GNOME Technical Documentation

Christopher H. Barker, Robert Jones, William J. Lehr,
Amy MacFadyen, Caitlin O'Connor, James Makela, Jay Hennen

April 11, 2023

Contents

Chapter 1

Overview

Dramatic incidences of marine pollution, such as the Deepwater Horizon oil well blowout and the wreckage of the oil tanker Exxon Valdez, have highlighted the potential for human-caused environmental damage. In attempting to mitigate or avoid future damage to valuable natural resources caused by marine pollution, research has been undertaken by the scientific community to study the processes affecting the weathering and distribution of marine pollution, and especially to model and simulate these processes.

One area of research is the computerized Lagrangian transport or trajectory model (ASCE). Trajectory models attempt to predict the movement of actual or hypothetical pollution spills. Needless to say, all existing computerized trajectory models have their virtues and their defects. It is probably not possible to devise a single computational model or framework to satisfy all users. In developing GNOME, we have tried to balance the contradictory notion of a comprehensive model that is both easy to use and that rapidly produces useful and accurate results. This is especially important since GNOME is primarily a forecaster tool.

TJB-Citation here?

GNOME is an interactive environmental simulation system designed for the rapid modeling of pollutant trajectories in the marine environment. The overall design is that of modular and integrated software. Inputs to GNOME include:

- maps,
- bathymetry,
- numerical circulation models,
- location and type of the spilled substance,
- oceanographic and meteorological observations, and
- other environmental data.

The output from the model consists of graphics, movies, and data files for post-processing or to import into a geographic information system (GIS).

redo this section

GNOME is written in Python; some of the more computationally intensive algorithms are written in Cython or C++, with careful attention to exploit the language's classes and objects. This makes the model easier to update, expand, and improve. The model also contains a web front end tested for clarity and ease of use. Variations of the GNOME code are used for contingency planning in conjunction with ERD's Trajectory Analysis Planner model, and in Ecological Risk Assessment workshops for evaluation of tradeoffs when using dispersants.

Add info on weathering. DAH

Chapter 2

Introduction

2.1 Trajectory

Often it is inquired if a trajectory model is accurate, adequate, or correct. Trajectory models should always be correct, as a direct function of the coding of their algorithms. The accuracy and adequacy of a model are more often associated with the data used for the calculations and physical processes modeled. Only by clearly separating the data from the model can the questions of accuracy and adequacy be clearly addressed.

GNOME can be used with various data sources. Because of the ease with which GNOME can handle many datasets, GNOME allows the user to compare datasets, and to explicitly show the various results from different datasets in the same format. The basic data components are maps, movers (wind, currents, and diffusion), and spills.

Note that it is the user's responsibility to determine the accuracy and adequacy of any proposed dataset that might be used. The model has been extensively tested and verified. A clear distinction between data inputs and the trajectory model is maintained. Keep in mind, however, that the results are only as good as the model inputs.

The model is general and applies to trajectory problems. It is an Eulerian/Lagrangian model that is three-dimensional (3-D) in space. Shoreline maps are inputs to the model, so any area can be modeled. The model automatically handles either hemisphere and east or west longitude. Shorelines (i.e., maps) of varying resolution are available for download via the GNOME Online Oceanographic Data Server (GOODS [1]) *Map generator tool* at `http://gnome.orr.noaa.gov/goods`. Alternately, sophisticated users should be able to manually generate a custom map file in less than two hours, consistent with the Emergency Response Division's (ERD) requirement to produce a trajectory within two hours of notification of a spill event. GNOME is also fully scriptable, so that batch runs can be driven by a text command file.

The model is designed so that both novice and more sophisticated users can get their needs met. Certain regions are modeled with Location Files (prepackaged map, tide, and current data with a simple Wizard interface to help people run scenarios on their own) where the parameter values are predetermined. The user merely selects the location file corresponding to his or her region of interest and is guided through dialog boxes to set the wind speed, direction, and spill information. Alternatively, users can set up a location from scratch. This requires a user sophisticated in oceanography and modeling to set up input data and interpret results. The user has control over inputs but must have a lot of knowledge of the modeled region to select parameters. The more sophisticated user can rapidly build trajectory models without a major investment in man-years for programming and testing of code. For instance, it is possible to build a complete model including shorelines, currents, winds, and spill distribution, and run the model in less than two hours, or in four hours, if starting from a paper chart for shoreline and bathymetry.

In utilizing a database management and an integrated software approach to model development, we have attempted to develop a generalized trajectory model with enough flexibility to satisfy a large number of users without making it overly complex. This is probably more hopeful than actual. It is taken as axiomatic that a model that does everything for everyone will be used by no one, because it will be too difficult to find

[1]GOODS was developed to help GNOME users access ocean currents or winds from various models and data sources.

one's way through the labyrinth of code. The fact that this model is used by researchers other than just ourselves is an encouraging sign that we have produced something of use and value.

2.2 Weathering

[this section empty]

2.3 Model Overview

2.3.1 Trajectory

GNOME is fundamentally a Lagrangian element (particle tracking) model. The oil or other substance is represented as Lagrangian elements (LEs), or particles, in the model, with their movement and properties tracked over time. The elements are acted on by a number of "movers," each representing a different physical process. For the most part, each mover moves the particles one way or another, but a mover can also act to change the nature of a particle, rather than moving it. Movers can be ocean currents, winds, turbulent diffusion, and weathering processes (evaporation, etc). Each move is initialized with the data it needs to compute the movement (or links to files with data in them). A GNOME model consists of a map, spills, and movers.

Spills in GNOME are a source of elements - they define where and when the elements are released into the model. Each spill is a composition of the type of substance spilled and how elements are released (instantaneous, continuous, etc). Each spill is initialized with an element type that defines what properties the elements have. The elements may represent something spilled such as oil, or they could be any other object, objects, or substances that you want to track – chemicals, floating debris, fish larvae, etc.

The element type itself contains a 'substance' that defines the properties of the substance spilled and a list of initializers. These characteristics are used to define data arrays associated with the type of spill, i.e. floating debris, floating weathering particles, subsurface, etc.

The map in GNOME defines the domain of the model. It can consist of bounds, shoreline (to define where land and water are), and properties of the shoreline. For 3-D modeling, it can also define the bathymetry.

The output options consist of data files for post-processing or GIS, graphs, and movies.

2.3.2 Element properties

The default properties that are always in the model are positions, next positions, last water positions, status codes, spill number, id, mass, and age.

Other properties that are included as needed are windage, windage range, windage persistence, initial mass, area, viscosity, water fraction, interfacial area, bull time, fraction lost, mass components, partition coefficient, droplet average size, rise velocity, droplet diameter, density, bulk initial volume, evaporation decay constant, at max area, substance, and weathering status.

There are also some internal properties that are used for diagnostic purposes – fay area and fraction coverage.

Table 2.1: Outputs required

Variable	Type	Computational units	Range (min,max)	Expected accuracy
LE density	scalar, calculated	kg/m^3	tbd	10 %
Slick average density	scalar, calculated	kg/m^3	tbd	10 %

See Table 2.1.

TJB-NOTE: this [pulled from elsewhere, maybe not useful

2.3.3 Weathering

[this section empty]

Chapter 3

Transport

3.1 Maps

GNOME is not specific for any particular region, and no specific shoreline is built-in. To run a scenario, the user must download a map via GOODS's "Map generator" tool at `http://gnome.orr.noaa.gov/goods`, or create a map. GNOME accepts maps with shoreline data in the form of boundary file atlas (BNA) maps (such as those produced by GOODS; see Appendix A for format description). For BNA maps, we use vector shorelines of varying resolutions that NOAA provides. The resolution of the map is not gridded, because it is a vector shoreline. Each map is rasterized into a land/water bitmap for the purposes of tracking the oil beaching. The land/water bitmap is of finite resolution, so it doesn't exactly match the map outline.

GNOME also has an all-water boundaries map by default, so the user can set a spill without importing a map file. To do this, the user specifies a latitude/longitude rectangle defining the domain (or they can use the default of the whole world). This is particularly useful for spills that are far offshore – particularly deep-water well blowout scenarios – or for diagnostic testing.

3.1.1 2-D Beaching

At each time step after the Lagrangian elements (LEs) have been moved, GNOME checks the map (i.e., as a bitmap image) to see whether the new LE positions are on land or in the water. The beaching algorithm checks the entire line on the bitmap between the old point and new point to make sure the LE didn't jump over land, and the LE is beached at the first land box it hits. The location in the water right before the land is reached is also stored to use as a starting point when a particle is re-floated. While an LE is beached, neither weathering processes nor interaction of the pollutant with sediment and biota on the beach are modeled.

3.1.2 2-D Refloating

After beaching, an LE can refloat. Refloat half-life is a single parameter that encompasses a number of different parameters and that empirically describes the adhesiveness of the oil to the shoreline. It is a function of substrate porosity, the presence or absence of vegetation, the inherent stickiness of the oil, and other physical properties and processes of the environment. Refloat half-life is the number of hours in which half of the oil on a given shoreline is expected to be removed if (1) there is an offshore wind or diffusive transport and (2) sea level is at the same level, or higher, than the level of the oil when it was beached.

The refloat half-life parameter, along with the other environmental data, allows refloating of oil after it has impacted a given shoreline. Refloat half-life is one hour by default; if the value is higher, the oil will stick to the shoreline longer (as for a marsh), whereas for very small values, the oil refloats immediately (as for riprap). In theory, the refloat half-life could be set to different values along different segments of the shoreline depending on the beach type, but we have not found it necessary to have this degree of refinement in the refloat half-life parameter value.

The probability of an LE refloating, Pr_{rf}, determined with the default refloat half-life of one hour, gives

$$Pr_{\mathrm{rf}} = 1 - e^{\frac{-t \cdot ln(2)}{(1\mathrm{hour})}} = 1 - 2^{-t} \tag{3.1}$$

where t is measured in hours. Refloating for an individual LE is determined by choosing a random number on the interval $(0,1) : R_{(0,1)}$ for the LE. If $R_{(0,1)} < Pr_{\mathrm{rf}}$, the LE is refloated.

When an LE is refloated, it is placed at its last water position before beaching.

3.1.3 3-D Interaction with Bottom

[This section empty]

3.2 Movers

Movers are any physics that cause movement of the pollutant (i.e., oil) parcel in the water – generally currents, winds, and diffusion. GNOME is designed to model the various physics influencing the movement of the oil using simple linear superposition. At each model time step, each process (mover) computes how it would move an element as though it were acting alone. The model then sums the movement from each mover to get the position at the next time step. In order to do this, each mover returns not a new position, but a "delta" – the difference between the previous and new positions. These deltas are applied to the element's position after all the movers have been applied. Thus the final position is independent of the order in which the movers are computed.

As different movers, maps, etc. may use different internal coordinate systems, GNOME uses geographic coordinates: latitude and longitude (stored as double precision floating point values) to keep track of the elements' positions. Thus lat-long is the "lingua franca" of the model – all interaction between movers, maps, etc. is done in lat-long coordinates. It is assumed that all data share the same datum – this could be an issue if your maps do not match hydrodynamic models, etc., but we have found that discretization of the grid is usually a bigger issue than datum.

Each mover is responsible for conversion to and from lat-long to whatever it uses internally. Positions are passed to the movers in lat-long, and they are expected to return the resulting movement in lat-long. In most of the GNOME movers, a simple spheroidal earth projection is used to convert from a distance to a change in latitude or longitude. It is assumed that the distances are small enough that a geodesic computation is unnecessary.

Each mover present in the model setup may be active or inactive at any given time. Only movers marked active will be used in the model calculation.

3.2.1 Gen 2 Movers

Generation 2 (GNOME) movers are the latest iterations of the algorithms implemented in the original GNOME project.

3.2.1.1 Current Movers

The various current movers in GNOME can work with currents provided by several hydrodynamic models using a variety of grid types, and both fully time-dependent patterns as well as patterns scaled by a time series, such as the tidal predictions. However most of the movers share the same computational scheme.

To get the overall movement in the u (east-west) and v (north-south) directions from currents, a forward Euler scheme (a.k.a., a 1st-order Runge-Kutta method) is used. The movers are given an initial point (x, y, z, t) and time step, and return a displacement $(\Delta x, \Delta y, \Delta z)$ at $t + \Delta t$ (see Equation 3.2). The displacements are in units of latitude and longitude. Equation 3.2 shows the simple Euler scheme as well as the conversion from distance in meters to latitude and longitude.

$$\Delta x = \frac{\frac{u(x,y,z,t)}{111{,}120.00024} \cdot \Delta t}{\cos(y)} \Delta y = \frac{v(x,y,z,t)}{111{,}120.00024} \cdot \Delta t \Delta z = w(x,y,z,t) \cdot \Delta t \tag{3.2}$$

where:
Δt, or $t_{i+1} - t_i$, is the time elapsed between time steps $i + 1$ and i,
$\Delta x, \Delta y$, and Δz are the 2-D longitude, latitude, and vertical displacement, respectively, for that time step,
w is the vertical component of velocity,
y is the latitude in radians, and
111,120.00024 is the number of m per degree of latitude (assumes 1' latitude = 1 nautical mile everywhere).

In the common case, w is taken to be zero, but if a hydrodynamic model has full 3-D currents, it can be included. The changing radius of the earth is neglected.

3.2.1.1.1 Current Mover Types The current movers generally fall into two categories: a current pattern that is scaled by an amplitude time series, and fully time-dependent currents generated by an external hydrodynamic model. In each case, the currents themselves can be on various grid types and loaded from a variety of file formats. Currents can be provided for any number of sources. One useful source of current data compatible with GNOME is the GNOME Online Oceanographic Data Server (GOODS) operated by NOAA's Emergency Response Division: `http://gnome.orr.noaa.gov/goods`.

3.2.1.1.2 Scaled Current Patterns The primary source of current patterns used with GNOME is NOAA's Current Analysis for Trajectory Simulations (CATS) model. This is used for most of the currents in GNOME's pre-setup location files. CATS is a 2-D depth-averaged, steady-state, finite-element circulation model. CATS generates constant patterns that are made time-dependent in GNOME by connecting them with a time series, such as tidal coefficients. These patterns are fairly quick to generate and easy to adjust during a response. The patterns are on a triangular mesh, which allows for good shoreline matching and higher resolution near the shore.

Steady-state current patterns such as those from CATS can be driven by various time series. If a current pattern represents, for example, the flow of a river, it would be scaled by a time series of river flow data. Most commonly, scaled current patterns are used for tidal currents. In this case, the pattern is scaled by the tidal current values, usually estimated from tidal constituents.

The tidal representation in most location files is of the form $\vec{U}(x, y)\vec{T}(t)$, where $\vec{T}(t)$ is the tidal velocity. A spatial pattern from CATS, $\vec{U}_{\text{CATS}}$, is used for the currents and then the NOAA tidal currents time series for the nearest tide station are used to adjust the velocity and direction (ebb or flood). So, for a station at the point x_0, y_0, the currents at any point $\vec{U}(x, y)$ are given by:

$$\vec{U}(x, y) = \vec{T}(x_0, y_0, t) \cdot \frac{\vec{U}_{\text{CATS}}(x, y)}{\vec{U}_{\text{CATS}}(x_0, y_0)} \tag{3.3}$$

Tide files are either constituent data from ERD's tide model SHIO, or a time file of data points that are interpolated in GNOME using a Hermite polynomial fit. The currents can also be scaled using tidal heights (constituent data from SHIO) or hydrology (a time file). In all of these cases the user can input a scale factor.

3.2.1.1.3 Current Cycle Mover [This section empty]

3.2.1.1.4 Component Mover The component mover is essentially a wind-driven current. The mover can have one or two current patterns that must be in the form of CATS currents, which are scaled by a constant or time-dependent wind. Each pattern can be scaled by the component of the wind in a given direction. Typically, there might be a pattern driven by the alongshore component of the wind and another pattern driven by the cross-shore component. The wind is scaled to a reference point in the current pattern that has a user-specified value for a given wind speed.

The components of a current can be scaled linearly by wind speed or by the square of wind speed (i.e., wind stress). The current pattern can also be scaled by a time average of past wind values. This time-averaged option captures the lag in the response of the currents to wind forcing, a behavior often observed in the advection of drifting matter such as harmful algal blooms (HABS). Here the current component can be scaled by any power of the time-averaged wind; after the wind is averaged, any multiplicative scalar is applied. If no historical winds are available (or the wind record is insufficient to satisfy the selected time over which to average), the model can be set to extrapolate the appropriate values, and it will use whatever wind is available until enough data are accumulated.

