|  |
| --- |
| Simulator Module |
| SAROPS Version 2.1 |
|  |
| This document describes the basic approach and algorithms of the Simulator Module in SAROPS 2.1. |
| This document contains data that is delivered to the Government with Unlimited Rights in accordance with contract HSCG44-16-F-001007 and FAR 52.227-17. |
| Thomas M Kratzke, Strategic Data Systems, 22 May 2017 |

Table of Contents

[I. Introduction 0](#_Toc483201232)

[A. Non-Sim and Non-Planner Modules of SAROPS Sim 0](#_Toc483201233)

[B. Overview of Sim 1](#_Toc483201234)

[C. Sim Input Files 1](#_Toc483201235)

[1. Sim.xml 1](#_Toc483201236)

[2. Sim.properties 2](#_Toc483201237)

[3. Environmental Files 2](#_Toc483201238)

[D. Overview of the Document 2](#_Toc483201239)

[II. Scenarios 3](#_Toc483201240)

[A. LKP and Area Scenarios 3](#_Toc483201241)

[B. LOB Scenanrio 4](#_Toc483201242)

[C. Flare Scenario 6](#_Toc483201243)

[D. Voyage 7](#_Toc483201244)

[E. LKP Plus Dead-Reckon 10](#_Toc483201245)

[III. Hazards and Distress Incident 10](#_Toc483201246)

[IV. Reverse Drift Scenario 12](#_Toc483201247)

[V. Winds and Currents 13](#_Toc483201248)

[A. Leeway Data and Using the Environment to Move Particles 13](#_Toc483201249)

[B. CorrelationE 14](#_Toc483201250)

[C. Ocean Interpolation 14](#_Toc483201251)

[D. Rivers Interpolation 14](#_Toc483201252)

[VI. Land Processing 14](#_Toc483201253)

[A. GeoMtx 14](#_Toc483201254)

[B. Slippery Shore, Stop-and-Block 14](#_Toc483201255)

[C. Rivers 14](#_Toc483201256)

[VII. Bathymetry and Anchoring 14](#_Toc483201257)

[VIII. Completed Searches 14](#_Toc483201258)

[A. POS and SRUs 15](#_Toc483201259)

[B. Ladder Patterns, Perfect Patterns, and Terminology 15](#_Toc483201260)

[C. Optimization Problem’s Constraints 16](#_Toc483201261)

[D. SruSolution and SruSolutionArray 17](#_Toc483201262)

[E. Non-Linear Programming and Parameterization of Boxes 18](#_Toc483201263)

[F. Frozen and Floating SRUs 19](#_Toc483201264)

[G. Summary 20](#_Toc483201265)

[IX. Problems with Planner 2.0.3 20](#_Toc483201266)

[A. Overlap Problem when Placing SRUs 21](#_Toc483201267)

[X. Grabbing a Rock 21](#_Toc483201268)

[A. Small Set of Particles and Large Number of Rocks 21](#_Toc483201269)

[B. Initial Placement of 1 SRU on a Probability Distribution 22](#_Toc483201270)

[XI. Bird’s Nest Algorithm 22](#_Toc483201271)

[XII. Putting an SRU on Mars 24](#_Toc483201272)

[XIII. Measuring Overlap 24](#_Toc483201273)

[A. Overlap Computation Algorithm Sketch 25](#_Toc483201274)

[XIV. Polishing a Rock, Different Stages 27](#_Toc483201275)

[A. Preliminary Stage: Increase POS 28](#_Toc483201276)

[B. Clear Shape Constraint Violations 28](#_Toc483201277)

[C. Clear Ovl Constraint Violations 28](#_Toc483201278)

[D. Improve POS: Zero In 28](#_Toc483201279)

[XV. Polishing a Rock, Clearing Constraints by Clearing “Worst” Constraint 28](#_Toc483201280)

[XVI. Different Approximations to POS Used During Optimization 30](#_Toc483201281)

[XVII. How SAROPS’ Uses Planner, and LatestSeed 31](#_Toc483201282)

[XVIII. Planner 2.2 33](#_Toc483201283)

[A. ESS, ESS+NVG, Asymmetric Sensors, and the Use of 33](#_Toc483201284)

[B. Parameterize a Solution without a Box 33](#_Toc483201285)

[Appendix A: Polishing a Rock; the List of Minor Moves 0](#_Toc483201286)

[Appendix B: XML 5](#_Toc483201287)

[A. Root Element 5](#_Toc483201288)

[B. SRU Elements 5](#_Toc483201289)

[1. Non-Pattern SRUs 6](#_Toc483201290)

