turbine fuel consumption curve and relationship to speed

2. Fuel Consumption Environmental Sensitivities

A ship's fuel consumption rate increases significantly with moderate waves, wind, and current. At constant speed, fuel consumption in sea state 4, with 1 knot current, increases by approximately 10% over the calm water value. Therefore, optimizing the ship route together with the speed profile to avoid adverse environmental conditions during transit can yield even greater fuel savings. This effect is illustrated in Figure 15.

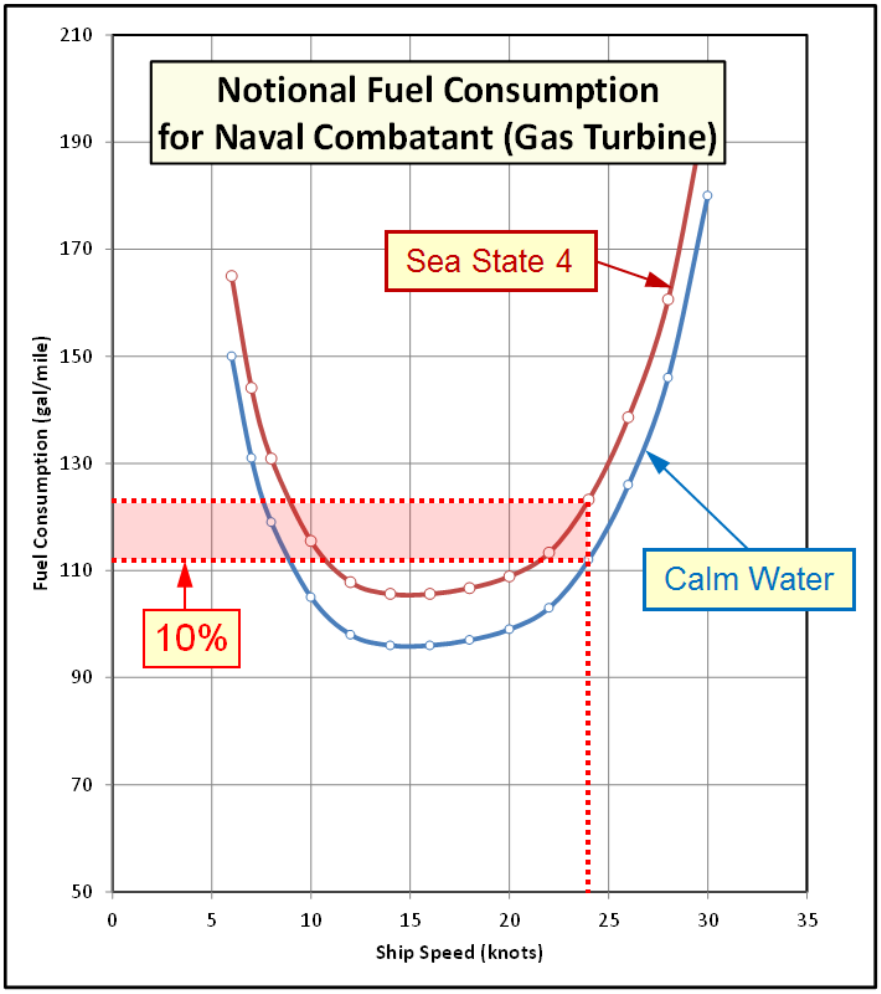


Figure 15. Typical gas turbine fuel consumption curve and relationship to sea state

3. Speed Reduction Curves

An additional set of ship performance curves are used to estimate the ship's Speed of Advance (SOA) while transiting the forecast sea states. These curves are called speed reduction curves or speed curves. The curves indicate the effect of head, beam, and following seas of various significant wave heights on the ship's speed. Each vessel will have its own speed reduction curves, which vary widely according to hull type, length, beam, shape, power, and tonnage. Recommendations for vessels must also account for wind speed, wind angle, and vessel speed (NIMA 2002). An example of speed reduction curves for the DDG-58 class ships are identified in Figure 16.

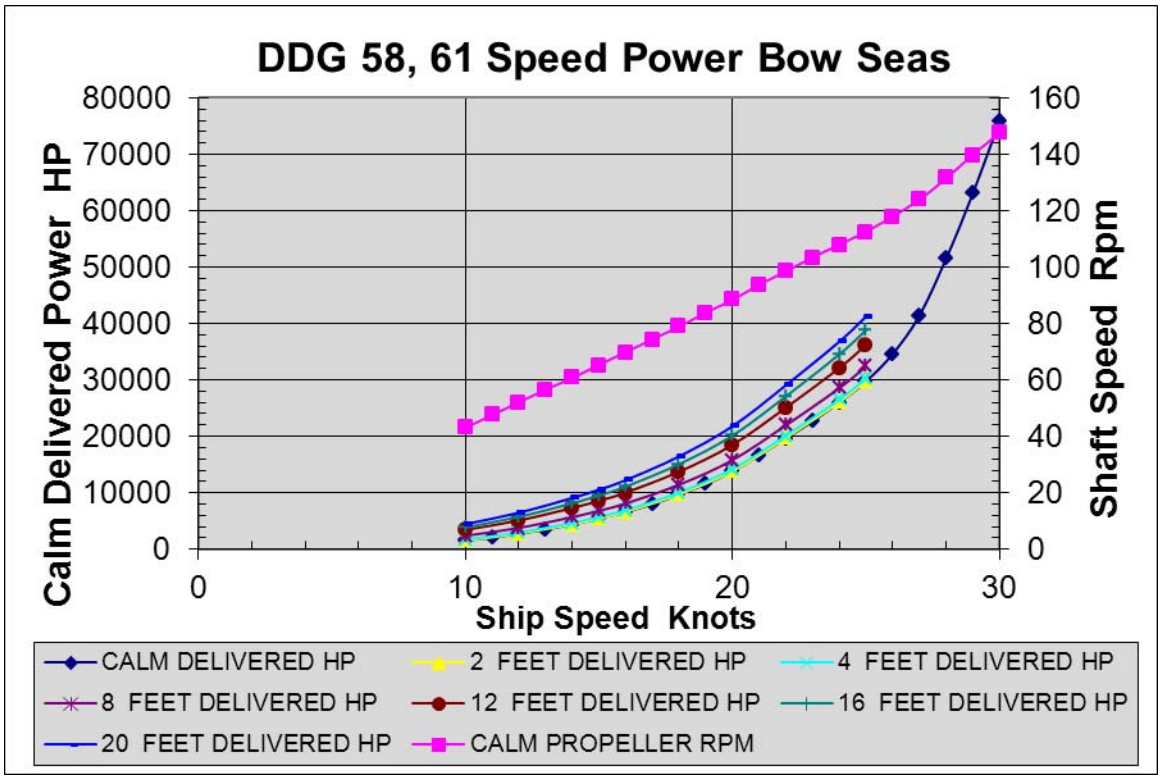


Figure 16. DDG 58 speed reduction curves for bow seas

B. SHIP CLASS INPUTS

To implement ship class specific inputs, a STARS compatible data-store format was created consisting of three files. For each class of ship, SHIPCLASS, is the name of the ship class read from an input file as follows:

- SHIPCLASS.dat - contains calm water powering information, fuel consumption information, and other general ship information
- SHIPCLASS.sea.dat - contains added resistance data for the "sea" component as functions of ship speed, wave height, wave period, and relative wave heading
- SHIPCLASS.swell.dat - contains added resistance data for the "swell" component as functions of ship speed, wave height, wave period, and relative wave heading

C. SHIP CLASSES USED FOR EXPERIMENTS

Two types of naval ship classes were used for this study; one naval auxiliary class (TAO) and one surface combatant class (DDG). Additionally, two DDGs of the same class were used that had variances in hull and propeller fouling. The modeled DDG-90 hull had a simulated 6 months of additional fouling relative to the DDG-93 hull.

1. USNS HENRY J. KAISER (TAO-187)

- Class & Type: Henry J. Kaiser class fleet replenishment oiler
- Tonnage: 31,200 tons
- Length: 677 ft (206 m)
- Beam: 97 ft 5 in (29.69 m)
- Draft: 35 ft (11 m) maximum
- Engines: 2 Diesels with 34,442 HP (25.683 MW) total sustained
- Propulsion: Two shafts with controllable pitch propellers

2. USS CHAFFE (DDG-90) & USS CHUNG-HOON (DDG-93)

- Class & Type: Arleigh Burke class destroyer
- Tonnage: 9,300 tons
- Length: 509 ft 6 in (155.30 m)
- Beam: 66 ft (20 m)
- Draft: 31 ft (9.4 m)
- Engines: 4 gas turbines with 100,000 shp (75 MW)
- Propulsion: Two shafts with controllable pitch propellers

THIS PAGE INTENTIONALLY LEFT BLANK

VI. MODEL ANALYSIS

One of the primary goals during my sensitivity studies was to assess the impacts and sensitivity of ocean and atmospheric modeling input parameters for an optimum ship routing model. Ensemble methods were utilized for quantifying the environmental model uncertainties and improving forecast skill. I also determined the benefits of using realistic platform characteristics for naval vessels. Finally, I attempted to determine impacts of the individual NOGAPS, WW3 and NCOM model's quality of input effects on SVPDA route outputs. In order to try and capture variability due to space and time, sensitivity analysis was conducted over the course of various seasons and various global locations. The model provides outputs in Extensible Markup Language (XML) format (see Appendix D) and required a robust scripting tool in order to properly parse all route output for each model run. I wrote a complex PYTHON script, which enabled parsing of the XML output files and created flat files in addition to several statistical figures. Sample model analysis figures are presented in Appendix A and analysis charts from the various cases are presented in Appendix B. The PYTHON computer code written to generate the flat files and statistical figures is presented in Appendix D.

A. ANALYSIS TOOL

1. Horizontal Bar Graphs

In order to visualize the various route costs by using ensembles, a customized display methodology was developed, which displays all ensembles in a horizontal rank order. By viewing this figure, a picture can be derived concerning how well the initial ensemble spread predicted the overall environment outcome and route costs vs. analysis costs by using analysis environmental data. This is observed by viewing the overall blue horizontal bar layout (predicted route cost) in relation to the green horizontal bar layout (predicted route run in the analysis environment cost). Observations indicated that sometimes initial ensemble spreads predicted the overall analysis route cost spread, and

at other times, it under predicts or over predicts the cost. This is a characteristic that is also seen in environmental forecast ensemble parameters when compared with analysis results.

As is typically the case with ensembles, the lowest cost member, may have actually been the highest cost member based on the analysis environmental data. The figures also clearly identify the rank order of all ensembles; both raw/corrected predicted and raw/corrected analysis. Additionally, the Great Circle route, analysis route, post-processing averaged routes, and the three different types of analysis model type routes are also displayed. For the three variants of model analysis forecast groups, I chose to use a combination of analysis results for one of the models and the respective "Hybrid ensemble" post-processing grouping for the other two models. This logic was chosen to provide 2 models with, what was thought to be, the best post processed ensemble combination and one model with the analysis data to determine the best case outcome for improving individual environmental models. Based on my testing, on average, the WW3 model with analysis data appeared to display the greatest improvement when rank ordered with NOGAPS and NCOM analysis input variants. An example of the horizontal bar graph is presented in Appendix A. The various ensemble member names, number of members, and member color schemes are identified in Table 2. This standard set of ensemble members was used for all case experiments and figure color schemes unless otherwise specified in figure captions.

Ensemble Input	Number	Color
Great Circle (base line route)	1	Red
Raw & BC Ensemble Members (A2…Z1;BC_A2…BC_Z1)	32	Blue
Analysis WEAX Raw & BC Ensemble Members (A2…Z1;BC_A2…BC_Z1)	32	Green
Post Processed Ensembles (Ensemble. Avg., Bias Corrected Avg., Hybrid)	3	Cyan
Model Analysis Ensembles (NOGAPS analysis, WW3 analysis, NCOM analysis)	3	Teal
Full Analysis Member	1	Magenta

Table 2. Various ensemble member names, numbers, and associated color schemes used in case experiments, horizontal bar graphs, and Google Earth presenations unless otherwise specified in figure captions

2. Histograms

Ensemble route Probability Distribution Functions (PDFs) were displayed using histograms. One version of the histogram identifies the aggregate of all route members, including the predicted ensemble costs, the analysis ensemble costs, the post processed ensemble costs, and the great circle cost. This aggregate histogram sometimes appears to exhibit negative skew, positive skew and bimodal distributions of fuel costs. For distance costs (length of routes), the distribution typical exhibited a Gaussian distribution pattern. Time was usually not sensitive to the various ensembles due to the programing logic of the SVPDA tool. The aggregate histograms were also displayed in a vertically separated arrangement with predicted ensemble route costs in the upper frame and the ensemble analysis route costs in the lower frame. For the fuel costs, these separate histograms typically displayed large variances causing them to fail "Student – T" tests. This identified that the population groups were not the same and that our forecasts are not perfect. It also identifies that we cannot just derive a delta function for route outputs to represent what might actually occur in reality compared to what is predicted. Additional general patterns identified were the clear differences noted between predicted vs. actual costs. In some cases the predicted routes exhibit a clear lower cost spread and the

analysis cost spread was shifted higher. Therefore, for this case, we can assume the analysis environment was less favorable for the SVP ensemble of routes. In other cases, the opposite was observed where the predicted costs were higher than the analysis costs. In this case, it can be surmised that the analysis environment was more favorable for the SVP ensemble of routes than the predicted environment. At other times, the overall distribution shape of the analysis weather costs spread became flatter at both tails. This could be representative of increased randomness in the environment compared to what was predicted. The aggregate distance cost predicted and analyses PDFs were typically Gaussian. The distance cost split PDFs were, on average, very similar and passed the "Student-T" test. Therefore, the distance parameter was sensitive to the various ensemble members, but there did not seem to be significant differences in sensitivity from the predicted vs. analysis forecasts. As previously noted, time cost PDFs typically did not display significant sensitivity among various ensemble members.

3. Flat File

The analysis PYTHON scripting also creates a flat file with all route ensemble member statistics. Due to the complex directory structures and multi-level XML files, the flat file allows for easy table and chart generation using tools such as Excel. Additionally, results of any limits exceeded along with the respective limit and associated ensemble member are written to the flat file. Excel charts with rank order depictions of the fuel, distance, and time parameters in Appendix B were created from these generated flat files.

VII. SVPDA MODEL RUN

The core of the SVPDA ship route engine consists of the STARS program, written in FORTRAN, which is executed in a UNIX environment with the following usage command:

/u/curr/bin/stars INPUT.DAT OUPUT.XML

The model is manipulated with a number of external scripts also inside the UNIX environment. As previously stated, my sensitivity analysis program was written in PYTHON and all other scripts were written in Bourne Again Shell (BASH) along with some auxiliary JAVA functions. BASH and PYTHON code examples are presented in Appendix D. The SVPDA model run process diagram in presented in Figure 17.

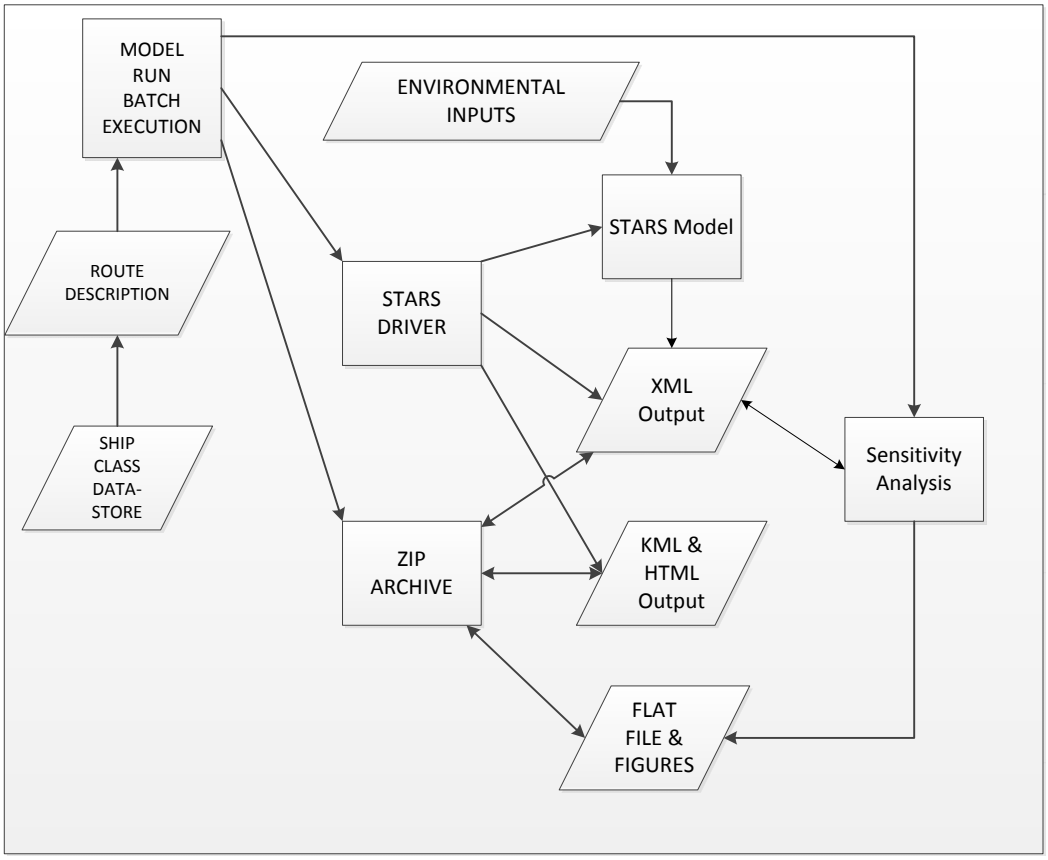


Figure 17. SVPDA model run process diagram

A. INPUT INTERFACE

The .in file contains user specified input unique to the proposed route as identified in Appendix D. The SHIPCLASS.*.dat files contain class specific information as described above. The synoptic environmental data fields are extracted as each model produces the required fields. Models used are NOGAPS, WW3, and NCOM. The environmental output grids are stored in directory /otsr/grid/[MODEL]/[DTG] in IEEE format. Grids are generated once per synoptic (12-hour) cycle. Climatology environmental data fields were developed by the FNMOC models department from NCEP re-analysis wind fields for the last 15 years. The FNMOC WWIII Hindcast model was used to generate sea and swell data from the NCEP winds. The input parameters include: marine wind u and v components, sea height, sea period, sea direction, swell height, swell period, swell direction, current u and v components. Forecast field Taus are saved at 6-hour intervals out to 240 hrs. The Tropical Cyclone Warnings (TCW) wind radii data are parsed from TCW files and stored in the /otsr/storms/]/[DTG] directory to simplify processing in the Ship Route Engine. Bathymetry is extracted from DBDBV 2-minute resolution bathymetry, provided by Naval Research Laboratory, Stennis Space Station, Mississippi.

B. OUTPUT INTERFACE

The OUPUT.XML file contains track data identifying the route simulation. An example OUTPUT.XML file is shown in Appendix D. Externally called BASH/JAVA output functions are executed next and automatically generate HTML and KML files for all modeled routes. The analysis PYTHON script is then called, which generates flat files from the XML output files along with the previously discussed figures. A final BASH script is then called to archive all generated files in a single .zip file.

C. TEST CASES

The SVPDA analysis tools that were developed can analyze parameter sensitivities (fuel, distance, time) when evaluating the spread of ensembles. In order to objectively assess the model output, test cases were run using archived Naval Research

Laboratory environmental data from 2010-2011. A comparison of the predicted route output was then made utilizing forecast ensembles and analyzing results after re-running the various ensemble routes using analysis data sets.

The various test cases are described as follows:

1. Case 1

Model inputs:

Route – Diego Garcia to Gulf of Oman/Gulf of Oman to Diego Garcia;
One way Great Circle distance – 1946.60 nm;
Ship classes & hulls– TAO-187, DDG-90 & 93;
Ship wind speed limits – 35 knots (18 m/s) for bow, beam, and stern;
Ship sea heights limits – 12 feet (3.66 m) for bow, beam, and stern;
Maximum allowable speed – 25 knots (12.86 m/s);
Minimum allowable speed – 10 knots (5.14 m/s);
Great Circle baseline speed – TAO (17.5 kts), DDG (16.6 kts);
Number of start and ending waypoints – 2;
Number of upper and lower bound waypoints – 4 each;
Dates – 20100601, 20100701, 20101201, & 20110601.

Results are summarized in Tables 3–5 and a sample ensemble of routes for a single run date is depicted in Figure 18.

Fuel Used (galx1000)	**TAO-187**	**DDG-90**	**DDG-93**
Min	94.65	172.40	168.57
Max	115.79	206.36	198.01
Mean	100.88	186.57	157.78
Variance	19.62	72.48	61.39
STD	3.65	7.35	6.38

Table 3. Case 1 Diego Garcia to Gulf of Oman and Gulf of Oman to Diego Garcia route overall fuel statistic averages (galx1000)

Ensemble (% fuel used vs. GC)	TAO-187	DDG-90	DDG-93
Best Member	-12.6	-5.04	-4.87
Analysis NOGAPS	-11.75	-3.22	-3.07
Ensemble Average	-11.53	-4.12	-3.97
Analysis WW3	-11.49	-2.78	-2.64
Bias Corrected	-11.47	-1.62	-1.49
Hybrid	-11.12	-3.3	-3.15
Analysis NCOM	-11.06	-2.77	-2.63

Table 4. Case 1 Gulf of Oman to Diego Garcia, north to south, ensemble post-processing % fuel used vs. GC (ensemble rank order sorted based on TAO-187 results; negative #'s denote % fuel savings vs. GC)

Ensemble (% fuel used vs. GC)	TAO-187	DDG-90	DDG-93
Best Member	-20.97	-12.71	-10.57
Ensemble Average	-19.04	-11.59	-8.58
Hybrid	-18.99	-9.18	-7.3
Bias Corrected	-18.87	-9.74	-6.45
Analysis WW3	-15.56	-7.21	-7.01
Analysis NOGAPS	-14.63	-6.18	-6
Analysis NCOM	-14.6	-5.25	-5.09

Table 5. Case 1 Diego Garcia to Gulf of Oman, south to north, ensemble post-processing % fuel used vs. GC (ensemble rank order sorted based on TAO-187 results; negative #'s denote % fuel savings vs. GC)

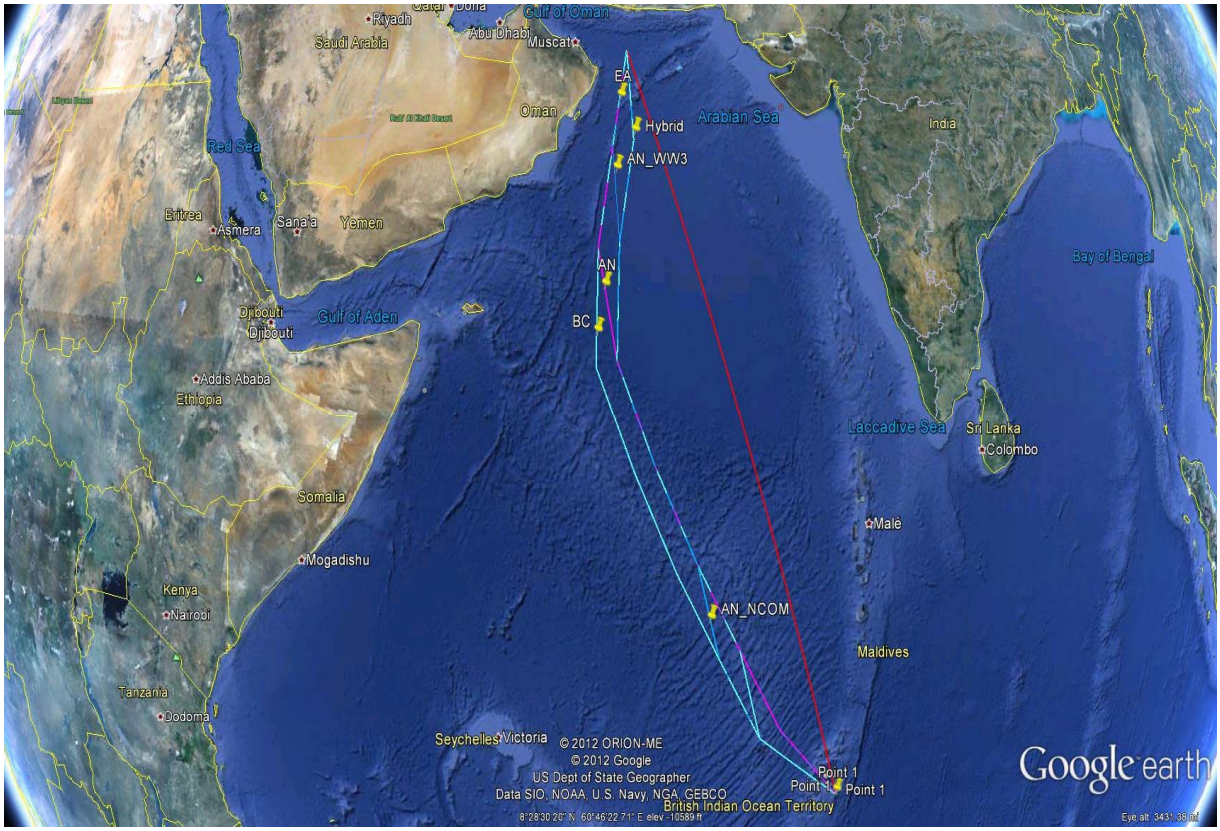


Figure 18. Case 1 (01 Jun 2010): Diego Garcia to Gulf of Oman ensemble of routes displaying GC (red), post processed (cyan), analysis model variant (teal), and analysis (magenta) ensembles

2. Case 2

Model inputs:

Route – San Diego to Pearl Harbor/Pearl Harbor to San Diego;

One way Great Circle distance – 2191.09 nm;

Ship classes & hulls– TAO-187, DDG-90 & 93;

Ship wind speed limits – 35 knots for bow, beam, and stern;

Ship sea heights limits – 12 feet (3.66 m) for bow, beam, and stern;

Maximum allowable speed – 25 knots;

Minimum allowable speed – 10 knots;

Great Circle baseline speed – TAO (17.5 kts), DDG (16.6 kts);

Number of start and ending waypoints – 2;

Number of upper and lower bound waypoints – 4 each;

Dates – 20100601, 20100701, 20101201, & 20110601.

Results are summarized in Tables 6–8 and a sample ensemble of routes for a single run date is depicted in Figure 19.

Fuel Used (galx1000)	TAO-187	DDG-90	DDG-93
Min	111.40	192.95	188.98
Max	131.74	233.92	227.64
Mean	120.23	216.20	210.70
Variance	30.86	159.98	141.69
STD	4.17	10.26	9.78

Table 6. Case 2 San Diego to Pearl Harbor a Pearl Harbor to San Diego overall fuel statistic averages (galx1000)

Ensemble (% fuel used vs. GC)	TAO-187	DDG-90	DDG-93
Best Member	-11.62	-9	-8.56
Analysis WW3	-11.05	-4.24	-4.15
Analysis NCOM	-10.77	-2.41	-2.33
Ensemble Average	-10.42	-4.89	-4.79
Hybrid	-10.41	-2.77	-2.69
Analysis NOGAPS	-10.18	-2.66	-2.59
Bias Corrected	-9.42	-3.09	-3.01

Table 7. Case 2 San Diego to Pearl Harbor, east to west, ensemble post-processing % fuel used vs. GC (ensemble rank order sorted based on TAO-187 results; negative #'s denote % fuel savings vs. GC)

Ensemble (% fuel used vs. GC)	TAO-187	DDG-90	DDG-93
Best Member	-10.63	-2.5	-1.78
Analysis WW3	-10.07	-2.06	-1.53
Analysis NOGAPS	-8.76	0.56	1.12
Hybrid	-8.41	1.09	1.14
Analysis NCOM	-8.35	-0.7	-0.1
Bias Corrected	-7.82	1.65	1.72
Ensemble Average	-7.26	0.44	1.02

Table 8. Case 2 Pearl Harbor to San Diego, west to east, ensemble post-processing % fuel used vs. GC (ensemble rank order sorted based on TAO-187 results; negative #'s denote % fuel savings vs. GC)

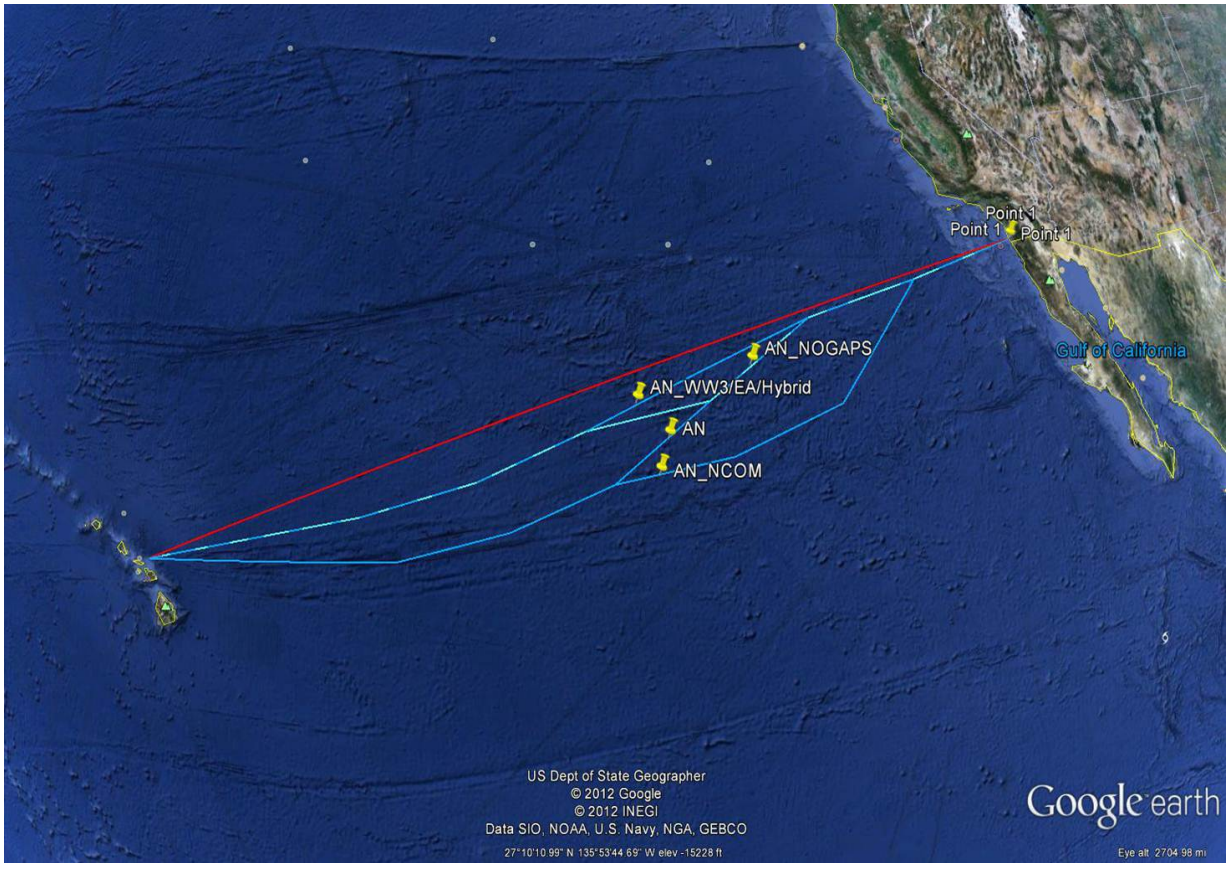


Figure 19. Case 2 (01 May 2010): San Diego to Pearl ensemble of routes displaying GC (red), post processed (cyan), and analysis model variant (teal) ensembles

3. Case 3

Model inputs:

Route – Norfolk to Rota/Rota to Norfolk

Ship classes & hulls– TAO-187, DDG-90 & 93;

Ship wind speed limits – 35 knots for bow, beam, and stern;

Ship sea heights limits – 12 feet (3.66 m) for bow, beam, and stern;

Maximum allowable speed – 25 knots;

Minimum allowable speed – 10 knots;

Great Circle baseline speed – TAO (17.5 kts), DDG (16.6 kts);

Number of start and ending waypoints – 2;

Number of upper and lower bound waypoints – 4 each;

Dates – 20100501, 20100901, 20101201, 20110201 & 20110601.

Results are summarized in Tables 9–11 and a sample ensemble of routes for a single run date is depicted in Figure 20.

Fuel Used (galx1000)	**TAO-187**	**DDG-90**	**DDG-93**
Min	174.20	271.59	243.51
Max	219.73	349.32	317.64
Mean	200.75	312.55	282.64
Variance	106.11	321.43	282.62
STD	9.50	16.71	15.79

Table 9. Case 3 Norfolk to Rota and Rota to Norfolk overall fuel statistic averages (galx1000)

Ensemble (% fuel used vs. GC)	TAO-187	DDG-90	DDG-93
Analysis WW3	-8.36	-8.52	-14.09
Best Member	-3.66	-8.62	-10.38
Analysis NOGAPS	0.29	-3.2	-2.69
Analysis NCOM	0.98	-3.34	-3.66
Hybrid	1.02	-1.3	-3.06
Ensemble Average	1.62	-4.46	-2.97
Bias Corrected	2.3	-0.62	-1.94

Table 10. Case 3 Norfolk to Rota, east to west, ensemble post-processing % fuel used vs. GC (ensemble rank order sorted based on TAO-187 results; negative #'s denote % fuel savings vs. GC)

Ensemble (% fuel used vs. GC)	TAO-187	DDG-90	DDG-93
Best Member	-0.21	-2.08	0.03
Analysis WW3	0.77	-0.68	-1.69
Analysis NCOM	2.11	2.96	5.45
Analysis NOGAPS	2.59	1	4.27
Hybrid	2.78	3.96	5.92
Bias Corrected	3.01	4.32	6.56
Ensemble Average	3.14	0.42	4.34

Table 11. Case 3 Rota to Norfolk, west to east, ensemble post-processing % fuel used vs. GC (ensemble rank order sorted based on TAO-187 results; negative #'s denote % fuel savings vs. GC)

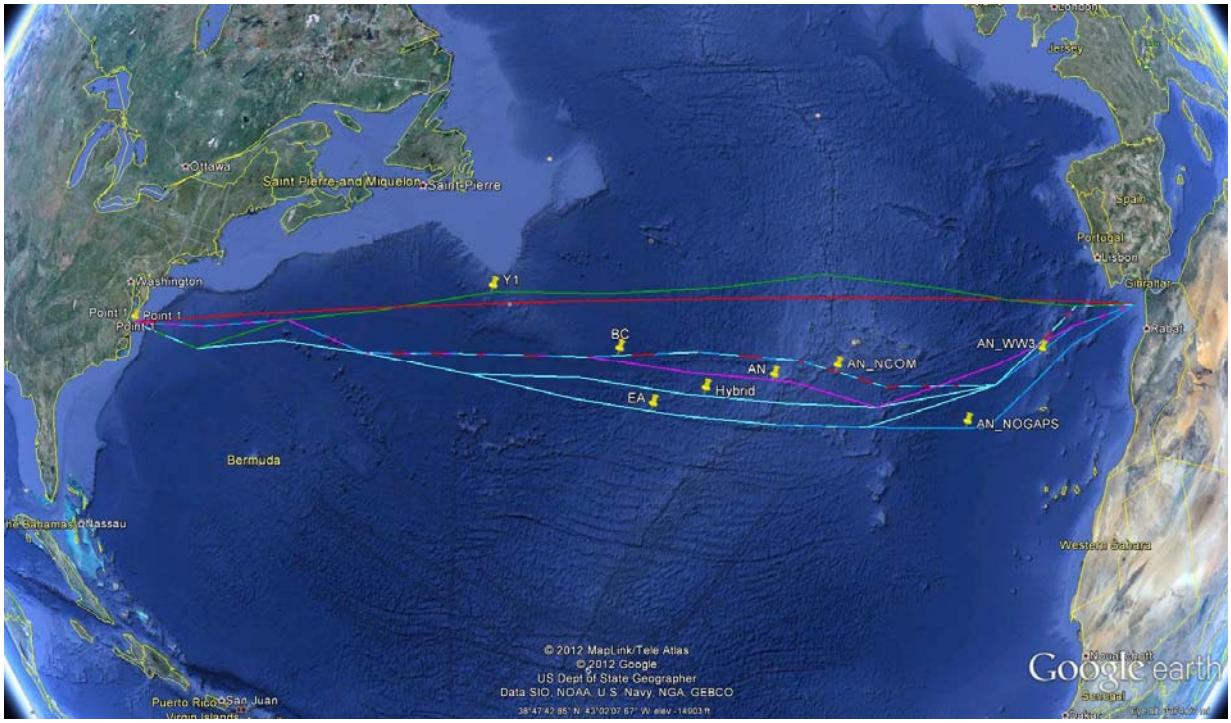


Figure 20. Case 3 (01 Dec 2010): Norfolk to Rota ensemble of routes displaying GC (red), raw member (green), post processed (cyan), analysis model variant (teal), and analysis (magenta) ensembles

4. Case 4

Model inputs:

Route location – Eastern Pacific northern hemisphere tests;

Route direction – East to west/west to east/north to south/south to north

One way Great Circle distance – 1500 nm;

Ship class and hull – TAO-187;

Ship wind speed limits – 35 knots for bow, beam, and stern;

Ship sea heights limits – 12 feet (3.66 m) for bow, beam, and stern;

Maximum allowable speed – 25 knots;

Minimum allowable speed – 10 knots;

Great Circle baseline speed – TAO (15 kts)

Number of start and ending waypoints – 2;

Number of upper and lower bound waypoints – 4 each;

Dates – 20100601, 20100801, 20101001, 20101201, 20110201 & 20110601.

Results are summarized in Tables 12, 13 and a sample ensemble of routes for a single run date is depicted in Figure 21.

Fuel Used (galx1000)	East to West & West to East	North to South & South to North	Combined
Min	75.96	125.58	70.38
Max	90.18	162.51	89.14
Mean	81.96	132.85	76.26
Variance	17.66	44.63	24.94
STD	3.18	8.11	4.08

Table 12. Case 4 Eastern Pacific northern hemisphere tests overall fuel statistic averages (galx1000)

Ensemble (% fuel used vs. GC)	East to West & West to East	North to South & South to North	Combined
Best Member	-1.79	-8.38	-3.96
Analysis WW3	2.1	-6.98	-2.44
Analysis NCOM	-0.4	-4.1	-2.25
Analysis NOGAPS	0.12	-2.42	-1.15
Bias Corrected	0.47	-2.6	-1.06
Ensemble Average	2.37	-4.46	-1.04
Hybrid	0.92	-2.09	-0.58

Table 13. Case 4 Directional and combined ensemble post-processing % fuel used vs. GC (ensemble rank order sorted based on combined results; negative #'s denote % fuel savings vs. GC)

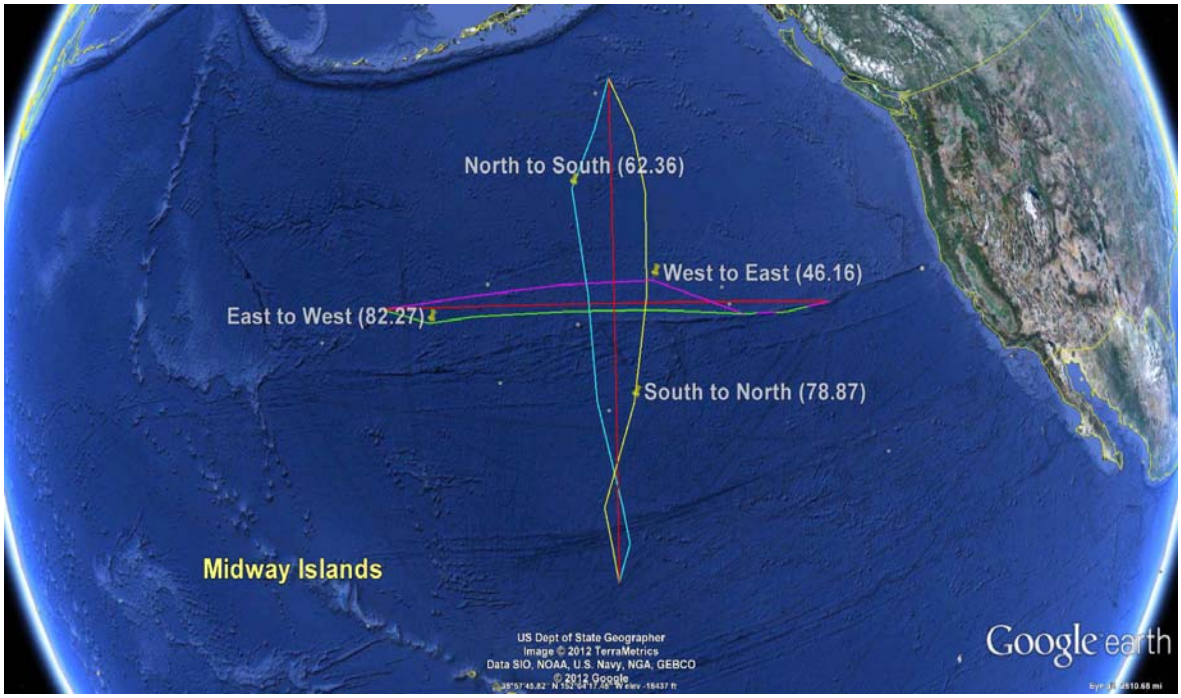


Figure 21. Case 4 (01 Jun 2010): Eastern Pacific N. Hem. displaying ensemble of routes w/ respective fuel costs (galx1000); N/S & E/W GC (red); Hybrid ensemble inputs for the following directions sailed: north to south (teal), south to north (yellow), east to west (green), west to east (magenta)

5. Case 5

Model inputs:

Route location – Eastern Pacific equatorial tests;

Route direction – East to west/west to east/north to south/south to north

One way Great Circle distance – 1500 nm;

Ship class and hull – TAO-187;

Ship wind speed limits – 35 knots for bow, beam, and stern;

Ship sea heights limits – 12 feet (3.66 m) for bow, beam, and stern;

Maximum allowable speed – 25 knots;

Minimum allowable speed – 10 knots;

Great Circle baseline speed – TAO (15 kts);

Number of start and ending waypoints – 2;

Number of upper and lower bound waypoints – 4 each;

Dates – 20100601, 20100801, 20101001, 20101201, 20110201 & 20110601.