3.2.1.2 Time-Dependent Currents

General circulation models such as nowcast/forecast models can also be used as input to GNOME. These can include surface currents only (2-D), or fully 3-D currents. In general, the vertical component of the velocity is not used, but it may be applied in some use cases.

GNOME accepts time-dependent current model data on rectangular, curvilinear, and triangular grids from various types of models.

update/add sections based on Jay's works

3.2.1.2.1 Rectangular Grids For rectangular grids, the velocities are considered constant throughout each grid box, rather than providing any interpolation.

3.2.1.2.2 Triangular Grids Triangular grids are generally used by finite-element models (such at CATS). In this case, the velocities are given on the triangles, and generally represent the average velocity of each triangle. GNOME applies the given velocity across the entirety of each triangle.

For triangular grids, if an imported model has velocities at the nodes, the values are interpolated. CATS produces velocities at the centers and there is no interpolation. All the time-dependent currents are interpolated linearly in time.

TJB-If average velocities are used across the triangles (paragraph 1), then when is interpolation needed (paragraph 2)?

3.2.1.2.3 Directional Acyclic Graph (DAG) Tree To navigate around the grid topology of triangular meshes, GNOME uses a Directed Acyclic Graph (DAG) data structure. The DAG tree provides the means to identify what grid cell each element is in, so that the appropriate velocity node can be applied. The LE advances in latitude and longitude space by that velocity.

The DAG tree is an ordered list of all the line segments connecting nodes in the grid domain. As the LE is checked whether it is to the left or right of a given segment, the DAG tree indicates where next in the list to check to find out in which triangle the LE is. This needs to be a very fast algorithm because typical model runs are comprised of 1,000+ elements in domains with 10,000+ triangles.

The DAG tree algorithm is $O(logN)$, thus performing much faster than searching for the nearest neighbor by checking every single point, which would take an enormous amount of computational time. In addition, the DAG tree allows GNOME to identify when an element has crossed out of the domain of the hydrodynamic model.

3.2.1.2.4 Curvilinear Grids Curvilinear grids are handled by dividing each grid box into two triangles, creating a triangular grid, so that GNOME can apply the same algorithms as for triangular grids. The velocity is interpolated over the cell.

3.2.1.3 Wind Movers

GNOME allows several different wind movers – constant, time-dependent, and time/space-dependent. The first two can be loaded either by hand through a wind dialog box, or loaded from a file in the On-Scene Spill Model (OSSM) wind file format (Torgrimson ref here).

Add missing ref. DAH

There are also some GOODS OSSM style options, such as a National Weather Service point forecast. The files contain date, time, speed, and direction information (see Appendix A for more details).

The spatially varying winds must be loaded via a file, either in GNOME's American Standard Code for Information Interchange (ASCII) or Network Common Data Form (NetCDF) format, and can be obtained via GOODS for a few select models. The time-dependent wind is interpolated linearly in time. The spatially varying wind is interpolated linearly in time, but not interpolated in space; it can be on either a regular or a curvilinear grid.

GNOME also allows for winds from the National Weather Service's National Digital Forecast Database, but these files require some pre-processing. An alternative to spatially varying wind files is to load data as current files, and "trick" GNOME by decreasing the value of the wind speeds to 3% of the original speeds.

3.2.1.3.1 Windage

Windage is the movement of oil by the wind. This is typically about 3% of the wind speed, based on analytical derivation and empirical observation that oil tends to spread out in the direction of the wind (see Appendix C) (Stolzenbach, Madsen and Adams ref here). Experience and observation have led us to use a factor in the range 1–4%, which is adjusted based on overflight reports (Lehr and Simecek-Beatty ref here). This range is used as the default in GNOME with a uniform distribution.

Add refs. DAH

Add refs. DAH

A given oil droplet will move differently depending on how close it is to the wind effects at the surface. The windage is lower as the oil weathers and spends more time below the surface.

As much as possible, the model should behave the same when the time step is changed. Thus GNOME accepts a range of windage percentages and a persistence time step. The LEs are moved accordingly to get the desired amount of spreading. The persistence time step is how long until the random value is reset.

Currently there are two options for persistent time step: 15 minutes is the typical (default) persistence time step for oil, and infinite persistence is used when modeling heavier floating objects. As a rule, one might use the 15-minute persistence for modeling something, such as oil, in which the windage of individual particles will increase and decrease with time as oil is pushed below the surface by waves, and then floats back up to the surface. Infinite persistence is used when each of the particles has a different windage, but the particles maintain that difference indefinitely, such as floating wreckage from a boat. The windage is a property on the LEs in a spill, and thus applies to any wind mover set up by the user.

GNOME picks a random number within the user-selected range of windage values for each LE, and moves the LE according to that number at each time step. This method is very similar to the method used to compute diffusion, except the spreading happens only in the direction of the wind. The amount of spreading is given by:

$$\frac{d\sigma^2}{dt} = S(t) \tag{3.4}$$

where:
σ^2 is the variance of the LE's locations, and
$S(t)$ is a spreading parameter that is a function of time (wind velocity is a function of time).

For a constant wind, S would be constant and $\sigma^2 = S(t)$. That is, the variance of the particles grows linearly in time, the same as diffusion with a constant diffusion coefficient (0) (reference here?).

Add ref? DAH

3.2.2 Gen 3 Movers

Generation 3 (PyGNOME) movers are movers that work with the new Python-only environment object system.

3.2.2.1 New Algorithms

As described in the previous section, the forward Euler method was the primary method for moving particles once a velocity had been computed. In the Gen 3 movers, this has been supplemented by implementations of second and fourth order Runge-Kutta methods, and it is possible to choose between them at any point. The second order method is as follows:

$$y_{n+1} = y_n + \frac{h}{2}(f(t_n, y_n + f(x_n + h, h * f(t_i, y_i))) \tag{3.5}$$

Confirm that closing parentheses are in correct locations in equation. DAH

where:
y_n is the particle position at step n,

t_n is the time at step n,
$f(t_n, y_n)$ is the velocity field function, and
h is the length of the time step.

Define x_n or $f(t_i, y_i)$? Add to list of symbols? DAH

This is also known as Heun's method, or the forward Euler method with trapezoidal rule corrector. The fourth order method is as follows:

$$y_{n+1} = y_n + h\sum_{i=1}^{4} b_i k_i \tag{3.6}$$

$$k_i = f(t_n + c_i h, y_n + h\sum_{j=1}^{i-1} a_{ij} k_j) \tag{3.7}$$

0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
	1/6	1/3	1/3	1/6

Table 3.1: The Butcher tableau for the classic 4th order Runge–Kutta method

Define b_i, k_i, c_i, a_{ij}, or k_j? Add to list of symbols? DAH

3.2.2.2 Data Interpolation

By default, data used by a mover are interpolated linearly in all dimensions. Time and depth are simple 1-D linear interpolations, whereas horizontal interpolation is bilinear. On triangular grids, simple barycentric interpolation is sufficient. Interpolation on quadrilateral grids (staggered and non-staggered) uses an inverse bilinear interpolation algorithm, which involves computing the mapping function from the unit square to the specific quadrilateral.

3.2.3 Horizontal Diffusion

Horizontal diffusion, or random spreading, is done by a simple random walk with a square unit probability. The random walk is based on the diffusion value D, which is set in the model, and which represents the horizontal eddy diffusivity in the water. A low value would be 1,000 cm^2/s, and a high value would be between 100,000 to 1,000,000 cm^2/s. The model default is 100,000 cm^2/s. During a spill, the value is calibrated based on overflight data.

Diffusion and spreading are treated as stochastic processes in GNOME. Gravitational and surface tension effects are ignored, as these are only important during the first moments of a spill. Complex representation of subgrid diffusion and spreading effects is ignored.

GNOME uses classical diffusion as given in Equations 3.8 and 3.9:

$$\frac{\partial C}{\partial t} = D\nabla^2 C \tag{3.8}$$

where C is the concentration of a material and D is the aforementioned diffusion coefficient, and:

$$\frac{\partial C}{\partial t} = D_x \cdot \frac{\partial^2 C}{\partial x^2} + D_y \cdot \frac{\partial^2 C}{\partial y^2} \tag{3.9}$$

where D_x and D_y are the scalar diffusion coefficients in the x and y directions.

The mean position remains zero, but the variance grows linearly with time. It can be shown that a long series of random steps will converge to a Gaussian distribution with variance growing linearly with time. The precise form of the transition probability distribution is irrelevant as long as its second moment is $2D\Delta t$(Csanady) (reference here?). The transition probability distribution is the distribution of displacements at each random walk step, and D is the diffusion coefficient in the diffusion equation. So, diffusion can be simulated with a random walk with any distribution, with the resulting diffusion coefficient being one-half the variance of the distribution of each step divided by the time step:

Add missing ref. DAH

$$D_x = \frac{1}{2} \cdot \frac{\sigma_x^2}{\Delta t} \tag{3.10}$$

In GNOME we compute a $\Delta x, \Delta y$ from the input diffusion coefficient D, and at each diffusion time step, dx and dy are chosen randomly from a uniform distribution of floating point numbers between -1 and 1, such that $-\Delta x \leq dx \leq \Delta x, -\Delta y \leq dy \leq \Delta y$, and $\Delta x = \Delta y$. This results in a distribution of points spread throughout square. The variance of this distribution is given by Equation 3.11 for σ_x^2, and is computed similarly for σ_y^2:

TJB - square what or of what?

Define σ_x^2 and σ_y^2 as variance of locations of LEs? Add to list of symbols? DAH

$$\sigma_x^2 = \int_{-\Delta x}^{\Delta x} \frac{x^2}{2 \cdot \Delta x} dx = \frac{\Delta x^2}{3} \tag{3.11}$$

Equation 3.12 then follows from 0 (missing reference here?), Equation 7:

Add missing ref. Is "Equation 7" the same as Equation 6.38? DAH

$$\Delta x = \frac{dx \cdot \frac{\sqrt{6 \cdot \frac{D}{10,000} \cdot \Delta t}}{111,120.00024}}{\cos(y)} \Delta y = dy \cdot \frac{\sqrt{6 \cdot \frac{D}{10,000} \cdot \Delta t}}{111,120.00024} \tag{3.12}$$

where:
Δt is the time elapsed between time steps $i+1$ and i,
$\Delta x, \Delta y$ are the 2-D longitude and latitude displacements, respectively, due to diffusion for that time step,
y is the latitude in radians, and
111,120.00024 is the number of m per degree of latitude (assumes 1' latitude = 1 nautical mile everywhere).

3.2.4 Vertical Movers

3.2.4.1 Rise Velocity

For rise velocity from a droplet distribution ($w(\delta_{droplet})$), GNOME uses an algorithm developed by Genwest Systems for the Response Options Calculator (ROC). This algorithm is Stokes' rise for droplets < 0.0002 m (200 µm), form drag for droplets > 0.01 m, and a Hermite cubic interpolation in-between.

The algorithm uses a Hermite cubic that matches the value and first derivative of the Stokes' formulation on one end, and the form drag equation on the other. This gives a smooth and twice differentiable formulation over the gap, which exhibits the Stokes' quadratic droplet dependence on the lower end of the curve, and the form drag square root droplet dependence on the upper end of the range:

$$w(\delta_{droplet}) = \frac{1}{18} \cdot \frac{(\rho_w - \rho_{oil})}{\rho_w \nu_w} g \delta_{droplet}^2 \text{ for } \delta_{droplet} < 0.0002\text{m} \tag{3.13}$$

$$w(\delta_{droplet}) = c_3 \delta_{droplet}^3 + c_2 \delta_{droplet}^2 + c_1 \delta_{droplet} + c_0 \text{ for } 0.0002 < \delta_{droplet} \leq 0.01\text{m} \tag{3.14}$$

and:

$$w(\delta_{droplet}) = \sqrt{\frac{1}{18}(\rho_w - \rho_{oil}) g \delta_{droplet}} \text{ for } \delta_{droplet} > 0.01\text{m} \tag{3.15}$$

where:
$w(\delta_{droplet})$ is the rise velocity from a droplet distribution,
$\rho_w - \rho_{oil}$ is the difference in density between sea water and the oil droplet,
ν_w is the kinematic viscosity of sea water,
g is the gravitational constant,
$\delta_{droplet}$ is the diameter of the droplet, and

the $\{c_3, c_2, c_1, c_0\}$ coefficients in the intermediate size range depend on density.

3.2.4.1.1 Other Approaches Many full blowout models use the method outlined in Zheng and Yapa (2000).

In that paper, the authors outline an approach that has three regimes – the Stokes' rise and form drag as described above, and a third regime for larger droplets referred to as the "spherical cap" solution. For larger bubbles, the droplets distort into a spherical cap shape, which has different drag characteristics than the sphere assumed above. However, this large size is primarily relevant for gas bubbles rather than oil droplets greater than approximately 18 mm in diameter. So the two-regime approach should be fine for oil, though it may need to be revisited if and when GNOME supports gas bubbles.

Perhaps a plot of rise velocity vs droplet size for a few densities would be good here

3.2.4.2 Vertical Diffusion

In GNOME we have implemented a two-layer vertical diffusion algorithm using a random walk algorithm. An issue with the use of a random walk algorithm with a non-constant vertical diffusivity (distinct values above and below the mixed layer) is that it leads to accumulation in low-diffusivity areas, as shown by Visser (1997). This can be addressed by using a random displacement model or a corrected random walk model.

Because we often do not have the profiles of diffusivity required for a random displacement model, for the time being we have implemented a simple two-layer vertical diffusion – above and below the mixed layer depth. The potential of particle accumulation below the mixed layer is accomidated by reflecting particles that move out of the mixed layer back in, and then applying a background diffusivity to all particles to let a few leak through.

3.2.4.2.1 Simple Two-Layer Diffusion Algorithm Description Particles that are in the mixed layer are moved with a specified diffusivity coefficient. Particles are reflected if necessary, i.e. some particles that move above the surface or below the mixed layer depth are reflected back into the mixed layer. If they are reflected all the way back out again, they are placed at $\epsilon = 1e-6$ below the surface, or above the bottom, depending on which way they reflected. In theory, this should not happen unless the mixed layer is very small, or diffusion or time step is very large. A background (representative lower layer) diffusion is then applied to all particles both above and below the mixed layer. This can result in particles moving into or out of the mixed layer, but because it is constant throughout the water column, it does not accumulate particles below the mixed layer.

$$D_z = \sqrt{6 \cdot (D_{ml}/10{,}000) \cdot \Delta t} \tag{3.16}$$

where:
D_z is the vertical diffusion coefficient, and
D_{ml} (in cm^2/s) is the mixed-layer diffusion coefficient.

Define ϵ? DAH

and:

$$\Delta z = rand \cdot D_z \tag{3.17}$$

where *rand* is taken from a uniform random distribution.

3.2.4.2.2 Random Displacement Model The random displacement model approach includes an additional non-random "advective" component from areas of low diffusivity to high diffusivity. Also, the diffusivity is not estimated at the particle location, but is offset by a distance proportional to the gradient of the diffusivity. It is not difficult to implement a random displacement model for vertical diffusion, but it requires a known (or estimated) continuously differentiable diffusion profile with gradients. Recent work ((Nordam et al. 2019)) has looked at diffusion gradients for oil droplet transport in the near surface and this approach could be considered for future development.