[2. Pattern SRUs 6](#_Toc483201291)

[3. Lateral Range Curve 7](#_Toc483201292)

[REFERENCES 0](#_Toc483201293)

List of Tables

[Table 1 Main Components of Sim.xml 2](#_Toc483201294)

[Table 2: Data of Bearing Call 4](#_Toc483201295)

[Table 3: Requirements for an LOB Fix 5](#_Toc483201296)

[Table 4: Leeway Data Structure 13](#_Toc483201297)

[Table 4: Leeway Data Structure 13](#_Toc483201298)

[Table 5 Parameterization of Boxes by other Modules of SAROPS 19](#_Toc483201299)

[Table 6 Internal Parameterization of TS-Boxes by Planner 19](#_Toc483201300)

[Table 7 Problematic Situation when Grabbing a Rock 20](#_Toc483201301)

[Table 8 Questions Combined to Produce Planner 2.1 “Grab a Rock” Approach 23](#_Toc483201302)

[Table 9. Different PosFunctions 31](#_Toc483201303)

[Table 10 Information of a Planner Problem 32](#_Toc483201304)

[Table 11 Steps of a Planner Run 32](#_Toc483201305)

[Table 12 Uses of Planner 32](#_Toc483201306)

[Table 13 Parameterization of a Pattern in Planner 2.2 34](#_Toc483201307)

[Table 14 Root (<PATH>) Element of Planner XML 5](#_Toc483201308)

List of Figures

[Figure 1: Non convex Polygon 3](#_Toc483201309)

[Figure 2: Area of an LOB 5](#_Toc483201310)

[Figure 3: Vertex from Snowcone and Uncertainties 6](#_Toc483201311)

[Figure 4: "Correlation” between Dwell Zones 7](#_Toc483201312)

[Figure 5: Connecting Centers of Mass, and Perfect Correlation 9](#_Toc483201313)

[Figure 6: Perfect Anti-Correlation and No Correlation 9](#_Toc483201314)

[Figure 7: Correlation = 0.75, used in Sim 10](#_Toc483201315)

[Figure 8: Intervals of a Hazard for an Example Particle 11](#_Toc483201316)

[Figure 9 Two Types of Ladder Pattern 16](#_Toc483201317)

[Figure 10 Optimization Problem Statement in Terms of Boxes 17](#_Toc483201318)

[Figure 11 Buffer around each leg, and Convex hull around all buffers 18](#_Toc483201319)

[Figure 12 Red Cross pair of Boxes 25](#_Toc483201320)

[Figure 13 Separating Edge 26](#_Toc483201321)

[Figure 14 No Separating Edge, and the Edge that is Closest to being a Separating Edge 26](#_Toc483201322)

[Figure 15 Reducing can be Misleading; 27](#_Toc483201323)

[Figure 16 Tangled Boxes 29](#_Toc483201324)

[Figure 17. Starting position, with “hand” at starting point of path 0](#_Toc483201325)

[Figure 18. MoveAlong and MoveNegAlong 0](#_Toc483201326)

[Figure 19. MoveAcross and MoveNegAcross 1](#_Toc483201327)

[Figure 20. IncCcwTwist and DecCcwTwist 1](#_Toc483201328)

[Figure 21. Flip and Rotate180 1](#_Toc483201329)

[Figure 22. IncAlong1 and IncAlong2 2](#_Toc483201330)

[Figure 23. IncAcross1 and IncAcross2 2](#_Toc483201331)

[Figure 24 Expand and Contract 2](#_Toc483201332)

[Figure 25. DecAlong1 and DecAlong2 3](#_Toc483201333)

[Figure 26. DecAcross1 and DecAcross2 3](#_Toc483201334)

[Figure 27. Rotate90Ccw and Rotate90Cw 3](#_Toc483201335)

Simulator Module

SAROPS Version 2.1

# Introduction

There are four major components of the SAROPS Sim module. Two of them are highly visible; the Simulator which produces a dynamic probability distribution, and the Planner that uses this probability distribution to assign boxes to SRUs. A description of Planner is included in the document (??). In this current document, we will discuss Simulator. We will refer to the Simulator module of SAROPS 2.1 as “Simulator 2.1” or simply “Simulator,” or even simply “Sim.” Similarly, we will refer to the Simulator module of SAROPS 2.0.3 as Simulator 2.0.3, and the Simulator module of SAROPS in general as simply Simulator or Sim.

The two other components of the SAROPS Sim module are the SimWebServer and the BuildSimLand modules. Beyond this introductory section, we will not discuss these, but here we give a very brief overview of these two code modules.