Results are summarized in Tables 14, 15 and a sample ensemble of routes for a single run date is depicted in Figure 22.

Fuel Used (galx1000)	East to West & West to East	North to South & South to North	Combined
Min	67.88	73.51	70.69
Max	75.80	80.86	78.33
Mean	70.43	77.80	74.11
Variance	2.65	1.76	2.21
STD	1.38	1.25	1.31

Table 14. Case 5 Eastern Pacific equatorial tests overall fuel statistic averages (galx1000)

Ensemble (% fuel used vs. GC)	East to West & West to East	North to South & South to North	Combined
Best Member	-7.68	4.47	-1.61
Analysis WW3	-7.5	5.54	-0.98
Bias Corrected	-6.59	5.12	-0.74
Analysis NCOM	-6.57	5.67	-0.45
Ensemble Average	-6.58	6.01	-0.29
Analysis NOGAPS	-6.71	6.28	-0.22
Hybrid	-6.28	6.09	-0.1

Table 15. Case 5 Directional and combined ensemble post-processing % fuel used vs. GC (ensemble rank order sorted based on combined results; negative #'s denote % fuel savings vs. GC)

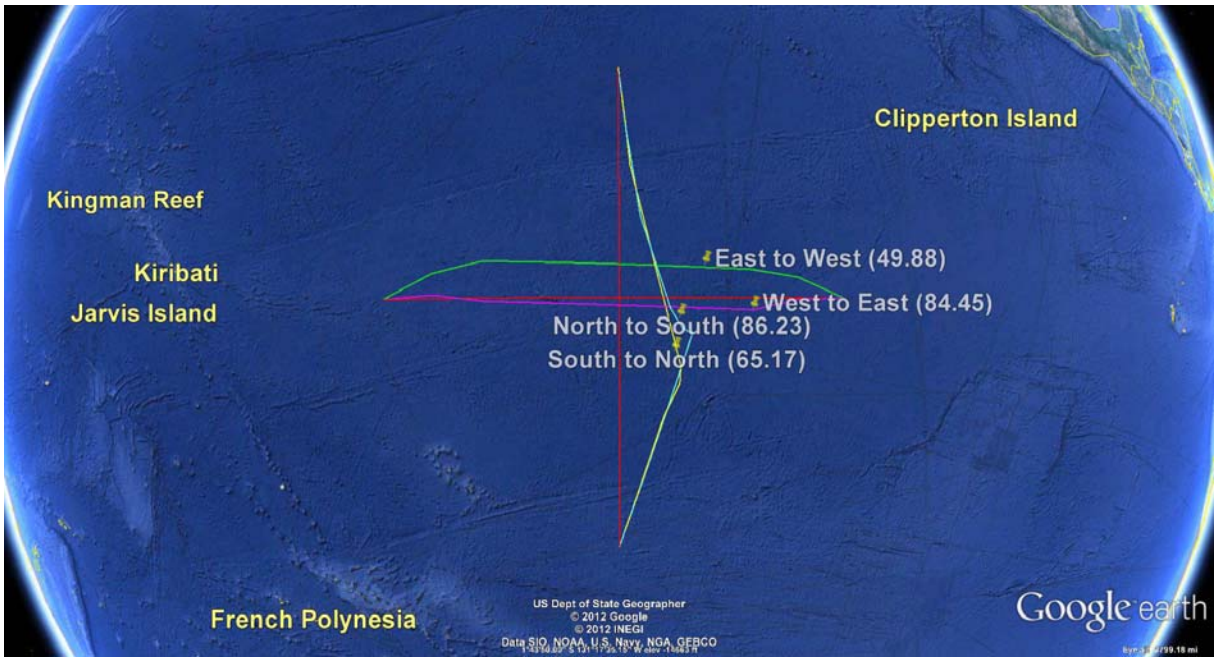


Figure 22. Case 5 (01 Jun 2010): Eastern Pacific equator displaying ensemble of routes w/ respective fuel costs (galx1000); N/S & E/W GC (red); Hybrid ensemble inputs for the following directions sailed: north to south (teal), south to north (yellow), east to west (green), west to east (magenta)

6. Case 6

Model inputs:

Route location – Eastern Pacific southern hemisphere tests;

Route direction – East to west/west to east/north to south/south to north

One way Great Circle distance – 1500 nm;

Ship class and hull – TAO-187;

Ship wind speed limits – 35 knots for bow, beam, and stern;

Ship sea heights limits – 12 feet (3.66 m) for bow, beam, and stern;

Maximum allowable speed – 25 knots;

Minimum allowable speed – 10 knots;

Great Circle baseline speed – TAO (15 kts);

Number of start and ending waypoints – 2;

Number of upper and lower bound waypoints – 4 each;

Dates –20101001, 20101201, 20110201 & 20110601.

Results are summarized in Tables 16, 17 and a sample ensemble of routes for a single run date is depicted in Figure 23.

Fuel Used (galx1000)	East to West & West to East	North to South & South to North	Combined
Min	62.55	60.12	61.34
Max	89.82	76.41	83.11
Mean	71.64	67.86	69.75
Variance	42.62	15.93	29.28
STD	5.43	3.61	4.52

Table 16. Case 6 Eastern Pacific southern hemisphere tests overall fuel statistics (galx1000)

Ensemble (% fuel used vs. GC)	East to West & West to East	North to South & South to North	Combined
Best Member	-7.62	-8.77	-8.2
Analysis WW3	-2.57	-6.9	-4.74
Bias Corrected	-3.02	-6.28	-4.65
Analysis NCOM	-3.53	-5.52	-4.52
Hybrid	-3.14	-5.02	-4.08
Analysis NOGAPS	-2.3	-5.47	-3.88
Ensemble Average	5.6	-1.69	1.95

Table 17. Case 6 Directional and combined ensemble post-processing % fuel used vs. GC (ensemble rank order sorted based on combined results; negative #'s denote % fuel savings vs. GC)

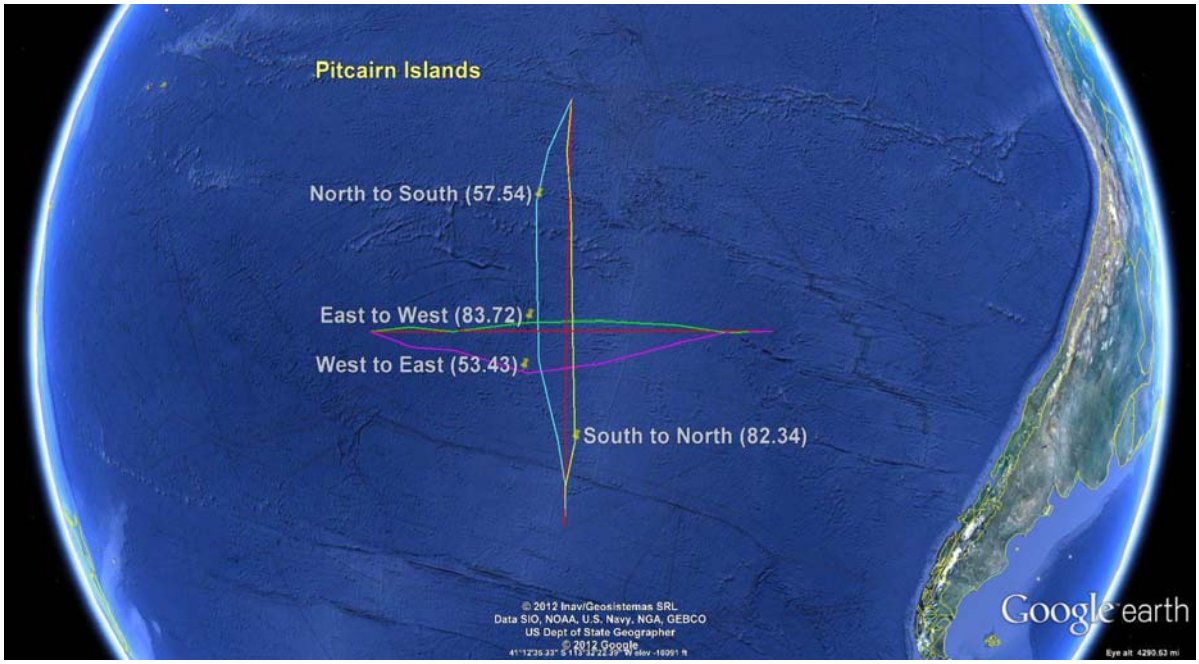


Figure 23. Case 6 (01 Dec 2010): Eastern Pacific S. Hem. displaying ensemble of routes w/ respective fuel costs (galx1000); N/S & E/W GC (red); Hybrid ensemble inputs for the following directions sailed: north to south (teal), south to north (yellow), east to west (green), west to east (magenta)

D. DETAILED MODEL OUTPUTS

Detailed fuel, distance and time costs for the various cases are presented in Appendix B. These results were created using flat file outputs from the sensitivity analysis PYTHON code. Graphical output charts in this Appendix were generated using Microsoft Excel.

VIII. SENSITIVITY STUDY RESULTS

A. SVPDA MODEL SENSITIVITY SUMMARY

1. Geographic Location Effects

Cases 1–6 identify variances in model output sensitivities based on geographic locations. In cases 4 and 6, high latitude testing of around 40N/S and higher indicated that, on average, using the pure bias corrected post-processing method for input into STARS provides the most fuel efficient routes vs. the hybrid ensemble and ensemble average methods. Also, at or near the equator and on average, the ensemble average post-processing method performed better than the hybrid and bias corrected post-processing methods.

2. Directional Effects

Large sensitivity was also identified based on ship route direction in all cases. This was evident by using 2 combinations of lat/long points and varying the direction sailed, (i.e. east to west, west to east, north to south and south to north as applicable). The change in direction alone had a profound effect on the computed SVP route cost and geometry. Additionally, for the limited cases tested in this thesis, the SVPDA appeared to save more fuel, on average, during longitudinal parallel routes vs. latitudinal parallel routes.

3. Seasonal Synoptic and Mesoscale Weather Effects

Seasons had a profound effect on the route variances as identified by reviewing the outputs from spring, summer, fall and winter runs. This effect was observed in all cases and also varied with different geographic locations. Using the GC route as a baseline, some longitudinal parallel routes displayed a clear geometric east of GC route bias during the summer season and a west of GC bias during the winter season. This was evident in the Indian Ocean where the shift in the Monsoon winds and weather patterns based on time of the year apparently has a large impact on optimal routes. Additionally,

heavy weather makes it difficult to simulate fuel savings vs. GC routes due to diverts around the weather to avoid exceeding ship environmental limits. However, the ship would have needed to divert anyway, or risk damage due to exceeding vessel limits. For this type of situation, the optimizer may have to speed the ship up to avoid heavy weather. This was evident in case 3 where some of the test dates required weather avoidance. Therefore, the "optimized" route may be more expensive than the great circle route, but it is safer. Another reason for the "optimized" route to have a higher cost is due to STARS performing route costs by sailing through only grid points from a particular grid location. STARS then evaluates the route cost plus the 6 hr. synoptic time points. This addition of points can sometimes result in a different cost than the cost used in the optimization process due to the potential of additional weather information. Also of note, is that some versions of STARS can be calibrated to force the ship to arrive within a specific time window. Due to this constraint and if there is heavy weather in the route path, the optimizer may have reduced abilities to achieve the safest route so caution should be used when placing this constraint on STARS.

4. Hull/Propulsion Type and Condition Effects

There was a very obvious variance among the various hull classes and propulsion types used as indicated in cases 1–3. To be as efficient as possible, the SVPDA needs to be carefully tuned based on the specific hull form, optimum ship speeds and advantageous propulsion plant lineups. An example of this was identified with the TAO, which used significantly less fuel than the gas turbine powered warships were projected to use. This was most likely due to the linearly increasing fuel curve that characterizes the TAO diesel power plant vs. the non-linear bowl shaped fuel curve used in the gas turbine powered ships. Additionally, diesel engines are, in general, more fuel efficient than gas turbine engines, especially at lower speeds. As identified in cases 1–3, the DDGs, on average, displayed a much larger variance and standard deviation in ensemble fuel route costs. This indicates that the SVPDA simulated gas turbine propulsion plant is more sensitive to environmental ensemble input variances. Therefore, careful choice of the best post processed ensemble member must be made in order to achieve the most fuel

savings in all cases, but especially for gas turbine powered vessels. The current version of STARS does not optimize based on specific propulsion plant lineups, but as explained in the following chapter, large additional fuel savings could be realized if this feature were implemented.

Variances in hull cleanliness also identified that the hull with increased fouling (DDG-90) burned more fuel for a given route and speed using the GC route as a reference. However, the interesting finding was that, on average, the SVP route for a hull with more fouling was able to generate an increased relative fuel savings by approximately 1% vs. the cleaner hull (DDG-93). This finding was identified by using the DDG-90 vs. DDG-93 class data-stores as inputs variants, while holding other inputs constant. The DDG-90 data-store modeled its hull and propeller cleaning 6 months before DDG-93; therefore DDG-90 should have increased fouling, inducing a greater friction cost and reduced propeller efficiency. The only exception to this finding was during periods of heavy weather where the cleaner (DDG-93) hull appeared to perform marginally better. In the former case, this outcome makes sense as the more fouled hull should have increased friction, so at low to moderate speeds, the optimizer can ensure the ships heading is placed in an optimal wave spectrum to minimize speed reduction and therefore increase efficiency to a slightly greater degree vs. a hull with less friction. Increased friction may also help efficiency in "following sea" situations. In the latter case, at higher speeds, the hull with a cleaner and more efficient propeller generates increased thrust and therefore greater fuel efficiency vs. a more fouled propeller. Therefore, at higher speeds, the greater propeller efficiency overcomes the optimizer advantage of increased hull friction. It must be noted that these results are based on each ship's relative optimized efficiency vs. the GC route for each vessel. The latter case was apparent in the Norfolk to Rota routes where the optimizer had to increase ship speed to avoid heavy weather. In general, the gas turbine ships also displayed sensitivity to both low and high speeds due to their bowl shaped fuel curves. Therefore, if the SVPDA were to slow the ships to below the speed where fuel consumption increases markedly, a severe fuel cost penalty could be incurred similar to traveling at too high a speed. The DDG's were also relatively slightly more efficient compared to the

TAO at higher speeds. This effect was again identified during the Norfolk to Rota experiment, which, as previously stated, the optimizer required higher speeds due to weather effects.

5. Route Length

Based on the SVPDA's spatial discretized resolution, it makes sense that longer transoceanic runs (2500 nm and greater) would suit this tool best to enable increased opportunities for more fuel efficient routes. However, relatively shorter routes, (around 1500 nm) also displayed noticeable improvements in some of the latitude tests, cases 4-6. Therefore the SVPDA tool may generate noticeable fuel savings when used for these relatively shorter routes in addition to the longer transoceanic routes.

6. Specific Model Improvements

As previously discussed and as could be seen in the detailed analysis from Appendix B, the model forecast fields were simulated to use "one at a time" perfect prognosis for the NOGAPS, WW3, and NCOM models. In all test cases, the results of this experiment indicated that the various models impacted the route costs differently and, on average, the simulated perfect prognosis WW3 appeared to have the largest percentage gain for increasing route optimization. However, during periods of heavy weather, sometimes the simulated perfect prognosis NOGAPS appeared to have the largest benefit on route optimization. Since WW3 is driven by 10 m winds, we must also consider the importance of the NOGAPS surface wind output quality effects on the WW3 model. Rank order of the various model output tests also varied slightly when comparing the TAO vs. DDG platforms.

7. Ensemble Post-processing Methods

It was originally thought that, in most cases, the hybrid ensemble performed the best among the current post-processing methods. However, as my experiments have shown, it appears that the hybrid ensemble is not always the best post processed member

for the SVPDA model environmental input. Specifically, the raw ensemble average appeared to perform best, on average, near the equator while the bias corrected ensemble appeared to perform best, on average, at higher latitudes. Additionally, as in the perfect prognosis model tests, the rank order varied slightly among the TAO and DDG platforms. As expected, the "best ensemble member" performance was almost always significantly better than all other post-processing methods. If a method could be developed to get closer the best member, then significant additional fuel savings gains could be made for the SVPDA.

THIS PAGE INTENTIONALLY LEFT BLANK

IX. USS PRINCETON CG-59 CONOPS TRIAL

A. SVPDA INTEGRATED PROGRAM TEAM

During work on this thesis, I was able to take part in a NAVSEA Integrated Program Team (IPT) to assist with development of the SVPDA tool for operational fleet use. This effort is part of the Navy's surface ship energy conservation program initiative. I also had the opportunity to test SVPDA CONOPS for the USS PRINCETON (CG-59) in an OPDEMO sea trial test following the 2012 Rim of the Pacific (RIMPAC) exercises.

B. CONOPS TRIAL

I developed a Data Collection and Analysis Plan (DCAP) in conjunction with Military Sealift Command (MSC) engineering staff to support this test. I then embarked on USS PRINCETON in Pearl Harbor, HI and sailed with the ship back to its home port in San Diego on a 6 day voyage. To support testing, communications were established via email, chat and Plain Old Telephone System (POTS) lines. A MOVEREP was generated by the ship and was the mechanism used to obtain a SVP route. Once, the MOVEREP was received by Fleet Weather System (FWC), San Diego, the request was then sent to NRL, Monterey for processing on the SVPDA model. An SVP output route was then sent back to FWC, San Diego and finally sent back to the ship. Once the smart voyage route was received by the ship, the way points and speeds were entered into the ship's ECDIS-N system. After verification by the ship's Navigator and chain of command, the ship began sailing the SVP route. This process is graphically displayed in Figure 24. Engineering, navigation and environmental logs were taken by the crew throughout the voyage for future analysis in accordance with the DCAP. During the cruise, various events occurred that required modification to the SVP route. The above process was then repeated as necessary to obtain new SVP routes. Unfortunately, due to uneventful weather, the routes were essentially Great Circles, but many valuable CONOPS lessons learned and best practices were discovered.

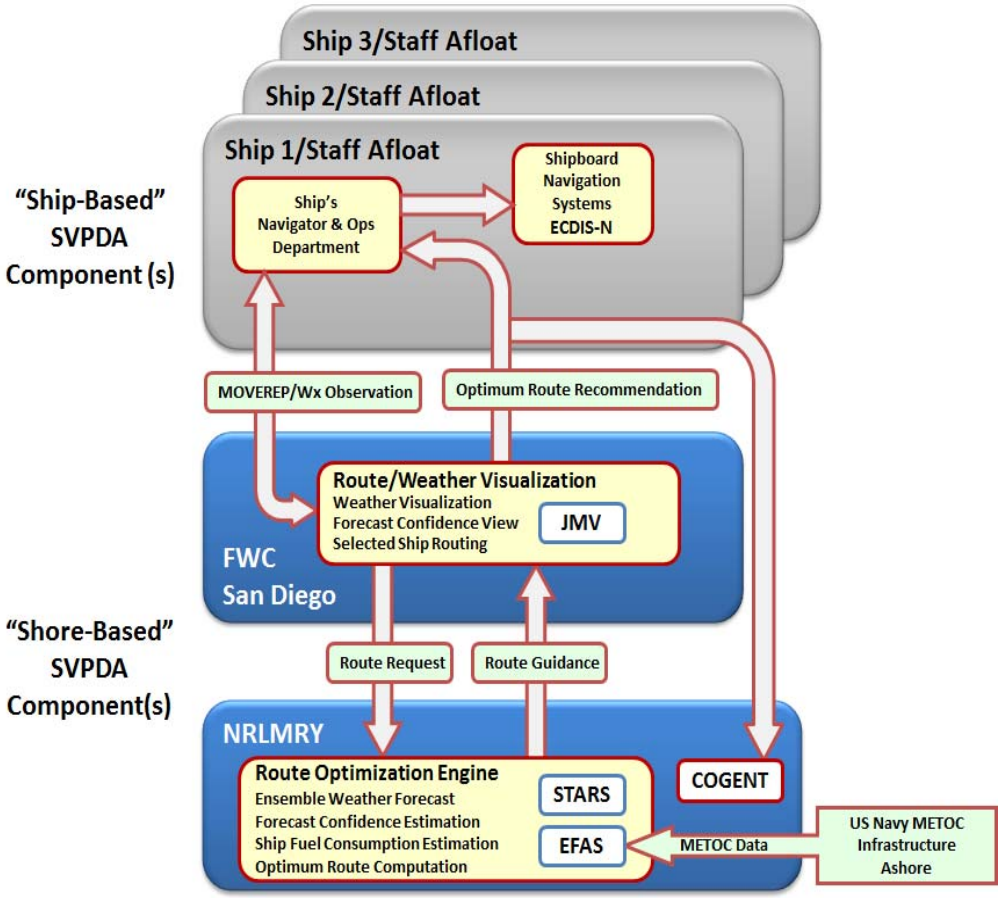


Figure 24. Ship to shore CONOPS for the USS PRINCETON SVP sea trial.

C. LESSONS LEARNED AND RECOMMENDATIONS

The original SVP was created from the ship's departure port berthing area to the arrival port berthing area. However, most if not all SVP routes should begin at the departure port marker and end at the arrival port marker. Before this point for departure and after this point for arrival, the Navigator will have full control of the ship's route due to control of tugs, numerous hazards to navigation, and speed limit constraints experienced while departing and entering ports. Also, the initial SVP route received by the ship contained over 60 waypoints. Currently, ships must enter these waypoints manually into the ECDIS-N systems and with a large number of points; this can be a time consuming and arduous process. Therefore, until an automated process is established, the routes should be smoothed as much as practical on a "not to interfere basis" with saving

fuel or compromising ship safety. The waypoints were also received in a format incompatible with the current ECDIS-N system, so a script was written on the shore side to correct the lat/lon formatting. CONOPS should be developed to import the SVP directly as a voyage plan into the Voyage Management System (VMS). This would alleviate the crew from the task of entering numerous waypoints into the system and minimize the possibility of manual entry errors.

The ship's crew found the DCAP to be easy to follow and execute. It was written in a somewhat generic manner so that it can be executed with minimal training and on multiple ship platform types. Additionally, once the SVP was entered into the ship's ECDIS-N system, it was relatively easy to execute and sail. Logs were collected on a daily basis from the engineering and navigation personnel. Some minor adjustments to the electronic format logs were required based on ship specific machinery, but this was expected and relatively easy to update after the principle department heads provided feedback. Of note, the deck log should be utilized as much as possible to identify when a SVP route begins/ends or if the ship is deviating from the SVP route along with the reason for future reference and voyage reconstruction analysis. Daily recorded logs were sent off the ship via classified email once every two days to a shore side repository.

The communication process also went well with ship to shore connectivity on non-classified & classified email, chat, POTS and normal message traffic channels. There were times when the ship had to secure executing the SVP or pause and then request an update based on its current lat/lon position. This process worked fairly seamlessly with the only limit being the round trip time that it took to request a new route, receive it and enter the update into the ECDIS-N system. On average, this process took approximately 2 hours. A lesson learned from this evolution was that the ship should dead reckon 2 hours down track and use this lat/lon for the starting point of the newly requested SVP route due to the round trip delay time. This enables the ship to begin sailing the new SVP route approximately at the same time that it has been entered into the system.

A list of ship operations that would affect the ship being able to follow the SVP route were examined, and actually encountered, during this CONOPS trial. The first

operational change that required an updated SVP route occurred when the ship had a training evolution change for a mandatory exercise. For this exercise, the ship required two new routes, since the Commanding Officer (CO) required 2 options with different lat/lon's. For future SVPDA CONOPS planning, it's feasible that multiple routes could be requested by a ship. For instance, a CO may need to have flexibility based on later decision points. During this route revision request, it was discovered that the speed of advance was not high enough to get from point A to point B w/in the allotted time specified. The operational version of STARS should contain some form of error checking to perform Quality Assurance (QA) checks for this type of erroneous request. It should then inform the operator of the problem and suggest a higher maximum allowable speed or increase the time of arrival window. Another possibility is that SVPDA would provide the operator a best route, but note that time of arrival is now later to account for the given maximum speed constraints. The second route revision was required when the ship's maximum speed was lowered due to an engineering plant degradation. The ship's maximum speed may vary based on propulsion plant line-ups, maintenance and other mechanical issues. Specific propulsion plant line-ups may limit maximum speed so that the ship can run in trail shaft mode. This mode is only available up to a certain speed and can provide a significant fuel efficiency advantage with this change alone. Therefore the SVP engine needs to be optimized to provide speed limits based on the propulsion plant efficiency capabilities or reduced capabilities (i.e. trail shaft or split plant ops only). One of the initial SVP routes run, during the CONOPS trial, required the ship to travel 18 kts for 90% of the voyage, but then slowing at the end. However if the ship could have sailed the entire route at 17 kts in trail shaft mode, then significant additional fuel savings could have been realized. This is assuming there were no environmental weather systems that could require a higher speed to divert and avoid exceeding ship limits. The third revision to the SVP route was required when an unscheduled Underway Replenishment (UNREP) became necessary. The above examples encompass many of the situations that may trigger revising or securing the SVP routes for naval vessels. The following list also includes the above situations and some additional cases:

- Operational tasking changes

- Training evolutions
- Propulsion plant degradation
- Propulsion plant line-up constraints
- Maintenance constraints
- Underway replenishments
- Flight operations
- Tactical operational requirements
- Escort requirements

FNMOC assisted in developing a useful decision aid that enabled importing the SVP routes and weather into a SIPRnet Google Earth application. Utilizing this tool, shipboard personnel were able to view an entire SVP route with overlaid weather. It was then possible to simulate sailing the ship down track in the future with the weather evolving each day. This tool enabled the CO and ship's senior leadership to quickly obtain situational awareness as to why a route was possibly zigging to avoid weather or slowing the Speed of Advance (SOA) to minimize the effects of strong head seas, or to possibly take advantage of strong following seas. For possible future CONOPS, the navigator and CO could use this tool to quickly view SVP routes before entering into the ships ECDIS-N system. After reviewing weather history from this tool following the trial, there appears to be value in obtaining updated SVP routes every day or at least every other day. The weather changed noticeably over the course of 4 days compared to what was initially predicted off the coast of the Western U.S. An example FNMOC Google Earth environmental overlay is depicted for a Norfolk to Rota SVP route in Figure 25. In this example, the SVP route diverts south of the GC to avoid high winds and seas.

Shallow water areas are obviously areas that we would hope the SVPDA tool would automatically navigate around and this function is currently built into the SVPDA tool as described in Chapter II. However, in addition to shallow water/land avoidance, other geographic areas of concern also exist. Some examples of these include exercise

"hot areas," training markers and other types of "stay out" or "remain clear" Areas of Operations (AORs). There should be a mechanism where the Navigator could submit supplemental lat/lon boxes to include these areas in the MOVEREP SVP request and the SVP routes would then stay clear of these areas. A simple coding solution to this problem would be to just treat these additional areas as "shallow areas" therefore forcing the SVPDA engine to find the most efficient route around the flagged areas.

Naval instructions and references that may require revision to incorporate SVPDA CONOPS were also reviewed. One of the identified references was NWP 1-03.1, Operational Reports. This reference contains codified requirements for MOVEREPS and will most likely have to be modified to incorporate the CONOPS of the SVP process. There may be a need for ships to submit marginal route changes or speed limit modifications while still meeting the intent of the original MOVEREP, so submitting a new MOVEREP may be unnecessary. Therefore a mechanism should exist to request a new SVP route without requiring a new MOVEREP if the intent of the original MOVEREP is still being met. Another reference requiring revision is the 3140.1M, United States Navy Meteorological and Oceanographic Support System Manual. This METOC instruction codifies requirements for FWC/FNMOC support related to MOVEREP OTSR and surveillance of ship route request.

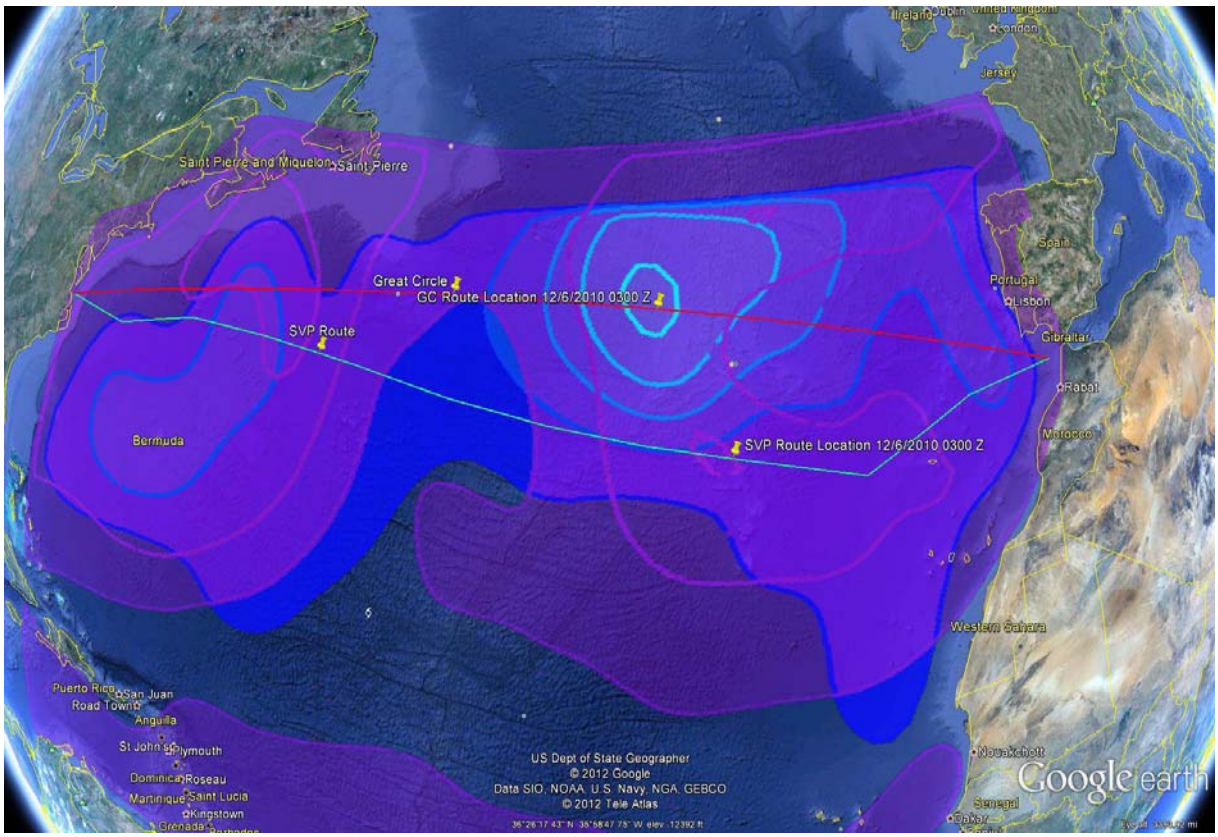


Figure 25. Google Earth Norfolk to Rota Great Circle route (red) and SVP route (cyan) with overlaid environmental winds/waves depicted as polygons. The SVP route tracks south of the high seas to maintain vessel within safe limits (shades of purple denote winds in 5 kt increments, shades of blue denote seas in 3 ft increments)

THIS PAGE INTENTIONALLY LEFT BLANK

X. CONCLUSION AND FUTURE WORK

A. CONCLUSIONS

One of the primary goals during my sensitivity studies was to assess impacts and sensitivity of ocean and atmospheric modeling input parameters for an optimum ship routing model. Ensemble methods were used for quantifying the environmental model uncertainties and to help improve forecast skill. I also evaluated the effects of individual environmental model improvements for input into the SVPDA tool. Finally, I attempted to determine the benefits of using realistic platform characteristics of various classes of naval vessels. My findings have identified that the SVPDA model is very sensitive to the following:

- Geographic location
- Direction
- Seasonal Synoptic and Mesoscale Weather
- Hull/Propulsion Type and Condition
- Route Length
- Specific Model Improvements
- Ensemble Post-processing Methods

During conduct of the CONOPS trial onboard USS PRINCETON, I determined and experienced various types of operations that could affect a combatant vessel in conducting a SVP route. Key lessons learned, best practices and recommendations were also gained during conduct of this operational trial.

B. FUTURE MODEL AND POST-PROCESSING IMPROVEMENTS

With the ongoing work in nesting higher resolution models, such as COAMPS, and actually coupling air/sea models, the SVPDA tool can directly benefit from increased model skill. This is with an assumption that the low-level wind skill will help drive

improved WW3 skill since it driven from surface winds. Specific model performance also plays a key role as WW3 was identified as having the biggest impact in improving the SVPDA tool if the output skill were increased. This was identified while using analysis data for each environmental model and holding the remaining two in a common post-processed environment.

Large fuel cost improvements were indicated by utilizing the best ensemble member. As expected, the best ensemble member consistently outperformed other post-processing techniques and came close to, if not equaling, the analysis forecast many times in terms of fuel savings. Therefore, if we can find a way to get closer to the best ensemble member, decision aids requiring the use of environmental inputs will see a significant benefit. Although it's impossible to pick the best member, utilizing improved "state of the art" post-processing techniques such as a Kalman filter may assist with the getting closer to a best member, vice using the current 30 day training bias correction and hybrid ensemble methods. Kalman filter techniques will also most likely reduce the number of training days required for bias correction, therefore may also improve processing times.

My experiments have shown there are very large variances in fuel savings based on the sensitivity factors discussed. On average, some routes and associated directions exhibited very good fuel savings, while a few exhibited only marginal savings. Therefore it would be beneficial to build a comprehensive expected fuel savings database of simulated seasonal, ship class specific SVP geographic routes. This database should then be validated by comparing expected with realized savings once naval vessels begin using this decision aid.

Taking advantage of specific propulsion plant lineups can play a key role in realizing substantial additional fuel savings. Specific ship classes can take advantage of alternate plant lineups, such as split plant or trail shaft operations based on maximum speed requirements. For example, a ship may be required to travel at 18 knots (9.26 m/s), but if it were able to slow by just one knot (.51 m/s) to 17 knots (8.75 m/s), allowing trail shaft operations, then an additional 10% in fuel savings may be realized with this change alone.

SVPDA processing time, while using multiple ensembles, can easily be sped up by running parallel execution processes. I experimented with running 4 ensemble threads in parallel vs. just 1 "at a time" sequentially and experienced a speed increase of approximately 300%. This was on a 4-core INTEL processor where 1 ensemble process was run per thread, up to a maximum of 4 threads. Based on additional trial runs and while still using the 4-core processor, increasing the number of threads to greater than 4 began to slow the processing time. The overall run execution time with a nominal 2000 nm route and 36 ensemble members takes approximately 1 hour on average. However, this run time can vary significantly if the route takes a path near shallow bathymetry or islands since the bathymetry check is currently in the optimization loop. If this TDA was fully parallelized on a 40 core machine, I would expect the processing time to be reduced significantly since each ensemble can run in its own thread.

Future experiments should focus on evaluating the sensitivity effects of updating the input forecast periodically throughout the SVPDA run. This change in CONOPS could be accomplished by updating the SVP forecast and re-running every 24 or 48 hours. I would expect to see a relatively good percentage gain in fuel savings vs. just using the initial STARS run, especially during the winter seasons. This change could also increase the safety of the SVP route since the environmental inputs would be more accurate as the forecast range becomes shorter term and forecast skill increases. A downside to this periodic updating of the STARS route would be increased workloads on the FWCs and affected ships that would have to the update the routes more frequently in their ECDIS-N navigation systems. Perhaps some kind of a "bell ringer" could be built into the SVPDA software, which would re-run the routes automatically and compare the newly run outputs to the initially executed route. If an updated route changes by a given threshold, or if a limit is exceeded by the original route, then a flag could be set warning operators that a new SVP route should be processed.

Utilization of the SVPDA tool in conjunction with additional decision aids being developed, such as the Replenishment at Sea Program (RASP) for MSC supply replenishment ships, should also be pursued. RASP is an ongoing project spearheaded by the Operations Research Department at Naval Post Graduate School (Brown 2008). This

tool optimizes frequency, departure/arrival times, and locations for conducting UNREPS. The synergistic combination of SVP routes and optimum RASP routes makes perfect sense. During future SVPDA CONOPS trials, SVP routes could easily be conducted in conjunction with RASP testing.

The SVPDA tool requires a robust enhancement to the user interface to enable full operational use. Significant coordination should be made between the developers and customers to ensure the tool meets "ease of use" requirements for operators and senior leadership. Since FWC's are already familiar with the JMV interface, perhaps the SVPDA user interface could be developed with a similar layout. Although, the SVPDA routes are currently compatible with Google Earth, compatibility of the outputs with other GIS tools, such as COGENT, should be explored. Additionally, a means of exporting KMLs or PNGs into external software packages should be available to allow for rapid display of routes in senior level operational briefs.

C. FINAL SUMMARY REMARKS

The U.S. Navy Fleet Weather Centers perform all missions safely and effectively. However, there has been a convergence of better environmental models, improved maritime communications and greatly improved computer processing power. This, in turn, makes operationalizing a smart voyage planning decision aid practical. The basic CONOPS trial conducted on USS PRINCETON proved this decision aid's feasibility, even in its early prototype stage. Additionally, some newer naval vessels have more narrow design safety margins where it's crucial that the ships operate within their safe environmental envelopes. In addition to saving fuel, the SVPDA tool has the ability to build in increased ship class specific safety margin buffers to avoid possible damaging environmental effects. This can decrease vessel maintenance requirements and improve their lifespans, which is extremely advantageous during budget constrained environments. Therefore, the SVPDA tool can effectively maximize fuel efficiency while also improve both ship safety and reduce cumulative stress from environmental factors thereby possibly reducing life of ship maintenance costs.