3.2.4.2.3 GNOME Results GNOME was run with vertical diffusivity as the only mover, and spills above the mixed layer (Figure 3.1) and below the mixed layer (Figure 3.2). The mixed layer diffusion was 5 cm^2/s, and the background diffusion below the mixed layer was 0.11 cm^2/s, with mixed layer depth of 10 m

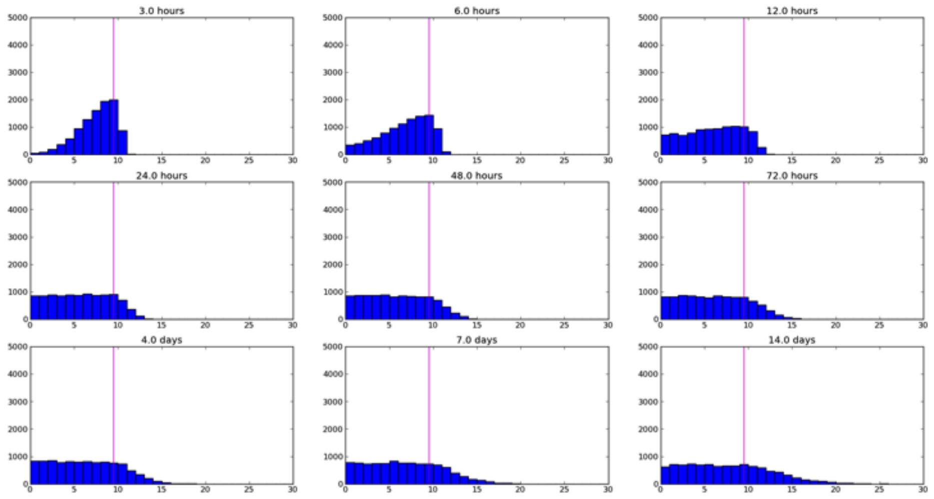


Figure 3.1: A series of histograms of particles in 1-meter bins for a spill above the mixed-layer depth (spill at 9.5 m)

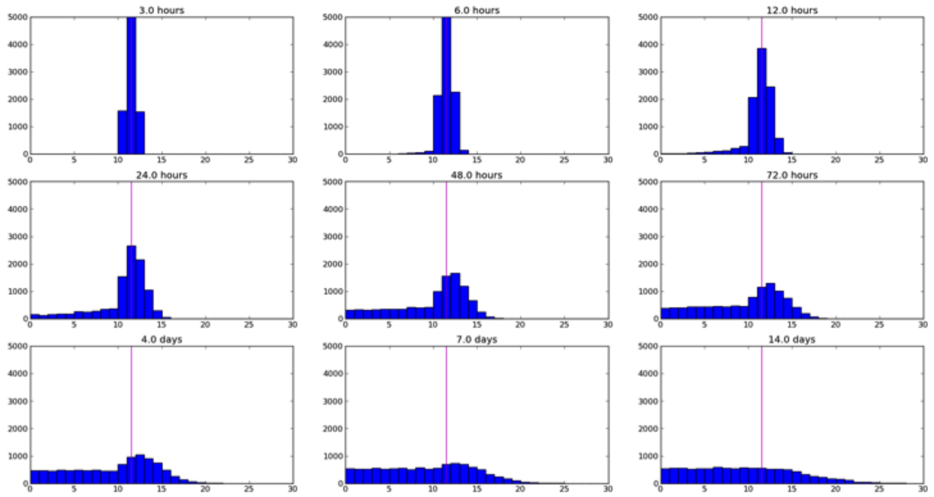


Figure 3.2: A series of histograms of particles in 1-meter bins for a spill below the mixed-layer depth (spill at 10.5 m)

3.2.4.3 Droplet Distribution

GNOME includes a Rosin-Rammler distribution from Sintef's research . This is a two-parameter Weibull distribution with shape $\alpha = 1.8$ and scale $\lambda = 1.22597 \cdot \delta_{50}$. The cumulative distribution function is:

Need ref? TJB

$$1 - exp(-0.693(\frac{d}{\delta_{50}})^{\alpha}) = 1 - exp(-(\frac{d}{\lambda})^{\alpha}) \tag{3.18}$$

where δ_{50} is the volume mean diameter of the oil droplets.

Figure 3.3 shows a histogram of the GNOME-generated droplet size distribution with 10,000 particles, where $\alpha = 1.8$ and $\delta_{50} = 0.0038$ meters ($\lambda = 0.00456$). The x-axis is droplet diameter in centimeters, and the y-axis is number of particles.

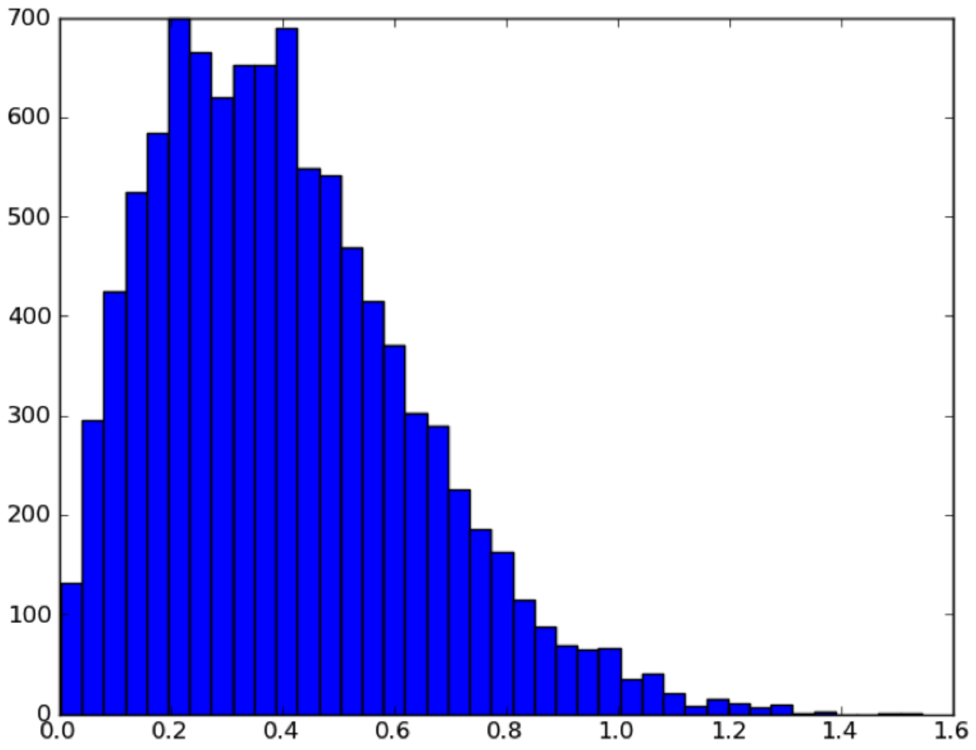


Figure 3.3: Histogram of the GNOME-generated droplet size distribution

There is further explanation of the algorithm in Johansen et al. (2013). The authors present a new model for prediction of initial droplet size distributions in subsea oil and gas releases, which has been derived from experimental data.

GNOME also has options for uniform, normal, and lognormal distributions.

3.2.4.3.1 Droplet Distribution – Other Approaches Michel Boufadel has developed a "comprehensive" numerical model, Vdrop, capable of simulating the transient droplet size distribution in turbulent regimes. This may be used in future versions to prpovide refined DSDs for subsurface releases.

TJB-need ref?

3.2.5 Transport Uncertainty

Forecast winds and currents, whether from a model or human forecaster always have a level of uncertainty. In operational oil spill forecasting, it's been found that they are usually not accurate enough to generate trajectories within 1 mile of accuracy after 48 hours (Galt 1999). In order to accommodate this inevitable uncertainty, GNOME supports user-specified uncertainty parameters, which are set according to the uncertainty in the input data. In operational use, input data are constantly updated during a spill event, and the model is rerun and re-calibrated at least once a day.

If the model uncertainty is turned on, a second solution will be calculated (and shown in red). This creates a second set of elements that move under the influence of the active movers. IN this case each ”uncertainty” element responds to a different time series of forcing, resulting in a different trajectory of that element. This results in a spreading of the elements, essentially a structured diffusion. However, as each element is independent, the concentration of the elements is no longer meaningful – the uncertainly elements are used to determine possible bounds of the transport of the oil, but not areas of relatively high or low concentration. All of the movers have default uncertainty parameters – diffusion, currents, winds.

Diffusion has a simple uncertainty in which the uncertainty LEs move with a different random spreading (a multiplicative factor of the user set value), and the random step is increased as the square root of the uncertainty factor (see Section 3.2.3 on Horizontal Diffusion).

The various current formats and winds all have parameters for start time and duration of the uncertainty. The uncertainty start time specifies the time into the model run that the wind or currents become uncertain. For example, if you are hindcasting and have observations up to a certain point and then a forecast, you may want the wind uncertainty to begin at the time the forecast starts. The duration indicates how long an element will have that particular uncertainty value before the uncertainty value is randomly reset. We rarely find it necessary to change the default duration, except when modeling large object drift. The currents, including component mover patterns, have cross- and along-current uncertainties, which capture variations in speed and direction.

The uncertainty angle scale and speed scale are parameterized for the wind uncertainty. The parameterization is log-normal in speed (because forecasters are more likely to over-predict the winds than under-predict them, since they are considering safety issues) and Gaussian in angle. Operationally, the defaults are usually used for wind uncertainty. Speed and angle scales relate to the expected error in wind speeds and wind forecast directions, respectively. The best way to examine how the uncertainty solution is calculated for each mover is to set a simple point-source spill and run the model with a single mover active at a time, while adjusting the uncertainty parameters.

3.2.5.1 General Model Inputs

Start time Time in model run when velocities (wind, current) become uncertain.

Duration: Time duration before resetting an LE's uncertainty parameter.

So, for example, if the trajectory of a particular LE is going a little faster and to the right of the wind, it will stay that way for the entirety of the given duration. Then, once that user-input duration has elapsed, it will be randomly assigned a different relative uncertainty value. Default values (based on experience) are 3 hours for wind and 48 hours for currents.

3.2.5.2 Currents

The current pattern uncertainty is a combination of two types of uncertainty: uncertainty in the currents and uncertainty from eddy mixing. The uncertainty in the currents is parameterized by four scales, which represent the percentage of the given velocity that the uncertainty spans in the parallel and normal directions: α_f and α_b in the forward and backward directions (along current), and β_l and β_r in the left and right directions (cross current), respectively. These coefficients are expressed as percentages of the given speed $\parallel \vec{v} \parallel$. The eddy scale γ is determined by Equation 3.19 using a separate uncertainty diffusion coefficient D, usually 0 (≤ 106 cm^2/s). The eddy scale is used only for steady state, tidally driven patterns such as CATS patterns.

$$\gamma_D = \frac{a_{Un} \cdot v_0}{v_0 + \parallel \vec{v} \parallel} \tag{3.19}$$

where:
$\parallel \vec{v} \parallel$ is the magnitude of the original current velocity vector,

v is a small velocity scale (0.1 m/s), and
$a_{Un} = \sqrt{6 \cdot D_{Un} \cdot \delta t}$ is the random walk length for the uncertainty diffusion coefficient D.

To get the random walk length for this eddy scale, a random number between zero and one, $R_{(0,1)}$, is multiplied by the eddy scale γ. This eddy scale is a maximum when $\vec{v} = 0$ and decreases quickly as $\| \vec{v} \|$ increases. Only CATS current patterns include the eddy uncertainty.

3.2.5.3 Currents – Model Inputs

Down-Current Uncertainty: forward ($\alpha_f > 0$) and backward ($\alpha_b < 0$) percentages of the velocity, which make up the down-current uncertainty range. The default value is 30% for CATS patterns. It is a required input for other model data; 50% is a reasonable value to use.

Cross-Current Uncertainty: left ($\beta_l < 0$) and right ($\beta_r > 0$) percentages of the velocity, which are used in the direction perpendicular to the velocity to make up the cross-current uncertainty range. The default value is 10% for CATS patterns. It is a required input for other model data; 25% is a reasonable value to use. For shallow water, the uncertainty in direction will be less than that for deep water.

Eddy Diffusivity Coverage: uncertainty diffusion value D, which simulates eddy mixing processes dominant during slack water.

3.2.5.4 Currents – Mover Outputs

The Current Uncertainty outputs are the displacements over one time step in the x and y directions, calculated with Equation 3.20 for $\| \vec{v} \| \geq 10^{-6}$m/s and Equation 3.21 for $\| \vec{v} \| < 10^{-6}$m/s. (Or $dx = u \cdot (1+\alpha) + \beta v$ and $dy = v \cdot (1+\alpha) - \beta u$ in standard notation.)

$$\Delta\vec{X} = (1 + R_{(\alpha_b,\alpha_f)} + R_{(0,1)} \cdot \gamma_D) \cdot \vec{v} + (1 + R_{(\beta_l,\beta_r)} + R_{(0,1)} \cdot \gamma_D) \cdot \vec{v}_\perp \tag{3.20}$$

Define u and $\vec{v}$? DAH

where:
$\Delta\vec{X}$ is the vector displacement ($\Delta X\hat{\imath} + \Delta Y\hat{\jmath}$),
$\vec{v}_\perp$ is a vector of the same magnitude but in an orthogonal direction to $\vec{v}$, and
$R_{(n,m)}$ is a random number with uniform probability on the interval n, m.

$$\Delta X = R_{(0,1)} \cdot \gamma_D \hat{\imath} \Delta Y = R_{(0,1)} \cdot \gamma_D \hat{\jmath} \tag{3.21}$$

3.2.5.5 Wind – Model Inputs

Speed Scale: measure of uncertainty in the wind speed; the default is 2. (unit-less)

Angle Scale: measure of uncertainty in the wind direction; the default is 0.4 radians.

3.2.5.6 Wind – Mover Outputs

The Wind Uncertainty outputs are displacements over one time step in the x and y directions. $\| \vec{v} \|$ is the magnitude of the original wind velocity vector, and for $\| \vec{v} \| < 10^{-6}$ m/s, the uncertainty displacement is zero. For $\| \vec{v} \| \geq 10^{-6}$ m/s, the vector uncertainty displacement is given by:

$$\Delta\vec{X} = [1 + \cos d\theta \cdot \vec{v} + \sin d\theta \cdot \vec{v}_\perp] \tag{3.22}$$

where:
$\Delta\vec{X}$ is the vector displacement ($\Delta X\hat{\imath} + \Delta Y\hat{\jmath}$),
$\vec{v}_\perp$ is a vector of the same magnitude but in an orthogonal direction to $\vec{v}$, and:

$$d\theta = \sigma_{dir} \cdot \frac{\pi}{180} \cdot \sin(2 \cdot \pi \cdot R_{(0,1)}) \cdot \sqrt{-2 \cdot ln(R_{(>0,<1)})} \tag{3.23}$$

where $R_{(n,m)}$ is a random number with uniform probability on the interval n, m.

The uncertainty given by Equation 3.23 is scaled to have a norm, w, with a minimum of 0.001 to ensure $w > 0$ (Equation 3.24). [Note that here our notation breaks with convention; w is not the vertical component of velocity, nor is x zonal displacement.]

Units? DAH

$$w = \frac{x^2}{\cos(d\theta)} \tag{3.24}$$

where:

$$x = s \cdot \cos(2 \cdot \pi \cdot R_{(0,1)}) \cdot \sqrt{-2 \cdot ln(R_{(>0,<1)})} + m$$

$$s = \sqrt{\| \vec{v} \|^2 - \sqrt{s_1}} \text{ and}$$

$$s_1 = \| \vec{v} \|^2 - \sigma_{speed}$$

If $s_1 > 0$, $m = \sqrt[4]{s_1}$, otherwise $m = 0$.

The standard deviations for speed and angle are updated at every time step. There is also a maximum angle scale of 60°. The random variables $R_{(n,m)}$ in the uncertainty calculation are updated when the user-set duration is exceeded.

Research and development of new wind uncertainty algorithms remains a continuing process within NOAA/ERD (Lehr et al. 1999).

3.2.5.7 Display of Uncertainty Results

Once the map, movers, and spills are set, the model is run and the solution is produced in the form of a trajectory forecast map. If uncertainty is activated, GNOME provides two solutions to an oil spill scenario:

1. A "best estimate," or forecast trajectory, and
2. A "minimum regret," or uncertainty trajectory.

The first solution ("best estimate") shows the model result with all of the input data assumed to be correct. However, models, observations, and forecasts are rarely perfect. Consequently, in GNOME we have incorporated our understanding of the uncertainties (such as variations in the wind or currents) that can occur. This second solution ("minimum regret") allows the model to predict other possible trajectories that are less likely to occur, but which may have higher associated risks. We call the trajectory that incorporates these uncertainties the "minimum regret" solution because it gives you information about areas that could be impacted if, for example, the wind blows from a somewhat different direction than you have specified, or if the currents in the area flow somewhat faster or slower than expected. In some cases, the areas within the minimum regret solutions might be especially valuable or sensitive to oiling.

Both trajectories are represented by Lagrangian elements, which are statistically independent pieces of the modeled pollutant. They appear as small "pollutant particles" on a map when you run your spill. The "best estimate" trajectory is represented by black "splots"; the "minimum regret" trajectory is represented by red "splots".