## Non-Sim and Non-Planner Modules of SAROPS Sim

SimWebServer is a class in the package com.tmk.sarops.control.simWebServer. It has a “main” routine which is the entry point for running the program SimWebServer. This main simply creates a single SimWebServer object, which creates a org.eclipse.jetty.server.Server and a SimCaseManager. This latter object is used to collect and run Sim and Planner cases in parallel. A SimCaseManager does not deal with handling http requests or port numbers or any of the other intricacies of being a web server; it simply queues up the Sim and Planner requests, and assigns them to its own set of “engines” as these engines become available. The commands that a SimWebServer’s Server handles are “Run a Sim or Planner Case with the following xml file,” and “Give the status of a particular case or all of the cases.”

BuildSimLand is a stand-alone program that builds the land files from a collection of input files. In 2.1, this run takes about 90 minutes, which is down from 36 hours. If additional corrections are needed (e.g. missing islands or errant coast lines), files are provided, and BuildSimLand will be rerun to produce new data files.

## Overview of Sim

Sim is the module that produces the paths of the particles. When a Sim run is completed, it leaves behind a file of particles, called the “Particle File,” that can be grouped together in cells and these cells are displayed as a probability distribution. The Planner, a separate program, reads the “Particle File” and computes rectangles for Search and Rescue Units (SRUs) based on this file. Hence, there is no direct communication between Sim and Planner. Indeed, once Sim runs, the Particle File is read for every Planner run in this case. That includes the “GetInitial,” the “Optimize,” and the “Eval” runs; these are all Planner runs, and each of these starts by reading the same Particle File that Sim produced.

## Sim Input Files

### Sim.xml

The main input to a Sim run is an xml that we call the “Sim.xml file” or simply “Sim.xml.” The key data within Sim.xml are a set of “Scenarios” and a set of “Object Types.” Each scenario is of a “scenario type,” and provides distributions of particles before or at distress. For example, the simplest scenario type is an “LKP” (last known position) scenario, which means that we have a central lat/lng and time of the distress incident, and there is no pre-distress motion. This is modelled by a circular bivariate normal in position and a truncated normal in time. Other scenario types include “LKP+Dead-Reckon,” flare sighting, Line-of-bearing, and voyages. There is pre-distress motion only in the Voyage and LKP+Dead-Reckon.

The object types represent different the types of things that we are looking for. Each object type is either the originating craft or a distress type. The “originating craft” is relevant only for pre-distress motion, and distress types are relevant only after the distress. Examples of distress object types are a “person in the water,” a “raft,” a “capsized boat,” etc.

Scenarios and Object Types are the main components of the data, but the largest set of data is the environmental data. Sim pushes around the distress particles by using wind and current data. The wind affects different object types in different ways, and each object type has “leeway data” that specifies this effect.

In addition, there may be completed (unsuccessful) searches and this has an impact on the probability distribution as well; particles that were well searched are less likely to represent the missing object and their probabilities or weights are reduced. Therefore, each object type needs a lateral range curve for Sim to update the particles’ weights from completed searches.

Summarizing, the main components of Sim.xml are:

1. Scenarios
   1. Type and Data
2. Object Types
   1. Type and Leeway Data
   2. Lateral Range Curve for each Completed Search
3. Specification of NetCdf files providing wind and current information
4. Area-of Interest specification for collecting land and bathymetric data

Table Main Components of Sim.xml

### Sim.properties

The data in Sim.xml are very simple xml: tags and attributes. A resource-file Sim.properties within the main jar file provides many defaults for the data in Sim.xml. The “fully qualified” name of a datum is its path of tags followed by its attribute name. For example, there is a slope attribute in the DWL tag, which is a sub-tag of the LEEWAY tag, which in turn is a sub-tag of the SEARCH\_OBJECT\_TYPE tag. Therefore, the fully qualified name of this datum is SEARCH\_OBJECT\_TYPE.LEEWAY.DWL.slope. The names of the properties are the fully qualified datum name, appended by either .DEFAULT or .OVERRIDE. The former indicates that the property should be used only if the datum is missing in Sim.xml and the latter indicates that the property datum should always be used.

In addition to the jar-file Sim.properties, there *can be* a Sim.properties file in the data subdirectory of the install directory. The properties in this file override the ones in the jar-file’s Sim.properties.

### Environmental Files

A Sim run uses wind and current information that are stored in NetCdf files. The rest of SAROPS produces these files and tells Sim where they are in the Sim.xml file. In addition, there are static files containing the shoreline and bathymetric data. These are large files, and the Sim.xml file tells Sim which parts of these files should be collected for the Sim run.