APPENDIX A. MODEL ANALYSIS TOOL EXAMPLES

A. HORIZONTAL BAR GRAPHS

1. TAO-187 Ensemble Fuel Spread (Norfolk to Rota)

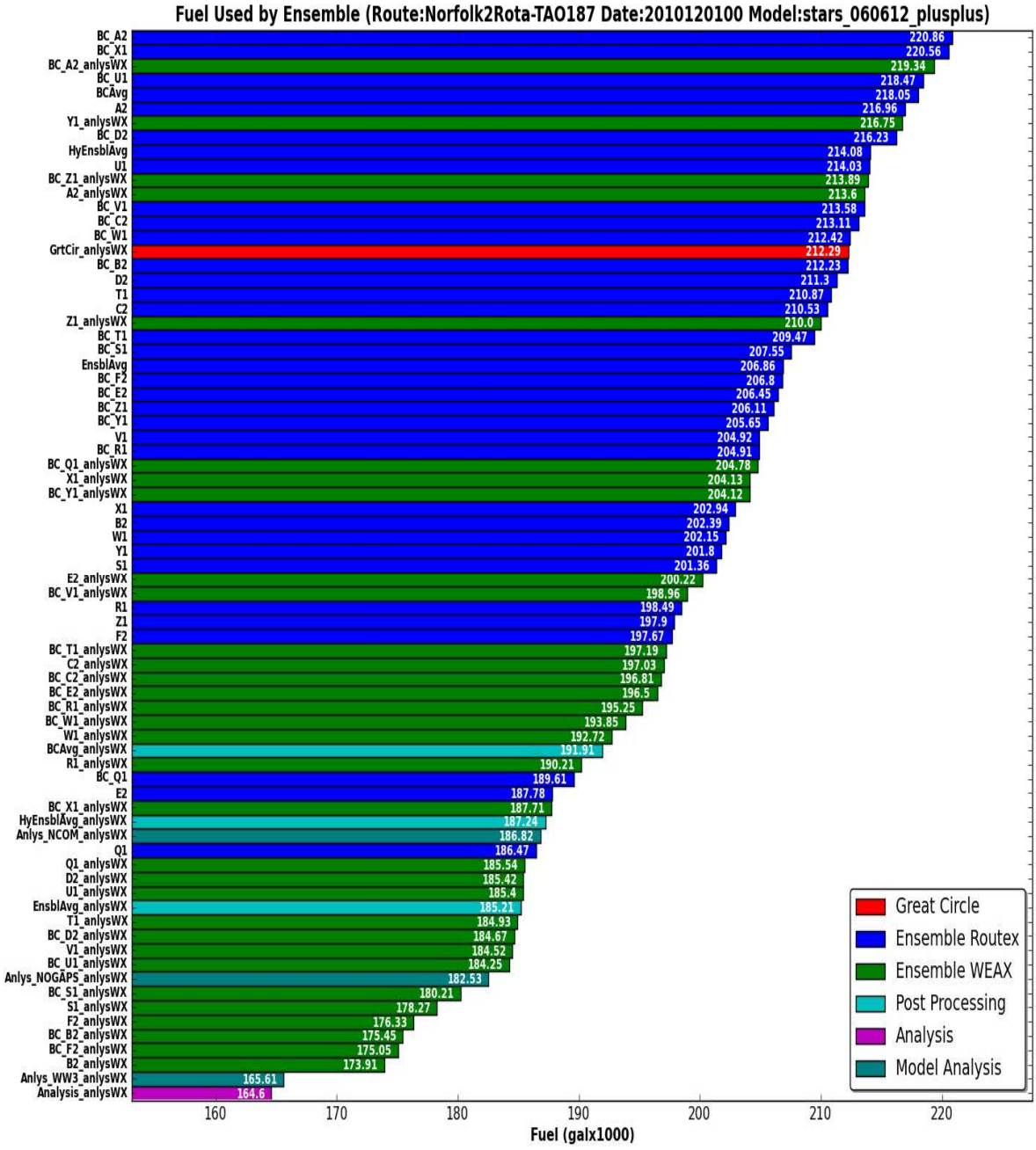


Figure 26. TAO-187 fuel used by ensemble spread distribution (Norfolk to Rota: 01 Dec 2010)

2. DDG-93 Ensemble Fuel Spread (Norfolk to Rota)

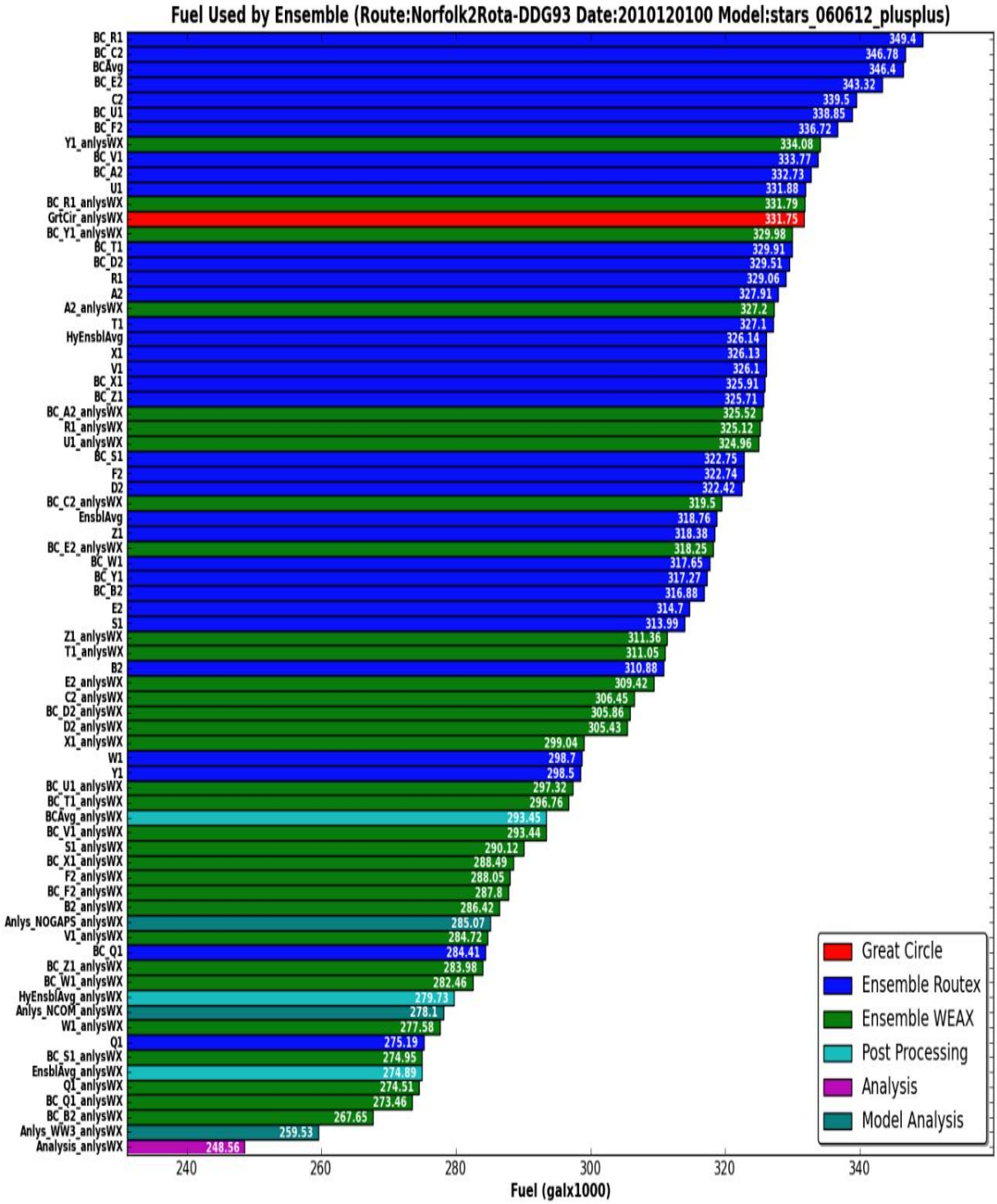


Figure 27. DDG-93 fuel used by ensemble spread distribution (Norfolk to Rota: 01 Dec 2010)

3. TAO-187 Ensemble Distance Spread (Norfolk to Rota)

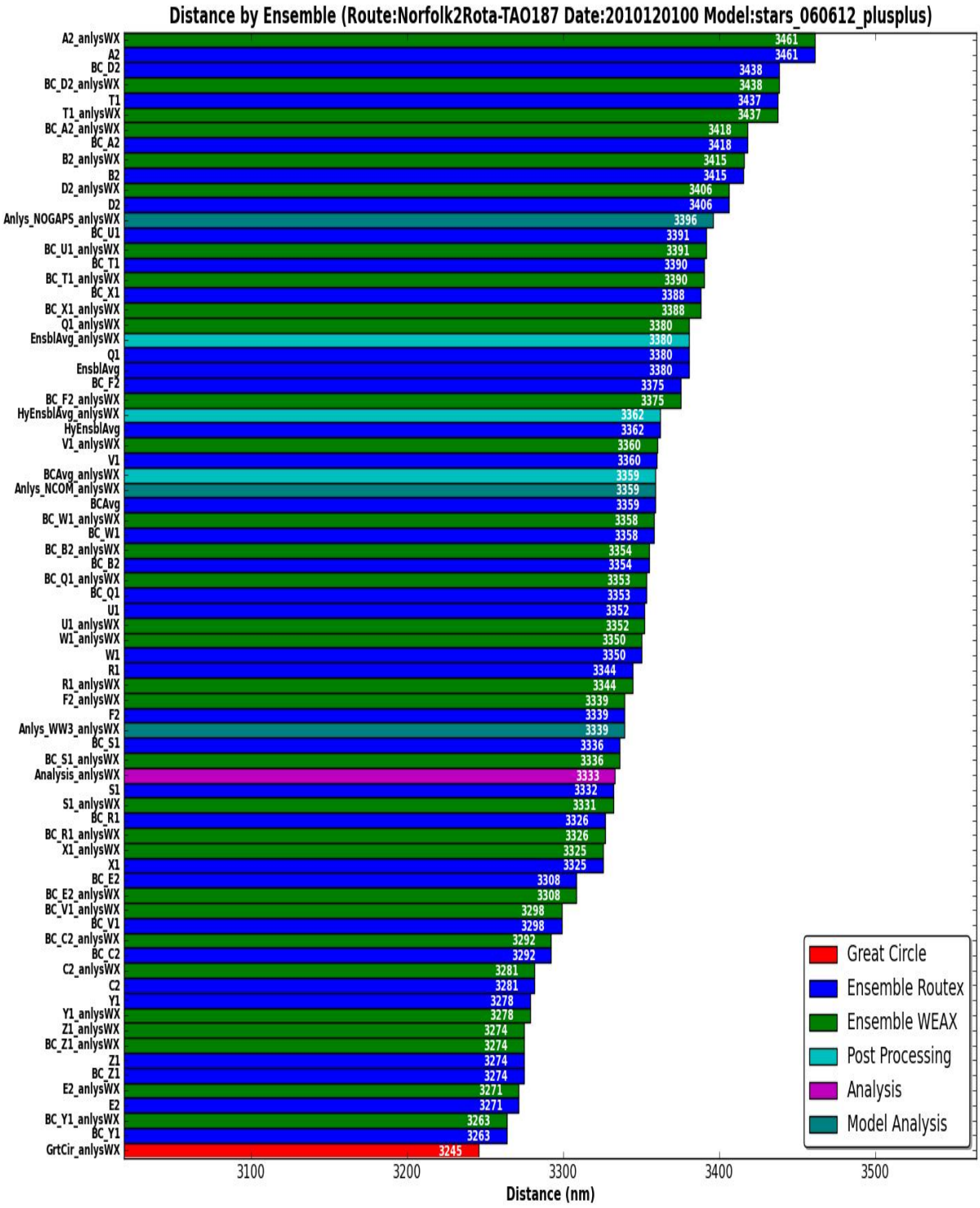


Figure 28. TAO-187 distance traveled ensemble spread distribution (Norfolk to Rota: 01 Dec 2010)

B. HISTOGRAMS

1. TAO-187 Fuel Histogram (Norfolk to Rota, 01 Dec 2010)

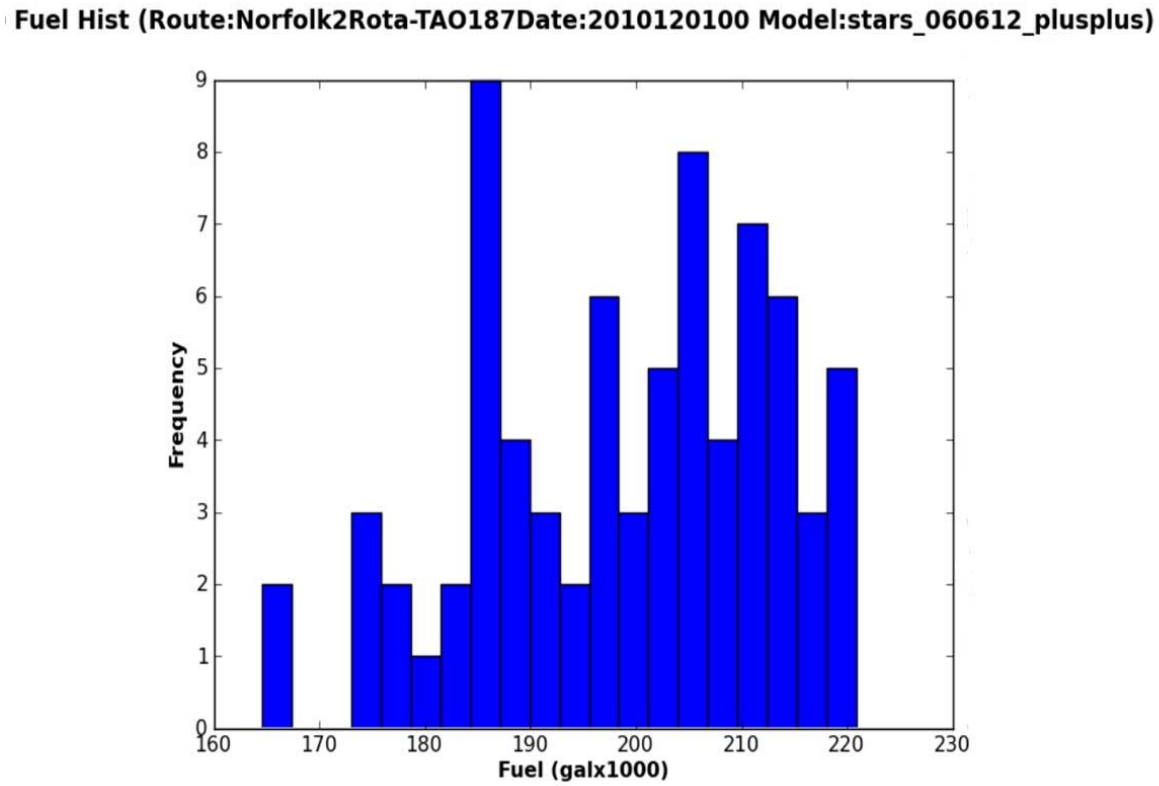


Figure 29. TAO-187 ensemble fuel histogram, combined model output predicted ensemble cost distributions and actual analysis environment ensemble route cost distributions (Norfolk to Rota: 01 Dec 2010)

2. TAO-187 Split Fuel Hist. (Norfolk to Rota, 01 Dec 2010)

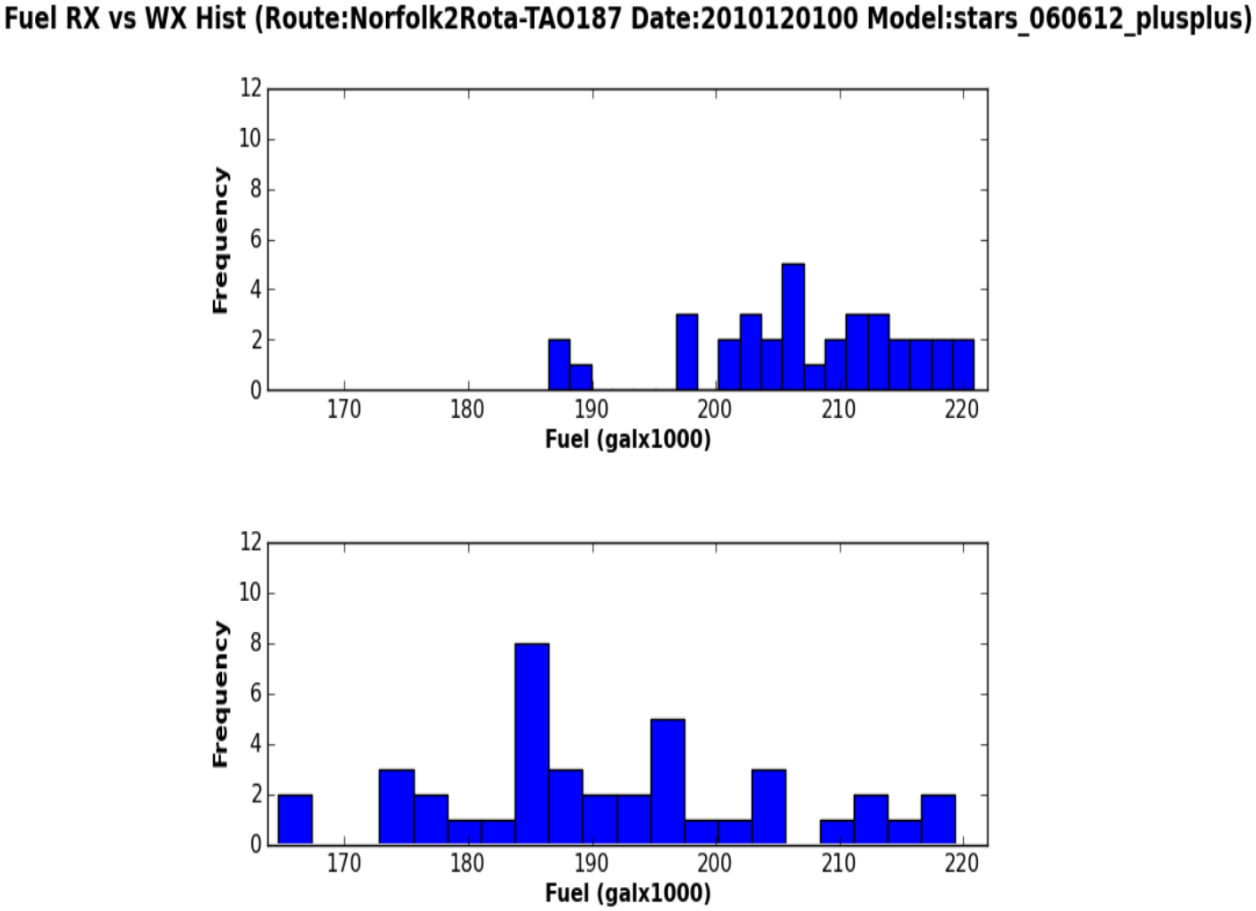


Figure 30. TAO-187 ensemble fuel histogram, model output ensemble predicted cost distributions in top pane and actual analysis environment ensemble route cost distributions in bottom pane (Norfolk to Rota: 01 Dec 2010)

3. TAO-187 Fuel Histogram (Norfolk to Rota, 01 Feb 2011)

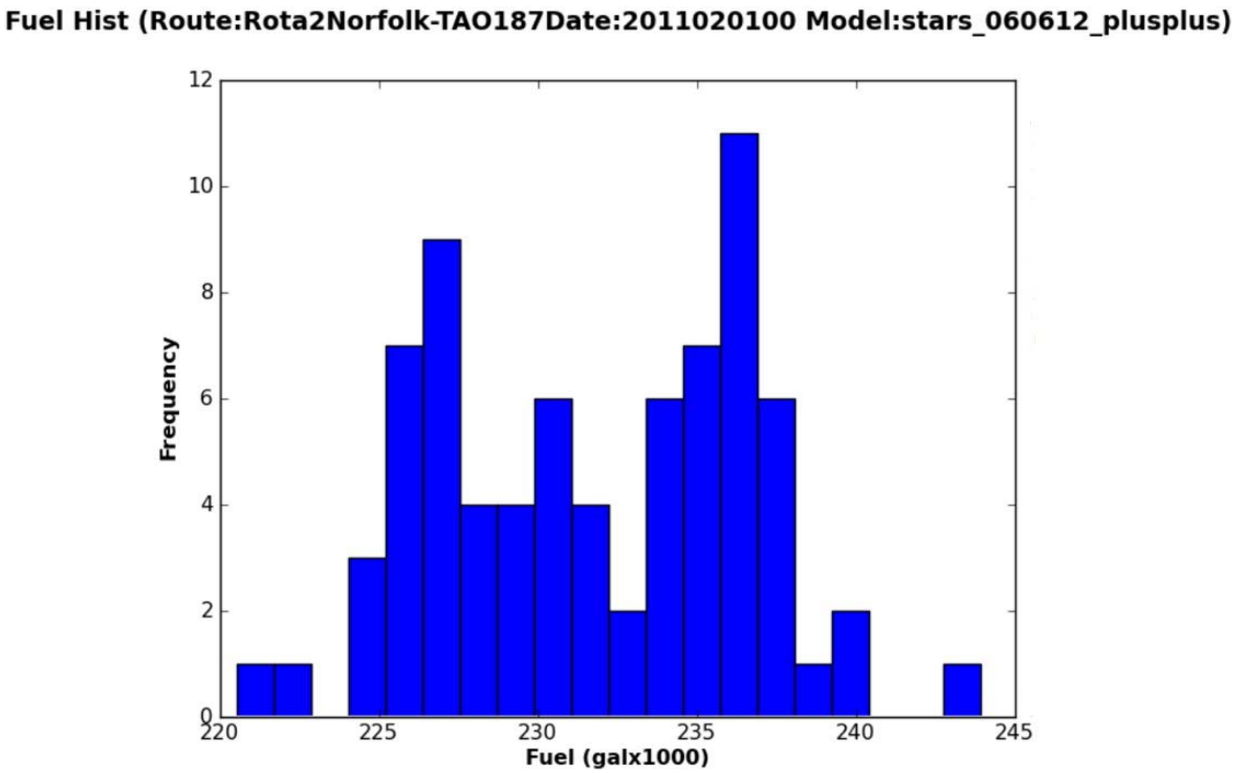


Figure 31. TAO-187 ensemble fuel histogram, combined model output predicted ensemble cost distributions and actual analysis environment ensemble route cost (Norfolk to Rota: 01 Feb 2011)

4. TAO-187 Split Fuel Hist. (Norfolk to Rota, 01 Feb 2011)

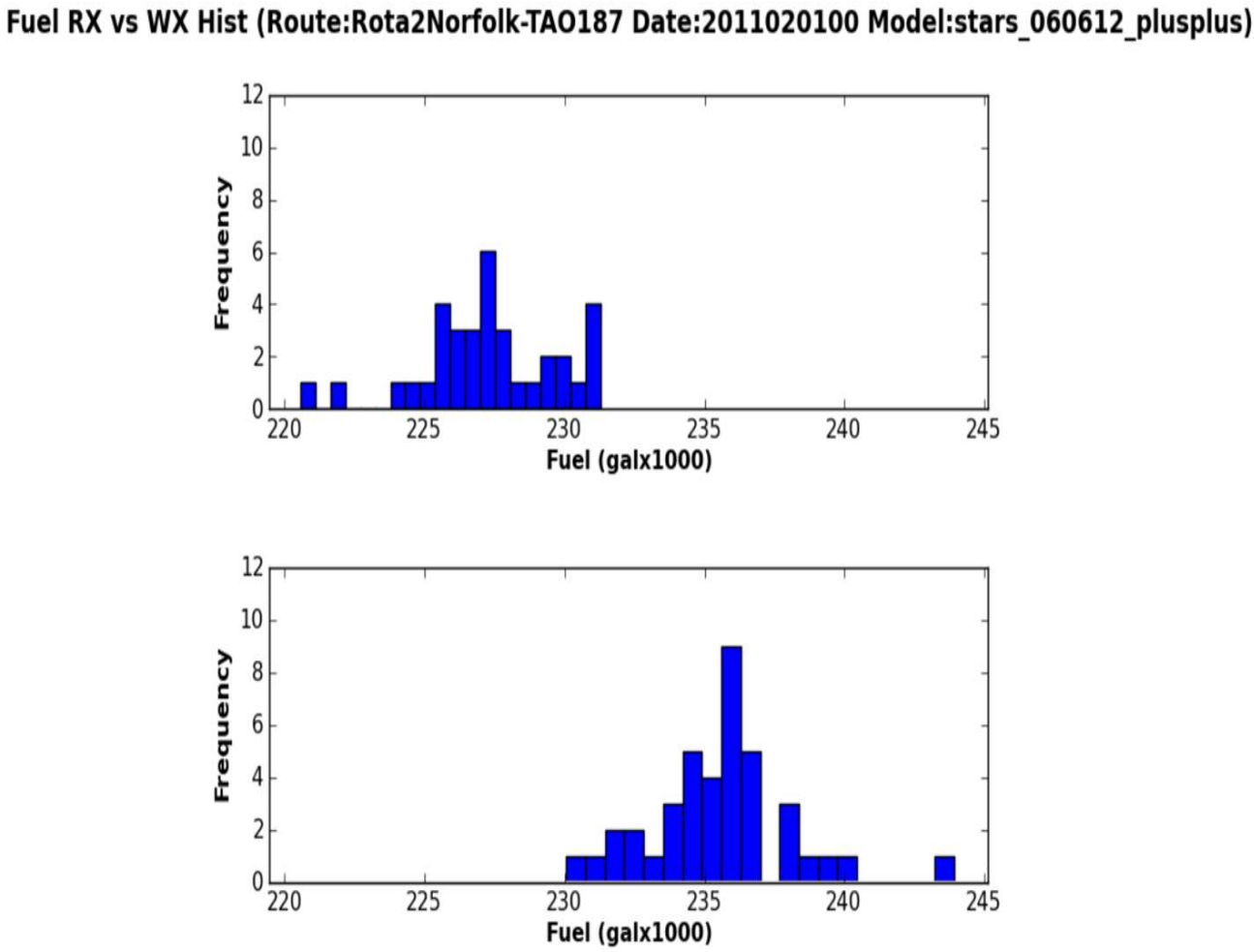


Figure 32. TAO-187 ensemble fuel histogram, model output ensemble predicted cost distributions in top pane and actual analysis environment ensemble route cost distributions in bottom pane (Norfolk to Rota: 01 Dec 2010)

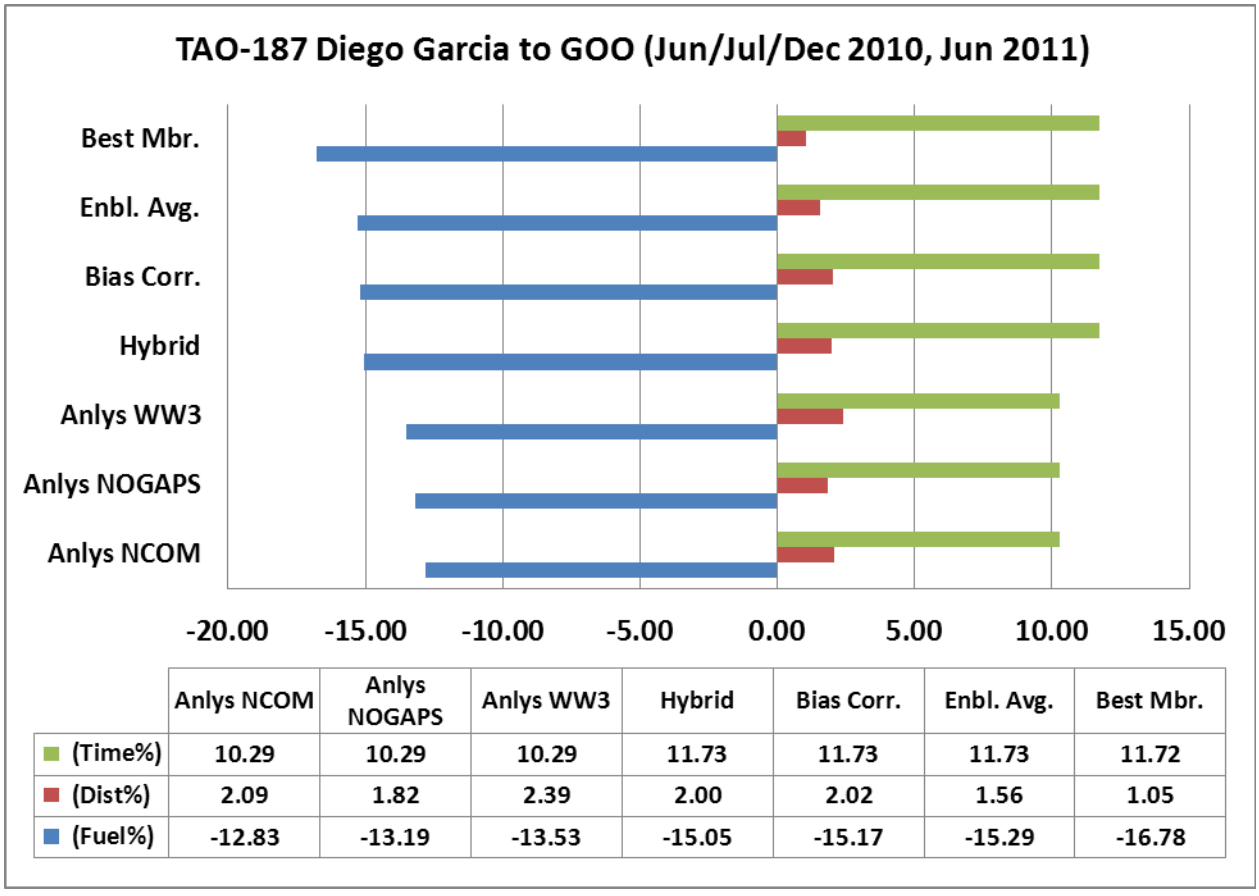
THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. DETAILED CASE SENSITIVITY RESULTS

Figures 33–50 identify the % fuel used (blue), distance traveled (red), and route duration (green) vs. the respective optimal speed Great Circle route parameters for each class tested.

A. CASE 1

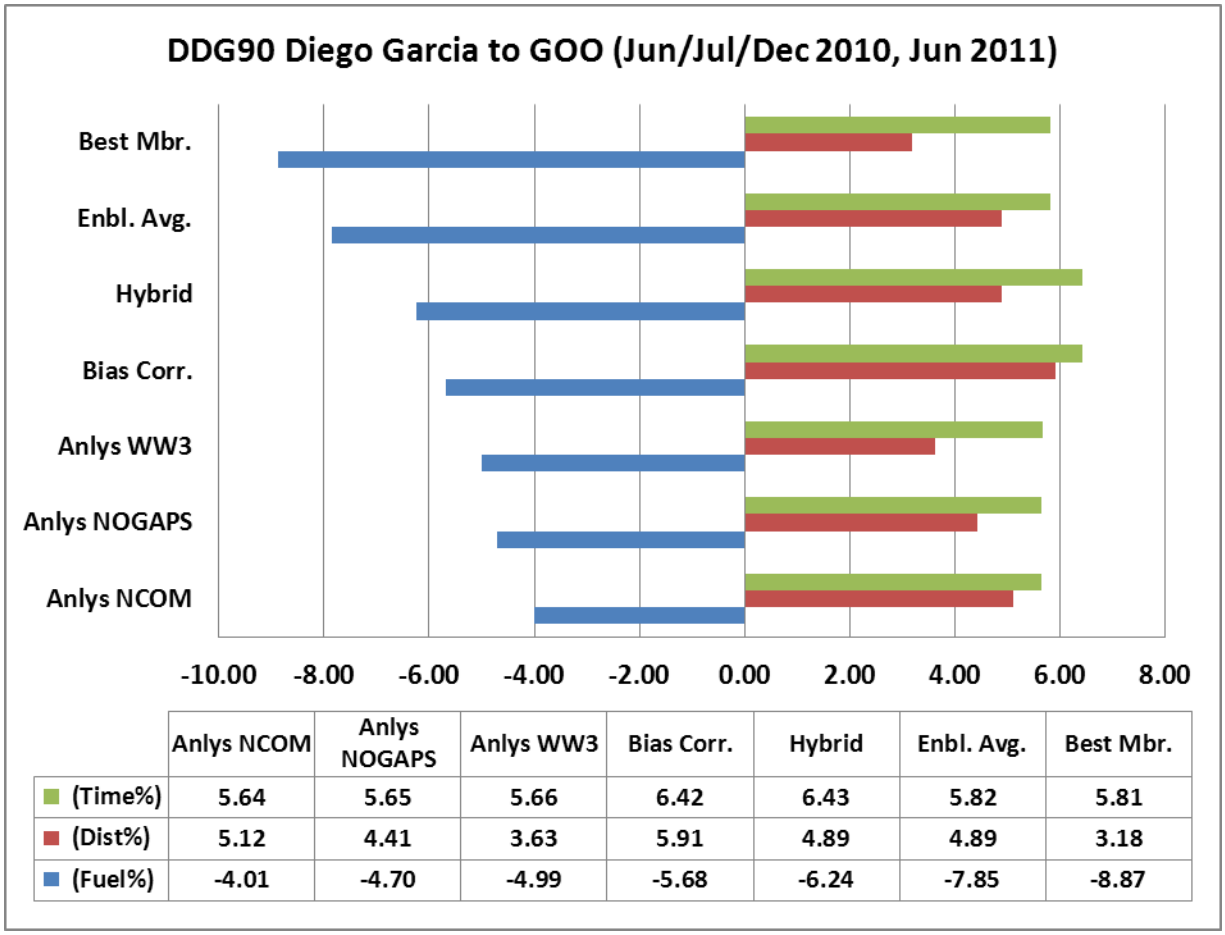
1. TAO-187 Diego Garcia to Gulf of Oman



	Anlys NCOM	Anlys NOGAPS	Anlys WW3	Hybrid	Bias Corr.	Enbl. Avg.	Best Mbr.
(Time%)	10.29	10.29	10.29	11.73	11.73	11.73	11.72
(Dist%)	2.09	1.82	2.39	2.00	2.02	1.56	1.05
(Fuel%)	-12.83	-13.19	-13.53	-15.05	-15.17	-15.29	-16.78

Figure 33. TAO-187 Diego Garcia to GOO ensemble % time/dist./fuel vs. GC; ensemble rank order sorted based on % fuel savings; negative #'s denote % less than GC and positive #'s % greater than GC (Jun/Jul/Dec 2010, Jun 2011)

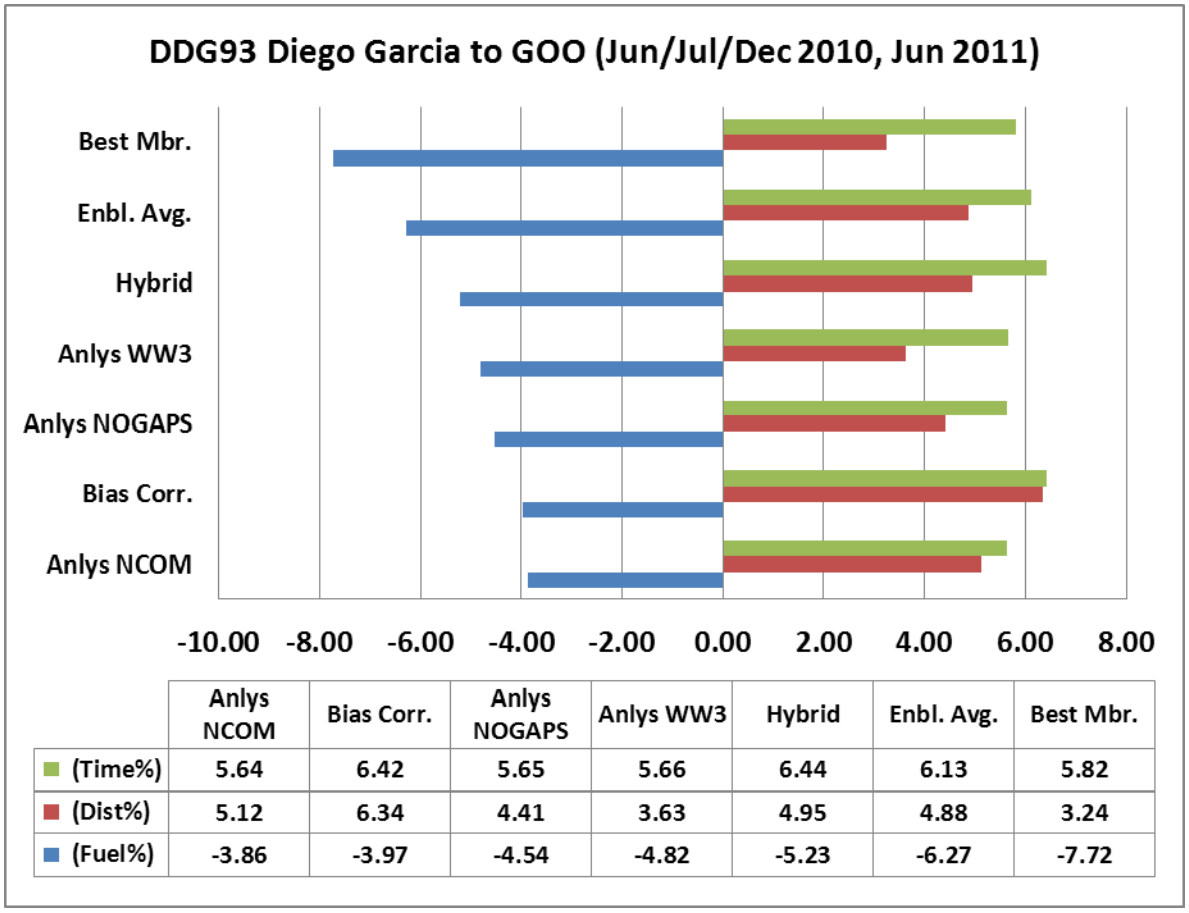
2. DDG-90 Diego Garcia to Gulf of Oman



	Anlys NCOM	Anlys NOGAPS	Anlys WW3	Bias Corr.	Hybrid	Enbl. Avg.	Best Mbr.
(Time%)	5.64	5.65	5.66	6.42	6.43	5.82	5.81
(Dist%)	5.12	4.41	3.63	5.91	4.89	4.89	3.18
(Fuel%)	-4.01	-4.70	-4.99	-5.68	-6.24	-7.85	-8.87

Figure 34. DDG90 Diego Garcia to Gulf of Oman ensemble % time/dist./fuel vs. GC (Jun/Jul/Dec 2010, Jun 2011)

3. DDG-93 Diego Garcia to Gulf of Oman

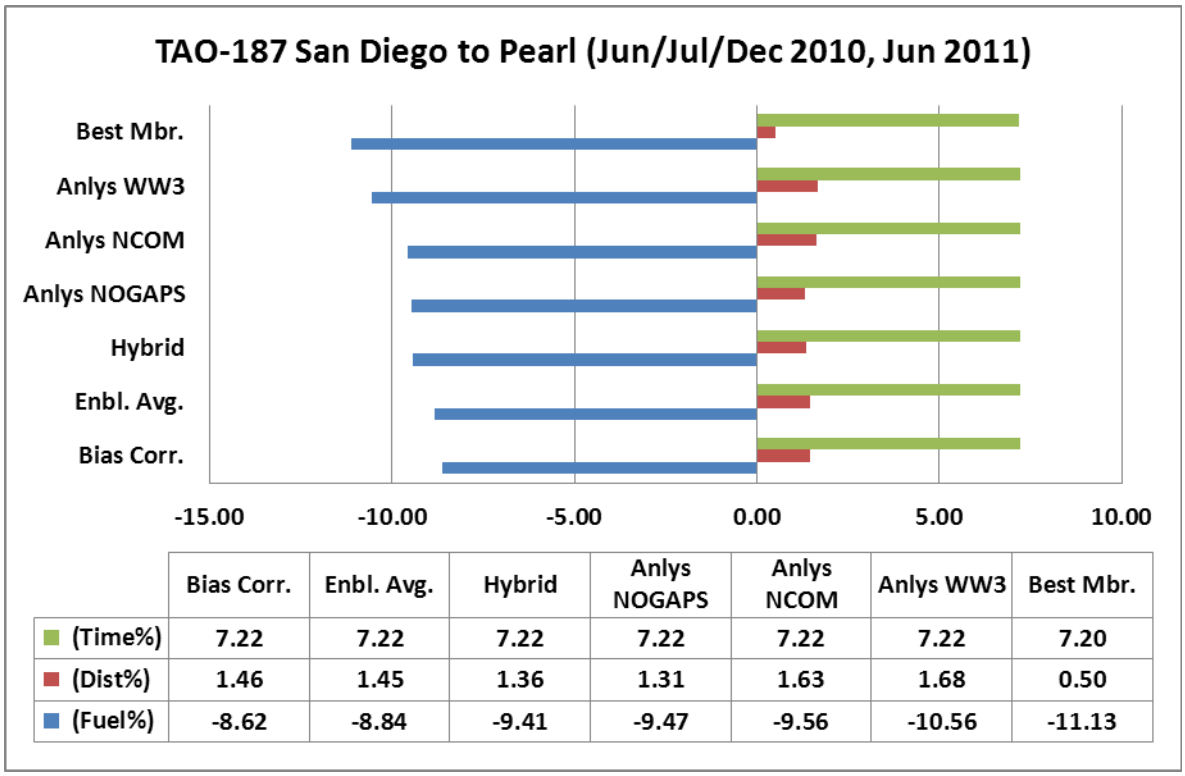


	Anlys NCOM	Bias Corr.	Anlys NOGAPS	Anlys WW3	Hybrid	Enbl. Avg.	Best Mbr.
(Time%)	5.64	6.42	5.65	5.66	6.44	6.13	5.82
(Dist%)	5.12	6.34	4.41	3.63	4.95	4.88	3.24
(Fuel%)	-3.86	-3.97	-4.54	-4.82	-5.23	-6.27	-7.72

Figure 35. DDG93 Diego Garcia to Gulf of Oman ensemble % time/dist./fuel vs. GC (Jun/Jul/Dec 2010, Jun 2011)

B. CASE 2

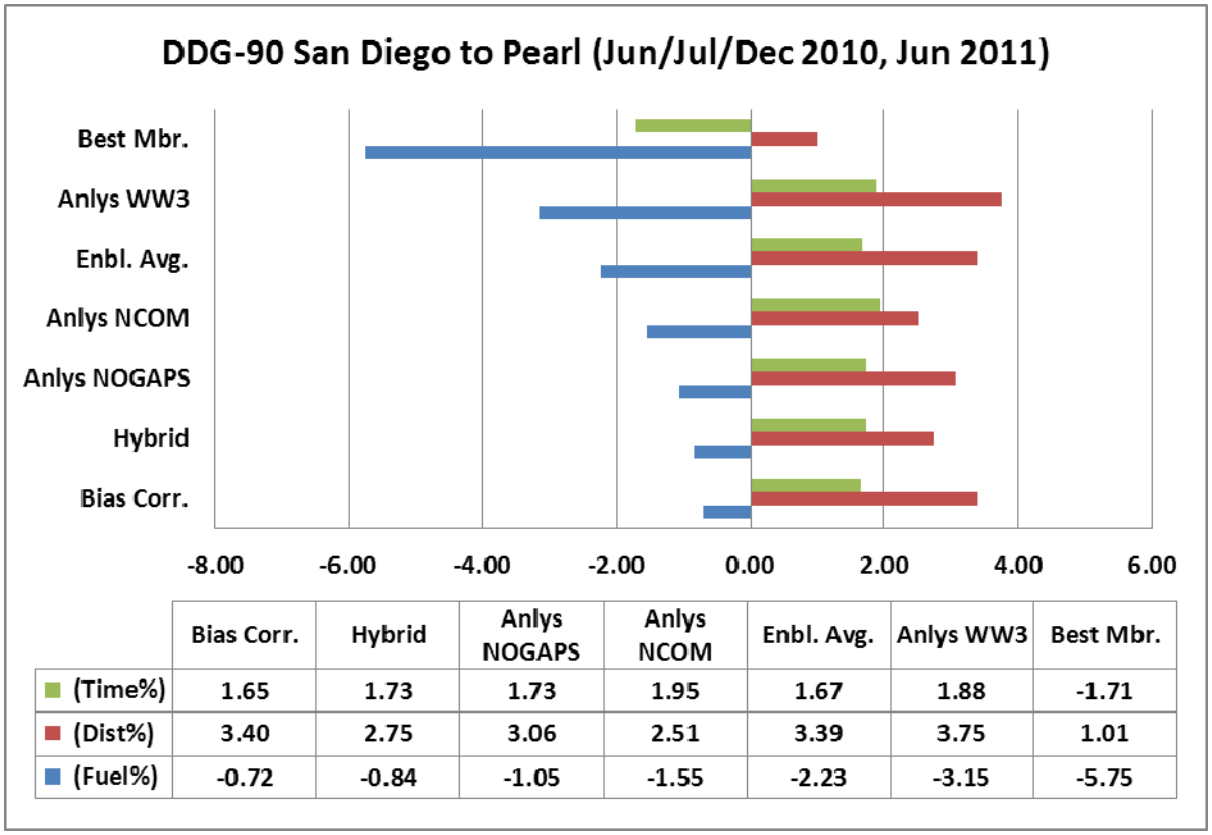
1. TAO-187 San Diego to Pearl Harbor



	Bias Corr.	Enbl. Avg.	Hybrid	Anlys NOGAPS	Anlys NCOM	Anlys WW3	Best Mbr.
(Time%)	7.22	7.22	7.22	7.22	7.22	7.22	7.20
(Dist%)	1.46	1.45	1.36	1.31	1.63	1.68	0.50
(Fuel%)	-8.62	-8.84	-9.41	-9.47	-9.56	-10.56	-11.13

Figure 36. TAO-187 San Diego to Pearl ensemble % time/dist./fuel vs. GC (Jun/Jul/Dec 2010, Jun 2011)