Interpreting the visualization of the individual elements can sometimes be deceptive: "how much oil does an individual dot represent?". IN order to provide more clear forecast products, the individual elements are often post-processed to produce a map with contours for high, medium and low surface concentrations, along with an uncertainly bound.

add more description and examples here

3.3 Time Step Calculations and LE Number

Bill 3-14-18: "Section 3.3 should be deleted and replaced with what-

Chapter 4

Framework for Near-Surface Processes

4.1 Background

We often refer to oil on the surface of the sea as a “slick” or “sheen” because the reduction in interfacial tension suppresses capillary waves and makes the surface of the sea look smooth or reflective. Following this observation, modelers refer to the surface “film” of oil, and develop algorithms that assume the oil is on the surface and has properties like surface area and thickness, e.g., see Section 6.3 [External references?]

Add refs? DAH

However, the situation in a real sea state in other than calm conditions is quite a bit more complex. Real sea states are characterized by many waves of varying height, period and phase, and often with breaking or white capping in stronger winds. Under these conditions, the oil at the surface is not really a continuous film.

This complication is considered in traditional modeling of the dispersion process: The turbulence of the waves, primarily white capping, is considered to break the oil film into droplets, the smallest of which do not rise to the surface and are thus “dispersed.” The original work by Delvigne and Sweeney (1988)[other citations here] is still used as the basis for most models. There has been more recent work expanding on the approach [cite more here: Zeinstra-Helfrich, Johanson, Boufadel, etc...], but the basic framework is the same.

Add refs. DAH

Add refs. DAH

Even for just the consideration of the dispersion process, there were some issues from the beginning. In the original work by Delvigne and Sweeney (1988), two sets of experiments were conducted: determining droplet size distributions as a function of energy dissipation rate in a “grid column,” and an empirical determination of oil entrainment rate from breaking waves in wave flumes. One key finding from the grid column experiments was that under a given level of turbulence, it took time for a “mature” droplet size distribution to develop. This timescale was much longer (> 5 min) than a typical wave period (on the order of seconds). In the flume experiments, they developed an entrainment rate per wave event, and tied this to white cap fraction. Some empirical measures of white cap fraction provide a spatial assessment, i.e., the fraction of the sea surface that is white, and not the fraction of time a given location is experiencing a white cap. There is also microscale wave breaking (Banner and Phillips 1974) that may drive dispersion well below the wind speeds at which white caps appear.

This approach has proved to give reasonable results for operational modeling, but there is a core inconsistency between oil as droplets and oil as a surface film. In GNOME, we have attempted to provide a framework for considering the film processes vs. droplet processes.

4.1.1 Simplification of the Sea State

In a real sea state, the level of turbulence is highly variable, and many processes are not simply linear with average turbulence, but are time dependent. The highest level of turbulence is driven by white-capping (breaking of the waves). As a gross simplification, the near surface is fairly calm in-between white caps, and experiences a large but brief burst of turbulence as a white cap passes by.

The oil "slick" is more-or-less stationary, with waves passing through it. So, the oil is fairly calm, and periodically a white cap passes through, providing a large burst of turbulence. This burst of turbulence breaks the oil into droplets, and pushes them under the water. After the white cap passes, the energy level rapidly drops, and the droplets rise to the surface. How fast they rise depends on droplet size and the densities of oil and water.

The periodicity of these white capping events is on the order of wave period – i.e., perhaps tens of seconds, whereas the duration of the turbulent event is typically a fraction of a second.

Due to the full spectrum of periods and wave heights in the real sea state, we seek a simple parameterization to create a manageable level of complexity. We make the extreme simplification of a single periodicity of white-cap events, and a single period and energy level (Figure 4.1). This allows a model with turbulence events as simple spikes of turbulence at some known periodicity.

In the figure, white cap is misspelled and T_wc should be t_{wc}. DAH

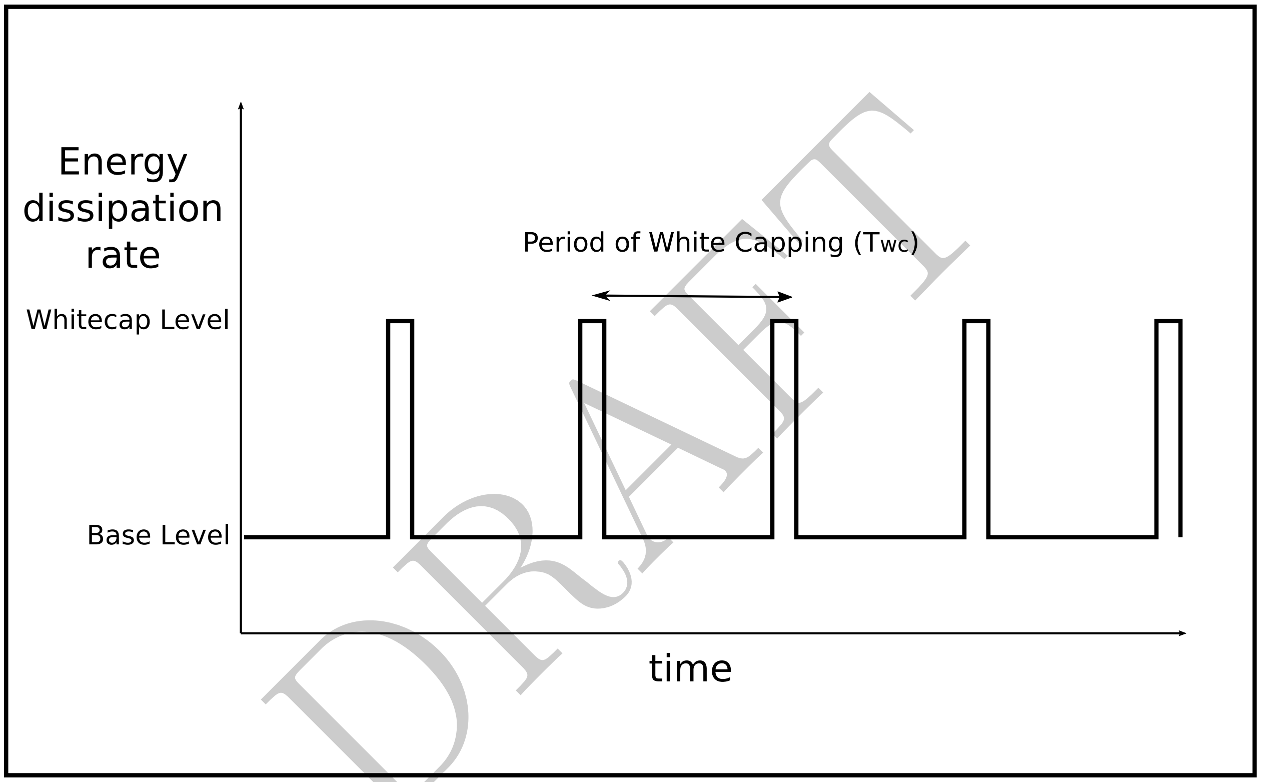


Figure 4.1: Schematic of white-capping energy time series

Under this theoretical time-dependent turbulence regime, the oil is broken into droplets and entrained under the water surface at each spike of turbulence, and then rises back to the surface in-between events. How long it takes to rise to the surface depends on the droplet size and how deep it was pushed. Droplet size is a function of the strength (and duration) of the white capping event, and depth is a function of the wave height. So for a given wave climate, we can compute the amount of time a "representative" droplet spends under water. If that is less than the periodicity of the white capping events (t_{wc}), then it spends some fraction of time under water in droplet form, and the rest of the time on the surface as a slick.

Typically, the model time step is far longer than the wave period (15 min vs. approximately 10 sec). This ratio is used to compute the time spent in underwater vs. surface processes. Underwater processes include dissolution, biodegradation, and oil-sediment aggregation, whereas evaporation and possible future photo-oxidation are surface processes.

This framework requires the following somewhat independent modules.

4.1.1.0.1 Wave Module In order to compute the near-surface processes, we need the wave climate, which is characterized by:

- mean wave period
- significant wave height
- periodicity of white capping
- maximum energy dissipation rate of white capping
- average energy dissipation rate

In practice, we get all of this information from a simple wind-wave model, but we could also get it from an operational wave model or a more sophisticated inline model.

4.1.1.0.2 Droplet Module The droplet formation module needs to compute the droplet size distribution as a function of the maximum energy dissipation rate of white capping, and perhaps the average energy dissipation rate, as well as the oil properties: density (maybe), viscosity, and water-oil interfacial tension.

4.1.1.0.3 Refloating Module The refloating module computes the time it takes for a representative oil droplet to rise to the surface. This is a function of the droplet size distribution and the densities of oil and water.

4.1.1.0.4 Dispersion Module The dispersion module computes how much oil has been broken into tiny droplets that are not expected to ever rise to the surface again. These are generally droplets smaller than a certain threshold size, which theoretically depends on density. This will be the fraction of oil in the droplet size distribution that is smaller than that threshold, multiplied by the mass of a given element, and by the number of white capping events in the time step.

Chaining these modules together results in a fraction of time that the oil is spending under water, and the representative droplet size while under water. The subsurface algorithms include dissolution, biodegradation, Oil Sediment Aggregation (OSA), and perhaps Marine Oil Snow Sedimentation and Flocculent Accumulation (MOSSFA) formation.

4.2 Wave Module

Bring in Bill's new stuff.

As presented in Section 4.1, all the near-surface processes require a parameterization of the wave climate. In future versions, GNOME may be able to use wave climate derived from operational wave models or in situ wave measurements, but in this version, the wave climate is derived from the winds provided for other aspects of the model.

Wind-generated waves depend on the duration and strength of the wind and on the fetch (distance traveled by the wind across open water). If the wave heights are not entered by the user, the wave height is estimated from the fetch and wind speed. Wind duration is not considered in the wave height calculations. However, a moving average of the wind speed is used to compute the wave climate, so that very short-term changes in wind speed will not destabilize the calculations. If there has been a rapid and large change in the wind vector, such as in the passing of a storm front, the user may wish to enter wave heights from field observations instead of relying on the default calculations in the model.

GNOME uses a modification of the formula for estimating wave heights from surface winds as suggested in the U.S. Army Corps of Engineers Short Protection Manual Coastal Engineering Research Center (1984). The manual and GNOME assume that the wind speed entered by the user refers to the wind speed at ten meters above the water surface (U_{10}). If the wind speed entered was observed at a different height, the wind can be scaled to the ten-meter wind speed by the formula:

$$U_{10} = U_z \cdot \left(\frac{10}{z}\right)^{1/2} \tag{4.1}$$

for heights less than 20 meters, where z is the height at which U_z was measured.

This wind speed is used to compute a wind stress factor (τ_{wind}) in m/s by:

$$\tau_{wind} = 0.71 \cdot (U_{10})^{1.23} \tag{4.2}$$

4.2.1 Wave Height – Fetch Limited Case

The fetch limited case reduces to the fetch unlimited case whenever the fetch (F) in m is such that:

$$1.63 \cdot 10^{-3} \cdot \frac{\sqrt{g \cdot F}}{\tau_{wind}} > 0.243 \tag{4.3}$$

Otherwise, for the fetch limited case, the significant wave height ($H_{1/2}$) in m is calculated by:

$$H_{1/2} = 5.112 \cdot 10^{-4} \cdot \tau_{wind} \cdot \sqrt{F} \tag{4.4}$$

4.2.2 Wave Height – Fetch Unlimited Case

For the fetch unlimited case, the significant wave height ($U_{1/3}$) in m is estimated by:

$$U_{1/3} = 0.0248 \cdot \tau_{wind}^2 \tag{4.5}$$

Duration is not used in the calculations. Instead, the program uses a three-hour moving average of the wind speed, and assumes a constant direction. For short-duration strong winds in the fetch unlimited case, this may cause a substantial overestimate of wave height, since it typically takes twelve hours or longer for a fully developed sea to arise.

This overestimate is adjusted for by treating wind speeds of greater than 32 knots as a fetch limited case, with a fetch of 1000 km, even if the user specifies fetch unlimited. This yields wave height values that are closer to the graphical data reported in the literature (Darbyshire and Draper 1963).

4.2.3 Wave Period

The wave periods also need to be estimated. The period of the peak of the wave spectrum (τ_{peak}) in sec is given by:

$$\begin{array}{ll} \tau_{peak} = 0.06238 \cdot (\tau_{wind} \cdot F)^{1/3} & \text{for the fetch limited case} \\ \tau_{peak} = 0.83 \cdot \tau_{wind} & \text{for the fetch unlimited case} \end{array}$$

Add fraction wave breaking, etc here!

Chapter 5

Oil Properties

5.1 Bulk Properties

Initially, the physical properties of spilled oil are defined by the original data in a oil record. However, these properties change as the oil weathers, and are tracked and re-calculated as the model is run.

5.1.1 Density

Density is an intrinsic property of a material. However, the density of an oil changes with temperature, and as the oil weathers, the density changes as well. In general, as the volatile fractions of an oil evaporate, the oil becomes more dense.

The density (ρ_{oil}) of each Lagrangian element is tracked as the model runs, and a mass-weighted average density is compiled for the whole oil.

5.1.1.1 Density as a Function of Temperature

As with most substances, the density of an oil varies with its temperature. The change is a non-linear function of temperature (American Society for Testing and Materials 2007). Fortunately, the correction is small and approximately linear over the temperature range of concern for oil spill modeling (0 – 30°C), so it can be described with a coefficient of thermal expansion. An ideal oil record will have multiple measured densities at various temperatures. In this case, the density at a given temperature is computed by linear interpolation. However, if outside the range of values reported, or if there is only one reported value, a coefficient of expansion is assumed, and the density can be computed with:

$$\rho_{oil} = (1 - k_{p1} (T - T_{ref})) \rho_{ref} \tag{5.1}$$

where:
k_{p1} is a coefficient of isobaric thermal expansion,
ρ_{oil} is the density of the oil, and
ρ_{ref} is the density of the oil at a reference temperature.

The value of 0.0008 was used by the NOAA ADIOS2 model (Lehr 2001), but Jones (2010) notes that less dense petroleum products such as gasoline have a higher k_{p1} around 0.00095. Therefore, the current model uses:

$$k_{p1} = 0.0008 \text{ for products with an initial API} < 30$$
$$k_{p1} = 0.0009 \text{ for products with an initial API} \geq 30$$

this creates an discontinuity – not a big deal obviously, but should we smooth that out?

5.1.2 Density Estimations

As oils' densities change with time due to weathering, GNOME more carefully estimates density because an oil may weather to non-buoyancy. Each pseudocomponent is assigned a density and initial mass fraction. (See Section 6.2)

$$
\begin{aligned}
dc &= \text{index of distillation cut}\\
\rho_{dc} &= \text{fixed density of this cut}\\
&\quad\text{(does not change as oil weathers)}\\
f_{dc} &= \text{mass fraction of cut } dc\\
&\quad\text{(varies with time, but the sum always equals 1)}
\end{aligned}
$$

Oil density for each element is a function of fraction evaporated/dissolved and, for surface elements, entrained water (if an emulsion has formed). Oil-sediment aggregates are not further tracked in the current model, so that sedimentation is simply considered a natural removal process.

The change in pure oil density as the oil weathers is given by:

$$\rho_{oil}(t) = \left[\sum_{dc} \rho_{dc} \cdot f_{dc}(t)\right] \tag{5.2}$$

and for (surface) water-in-oil emulsion:

$$\rho_{emul} = f_w \cdot \rho_w + (1 - f_w) \cdot \rho_{oil} \tag{5.3}$$

As a bulk property, density is computed for each element at each time step, after adjustments for mass loss. The overall slick density is computed as a mass average of all element densities.

5.1.3 Viscosity

5.1.3.1 General Description

The viscosity of each individual element is tracked as the oil weathers as the model runs, and a mass-weighted average viscosity is compiled for the whole slick.

Kinematic viscosity has units of area per time. The model uses kinematic viscosity units of m^2/s. For final display, other units may be used, for instance centistokes (cSt) with $1\,\mathrm{m}^2/\mathrm{s} = 10^6\,\mathrm{cSt}$.

A related quantity is dynamic viscosity, which is kinematic viscosity multiplied by the oil density. Its default display unit is centipoise. Fresh water has kinematic and dynamic viscosity of approximately one centistokes or one centipoise. Oil and refined products will have kinematic viscosities of between a few centistokes to more than a million centistokes. Oils that have very large viscosities cease to behave as Newtonian fluids and may not be accurately described by the GNOME algorithms.