## Overview of the Document

In §II, we will discuss the different types of Scenarios. In §III, we will explain the data and method that a distress time is chosen. There are new features here. We give a brief description of reverse drift in §IV.

In §**Error! Reference source not found.**, VI, and VII, we describe how the environment affects the particles, and we finish with a discussion of Completed Searches, comparing them to Planner’s reports, in §VIII.

# Scenarios

Scenarios are basic way we describe what we know about the case. For example, we might know that the lost person intended to follow a certain route, or we might know only where the

## LKP and Area Scenarios

These are very similar. LKP stands for “Last known position.” A lat/lng is given together with parameters specifying a bivariate normal distribution. Because a bivariate normal is a concept on a 2-dimensional plane, a 2-dimensional approximation to the earth is created and draws are made for the initial position of each particle, from a bivariate normal distribution.

In addition, a time uncertainty is part of this data, and the initial time of the particles are also randomly drawn. This will be a repeated theme for all the scenarios, and it will have consequences described in §III.

The Area scenario is very similar, except that the initial distribution is a uniform distribution over a polygon. This time, we choose a planar approximation out of convenience. We implemented the distribution of a uniform polygon so that it would be efficient for all polygons, not just convex simple ones. Consider the blue polygon of Figure 1.

Some Island

Figure : Non convex Polygon

One common approach is to form a “bounding rectangle” around the polygon and make draws from it. Once such a point was drawn, one could test for containment in the blue polygon, and re-draw if the point were not in the blue polygon.

In this case, there would be many re-draws; most points within the bounding rectangle are not in the polygon itself. Moreover, the containment test can be quite expensive for complicated polygons.

Instead, a one-time setup of a data structure is computed and this allows 2 draws to produce a single point within the polygon itself. The first uniform draw represents the and this is translated to the -coordinate. The second uniform draw represents the , where is already known, and this is translated to the -coordinate.

We will use this track in the Voyage scenario of II.D, but only after a change of coordinates. The Voyage scenario has pre-distress motion and is a sequence of bivariate normal and uniform polygonal distributions.

## LOB Scenanrio

The LOB Scenario is similar to the LKP and Area scenarios in that there is no pre-distress motion; particles drift according to the winds and currents and their leeway data. However, the similarity ends there.

A collection of Lines-of-bearing calls (LOBs) are received and a single distribution must be made from them. This distribution plays the role of the bivariate normal or uniform polygon distribution of the previous section and, other than that, an LOB scenario is the same as these two.

If the LOBs are “consistent” (to be made precise shortly), a single elliptical distribution, is created and draws are made from that. If they are not consistent, the ‘worst’ LOB is removed and we try again. If we get down to two LOBs and these are still not consistent, then the LOBs are all used, but individually.

The data of a single LOB is:

1. Center Point (Lat/Lng) of Origin of Call, and x-error of center point
2. Called Bearing
3. Min and max range
4. Standard Deviation

Table : Data of Bearing Call

We follow the paper *A New Method of Position Estimation using Bearing Measurements*, by Dennis Wangsness and found in IEEE Transactions on Aerospace and Electronic Systems, and we call this algorithm “Wangsness’ Algorithm.” This method produces an elliptical distribution for the position. We modified the technique so we could consider a circular uncertainty in the origin of each LOB.

The ellipse must pass the requirements listed below. In this table, the center of the ellipse is called the “fix” and the ellipse itself is the 95% containment ellipse.

1. The area of the ellipse must be at most 3000 sq nmi
2. The semi-major axis of the ellipse must be at most 500 nmi
3. The ratio of the two ellipse’s axes must be at most 64
4. When one projects the fix to each bearing call’s great circle, the fix must be in the direction of the called bearing, as opposed to 180 degrees off
5. If there are exactly two bearing calls, their bearing calls’ intersections must be at least 5 degrees
6. The fix must be between the min and max ranges for each bearing call

Table : Requirements for an LOB Fix

The constants of Table 3 have never been given in Sim.xml; they are stored in Sim.properties.

If any of these conditions fails, we consider the set of LOBs to be a “bad ensemble” and an LOB is discarded. The one that is discarded is the one that is farthest from the fix, after normalizing by its maximum distance.

If we reach a situation where there are only two bearings left, and they still fail a test in Table 3, then we use a uniform distribution of all the lines of bearing as follows. Each line of bearing has an area; it is the section of the annulus defined by the minimum and maximum ranges, with angular bounds given by ± the standard deviation of the bearing. This is illustrated in orange in Figure 2.