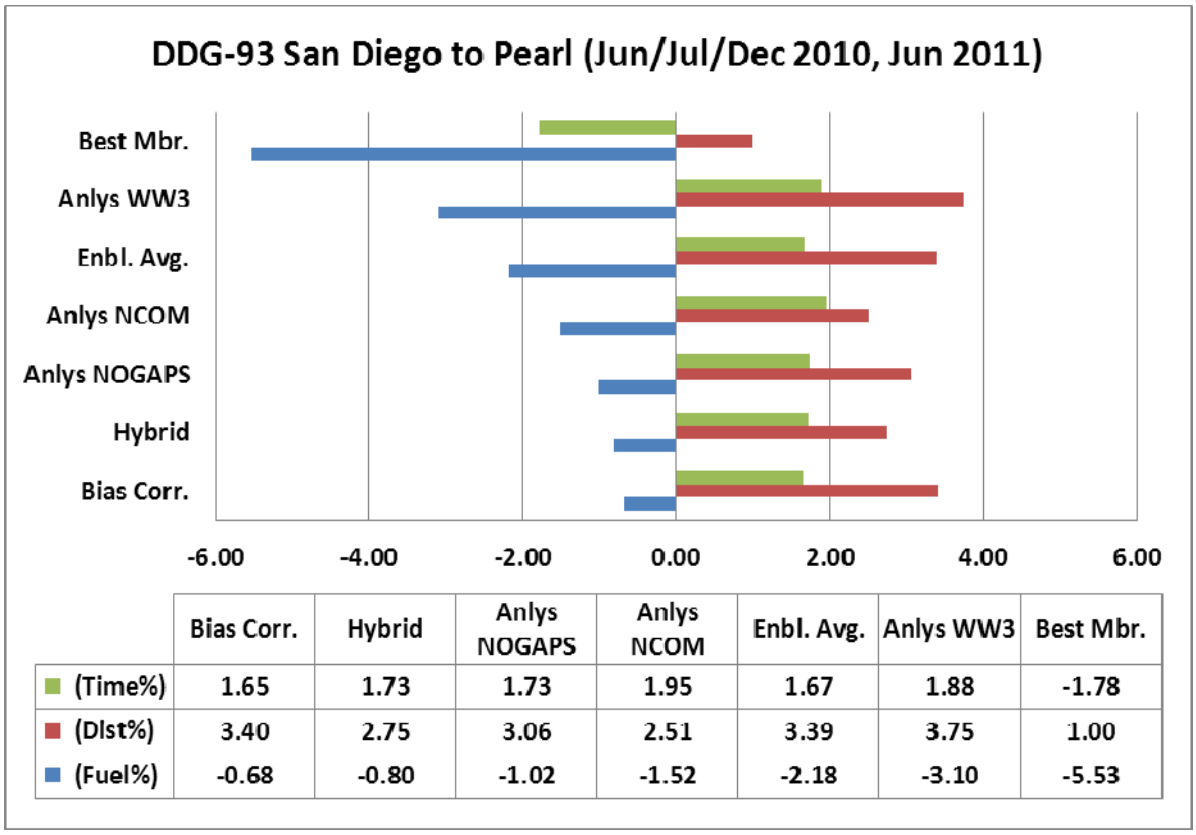
2. DDG-90 San Diego to Pearl Harbor



	Bias Corr.	Hybrid	Anlys NOGAPS	Anlys NCOM	Enbl. Avg.	Anlys WW3	Best Mbr.
(Time%)	1.65	1.73	1.73	1.95	1.67	1.88	-1.71
(Dist%)	3.40	2.75	3.06	2.51	3.39	3.75	1.01
(Fuel%)	-0.72	-0.84	-1.05	-1.55	-2.23	-3.15	-5.75

Figure 37. DDG-90 San Diego to Pearl ensemble % time/dist./fuel vs. GC (Jun/Jul/Dec 2010, Jun 2011)

3. DDG-93 San Diego to Pearl Harbor

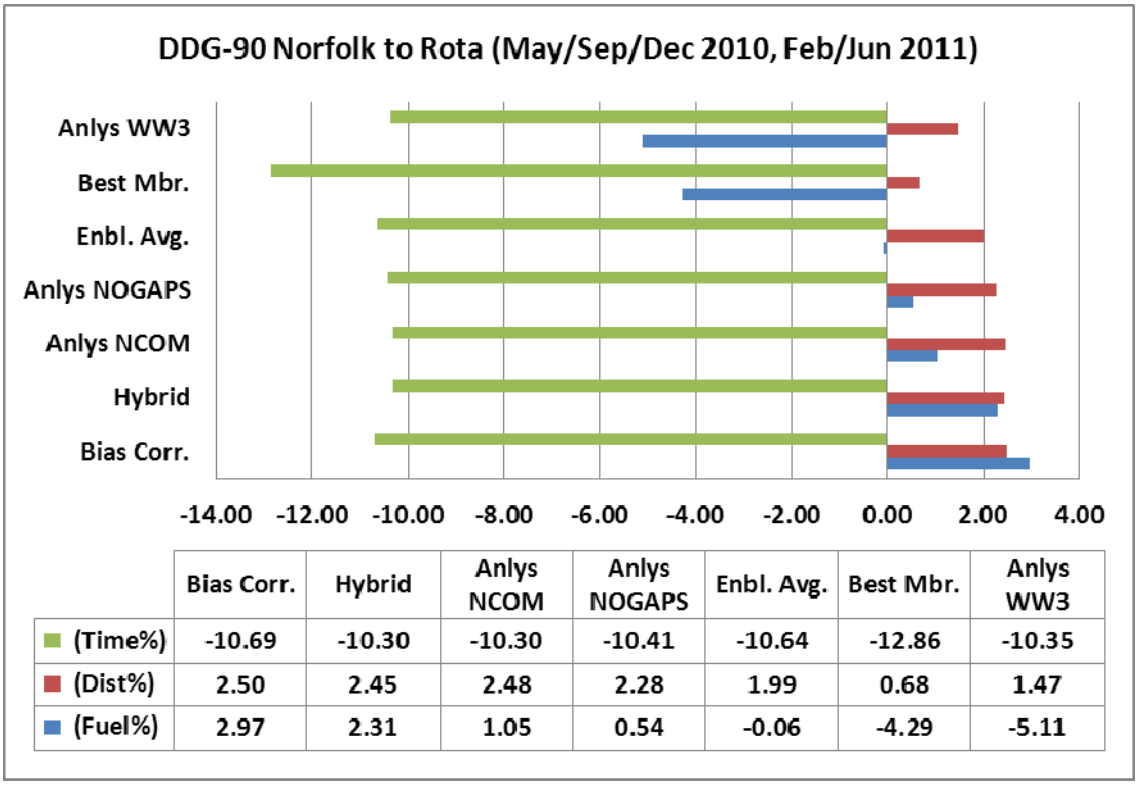


	Bias Corr.	Hybrid	Anlys NOGAPS	Anlys NCOM	Enbl. Avg.	Anlys WW3	Best Mbr.
(Time%)	1.65	1.73	1.73	1.95	1.67	1.88	-1.78
(Dist%)	3.40	2.75	3.06	2.51	3.39	3.75	1.00
(Fuel%)	-0.68	-0.80	-1.02	-1.52	-2.18	-3.10	-5.53

Figure 38. DDG-93 San Diego to Pearl ensemble % time/dist./fuel vs. GC (Jun/Jul/Dec 2010, Jun 2011)

C. CASE 3

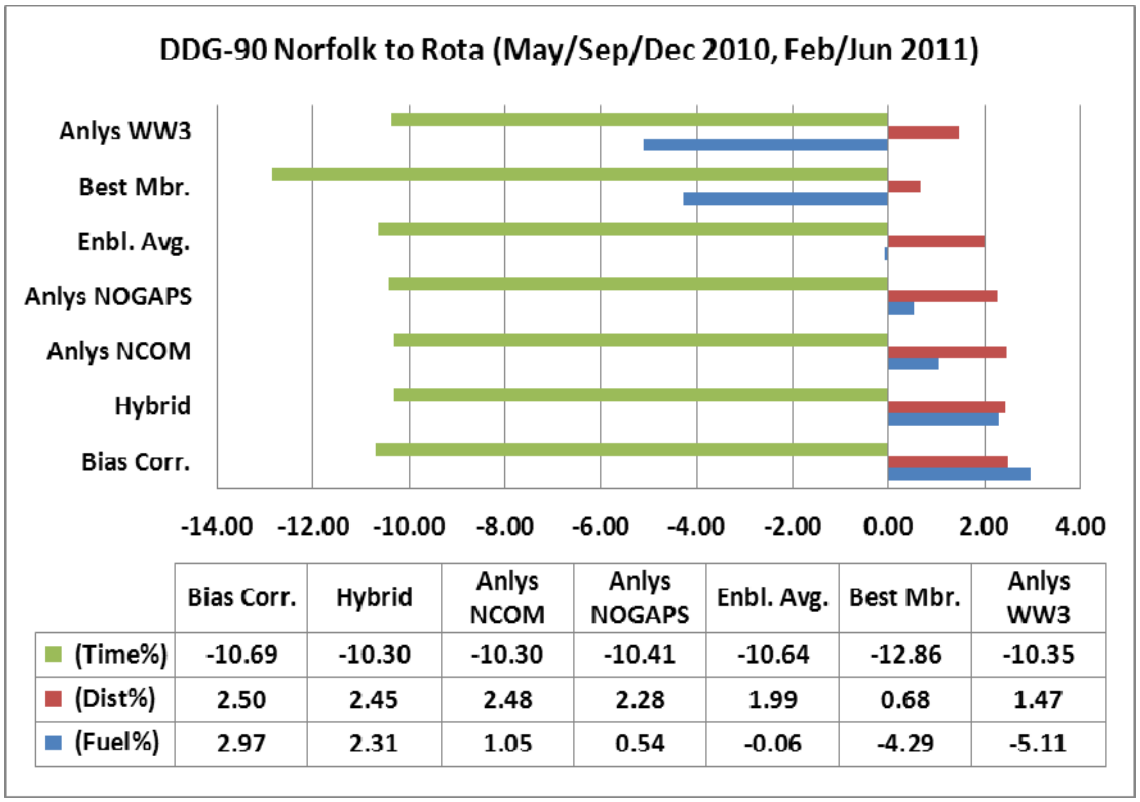
1. TAO-187 Norfolk to Rota



	Bias Corr.	Hybrid	Anlys NCOM	Anlys NOGAPS	Enbl. Avg.	Best Mbr.	Anlys WW3
(Time%)	-10.69	-10.30	-10.30	-10.41	-10.64	-12.86	-10.35
(Dist%)	2.50	2.45	2.48	2.28	1.99	0.68	1.47
(Fuel%)	2.97	2.31	1.05	0.54	-0.06	-4.29	-5.11

Figure 39. DDG-90 Norfolk to Rota ensemble % time/dist./fuel vs. GC (May/Sep/Dec 2010, Feb/Jun 2011)

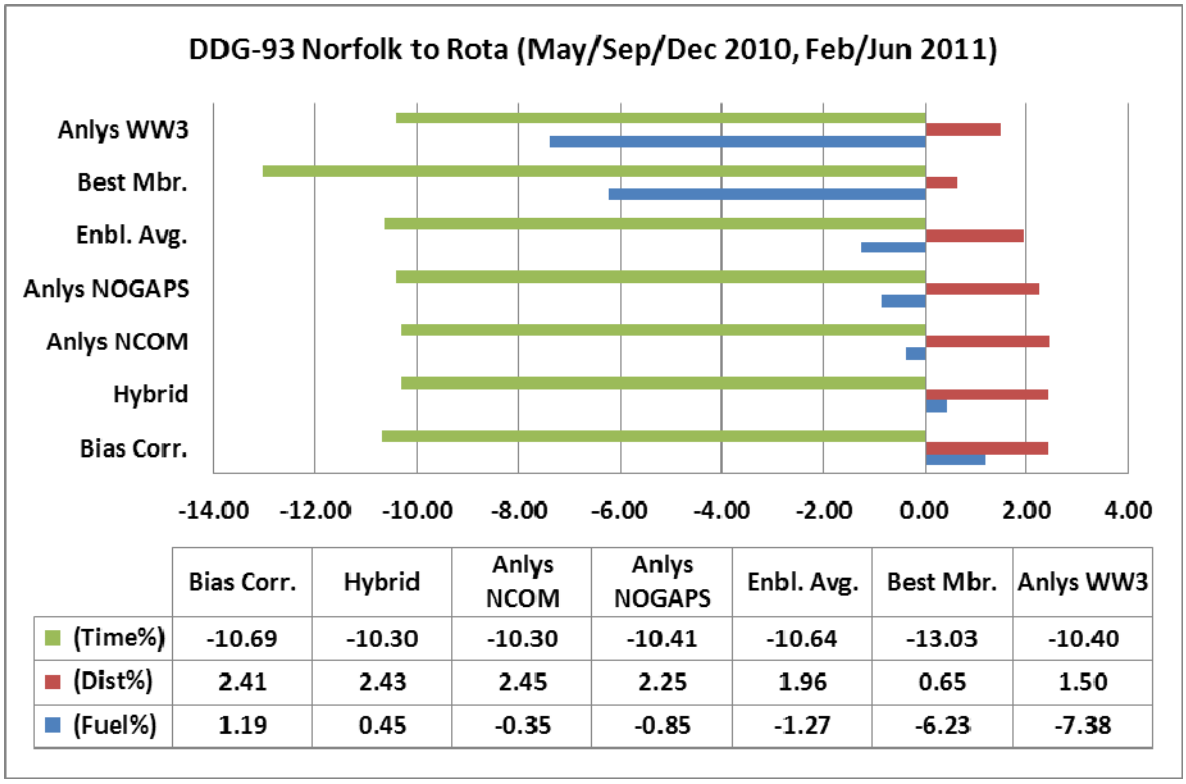
2. DDG-90 Norfolk to Rota



	Bias Corr.	Hybrid	Anlys NCOM	Anlys NOGAPS	Enbl. Avg.	Best Mbr.	Anlys WW3
(Time%)	-10.69	-10.30	-10.30	-10.41	-10.64	-12.86	-10.35
(Dist%)	2.50	2.45	2.48	2.28	1.99	0.68	1.47
(Fuel%)	2.97	2.31	1.05	0.54	-0.06	-4.29	-5.11

Figure 40. DDG-90 Norfolk to Rota ensemble % time/dist./fuel vs. GC (May/Sep/Dec 2010, Feb/Jun 2011)

3. DDG-93 Norfolk to Rota

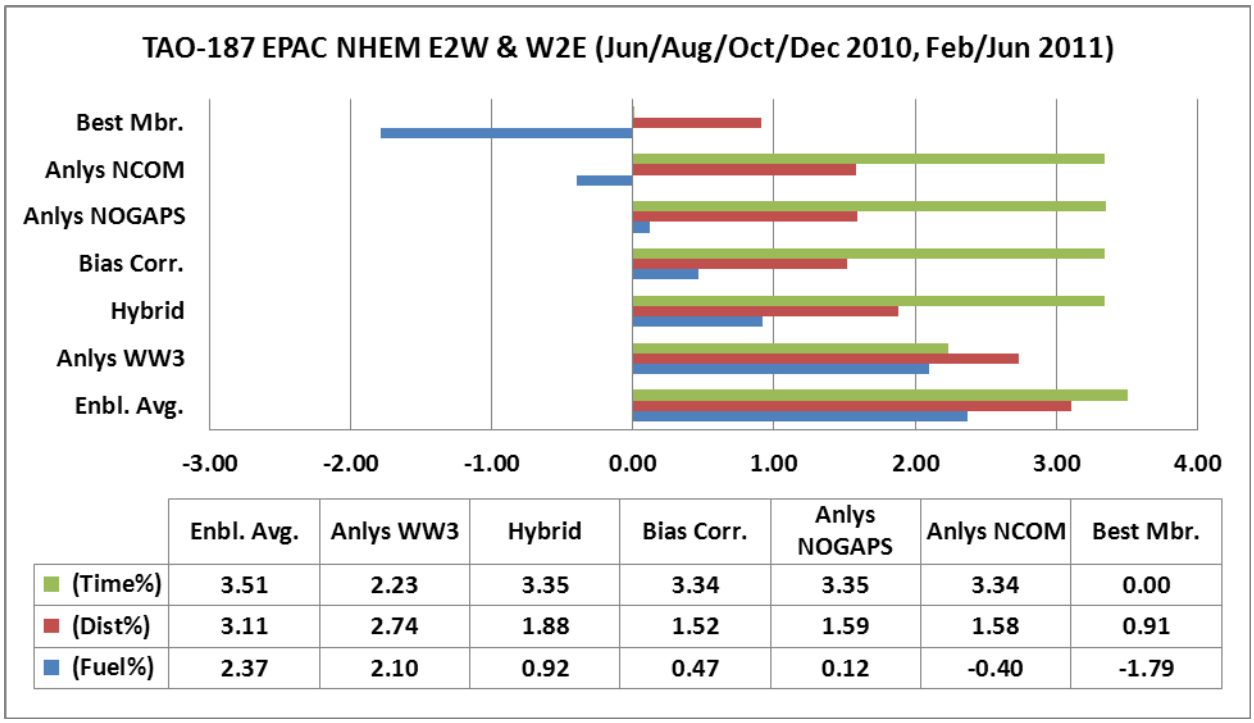


	Bias Corr.	Hybrid	Anlys NCOM	Anlys NOGAPS	Enbl. Avg.	Best Mbr.	Anlys WW3
(Time%)	-10.69	-10.30	-10.30	-10.41	-10.64	-13.03	-10.40
(Dist%)	2.41	2.43	2.45	2.25	1.96	0.65	1.50
(Fuel%)	1.19	0.45	-0.35	-0.85	-1.27	-6.23	-7.38

Figure 41. DDG-93 Norfolk to Rota ensemble % time/dist./fuel vs. GC (May/Sep/Dec 2010, Feb/Jun 2011)

D. CASE 4

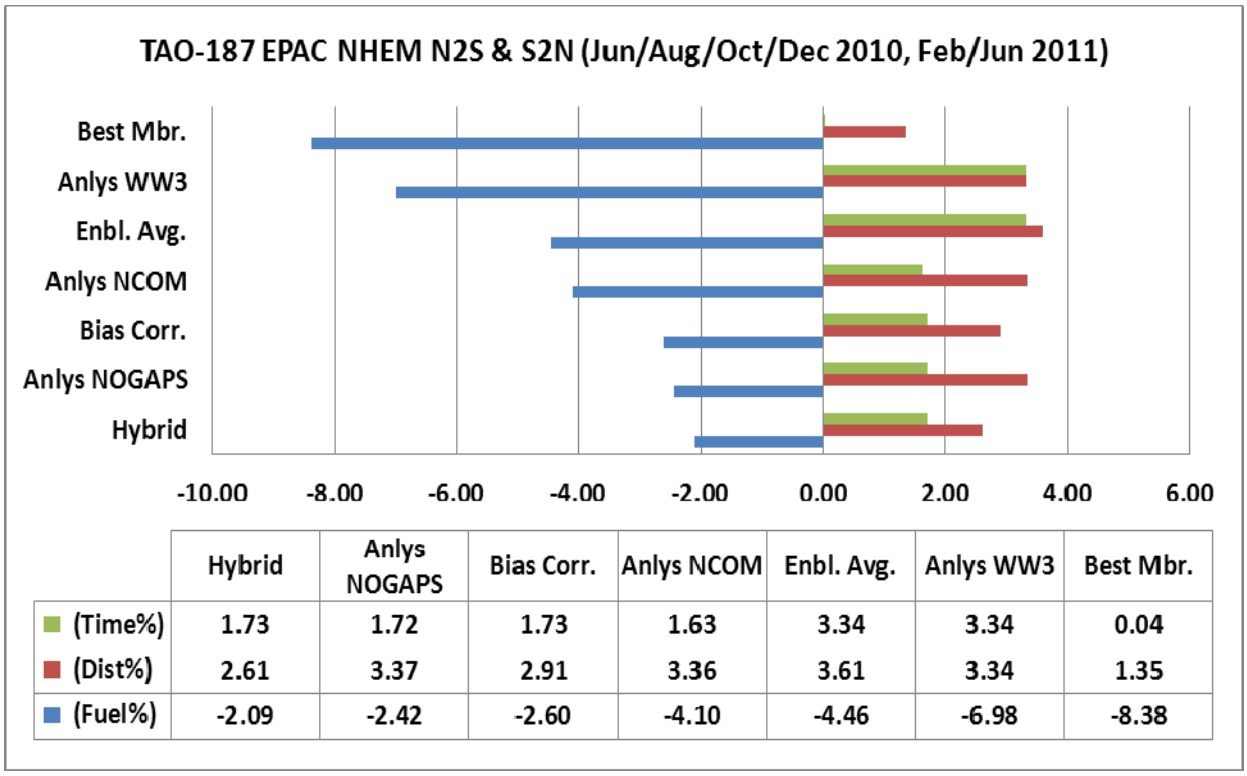
1. Eastern Pacific Northern Hemisphere (Latitude Parallel)



	Enbl. Avg.	Anlys WW3	Hybrid	Bias Corr.	Anlys NOGAPS	Anlys NCOM	Best Mbr.
(Time%)	3.51	2.23	3.35	3.34	3.35	3.34	0.00
(Dist%)	3.11	2.74	1.88	1.52	1.59	1.58	0.91
(Fuel%)	2.37	2.10	0.92	0.47	0.12	-0.40	-1.79

Figure 42. TAO-187 Eastern Pacific ocean east to west & west to east ensemble % time/dist./fuel vs. GC (Jun/Aug/Oct/Dec 2010, Feb/Jun 2011)

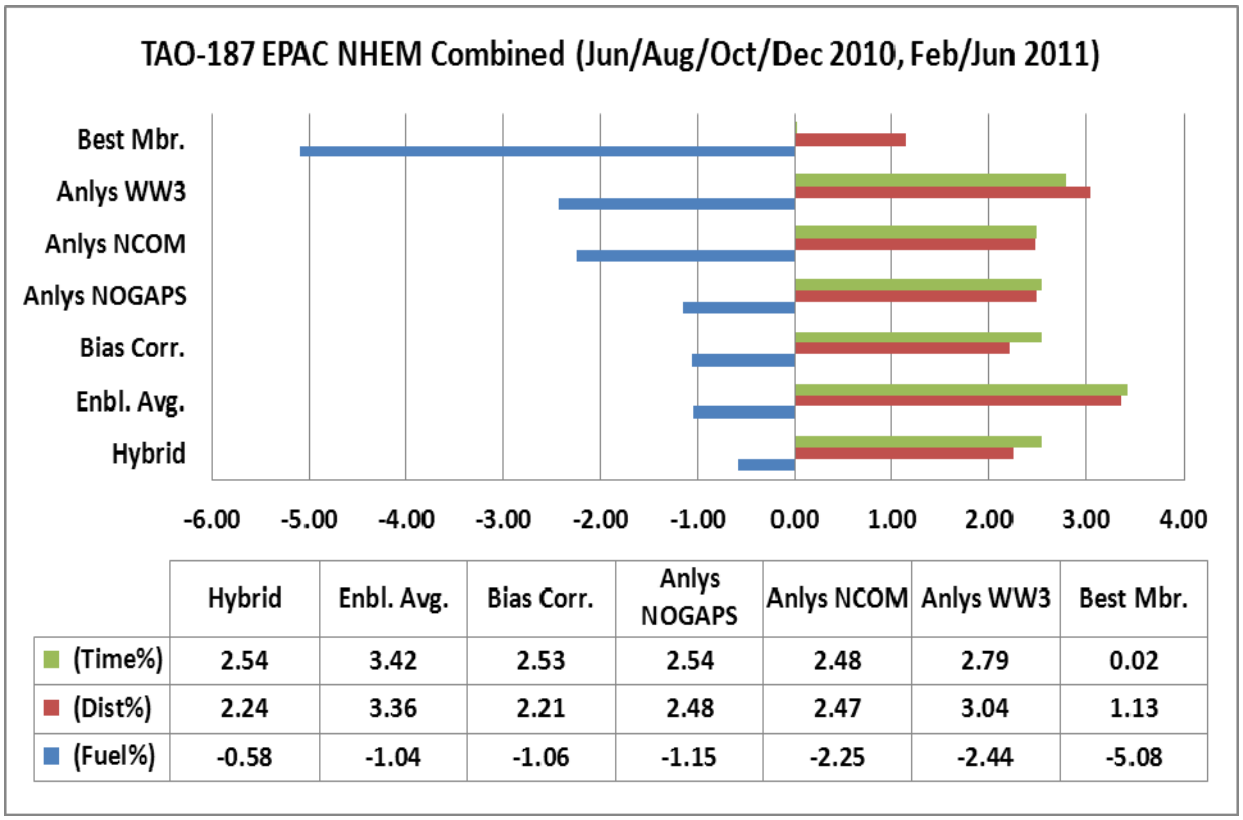
2. Eastern Pacific Northern Hemisphere (Longitude Parallel)



	Hybrid	Anlys NOGAPS	Bias Corr.	Anlys NCOM	Enbl. Avg.	Anlys WW3	Best Mbr.
(Time%)	1.73	1.72	1.73	1.63	3.34	3.34	0.04
(Dist%)	2.61	3.37	2.91	3.36	3.61	3.34	1.35
(Fuel%)	-2.09	-2.42	-2.60	-4.10	-4.46	-6.98	-8.38

Figure 43. TAO-187 EPAC NHEM north to south & south to north ensemble % time/dist./fuel vs. GC (Jun/Aug/Oct/Dec 2010, Feb/Jun 2011)

3. Eastern Pacific Northern Hemisphere (Combined Results)

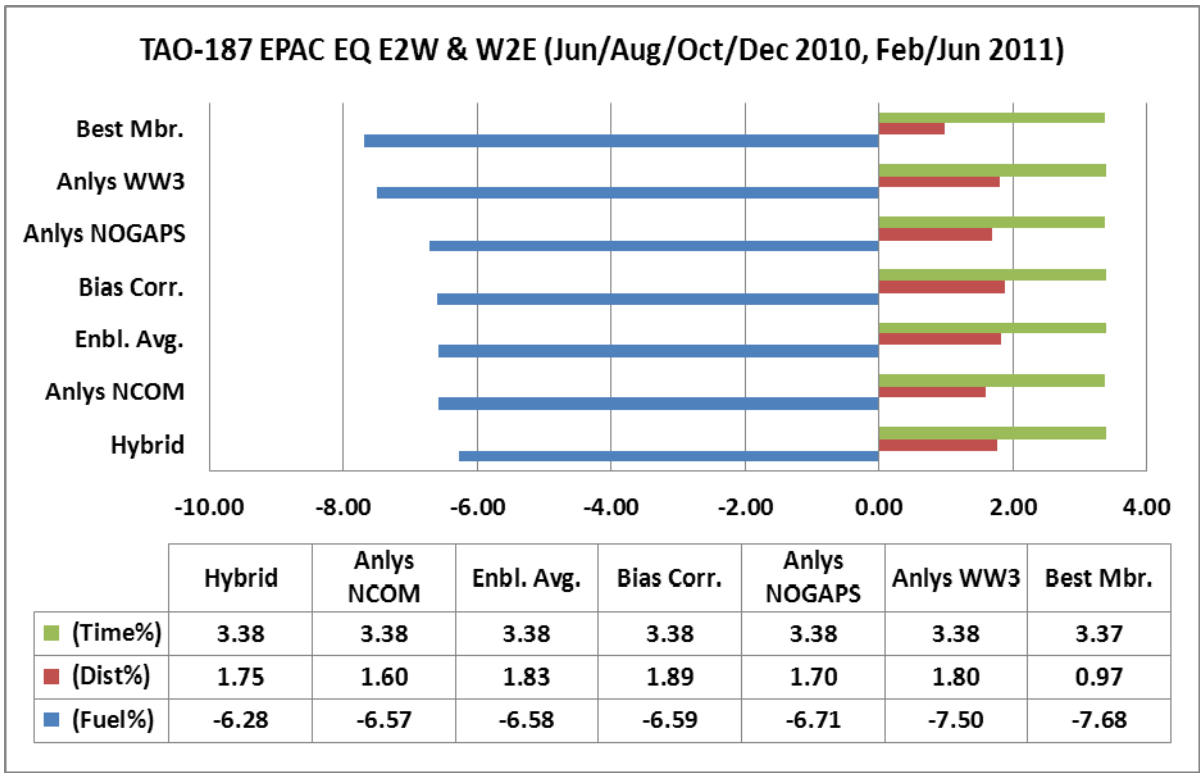


	Hybrid	Enbl. Avg.	Bias Corr.	Anlys NOGAPS	Anlys NCOM	Anlys WW3	Best Mbr.
(Time%)	2.54	3.42	2.53	2.54	2.48	2.79	0.02
(Dist%)	2.24	3.36	2.21	2.48	2.47	3.04	1.13
(Fuel%)	-0.58	-1.04	-1.06	-1.15	-2.25	-2.44	-5.08

Figure 44. TAO-187 EPAC NHEM combined directions ensemble % time/dist./fuel vs. GC (Jun/Aug/Oct/Dec 2010, Feb/Jun 2011)

E. CASE 5

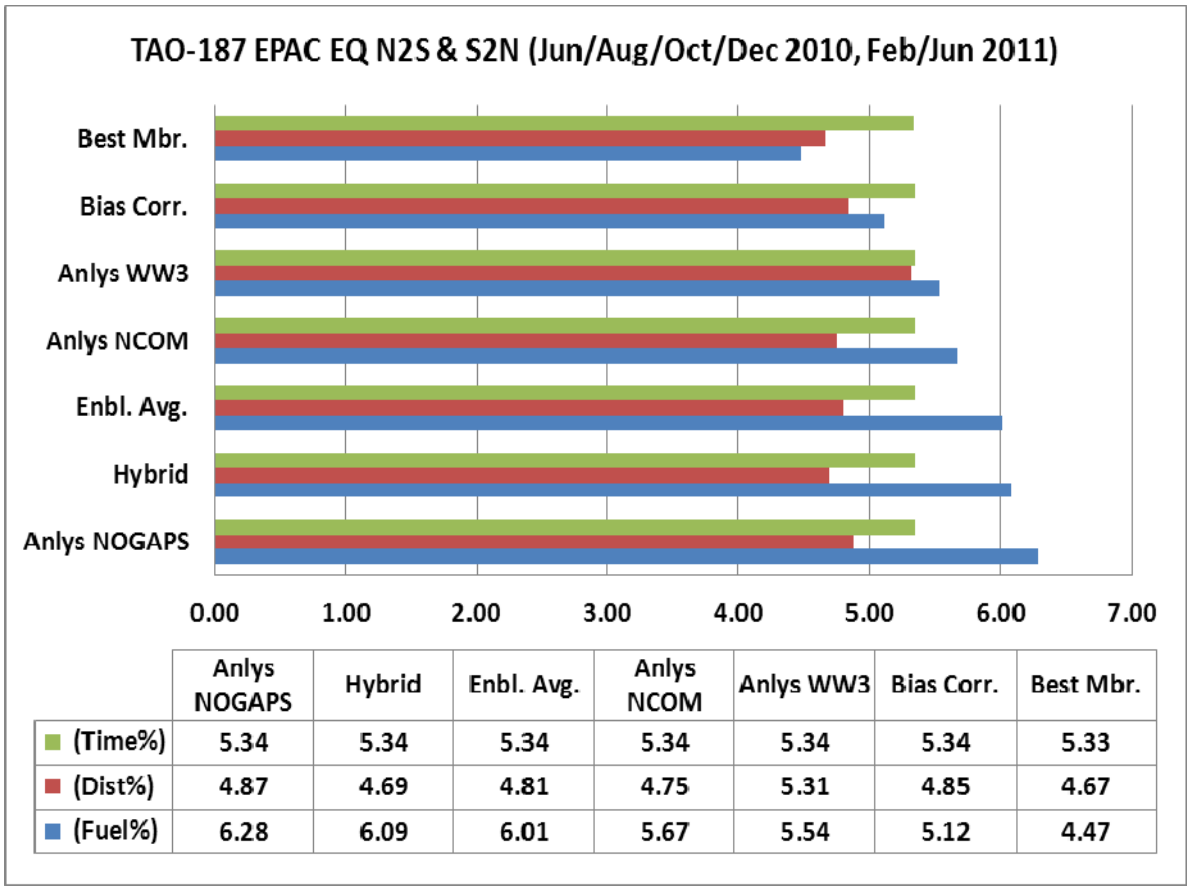
1. Eastern Pacific Equatorial (Latitude Parallel)



	Hybrid	Anlys NCOM	Enbl. Avg.	Bias Corr.	Anlys NOGAPS	Anlys WW3	Best Mbr.
(Time%)	3.38	3.38	3.38	3.38	3.38	3.38	3.37
(Dist%)	1.75	1.60	1.83	1.89	1.70	1.80	0.97
(Fuel%)	-6.28	-6.57	-6.58	-6.59	-6.71	-7.50	-7.68

Figure 45. TAO-187 EPAC EQ ocean east to west & west to east ensemble % time/dist./fuel vs. GC (Jun/Aug/Oct/Dec 2010, Feb/Jun 2011)

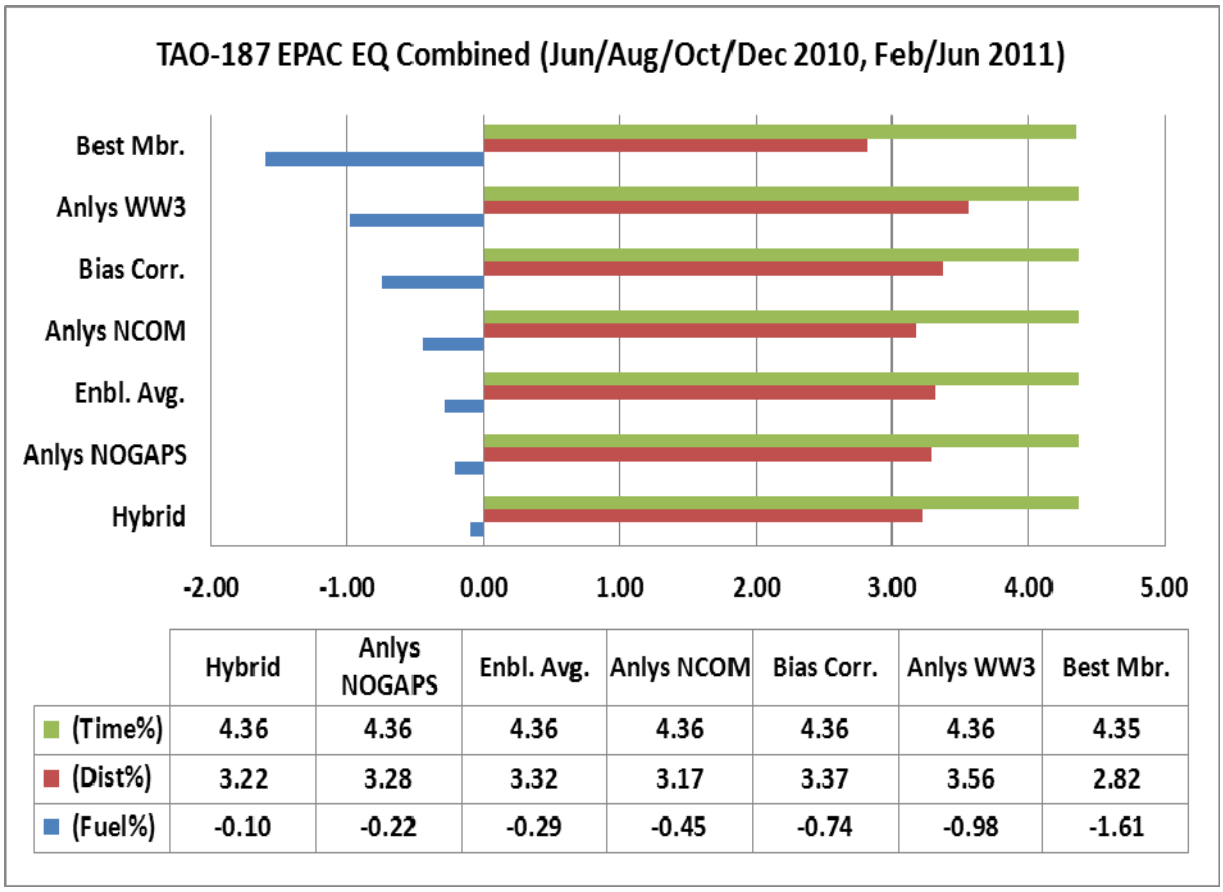
2. Eastern Pacific Equatorial (Longitude Parallel)



	Anlys NOGAPS	Hybrid	Enbl. Avg.	Anlys NCOM	Anlys WW3	Bias Corr.	Best Mbr.
(Time%)	5.34	5.34	5.34	5.34	5.34	5.34	5.33
(Dist%)	4.87	4.69	4.81	4.75	5.31	4.85	4.67
(Fuel%)	6.28	6.09	6.01	5.67	5.54	5.12	4.47

Figure 46. TAO-187 EPAC equator north to south & south to north ensemble % time/dist./fuel vs. GC (Jun/Aug/Oct/Dec 2010, Feb/Jun 2011)

3. Eastern Pacific Equatorial (Combined Results)

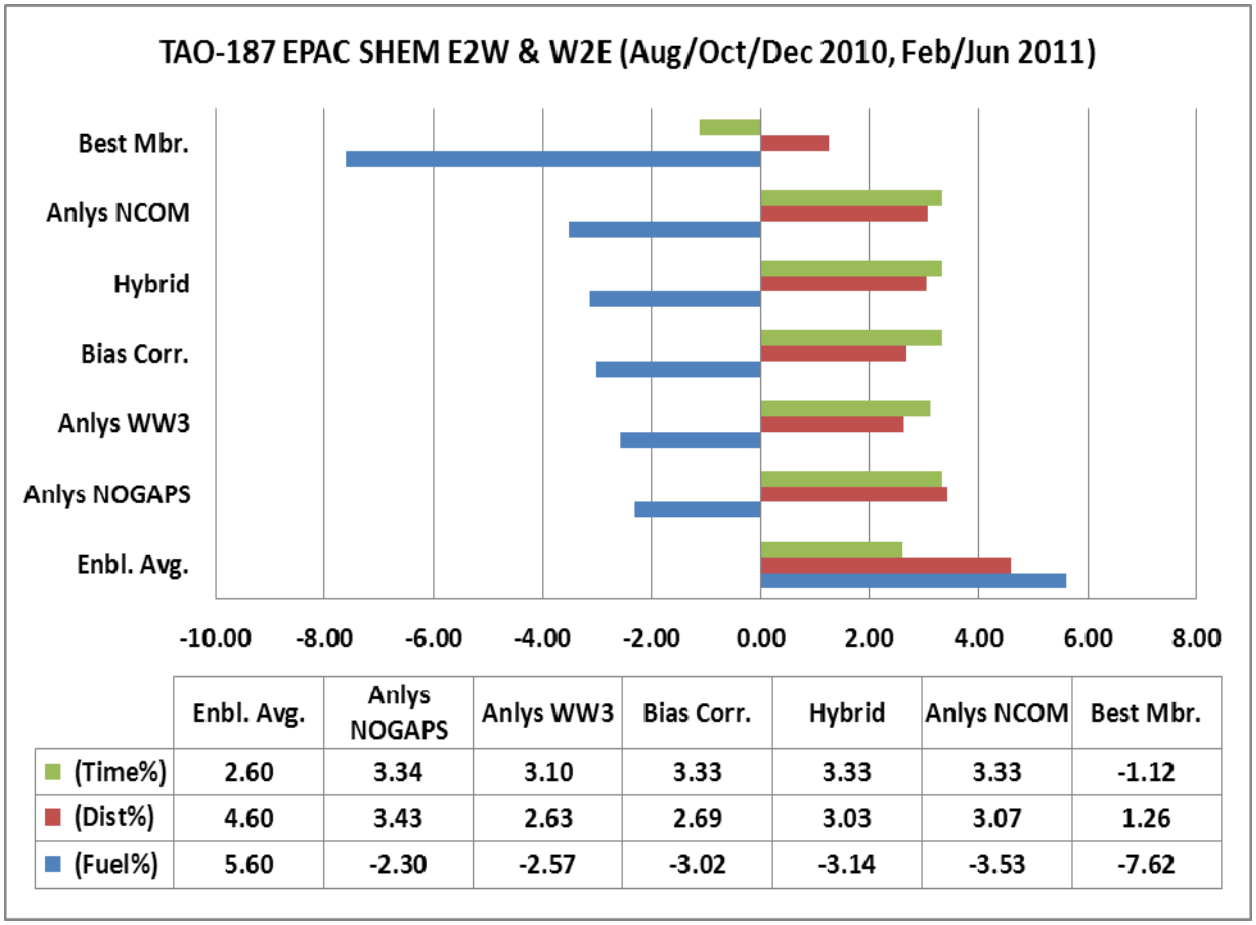


	Hybrid	Anlys NOGAPS	Enbl. Avg.	Anlys NCOM	Bias Corr.	Anlys WW3	Best Mbr.
(Time%)	4.36	4.36	4.36	4.36	4.36	4.36	4.35
(Dist%)	3.22	3.28	3.32	3.17	3.37	3.56	2.82
(Fuel%)	-0.10	-0.22	-0.29	-0.45	-0.74	-0.98	-1.61

Figure 47. TAO-187 EPAC equator combined directions ensemble % time/dist./fuel vs. GC (Jun/Aug/Oct/Dec 2010, Feb/Jun 2011)