There seems to have been little recent productive research in viscosity alterations due to weathering. Published literature mostly refers to the formulas used by ADIOS2 However, industry empirical fits for viscosity change in pipelines due to emulsification (Schramm 1992) are well known and used in GNOME.

Viscosity plays a major role in natural and chemical dispersion estimates and a lesser role in mechanical cleanup. While it should significantly affect oil spreading, the standard spread models do not depend upon viscosity. However, viscosity does play a role in GNOME's "terminal thickness" for spreading. It should be noted that viscosity will increase due to evaporation, there are much larger changes if the oil emulsifies. An usually most of the spreading has already occurred before emulsification takes place.

CHB: maybe we should turn off spreading at a viscosity threshold?

Viscosity is recomputed for each element at each time step, after adjustments for mass loss and emulsification.

5.1.4 Formulas Used

GNOME uses the approach of ADIOS2, itself based upon the work of MacKay, Shiu, Hossain, Stiver, McCurdy, and Paterson (1982).

5.1.4.1 Temperature Adjustment

Temperature has a strong effect on the viscosity of the oil: increased temperature results in lower viscosity. This effect is strong enough, even at environmentally relevant temperatures, to require a correction when modeling oil behavior.

It has been observed that the decrease in viscosity with temperature follows an exponential decay (derived from Eyring's equation (Eyring 1936)), such that it can be modeled as:

$$\nu_0 = a \cdot e^{\left(\frac{k_{\nu 2}}{T}\right)} \tag{5.4}$$

with:

$$a = \left(\nu \cdot e^{\frac{-k_{\nu 2}}{T}}\right) \tag{5.5}$$

where ν_0 is the viscosity of fresh oil, a is the initial constant (the theoretical value at infinite temperature – a mathematical construct in this case), and $k_{\nu 2}$ is the decay constant.
with:

$$0 < \nu_0 \leq \nu_{max}$$

As there are two constants in this formulation, they can be determined from two viscosities measured at two different temperatures:

$$k_{\nu 2} = \frac{\ln(\nu_1) - \ln(\nu_2)}{\frac{1}{T_1} - \frac{1}{T_2}} \tag{5.6}$$

where ν_1 and ν_2 are kinematic viscosities measured at temperatures T_1 and T_2. a can then be computed from eq. 5.5
Temperatures are in K, and kinematic viscosity in m^2/s. Note that 1 $m^2/s = 10^6$ cSt.

Note that ideally the two temperatures should span an environmentally relevant range: approx. 0 °C to 30 °C. If they are far from that range, extrapolation errors may occur.

If there are viscosity data available at more than two temperatures, the two constants are determined by a least squares fit to the available data. Only data points at temperatures near environmentally relevant ranges are used for the least squares fit.

If there is only one viscosity measurement at a single temperature, then both constants cannot be determined, and an estimate is used for the decay constant: $k_{\nu 2}$, with a being computed from the single viscosity measurement.

A number of estimates for $k_{\nu 2}$ have been provided in the literature, including:

- $k_{\nu 2} = 9000\,\text{K}$ (Payne, Kirstein, McNabb, Lambach, Redding, Jordan, Hom, DeOliveira, Smith, Baxter, and Gaegel 1984)
- $k_{\nu 2} = 5000$ K (Bobra and Callaghan 1990)
- $k_{\nu 2} = exp(5.471 + 0.00342 \cdot T_{b50})$ (Abu-Eishah 1999)

where T_{b50} is the temperature (in K) at which 50% of the oil mass fraction would boil. These data are available from the distillation cuts. For example, for Arabian medium crude:

$$T_{b50} = 678 \text{ K}$$
$$k_{\nu 2} = 2416 \text{ K(Abu-Eishah 1999)}$$

NOAA's oil library (as of 2018) has roughly 250 oils that contain multiple kinematic viscosity measurements, and it is possible for us to determine the distribution of $k_{\nu 2}$ for this set (Figure 5.1).

Test and maybe implement the Abu-Eishah:1999 approach: there are currently 254 records that have one viscosity

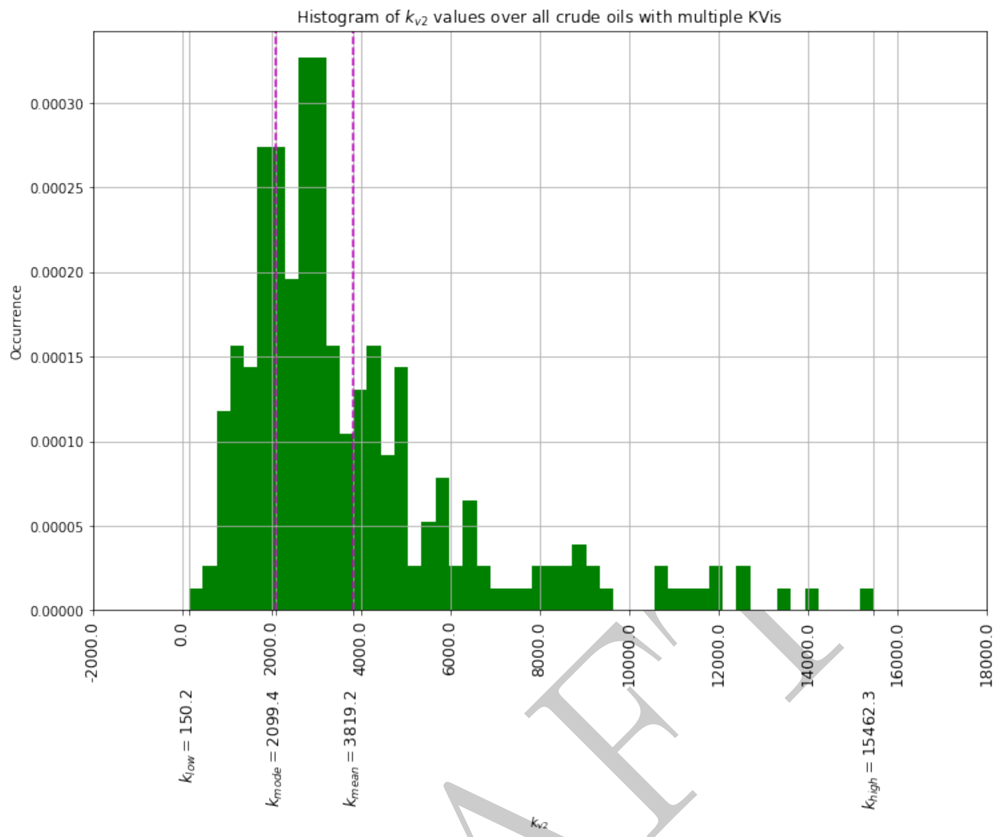


Figure 5.1: Distribution of estimable $k_{\nu 2}$ values in the oil library

A histogram of these values indicates a skewed distribution. The mean is about 3820 K and the log-normal mode is roughly 2100 K. This gives some indication of the numeric range we should expect, but we should also consider that our sample size may not be large enough to give a "true" value. Based on these data, the mode value: 2100 K is used as a default if no distillation data are available.

check these values with all the data in the new database – and maybe check for refined vs crude or other correlations (with API maybe?)

5.1.4.2 Evaporation, Dissolution, and Emulsification Adjustment

Evaporation, dissolution, and emulsification increase viscosity. For surface spills, only evaporation and emulsification play an important role. For deep subsurface spills, dissolution acts similarly to evaporation.

The current emulsification model does not allow de-watering of the emulsion; the computed weathered viscosity for an individual element must monotonically increase over time. Viscosity is the result of two effects: an exponential increase due to fraction evaporated or dissolved, and an emulsion increase based on the Mooney equation (Mooney 1951).

$$\nu = \nu_0 \cdot exp(k_{\nu 1} \cdot f_{evap}) \cdot exp\left(\frac{k_{\nu 3} \cdot f_w}{1 - k_{\nu 4} f_w}\right) \tag{5.7}$$

where f_{evap} is defined as the fraction lost due to evaporation/dissolution, and f_w is the water content fraction of the emulsion.

MacKay, Shiu, Hossain, Stiver, McCurdy, and Paterson (1982) and Reed (1989) used fixed values between 1–10 for $k_{\nu 1}$, depending on the type of oil. Based upon the measurement results of eight Environment Canada artificially weathered oils, ADIOS2 used a fractional curve fit for $k_{\nu 1}$:

$$k_{\nu 1} = k_{\nu 5} \cdot v_0 \tag{5.8}$$

Define $k_{\nu 1}$, $k_{\nu 3}$, $k_{\nu 4}$, and $k_{\nu 5}$. DAH

with:
$1 \le k_{\nu 1} \le 10$ and
$k_{\nu 5} = 1.5 \cdot 10^3 \ \sqrt{\text{sec}}/\text{m}$.

GNOME modifies the emulsion increase term. Use the Mooney equation to determine a water fraction that increases viscosity by a factor of 100, estimated to be f_{ref}=0.84. Pal and Rhodes (1989) suggest that a better correlation than the Mooney equation is given by:

Missing ref? DAH

Ref missing in bib file. DAH

$$\nu = \nu_0 \cdot exp(k_{\nu 1} \cdot f_{evap}) \cdot \left(1 + \frac{f_w/f_{ref}}{1.187 - f_w/f_{ref}}\right)^{2.49} \tag{5.9}$$

Their data (Figure 15 in the cited article), which included non-Newtonian oils, show a good curve fit for a wide range of water fractions up to the ADIOS3 water fraction limit of 0.9.

Define f_{ref}? DAH

TJB-GNOME?

5.1.5 Computing Fraction Evaporated/Dissolved

LE mass loss due to dissolution and/or evaporation occurs at different rates for the various pseudocomponents and, for dissolution, according to hydrocarbon structure. The greater loss of the lighter components increases the viscosity of the remaining oil. However, mass loss from the LE can also occur due to dispersion and cleanup. We define mass fraction lost to all processes (f_{lost}) as one minus the ratio of the present LE mass to its initial mass:

More rapid? DAH

$$f_{lost} = 1 - \frac{m_{LE}}{m_{LE}(t_0)} = f_{evap} + f_{other} \tag{5.10}$$

with the last equality recognizing that this fraction has two components: the fraction lost to evaporation/dissolution (f_{evap}) and the fraction lost to other processes (f_{other}).

The LE asphaltene mass (m_{asph}) and resin mass (m_{res}) do not change due to evaporation or dissolution. Therefore, f_{other} can be estimated by:

$$f_{other} = 1 - \left(\frac{m_{res} + m_{asph}}{m_{res}(t_0) + m_{asph}(t_0)}\right) = 1 - \left(\frac{m_{LE}}{m_{LE}(t_0)} \cdot \frac{f_{res} + f_{asph}}{f_{res}(t_0) + f_{asph}(t_0)}\right) \tag{5.11}$$

and:

$$f_{evap} = f_{lost} - f_{other} \tag{5.12}$$

or:

$$f_{evap} = \frac{m_{LE}}{m_{LE}(t_0)} \cdot \left(\left(\frac{f_{res} + f_{asph}}{f_{res}(t_0) + f_{asph}(t_0)}\right) - 1\right) \tag{5.13}$$

NEED SOMEONE TO CHECK MY ALGEBRA

WILL NEED MODIFICATION FOR OILS THAT DO NOT HAVE ASPHALTENES OR RESINS - WILL REVISE

5.1.6 Average Slick Viscosity

Slick viscosity is a mass-weighted average of the element viscosities.

$$\nu = \left[\frac{\sum_{N_{LE}}(m_{LE} \cdot \nu_{LE})}{\sum_{N_{LE}} m_{LE}}\right] \tag{5.14}$$

5.2 Water Fraction

This belongs in the Emulsification section – unless this about vis-

Chapter 6

Weathering

6.1 Spreading

From observations of oil behavior on water, and common sense, spilled oil clearly spreads out on the sea, becoming thinner and covering a wide area. In the oil spill modeling literature, spreading is often defined as one of the core processes that must be modeled in order to capture the oil's weathering and transport. However, as a driver of the weathering processes, it is not so much the spreading (or thickness) that is required, but an estimation of the exposed surface area for processes, most notably evaporation. When reading the literature, one must be careful to understand what a particular study is defining as "spreading" or "thickness" or "area of the slick."

Observations of real oil spills consistently indicate that the oil on the surface is patchy, often with large areas of thin sheen and much smaller areas of thicker, darker, oil in patches, streamers, windrows, and convergence lines. A rule of thumb is that 90% of the oil is in 10% of the area (i.e., most of the oil is in the thick, dark patches). These are also the patches that result in the largest impacts, and are most amenable to clean-up measures.

Due to these observations, GNOME's model of the spreading process seeks to capture a reasonable estimate of exposed surface area for the bulk of the oil (i.e., the parts of the oil slick that are at least 100 µm thick). The assumption is that this represents the majority of the mass of the slick, and oil in very thin sheens can be neglected. Because of this assumption, GNOME's "area" predictions are not intended to match the observations of the full extent of the sheen from overflight or satellite sensors. The "spread" of a slick as defined in Elliott (1986) is referring to the spread of an entire slick, including the thicker parts near the downwind end of the slick, and the thin sheen toward the upwind end. This does not provide what is needed to determine weathering processes, such as evaporation, unless an accurate determination of the distribution of oil thickness is also made.

6.1.1 Physics of Spreading

Over the years, there have been a number of processes identified in the scientific community as drivers of spreading.

6.1.1.1 Fay Spreading

The seminal work on spreading is Fay (1971). In that work, Fay identified three phases of spreading.

1. Gravity-inertial spreading: The initial "blob" of oil slumps and spreads out due to the force of gravity, and it is supposed that inertia is the primary opposition to the buoyancy force imbalance driven by gravity.

2. Gravity-viscous spreading: The buoyancy force is primarily balanced by the viscosity of the water underlying the surface layer.

3. Interfacial-tension spreading: The balance between the air-water interfacial tension and the oil-water and oil-air interfacial tensions continue to drive the oil to spread into extremely thin sheens.

Fay spreading captures the physics appropriate to a simple film of oil on a quiescent sea (with a clean surface – i.e. no natural surfactants). However, in the real ocean, there are a number of other forces that serve to spread out (and sometimes coalesce) an oil slick. The primary forces are wind spreading, turbulent diffusion, and sub-mesoscale circulations.

6.1.1.2 Wind Spreading

It has long been observed that oil slicks tend to spread out in the direction of the wind, with thicker oil being observed at the downwind end, and thinner sheens upwind. This is sometimes referred to as the "comet effect". It is generally thought that this is a result of small-scale wave action – the slick is periodically mixed into the surface layer and broken up into droplets, which rise back to the surface. The larger droplets rise quickly, remaining near the surface, and are moved more by the wind sheer, whereas the smaller droplets remain under the surface longer, feel the effect of the wind less, and thus fall behind. This process has been quantified with a models in Galt and Overstreet (2014) and Zeinstra-Helfrich et al. (2017).

6.1.1.3 Diffusion

Big whirls have little whirls that feed on their velocity, and little whirls have lesser whirls and so on to viscosity.

— Richardson (1922)

The ocean (and other water bodies) are dominated by turbulent flows – this turbulence results in a large range of sizes and strengths of eddies. These eddies result in a diffusion or spreading of materials in the water or on the water surface. Hydrodynamic models can only resolve eddy circulation down to a particular size, depending on grid size used and physics captured in the model. The effect of the eddies that are not resolved in the circulation model is usually captured as "diffusion".

However, the rate of diffusion is dependent on the spatial scale of the material being diffused; if two parcels are close together and are caught in a large eddy, they will tend to be moved together, resulting in advection rather than diffusion. On the other hand, if two parcels are about the same distance apart as the size of an eddy they are caught in, then they will tend to move apart, resulting in diffusion. There is more energy in the larger eddies, so it has been observed that the rate of diffusion increases with the spatial scale of the diffused material. This has been captured in what is known as "Richardson's $\frac{4}{3}$ power law" (Equation 6.1). This law is well known in the literature; good discussions can be found in Fisher et al. (1979) and Thorpe (2007), as well as in many other references.

$$D = \alpha L^{4/3} \tag{6.1}$$

where:
D is the diffusion coefficient,
L is the length scale of the plume (the averaging length for determining turbulence), and
α is a constant that appears to be pretty consistent with environmental flows ($0.002 < \alpha < 0.01$ cm$^{2/3}$/s).