Max Range

Min Range

Figure : Area of an LOB

The particles are divided (nearly) equally among thes lines of bearings’ areas, and the set of particles assigned to a line of bearing are distributed with a uniform distribution within the LOB’s area. Note that such a distribution has a closed form and is easy to generate.

It is possible to directly give an ellipse to an LOB scenario. In this case, all other inputs (LOBs and other ellipses) are ignored.

Wangsness’ Algorithm is quite elegant and straightforward. However, it does involve eigenvalues and linear regression, and we will not reproduce it here. Moreover, it does not allow for position uncertainty of the origin. We describe our modification for that here.

We will call the origin of the bearing call the “center,” and will define another point that we call the “vertex.” This vertex will be on the same great circle as the line of bearing, but behind it. It is based purely on geometry and not probability, and depends on the circle with containment of 0.5 in a bivariate normal. We call this circle the uncertainty circle, or COU. The radius of the COU is precisely the data that is provided for the LOB.

After the COU is given, we use the standard deviation of the bearing call to project backward to the vertex, as illustrated in Figure 3 and the following discussion.

Call the bearing uncertainty . In Figure 3, there is a great circle perpendicular to the line of bearing at the center (not shown in the picture). We went along the COU by , away from the LOB, to find the tangent points on the circle. We took the great circles that are tangent to the COU at these tangent points, and found their intersection; that is our vertex. We call the diagram in Figure 3 the “snowcone” diagram, where the COU is the “snow” and the projection back to the vertex is the “cone.”

From the vertex, we compute the ½-range of angles between the vertex and the tangent points and used that as the standard deviation of the bearing call. Note that if the bearing uncertainty is 0, the vertex is the antipodal point of the center. If the position uncertainty is 0, we simply define the vertex to be the center, and use the given standard deviation.

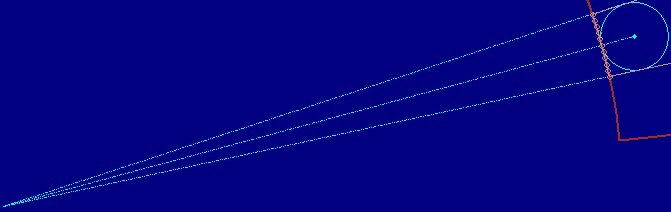


Figure : Vertex from Snowcone and Uncertainties

## Flare Scenario

A flare scenario is like an LOB scenario. In fact, the code represents a flare scenario as a special type of an LOB scenario with one LOB. Both are represented by the class com.tmk.sarops.model.LobScenario. The main difference is that a flare scenario behaves much more like an area scenario, where the area is a polygonal approximation to the orange area of Figure 2.

In theory, there could be more than one flare sighting for a scenario. In practice, SAROPS forbids this. If it were allowed, the intersection of the orange areas would be used. Additional sightings are allowed until the intersection becomes empty.

## Voyage

The voyage scenario is the first of two scenarios with pre-distress motion. The voyage scenario represents a situation in which the subject scheduled one or more of a sequence of stops or Dwell Zones, but is not proceeding as suggested.

The Dwell Zone are either polygons or circles, like the LKP and area scenarios. For each particle, a sequence of positions is chosen, one from each Dwell Zone. A loiter time is also picked for each Dwell Zone, and these. For each particle and leg (transit between Dwell Zones), a speed is selected. The result of all this is a complete trajectory or path for each particle.

For each particle, the position within each Dwell Zone is not independent of the previous or next Dwell Zone. Rather, the draws are correlated. In the code (not even in Sim.properties), there is a constant , called the correlation value, and it is currently set to 0. The name of the constant is paying homage to the correlation coefficient, but we make no claims that this is the correlation coefficient. For that matter, we’re not entirely sure what the correlation coefficient between two 2-d random variables is anyway. However we define 2-d correlation, we would get perfect correlation if we set to one, independence if we set it to 0, and perfect anti-correlation if we set it to -1.

Dwell Zone A

Dwell Zone B

Figure : "Correlation” between Dwell Zones

We illustrate our technique in Figure 4. We suppose we are going from Dwell Zone A to Dwell Zone B, the stars indicate the centers of mass of the two zones, the red axis is the great circle connecting the two stars, and the green axis is perpendicular to the red axis at Dwell Zone A’s center of mass. Suppose also we wish to “correlate” the brown triangle to some point in Dwell Zone B.

With respect to the green axis, let’s suppose that the brown triangle has a value of 0.8. If we did “perfect correlation,” we would pick a point in Dwell Zone B that had a value of 0.8 with respect to the purple axis. That axis was chosen as the great circle perpendicular at Dwell Zone B’s center of mass to the red great circle. To get the value within Dwell Zone B, we use something similar, but base it on the conditional of . Let’s say that the result is the purple square that is connected to the brown triangle by the dashed line. We call this point the “partner” of the brown triangle.