F. CASE 6

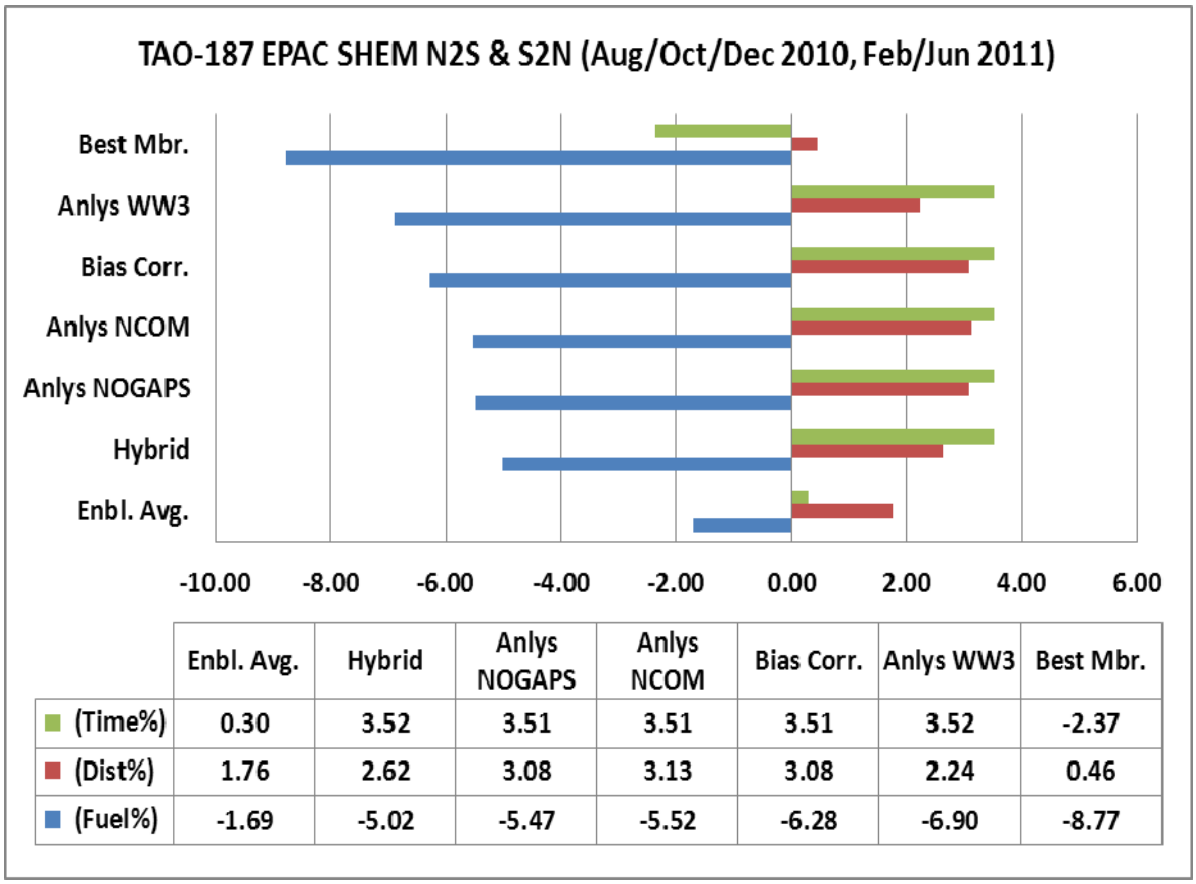
1. Eastern Pacific Southern Hemisphere (Latitude Parallel)



	Enbl. Avg.	Anlys NOGAPS	Anlys WW3	Bias Corr.	Hybrid	Anlys NCOM	Best Mbr.
(Time%)	2.60	3.34	3.10	3.33	3.33	3.33	-1.12
(Dist%)	4.60	3.43	2.63	2.69	3.03	3.07	1.26
(Fuel%)	5.60	-2.30	-2.57	-3.02	-3.14	-3.53	-7.62

Figure 48. TAO-187 EPAC SHEM ocean east to west & west to east ensemble % time/dist./fuel vs. GC (Aug/Oct/Dec 2010, Feb/Jun 2011)

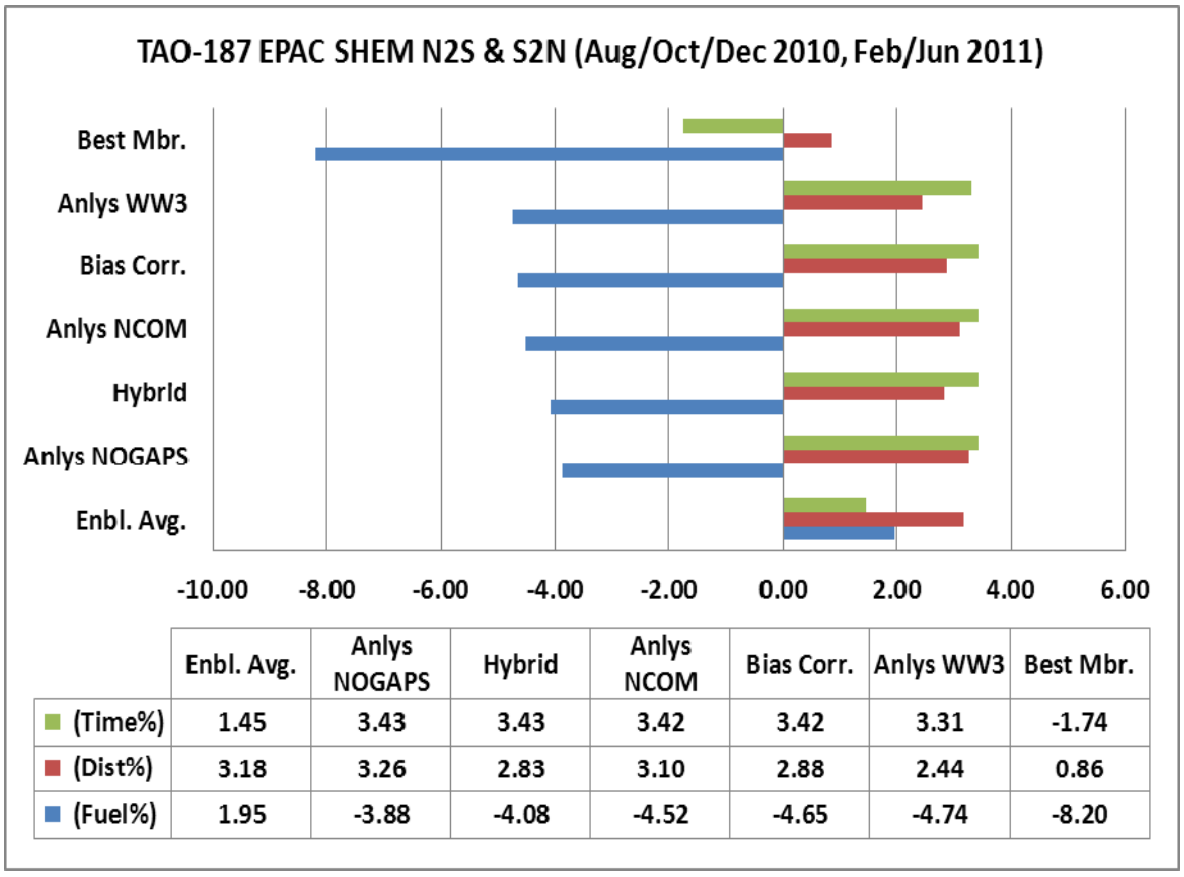
2. Eastern Pacific Southern Hemisphere (Longitude Parallel)



	Enbl. Avg.	Hybrid	Anlys NOGAPS	Anlys NCOM	Bias Corr.	Anlys WW3	Best Mbr.
(Time%)	0.30	3.52	3.51	3.51	3.51	3.52	-2.37
(Dist%)	1.76	2.62	3.08	3.13	3.08	2.24	0.46
(Fuel%)	-1.69	-5.02	-5.47	-5.52	-6.28	-6.90	-8.77

Figure 49. TAO-187 EPAC SHEM north to south & south to north ensemble % time/dist./fuel vs. GC (Aug/Oct/Dec 2010, Feb/Jun 2011)

3. Eastern Pacific Southern Hemisphere (Combined Results)



	Enbl. Avg.	Anlys NOGAPS	Hybrid	Anlys NCOM	Bias Corr.	Anlys WW3	Best Mbr.
(Time%)	1.45	3.43	3.43	3.42	3.42	3.31	-1.74
(Dist%)	3.18	3.26	2.83	3.10	2.88	2.44	0.86
(Fuel%)	1.95	-3.88	-4.08	-4.52	-4.65	-4.74	-8.20

Figure 50. TAO-187 EPAC SHEM combined directions ensemble % time/dist./fuel vs. GC (Aug/Oct/Dec 2010, Feb/Jun 2011)

APPENDIX C. SPATIAL DISTRIBUTION

A. SEASONAL VARIANCE

1. TAO-187 Seasonal Spread (Diego Gar. to GOO, 01 Jun 2010)

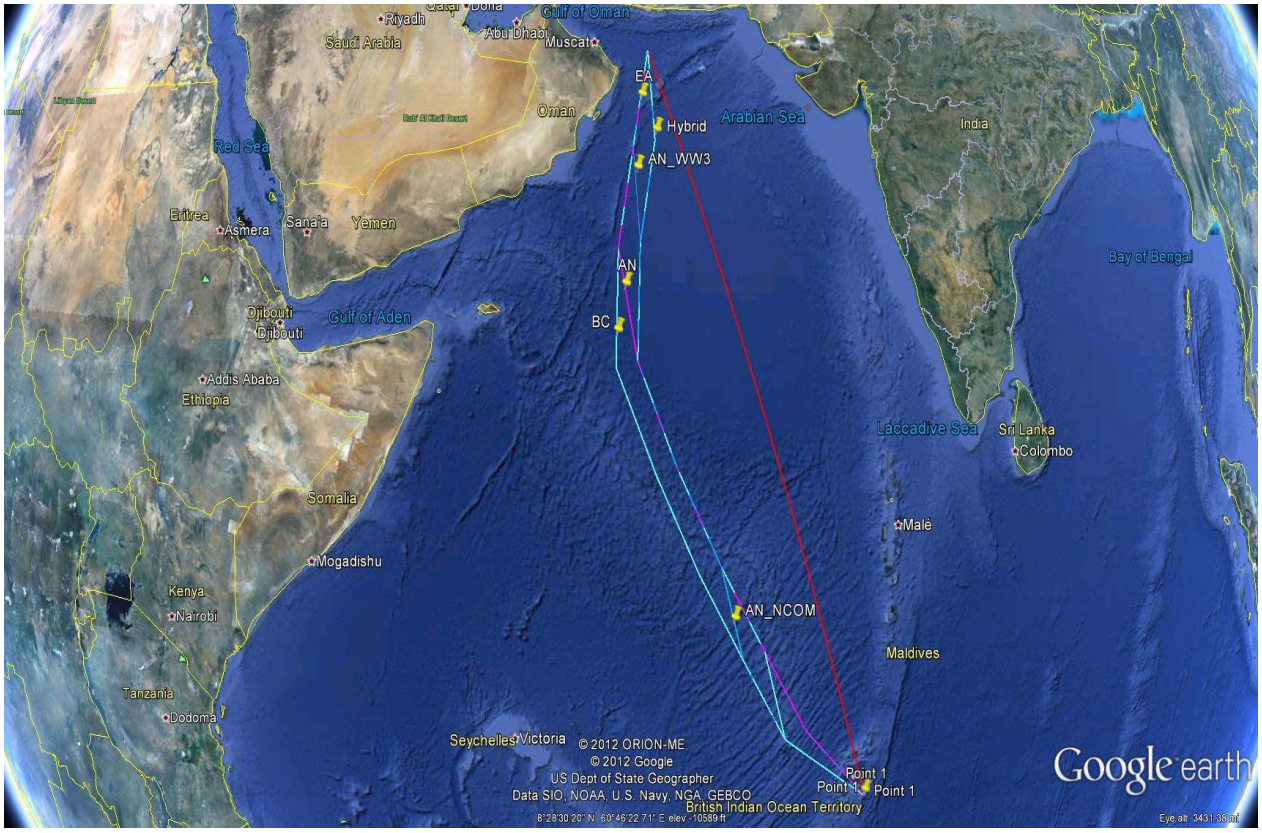


Figure 51. TAO-187 spatial ensemble seasonal variance spread example displaying GC (red), post processed (cyan), analysis model variants (teal), and analysis (magenta) ensembles (Diego Garcia to Gulf of Oman: 01 Jun 2010)

2. TAO-187 Seasonal Spread (Diego Gar. to GOO, 01 Dec 2010)

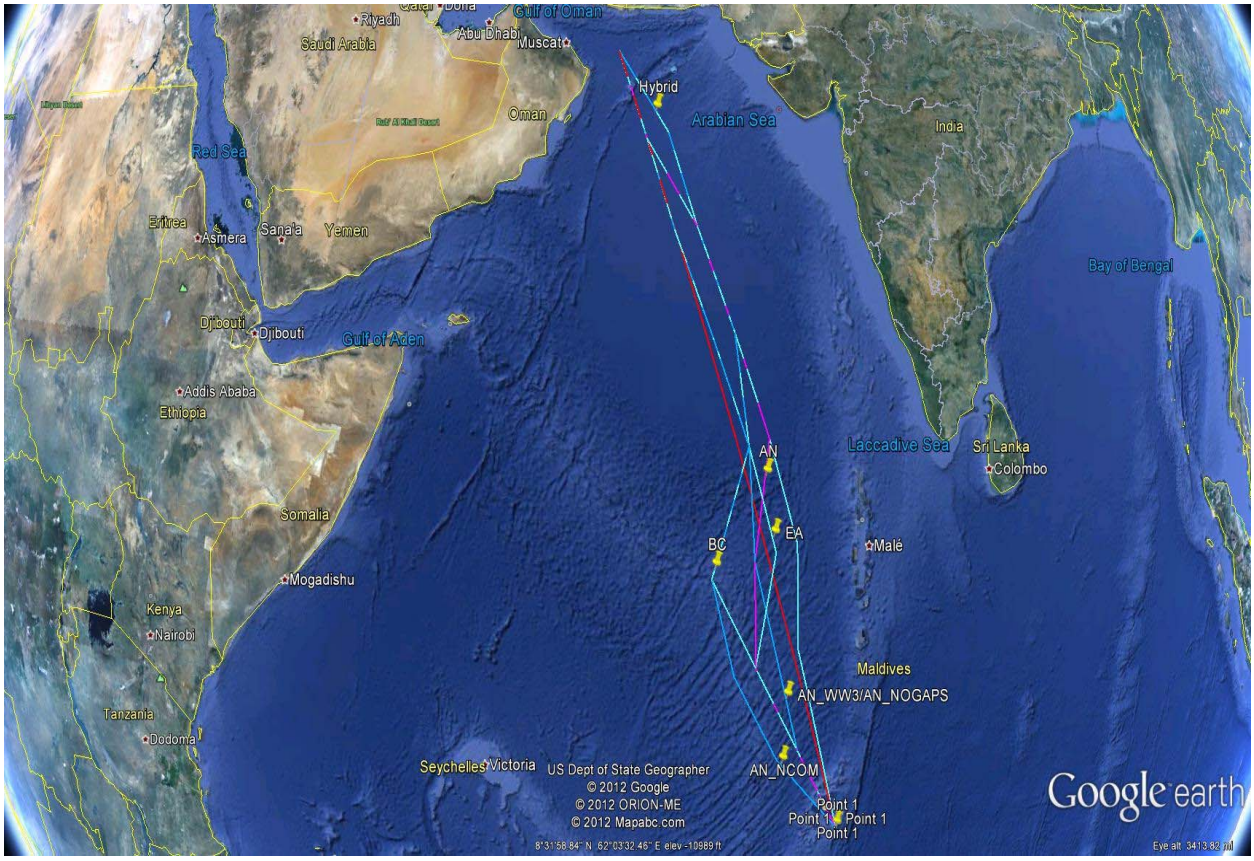


Figure 52. TAO-187 spatial ensemble seasonal variance spread example displaying GC (red), post processed (cyan), analysis model variants (teal), and analysis (magenta) ensembles (Diego Garcia to Gulf of Oman: 01 Dec 2010)

B. DIRECTIONAL VARIANCE

1. TAO-187 Directional Spread (SD to Pearl, 01 May 2010)

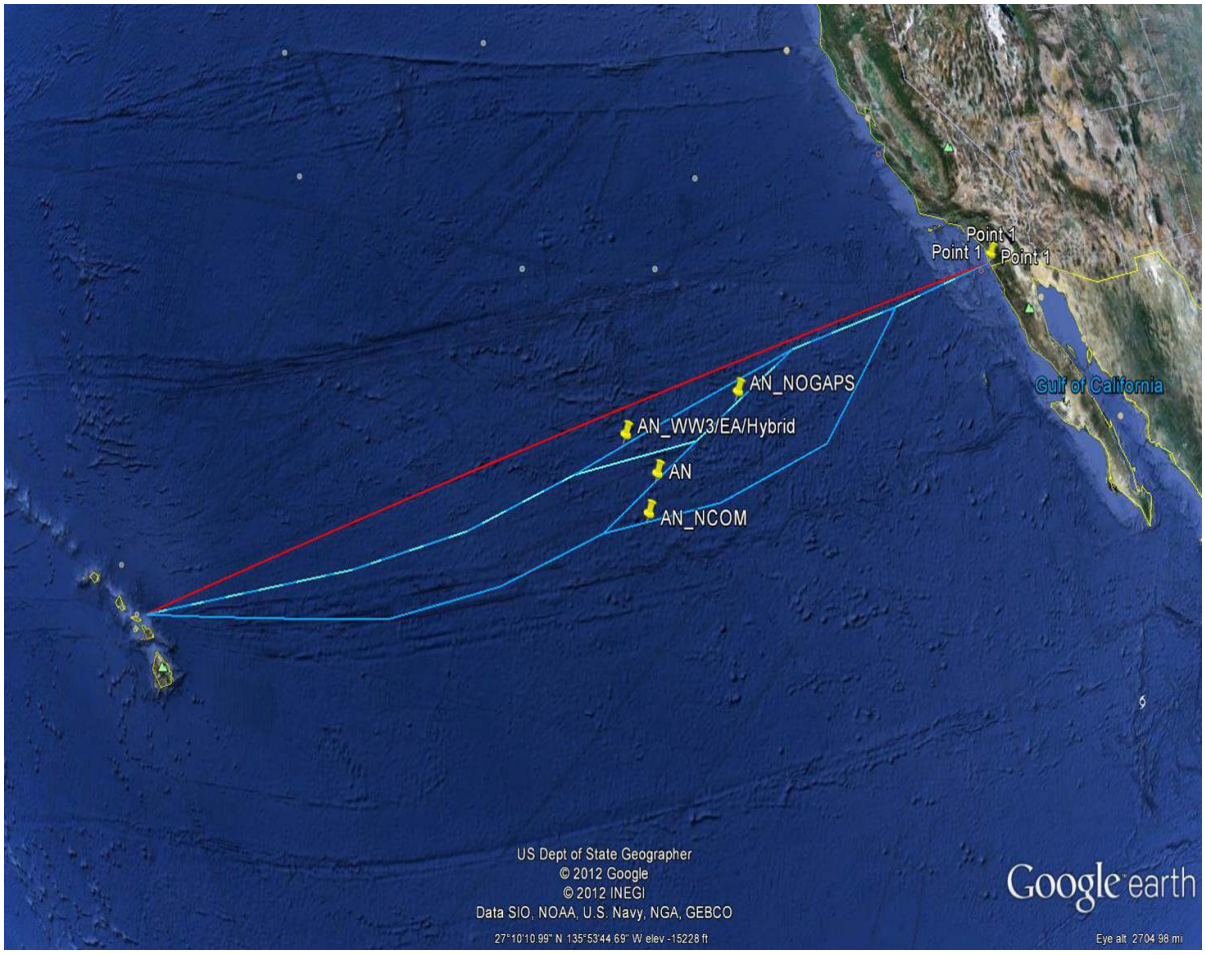


Figure 53. TAO-187 spatial ensemble spread directional variance example displaying GC (red), post processed (cyan), and analysis model variants (teal) ensembles (San Diego to Pearl Harbor: 01 May 2010)

2. TAO-187 Directional Spread (Pearl to SD, 01 May 2010)

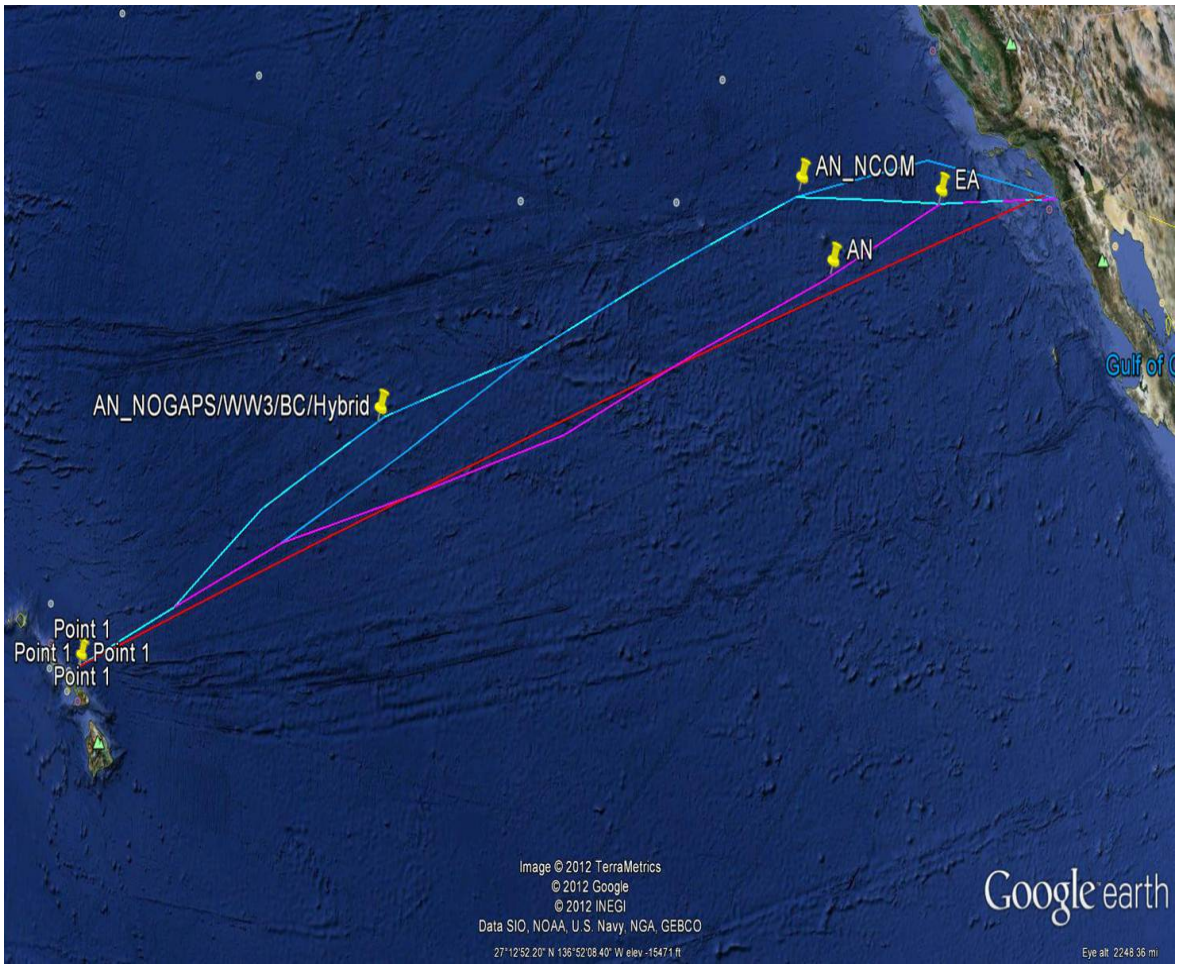


Figure 54. TAO-187 spatial ensemble spread directional variance example displaying GC (red), post processed (cyan), analysis model variants (teal), and analysis (magenta) ensembles (Pearl Harbor to San Diego: 01 May 2010)

C. LATITUDINAL VARIANCE

1. TAO-187 Eastern Pacific (Overall Latitudinal Route Spreads)

Figure 55. TAO-187 Eastern Pacific multi-directional and multiple date (2010-2011) latitude tests displaying GC and Hybrid ensemble inputs (both red w/yellow way pts) for directions sailed: north to south, south to north, east to west, west to east (northern hemisphere, equator, and southern hemisphere regions)

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. MODEL EXECUTION

A. MODEL ROUTE INPUT EXAMPLE

1. SVP Route Generation

REQUEST ID = $REQUESTID
CLASSIFICATION = UNCLASSIFIED
CAVEAT = NONE
REQUEST TYPE = $REQUESTTYPE
DESCRIPTION = Diego Garcia to Gulf of Oman SVP Routes
PASSAGE = Diego Garcia to Gulf of Oman
SHIP NAME = EFAS
SHIP CLASS = TAO187
SHIP FOR DRAFT = 11.7
SHIP AFT DRAFT = 12.0
SHIP TRANSV GM = 2.6
MAX HEAD SEA = 12.0
MAX BEAM SEA = 12.0
MAX FOLLOW SEA = 12.0
MAX TRUE WIND = 35.0
MAX REL WIND = 65.0
MAX SPEED = 25.0
MIN SPEED = 10.0
MIN DIST 35 = 65
MIN DIST 50 = 120
DEPARTURE DATE = $DEPARTUREDATE
DEPARTURE TIME = $DEPARTURETIME
ARRIVAL DATE = $ARRIVALDATE
ARRIVAL TIME = $ARRIVALTIME
WIND MODEL = NOGAPS
WAVE MODEL = WW3_GLOBAL
CURRENT MODEL = NCOM_GLOBAL
UPPER BOUND WAYPOINTS = 4
UB NUMBER = 01
UB LATITUDE = -7.24
UB LONGITUDE = 072.5
UB NAV TYPE = GC
UB NUMBER = 02
UB LATITUDE = 5
UB LONGITUDE = 72
UB NAV TYPE = GC
UB NUMBER = 03
UB LATITUDE = 17.5
UB LONGITUDE = 070.5
UB NAV TYPE = GC
UB NUMBER = 04
UB LATITUDE = 23.3
UB LONGITUDE = 61.3
UB NAV TYPE = GC
LOWER BOUND WAYPOINTS = 4
LB NUMBER = 01

LB LATITUDE = -7.24
LB LONGITUDE = 072.5
LB NAV TYPE = GC
LB NUMBER = 02
LB LATITUDE = 001
LB LONGITUDE = 60
LB NAV TYPE = GC
LB NUMBER = 03
LB LATITUDE = 15
LB LONGITUDE = 060
LB NAV TYPE = GC
LB NUMBER = 04
LB LATITUDE = 23.3
LB LONGITUDE = 61.3
LB NAV TYPE = GC
END

2. SVP Great Circle Baseline

REQUEST ID = $REQUESTID
CLASSIFICATION = UNCLASSIFIED
CAVEAT = NONE
REQUEST TYPE = WEAX
DESCRIPTION = Diego Garcia to Gulf of Omam GC Baseline
PASSAGE = Diego Garcia to Gulf of Omam
SHIP NAME = EFAS
SHIP CLASS = TAO187
SHIP FOR DRAFT = 11.7
SHIP AFT DRAFT = 12.0
SHIP TRANSV GM = 2.6
MAX HEAD SEA = 12.0
MAX BEAM SEA = 12.0
MAX FOLLOW SEA = 12.0
MAX TRUE WIND = 35.0
MAX REL WIND = 65.0
MAX SPEED = 25.0
MIN SPEED = 18.0
MIN DIST 35 = 60
MIN DIST 50 = 120
DEPARTURE DATE = $DEPARTUREDATE
DEPARTURE TIME = $DEPARTURETIME
ARRIVAL DATE = $ARRIVALDATE
ARRIVAL TIME = $ARRIVALTIME
WIND MODEL = NOGAPS
WAVE MODEL = WW3_GLOBAL
CURRENT MODEL = NCOM_GLOBAL
NUMBER OF WAYPOINTS = 2
POINT ID =1
WAYPOINT NUMBER =
LATITUDE = -7.24
LONGITUDE = 072.5
INTEND SPD = 17.5
LB NAV TYPE = GC

```
POINT ID =2
WAYPOINT NUMBER =
LATITUDE   = 23.3
LONGITUDE  = 61.3
INTEND SPD = 17.5
UB NAV TYPE =GC
END
```

B. MODEL RUN EXAMPLE

```
#!/bin/bash
# must type "module load python" to load required Python module on Gigaton
# To run in background use "nohup ./Test_Multi_Driver &"
#GC Baseline Speed for TAO class
#TAO 17.5 kts
STARSDRIVER=/ITRI_AOTSR/otsr/scripts/stars_driver.1A
############################PYTHON SCRIPT ######################
COLLECTSTATS=/ITRI_AOTSR/otsr/Scott/StarsDriver/EXE/Calc_Stats
############################ARCHIVE SCRIPT ######################
COLLECTRESULTS=/ITRI_AOTSR/otsr/Scott/Results/zipSTARSKmlHtmlResults
############################SETUP STARS ENV#####################
STARS_ENV=/ITRI_AOTSR/otsr/Scott/Runs/THESIS/setupstarsenv
############################MODEL TYPE ##########################
Model_Type=stars_060612_plusplus
############################DTG/ROUTE###########################
STARS_EXE=/ITRI_AOTSR/otsr/bin/$Model_Type
############################DTG/ROUTE###########################
Route_Name=SanDiego2PearlHarbor-TAO187
Route_NameGC=SanDiego2PearlHarbor-TAO187GC
############################DTG/ROUTE GC & POSTPROCESSED AVG###
Departure_Date=20100601
Arrival_Date=20101028
Run_Type=small

$STARSDRIVER $STARS_ENV \
  Route $Route_NameGC \
  DepartureDate $Departure_Date \
  ArrivalDate   $Arrival_Date \
  RoutexEnvironment "analysisV2" \
  WeaxEnvironment "analysisV2" \
  forecastdtg $Departure_Date'00' \
DisplayNameStub $Model_Type"-" \
StarsExecutable $STARS_EXE

$STARSDRIVER $STARS_ENV \
  Route $Route_Name \
  DepartureDate $Departure_Date \
  ArrivalDate   $Arrival_Date \
 RoutexEnvironment "Analysis_NOGAPS Analysis_WW3 Analysis_NCOM hybridensembleaverage
ensembleaverage biascorrectedaverage analysisV2 \
                        A2 BC_Z1" \
  WeaxEnvironment "analysisV2" \
  forecastdtg $Departure_Date'00' \
DisplayNameStub $Model_Type"-" \
```

```
StarsExecutable $STARS_EXE
##########################COLLECT STATS#################################
$COLLECTSTATS \
  Route $Route_Name \
  DepartureDate $Departure_Date'00' \
  Resolution $Model_Type \
##########################COLLECT RESULTS##############################
  $COLLECTRESULTS \
  Route $Route_Name \
  GCRoute $Route_NameGC \
  DepartureDate $Departure_Date'00' \
  ModelType $Model_Type  \
  Runtype $Run_Type \
############################DTG/ROUTE RAW & BC#########################
Run_Type=large

$STARSDRIVER $STARS_ENV \
   Route $Route_Name \
   DepartureDate $Departure_Date \
   ArrivalDate   $Arrival_Date \
  RoutexEnvironment "B2 C2 D2 E2 F2 Q1 R1 S1 T1 U1 V1 W1 X1 Y1 Z1 \
                        BC_A2 BC_B2 BC_C2 BC_D2 BC_E2 BC_F2 BC_Q1 BC_R1 \
                        BC_S1 BC_T1 BC_U1 BC_V1 BC_W1 BC_X1 BC_Y1" \
   WeaxEnvironment "analysisV2" \
   forecastdtg $Departure_Date'00' \
DisplayNameStub $Model_Type"-" \
StarsExecutable $STARS_EXE
##########################COLLECT STATS#################################
$COLLECTSTATS \
  Route $Route_Name \
  DepartureDate $Departure_Date'00' \
  Resolution $Model_Type \
 ##########################COLLECT RESULTS##############################
  $COLLECTRESULTS \
  Route $Route_Name \
  GCRoute $Route_NameGC \
  DepartureDate $Departure_Date'00' \
  ModelType $Model_Type  \
  Runtype $Run_Type \
exit
```

C. MODEL XML OUTPUT EXAMPLE


```
<RouteGetResponse>
  <ResponseStatus><Success/></ResponseStatus>
  <Classification>
    <Level>UNCLASSIFIED</Level>
    <Caveat>NONE</Caveat>
    <Derivation></Derivation>
    <Declass></Declass>
  </Classification>
  <Header>
    <RequestId>EFAS_2010120100</RequestId>
    <RequestType>ROUTEX</RequestType>
   <Description>Diego Garcia to Gulf of Oman SVP Routes</Description>
   <Passage>Diego Garcia to Gulf of Oman</Passage>
    <Units>english</Units>
    <CreationDate>08/23/2012</CreationDate>
    <CreationTime>19:32:02</CreationTime>
    <Ship>EFAS</Ship>
    <DepartureDate>12/01/2010</DepartureDate>
    <DepartureTime>12:00:00</DepartureTime>
    <TimeEnroute>124.88</TimeEnroute>
    <DistanceEnroute>1965.96</DistanceEnroute>
    <FuelEnroute>193.54</FuelEnroute>
    <RequiredSpeed>15.7</RequiredSpeed>
     <Models>
       <WindModel>NOGAPS</WindModel>
       <WaveModel>WW3_GLOBAL</WaveModel>
       <CurrentModel>NCOM_GLOBAL</CurrentModel>
     </Models>
  </Header>
  <Input>
    <REQUEST_ID>EFAS_2010120100</REQUEST_ID>
    <CLASSIFICATION>UNCLASSIFIED</CLASSIFICATION>
    <CAVEAT>NONE</CAVEAT>
    <DERIVATION></DERIVATION>
    <DECLASS></DECLASS>
    <REQUEST_TYPE>ROUTEX</REQUEST_TYPE>
    <DESCRIPTION>Diego Garcia to Gulf of Oman SVP Routes</DESCRIPTION>
    <PASSAGE>Diego Garcia to Gulf of Oman</PASSAGE>
    <SHIP_NAME>EFAS</SHIP_NAME>
    <REQUESTED_SHIP_CLASS>TAO187</REQUESTED_SHIP_CLASS>
    <EXECUTED_SHIP_CLASS>TAO187</EXECUTED_SHIP_CLASS>
    <SHIP_FOR_DRAFT>   38.38536    </SHIP_FOR_DRAFT>
    <SHIP_AFT_DRAFT>   39.36960    </SHIP_AFT_DRAFT>
    <SHIP_TRANSV_GM>   8.530080    </SHIP_TRANSV_GM>
    <MAX_SPEED>   25.00000    </MAX_SPEED>
    <MIN_SPEED>   10.00000    </MIN_SPEED>
    <MAX_HEAD_SEA>   12    </MAX_HEAD_SEA>
    <MAX_BEAM_SEA>   12    </MAX_BEAM_SEA>
    <MAX_FOLLOW_SEA>   12    </MAX_FOLLOW_SEA>
    <MAX_TRUE_WIND>   35.00000    </MAX_TRUE_WIND>
    <MAX_REL_WIND>   65.00000    </MAX_REL_WIND>
    <MIN_DIST_35>   65.00000    </MIN_DIST_35>
```

<MIN_DIST_50> 120.0000 </MIN_DIST_50>
<DEPARTURE_DATETIME> 58933092.0000000 </DEPARTURE_DATETIME>
<ARRIVAL_DATETIME> 58933740.0000000 </ARRIVAL_DATETIME>
<WIND_MODEL>NOGAPS</WIND_MODEL>
<WAVE_MODEL>WW3_GLOBAL</WAVE_MODEL>
<CURRENT_MODEL>NCOM_GLOBAL</CURRENT_MODEL>
<UPPER_BOUND_WAYPOINTS>
 <UPPER_BOUND_WAYPOINT>
  <NUMBER> 1 </NUMBER>
  <LATITUDE> -7.240000 </LATITUDE>
  <LONGITUDE> 72.50000 </LONGITUDE>
  <NAV_TYPE>GC</NAV_TYPE>
 </UPPER_BOUND_WAYPOINT>
 <UPPER_BOUND_WAYPOINT>
  <NUMBER> 2 </NUMBER>
  <LATITUDE> 5.000000 </LATITUDE>
  <LONGITUDE> 72.00000 </LONGITUDE>
  <NAV_TYPE>GC</NAV_TYPE>
 </UPPER_BOUND_WAYPOINT>
 <UPPER_BOUND_WAYPOINT>
  <NUMBER> 3 </NUMBER>
  <LATITUDE> 17.50000 </LATITUDE>
  <LONGITUDE> 70.50000 </LONGITUDE>
  <NAV_TYPE>GC</NAV_TYPE>
 </UPPER_BOUND_WAYPOINT>
 <UPPER_BOUND_WAYPOINT>
  <NUMBER> 4 </NUMBER>
  <LATITUDE> 23.30000 </LATITUDE>
  <LONGITUDE> 61.30000 </LONGITUDE>
  <NAV_TYPE>GC</NAV_TYPE>
 </UPPER_BOUND_WAYPOINT>
</UPPER_BOUND_WAYPOINTS>
<LOWER_BOUND_WAYPOINTS>
 <LOWER_BOUND_WAYPOINT>
  <NUMBER> 1 </NUMBER>
  <LATITUDE> -7.240000 </LATITUDE>
  <LONGITUDE> 72.50000 </LONGITUDE>
  <NAV_TYPE>GC</NAV_TYPE>
 </LOWER_BOUND_WAYPOINT>
 <LOWER_BOUND_WAYPOINT>
  <NUMBER> 2 </NUMBER>
  <LATITUDE> 1.000000 </LATITUDE>
  <LONGITUDE> 60.00000 </LONGITUDE>
  <NAV_TYPE>GC</NAV_TYPE>
 </LOWER_BOUND_WAYPOINT>
 <LOWER_BOUND_WAYPOINT>
  <NUMBER> 3 </NUMBER>
  <LATITUDE> 15.00000 </LATITUDE>
  <LONGITUDE> 60.00000 </LONGITUDE>
  <NAV_TYPE>GC</NAV_TYPE>
 </LOWER_BOUND_WAYPOINT>
 <LOWER_BOUND_WAYPOINT>
  <NUMBER> 4 </NUMBER>
  <LATITUDE> 23.30000 </LATITUDE>
  <LONGITUDE> 61.30000 </LONGITUDE>

```
      <NAV_TYPE>GC</NAV_TYPE>
    </LOWER_BOUND_WAYPOINT>
  </LOWER_BOUND_WAYPOINTS>
</Input>
<PointsList>
 <Point>
 <PointId>1</PointId>
 <WpNumber></WpNumber>
 <DTG>201012011200</DTG>
 <Latitude>-7.240</Latitude>
 <Longitude>72.500</Longitude>
 <NavType>GC</NavType>
 <ShipSpeed>10.50</ShipSpeed>
 <ShipCourse>342.61</ShipCourse>
 <WindSpeed>8.22</WindSpeed>
 <WindDirection>226.18</WindDirection>
 <SigWaveHeight>5.55</SigWaveHeight>
 <SeaHeight>0.04</SeaHeight>
 <SeaPeriod></SeaPeriod>
 <SeaDirection></SeaDirection>
 <SwellHeight>5.55</SwellHeight>
 <SwellPeriod>8.53</SwellPeriod>
 <SwellDirection>150.19</SwellDirection>
 <CurrentSpeed>0.54</CurrentSpeed>
 <CurrentDirection>336.10</CurrentDirection>
 <EnvironLimits>
   <MinDist35></MinDist35>
   <MinDist50></MinDist50>
   <RelWind>0</RelWind>
   <SwlHtBeam>0</SwlHtBeam>
   <SwlHtFollow>0</SwlHtFollow>
   <SwlHtHead>0</SwlHtHead>
   <TrueWind>0</TrueWind>
   <WvHtBeam>0</WvHtBeam>
   <WvHtFollow>0</WvHtFollow>
   <WvHtHead>0</WvHtHead>
   <DepthWrngs></DepthWrngs>
 </EnvironLimits>
 <Fuel>4.99</Fuel>
 <Distance>63.00</Distance>
 <WindSource>NOGAPS-2010120100</WindSource>
 <WaveSource>WW3_GLOBAL-2010120100</WaveSource>
 <CurrentSource>NCOM_GLOBAL-2010120100</CurrentSource>
 </Point>
.
.
.
.
.
.
.
.
.
```

```
<Point>
   <PointId>29</PointId>
   <WpNumber></WpNumber>
   <DTG>201012061652</DTG>
   <Latitude>23.300</Latitude>
   <Longitude>61.300</Longitude>
   <NavType>NO</NavType>
   <ShipSpeed>0.00</ShipSpeed>
   <ShipCourse>0.00</ShipCourse>
   <WindSpeed>13.74</WindSpeed>
   <WindDirection>67.16</WindDirection>
   <SigWaveHeight>3.99</SigWaveHeight>
   <SeaHeight>1.64</SeaHeight>
   <SeaPeriod>4.05</SeaPeriod>
   <SeaDirection>32.99</SeaDirection>
   <SwellHeight>3.64</SwellHeight>
   <SwellPeriod>5.79</SwellPeriod>
   <SwellDirection>79.06</SwellDirection>
   <CurrentSpeed>0.01</CurrentSpeed>
   <CurrentDirection>146.31</CurrentDirection>
   <EnvironLimits>
     <MinDist35></MinDist35>
     <MinDist50></MinDist50>
     <RelWind>0</RelWind>
     <SwlHtBeam>0</SwlHtBeam>
     <SwlHtFollow>0</SwlHtFollow>
     <SwlHtHead>0</SwlHtHead>
     <TrueWind>0</TrueWind>
     <WvHtBeam>0</WvHtBeam>
     <WvHtFollow>0</WvHtFollow>
     <WvHtHead>0</WvHtHead>
     <DepthWrngs></DepthWrngs>
   </EnvironLimits>
   <Fuel>0.00</Fuel>
   <Distance>0.00</Distance>
   <WindSource>NOGAPS-2010120100</WindSource>
   <WaveSource>WW3_GLOBAL-2010120100</WaveSource>
   <CurrentSource>NCOM_GLOBAL-2010120100</CurrentSource>
   </Point>
  </PointsList>
 </RouteGetResponse>
```