For a release from a point source, this results in a plume or slick spreading slowly at first, with the rate of spreading increasing as the slick grows larger and encounters larger eddies This is caused by the effect of a changing of diffusion with spatial scale, but for a point source, it has the effect of diffusion increasing with time. For example, from experimental results, Elliott and Hurford (1989) recommend:

$$D = 0.33t^{1/6} \tag{6.2}$$

where:
D is the diffusion coefficient in m^2/s and
t is time in s.

Where is this in that article? That was a die study, done on a pretty small time and spatial scale? time was 0 to 5 min (very short!) so

The constant has units of $m^2s^{-7/6}$.

At a larger scale, diffusion is part of the transport processes in GNOME (Section 3.2.3). Diffusion is used in most oil spill models to capture transport by the largest of the eddies that are not included in the hydrodynamic models of ocean circulation. The diffusion coefficient is selected to match the location and field observations, but is typically on the order of $1 \times 10^5\,\mathrm{cm}^2/\mathrm{s}$ in coastal regions. This corresponds to a Richardson's scale of about 50km. The observed darker patches of oil are much smaller than that scale, so the random walk diffusion in GNOME shuld be considered a transport process rather than a process that makes the individual patches of oil thinner. That is, the GNOME random walk should be considered to be moving collections of patches of oil, rather than spreading out and thinning individual patches.

All of these processes serve to spread out individual patches of oil over time, but they all are happening at once, interacting with the same oil patch, and all at different time and length scales. Some processes (e.g., Fay spreading) are dependent on the oil thickness, whereas others (e.g., diffusion) are independent of oil thickness. What they all have in common is that they continue to spread out the oil film indefinitely. However, observations are clear that much of the oil in a real spill is in patches, streamers, and windrows of thicker oil. This is due to other physical processes at play.

6.1.1.4 Sub-mesoscale Circulations

Thickening of the oil slick has to be the result of a convergence of some kind. There are many convergences in the surface flow of the ocean – some of these are fairly large scale and can be identified at the mouths of estuaries, etc., but many are the result of mesoscale and submesoscale circulations that are not captured in hydrodynamic models These circulations serve to diffuse oil on large scales (GNOME transport diffusion), but can also serve to collect oil at smaller scales, resulting in thicker regions of oil.

[NOTE: there are some references from the GOMRI conference I should find]. Insert refs. CHB

While we are unable to model these processes precisely, this effect has been successfully captured in previous models (Dodge, Park, Buckingham, and Magott 1983; Reed 1989) by limiting the spreading to a minimum thickness dependent on initial oil viscosity.

flesh out this section with references, etc.

$$\delta_{min} = 10^{-4}\mathrm{m} \quad \text{if} \quad \nu_{oil,0} \geq 10^{-4}\mathrm{m}^2/\mathrm{s} \tag{6.3}$$

$$\delta_{min} = 10^{-5}\mathrm{m} + 0.909(\nu_{oil,0} - 10^{-6}) \quad \text{if} \quad 10^{-6}\mathrm{m}^2/\mathrm{s} \leq \nu_{oil,0} \leq 10^{-4}\mathrm{m}^2/\mathrm{s} \text{ and} \tag{6.4}$$

$$\delta_{min} = 10^{-5}\mathrm{m} \quad \text{if} \quad \nu_{oil,0} \leq 10^{-6}\mathrm{m}^2/\mathrm{s} \tag{6.5}$$

where $\nu_{oil,0}$ is the initial oil kinematic viscosity (in m^2/s).

Confirm that the Reed paper has the lesser terminal thickness for less Viscous oils

This approach is adequate for forecasting weathering processes because, surprisingly, slick area has little impact on other weathering processes. Spreading is a secondary process in the dissolution model and is not a factor in dispersion, emulsification, sedimentation, or biodegradation. Only evaporation is affected, with most, but not all, models assuming a nearly linear relationship between evaporation rate and slick surface area. Evaporation is thus fairly sensitive to the magnitude of spreading, but it is also somewhat self-correcting; if the model overspreads the oil, the oil will evaporate too fast, but once the volatile components are gone, evaporation slows, resulting in similar net evaporation rates regardless of the rate of spreading.

With a minimum thickness established, all the spreading processes discussed above serve to control how fast the terminal thickness is reached. Thus, the results are not very sensitive to those details.

6.1.1.5 Langmuir Circulation

Langmuir Circulation is one of the submesoscale processes that has been studied and can be parameterized enough to aid in the determination of the thickness of oil patches.

The spreading processes share the common factor that they are parameterized; i.e., they are not a function of the exact position of the oil, but rather of bulk properties of the ocean and oil itself. This is in contrast to the transport processes, where a Lagrangian-element approach serves to capture the highly spatially variable transport processes: the winds, currents, and shoreline interactions. Since they are parameterized, the model can apply the spreading processes to each individual element, so as to obtain a reasonable exposed area for evaporation of that element.

6.1.2 Time and Spatial Scales of Spreading Processes

All of the spreading processes are happening at the same time, but they each dominate over different timescales.

Immediately after a rapid release, Fay gravity-inertial spreading is dominant. After that, Fay gravity-viscous spreading becomes dominant, followed by diffusion.

6.1.2.1 Spill Rate

Often modelers refer to an "instantaneous" spill. However, in reality it takes some time for oil to spill out of a hole in a vessel or tank. Even if a tanker were to break in half, it would still take some time for all of the oil to escape.

We can get an idea of the timescale of spill releases with the simple leaking tank model in ADIOS2 (National Oceanic and Atmospheric Administration 1999):

$$\dot{V}_{oil} = 0.6\ A\sqrt{2g\Delta h} \approx 0.8\ A\ \sqrt{g}V_{oil}^{1/6} \tag{6.6}$$

where the ship or tank is treated as a full cube with a hole at the bottom,
$\dot{V}_{oil}$ is oil volume spill rate,
A is the area of the hole in the vessel, and
V_{oil} is the volume of oil in the vessel.

Assuming a $1\,\mathrm{m}^2$ hole, we get the following graph (Figure 6.1), and integrating both sides allows the time to drain the vessel in min (t_{drain}) to be defined as:

$$t_{drain} = \frac{1}{125}V_{oil}^{5/6} \tag{6.7}$$

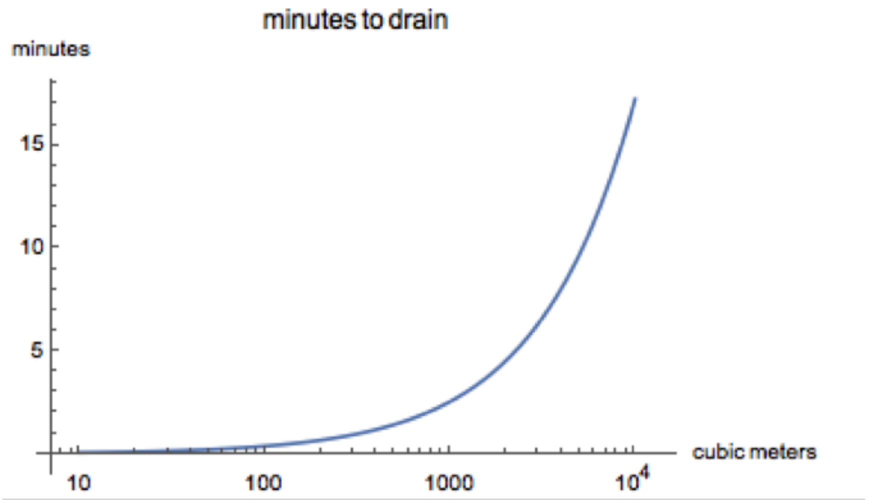


Figure 6.1: Drain rate of a tank as a function of time

remake this figure

Of course, real accidents would likely involve smaller, less regular holes, and water ingestion would slow the process.

In consideration of this, users should exhibit caution in setting instantaneous spills of large volume. Large spills should be modeled as a continuous spill appropriate to the size of and type of release. If the user is unsure the following volume-time relationships can be used as a guideline.

- If spill < 1000 bbl ≈ 15 min
- If 1000 bbl $<$ spill $< 10{,}000$ bbl ≈ 1 hr
- If spill $> 10{,}000$ bbl ≈ 2 hr

6.1.2.2 Gravity-Inertial Phase

For a rapid spill, the slick will spread quickly, due to Fay gravity-inertial spreading. This will occur, according to Dodge et al. (1983), at initial time $t_{drain,0}$:

$$t_{drain,0} = \left(\frac{K_2}{K_1}\right)^4 \sqrt[3]{\frac{V_{oil}}{\nu_w \cdot g \cdot \Delta rho}} \tag{6.8}$$

where:
$K_1 = 1.53$ (inertial phase constant) and
$K_2 = 1.21$ (viscous phase constant; Dodge et al. 1983, p. 135).

Or, with the constants combined:

$$t_{drain,0} = 0.4 \sqrt[3]{\frac{V_{oil}}{\nu_w \cdot g \cdot \Delta\rho_{ow}}} \tag{6.9}$$

check math on constant

where:
ν_w is the kinematic viscosity of water ($\approx 1 \times 10^{-6}\,\mathrm{m}^2/\mathrm{s}$),
g is the gravitational constant, and
$\Delta\rho_{ow}$ is the relative oil-water density difference.

Assuming some realistic values for oil density, we get:

$$T_0 \approx \frac{1}{2} V_{oil}^{1/3} \tag{6.10}$$

The timescale of Fay gravity-inertial spreading is shown in Figure 6.2.

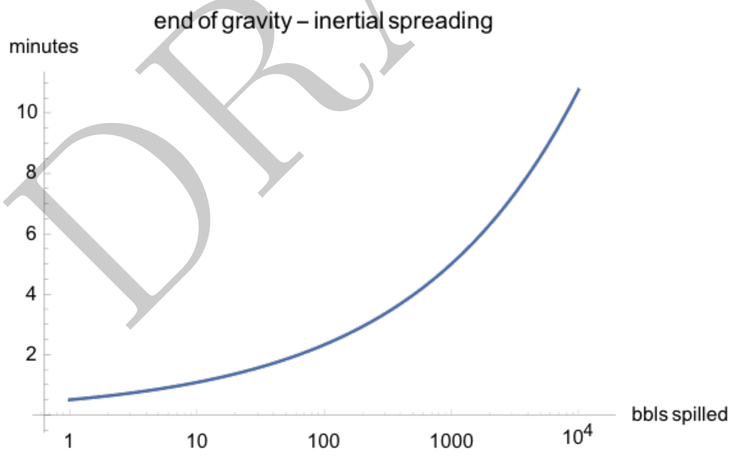


Figure 6.2: Timescale of Fay gravity-inertial phase of spreading

re-make this figure

By comparing Figures 6.1 and 6.2, it can be seen that the timescale for gravity-inertial spreading is generally far less than the timescale for spill rate. So the gravity-inertial phase can be considered effectively "instantaneous," and use the area at the end of this phase as the initial area (Dodge et al. 1983):

$$A_0 = \pi \frac{{K_2}^4}{{K_1}^2} \left(\frac{g \cdot \Delta\rho \cdot V_0^5}{\nu_w^2}\right)^{1/6} \tag{6.11}$$

or, with the constants combined:

$$A_0 = C_{ia} \cdot \Delta\rho^{1/6} \cdot V_0^{5/6} \tag{6.12}$$

$$C_{ia} = \pi \frac{{K_2}^4}{{K_1}^2} \left(\frac{g}{\nu_w^2}\right)^{1/6} \approx 42.09 \tag{6.13}$$

Add figure of initial area as a function of spill volume. and maybe initial thickness, too

6.1.2.3 Gravity-Viscous Phase

After the initial spill, the slick will continue to grow according to Fay gravity-viscous spreading. This occurs simultaneously with diffusion. In the early stages, gravity-viscous spreading dominates, but as the slick grows larger, diffusion takes over until the terminal area (thickness) is obtained.

The original work by Fay (1969) is the basis for most spreading in the gravity-viscous range. It was refined and re-published in Fay (1971), and then further refined by Dodge, Park, Buckingham, and Magott (1983). In that work, the radius of the slick as a function of time ($r_{slick}(t)$) (usually in m) is given as:

$$r_{slick}(t) = k_\nu \left(\frac{\Delta\rho_{ow} \cdot g \cdot V_{oil}^2 \cdot t^{3/2}}{\nu_w^{1/2}}\right)^{1/6} \tag{6.14}$$

where:
k_ν is the Spreading Law coefficient ($k_\nu = 1.45$),
$\Delta\rho_{ow} = \frac{\rho_w - \rho_o}{\rho_w}$ is the oil-water density difference (unitless),
V_{oil} is the volume of oil (in m^3),
t is time (in s), and
ν_w is the kinematic viscosity of water (in m^2/s).

This can be recast as the area for a circular slick (A_{slick}) in m^2 as:

$$A_{slick}(t) = \pi \cdot k_\nu^2 \left(\frac{\Delta\rho_{ow} \cdot g \cdot V_{oil}^2 \cdot t^{3/2}}{\nu_w^{1/2}}\right)^{1/3} \tag{6.15}$$

create figures with values for spreading for each of these

6.1.3 Computational Approach

6.1.3.1 Area-based Computation

Most of the estimations are provided as a function of time, but each of these spreading processes is occurring simultaneously. The physics used to derive the equations actually depend on the spreading that has already occurred. So to simultaneously apply these processes, they need to be recast as a differential equation in time:

$$\frac{dA}{dt} = f_F(A,t) + f_D(A,t) + ... \tag{6.16}$$

with the initial area (A_0) defined by Equation 6.11.

Recasting Equation 6.15 as a differential equation in time yields:

$$f_F(A) = \frac{dA}{dt} = \frac{1}{2}\pi \cdot k_\nu^2 \left(\frac{\Delta\rho_{ow} \cdot g \cdot V_{oil}^2}{\nu_w^{1/2}}\right)^{1/3} t^{-1/2} \tag{6.17}$$

If we group the constants as:

$$C = \pi \cdot k_\nu^2 \left(\frac{\Delta\rho_{ow} \cdot g \cdot V_{oil}^2}{\nu_w^{1/2}}\right)^{1/3}$$

This becomes:

$$f_F(A) = \frac{dA}{dt} = \frac{1}{2}Ct^{-1/2} \tag{6.18}$$

Then recasting as a differential equation in area yields:

$$f_F(A) = \frac{dA}{dt} = \frac{1}{2}\frac{C^2}{A} \tag{6.19}$$

Recasting the diffusion equation (Equation 6.2) from Elliott and Hurford (1989) as a differential equation in area yields:

$$f_D(A) = \frac{dA}{dt} = \frac{7}{6}K\left(\frac{A}{K}\right)^{1/7} \tag{6.20}$$

where:

$$K = 4 \cdot \pi \cdot 2 \cdot 0.033\,\mathrm{m}^2/\mathrm{s}^{7/6}$$

These can now be applied to determine the "exposed area" using standard numerical integration. From computational experiments, the Euler method is perfectly adequate as long as the time steps are on the order of one hour or less. The increase in area will terminate for a given element when the terminal thickness is reached (based on the initial viscosity).