We do not use perfect correlation, but we do base our draws on the partner. Now note that, for any continuous random variable , is a -(0,0) random variable. We convert the value to a Gaussian one by using the inverse of a Gaussian, and make a (truly) correlated draw for a new Gaussian by using the formula to get a new Gaussian. We convert that to a by finding its value, and then using that uniform and our method of drawing a uniform polygonal draw, to get the draw in Dwell Zone B.

This correlation was requested long ago because it really doesn’t make sense for the two draws to be independent. What this is capturing is simply the notion that “if someone starts to the left, he generally stays more left than right.”

In the following figures, we show the same simulation case with 100 particles, but change the correlation value. The dark green segment is the section of the great circle that connects the two areas’ centers of mass. This is most visible in the first picture of Figure 5. The other pictures are for “perfect correlation,” perfect anti-correlation,” “no correlation,” and “correlation = 0.75,” which is what is use in Sim.

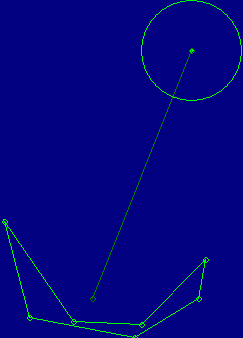
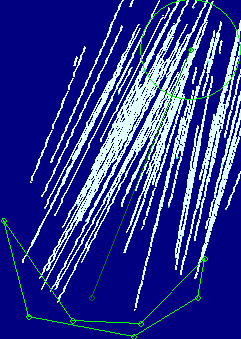
 

Figure : Connecting Centers of Mass, and Perfect Correlation

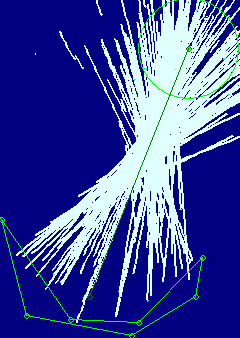
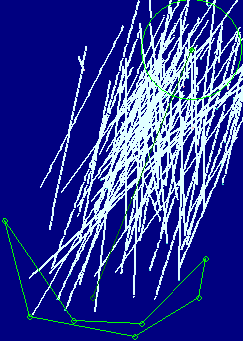
 

Figure : Perfect Anti-Correlation and No Correlation

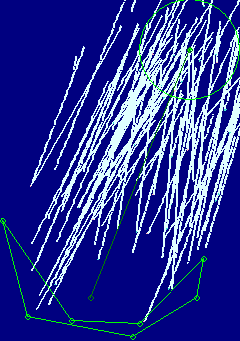


Figure : Correlation = 0.75, used in Sim

## LKP Plus Dead-Reckon

This one is conceptually simpler. For each particle, we draw a starting point and time as with LKP’s, and a direction. The particle moves in that direction until distress; the last random characteristic for the pre-distress is the direction.

# Hazards and Distress Incident

For the pre-distress motion of a particle from a Voyage or LKP-Dead-Reckon scenario, a distress time and place must be chosen. The simplest way of doing this is to lay out a route for the particle and ignore distress considerations. Compute the beginning and ending time of the route and make a uniform draw for the time of distress. This is essentially what is done, but it is complicated by “hazards,” prescribed distress times, and the requirement that a particle go into distress over water.

A hazard is an area, a time-interval, and an intensity. The area is fixed, the time-interval tells us when the hazard is active, and the intensity tells us how much more likely particle was to get into trouble while in the hazard. Hazards do not affect the route of the particle; only when it gets into distress. To illustrate, suppose we have a particle whose planned route (ignoring that it will get into distress at some point) is shown in Figure 8. For example, this particle is 3 times as likely per unit time, to go into distress during times to . His must vulnerable period is between and ; there his “hazard score” is .

Let’s suppose the particle is travelling between and . If there were no hazards, a uniform draw between 0 and 20 could be used to assign the particle a distress time. But with hazards, we compute an “expanded time.” Note that the intensity of a hazard-free section of the trip is the same as a hazard with intensity 1, our expanded time is 5-2). The draw is translated to a normal time. The code is somewhat complex since it must deal with overlapping hazard intervals, but the underlying idea is no more complicated than this example. It’s slightly more complicated going from a draw between 0 and 177 to a time, but the formulae are more cumbersome to write out than difficult to understand.

3

5

2

10

Figure : Intervals of a Hazard for an Example Particle

An additional complication is that, for hazards, there are Dwell Zones, and the hazards may turn on or off while “dwelling.” Still, the main idea of using hazards to determine a distress time, is captured in the above example.