D. MODEL STATISTICS CODE

```
#!/usr/bin/env python
# module load python
#
#Original Code by:          Scott E. Miller
#                           LCDR USN
#
#Date Written:              July 2012
#NPS Thesis Requirement
#Statistical Analysis of STARS Model XML Output
#Written in PYTHON
#
#

import re
import sys
import glob
import os
from os.path import join, getsize
import math
import numpy as np
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
import pylab
import scipy
import operator
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.pyplot import *

def make_label(str):
   components=str.split('.')
   print components[2]+"."+components[3]
   if components[3]=="hybridensembleaverage" :
      components[3]="HyEnsblAvg"
   if components[3]=="ensembleaverage" :
      components[3]="EnsblAvg"
   if components[3]=="biascorrectedaverage" :
      components[3]="BCAvg"
   if components[3]=="Analysis_NOGAPS" :
      components[3]="Anlys_NOGAPS"
   if components[3]=="Analysis_WW3" :
      components[3]="Anlys_WW3"
   if components[3]=="Analysis_NCOM" :
      components[3]="Anlys_NCOM"
   if components[3]=="analysis" :
      components[3]="Analysis"
   if components[3]=="analysisV2" :
      components[3]="Analysis"

   if len(components)==5 :
      if components[4]=="analysis" :
         components[4]="anlys"
      if components[4]=="analysisV2" :
```

```
            components[4]="anlys"
        if components[2].endswith('GC'):
            print "Great Circle Fuel="+ GC_Fuel
            print "Hybrid analys Fuel="+  Hy_avg_alys_Fuel
            print "Hybrid Fuel="+  Hy_avg_Fuel
            print "BC analys Fuel="+  BC_avg_alys_Fuel
            print "BC Fuel="+  BC_avg_Fuel
            print "Esmbl analys Fuel="+  EA_avg_alys_Fuel
            print "Esmbl Fuel="+  EA_avg_Fuel
            components[3]="GrtCir"
            components[4]="anlys"
        return '_'.join(components[3:])+"WX"
    else:
        return '_'.join(components[3:])

rdate=sys.argv[1]
rname=sys.argv[2]
rres=sys.argv[3]

AVG_COLOR='c'
GC_COLOR='r'
WEAX_COLOR='g'
ROUTEX_COLOR='b'
AN_COLOR='m'
MODEL_COLOR='teal'

os.getcwd()

root2='/ITRI_AOTSR/otsr/Scott/Python/'
root='/ITRI_AOTSR/otsr/Scott/resultdata/*/'+rdate
root3='/ITRI_AOTSR/otsr/Scott/figures/'+rname+'/'+rdate+'/'+rres+'/'

strip='.'+rdate+'.'

if not os.path.exists(root3):
    os.makedirs(root3)

print root

dirnames=glob.glob(root+'*')
write_stats = open(root3+'stats.'+rdate+"."+rname+"."+rres, 'w')
write_stats.write('Name, Power/Fuel, Distance, Time\n')
i=0
print dirnames
Array_list=["Name","HP","Dist","Time"]
Array=[Array_list]
MLM_Fuel=10000
MLM_Time=100000
MLM_Dist=100000
GC_Exceed_Lim_count=0
HY_Exceed_Lim_count=0
BC_Exceed_Lim_count=0
EA_Exceed_Lim_count=0
AN_Exceed_Lim_count=0
```

AN_NOGAPS_avg_alys_Fuel=0
AN_NOGAPS_avg_alys_Dist=0
AN_NOGAPS_avg_alys_Time=0
AN_WW3_avg_alys_Fuel=0
AN_WW3_avg_alys_Dist=0
AN_WW3_avg_alys_Time=0

GCMin_dist35_Exceed_Lim_count=0
GCMin_dist50_Exceed_Lim_count=0
GCRelWind_Exceed_Lim_count=0
GCSwlHtBeam_Exceed_Lim_count=0
GCSwlHtFollow_Exceed_Lim_count=0
GCSwlHtHead_Exceed_Lim_count=0
GCTrueWind_Exceed_Lim_count=0
GCWvHtBeam_Exceed_Lim_count=0
GCWvHtFollow_Exceed_Lim_count=0
GCWvHtHead_Exceed_Lim_count=0

HYMin_dist35_Exceed_Lim_count=0
HYMin_dist50_Exceed_Lim_count=0
HYRelWind_Exceed_Lim_count=0
HYSwlHtBeam_Exceed_Lim_count=0
HYSwlHtFollow_Exceed_Lim_count=0
HYSwlHtHead_Exceed_Lim_count=0
HYTrueWind_Exceed_Lim_count=0
HYWvHtBeam_Exceed_Lim_count=0
HYWvHtFollow_Exceed_Lim_count=0
HYWvHtHead_Exceed_Lim_count=0

BCMin_dist35_Exceed_Lim_count=0
BCMin_dist50_Exceed_Lim_count=0
BCRelWind_Exceed_Lim_count=0
BCSwlHtBeam_Exceed_Lim_count=0
BCSwlHtFollow_Exceed_Lim_count=0
BCSwlHtHead_Exceed_Lim_count=0
BCTrueWind_Exceed_Lim_count=0
BCWvHtBeam_Exceed_Lim_count=0
BCWvHtFollow_Exceed_Lim_count=0
BCWvHtHead_Exceed_Lim_count=0

EAMin_dist35_Exceed_Lim_count=0
EAMin_dist50_Exceed_Lim_count=0
EARelWind_Exceed_Lim_count=0
EASwlHtBeam_Exceed_Lim_count=0
EASwlHtFollow_Exceed_Lim_count=0
EASwlHtHead_Exceed_Lim_count=0
EATrueWind_Exceed_Lim_count=0
EAWvHtBeam_Exceed_Lim_count=0
EAWvHtFollow_Exceed_Lim_count=0
EAWvHtHead_Exceed_Lim_count=0

ANMin_dist35_Exceed_Lim_count=0
ANMin_dist50_Exceed_Lim_count=0
ANRelWind_Exceed_Lim_count=0
ANSwlHtBeam_Exceed_Lim_count=0

```
ANSwlHtFollow_Exceed_Lim_count=0
ANSwlHtHead_Exceed_Lim_count=0
ANTrueWind_Exceed_Lim_count=0
ANWvHtBeam_Exceed_Lim_count=0
ANWvHtFollow_Exceed_Lim_count=0
ANWvHtHead_Exceed_Lim_count=0

for name in dirnames:
      print(name+'/*.'+rname+'.*.xml')
      print(name+'/*.'+rres+'*'+rname+'*.*.xml')
      list_of_files = glob.glob(name+'/*.'+rres+'*'+rname+'*.*.xml')          # create the list of files
      print list_of_files
      #import xml parser minidom:
      from xml.dom.minidom import parseString

      for file_name in list_of_files:
         #open xml file for reading:
         file = open(file_name,'r')
         #convert to string:
         data = file.read()
         #close file:
         file.close()
         #parse the xml file
         dom = parseString(data)
         WP_tag = dom.getElementsByTagName('WpNumber')[0].toxml()
         if WP_tag =="NONONONONONONONONONO" :
            continue
         HP_xml_Tag = dom.getElementsByTagName('FuelEnroute')[0].toxml()
         Dist_xml_Tag = dom.getElementsByTagName('DistanceEnroute')[0].toxml()
         Time_xml_Tag = dom.getElementsByTagName('TimeEnroute')[0].toxml()
         HP_xml_data=HP_xml_Tag.replace('<FuelEnroute>','').replace('</FuelEnroute>','')
         Dist_xml_data=Dist_xml_Tag.replace('<DistanceEnroute>','').replace('</DistanceEnroute>','')
         Time_xml_data=Time_xml_Tag.replace('<TimeEnroute>','').replace('</TimeEnroute>','')
         print file_name, " stats:"
         print "Total HP =", HP_xml_data
         print "Total Dist =", Dist_xml_data
         print "Total Time =", Time_xml_data
          ##########################################################################

         if file_name.endswith('GC.analysis.analysis.xml') or
file_name.endswith('GC.analysisV2.analysisV2.xml'):
            GC_Fuel=HP_xml_data
            GC_Dist=Dist_xml_data
            GC_Time=Time_xml_data
            points = dom.getElementsByTagName("PointId")
            #Total_GCEnviro_lim_Exceed=0
            point_ind=0
            for point in points:
               Exceed_enviro_lim_xml_tag = dom.getElementsByTagName('EnvironLimits')[point_ind]
               Min_dist35= Exceed_enviro_lim_xml_tag.childNodes[1].toxml()
               Min_dist35=Min_dist35.replace('<MinDist35>','').replace('</MinDist35>','')
               print "MinDist35", Min_dist35
               if not (Min_dist35=='<MinDist35/>'):
                  GC_Exceed_Lim_count=GC_Exceed_Lim_count+1
                  GCMin_dist35_Exceed_Lim_count=GCMin_dist35_Exceed_Lim_count+1
```

```
Min_dist50= Exceed_enviro_lim_xml_tag.childNodes[3].toxml()
Min_dist50=Min_dist50.replace('<MinDist50>','').replace('</MinDist50>','')
print "MinDist50", Min_dist50

if not (Min_dist50=='<MinDist50/>'):
   GC_Exceed_Lim_count=GC_Exceed_Lim_count+1
   GCMin_dist50_Exceed_Lim_count=GC_Exceed_Lim_count+1

RelWind= Exceed_enviro_lim_xml_tag.childNodes[5].toxml()
RelWind=RelWind.replace('<RelWind>','').replace('</RelWind>','')
print "rel wind", RelWind

if not (RelWind=='0'):
   GC_Exceed_Lim_count=GC_Exceed_Lim_count+1
   GCRelWind_Exceed_Lim_count=GCRelWind_Exceed_Lim_count+1
SwlHtBeam= Exceed_enviro_lim_xml_tag.childNodes[7].toxml()
SwlHtBeam=SwlHtBeam.replace('<SwlHtBeam>','').replace('</SwlHtBeam>','')
print "SwlHtBeam", SwlHtBeam
if not (SwlHtBeam=='0'):
   GC_Exceed_Lim_count=GC_Exceed_Lim_count+1
   GCSwlHtBeam_Exceed_Lim_count=GCSwlHtBeam_Exceed_Lim_count+1
SwlHtFollow= Exceed_enviro_lim_xml_tag.childNodes[9].toxml()
SwlHtFollow=SwlHtFollow.replace('<SwlHtFollow>','').replace('</SwlHtFollow>','')
print "SwlHtFollow", SwlHtFollow
if not (SwlHtFollow=='0'):
   GC_Exceed_Lim_count=GC_Exceed_Lim_count+1
   GCSwlHtFollow_Exceed_Lim_count=GCSwlHtFollow_Exceed_Lim_count+1
SwlHtHead= Exceed_enviro_lim_xml_tag.childNodes[11].toxml()
SwlHtHead=SwlHtHead.replace('<SwlHtHead>','').replace('</SwlHtHead>','')
print "SwlHtHead", SwlHtHead
if not (SwlHtHead=='0'):
   GC_Exceed_Lim_count=GC_Exceed_Lim_count+1
   GCSwlHtHead_Exceed_Lim_count=GCSwlHtHead_Exceed_Lim_count+1
TrueWind= Exceed_enviro_lim_xml_tag.childNodes[13].toxml()
TrueWind=TrueWind.replace('<TrueWind>','').replace('</TrueWind>','')
print "TrueWind", TrueWind
if not (TrueWind=='0'):
   GC_Exceed_Lim_count=GC_Exceed_Lim_count+1
   GCTrueWind_Exceed_Lim_count=GCTrueWind_Exceed_Lim_count+1
WvHtBeam= Exceed_enviro_lim_xml_tag.childNodes[15].toxml()
WvHtBeam=WvHtBeam.replace('<WvHtBeam>','').replace('</WvHtBeam>','')
print "WvHtBeam", WvHtBeam
if not (WvHtBeam=='0'):
   GC_Exceed_Lim_count=GC_Exceed_Lim_count+1
   GCWvHtBeam_Exceed_Lim_count=GCWvHtBeam_Exceed_Lim_count+1
WvHtFollow= Exceed_enviro_lim_xml_tag.childNodes[17].toxml()
WvHtFollow=WvHtFollow.replace('<WvHtFollow>','').replace('</WvHtFollow>','')
print "WvHtFollow", WvHtFollow
if not (WvHtFollow=='0'):
   GC_Exceed_Lim_count=GC_Exceed_Lim_count+1
   GCWvHtFollow_Exceed_Lim_count=GCWvHtFollow_Exceed_Lim_count+1
WvHtHead= Exceed_enviro_lim_xml_tag.childNodes[19].toxml()
WvHtHead=WvHtHead.replace('<WvHtHead>','').replace('</WvHtHead>','')
print "WvHtHead", WvHtHead
```

```
            if not (WvHtHead=='0'):
               GC_Exceed_Lim_count=GC_Exceed_Lim_count+1
               GCWvHtHead_Exceed_Lim_count=GCWvHtHead_Exceed_Lim_count+1

            point_ind=point_ind+1

         print "GC_Exceed_Lim_count=", GC_Exceed_Lim_count

      ###########################################################################

      if file_name.endswith('hybridensembleaverage.analysis.xml') or
file_name.endswith('hybridensembleaverage.analysisV2.xml'):
         Hy_avg_alys_Fuel=HP_xml_data
         Hy_avg_alys_Dist=Dist_xml_data
         Hy_avg_alys_Time=Time_xml_data

         points = dom.getElementsByTagName("PointId")
         point_ind=0
         for point in points:
            Exceed_enviro_lim_xml_tag = dom.getElementsByTagName('EnvironLimits')[point_ind]
            Min_dist35= Exceed_enviro_lim_xml_tag.childNodes[1].toxml()
            Min_dist35=Min_dist35.replace('<MinDist35>','').replace('</MinDist35>','')
            print "MinDist35", Min_dist35
            if not (Min_dist35=='<MinDist35/>'):
               HY_Exceed_Lim_count=HY_Exceed_Lim_count+1
               HYMin_dist35_Exceed_Lim_count=HYMin_dist35_Exceed_Lim_count+1

            Min_dist50= Exceed_enviro_lim_xml_tag.childNodes[3].toxml()
            Min_dist50=Min_dist50.replace('<MinDist50>','').replace('</MinDist50>','')
            print "MinDist50", Min_dist50

            if not (Min_dist50=='<MinDist50/>'):
               HY_Exceed_Lim_count=HY_Exceed_Lim_count+1
               HYMinDist50_Exceed_Lim_count=HYMinDist50_Exceed_Lim_count+1

            RelWind= Exceed_enviro_lim_xml_tag.childNodes[5].toxml()
            RelWind=RelWind.replace('<RelWind>','').replace('</RelWind>','')
            print "rel wind", RelWind

            if not (RelWind=='0'):
               HY_Exceed_Lim_count=HY_Exceed_Lim_count+1
               HYRelWind_Exceed_Lim_count=HYRelWind_Exceed_Lim_count+1

            SwlHtBeam= Exceed_enviro_lim_xml_tag.childNodes[7].toxml()

            SwlHtBeam=SwlHtBeam.replace('<SwlHtBeam>','').replace('</SwlHtBeam>','')
            print "SwlHtBeam", SwlHtBeam

            if not (SwlHtBeam=='0'):
               HY_Exceed_Lim_count=HY_Exceed_Lim_count+1
               HYSwlHtBeam_Exceed_Lim_count=HYSwlHtBeam_Exceed_Lim_count+1

            SwlHtFollow= Exceed_enviro_lim_xml_tag.childNodes[9].toxml()
            SwlHtFollow=SwlHtFollow.replace('<SwlHtFollow>','').replace('</SwlHtFollow>','')
            print "SwlHtFollow", SwlHtFollow
```

```
        if not (SwlHtFollow=='0'):
            HY_Exceed_Lim_count=HY_Exceed_Lim_count+1
            HYSwlHtFollow_Exceed_Lim_count=HYSwlHtFollow_Exceed_Lim_count+1

        SwlHtHead= Exceed_enviro_lim_xml_tag.childNodes[11].toxml()
        SwlHtHead=SwlHtHead.replace('<SwlHtHead>','').replace('</SwlHtHead>','')
        print "SwlHtHead", SwlHtHead

        if not (SwlHtHead=='0'):
            HY_Exceed_Lim_count=HY_Exceed_Lim_count+1
            HYSwlHtHead_Exceed_Lim_count=HYSwlHtHead_Exceed_Lim_count+1

        TrueWind= Exceed_enviro_lim_xml_tag.childNodes[13].toxml()
        TrueWind=TrueWind.replace('<TrueWind>','').replace('</TrueWind>','')
        print "TrueWind", TrueWind

        if not (TrueWind=='0'):
            HY_Exceed_Lim_count=HY_Exceed_Lim_count+1
            HYTrueWind_Exceed_Lim_count=HYTrueWind_Exceed_Lim_count+1

        WvHtBeam= Exceed_enviro_lim_xml_tag.childNodes[15].toxml()
        WvHtBeam=WvHtBeam.replace('<WvHtBeam>','').replace('</WvHtBeam>','')
        print "WvHtBeam", WvHtBeam

        if not (WvHtBeam=='0'):
            HY_Exceed_Lim_count=HY_Exceed_Lim_count+1
            HYWvHtBeam_Exceed_Lim_count=HYWvHtBeam_Exceed_Lim_count+1

        WvHtFollow= Exceed_enviro_lim_xml_tag.childNodes[17].toxml()
        WvHtFollow=WvHtFollow.replace('<WvHtFollow>','').replace('</WvHtFollow>','')
        print "WvHtFollow", WvHtFollow

        if not (WvHtFollow=='0'):
            HY_Exceed_Lim_count=HY_Exceed_Lim_count+1
            HYWvHtFollow_Exceed_Lim_count=HYWvHtFollow_Exceed_Lim_count+1

        WvHtHead= Exceed_enviro_lim_xml_tag.childNodes[19].toxml()
        WvHtHead=WvHtHead.replace('<WvHtHead>','').replace('</WvHtHead>','')
        print "WvHtHead", WvHtHead

        if not (WvHtHead=='0'):
            HY_Exceed_Lim_count=HY_Exceed_Lim_count+1
            HYWvHtHead_Exceed_Lim_count=HYWvHtHead_Exceed_Lim_count+1

        point_ind=point_ind+1

    print "HY_Exceed_Lim_count=", HY_Exceed_Lim_count
 ##############################################################################

if file_name.endswith('hybridensembleaverage.xml'):
    Hy_avg_Fuel=HP_xml_data
    Hy_avg_Dist=Dist_xml_data
    Hy_avg_Time=Time_xml_data
```

```
        ##########################

        if file_name.endswith('biascorrectedaverage.analysis.xml') or
file_name.endswith('biascorrectedaverage.analysisV2.xml'):
            BC_avg_alys_Fuel=HP_xml_data
            BC_avg_alys_Dist=Dist_xml_data
            BC_avg_alys_Time=Time_xml_data

            points = dom.getElementsByTagName("PointId")
            Total_BCEnviro_lim_Exceed=0
            point_ind=0
            for point in points:
               #Point_xml_Tag = dom.getElementsByTagName('PointId')[i].toxml()
               Exceed_enviro_lim_xml_tag = dom.getElementsByTagName('EnvironLimits')[point_ind]

               Min_dist35= Exceed_enviro_lim_xml_tag.childNodes[1].toxml()
               Min_dist35=Min_dist35.replace('<MinDist35>','').replace('</MinDist35>','')
               print "MinDist35", Min_dist35

               if not (Min_dist35=='<MinDist35/>'):
                  BC_Exceed_Lim_count=BC_Exceed_Lim_count+1
                  BCMin_dist35_Exceed_Lim_count=BCMin_dist35_Exceed_Lim_count+1

               Min_dist50= Exceed_enviro_lim_xml_tag.childNodes[3].toxml()
               Min_dist50=Min_dist50.replace('<MinDist50>','').replace('</MinDist50>','')
               print "MinDist50", Min_dist50

               if not (Min_dist50=='<MinDist50/>'):
                  BC_Exceed_Lim_count=BC_Exceed_Lim_count+1
                  BCMin_dist50_Exceed_Lim_count=BCMin_dist50_Exceed_Lim_count+1

               RelWind= Exceed_enviro_lim_xml_tag.childNodes[5].toxml()
               RelWind=RelWind.replace('<RelWind>','').replace('</RelWind>','')
               print "rel wind", RelWind

               if not (RelWind=='0'):
                  BC_Exceed_Lim_count=BC_Exceed_Lim_count+1
                  BCRelWind_Exceed_Lim_count=BCRelWind_Exceed_Lim_count+1

               SwlHtBeam= Exceed_enviro_lim_xml_tag.childNodes[7].toxml()

               SwlHtBeam=SwlHtBeam.replace('<SwlHtBeam>','').replace('</SwlHtBeam>','')
               print "SwlHtBeam", SwlHtBeam

               if not (SwlHtBeam=='0'):
                  BC_Exceed_Lim_count=BC_Exceed_Lim_count+1
                  BCSwlHtBeam_Exceed_Lim_count=BCSwlHtBeam_Exceed_Lim_count+1

               SwlHtFollow= Exceed_enviro_lim_xml_tag.childNodes[9].toxml()
               SwlHtFollow=SwlHtFollow.replace('<SwlHtFollow>','').replace('</SwlHtFollow>','')
               print "SwlHtFollow", SwlHtFollow

               if not (SwlHtFollow=='0'):
                  BC_Exceed_Lim_count=BC_Exceed_Lim_count+1
                  BCSwlHtFollow_Exceed_Lim_count=BCSwlHtFollow_Exceed_Lim_count+1
```

```
            SwlHtHead= Exceed_enviro_lim_xml_tag.childNodes[11].toxml()
            SwlHtHead=SwlHtHead.replace('<SwlHtHead>','').replace('</SwlHtHead>','')
            print "SwlHtHead", SwlHtHead

            if not (SwlHtHead=='0'):
               BC_Exceed_Lim_count=BC_Exceed_Lim_count+1
               BCSwlHtHead_Exceed_Lim_count=BCSwlHtHead_Exceed_Lim_count+1

            TrueWind= Exceed_enviro_lim_xml_tag.childNodes[13].toxml()
            TrueWind=TrueWind.replace('<TrueWind>','').replace('</TrueWind>','')
            print "TrueWind", TrueWind

            if not (TrueWind=='0'):
               BC_Exceed_Lim_count=BC_Exceed_Lim_count+1
               BCTrueWind_Exceed_Lim_count=BCTrueWind_Exceed_Lim_count+1

            WvHtBeam= Exceed_enviro_lim_xml_tag.childNodes[15].toxml()
            WvHtBeam=WvHtBeam.replace('<WvHtBeam>','').replace('</WvHtBeam>','')
            print "WvHtBeam", WvHtBeam

            if not (WvHtBeam=='0'):
               BC_Exceed_Lim_count=BC_Exceed_Lim_count+1
               BCWvHtBeam_Exceed_Lim_count=BCWvHtBeam_Exceed_Lim_count+1

            WvHtFollow= Exceed_enviro_lim_xml_tag.childNodes[17].toxml()
            WvHtFollow=WvHtFollow.replace('<WvHtFollow>','').replace('</WvHtFollow>','')
            print "WvHtFollow", WvHtFollow

            if not (WvHtFollow=='0'):
               BC_Exceed_Lim_count=BC_Exceed_Lim_count+1
               BCWvHtFollow_Exceed_Lim_count=BCWvHtFollow_Exceed_Lim_count+1

            WvHtHead= Exceed_enviro_lim_xml_tag.childNodes[19].toxml()
            WvHtHead=WvHtHead.replace('<WvHtHead>','').replace('</WvHtHead>','')
            print "WvHtHead", WvHtHead

            if not (WvHtHead=='0'):
               BC_Exceed_Lim_count=BC_Exceed_Lim_count+1
               BCWvHtHead_Exceed_Lim_count=BCWvHtHead_Exceed_Lim_count+1

            point_ind=point_ind+1

         print "BC_Exceed_Lim_count=", BC_Exceed_Lim_count

      if file_name.endswith('biascorrectedaverage.xml'):
         BC_avg_Fuel=HP_xml_data
         BC_avg_Dist=Dist_xml_data
         BC_avg_Time=Time_xml_data

#########################################################################

      if file_name.endswith('.ensembleaverage.analysis.xml') or
file_name.endswith('.ensembleaverage.analysisV2.xml'):
         EA_avg_alys_Fuel=HP_xml_data
```

```
EA_avg_alys_Dist=Dist_xml_data
EA_avg_alys_Time=Time_xml_data

points = dom.getElementsByTagName("PointId")
Total_EAEnviro_lim_Exceed=0
point_ind=0
for point in points:
   #Point_xml_Tag = dom.getElementsByTagName('PointId')[i].toxml()
   Exceed_enviro_lim_xml_tag = dom.getElementsByTagName('EnvironLimits')[point_ind]

   Min_dist35= Exceed_enviro_lim_xml_tag.childNodes[1].toxml()
   Min_dist35=Min_dist35.replace('<MinDist35>','').replace('</MinDist35>','')
   print "MinDist35", Min_dist35

   if not (Min_dist35=='<MinDist35/>'):
      EA_Exceed_Lim_count=EA_Exceed_Lim_count+1
      EAMin_dist35_Exceed_Lim_count=EAMin_dist35_Exceed_Lim_count+1

   Min_dist50= Exceed_enviro_lim_xml_tag.childNodes[3].toxml()
   Min_dist50=Min_dist50.replace('<MinDist50>','').replace('</MinDist50>','')
   print "MinDist50", Min_dist50

   if not (Min_dist50=='<MinDist50/>'):
      EA_Exceed_Lim_count=EA_Exceed_Lim_count+1
      EAMin_dist50_Exceed_Lim_count=EAMin_dist50_Exceed_Lim_count+1

   RelWind= Exceed_enviro_lim_xml_tag.childNodes[5].toxml()
   RelWind=RelWind.replace('<RelWind>','').replace('</RelWind>','')
   print "rel wind", RelWind

   if not (RelWind=='0'):
      EA_Exceed_Lim_count=EA_Exceed_Lim_count+1
      EARelWind_Exceed_Lim_count=EARelWind_Exceed_Lim_count+1

   SwlHtBeam= Exceed_enviro_lim_xml_tag.childNodes[7].toxml()

   SwlHtBeam=SwlHtBeam.replace('<SwlHtBeam>','').replace('</SwlHtBeam>','')
   print "SwlHtBeam", SwlHtBeam

   if not (SwlHtBeam=='0'):
      EA_Exceed_Lim_count=EA_Exceed_Lim_count+1
      EASwlHtBeam_Exceed_Lim_count=EASwlHtBeam_Exceed_Lim_count+1

   SwlHtFollow= Exceed_enviro_lim_xml_tag.childNodes[9].toxml()
   SwlHtFollow=SwlHtFollow.replace('<SwlHtFollow>','').replace('</SwlHtFollow>','')
   print "SwlHtFollow", SwlHtFollow

   if not (SwlHtFollow=='0'):
      EA_Exceed_Lim_count=EA_Exceed_Lim_count+1
      EASwlHtFollow_Exceed_Lim_count=EASwlHtFollow_Exceed_Lim_count+1

   SwlHtHead= Exceed_enviro_lim_xml_tag.childNodes[11].toxml()
   SwlHtHead=SwlHtHead.replace('<SwlHtHead>','').replace('</SwlHtHead>','')
   print "SwlHtHead", SwlHtHead
```

```
            if not (SwlHtHead=='0'):
               EA_Exceed_Lim_count=EA_Exceed_Lim_count+1
               EASwlHtHead_Exceed_Lim_count=EASwlHtHead_Exceed_Lim_count+1

            TrueWind= Exceed_enviro_lim_xml_tag.childNodes[13].toxml()
            TrueWind=TrueWind.replace('<TrueWind>','').replace('</TrueWind>','')
            print "TrueWind", TrueWind

            if not (TrueWind=='0'):
               EA_Exceed_Lim_count=EA_Exceed_Lim_count+1
               EATrueWind_Exceed_Lim_count=EATrueWind_Exceed_Lim_count+1

            WvHtBeam= Exceed_enviro_lim_xml_tag.childNodes[15].toxml()
            WvHtBeam=WvHtBeam.replace('<WvHtBeam>','').replace('</WvHtBeam>','')
            print "WvHtBeam", WvHtBeam

            if not (WvHtBeam=='0'):
               EA_Exceed_Lim_count=EA_Exceed_Lim_count+1
               EAWvHtBeam_Exceed_Lim_count=EAWvHtBeam_Exceed_Lim_count+1

            WvHtFollow= Exceed_enviro_lim_xml_tag.childNodes[17].toxml()
            WvHtFollow=WvHtFollow.replace('<WvHtFollow>','').replace('</WvHtFollow>','')
            print "WvHtFollow", WvHtFollow

            if not (WvHtFollow=='0'):
               EA_Exceed_Lim_count=EA_Exceed_Lim_count+1
               EAWvHtFollow_Exceed_Lim_count=EAWvHtFollow_Exceed_Lim_count+1

            WvHtHead= Exceed_enviro_lim_xml_tag.childNodes[19].toxml()
            WvHtHead=WvHtHead.replace('<WvHtHead>','').replace('</WvHtHead>','')
            print "WvHtHead", WvHtHead

            if not (WvHtHead=='0'):
               EA_Exceed_Lim_count=EA_Exceed_Lim_count+1
               EAWvHtHead_Exceed_Lim_count=EAWvHtHead_Exceed_Lim_count+1

            point_ind=point_ind+1

         print "EA_Exceed_Lim_count=", EA_Exceed_Lim_count

#########################################################################

      if file_name.endswith('.analysis.analysis.xml') or file_name.endswith('.analysisV2.analysisV2.xml'):
         AN_avg_alys_Fuel=HP_xml_data
         AN_avg_alys_Dist=Dist_xml_data
         AN_avg_alys_Time=Time_xml_data

         points = dom.getElementsByTagName("PointId")
         Total_ANEnviro_lim_Exceed=0
         point_ind=0
         for point in points:
            #Point_xml_Tag = dom.getElementsByTagName('PointId')[i].toxml()
            Exceed_enviro_lim_xml_tag = dom.getElementsByTagName('EnvironLimits')[point_ind]

            Min_dist35= Exceed_enviro_lim_xml_tag.childNodes[1].toxml()
```

```
Min_dist35=Min_dist35.replace('<MinDist35>','').replace('</MinDist35>','')
print "MinDist35", Min_dist35

if not (Min_dist35=='<MinDist35/>'):
   AN_Exceed_Lim_count=AN_Exceed_Lim_count+1
   ANMin_dist35_Exceed_Lim_count=ANMin_dist35_Exceed_Lim_count+1

Min_dist50= Exceed_enviro_lim_xml_tag.childNodes[3].toxml()
Min_dist50=Min_dist50.replace('<MinDist50>','').replace('</MinDist50>','')
print "MinDist50", Min_dist50

if not (Min_dist50=='<MinDist50/>'):
   AN_Exceed_Lim_count=AN_Exceed_Lim_count+1
   ANMin_dist50_Exceed_Lim_count=ANMin_dist50_Exceed_Lim_count+1

RelWind= Exceed_enviro_lim_xml_tag.childNodes[5].toxml()
RelWind=RelWind.replace('<RelWind>','').replace('</RelWind>','')
print "rel wind", RelWind

if not (RelWind=='0'):
   AN_Exceed_Lim_count=AN_Exceed_Lim_count+1
   ANRelWind_Exceed_Lim_count=ANRelWind_Exceed_Lim_count+1

SwlHtBeam= Exceed_enviro_lim_xml_tag.childNodes[7].toxml()

SwlHtBeam=SwlHtBeam.replace('<SwlHtBeam>','').replace('</SwlHtBeam>','')
print "SwlHtBeam", SwlHtBeam

if not (SwlHtBeam=='0'):
   AN_Exceed_Lim_count=AN_Exceed_Lim_count+1
   ANSwlHtBeam_Exceed_Lim_count=ANSwlHtBeam_Exceed_Lim_count+1

SwlHtFollow= Exceed_enviro_lim_xml_tag.childNodes[9].toxml()
SwlHtFollow=SwlHtFollow.replace('<SwlHtFollow>','').replace('</SwlHtFollow>','')
print "SwlHtFollow", SwlHtFollow

if not (SwlHtFollow=='0'):
   AN_Exceed_Lim_count=AN_Exceed_Lim_count+1
   ANSwlHtFollow_Exceed_Lim_count=ANSwlHtFollow_Exceed_Lim_count+1

SwlHtHead= Exceed_enviro_lim_xml_tag.childNodes[11].toxml()
SwlHtHead=SwlHtHead.replace('<SwlHtHead>','').replace('</SwlHtHead>','')
print "SwlHtHead", SwlHtHead

if not (SwlHtHead=='0'):
   AN_Exceed_Lim_count=AN_Exceed_Lim_count+1
   ANSwlHtHead_Exceed_Lim_count=ANSwlHtHead_Exceed_Lim_count+1

TrueWind= Exceed_enviro_lim_xml_tag.childNodes[13].toxml()
TrueWind=TrueWind.replace('<TrueWind>','').replace('</TrueWind>','')
print "TrueWind", TrueWind

if not (TrueWind=='0'):
   AN_Exceed_Lim_count=AN_Exceed_Lim_count+1
   ANTrueWind_Exceed_Lim_count=ANTrueWind_Exceed_Lim_count+1
```

```
        WvHtBeam= Exceed_enviro_lim_xml_tag.childNodes[15].toxml()
        WvHtBeam=WvHtBeam.replace('<WvHtBeam>','').replace('</WvHtBeam>','')
        print "WvHtBeam", WvHtBeam

        if not (WvHtBeam=='0'):
           AN_Exceed_Lim_count=AN_Exceed_Lim_count+1
           ANWvHtBeam_Exceed_Lim_count=ANWvHtBeam_Exceed_Lim_count+1

        WvHtFollow= Exceed_enviro_lim_xml_tag.childNodes[17].toxml()
        WvHtFollow=WvHtFollow.replace('<WvHtFollow>','').replace('</WvHtFollow>','')
        print "WvHtFollow", WvHtFollow

        if not (WvHtFollow=='0'):
           AN_Exceed_Lim_count=AN_Exceed_Lim_count+1
           ANWvHtFollow_Exceed_Lim_count=ANWvHtFollow_Exceed_Lim_count+1

        WvHtHead= Exceed_enviro_lim_xml_tag.childNodes[19].toxml()
        WvHtHead=WvHtHead.replace('<WvHtHead>','').replace('</WvHtHead>','')
        print "WvHtHead", WvHtHead

        if not (WvHtHead=='0'):
           AN_Exceed_Lim_count=AN_Exceed_Lim_count+1
           ANWvHtHead_Exceed_Lim_count=ANWvHtHead_Exceed_Lim_count+1

        point_ind=point_ind+1

     print "AN_Exceed_Lim_count=", AN_Exceed_Lim_count

if file_name.endswith('.ensembleaverage.xml'):
   EA_avg_Fuel=HP_xml_data
   EA_avg_Dist=Dist_xml_data
   EA_avg_Time=Time_xml_data

########################## Analysis

if file_name.endswith(rname+'.analysis.xml') or file_name.endswith(rname+'.analysisV2.xml'):
   AN_Fuel=HP_xml_data
   AN_Dist=Dist_xml_data
   AN_Time=Time_xml_data

#########################Best Member Search##############

if re.search(r'\.[A-Z][12]\.analysis', file_name):
   print "file name MLM match"+file_name
   if float(HP_xml_data)<MLM_Fuel:
      MLM_Fuel=float(HP_xml_data)
      print "MLM fuel=",MLM_Fuel
   if float(Dist_xml_data)<MLM_Dist:
      MLM_Dist=float(Dist_xml_data)
      print "MLM dist=",MLM_Dist
   if float(Time_xml_data)<MLM_Time:
      MLM_Time=float(Time_xml_data)
      print "MLM time=",MLM_Time
```

```
        if re.search(r'\.BC_[A-Z][12]\.analysis', file_name):
           print "file name MLM match"+file_name
           if float(HP_xml_data)<MLM_Fuel:
              MLM_Fuel=float(HP_xml_data)
              print "MLM fuel=",MLM_Fuel
           if float(Dist_xml_data)<MLM_Dist:
              MLM_Dist=float(Dist_xml_data)
              print "MLM dist=",MLM_Dist
           if float(Time_xml_data)<MLM_Time:
              MLM_Time=float(Time_xml_data)
              print "MLM time=",MLM_Time

#############################################################################

        if file_name.endswith('.Analysis_NOGAPS.analysis.xml') or
file_name.endswith('.Analysis_NOGAPS.analysisV2.xml'):
           AN_NOGAPS_avg_alys_Fuel=float(HP_xml_data)
           AN_NOGAPS_avg_alys_Dist=Dist_xml_data
           AN_NOGAPS_avg_alys_Time=Time_xml_data

        if file_name.endswith('.Analysis_WW3.analysis.xml') or
file_name.endswith('.Analysis_WW3.analysisV2.xml'):
           AN_WW3_avg_alys_Fuel=HP_xml_data
           AN_WW3_avg_alys_Dist=Dist_xml_data
           AN_WW3_avg_alys_Time=Time_xml_data

        if file_name.endswith('.Analysis_NCOM.analysis.xml') or
file_name.endswith('.Analysis_NCOM.analysisV2.xml'):
           AN_NCOM_avg_alys_Fuel=HP_xml_data
           AN_NCOM_avg_alys_Dist=Dist_xml_data
           AN_NCOM_avg_alys_Time=Time_xml_data

#############################################################################

        if Dist_xml_Tag is None:
           continue
        i+=1
        small_filename = file_name[len(name)+1:]

file_line=(small_filename+','+str(HP_xml_data)+','+str(Dist_xml_data)+','+str(Time_xml_data)+'\n')
        file_string=str(file_line)
        write_stats.write(file_line)
        Array_list=[small_filename,HP_xml_data,Dist_xml_data,Time_xml_data]
        Array.append(Array_list)

Array2 = Array[1:]
print "Array="
print Array2
print "Sorted Array="

###########################################

Array_no_noroute= [x for x in Array2 if 'NOROUTEENV' not in x[0] and 'Analysis_WW3.xml' not in x[0]
           and 'Analysis_NOGAPS.xml' not in x[0] and rname+'.analysis.xml' not in x[0]
           and rname+'.analysisV2.xml' not in x[0] and 'Analysis_NCOM.xml' not in x[0] ]
```

```
Array_no_routeGC= [x for x in Array_no_noroute if 'GC.analysis.xml' not in x[0] and Array_no_noroute if
'GC.analysisV2.xml' not in x[0]]

Array_new=Array_no_routeGC[:]

names, powers, distances, times = zip(*Array_new)
#name_value_vec  = [make_label(name.rstrip('.xml')) for name in names]
power_value_vec_Orig = [float(power) for power in powers]
dist_value_vec_Orig  = [float(distance) for distance in distances]
time_value_vec_Orig = [float(time) for time in times]

######################routex only################################

Array_only_routex= [x for x in Array_no_routeGC if 'weax' not in x[0]]
names, powers, distances, times = zip(*Array_only_routex)
name_value_vec_RX=names
power_value_vec_routex = [float(power) for power in powers]
dist_value_vec_routex  = [float(distance) for distance in distances]
time_value_vec_routex = [float(time) for time in times]

######################WX only################################

Array_only_WX= [x for x in Array_no_routeGC if 'weax' in x[0]]

Array_only_WX_plus=[x for x in Array_only_WX if 'Analysis_WW3.xml' not in x[0]
          and 'Analysis_NOGAPS.xml' not in x[0] and rname+'.analysis.xml' not in x[0]
          and rname+'.analysisV2.xml' not in x[0]]

Array_only_WX_light=[x for x in Array_only_WX_plus if 'GC.analysis' not in x[0] and
Array_only_WX_plus if 'GC.analysisV2' not in x[0]
              and 'GC.analysis.analysis' not in x[0] and 'GC.analysisV2.analysisV2' not in x[0]
              and 'Analysis_WW3.analysis.xml' not in x[0] and 'Analysis_WW3.analysisV2.xml' not in x[0]
              and 'Analysis_NOGAPS.analysis.xml' not in x[0] and 'Analysis_NOGAPS.analysisV2.xml'
not in x[0]
              and 'Analysis_NCOM.analysis.xml' not in x[0] and 'Analysis_NCOM.analysisV2.xml' not in
x[0]
              and rname+'.analysis.analysis.xml' not in x[0] and rname+'.analysisV2.analysisV2.xml' not in
x[0]]

names, powers, distances, times = zip(*Array_only_WX_plus)
name_value_vec_WX=names
power_value_vec_WX = [float(power) for power in powers]
dist_value_vec_WX  = [float(distance) for distance in distances]
time_value_vec_WX = [float(time) for time in times]

names, powers, distances, times = zip(*Array_only_WX_light)
name_value_vec_WX_light=names
power_value_vec_WX_light = [float(power) for power in powers]
dist_value_vec_WX_light  = [float(distance) for distance in distances]
time_value_vec_WX_light = [float(time) for time in times]

Array_HP_Sort=Array_new[:]
Array_HP_Sort.sort(key = lambda x:float(x[1]))
names, powers, distances, times = zip(*Array_HP_Sort)
```

```
name_value_vec_HP  = [make_label(name.rstrip('.xml')) for name in names]
exp_types_HP=[name.split('.')[0] for name in names]
exp_types_HP1=[name.split('.')[2] for name in names]
exp_types_HP2=[name.split('.')[3] for name in names]
exp_types_HP3=[name.split('.')[4] for name in names]
bar_colors_HP=[WEAX_COLOR if e.lower()=='weax' else ROUTEX_COLOR for e in exp_types_HP]

bar_colors_HP=[AVG_COLOR if e.endswith('rage') and e2.endswith('sis') else orig_color for (orig_color,
e, e2) in zip(bar_colors_HP, exp_types_HP2, exp_types_HP3)]

bar_colors_HP=[AVG_COLOR if e.endswith('rage') and e2.endswith('sisV2') else orig_color for
(orig_color, e, e2) in zip(bar_colors_HP, exp_types_HP2, exp_types_HP3)]

bar_colors_HP=[MODEL_COLOR if e.endswith('GAPS') and e2.endswith('sis') else orig_color for
(orig_color, e, e2) in zip(bar_colors_HP, exp_types_HP2, exp_types_HP3)]

bar_colors_HP=[MODEL_COLOR if e.endswith('GAPS') and e2.endswith('sisV2') else orig_color for
(orig_color, e, e2) in zip(bar_colors_HP, exp_types_HP2, exp_types_HP3)]

bar_colors_HP=[MODEL_COLOR if e.endswith('WW3') and e2.endswith('sis') else orig_color for
(orig_color, e, e2) in zip(bar_colors_HP, exp_types_HP2, exp_types_HP3)]

bar_colors_HP=[MODEL_COLOR if e.endswith('WW3') and e2.endswith('sisV2') else orig_color for
(orig_color, e, e2) in zip(bar_colors_HP, exp_types_HP2, exp_types_HP3)]

bar_colors_HP=[MODEL_COLOR if e.endswith('NCOM') and e2.endswith('sis') else orig_color for
(orig_color, e, e2) in zip(bar_colors_HP, exp_types_HP2, exp_types_HP3)]

bar_colors_HP=[MODEL_COLOR if e.endswith('NCOM') and e2.endswith('sisV2') else orig_color for
(orig_color, e, e2) in zip(bar_colors_HP, exp_types_HP2, exp_types_HP3)]

bar_colors_HP=[AN_COLOR if e.endswith('sis') and e2.endswith('sis') else orig_color for (orig_color, e,
e2) in zip(bar_colors_HP, exp_types_HP2, exp_types_HP3)]

bar_colors_HP=[AN_COLOR if e.endswith('sisV2') and e2.endswith('sisV2') else orig_color for
(orig_color, e, e2) in zip(bar_colors_HP, exp_types_HP2, exp_types_HP3)]

bar_colors_HP=[GC_COLOR if f.endswith('GC') else orig_color for (orig_color, f) in zip(bar_colors_HP,
exp_types_HP1)]

power_value_vec = [float(power) for power in powers]

############################################################

Array_Dist_Sort=Array_new[:]
Array_Dist_Sort.sort(key = lambda x:float(x[2]))

names, powers, distances, times = zip(*Array_Dist_Sort)
name_value_vec_dist  = [make_label(name.rstrip('.xml')) for name in names]
exp_types_dist=[name.split('.')[0] for name in names]
exp_types_dist1=[name.split('.')[2] for name in names]
exp_types_dist2=[name.split('.')[3] for name in names]
exp_types_dist3=[name.split('.')[4] for name in names]

bar_colors_dist=[WEAX_COLOR if e.lower()=='weax' else ROUTEX_COLOR for e in exp_types_dist]
```

```
bar_colors_dist=[AVG_COLOR if e.endswith('rage') and e2.endswith('sis') else orig_color for (orig_color, e, e2) in zip(bar_colors_dist, exp_types_dist2, exp_types_dist3)]

bar_colors_dist=[AVG_COLOR if e.endswith('rage') and e2.endswith('sisV2') else orig_color for (orig_color, e, e2) in zip(bar_colors_dist, exp_types_dist2, exp_types_dist3)]

bar_colors_dist=[MODEL_COLOR if e.endswith('GAPS') and e2.endswith('sis') else orig_color for (orig_color, e, e2) in zip(bar_colors_dist, exp_types_dist2, exp_types_dist3)]

bar_colors_dist=[MODEL_COLOR if e.endswith('GAPS') and e2.endswith('sisV2') else orig_color for (orig_color, e, e2) in zip(bar_colors_dist, exp_types_dist2, exp_types_dist3)]

bar_colors_dist=[MODEL_COLOR if e.endswith('WW3') and e2.endswith('sis') else orig_color for (orig_color, e, e2) in zip(bar_colors_dist, exp_types_dist2, exp_types_dist3)]

bar_colors_dist=[MODEL_COLOR if e.endswith('WW3') and e2.endswith('sisV2') else orig_color for (orig_color, e, e2) in zip(bar_colors_dist, exp_types_dist2, exp_types_dist3)]

bar_colors_dist=[MODEL_COLOR if e.endswith('NCOM') and e2.endswith('sis') else orig_color for (orig_color, e, e2) in zip(bar_colors_dist, exp_types_dist2, exp_types_dist3)]

bar_colors_dist=[MODEL_COLOR if e.endswith('NCOM') and e2.endswith('sisV2') else orig_color for (orig_color, e, e2) in zip(bar_colors_dist, exp_types_dist2, exp_types_dist3)]

bar_colors_dist=[AN_COLOR if e.endswith('sis') and e2.endswith('sis') else orig_color for (orig_color, e, e2) in zip(bar_colors_dist, exp_types_dist2, exp_types_dist3)]

bar_colors_dist=[AN_COLOR if e.endswith('sisV2') and e2.endswith('sisV2') else orig_color for (orig_color, e, e2) in zip(bar_colors_dist, exp_types_dist2, exp_types_dist3)]

bar_colors_dist=[GC_COLOR if f.endswith('GC') else orig_color for (orig_color, f) in zip(bar_colors_dist, exp_types_dist1)]

dist_value_vec  = [float(distance) for distance in distances]

#########################################################################

Array_Time_Sort=Array_new[:]
Array_Time_Sort.sort(key = lambda x:float(x[3]))

names, powers, distances, times = zip(*Array_Time_Sort)
name_value_vec_time  = [make_label(name.rstrip('.xml')) for name in names]

exp_types_time=[name.split('.')[0] for name in names]
exp_types_time1=[name.split('.')[2] for name in names]
exp_types_time2=[name.split('.')[3] for name in names]
exp_types_time3=[name.split('.')[4] for name in names]

bar_colors_time=[WEAX_COLOR if e.lower()=='weax' else ROUTEX_COLOR for e in exp_types_time]

bar_colors_time=[AVG_COLOR if e.endswith('rage') and e2.endswith('sis') else orig_color for (orig_color, e, e2) in zip(bar_colors_time, exp_types_time2, exp_types_time3)]
```

```
bar_colors_time=[AVG_COLOR if e.endswith('rage') and e2.endswith('sisV2') else orig_color for
(orig_color, e, e2) in zip(bar_colors_time, exp_types_time2, exp_types_time3)]

bar_colors_time=[MODEL_COLOR if e.endswith('GAPS') and e2.endswith('sis') else orig_color for
(orig_color, e, e2) in zip(bar_colors_time, exp_types_time2, exp_types_time3)]

bar_colors_time=[MODEL_COLOR if e.endswith('GAPS') and e2.endswith('sisV2') else orig_color for
(orig_color, e, e2) in zip(bar_colors_time, exp_types_time2, exp_types_time3)]

bar_colors_time=[MODEL_COLOR if e.endswith('WW3') and e2.endswith('sis') else orig_color for
(orig_color, e, e2) in zip(bar_colors_time, exp_types_time2, exp_types_time3)]

bar_colors_time=[MODEL_COLOR if e.endswith('WW3') and e2.endswith('sisV2') else orig_color for
(orig_color, e, e2) in zip(bar_colors_time, exp_types_time2, exp_types_time3)]

bar_colors_time=[MODEL_COLOR if e.endswith('NCOM') and e2.endswith('sis') else orig_color for
(orig_color, e, e2) in zip(bar_colors_time, exp_types_time2, exp_types_time3)]

bar_colors_time=[MODEL_COLOR if e.endswith('NCOM') and e2.endswith('sisV2') else orig_color for
(orig_color, e, e2) in zip(bar_colors_time, exp_types_time2, exp_types_time3)]

bar_colors_time=[AN_COLOR if e.endswith('sis') and e2.endswith('sis') else orig_color for (orig_color, e,
e2) in zip(bar_colors_time, exp_types_time2, exp_types_time3)]

bar_colors_time=[AN_COLOR if e.endswith('sisV2') and e2.endswith('sisV2') else orig_color for
(orig_color, e, e2) in zip(bar_colors_time, exp_types_time2, exp_types_time3)]

bar_colors_time=[GC_COLOR if f.endswith('GC') else orig_color for (orig_color, f) in
zip(bar_colors_time, exp_types_time1)]

time_value_vec = [float(time) for time in times]

####################Fuel Stats################################
size_RX=len(power_value_vec_routex)
size_WX=len(power_value_vec_WX)
size_WX_light=len(power_value_vec_WX_light)

a = scipy.array(power_value_vec)
power_RX = scipy.array(power_value_vec_routex)
power_WX = scipy.array(power_value_vec_WX)
power_WX_light = scipy.array(power_value_vec_WX_light)

dist_RX = scipy.array(dist_value_vec_routex)
dist_WX = scipy.array(dist_value_vec_WX)
dist_WX_light = scipy.array(dist_value_vec_WX_light)

time_RX = scipy.array(time_value_vec_routex)
time_WX = scipy.array(time_value_vec_WX)
time_WX_light = scipy.array(time_value_vec_WX_light)

minp=min(a)
minp_RX=min(power_RX)
minp_WX=min(power_WX)
minp_WX_light=min(power_WX_light)
```

```
print "Power min",minp

maxp=max(a)
maxp_RX=max(power_RX)
maxp_WX=max(power_WX)
maxp_WX_light=max(power_WX_light)
print "Power max=",maxp

m = scipy.mean(a)       # compute mean along the specified axis (over entire array if axis=None)
meanp_RX=scipy.mean(power_RX)
meanp_WX=scipy.mean(power_WX)
meanp_WX_light=scipy.mean(power_WX_light)

print "Power mean=", m

v=scipy.var(a)
varp_RX=scipy.var(power_RX)
varp_WX=scipy.var(power_WX)
varp_WX_light=scipy.var(power_WX_light)

print "Power variance=",v

s = scipy.std(a)       # compute standard deviation along the specified axis (over entire array if axis=None)
stdp_RX=scipy.std(power_RX)
stdp_WX=scipy.std(power_WX)
stdp_WX_light=scipy.std(power_WX_light)

print "Power std dev=", s

HP_hist=scipy.histogram(a)

minp=round(minp, 2)
minp_RX=round(minp_RX, 2)
minp_WX=round(minp_WX, 2)
minp_WX_light=round(minp_WX_light, 2)

maxp=round(maxp,2 )
maxp_RX=round(maxp_RX, 2)
maxp_WX=round(maxp_WX, 2)
maxp_WX_light=round(maxp_WX_light, 2)

m=round(m, 2)
meanp_RX=round(meanp_RX, 2)
meanp_WX=round(meanp_WX, 2)
meanp_WX_light=round(meanp_WX_light, 2)

v=round(v, 2)
varp_RX=round(varp_RX, 2)
varp_WX=round(varp_WX, 2)
varp_WX_light=round(varp_WX_light, 2)

s=round(s, 2)
stdp_RX=round(stdp_RX, 2)
stdp_WX=round(stdp_WX, 2)
stdp_WX_light=round(stdp_WX_light, 2)
```

```
###################Dist Stats#################################

b = scipy.array(dist_value_vec)

mind=min(b)
mind_RX=min(dist_RX)
mind_WX=min(dist_WX)
mind_WX_light=min(dist_WX_light)

print "Dist min=",mind

maxd=max(b)
maxd_RX=max(dist_RX)
maxd_WX=max(dist_WX)
maxd_WX_light=max(dist_WX_light)
print "Dist max=",maxd

md = scipy.mean(b)       # compute mean along the specified axis (over entire array if axis=None)
meand_RX=scipy.mean(dist_RX)
meand_WX=scipy.mean(dist_WX)
meand_WX_light=scipy.mean(dist_WX_light)
print "Dist mean=", md

vd=scipy.var(b)
vard_RX=scipy.var(dist_RX)
vard_WX=scipy.var(dist_WX)
vard_WX_light=scipy.var(dist_WX_light)
print "Dist variance=",vd

sd = scipy.std(b)        # compute standard deviation along the specified axis (over entire array if axis=None)
stdd_RX=scipy.std(dist_RX)
stdd_WX=scipy.std(dist_WX)
stdd_WX_light=scipy.std(dist_WX_light)
print "Dist std dev=", sd

###################Time Stats#################################

t = scipy.array(time_value_vec)

mint=min(t)
mint_RX=min(time_RX)
mint_WX=min(time_WX)
mint_WX_light=min(time_WX_light)

print "Time min=",mint
maxt=max(t)
maxt_RX=max(time_RX)
maxt_WX=max(time_WX)
maxt_WX_light=max(time_WX_light)
print "Time max=",maxt

mt = scipy.mean(t)       # compute mean along the specified axis (over entire array if axis=None)
meant_RX=scipy.mean(time_RX)
meant_WX=scipy.mean(time_WX)
```

```
meant_WX_light=scipy.mean(time_WX_light)
print "Time mean=", mt

vt=scipy.var(t)
vart_RX=scipy.var(time_RX)
vart_WX=scipy.var(time_WX)
vart_WX_light=scipy.var(time_WX_light)
print "Time variance=",vt

st = scipy.std(t)       # compute standard deviation along the specified axis (over entire array if axis=None)
stdt_RX=scipy.std(time_RX)
stdt_WX=scipy.std(time_WX)
stdt_WX_light=scipy.std(time_WX_light)
print "Time std dev=", st

#return mean, variance, standard_deviation

mind=round(mind, 2)
mind_RX=round(mind_RX, 2)
mind_WX=round(mind_WX, 2)
mind_WX_light=round(mind_WX_light, 2)

maxd=round(maxd,2 )
maxd_RX=round(maxd_RX, 2)
maxd_WX=round(maxd_WX, 2)
maxd_WX_light=round(maxd_WX_light, 2)

md=round(md, 2)
meand_RX=round(meand_RX, 2)
meand_WX=round(meand_WX, 2)
meand_WX_light=round(meand_WX_light, 2)

vd=round(vd, 2)
vard_RX=round(vard_RX, 2)
vard_WX=round(vard_WX, 2)
vard_WX_light=round(vard_WX_light, 2)

sd=round(sd, 2)
stdd_RX=round(stdd_RX, 2)
stdd_WX=round(stdd_WX, 2)
stdd_WX_light=round(stdd_WX_light, 2)

mint=round(mint, 2)
mint_RX=round(mint_RX, 2)
mint_WX=round(mint_WX, 2)
mint_WX_light=round(mint_WX_light, 2)

maxt=round(maxt,2 )
maxt_RX=round(maxt_RX, 2)
maxt_WX=round(maxt_WX, 2)
maxt_WX_light=round(maxt_WX_light, 2)

m=round(m, 2)
meant_RX=round(meant_RX, 2)
meant_WX=round(meant_WX, 2)
```

```
meant_WX_light=round(meant_WX_light, 2)

v=round(v, 2)
vart_RX=round(vart_RX, 2)
vart_WX=round(vart_WX, 2)
vart_WX_light=round(vart_WX_light, 2)

s=round(s, 2)
stdt_RX=round(stdt_RX, 2)
stdt_WX=round(stdt_WX, 2)
stdt_WX_light=round(stdt_WX_light, 2)

###################Fuel Comparison STATS##############################

GC_Fuel=float(GC_Fuel)
GC_Dist=float(GC_Dist)
GC_Time=float(GC_Time)

##############################
Hy_avg_alys_Fuel=float(Hy_avg_alys_Fuel)
Hy_avg_Fuel=float(Hy_avg_Fuel)
BC_avg_alys_Fuel=float(BC_avg_alys_Fuel)
BC_avg_Fuel=float(BC_avg_Fuel)
EA_avg_alys_Fuel=float(EA_avg_alys_Fuel)
EA_avg_Fuel=float(EA_avg_Fuel)
AN_Fuel=float(AN_Fuel)
AN_NOGAPS_avg_alys_Fuel=float(AN_NOGAPS_avg_alys_Fuel)
AN_WW3_avg_alys_Fuel=float(AN_WW3_avg_alys_Fuel)
AN_NCOM_avg_alys_Fuel=float(AN_NCOM_avg_alys_Fuel)

##############################
Hy_avg_alys_Dist=float(Hy_avg_alys_Dist)
Hy_avg_Dist=float(Hy_avg_Dist)
BC_avg_alys_Dist=float(BC_avg_alys_Dist)
BC_avg_Dist=float(BC_avg_Dist)
EA_avg_alys_Dist=float(EA_avg_alys_Dist)
EA_avg_Dist=float(EA_avg_Dist)
AN_Dist=float(AN_Dist)
AN_NOGAPS_avg_alys_Dist=float(AN_NOGAPS_avg_alys_Dist)
AN_WW3_avg_alys_Dist=float(AN_WW3_avg_alys_Dist)
AN_NCOM_avg_alys_Dist=float(AN_NCOM_avg_alys_Dist)

##############################
Hy_avg_alys_Time=float(Hy_avg_alys_Time)
Hy_avg_Time=float(Hy_avg_Time)
BC_avg_alys_Time=float(BC_avg_alys_Time)
BC_avg_Time=float(BC_avg_Time)
EA_avg_alys_Time=float(EA_avg_alys_Time)
EA_avg_Time=float(EA_avg_Time)
AN_Time=float(AN_Time)
AN_NOGAPS_avg_alys_Time=float(AN_NOGAPS_avg_alys_Time)
AN_WW3_avg_alys_Time=float(AN_WW3_avg_alys_Time)
AN_NCOM_avg_alys_Time=float(AN_NCOM_avg_alys_Time)

##############################################
```