6.1.3.2 Oil-thickness-based Computation

$$\frac{dA}{dt} = \frac{dA}{d\delta} \cdot \frac{d\delta}{dt} = f_F(A,t) + f_D(A,t) + ... \tag{6.21}$$

where

$$A = \frac{V_{oil}}{\delta},\ and\ \frac{dA}{d\delta} = -\frac{V_{oil}}{\delta^2} \tag{6.22}$$

Then, one obtains

$$\frac{d\delta}{dt} = -\frac{1}{2}\pi^2 \cdot k_\nu^4 \left(\frac{\Delta\rho_{ow} \cdot g}{\nu_w^{1/2} \cdot V_{oil}}\right)^{2/3} \delta^3 - \frac{7}{6} \cdot \left(\frac{K}{V_{oil}}\right)^{6/7} \cdot \delta^{13/7} \tag{6.23}$$

where

$$\delta(0) = \frac{1}{\pi}\frac{{K_1}^2}{{K_2}^4}\left(\frac{\nu_w^2 \cdot V_{oil}}{\Delta\rho_{ow} \cdot g}\right)^{1/6} \tag{6.24}$$

The equation is simplified as follows

$$\frac{d\delta}{dt} = -C_1\delta^3 - C_2\delta^{13/7} \tag{6.25}$$

where

$$C_1 = \frac{1}{2}\pi^2 \cdot k_\nu^4 \left(\frac{\Delta\rho_{ow} \cdot g}{\nu_w^{1/2} \cdot V_{oil}}\right)^{2/3} \tag{6.26}$$

$$C_2 = \frac{7}{6} \cdot \left(\frac{K}{V_{oil}}\right)^{6/7} \tag{6.27}$$

At each time step

$$\delta(t + \Delta t) = \delta(t) - (C_1 \cdot \delta(t)^3 + C_2 \cdot \delta(t)^{13/7}) \cdot \Delta t \tag{6.28}$$

Then, at each time step, the area can be computed as follows

$$A = \frac{V}{\delta} \tag{6.29}$$

For stability of such numerical scheme, sub-time step is introduced and calculated as follows:

$$\Delta t_{sub} = \frac{\delta(t)}{(C_1 \cdot \delta(t)^3 + C_2 \cdot \delta(t)^{13/7})} \tag{6.30}$$

The time step will be first divided into multiple sub-time steps, and fourth order Runge-Kutta method is then implemented at every sub-time step:

$$k_1 = C_1 \cdot \delta(t)^3 + C_2 \cdot \delta(t)^{13/7} \tag{6.31}$$

$$k_2 = C_1 \cdot (\delta(t) + \Delta t_{sub} \cdot \frac{k_1}{2})^3 + C_2 \cdot (\delta(t) + \Delta t_{sub} \cdot \frac{k_1}{2})^{13/7} \tag{6.32}$$

$$k_3 = C_1 \cdot (\delta(t) + \Delta t_{sub} \cdot \frac{k_2}{2})^3 + C_2 \cdot (\delta(t) + \Delta t_{sub} \cdot \frac{k_2}{2})^{13/7} \tag{6.33}$$

$$k_4 = C_1 \cdot (\delta(t) + \Delta t_{sub} \cdot k_3)^3 + C_2 \cdot (\delta(t) + \Delta t_{sub} \cdot k_3)^{13/7} \tag{6.34}$$

$$\delta(t + \Delta t) = \delta(t) - \frac{1}{6}\Delta t_{sub} \cdot (k_1 + 2k_2 + 2k_3 + k_4) \tag{6.35}$$

6.1.3.3 Application of Langmuir Circulation

To fill in from Bill's notes.

NOTE: is the Langmuir "correction" applied the entire time? i.e., early after release, the oil is quite thick, with a small area. Perhaps Langmuir should not thicken it more. Perhaps we can have a threshold: compute the Langmuir correction on the minimum thickness, and essentially use this as the "new" minimum thickness – that is, with Langmuir applied, the oil will never be thinner than that. details are not obvious to me (Chris) right now.

6.1.3.4 Multiple Elements

The transport part of GNOME requires many individual elements to capture the transport, so the weathering code, in particular the spreading algorithm, needs to be insensitive to the number of elements used for the computation. If the mass released was divided equally among the elements, and each element's mass (volume) was used to initiate the spreading, then the oil would spread less as the number of elements increased. To compensate for this effect, each element is initialized with an area that is computed from the full amount of oil released at the same time:

put this in equation form

With this procedure, the total area of all the elements will remain the same, regardless of the number of elements used. If the total area is the same, the rest of the weathering algorithms will produce the same results.

show results from computational experiments demonstrating this

6.1.3.5 Continuous Release

Most spills are not instantaneous and are often over much longer timescales than the time step of the model. However, most of the algorithms assume an instantaneous release. For a continuous release, there is more oil being added to the slick as the slick spreads. Therefore, instead of a constant oil volume, for a continuous release the effective oil volume changes with time Dodge, Park, Buckingham, and Magott (1983):

$$V_{oil}(t) = \dot{V}_{oil} \cdot t \tag{6.36}$$

However, the timescale of a continuous release is generally longer than the timescale of the initial gravity-inertial spreading phase. So, oil released later on does not interact with oil released at the beginning. Also, in the coastal ocean, there generally are currents, winds, and perhaps a moving source that all ensure that oil released early in the event is moved away from the oil currently being released. To consider such effects, a 'spreading cumulative time scale', t_r, (e.g., 12 hours) is introduced to define the maximum time length under which the volume accumulation of released oil is considered, and therefore:

$$V_{oil}(t) = \dot{V}_{oil} \cdot min(t, t_r) \tag{6.37}$$

computationa results demonstrating this

6.1.3.6 Element-based computation

In my occasions, released elements might move out of the simulated domain or need to be eliminated within the domain due to weathering processes. Therefore, element-based computation is more convenient for the oil spreading algorithm as it allows the computation to be performed on each individual element rather than bundling them together as an oil slick. To implement the element-based computation, a volume fraction is introduced and calculated for elements as they are initially released:

$$v_{frac,LE} = \frac{V_{LE}}{V_{oil}} \tag{6.38}$$

where V_{LE} denotes the initial volume assigned to each element, and V_{oil} represents the cumulative oil volume at a specific time step.

Then, the element-based algorithm can be expressed as follows:

$$A_{0,LE} = A_0 \cdot v_{frac,LE} \tag{6.39}$$

$$f_{F,LE}(A_{LE}) = \frac{dA_{LE}}{dt} = [\frac{1}{2}\frac{C^2}{\frac{A_{LE}}{v_{frac,LE}}} + \frac{7}{6}K(\frac{\frac{A_{LE}}{v_{frac,LE}}}{K})^{1/7}] \cdot v_{frac,LE} \tag{6.40}$$

where

$$C = \pi \cdot k_\nu^2 \left(\frac{\Delta\rho_{ow} \cdot g \cdot V_{oil}^2}{\nu_w^{1/2}}\right)^{1/3} \tag{6.41}$$

6.2 Pseudocomponents

Petroleum is a complex mixture of thousands of individual compounds. As it is impossible to characterize an oil down to the compound level, and it would be computationally impossible to model each compound individual, GNOME (similarly to all known oil weathering models) uses a "pseudo-component" (PC) approach.

In the pseudo-component approach, crude oils and refined products are modeled as mixtures of discrete non-interacting components. The whole oil is divided up into a small number of PCs that each represent a collection of similar compounds. In this way, the code can track how the whole oil changes as individual PCs interact with the environment in different ways.

In order to be a flexible framework, the weathering algorithms in GNOME can work with any number ($\geq$ 1) of PCs, each with a distinct set of properties. Thus depending on the application, the oil can be described by PCs that best represent the oil for that application. A description of how PCs are usually constructed from an oil assay for typical oil modeling application with GNOME, see Appendix **??**.

Update the OilLibrary Section to follow what is done in the new adiosdb code.

6.2.1 Construction of Pseudo-components

An oil assay does not directly have the data for the PCs – effective construction of PCs depends on the availability of oil data measurements. One of the key weathering processes that acts on different component differently is evaporation, which is highly dependent on the boiling point of the component. As oil is a mixture of many compounds with a wide range of boiling points, different fractions will boil off at different temperatures. This distillation process is used to refine petroleum into useful products, and thus a distillation curve is a commonly produced as part of an oil analysis, and therefore is used herein to construct PCs. The range of boiling points (BPs) have been considered in different ways due to different modeling needs. The low-BP fraction usually evaporates very quickly and thus can be considered as a single PC. The high-BP fraction is non-volatile and those compounds can be treated as a single PC as well. The fraction within the middle-range BP is moderately volatile and thus good resolution is required to resolve the evaporation process in the environment.

As evaporation is a key weathering process, construction of PCs is determined by the boiling point of oil compounds. Note that separation of PCS by other properties, such as solubility, may be required, but in any case, no one PC can include too wide a range of boiling points.

The PCs are classified into three categories: components with low BPs, mid-range BPs, high BPs, respectively. The first PC includes the fraction from the initial BP to 8 °C to represent the compounds with BPs lower than benzene. There is no need to further refine this range, as these components will all evaporate quickly. The second PC includes the fraction from 8 °Cto 144 °C to represent BTEX and whatever comopounds fall in that BP range. One PC includes the fraction of high BP compounds, and its representative BP is the average of the threshold of high boiling temperature (e.g., 400 °CC) and the terminal boiling temperature. Linear interpolation of oil distillation data is performed to obtain the mass fraction for the PCs constructed at the low and high BPs. The middle range of BPs is divided into PCs based on a temperature resolution (e.g., 20 °C). The BPs of PCs is determined by the midpoint of each boiling temperature range, and the corresponding mass fractions are calculated by linear interpolation of oil distillation data. An example of PCs constructed for the Alaskan North Slope (ANS) oil is shown below.

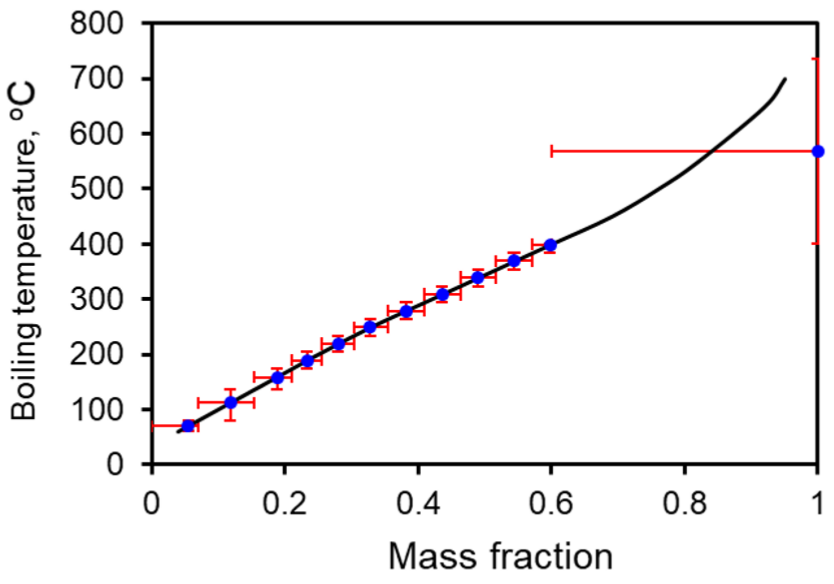


Figure 6.3: PCs constructed for the ANS oil

6.2.2 Properties of Pseudocomponents

Each PC represents a certain mass fraction of the whole oil. The sum of all mass fractions must be 1.0 to represent a full oil.

6.2.2.1 Basic Properties

Each PC must have all of the following properties:

Fill in the rest of this table – make sure it matches the appendix

Property	Units	Description
Mass Fraction	unitless	Sum of all PCs must be 1.0
Molecular Weight	kg per mole	
Representative Boiling Point	K	
Density	kg/m^3	
Kinematic Viscosity	m/s^2	
Solubilty constant	unitless	(may be zero or one)
Biodegradation constant	unitless	

6.2.2.2 Molecular weight of Pseudocomponents

The molecular weight of PCs is calculated based on Equation (2.42) proposed in Riazi [2005]:

$$MW = \left(\frac{1}{b}[a - ln(T_\infty - T)]\right)^q \tag{6.42}$$

where MW and T denote the molecular weight in g/mol and temperature in K, respectively; coefficient T_∞, a, b and q are specific for each property, which have been estimated in Riazi [2005] for paraffinic, naphthenic, and aromatic hydrocarbon groups, reported in the table below. As each PC represents a range of boiling temperature, an integration is performed to obtain the representative boiling point for each PC, expressed as:

$$MW = \frac{\int_{T_0}^{T_1} MW(t)dt}{T_1 - T_0} \tag{6.43}$$

The simulation tests indicate that the molecular weight calculated from different set of coefficients has negligible impacts on forecasting oil evaporation in GNOME. Therefore, the average of the three hydrocarbon groups is simply used to compute the molecular weight of each PC.

Hydrocarbon group	a	b	T_∞
Paraffinic	6.98291	0.02013	1070
Naphthenic	6.95649	0.02239	1028
Aromatic	6.91062	0.02247	1015

Table 6.1: Parameter values estimated in Riazi [2005]

6.2.2.3 Density of Pseudocomponents

The density of pseudocomponents is estimated based on a third-order polynomial curve (density as a function of boiling temperature) fitted to ExxonMobile datasets shown below.

6.3 Evaporation

6.3.1 Theory

Each pseudocomponent is treated as a single substance with an associated vapor pressure. The evaporation of each component is tracked separately.

Evaporation of volatile components is found from the solution of coupled simple differential equations. This is the same formulation as that found in ADIOS2.

$$\frac{dm_i}{dt} = -(1-f_w)\frac{AKMW_iP_i}{RT_w}\left[\frac{m_i/MW_i}{\sum m_i/MW_i}\right] \tag{6.44}$$

CHB: Is this still right? I though there was a mole-fraction factor in here?

where:
m_i is the mass of PC i in the LE (in kg),
f_w is the fractional water content in the emulsion,
A is the surface area associated that element (in m^2),
MW_i is the molecular weight of PC i (in kg/mol),
P_i is the vapor pressure at the water temperature of PC i (in Pa),
R is the universal gas constant in SI units (8.3144 J/(K mol)),
T_w is the water temperature (in K),
K_i is the mass transfer coefficient (in m/s), expressed as follows

velocity units???

$K = c \cdot U_{10}^{0.78} \cdot Sc^{-2/3}$
where:
U_{10} is the wind speed at 10 meters above water surface (in m/s),
$c = 0.0048$ (in m$^{1/3}$/s$^{2/9}$),
Sc denotes Schmidt number, and $Sc = (\frac{D_{air}}{D_{water}\sqrt{\frac{MW_{water}}{MW_i}}})$
The vapor pressures of the components are derived from their boiling points using Antoine's equation,
$ln\frac{P_i}{P_0} = \frac{\Delta S_i(BP_i-C)^2}{R \cdot BP_i}\left[\frac{1}{BP_i-C} - \frac{1}{T_w-C}\right]$
where:
$C = 0.19 \cdot BP_i - 18$ and $\Delta S_i = 8.75 + 1.987 \cdot log(BP_i)$

Add ref to bib file. DAH

We need a section on how to get Vapor Pressure from MW and BP

6.3.2 Numerical Implementation

The differential equations are reorganized as follows:

$$\frac{dm_i}{dt} = -edc \cdot m_i \tag{6.45}$$

where

$$edc = \frac{(1-f_w)AK_iP_i}{RT_w\sum(m_i/MW_i)} \tag{6.46}$$

The equation is solved as follows:

$$m_i(t) = m_i(0)exp(-edc \cdot t) \tag{6.47}$$

The equation is further organized as follows, and used for the computation at each time step:

$$m_i(t+\Delta t) = m_i(t)exp(-edc \cdot \Delta t) \tag{6.48}$$

6.4 Dispersion

Entrainment is a process of oil on the surface (slick) being broken up into droplets and submerged beneath the surface. It is primarily driven by breaking waves in the surf zone, or white capping in the open ocean (entrainment can also occur in rivers via different processes, not currently included in GNOME). The entrainment process generates droplets with a distribution of droplet sizes, and pushes them below the water surface to some limited depth. Once droplets have been entrained, they are subject to subsurface processes as droplets: they rise toward the surface (if less dense than the water), and are subject to (limited) dissolution and bio degradation.

Most of the oil does not dissolve in water the same way that, for example, salt does, so droplets are relatively persistent. The rise velocity is a function of droplet size, with smaller droplets rising more slowly. Very small droplets have a small enough rise velocity that natural turbulence in the ocean will prevent the oil from resurfacing, just as turbulence in the air keeps small dust particles aloft. This process is known in the oil spill community as dispersion. Chemical surfactants (known as dispersants) are sometimes sprayed on slicks to enhance this process, but it will occur naturally provided that there is sufficient ocean energy and the spilled oil has a low enough viscosity.

For oil spills during storm events, dispersion is the chief removal mechanism from the surface slick. For spills under more normal weather conditions, evaporation will usually be more significant, but dispersion can still be important. Once in the water column, the oil may be ingested by planktonic organisms, a process not modeled in GNOME, or may attach to organic or inorganic sediment mixed with the water. This latter process is called sedimentation, or Oil-Sediment Aggregation (OSA). More recently, a process known as Marine Oil Snow Sedimentation and Flocculent Accumulation (MOSSFA) was identified, apparently being a significant factor in the ultimate fate of oil in the Deepwater Horizon incident. This process also begins with the entrainment of oil droplets from surface slicks. MOSSFA processes are not currently included in the GNOME model.

Vertical dispersion of oil into the water column is estimated using a modified version the hydraulic model developed by Delvigne and Sweeney (1988). They measured the number and size distribution of oil droplets which are driven into the water column by breaking waves, usually white capping in the open ocean. The vertical entrainment of oil is directly proportional to the dissipation of energy of a single white cap, the total dissipation rate for given wave spectrum, and the volume of oil injected into the surface layer each time a wave breaks.