Whether there are hazards, loiter times, both, or neither, the distress time for a particle can be computed with the aid of a single uniform draw. There is an additional wrinkle though; the distress *interval* can be given within the xml. Sim simply computes a distress time that is within this interval by repeatedly making draws until it gets a distress time in the interval.

The same strategy is used if Sim.xml states that the particle should go into distress in water. For example, if the pre-distress motion is modelling a plane flying over an island, then the human planner might not want particles to start on that island. If a particle does go into distress over land, again a draw is made.

This strategy of repeating draws until something acceptable occurs, could result in too long a runtime. Sim has code in place that prevents this, and it is summarized in the following rule:

If at least 1000 draws have been made, and Sim has been successful less one percent of the time, the next draw is accepted. Such a draw is not counted as a success. The number of draws and successes accumulates across the particles.

Equation : Formula for Ensuring Particles get Drawn

Finally, we mention a rule that is implemented in the code that calls Sim.

If a distress interval is given, and the maximum start time of the particles is after the beginning of the distress interval, the distress interval is ignored. Similarly, if we apply the maximum speeds between the means of the Dwell Zones, and the earliest possible start time for a particle, and the result is an arrival time that is less than the end of the distress interval, the distress interval is ignored.

Equation : Ignoring a Distress Interval

What this enforces is that the planner should not specify a distress interval that includes times for which some of the particles are not in distress. This is easy to enforce on the early side; we know the latest time that the particles can go into distress. In other words, it’s easy to compute the “latest possible pre-distress time” for a particle.

The “earliest possible post-distress time” seems like a strange concept, but we use that expression only to be analogous to “latest possible pre-distress time,” and we mean the earliest possible ‘arrival’ time.” Here, ‘arrival’ means that the subject would have completed its scheduled itinerary. This is not nearly as easy to compute, and so we use the fastest speeds, earliest start time, and the *means* of the Dwell Zones. It’s not hard to create pathological cases where the earliest arrival time is considerably earlier, but the technique we use will work for practical cases.

# Reverse Drift Scenario

Sim runs “reverse drifts.” The situation being modelled is that debris is spotted, and a reverse drift is used to find the distribution for distress point. A subsequent “forward drift,” (i.e. normal case) would be run.

Reverse drift cases have only that one scenario and are either LKP or Area. The Particle File’s times must be ascending, but in a reverse drift, the natural way of computing the time steps is descending. Furthermore, the environmental data must be turned around to drift back in time.

The strategy within the code is to run a forward drift, and use the concepts “RefTime” and “SimTime” to distinguish between the times in this forward drift, the real times, and the times used in the environmental look-ups.

# Winds and Currents

The data for winds and currents comes as two NetCdf files, and these files are specified in Sim.xml. However, constant values can also be used, and that would be specified in Sim.xml. This latter option is used primarily for debugging.

The effects of the winds and currents on a particle’s movement depend on the type of particle. The current is modelling the “wind-driven current,” so the effect of the current on every particle is the same; the particle goes where the current goes.

The effect on the particle’s position due to the wind is more complicated and it is here that we have dependence on the Object Type, and we will discuss that shortly. However, there is nothing complicated about how we combine the movement due to the wind, and the movement due to the currents; we simply add these two vectors.

## Leeway Data and Using the Environment to Move Particles

Every Object Type has a Leeway Dat, which we use to compute the movement due to the wind. The constants in the Leeway Dataare given in Table 4.

1. Downwind Slope and Constant
2. Downwind Spread (twice the standard deviation)
3. Crosswind Slope and Constant (for Positive)
4. Crosswind Spread (for Positive)
5. Crosswind Slope and Constant (for Negative)
6. Crosswind Spread (for Negative)
7. Gibing Frequency per second
8. Nominal Speed
9. Use Rayleigh (Boolean)

Table : Leeway Data

For each particle, we make random draws and convert a Leeway Data into that particle’s (constant) Leeway Calculator, which is given below:

1. Downwind Slope and Constant (Initialized and then Constant for this Particle)
2. Crosswind Plus Slope and Constant (Initialized and then Constant for this Particle)
3. Crosswind Minus Slope and Constant (Initialized and then Constant for this Particle)
4. Gibing Frequency (Initialized and then Constant for this Particle)
5. Using plus (Boolean)
6. Next ‘flip’ (seconds)

Table : Leeway Calculator

The first 4 entries of this table are constant; the last two are used to determine when to switch from crosswind-plus to crosswind-minus.