Hy_avg_alys_Fuel_GC=Hy_avg_alys_Fuel-GC_Fuel
Hy_avg_alys_Fuel_AVG=(GC_Fuel+Hy_avg_alys_Fuel)/2
Hy_avg_alys_Fuel_GC_DIFF=(Hy_avg_alys_Fuel_GC/Hy_avg_alys_Fuel_AVG)*100

Hy_avg_alys_Fuel_Pred=Hy_avg_Fuel-Hy_avg_alys_Fuel
Hy_avg_alys_Fuel_AVG_Pred=(Hy_avg_alys_Fuel+Hy_avg_Fuel)/2
Hy_avg_alys_Fuel_Pred_DIFF=(Hy_avg_alys_Fuel_Pred/Hy_avg_alys_Fuel_AVG_Pred)*100

Hy_avg_alys_Fuel_GC_DIFF=round(Hy_avg_alys_Fuel_GC_DIFF,2)
Hy_avg_alys_Fuel_Pred_DIFF=round(Hy_avg_alys_Fuel_Pred_DIFF,2)

#############

Hy_avg_alys_Dist_GC=Hy_avg_alys_Dist-GC_Dist
Hy_avg_alys_Dist_AVG=(GC_Dist+Hy_avg_alys_Dist)/2
Hy_avg_alys_Dist_GC_DIFF=(Hy_avg_alys_Dist_GC/Hy_avg_alys_Dist_AVG)*100

Hy_avg_alys_Dist_Pred=Hy_avg_Dist-Hy_avg_alys_Dist
Hy_avg_alys_Dist_AVG_Pred=(Hy_avg_alys_Dist+Hy_avg_Dist)/2
Hy_avg_alys_Dist_Pred_DIFF=(Hy_avg_alys_Dist_Pred/Hy_avg_alys_Dist_AVG_Pred)*100

Hy_avg_alys_Dist_GC_DIFF=round(Hy_avg_alys_Dist_GC_DIFF,2)
Hy_avg_alys_Dist_Pred_DIFF=round(Hy_avg_alys_Dist_Pred_DIFF,2)

#############

Hy_avg_alys_Time_GC=Hy_avg_alys_Time-GC_Time
Hy_avg_alys_Time_AVG=(GC_Time+Hy_avg_alys_Time)/2
Hy_avg_alys_Time_GC_DIFF=(Hy_avg_alys_Time_GC/Hy_avg_alys_Time_AVG)*100

Hy_avg_alys_Time_Pred=Hy_avg_Time-Hy_avg_alys_Time
Hy_avg_alys_Time_AVG_Pred=(Hy_avg_alys_Time+Hy_avg_Time)/2
Hy_avg_alys_Time_Pred_DIFF=(Hy_avg_alys_Time_Pred/Hy_avg_alys_Time_AVG_Pred)*100

Hy_avg_alys_Time_GC_DIFF=round(Hy_avg_alys_Time_GC_DIFF,2)
Hy_avg_alys_Time_Pred_DIFF=round(Hy_avg_alys_Time_Pred_DIFF,2)

##

BC_avg_alys_Fuel_GC=BC_avg_alys_Fuel-GC_Fuel
BC_avg_alys_Fuel_AVG=(GC_Fuel+BC_avg_alys_Fuel)/2
BC_avg_alys_Fuel_GC_DIFF=(BC_avg_alys_Fuel_GC/BC_avg_alys_Fuel_AVG)*100

BC_avg_alys_Fuel_Pred=BC_avg_Fuel-BC_avg_alys_Fuel
BC_avg_alys_Fuel_AVG_Pred=(BC_avg_alys_Fuel+BC_avg_Fuel)/2
BC_avg_alys_Fuel_Pred_DIFF=(BC_avg_alys_Fuel_Pred/BC_avg_alys_Fuel_AVG_Pred)*100

BC_avg_alys_Fuel_GC_DIFF=round(BC_avg_alys_Fuel_GC_DIFF,2)
BC_avg_alys_Fuel_Pred_DIFF=round(BC_avg_alys_Fuel_Pred_DIFF,2)