A Leeway Calculator is used with random draws to produce the offset from the means. The means are simple look-ups from the environmental data, combined with interpolation.

For each particle, the draws that are used in conjunction with the Leeway Calculator are correlated with draws for the same particle and previous time steps. For each particle and time step, there are two such draws; one for the east direction and one for the north. The east direction draw is correlated with the draw for the previous time step’s east draw, and similarly with the north draws. The correlation coefficient exponentially decreases from 1.0 to 0 as a function of the time between time steps. In this way, and we skip the details here, the correlation between non-consecutive time-steps does not depend on the length of the time-steps.

With the effect due to the water, we have no such complicated notion; the effect due to the currents is simply the movement of the water itself; we say that the particles simply go where the water goes. Correlation between time-steps is the same though.

## Interpolation for finding the Mean

We have mentioned interpolation. The data that we use has data for a fixed set of lat/lngs. This set is not necessarily a grid, and it consists of east and north speeds for a set of times, for each lat/lng in the set. If some time is missing, we fill it in. Hence, we have a dense multi-dimensional matrix indexed by lat, lng, time, and {east,north}.

To compute the mean (say) east speed for a given lat, lng, time combination, we find the bordering two closest time-layers, compute the mean in each of these time-layers, and interpolate these values using time.

Within a time-layer, we do not interpolate in lat and lng. Our method is different for ocean cases compared to river cases. In ocean cases, the far simpler case, we find the closest 3 points for which we have data, and take a weighted average. Here the weights are the inverses of the distances to the three points.

For “river” cases, the interpolation is far more complicated. We consider a case in the Pacific Northwest, where there are two rivers; the Willamette and the Columbia. The Willamette flows into the Columbia, but for now, we ignore that and simply suppose that we have two rivers.

To be a river case, the each lat/lng in the data must have a strip identifier (center, left, or right), a river identifier, and a sequence number, and these sequence numbers increase as we go downstream. These sequence numbers are used to form 3 sequences of edges (or strips); one for the center, one for the left, and one for the right, for each river. So, in our case, each datum will have whether it is for the Willamette or the Columbia, whether it is center, left, or right, and its sequence number. All o fthis is in addition to the time of the datum, the lat/lng of the datum, and of course the east or north speed itself.

We use the river/strip/sequence information to form edges rather than points, and we look for the closest edge instead of the three closest points. Note that in our case, we have 6 strips of edges.

We are currently using “Center Dominated.” This means that we find the closest edge from a center strip. Suppose we started with LatLng and the closest center-strip point is , and the distance between these two points is . We then check to see if we go the opposite direction from to , will we run into either edge-strip within ? If so, we use the east and north speeds for that point; otherwise we use the data for .

If we do not use Center-dominated, we simply find the closest edge and use that.

# Land Processing

## GeoMtx

## Slippery Shore, Stop-and-Block

# Bathymetry and Anchoring

# Completed Searches

Table Parameterization of a Pattern in *Planner* 2.2

# Appendix A: Polishing a Rock; the List of Minor Moves

We call the moves to nearby solutions “minor moves.” When we say “nearby,” we mean that the corresponding sruSolutionArrays are very similar. In fact, all but one of them will be the same, and the one that is different will differ only by a slightly different orientation, or a slightly different center point, or some other such minor adjustment.

# REFERENCES

1. Computer Aided Search and Rescue Planning (CASP), Volume 1
2. Computer Aided Search and Rescue Planning (CASP), Volume 2
3. National Search and Rescue Manual, Volume I: National Search and Rescue System, Joint Chiefs of Staff, U.S. Coast Guard, Joint Pub 3 50, COMDTINST M16120.5a, 1 February 1991
4. Addendum to The United States National Search and Rescue Supplement (NSS)  
   To The International Aeronautical And Maritime Search And Rescue Manual (IAMAR),  
   U.S. Department Of Homeland Security,  
   U.S. Coast Guard, COMDTINST M16130.2d, 29 April 2004
5. Memorandum “SAROPS Lateral Range Curves,” USCG (G-Opr-1), 16131, 11 March 2004
6. Enclosure (1) To Memorandum “SAROPS Lateral Range Curves,” USCG (G-Opr-1), 16131, 11 March 2004, “Discussion of SAROPS LRC And MSPP Issues”
7. Enclosure (2) To Memorandum “SAROPS Lateral Range Curves,” USCG (G-Opr-1), 16131, 11 March 2004, Excerpt From “Elements, Modules, And Algorithms For Planning Optimal Searches
8. Software Requirements Specification For The Naval Search And Rescue (NAVSAR), Naval Oceanographic Office, NAVSAR 01-12, December 2001