#########################

BC_avg_alys_Dist_GC=BC_avg_alys_Dist-GC_Dist
BC_avg_alys_Dist_AVG=(GC_Dist+BC_avg_alys_Dist)/2
BC_avg_alys_Dist_GC_DIFF=(BC_avg_alys_Dist_GC/BC_avg_alys_Dist_AVG)*100

```
BC_avg_alys_Dist_Pred=BC_avg_Dist-BC_avg_alys_Dist
BC_avg_alys_Dist_AVG_Pred=(BC_avg_alys_Dist+BC_avg_Dist)/2
BC_avg_alys_Dist_Pred_DIFF=(BC_avg_alys_Dist_Pred/BC_avg_alys_Dist_AVG_Pred)*100

BC_avg_alys_Dist_GC_DIFF=round(BC_avg_alys_Dist_GC_DIFF,2)
BC_avg_alys_Dist_Pred_DIFF=round(BC_avg_alys_Dist_Pred_DIFF,2)

##########################

BC_avg_alys_Time_GC=BC_avg_alys_Time-GC_Time
BC_avg_alys_Time_AVG=(GC_Time+BC_avg_alys_Time)/2
BC_avg_alys_Time_GC_DIFF=(BC_avg_alys_Time_GC/BC_avg_alys_Time_AVG)*100

BC_avg_alys_Time_Pred=BC_avg_Time-BC_avg_alys_Time
BC_avg_alys_Time_AVG_Pred=(BC_avg_alys_Time+BC_avg_Time)/2
BC_avg_alys_Time_Pred_DIFF=(BC_avg_alys_Time_Pred/BC_avg_alys_Time_AVG_Pred)*100

BC_avg_alys_Time_GC_DIFF=round(BC_avg_alys_Time_GC_DIFF,2)
BC_avg_alys_Time_Pred_DIFF=round(BC_avg_alys_Time_Pred_DIFF,2)

############################################################

EA_avg_alys_Fuel_GC=EA_avg_alys_Fuel-GC_Fuel
EA_avg_alys_Fuel_AVG=(GC_Fuel+EA_avg_alys_Fuel)/2
EA_avg_alys_Fuel_GC_DIFF=(EA_avg_alys_Fuel_GC/EA_avg_alys_Fuel_AVG)*100

EA_avg_alys_Fuel_Pred=EA_avg_Fuel-EA_avg_alys_Fuel
EA_avg_alys_Fuel_AVG_Pred=(EA_avg_alys_Fuel+EA_avg_Fuel)/2
EA_avg_alys_Fuel_Pred_DIFF=(EA_avg_alys_Fuel_Pred/EA_avg_alys_Fuel_AVG_Pred)*100

EA_avg_alys_Fuel_GC_DIFF=round(EA_avg_alys_Fuel_GC_DIFF,2)
EA_avg_alys_Fuel_Pred_DIFF=round(EA_avg_alys_Fuel_Pred_DIFF,2)

################################

EA_avg_alys_Dist_GC=EA_avg_alys_Dist-GC_Dist
EA_avg_alys_Dist_AVG=(GC_Dist+EA_avg_alys_Dist)/2
EA_avg_alys_Dist_GC_DIFF=(EA_avg_alys_Dist_GC/EA_avg_alys_Dist_AVG)*100

EA_avg_alys_Dist_Pred=EA_avg_Dist-EA_avg_alys_Dist
EA_avg_alys_Dist_AVG_Pred=(EA_avg_alys_Dist+EA_avg_Dist)/2
EA_avg_alys_Dist_Pred_DIFF=(EA_avg_alys_Dist_Pred/EA_avg_alys_Dist_AVG_Pred)*100

EA_avg_alys_Dist_GC_DIFF=round(EA_avg_alys_Dist_GC_DIFF,2)
EA_avg_alys_Dist_Pred_DIFF=round(EA_avg_alys_Dist_Pred_DIFF,2)

################################

EA_avg_alys_Time_GC=EA_avg_alys_Time-GC_Time
EA_avg_alys_Time_AVG=(GC_Time+EA_avg_alys_Time)/2
EA_avg_alys_Time_GC_DIFF=(EA_avg_alys_Time_GC/EA_avg_alys_Time_AVG)*100

EA_avg_alys_Time_Pred=EA_avg_Time-EA_avg_alys_Time
EA_avg_alys_Time_AVG_Pred=(EA_avg_alys_Time+EA_avg_Time)/2
```

```
EA_avg_alys_Time_Pred_DIFF=(EA_avg_alys_Time_Pred/EA_avg_alys_Time_AVG_Pred)*100

EA_avg_alys_Time_GC_DIFF=round(EA_avg_alys_Time_GC_DIFF,2)
EA_avg_alys_Time_Pred_DIFF=round(EA_avg_alys_Time_Pred_DIFF,2)

############################################################

AN_NOGAPS_avg_alys_Fuel_GC=AN_NOGAPS_avg_alys_Fuel-GC_Fuel
AN_NOGAPS_avg_alys_Fuel_AVG=(GC_Fuel+AN_NOGAPS_avg_alys_Fuel)/2
AN_NOGAPS_avg_alys_Fuel_GC_DIFF=(AN_NOGAPS_avg_alys_Fuel_GC/AN_NOGAPS_avg_alys_
Fuel_AVG)*100
AN_NOGAPS_avg_alys_Fuel_GC_DIFF=round(AN_NOGAPS_avg_alys_Fuel_GC_DIFF,2)

################################

AN_NOGAPS_avg_alys_Dist_GC=AN_NOGAPS_avg_alys_Dist-GC_Dist
AN_NOGAPS_avg_alys_Dist_AVG=(GC_Dist+AN_NOGAPS_avg_alys_Dist)/2
AN_NOGAPS_avg_alys_Dist_GC_DIFF=(AN_NOGAPS_avg_alys_Dist_GC/AN_NOGAPS_avg_alys_D
ist_AVG)*100
AN_NOGAPS_avg_alys_Dist_GC_DIFF=round(AN_NOGAPS_avg_alys_Dist_GC_DIFF,2)

################################
AN_NOGAPS_avg_alys_Time_GC=AN_NOGAPS_avg_alys_Time-GC_Time
AN_NOGAPS_avg_alys_Time_AVG=(GC_Time+AN_NOGAPS_avg_alys_Time)/2
AN_NOGAPS_avg_alys_Time_GC_DIFF=(AN_NOGAPS_avg_alys_Time_GC/AN_NOGAPS_avg_alys
_Time_AVG)*100
AN_NOGAPS_avg_alys_Time_GC_DIFF=round(AN_NOGAPS_avg_alys_Time_GC_DIFF,2)

############################################

AN_WW3_avg_alys_Fuel_GC=AN_WW3_avg_alys_Fuel-GC_Fuel
AN_WW3_avg_alys_Fuel_AVG=(GC_Fuel+AN_WW3_avg_alys_Fuel)/2
AN_WW3_avg_alys_Fuel_GC_DIFF=(AN_WW3_avg_alys_Fuel_GC/AN_WW3_avg_alys_Fuel_AVG)*
100

AN_WW3_avg_alys_Fuel_GC_DIFF=round(AN_WW3_avg_alys_Fuel_GC_DIFF,2)

################################

AN_WW3_avg_alys_Dist_GC=AN_WW3_avg_alys_Dist-GC_Dist
AN_WW3_avg_alys_Dist_AVG=(GC_Dist+AN_WW3_avg_alys_Dist)/2
AN_WW3_avg_alys_Dist_GC_DIFF=(AN_WW3_avg_alys_Dist_GC/AN_WW3_avg_alys_Dist_AVG)*
100
AN_WW3_avg_alys_Dist_GC_DIFF=round(AN_WW3_avg_alys_Dist_GC_DIFF,2)

AN_WW3_avg_alys_Time_GC=AN_WW3_avg_alys_Time-GC_Time
AN_WW3_avg_alys_Time_AVG=(GC_Time+AN_WW3_avg_alys_Time)/2
AN_WW3_avg_alys_Time_GC_DIFF=(AN_WW3_avg_alys_Time_GC/AN_WW3_avg_alys_Time_AV
G)*100

AN_WW3_avg_alys_Time_GC_DIFF=round(AN_WW3_avg_alys_Time_GC_DIFF,2)

############################################

AN_NCOM_avg_alys_Fuel_GC=AN_NCOM_avg_alys_Fuel-GC_Fuel
```

```
AN_NCOM_avg_alys_Fuel_AVG=(GC_Fuel+AN_NCOM_avg_alys_Fuel)/2
AN_NCOM_avg_alys_Fuel_GC_DIFF=(AN_NCOM_avg_alys_Fuel_GC/AN_NCOM_avg_alys_Fuel_A
VG)*100

AN_NCOM_avg_alys_Fuel_GC_DIFF=round(AN_NCOM_avg_alys_Fuel_GC_DIFF,2)
################################
AN_NCOM_avg_alys_Dist_GC=AN_NCOM_avg_alys_Dist-GC_Dist
AN_NCOM_avg_alys_Dist_AVG=(GC_Dist+AN_NCOM_avg_alys_Dist)/2
AN_NCOM_avg_alys_Dist_GC_DIFF=(AN_NCOM_avg_alys_Dist_GC/AN_NCOM_avg_alys_Dist_AV
G)*100
AN_NCOM_avg_alys_Dist_GC_DIFF=round(AN_NCOM_avg_alys_Dist_GC_DIFF,2)
################################

AN_NCOM_avg_alys_Time_GC=AN_NCOM_avg_alys_Time-GC_Time
AN_NCOM_avg_alys_Time_AVG=(GC_Time+AN_NCOM_avg_alys_Time)/2
AN_NCOM_avg_alys_Time_GC_DIFF=(AN_NCOM_avg_alys_Time_GC/AN_NCOM_avg_alys_Time_
AVG)*100

AN_NCOM_avg_alys_Time_GC_DIFF=round(AN_NCOM_avg_alys_Time_GC_DIFF,2)

##############################################

AN_avg_alys_Fuel_GC=AN_Fuel-GC_Fuel
AN_avg_alys_Fuel_AVG=(GC_Fuel+AN_Fuel)/2
AN_avg_alys_Fuel_GC_DIFF=(AN_avg_alys_Fuel_GC/AN_avg_alys_Fuel_AVG)*100
AN_avg_alys_Fuel_GC_DIFF=round(AN_avg_alys_Fuel_GC_DIFF,2)

################################

AN_avg_alys_Dist_GC=AN_Dist-GC_Dist
AN_avg_alys_Dist_AVG=(GC_Dist+AN_Dist)/2
AN_avg_alys_Dist_GC_DIFF=(AN_avg_alys_Dist_GC/AN_avg_alys_Dist_AVG)*100
AN_avg_alys_Dist_GC_DIFF=round(AN_avg_alys_Dist_GC_DIFF,2)

################################

AN_avg_alys_Time_GC=AN_Time-GC_Time
AN_avg_alys_Time_AVG=(GC_Time+AN_Time)/2
AN_avg_alys_Time_GC_DIFF=(AN_avg_alys_Time_GC/AN_avg_alys_Time_AVG)*100
AN_avg_alys_Time_GC_DIFF=round(AN_avg_alys_Time_GC_DIFF,2)

##############################################

MLM_avg_alys_Fuel_GC=MLM_Fuel-GC_Fuel
MLM_avg_alys_Fuel_AVG=(GC_Fuel+MLM_Fuel)/2
MLM_avg_alys_Fuel_GC_DIFF=(MLM_avg_alys_Fuel_GC/MLM_avg_alys_Fuel_AVG)*100
MLM_avg_alys_Fuel_GC_DIFF=round(MLM_avg_alys_Fuel_GC_DIFF,2)

##############################################

MLM_avg_alys_Dist_GC=MLM_Dist-GC_Dist
MLM_avg_alys_Dist_AVG=(GC_Dist+MLM_Dist)/2
MLM_avg_alys_Dist_GC_DIFF=(MLM_avg_alys_Dist_GC/MLM_avg_alys_Dist_AVG)*100
MLM_avg_alys_Dist_GC_DIFF=round(MLM_avg_alys_Dist_GC_DIFF,2)
```

```
##############################################

MLM_avg_alys_Time_GC=MLM_Time-GC_Time
MLM_avg_alys_Time_AVG=(GC_Time+MLM_Time)/2
MLM_avg_alys_Time_GC_DIFF=(MLM_avg_alys_Time_GC/MLM_avg_alys_Time_AVG)*100
MLM_avg_alys_Time_GC_DIFF=round(MLM_avg_alys_Time_GC_DIFF,2)

############################################################

mind=round(mind, 2)
maxd=round(maxd,2 )
md=round(md, 2)
vd=round(vd, 2)
sd=round(sd, 2)

mint=round(mint, 2)
maxt=round(maxt,2 )
mt=round(mt, 2)
vt=round(vt, 2)
st=round(st, 2)

print "Hybrid_ensemble_GC_diff=", Hy_avg_alys_Fuel_GC_DIFF
print "Hybrid_ensemble_pred_diff=", Hy_avg_alys_Fuel_Pred_DIFF

print "BC_ensemble_GC_diff=", BC_avg_alys_Fuel_GC_DIFF
print "BC_ensemble_pred_diff=", BC_avg_alys_Fuel_Pred_DIFF

print "Ensemble_AVG_GC_diff=", EA_avg_alys_Fuel_GC_DIFF
print "Ensemble_AVG_pred_diff=", EA_avg_alys_Fuel_Pred_DIFF

print "Dist Hybrid_ensemble_GC_diff=", Hy_avg_alys_Dist_GC_DIFF
print "Hybrid_ensemble_pred_diff=", Hy_avg_alys_Dist_Pred_DIFF

print "BC_ensemble_GC_diff=", BC_avg_alys_Dist_GC_DIFF
print "BC_ensemble_pred_diff=", BC_avg_alys_Dist_Pred_DIFF

print "Ensemble_AVG_GC_diff=", EA_avg_alys_Dist_GC_DIFF
print "Ensemble_AVG_pred_diff=", EA_avg_alys_Dist_Pred_DIFF

print "Time Hybrid_ensemble_GC_diff=", Hy_avg_alys_Time_GC_DIFF
print "Hybrid_ensemble_pred_diff=", Hy_avg_alys_Time_Pred_DIFF

print "BC_ensemble_GC_diff=", BC_avg_alys_Time_GC_DIFF
print "BC_ensemble_pred_diff=", BC_avg_alys_Time_Pred_DIFF

print "Ensemble_AVG_GC_diff=", EA_avg_alys_Time_GC_DIFF
print "Ensemble_AVG_pred_diff=", EA_avg_alys_Time_Pred_DIFF

########################Write Stats#################
write_stats.write('Overall:, Power Min, Max, Mean, Variance, STD'+'\n')
write_stats.write(str(minp)+','+str(maxp)+','+str(m)+','+str(v)+','+str(s)+'\n')

write_stats.write('Overall:, Distance Min, Max, Mean, Variance, STD'+'\n')
write_stats.write(str(mind)+','+str(maxd)+','+str(md)+','+str(vd)+','+str(sd)+'\n')
```

```
write_stats.write('Overall:, Time Min, Max, Mean, Variance, STD'+'\n')
write_stats.write(str(mint)+','+str(maxt)+','+str(mt)+','+str(vt)+','+str(st)+'\n')

####################################write stats HY######

write_stats.write('HY Ensemble GC Diff:, (Fuel), (Dist), (Time)'+'\n')
write_stats.write(str(Hy_avg_alys_Fuel_GC_DIFF)+','+str(Hy_avg_alys_Dist_GC_DIFF)+','+str(Hy_avg_
alys_Time_GC_DIFF)+'\n')

write_stats.write('HY Ensemble Fcst Diff:, (Fuel), (Dist), (Time)'+'\n')
write_stats.write(str(Hy_avg_alys_Fuel_Pred_DIFF)+','+str(Hy_avg_alys_Dist_Pred_DIFF)+','+str(Hy_av
g_alys_Time_Pred_DIFF)+'\n')

####################################write stats BC###########

write_stats.write('BC Ensemble GC Diff:, (Fuel), (Dist), (Time)'+'\n')
write_stats.write(str(BC_avg_alys_Fuel_GC_DIFF)+','+str(BC_avg_alys_Dist_GC_DIFF)+','+str(BC_avg
_alys_Time_GC_DIFF)+'\n')

write_stats.write('BC Ensemble Fcst Diff:, (Fuel), (Dist), (Time)'+'\n')
write_stats.write(str(BC_avg_alys_Fuel_Pred_DIFF)+','+str(BC_avg_alys_Dist_Pred_DIFF)+','+str(BC_av
g_alys_Time_Pred_DIFF)+'\n')

####################################write stats EA###########

write_stats.write('EA Ensemble GC Diff:, (Fuel), (Dist), (Time)'+'\n')
write_stats.write(str(EA_avg_alys_Fuel_GC_DIFF)+','+str(EA_avg_alys_Dist_GC_DIFF)+','+str(EA_avg
_alys_Time_GC_DIFF)+'\n')

write_stats.write('EA Ensemble Fcst Diff:, (Fuel), (Dist), (Time)'+'\n')
write_stats.write(str(EA_avg_alys_Fuel_Pred_DIFF)+','+str(EA_avg_alys_Dist_Pred_DIFF)+','+str(EA_av
g_alys_Time_Pred_DIFF)+'\n')

####################################write stats AN##########

write_stats.write('AN Ensemble GC Diff:, (Fuel), (Dist), (Time)'+'\n')
write_stats.write(str(AN_avg_alys_Fuel_GC_DIFF)+','+str(AN_avg_alys_Dist_GC_DIFF)+','+str(AN_avg
_alys_Time_GC_DIFF)+'\n')

####################################write stats best member#######

write_stats.write('Best Ensemble GC Diff:, (Fuel), (Dist), (Time)'+'\n')
write_stats.write(str(MLM_avg_alys_Fuel_GC_DIFF)+','+str(MLM_avg_alys_Dist_GC_DIFF)+','+str(ML
M_avg_alys_Time_GC_DIFF)+'\n')

####################################write stats AN_NOGAPS######

write_stats.write('AN_NOGAPS GC Diff:, (Fuel), (Dist), (Time)'+'\n')
write_stats.write(str(AN_NOGAPS_avg_alys_Fuel_GC_DIFF)+','+str(AN_NOGAPS_avg_alys_Dist_GC_
DIFF)+','+str(AN_NOGAPS_avg_alys_Time_GC_DIFF)+'\n')

####################################write stats AN_WW3#########

write_stats.write('AN_WW3 GC Diff:, (Fuel), (Dist), (Time)'+'\n')
```

```
write_stats.write(str(AN_WW3_avg_alys_Fuel_GC_DIFF)+','+str(AN_WW3_avg_alys_Dist_GC_DIFF)+'
,'+str(AN_WW3_avg_alys_Time_GC_DIFF)+'\n')

####################################write stats AN_NCOM#########

write_stats.write('AN_NCOM GC Diff:, (Fuel), (Dist), (Time)'+'\n')
write_stats.write(str(AN_NCOM_avg_alys_Fuel_GC_DIFF)+','+str(AN_NCOM_avg_alys_Dist_GC_DIF
F)+','+str(AN_NCOM_avg_alys_Time_GC_DIFF)+'\n')

####################################write stats Exceed Limits########
write_stats.write('Exceed Limits: (GC), (HY), (BC), (EA), (AN)'+'\n')
write_stats.write(str(GC_Exceed_Lim_count)+','+str(HY_Exceed_Lim_count)+','+str(BC_Exceed_Lim_co
unt)+','+str(EA_Exceed_Lim_count)+','+str(AN_Exceed_Lim_count)+'\n')

write_stats.write('Exceed Limits (Great Circle):'+'\n')
write_stats.write('Min_dist35='+str(GCMin_dist35_Exceed_Lim_count)+'\n')
write_stats.write('Min_dist50='+str(GCMin_dist50_Exceed_Lim_count)+'\n')
write_stats.write('RelWind='+str(GCRelWind_Exceed_Lim_count)+'\n')
write_stats.write('SwlHtBeam='+str(GCSwlHtBeam_Exceed_Lim_count)+'\n')
write_stats.write('SwlHtFollow='+str(GCSwlHtFollow_Exceed_Lim_count)+'\n')
write_stats.write('SwlHtHead='+str(GCSwlHtHead_Exceed_Lim_count)+'\n')
write_stats.write('TrueWind='+str(GCTrueWind_Exceed_Lim_count)+'\n')
write_stats.write('WvHtBeam='+str(GCWvHtBeam_Exceed_Lim_count)+'\n')
write_stats.write('WvHtFollow='+str(GCWvHtFollow_Exceed_Lim_count)+'\n')
write_stats.write('WvHtHead='+str(GCWvHtHead_Exceed_Lim_count)+'\n')

write_stats.write('Exceed Limits (Hybrid):'+'\n')
write_stats.write('Min_dist35='+str(HYMin_dist35_Exceed_Lim_count)+'\n')
write_stats.write('Min_dist50='+str(HYMin_dist50_Exceed_Lim_count)+'\n')
write_stats.write('RelWind='+str(HYRelWind_Exceed_Lim_count)+'\n')
write_stats.write('SwlHtBeam='+str(HYSwlHtBeam_Exceed_Lim_count)+'\n')
write_stats.write('SwlHtFollow='+str(HYSwlHtFollow_Exceed_Lim_count)+'\n')
write_stats.write('SwlHtHead='+str(HYSwlHtHead_Exceed_Lim_count)+'\n')
write_stats.write('TrueWind='+str(HYTrueWind_Exceed_Lim_count)+'\n')
write_stats.write('WvHtBeam='+str(HYWvHtBeam_Exceed_Lim_count)+'\n')
write_stats.write('WvHtFollow='+str(HYWvHtFollow_Exceed_Lim_count)+'\n')
write_stats.write('WvHtHead='+str(HYWvHtHead_Exceed_Lim_count)+'\n')

write_stats.write('Exceed Limits (Bias Corrected):'+'\n')
write_stats.write('Min_dist35='+str(BCMin_dist35_Exceed_Lim_count)+'\n')
write_stats.write('Min_dist50='+str(BCMin_dist50_Exceed_Lim_count)+'\n')
write_stats.write('RelWind='+str(BCRelWind_Exceed_Lim_count)+'\n')
write_stats.write('SwlHtBeam='+str(BCSwlHtBeam_Exceed_Lim_count)+'\n')
write_stats.write('SwlHtFollow='+str(BCSwlHtFollow_Exceed_Lim_count)+'\n')
write_stats.write('SwlHtHead='+str(BCSwlHtHead_Exceed_Lim_count)+'\n')
write_stats.write('TrueWind='+str(BCTrueWind_Exceed_Lim_count)+'\n')
write_stats.write('WvHtBeam='+str(BCWvHtBeam_Exceed_Lim_count)+'\n')
write_stats.write('WvHtFollow='+str(BCWvHtFollow_Exceed_Lim_count)+'\n')
write_stats.write('WvHtHead='+str(BCWvHtHead_Exceed_Lim_count)+'\n')

write_stats.write('Exceed Limits (Ensemble Avg):'+'\n')
write_stats.write('Min_dist35='+str(EAMin_dist35_Exceed_Lim_count)+'\n')
write_stats.write('Min_dist50='+str(EAMin_dist50_Exceed_Lim_count)+'\n')
write_stats.write('RelWind='+str(EARelWind_Exceed_Lim_count)+'\n')
write_stats.write('SwlHtBeam='+str(EASwlHtBeam_Exceed_Lim_count)+'\n')
```

```
write_stats.write('SwlHtFollow='+str(EASwlHtFollow_Exceed_Lim_count)+'\n')
write_stats.write('SwlHtHead='+str(EASwlHtHead_Exceed_Lim_count)+'\n')
write_stats.write('TrueWind='+str(EATrueWind_Exceed_Lim_count)+'\n')
write_stats.write('WvHtBeam='+str(EAWvHtBeam_Exceed_Lim_count)+'\n')
write_stats.write('WvHtFollow='+str(EAWvHtFollow_Exceed_Lim_count)+'\n')
write_stats.write('WvHtHead='+str(EAWvHtHead_Exceed_Lim_count)+'\n')

write_stats.write('Exceed Limits (Analysis):'+'\n')
write_stats.write('Min_dist35='+str(ANMin_dist35_Exceed_Lim_count)+'\n')
write_stats.write('Min_dist50='+str(ANMin_dist50_Exceed_Lim_count)+'\n')
write_stats.write('RelWind='+str(ANRelWind_Exceed_Lim_count)+'\n')
write_stats.write('SwlHtBeam='+str(ANSwlHtBeam_Exceed_Lim_count)+'\n')
write_stats.write('SwlHtFollow='+str(ANSwlHtFollow_Exceed_Lim_count)+'\n')
write_stats.write('SwlHtHead='+str(ANSwlHtHead_Exceed_Lim_count)+'\n')
write_stats.write('TrueWind='+str(ANTrueWind_Exceed_Lim_count)+'\n')
write_stats.write('WvHtBeam='+str(ANWvHtBeam_Exceed_Lim_count)+'\n')
write_stats.write('WvHtFollow='+str(ANWvHtFollow_Exceed_Lim_count)+'\n')
write_stats.write('WvHtHead='+str(ANWvHtHead_Exceed_Lim_count)+'\n')
write_stats.close()
print "end write stats"
print "begin index find"
ind = np.arange(len(power_value_vec))    # the x locations for the groups
print "end index find"

################################################Fuel 2D #####
width = 1       # the width of the bars
plt.figure(1)
fig = plt.figure(1)
axes = fig.add_subplot(111)
axes.set_autoscaley_on(True)
axes.autoscale_view(True,True,True)
p1 = plt.barh(ind, power_value_vec, width, color='r')
plt.yticks(ind+width/2., name_value_vec_HP, fontsize=9, fontweight='bold')
f=plt.gcf()
DefaultSize = f.get_size_inches()
f.set_size_inches( (DefaultSize[0]*2, DefaultSize[1]*2) )
plt.xlim((minp-minp*.07),(maxp+maxp*.03))
plt.xlabel('Fuel (galx1000)', fontweight='bold')
plt.title('Fuel Used by Ensemble (Route:'+rname+' Date:'+rdate+' Model:'+rres+'")", fontweight='bold')
# write in the ranking inside each bar to aid in interpretation
artist={}
for rect,bar_color in zip(p1,bar_colors_HP):
  width = (rect.get_width())
  artist[bar_color]=rect
  rect.set_facecolor(bar_color)

  lastDigit = width % 10
  power_vec_str = str(width)# + suffix
  if (width < 5): # The bars aren't wide enough to print the ranking inside
     xloc = width + 1 # Shift the text to the right side of the right edge
     clr = 'black' # Black against white background
     align = 'left'
  else:
     xloc = 0.997*width # Shift the text to the left side of the right edge
     clr = 'white' # White on magenta
```

```
      align = 'right'
   yloc = rect.get_y()+rect.get_height()/2.0 #Center the text vertically in the bar
   plt.text(xloc, yloc, power_vec_str, horizontalalignment=align,
         verticalalignment='center', color=clr, fontsize=9, fontweight='bold')

label=['Great Circle', 'Ensemble Routex', 'Ensemble WEAX', 'Post Processing','Analysis','Model Analysis']

legend([artist[color] for color in 'r','b','g','c','m','teal'], label, loc='best', shadow=True, fancybox=True)

plt.savefig(root3+'HP_Graph.'+rdate+"."+rname+".png", bbox_inches='tight' )

################################################Distance 2D #######
width = 1
plt.figure(2)
fig = plt.figure(2)
axes = fig.add_subplot(111)
axes.set_autoscaley_on(True)
axes.autoscale_view(True,True,True)
p1 = plt.barh(ind, dist_value_vec,   width, color='r')
plt.yticks(ind+width/2., name_value_vec_dist, fontsize=9, weight='bold')
f1=plt.gcf()
DefaultSize = f1.get_size_inches()
f1.set_size_inches( (DefaultSize[0]*2, DefaultSize[1]*2) )
plt.xlim((mind-mind*.07),(maxd+maxd*.03))
plt.xlabel('Distance (nm)', fontweight='bold')
plt.title('Distance by Ensemble (Route:'+rname+' Date:'+rdate+' Model:'+rres+")", fontweight='bold')
artist={}
for rect2, bar_color2 in zip(p1,bar_colors_dist):
   width2 = int(rect2.get_width())
   artist[bar_color2]=rect2
   rect2.set_facecolor(bar_color)
   lastDigit = width2 % 10
   rect2.set_facecolor(bar_color2)
   power_vec_str2 = str(width2)# + suffix
   if (width2 < 5): # The bars aren't wide enough to print the ranking inside
      xloc = width2 + 1 # Shift the text to the right side of the right edge
      clr = 'black' # Black against white background
      align = 'left'
   else:
      xloc = 0.997*width2 # Shift the text to the left side of the right edge
      clr = 'white' # White on magenta
      align = 'right'
   yloc = rect2.get_y()+rect2.get_height()/2.0 #Center the text vertically in the bar
   plt.text(xloc, yloc, power_vec_str2, horizontalalignment=align,
         verticalalignment='center', color=clr, fontsize=9, weight='bold')
label=['Great Circle', 'Ensemble Routex', 'Ensemble WEAX', 'Post Processing','Analysis','Model Analysis']
legend([artist[color] for color in 'r','b','g','c','m','teal'], label, loc='best', shadow=True, fancybox=True)
plt.savefig(root3+'Distance_Graph.'+rdate+"."+rname+".png", bbox_inches='tight')

################################################Time 2D #######

width = 1
plt.figure(3)
fig3 = plt.figure(3)
axes = fig3.add_subplot(111)
```

```
axes.set_autoscaley_on(True)
axes.autoscale_view(True,True,True)

p3 = plt.barh(ind, time_value_vec,   width, color='r')
plt.yticks(ind+width/2., name_value_vec_time, fontsize=9, weight='bold')
f=plt.gcf()
DefaultSize = f.get_size_inches()
f.set_size_inches( (DefaultSize[0]*2, DefaultSize[1]*2) )
plt.xlim((mint-mint*.07),(maxt+maxt*.04))
plt.xlabel('Time (hrs)', fontweight='bold')
plt.title('Time by Ensemble (Route:'+rname+' Date:'+rdate+' Model:'+rres+')", fontweight='bold')
artist={}
for rect3,bar_color3 in zip(p3,bar_colors_time):
  width3 = int(rect3.get_width())
  artist[bar_color3]=rect3
  rect3.set_facecolor(bar_color3)
  lastDigit = width3 % 10
  rect3.set_facecolor(bar_color3)
  power_vec_str3 = str(width3)
  if (width3 < 5): # The bars aren't wide enough to print the ranking inside
     xloc = width3 + 1 # Shift the text to the right side of the right edge
     clr = 'black' # Black against white background
     align = 'left'
  else:
     xloc = 0.990*width3 # Shift the text to the left side of the right edge
     clr = 'white' # White on magenta
     align = 'right'
  yloc = rect3.get_y()+rect3.get_height()/2.0 #Center the text vertically in the bar
  plt.text(xloc, yloc, power_vec_str3, horizontalalignment=align,
        verticalalignment='center', color=clr, fontsize=9, weight='bold')
label=['Great Circle', 'Ensemble Routex', 'Ensemble WEAX', 'Post Processing','Analysis','Model Analysis']
legend([artist[color] for color in 'r','b','g','c','m','teal'], label, loc='best', shadow=True, fancybox=True)
plt.savefig(root3+'Time_Graph.'+rdate+"."+rname+".png", bbox_inches='tight')

####################Fuel Hist##############################

plt.figure(4)
fig4 = plt.figure(4)
axes = fig4.add_subplot(111)
axes.set_autoscalex_on(True)
axes.autoscale_view(True,True,True)
fig, ax = plt.subplots(1)
plt.hist(a,bins=20)
plt.title('Fuel Hist (Route:'+rname+'Date:'+rdate+' Model:'+rres+')\n\n", fontweight='bold')
plt.xlabel("Fuel (galx1000)", fontweight='bold')
plt.ylabel("Frequency", fontweight='bold')
textstr = ('ROUTE STATISTICS:\nmin=%.2f\nmax=%.2f\nmean=%.2f\nvar=%.2f\nstd=%.2f\n\nPOST
PROC. VS GC STATS:\nHybrid Ensemble=%.2f%%\nBias Corr. Ensbl=%.2f%%\nEnsemble.
Avg=%.2f%%\n' %
      (minp,maxp,m, v, s, Hy_avg_alys_Fuel_GC_DIFF, BC_avg_alys_Fuel_GC_DIFF,
EA_avg_alys_Fuel_GC_DIFF))
textstr2 =('Best Member=%.2f%%\nCombined Anlys=%.2f%%\nAnalysis NOGAPS=%.2f%%\nAnalysis
WW3=%.2f%%\nAnalysis NCOM=%.2f%%\n\n' %
```

```
        (MLM_avg_alys_Fuel_GC_DIFF, AN_avg_alys_Fuel_GC_DIFF,
AN_NOGAPS_avg_alys_Fuel_GC_DIFF, AN_WW3_avg_alys_Fuel_GC_DIFF,
AN_NCOM_avg_alys_Fuel_GC_DIFF))
textstr25 =('POST PROC. FCST VS ACT:\nHY_fcst vs HY_anlys=%.2f%%\nBC_fcst vs
BC_anlys=%.2f%%\nEA_fcst vs EA_anlys=%.2f%%' %
        (Hy_avg_alys_Fuel_Pred_DIFF, BC_avg_alys_Fuel_Pred_DIFF, EA_avg_alys_Fuel_Pred_DIFF ))
textstr3 =('\n\nLIMITS EXCEEDED:\nGreat Circle=%i\nHybrid Ensemble=%i\nBias
Corrected=%i\nEnsemble Avg=%i\nAnalysis=%i' %
        (GC_Exceed_Lim_count, HY_Exceed_Lim_count, BC_Exceed_Lim_count,
EA_Exceed_Lim_count, AN_Exceed_Lim_count))

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)
# place a text box in upper left in axes coords
ax.text(1.03, 1.045, textstr+textstr2+textstr25+textstr3, transform=ax.transAxes, fontsize=12,
      verticalalignment='top', bbox=props)
plt.savefig(root3+'HPhist_Graph.'+rdate+"."+rname+".png", bbox_inches='tight',pad_inches=2)

###################Fuel Hist (ROUTEX) ##############################

plt.figure(41)
fig5 = plt.figure(41)
axes = fig5.add_subplot(111)
axes.set_autoscaley_on(True)
axes.autoscale_view(True,True,True)
fig, ax = plt.subplots(1)

plt.hist(power_RX,bins=20) #a
plt.title('Fuel RX Hist (Route:'+rname+' Date:'+rdate+' Model:'+rres+')\n\n", fontweight='bold')
plt.xlabel("Fuel (galx1000)", fontweight='bold')
plt.ylabel("Frequency", fontweight='bold')
plt.xlim((minp-minp*.005),(maxp+maxp*.005))
plt.ylim(0,12)

textstr = ('ROUTE STATISTICS:\nmin=%.2f\nmax=%.2f\nmean=%.2f\nvar=%.2f\nstd=%.2f\n' %
        (minp_RX,maxp_RX,meanp_RX, varp_RX, stdp_RX))

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)
# place a text box in upper left in axes coords
plt.text(1.03, .95, textstr, transform=ax.transAxes, fontsize=12,
      verticalalignment='top', bbox=props)

plt.savefig(root3+'HPhist_Graph_RX.'+rdate+"."+rname+".png", bbox_inches='tight',pad_inches=2)

###################Fuel Hist (WEAX) ##############################

plt.figure(42)
fig, ax = plt.subplots(1)

plt.hist(power_WX,bins=20) #a
plt.title('Fuel WX Hist (Route:'+rname+' Date:'+rdate+' Model:'+rres+')\n\n", fontweight='bold')
plt.xlabel("Fuel (galx1000)", fontweight='bold')
plt.ylabel("Frequency", fontweight='bold')
plt.xlim((minp-minp*.005),(maxp+maxp*.005))
plt.ylim(0,12)
textstr = ('ROUTE STATISTICS:\nmin=%.2f\nmax=%.2f\nmean=%.2f\nvar=%.2f\nstd=%.2f\n' %
```

```
        (minp_WX,maxp_WX,meanp_WX, varp_WX, stdp_WX))

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)
# place a text box in upper left in axes coords
ax.text(1.03, .95, textstr, transform=ax.transAxes, fontsize=12,
      verticalalignment='top', bbox=props)

plt.savefig(root3+'HPhist_Graph_WX.'+rdate+"."+rname+".png", bbox_inches='tight',pad_inches=2)

###################Fuel Hist (WEAX_light) ##############################

plt.figure(71)
fig, ax = plt.subplots(1)
plt.hist(power_WX_light,bins=20) #a
plt.title('Fuel WX_Light Hist (Route:'+rname+' Date:'+rdate+' Model:'+rres+")\n\n", fontweight='bold')
plt.xlabel("Fuel (galx1000)", fontweight='bold')
plt.ylabel("Frequency", fontweight='bold')
plt.xlim((minp-minp*.005),(maxp+maxp*.005))
plt.ylim(0,12)
textstr = ('ROUTE STATISTICS:\nmin=%.2f\nmax=%.2f\nmean=%.2f\nvar=%.2f\nstd=%.2f\n' %
        (minp_WX_light,maxp_WX_light,meanp_WX_light, varp_WX_light, stdp_WX_light))
props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)

# place a text box in upper left in axes coords
ax.text(1.03, .95, textstr, transform=ax.transAxes, fontsize=12,
      verticalalignment='top', bbox=props)
plt.savefig(root3+'HPhist_Graph_WX_light.'+rdate+"."+rname+".png", bbox_inches='tight',pad_inches=2)

###################Fuel Hist (ROUTEX/WX) ##############################

plt.figure(45)
fig6 = plt.figure(45)
axes = fig6.add_subplot(111)
axes.set_autoscalex_on(True)
#axes.set_autoscaley_on(True)
axes.autoscale_view(True,True,True)
plt.subplots_adjust(hspace=.5)
plt.subplot(211)
plt.title('Fuel RX vs WX Hist (Route:'+rname+' Date:'+rdate+' Model:'+rres+")\n\n", fontweight='bold')
plt.hist(power_RX,bins=20) #a
plt.xlabel("Fuel (galx1000)", fontweight='bold')
plt.ylabel("Frequency", fontweight='bold')
plt.xlim((minp-minp*.005),(maxp+maxp*.005))
plt.ylim(0,12)
textstr = ('ROUTE STATISTICS:\nmin=%.2f\nmax=%.2f\nmean=%.2f\nvar=%.2f\nstd=%.2f\n' %
        (minp_RX,maxp_RX,meanp_RX, varp_RX, stdp_RX))
props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)
# place a text box in upper left in axes coords
plt.text(1.03, .95, textstr, fontsize=12,
      verticalalignment='top', bbox=props)
plt.subplot(212)
plt.hist(power_WX,bins=20) #a
plt.xlabel("Fuel (galx1000)", fontweight='bold')
plt.ylabel("Frequency", fontweight='bold')
plt.xlim((minp-minp*.005),(maxp+maxp*.005))
```

```
plt.ylim(0,12)
textstr = ('ROUTE STATISTICS:\nmin=%.2f\nmax=%.2f\nmean=%.2f\nvar=%.2f\nstd=%.2f\n' %
        (minp_WX,maxp_WX,meanp_WX, varp_WX, stdp_WX))
props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)
# place a text box in upper left in axes coords
plt.text(1.03, .95, textstr,  fontsize=12,
      verticalalignment='top', bbox=props)

plt.savefig(root3+'HPhist_Graph_RX_WX.'+rdate+"."+rname+".png", bbox_inches='tight',pad_inches=2)

##################################################Distance Hist########
plt.figure(5)
fig, ax = plt.subplots(1)
plt.hist(b,bins=20)
plt.title('Dist Hist (Route:'+rname+' Date:'+rdate+' Model:'+rres+")\n", fontweight='bold')
plt.xlabel("Distance (nm)", fontweight='bold')
plt.ylabel("Frequency", fontweight='bold')
textstr = ('ROUTE STATISTICS:\nmin=%.2f\nmax=%.2f\nmean=%.2f\nvar=%.2f\nstd=%.2f\n\nPOST
PROC. VS GC STATS:\nHybrid Ensemble=%.2f%%\nBias Corr. Ensbl=%.2f%%\nEnsemble.
Avg=%.2f%%\n' %
        (mind,maxd,md, vd, sd, Hy_avg_alys_Dist_GC_DIFF, BC_avg_alys_Dist_GC_DIFF,
EA_avg_alys_Dist_GC_DIFF))
textstr2 =('Best Member=%.2f%%\nCombined Anlys=%.2f%%\nAnalysis NOGAPS=%.2f%%\nAnalysis
WW3=%.2f%%\nAnalysis NCOM=%.2f%%\n\n' %
        (MLM_avg_alys_Dist_GC_DIFF, AN_avg_alys_Dist_GC_DIFF,
AN_NOGAPS_avg_alys_Dist_GC_DIFF, AN_WW3_avg_alys_Dist_GC_DIFF,
AN_NCOM_avg_alys_Dist_GC_DIFF))
textstr25 =('POST PROC. FCST VS ACT:\nHY_fcst vs HY_anlys=%.2f%%\nBC_fcst vs
BC_anlys=%.2f%%\nEA_fcst vs EA_anlys=%.2f%%' %
        (Hy_avg_alys_Dist_Pred_DIFF, BC_avg_alys_Dist_Pred_DIFF, EA_avg_alys_Dist_Pred_DIFF ))
props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)
ax.text(1.03, .95, textstr+textstr2+textstr25, transform=ax.transAxes, fontsize=12,
      verticalalignment='top', bbox=props)
plt.savefig(root3+'Disthist_Graph.'+rdate+"."+rname+".png", bbox_inches='tight',pad_inches=2)


##################################################Distance Hist RX########
plt.figure(41)
fig, ax = plt.subplots(1)
plt.hist(dist_RX,bins=20) #a
plt.title('Dist RX Hist (Route:'+rname+' Date:'+rdate+' Model:'+rres+")\n\n", fontweight='bold')
plt.xlabel("Distance (nm)", fontweight='bold')
plt.ylabel("Frequency", fontweight='bold')
plt.xlim((mind-mind*.005),(maxd+maxd*.005))
plt.ylim(0,12)
textstr = ('ROUTE STATISTICS:\nmin=%.2f\nmax=%.2f\nmean=%.2f\nvar=%.2f\nstd=%.2f\n' %
        (mind_RX,maxd_RX,meand_RX, vard_RX, stdd_RX))
props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)
# place a text box in upper left in axes coords
plt.text(1.03, .95, textstr, transform=ax.transAxes, fontsize=12,
      verticalalignment='top', bbox=props)

plt.savefig(root3+'Disthist_Graph_RX.'+rdate+"."+rname+".png", bbox_inches='tight',pad_inches=2)

###################Dist Hist (WEAX) ##############################
```

```
plt.figure(42)
fig, ax = plt.subplots(1)
plt.hist(dist_WX,bins=20) #a
plt.title('Dist WX Hist (Route:'+rname+' Date:'+rdate+' Model:'+rres+')\n\n", fontweight='bold')
plt.xlabel("Distance (nm)", fontweight='bold')
plt.ylabel("Frequency", fontweight='bold')
plt.xlim((mind-mind*.005),(maxd+maxd*.005))
plt.ylim(0,12)
textstr = ('ROUTE STATISTICS:\nmin=%.2f\nmax=%.2f\nmean=%.2f\nvar=%.2f\nstd=%.2f\n' %
        (mind_WX,maxd_WX,meand_WX, vard_WX, stdd_WX))
props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)
# place a text box in upper left in axes coords
ax.text(1.03, .95, textstr, transform=ax.transAxes, fontsize=12,
      verticalalignment='top', bbox=props)
plt.savefig(root3+'Disthist_Graph_WX.'+rdate+"."+rname+".png", bbox_inches='tight',pad_inches=2)

####################Dist Hist (WEAX_light) ##############################

plt.figure(42)
fig, ax = plt.subplots(1)
plt.hist(dist_WX_light,bins=20) #a
plt.title('Dist WX_Light Hist (Route:'+rname+' Date:'+rdate+' Model:'+rres+')\n\n", fontweight='bold')
plt.xlabel("Distance (nm)", fontweight='bold')
plt.ylabel("Frequency", fontweight='bold')
plt.xlim((mind-mind*.005),(maxd+maxd*.005))
plt.ylim(0,12)

textstr = ('ROUTE STATISTICS:\nmin=%.2f\nmax=%.2f\nmean=%.2f\nvar=%.2f\nstd=%.2f\n' %
        (mind_WX_light,maxd_WX_light,meand_WX_light, vard_WX_light, stdd_WX_light))

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)
# place a text box in upper left in axes coords
ax.text(1.03, .95, textstr, transform=ax.transAxes, fontsize=12,
      verticalalignment='top', bbox=props)
plt.savefig(root3+'Disthist_Graph_WX_light.'+rdate+"."+rname+".png",
bbox_inches='tight',pad_inches=2)

####################Dist Hist (ROUTEX/WX) ##############################

plt.figure(41)
plt.subplots_adjust(hspace=.5)
plt.subplot(211)
plt.title('Dist RX vs WX Hist (Route:'+rname+' Date:'+rdate+' Model:'+rres+')\n\n", fontweight='bold')
plt.hist(dist_RX,bins=20, color='b') #a
plt.xlabel("Distance (nm)", fontweight='bold')
plt.ylabel("Frequency", fontweight='bold')
plt.xlim((mind-mind*.005),(maxd+maxd*.005))
plt.ylim(0,12)
textstr = ('ROUTE STATISTICS:\nmin=%.2f\nmax=%.2f\nmean=%.2f\nvar=%.2f\nstd=%.2f\n' %
        (mind_RX,maxd_RX,meand_RX, vard_RX, stdd_RX))
props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)
# place a text box in upper left in axes coords
plt.text(1.03, .95, textstr, fontsize=12,
      verticalalignment='top', bbox=props)
```

```
plt.subplot(212)
plt.hist(dist_WX,bins=20, color='b') #a
plt.xlabel("Distance (nm)", fontweight='bold')
plt.ylabel("Frequency", fontweight='bold')
plt.xlim((mind-mind*.005),(maxd+maxd*.005))
plt.ylim(0,12)
textstr = ('ROUTE STATISTICS:\nmin=%.2f\nmax=%.2f\nmean=%.2f\nvar=%.2f\nstd=%.2f\n' %
        (mind_WX,maxd_WX,meand_WX, vard_WX, stdd_WX))
props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)
# place a text box in upper left in axes coords
plt.text(1.03, .95, textstr,  fontsize=12,
      verticalalignment='top', bbox=props)
plt.savefig(root3+'Disthist_Graph_RX_WX.'+rdate+"."+rname+".png", bbox_inches='tight',pad_inches=2)

################################################Time Hist#######
plt.figure(6)
fig, ax = plt.subplots(1)
plt.hist(t,bins=40)
plt.title('Time Hist (Route:'+rname+' Date:'+rdate+' Model:'+rres+")\n", fontweight='bold')
plt.xlabel("Time (hrs)", fontweight='bold')
plt.ylabel("Frequency", fontweight='bold')

textstr = ('ROUTE STATISTICS:\nmin=%.2f\nmax=%.2f\nmean=%.2f\nvar=%.2f\nstd=%.2f\n\nPOST
PROC. VS GC STATS:\nHybrid Ensemble=%.2f%%\nBias Corr. Ensbl=%.2f%%\nEnsemble.
Avg=%.2f%%\n' %
        (mint,maxt,mt, vt, st, Hy_avg_alys_Time_GC_DIFF, BC_avg_alys_Time_GC_DIFF,
EA_avg_alys_Time_GC_DIFF))
textstr2 =('Best Member=%.2f%%\nCombined Anlys=%.2f%%\nAnalysis NOGAPS=%.2f%%\nAnalysis
WW3=%.2f%%\nAnalysis NCOM=%.2f%%\n\n' %
        (MLM_avg_alys_Time_GC_DIFF, AN_avg_alys_Time_GC_DIFF,
AN_NOGAPS_avg_alys_Time_GC_DIFF, AN_WW3_avg_alys_Time_GC_DIFF,
AN_NCOM_avg_alys_Time_GC_DIFF))
textstr25 =('POST PROC. FCST VS ACT:\nHY_fcst vs HY_anlys=%.2f%%\nBC_fcst vs
BC_anlys=%.2f%%\nEA_fcst vs EA_anlys=%.2f%%' %
        (Hy_avg_alys_Time_Pred_DIFF, BC_avg_alys_Time_Pred_DIFF, EA_avg_alys_Time_Pred_DIFF
))

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)
ax.text(1.03, .95, textstr+textstr2+textstr25, transform=ax.transAxes, fontsize=12,
      verticalalignment='top', bbox=props)
plt.savefig(root3+'Timehist_Graph.'+rdate+"."+rname+".png", bbox_inches='tight', pad_inches=2)

###################Time Hist (ROUTEX) ##############################

plt.figure(7)
fig, ax = plt.subplots(1)
hist,bins=np.histogram(time_RX,bins=40)
width=0.7*(bins[1]-bins[0])
center=(bins[:-1]+bins[1:])/2
plt.bar(center,hist,align='center',width=width+.05)
plt.title('Time RX Hist (Route:'+rname+' Date:'+rdate+' Model:'+rres+")\n\n", fontweight='bold')
plt.xlabel("Time (hrs)", fontweight='bold')
plt.ylabel("Frequency", fontweight='bold')
plt.xlim((mint_RX-mint_RX*.01),(maxt_RX+maxt_RX*.01))
textstr = ('ROUTE STATISTICS:\nmin=%.2f\nmax=%.2f\nmean=%.2f\nvar=%.2f\nstd=%.2f\n' %
```

```
        (mint_RX,maxt_RX,meant_RX, vart_RX, stdt_RX))
props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)
# place a text box in upper left in axes coords
plt.text(1.03, .95, textstr, transform=ax.transAxes, fontsize=12,
      verticalalignment='top', bbox=props)
plt.savefig(root3+'Timehist_Graph_RX.'+rdate+"."+rname+".png", bbox_inches='tight',pad_inches=2)

###################Time Hist (WEAX) ##############################

plt.figure(8)
fig, ax = plt.subplots(1)
hist,bins=np.histogram(time_WX,bins=40)
width=0.7*(bins[1]-bins[0])
center=(bins[:-1]+bins[1:])/2
plt.bar(center,hist,align='center',width=width+.05)
plt.title('Time WX Hist (Route:'+rname+' Date:'+rdate+' Model:'+rres+")\n\n", fontweight='bold')
plt.xlabel("Time (hrs)", fontweight='bold')
plt.ylabel("Frequency", fontweight='bold')
plt.xlim((mint_WX-mint_WX*.01),(maxt_WX+maxt_WX*.01))
textstr = ('ROUTE STATISTICS:\nmin=%.2f\nmax=%.2f\nmean=%.2f\nvar=%.2f\nstd=%.2f\n' %
        (mint_WX,maxt_WX,meant_WX, vart_WX, stdt_WX))

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)
# place a text box in upper left in axes coords
ax.text(1.03, .95, textstr, transform=ax.transAxes, fontsize=12,
      verticalalignment='top', bbox=props)
plt.savefig(root3+'Timehist_Graph_WX.'+rdate+"."+rname+".png", bbox_inches='tight',pad_inches=2)

###################Time Hist (WEAX_light) ##############################

plt.figure(8)
fig, ax = plt.subplots(1)
hist,bins=np.histogram(time_WX_light,bins=40)
width=0.7*(bins[1]-bins[0])
center=(bins[:-1]+bins[1:])/2
plt.bar(center,hist,align='center',width=width+.05)
plt.title('Time WX_light Hist (Route:'+rname+' Date:'+rdate+' Model:'+rres+")\n\n", fontweight='bold')
plt.xlabel("Time (hrs)", fontweight='bold')
plt.ylabel("Frequency", fontweight='bold')
plt.xlim((mint_WX_light-mint_WX_light*.01),(maxt_WX_light+maxt_WX_light*.01))
textstr = ('ROUTE STATISTICS:\nmin=%.2f\nmax=%.2f\nmean=%.2f\nvar=%.2f\nstd=%.2f\n' %
        (mint_WX_light,maxt_WX_light,meant_WX_light, vart_WX_light, stdt_WX_light))
props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)
# place a text box in upper left in axes coords
ax.text(1.03, .95, textstr, transform=ax.transAxes, fontsize=12,
      verticalalignment='top', bbox=props)
plt.savefig(root3+'Timehist_Graph_WX_light.'+rdate+"."+rname+".png",
bbox_inches='tight',pad_inches=2)

###################Time Hist (ROUTEX/WX) ##############################

plt.figure(9)
plt.subplots_adjust(hspace=.5)
if mint_RX < mint_WX:
   mintComb=mint_RX
```

```
else:
   mintComb=mint_WX
if maxt_RX > maxt_WX:
   maxtComb=maxt_RX
else:
   maxtComb=maxt_WX

plt.subplot(211)
plt.title('Time RX vs WX Hist (Route:'+rname+' Date:'+rdate+' Model:'+rres+')\n\n", fontweight='bold')
hist,bins=np.histogram(time_RX,bins=40)
width=0.7*(bins[1]-bins[0])
center=(bins[:-1]+bins[1:])/2
plt.bar(center,hist,align='center',width=width+.2)
plt.xlabel("Time (hrs)", fontweight='bold')
plt.ylabel("Frequency", fontweight='bold')
plt.xlim((mintComb-mintComb*.005),(maxtComb+maxtComb*.005))
plt.ylim(0,40)
props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)

# place a text box in upper left in axes coords
plt.text(1.03, .95, textstr, fontsize=12,
      verticalalignment='top', bbox=props)

plt.subplot(212)

hist,bins=np.histogram(time_WX,bins=40)
width=0.7*(bins[1]-bins[0])
center=(bins[:-1]+bins[1:])/2
plt.bar(center,hist,align='center',width=width+.05)
plt.xlabel("Time (hrs)", fontweight='bold')
plt.ylabel("Frequency", fontweight='bold')
plt.xlim((mintComb-mintComb*.005),(maxtComb+maxtComb*.005))
plt.ylim(0,40)

# these are matplotlib.patch.Patch properies
props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)

# place a text box in upper left in axes coords
plt.text(1.03, .95, textstr,  fontsize=12,
      verticalalignment='top', bbox=props)

plt.savefig(root3+'Timehist_Graph_RX_WX.'+rdate+"."+rname+".png",
bbox_inches='tight',pad_inches=2)
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Arthur, Michael, and Chris Marone. 2011. Gyres and Surface Currents. Essentials of Oceanography, https://www.e-education.psu.edu/earth540/content/c4_p5.html.

Barron, Charlie N., A. Birol Kara, Harley E. Hurlburt, C. Rowley, and Lucy F. Smedstad. 2004. Sea Surface Height Predictions from the Global Navy Coastal Ocean Model (NCOM) during 1998-2001. *Journal of Atmospheric and Oceanic Technology* 21, no. 12 (December): 1876–94.

Barron, Charlie N., A. Birol Kara, Paul. J. Martin, Robert. C. Rhodes, and Lucy. F. Smedstad. 2006. Formulation, Implementation and Examination of Vertical Coordinate Choices in the Global Navy Coastal Ocean Model (NCOM). *Ocean Modeling* 11: 347–75.

Brown, Gerald G., and W. Matthew Carlyle. 2008. Optimizing the US Navy's Combat Logistics Force. Wiley InterScience - Naval Research Logistics 55: 800-10, doi: 10.1002/nav.20318.

Brown, Gerald G., Jeffrey E. Kline, Richard E. Rosenthal, and Alan R. Washburn. 2007. Steaming on Convex Hulls. *Interfaces* 37, no. 4 (July): 342–52.

Chen, Henry. 1978. A Dynamic Program for Minimum Cost Ship Routing Under Uncertainty. PhD diss., Massachusetts Institute of Technology.

Chen, Henry, Vincent Cardone, and Peter Lacey. 1998. Use of Operation Support Information Technology to Increase Ship Safety and Efficiency. *The Society of Naval Architects and Marine Engineers* 106: 105–27.

Commander Naval Meteorology and Oceanography Command (CNMOC). 2011. *U.S. Navy Meteorological and Oceanographic Support Manual*, COMNAVMETOCCOMINST 3140.1M.

Department of the Navy (DON). 1997. *Operational Reports*, NWP 1-03.1.

DRS. 2012. *Smart Voyage Planning Decision Aid (SVPDA) Phase I Technical Discussions*. Technical presentation.

Fleet Numerical Meteorology and Oceanography Center (FNMOC). 2008. *Ship Route Subsystem*, Standard Operating Procedure.

Fox, Daniel. N., Charlie N. Barron, W. J. Teague, Michael R. Carnes, and Craig.M. Lee. 2002. The Modular Ocean Data Analysis System (MODAS). *Journal of Atmospheric and Oceanic Technology* 19: 240–52.

Hanley, Kirsty. E., Stephen Belcher, and Peter Sullivan. 2010. A Global Climatology of Wind–Wave Interaction. *American Meteorological Society* 40, no. 6 (June): 1263–82.

Itri. 2010. *Probabilistic Mission Impact Information Exploitation Ensemble Forecast Application System (EFAS),* by James F. Etro, Arthur Etro, James Hanson, and John Cook. Technical report.

HYCOM Organization. 2012. *Impacts of Ocean Currents and Waves on the Wind Stress Drag Coefficient: Relevance to HYCOM,* by Kara Birol, Joe Metzger, and Mark Bourassa. Technical report.

Komen, Gerbrand J., Luigi Cavaleri, Mark Donelan, Klaus Hasselmann, and Peter Janssen. 1996. *Dynamics and Modelling of Ocean Waves*. Cambridge, United Kingdom: Cambridge University Press.

Montes, Anel. 2005. Network Shortest Path Application for Optimum Track Ship Routing. Master's thesis, Naval Postgraduate School.

National Imagery and Mapping Agency (NIMA). 2002. *The American Practical Navigator.* Bethesda, MD: Publication no. 9.

National Oceanic and Atmospheric Administration (NOAA). 2009. *User Manual and System Documentation of WAVEWATCH III Version 3.14,* by Hendrik L. Tolman. Technical report no. 276.

National Oceanic and Atmospheric Administration (NOAA). 2010. WAVEWATCH III Model. Environmental Modeling Center. (June 10). http://polar.ncep.noaa.gov/waves/wavewatch/wavewatch.shtml.

National Oceanic and Atmospheric Administration (NOAA). 2011. NCOM Model Ocean Currents Areas." Ocean Prediction Center. (November 22). http://www.opc.ncep.noaa.gov/newNCOM/NCOM_currents.shtml.

National Weather Service (NWS). 1995. *On the selection of propagation schemes for a spectral wind wave model,* by Hendrik L. Tolman. Office note no. 411.

Naval Research Laboratory (NRL). 2006. First Global Ocean Model Developed at NRL Stennis. Naval Research Laboratory, Washington D.C. 2006. http://www.nrl.navy.mil/media/news-releases/2006/first-global-ocean-model-developed-at-nrl-stennis.

Naval Research Laboratory, Monterey (NRLMRY). 2003. NOGAPS History. Naval Research Laboratory, Monterey, CA. (August 6). http://www.nrlmry.navy.mil/nogaps_his.htm.

Naval Research Lab, Stennis Space Center. 2000. *Description of the Navy Coastal Ocean Model Version 1.0.*, by Paul J. Martin. Technical report no FR/7322-00-9962.

Office of Naval Research (ONR). 2011. *METOC System Improvemetns for a Smart Voyage Planning Decision Aid*, Primary POM-14-03.

Rhodes, Robert C., Harley E. Hurlburt, Alan J. Wallcraft, Charlie N. Barron, Paul J. Martin, E. Joseph Metzger, Jay F. Shriver, Dong Ko, O. Martin Smedstad, Scott L. Cross, and A. Birol Kara. 2002. Navy Real-time Global Modeling Systems. *Oceanography* 15, no. 1: 29–43.

Saltelli, Andrea, Marco Ratto, Francesca Campolongo, Jessica Cariboni, Terry Andres, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. 2008. *Global Sensitivity Analysis*. West Sussex, United Kingdom: John Wiley & Sons.

Scripter, Sam. 2012. Global Pressure Patterns and General Atmospheric Circulation. Physical Geography. http://www.sci.uidaho.edu/scripter/geog100/lect/04-atmos-oceanic-circ/ch4-part-5-high-low-areas-wind-belts.htm.

Tolman, Hendrik L. 1989. The Numerical Model WAVEWATCH: A Third Generation Model for the Hindcasting of Wind Waves on Tides in Shelf Seas. *Communications on Hydraulic and Geotechnical Engineering* 89, no. 2: 72.

Tolman, Hendrik L., and Nico Booij. 1998. Modeling Wind Waves using Wavenumber-Direction Spectra and a Variable Wavenumber Grid. *The Global Atmosphere and Ocean System*, no. 6: 295–309.

Tolman, Hendrik L. 1992. Effects of Numerics on the Physics in a Third-Generation Wind-Wave Model. *Journal of Physical Oceanography* 22: 1095–1111.

University Corporation for Atmospheric Research (UCAR). 2012. Introduction to Ocean Currents. COMET Website. (August 4). http://www.meted.ucar.edu/oceans/currents/print.htm.

University Corporation for Atmospheric Research (UCAR). 2012. Numerical Modeling (NWP) NOGAPS 4.0. COMET Website. (August 1). http://www.meted.ucar.edu/nwp/pcu2/nogaps/index.htm.

University Corporation for Atmospheric Research (UCAR). 2012. WW3 Marine Wave Model. COMET Website. (August 2). http://www.meted.ucar.edu/nwp/pcu2/marine_matrix/WW3_regional_eqns.htm.

Wave Model Development and Implementation Group (WAMDIG). 1988. The WAM model - A third generation ocean wave prediction model. *Journal of Physical Oceanography* 18: 1775–1810.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. RADM Jonathan White, USN
   Oceanographer and Navigator of the Navy
   Washington, DC

4. RADM Brian Brown
   Commander, Naval Meteorology and Oceanography Command
   Stennis Space Center, Mississippi

5. Mr. Robert Winokur
   Technical Director
   Oceanographer & Navigator of the Navy
   Washington DC

6. Dr. William H. Burnett
   Technical Director
   Commander, Naval Meteorology and Oceanography Command
   Stennis Space Center, Mississippi

7. CAPT Greg Ulses
   Director USW
   Stennis Space Center, Mississippi

8. CAPT Paul Oosterling
   Commander, Naval Oceanographic Office
   Stennis Space Center, Mississippi

9. Mr. Thomas Cuff
   Technical Director
   Naval Oceanographic Office
   Stennis Space Center, Mississippi

10. Dr. Richard Jeffries
    Commander, Naval Meteorology and Oceanography Command
    Stennis Space Center, Mississippi

11. NOMWAC CD

12. NOAC CD

13. Professor Mary L. Batteen
Naval Postgraduate School
Monterey, California

14. Professor Jeffrey Paduan
Naval Postgraduate School
Monterey, California

15. Professor Peter Chu
Naval Postgraduate School
Monterey, California

16. Ronald E. Betsch
MIW Program Manager
Stennis Space Center, Mississippi

17. Dr. James Rigney
Naval Oceanographic Office
Stennis Space Center, Mississippi

18. Dr. Jim Hansen
Naval Research Laboratory
Monterey, California

19. LCDR Scott E. Miller, USN
COMCARSTRKGRU NINE
San Diego, CA