The entrainment of oil with units in kilogram per second is given by

$$Q_{disp} = C_{Roy} * C_{disp} * V_{entrain} * (1.0 - Y) * A \tag{6.49}$$

Here $V_{entrain} = 3.9e^{-8}m^3$ (Delvigne and Sweeney 1988) is the volume of oil entrained for unit area of oil slick. C_{Roy} is an experimentally determined parameter that captures the effect of oil viscosity.

$$C_{Roy} = 2400.0 * \exp(-73.682 * \sqrt{\nu}) \tag{6.50}$$

and $C_{disp} = D_e^{.57} * f_{bw}$, where f_{bw} is fraction of breaking waves per wave period and D_e is the dissipative wave energy. The dissipation of wave energy per unit surface area is given by

$$D_e = .0034 * \rho_w g H_{rms}^2 \tag{6.51}$$

where g is gravitational acceleration, ρ_w is the water density, and H_{rms}, is the root-mean wave height in meters, assumed to be related to the spectrally based significant wave height, H_0, as

$$H_{rms} = .707 H_0 \tag{6.52}$$

$V_{entrain}$ is the volume of oil "entrained" (dispersed) by a given white cap event. It is the volume of droplets that are small enough to stay submerged for some time by background turbulence.

$V_{entrain}$ is computed by integrating the droplet volume distribution, computed as the product of the droplet volume and the droplet size distribution over the volume of oil. In practice, the integration is performed between the minimum droplet size and maximum droplet size, determined from experimental data. This yields

$$V_{entrain} \propto \int_{d_{min}}^{d_{max}} N(\delta)\delta^3 d\delta \tag{6.53}$$

where a maximum droplet size, d_{max}, is set equal to 70 microns. Larger droplets will refloat in times shorter than the time for the surface slick to traverse the area covered by the dispersed oil and hence will rejoin the surface slick. Drops 70 microns or smaller are effectively held in suspension as shown by examining the steady-state tail of the of the droplet diameter versus refloat time curve as measured by Delvigne. The minimum droplet size is probably about five microns although setting d_{min} to zero introduces negligible error. $N(\delta)$ is the number of oil droplets per unit volume of water per unit droplet diameter. GNOME uses the equation

$$N(\delta) = N_0 \left(\frac{\delta_0}{\delta}\right)^{\frac{2}{3}} \tag{6.54}$$

where N_0 and δ_0 are experimental reference values. The result is a constant value that represents the shape of the DSD.

6.4.1 Li et al Formulation

The above is an adaptation of the original work by Delivigne and Sweeny – the venerable solution in spill modeling. In spill response observations, it has been proven to capture the general order of magnitude of the process, but it notably can not take into account the effects of chemical dispersant application, and it it questionably applicable to high viscosity oils.

There has been more recent work on oil entrainment / dispersion modeling. GNOME currently (optionally) includes an implementation of the one of the newer formulations, based on Li, Spaulding, French McCay, Crowley, and Payne (2017, Li, Spaulding, and French-McCay (2017)

In this approach, the entrainment of oil is given by:

$$Q_{disp} = \rho * \delta * F_{bw} * (1.0 - Y) * Q_0 \tag{6.55}$$

where F_{bw} represents the fraction of the sea surface covered with breaking waves, and Q_0 represents the dimensionless flux from the sea surface into the water column. Q_0 can be expressed in terms of two well-known dimensionless groups: the Weber (We) and Ohnesorge (Oh) numbers.

$$Q_0 = aWe^b * Oh^c \tag{6.56}$$

The Weber number, We, is the ratio of the disruptive momentum (hydrodynamic) forces to the restorative (interfacial tension) forces. It is a function of seawater density, gravity, wave height, the oil-water interfacial tension, $\Delta\rho$, and the Rayleigh-Taylor (R-T) instability maximum diameter, d_o, which is given by Grace et al., (1978) as:

$$d_o = 4 * [\frac{\sigma_{o-w}}{\Delta\rho * g}]^{0.5} \tag{6.57}$$

Assuming that the product of gravity and the significant wave height, $g * H_s$, represents the potential energy in the breaking waves, the Weber number can be expressed as Reed et al., (2009):

$$We = \frac{\rho_w * g * H_s * d_o}{\sigma_{o-w}} \tag{6.58}$$

The Ohnesorge number (**?**; **?**), Oh, is the ratio of viscous to interfacial tension forces and is a function of the oil dynamic viscosity, the oil density, the oil-water interfacial tension,k and the R-T instability maximum diameter.

$$Oh = \frac{\mu_o}{\sqrt{\rho\sigma_{o-w}d_o}} \tag{6.59}$$

In this equation, a, b, and c are empirical constants estimated in Li, Spaulding, French McCay, Crowley, and Payne (2017, Li, Spaulding, and French-McCay (2017), equal to $4.604*10^{-10}$, 1.805, -1.023, respectively.

6.5 Dissolution

Note: The dissolution module is considered experimental, and it turned off by default in the WebGNOME system. It can be experimented with via the PyGNOME scripting environment.

GNOME replaces the ADIOS2 formulas with an expanded version of the new SOLUTE-SINK, or BM model, of LSU (Thibodeaux and Overton 2014; Stevens, Thibodeaux, Overton, Valsaraj, Rao, and Walker 2015). The BM model separates the spilled oil into volumes that will dissolve and volumes that will not dissolve.

Based on estimated low Biot numbers, early MacKay studies, and recent LSU research (Loebig 2015), practical dissolution is considered to be:

1. Dependent only on the oil-water surface resistance,

2. Limited to aromatics, and
3. Decreasing with increasing carbon number.

Assumption 1 may not be true for a very viscous, thick oil.

6.5.1 Mass Transfer Rate

According to Equation 19 in Cohen et al. (1980), we can estimate the mass transfer rate (N) in kg/m^2/s of the soluble component as:

$$N = k_w \cdot c_{oil} \cdot \frac{1}{K_{ow}} \tag{6.60}$$

where:
k_w is the water phase transfer velocity (in m/s) taken as the rise velocity for the individual droplet,
c_{oil} is the saturation concentration in oil (in kg/m^3), and
K_{ow} is the partition coefficient (unitless).

6.5.2 Partition Coefficient

GNOME uses a partition coefficient that compares mass concentration ratios, not mole concentrations. However, mass concentration (c_{oil}) in kg/m^3 is related to the molar concentration (M_{oil}) in mol/m^3 through the component's molecular weight (MW) by:

saturation? DAH

$$c_{oil} = MW \cdot M_{oil} \tag{6.61}$$

so that the two dimensionless partition coefficients are equivalent.

Equation 4 in Lee et al. (1992) relates the partition coefficient to the molar concentration in the oil phase and the molar solubility in the water phase. However, the same form holds for mass concentrations and mass solubility.

We switched away from Lee so this should be updated or removed

Mass concentrations and mass solubility can then be used to compute the partition coefficient (K_{oil}) using:

missing equation here?

$$K_{oil} = \frac{c_{oil}}{f_i \cdot S_i} \tag{6.62}$$

where:
f_i is the mass fraction of hydrocarbon i (unitless), and
S_i is the aqueous solubility for hydrocarbon i (in kg/m^3).

Dissolution may occur from the bottom of the surface slick and from the surface of dispersed oil droplets. If one compares specific saturate hydrocarbons to aromatic hydrocarbons with similar molecular weights, e.g., n-octane with $MW = 114$ and toluene with $MW = 92$, one finds that the partition coefficient of the saturate is orders of magnitude smaller than that for the weight-equivalent aromatic: 10^{-6} for n-octane versus 10^{-3} for toluene. Therefore, non-aromatic compounds are treated as essentially insoluble.

See Figure 6.4.

Note since we switched from Lee to Banerjee the figure needs to be updated

Among aromatics, the partition coefficient (K_{ow}) decreases with an increase in density. The correlations by Huibers and Katritzky (1998) and Banerjee et al. (1980) are combined to estimate the correlation between a specific aromatic hydrocarbon's density and molecular weight with its partition coefficient, as illustrated in Figure 6.4. Huibers and Katritzky (1998) relates solubility in mol/L to molar volume (mv) measured in Å^3. Assuming that molar volume can be expressed as molecular weight (MW) divided by density (ρ), and using proper MKS units for molecular weight (kg/kmol) and density (kg/m^3), we get Equation 7 from Huibers and Katritzky (1998):

Make these MW_i and ρ_i? DAH

$$S_i = 10^{1.27} \cdot 10^{-0.038 V_i} = 18.62 \cdot exp\left[-0.87\left(\frac{MW_i}{\rho_i}\right)\right] \tag{6.63}$$

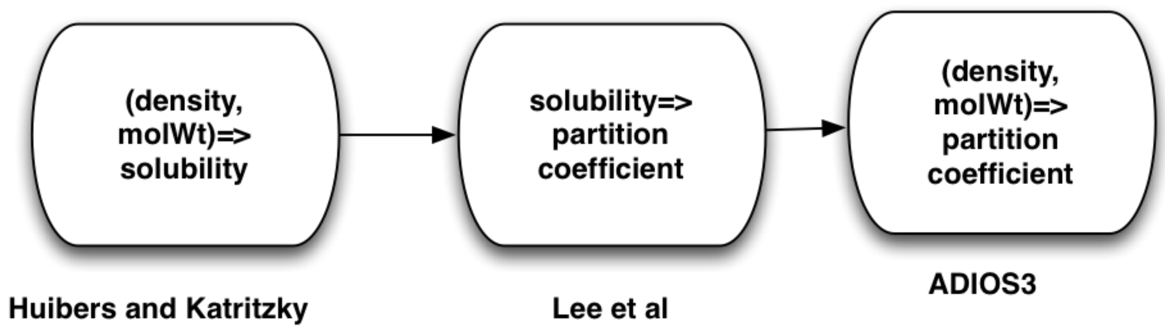

(density, molWt)=> solubility
solubility=> partition coefficient
(density, molWt)=> partition coefficient
Huibers and Katritzky
Lee et al
ADIOS3

Figure 6.4: Correlations by Huibers and Katritzky (1998) and Banerjee et al. (1980)

NOTE: the paper uses mv in m^3/mol, so the units are converted to molwt/density in the code, which uses m^3/mol, to match. Also they acknowledge that the equation produces solubility numbers that are on the low side.

is this right? those are the same units? and is that mv or MW?

Previously, we used the Lee approximation (Lee et al. 1992). This does not seem to work. Therefore, I (Bill) suggest Banerjee et al. (1980), Equation 4. They use micromoles, so we need to convert to moles to match the Huibers and Katritzky equation.

What is the final decision here?

Banerjee et al. (1980) relates K_{ow} (unitless ratio) to S_w (µm) as:

$$K_{ow} \propto {S_w}^{-2/3} \tag{6.64}$$

Combining the two equations gives:

$$K_{ow} = \frac{10^{1.12}}{{S_i}^{0.68}} = 13.18 \cdot 18.62^{-0.68} exp(0.087 \cdot 0.68 mv_i) = 1.8 \cdot exp\left(0.059 \cdot \frac{MW_i}{\rho_i}\right) \tag{6.65}$$

6.5.3 Estimating the Partition Coefficient

The pseudocomponents (Section 6.2) provide a fraction of aromatics for each distillation cut boiling point. For each aromatic PC, molecular weight and density are provided.

For example, using measured values of toluene: molecular weight (MW) = 92.1 kg/mol, density (ρ) = 866 kg/m^3, and the partition coefficient (K_{ow}) for toluene = 1000, the constant A can be estimated as 828.

Density and molecular weight typically are estimated from distillation cut temperatures, so it makes sense to compute K_{ow} directly from the distillation cuts.

Using Riazi (2005), aromatic molecular weight (MW) in kg/kmol can be approximated from the distillation cut temperature (Θ_j) in K as:

$$MW = 350(6.98 - ln(1070 - \Theta_j))^{3/2} \tag{6.66}$$

and aromatic density (ρ_j) in kg/m^3 is:

$$\rho_j = 100 \cdot \Theta_j^{1/3} \tag{6.67}$$

where j refers to the particular distillation cut.

Setting the ratio of molecular weight to density (λ_j) for each distillation cut (j) as:

$$\lambda_j = \frac{m_i}{\rho_i} \tag{6.68}$$

and plotting λ over the typical range of distillation cut temperatures results in the graph in Figure 6.5.

In figure, correct spelling of Kelvin and standardize parentheses. DAH

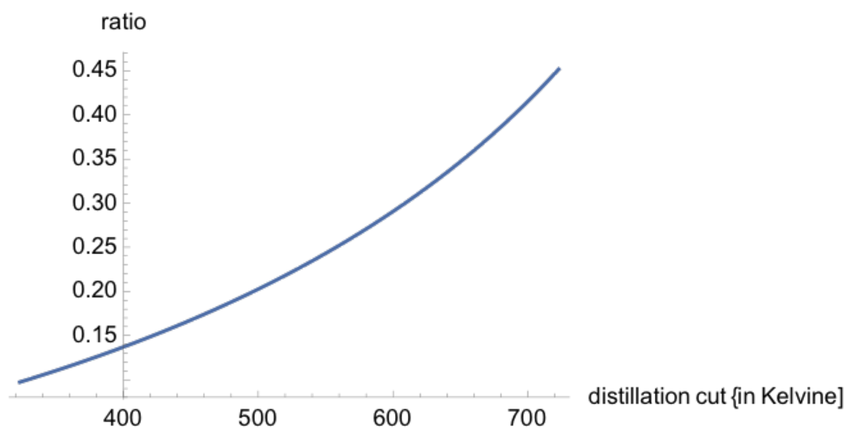


Figure 6.5: Ratio of molecular weight to density vs. distillation cut temperature

Then plotting the partition coefficient (K_{ow}) as a function of the ratio (λ) results in the graph in Figure 6.6.

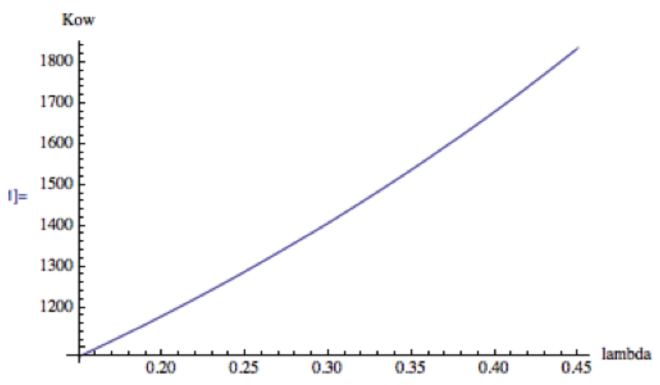


Figure 6.6: Partition coefficient (K_{ow}) vs. ratio of molecular weight to density (λ)

Check label to left of y-axis. DAH

6.5.4 Overall Partition Coefficient

To calculate an overall partition coefficient (K_{ow}) for the aromatic hydrocarbon (j), it is necessary to do a molar average of the aromatic components. Mass fraction (f_j) for each component (j) is stored using:

$$K_{ow} = \frac{\sum_j \frac{f_j}{MW_j} \cdot K_{ow,j}}{\sum_j \frac{f_j}{MW_j}} \tag{6.69}$$

If we define α_j as:

Is α_j the soluble proportion of component j? DAH

$$\alpha_j = \frac{\frac{f_j}{MW_j}}{\sum_j \frac{f_j}{MW_j}}, \text{ with } \sum_j \alpha_j = 1 \tag{6.70}$$

for each component (j) in the aromatic hydrocarbon, then the preceding equation for the partition coefficient can be written as:

$$K_{ow} = \sum_j \alpha_j \cdot K_{ow,j} \tag{6.71}$$

6.5.5 Water Phase Transfer Velocity

Water phase transfer velocity (k_w) can be calculated for small buoyant droplets using Stoke's law, assuming steady-state velocity given by:

$$k_w = \frac{1}{18\pi} \cdot \Delta\rho_{ow} \cdot \frac{g \cdot A_{droplet}}{\nu_w} = 176 \cdot 10^3 \cdot \Delta\rho_{ow} \cdot A_{droplet} \tag{6.72}$$

where:
$\Delta\rho_{ow}$ is the relative oil-water density difference,
ν_w is the kinematic water viscosity (approx. 1 centistoke), and
$A_{droplet} = \pi r^2_{droplet}$ (surface area of droplet in m^2).

This assumes spherical droplets with diameters ($\delta_{droplet}$) less than 400 µm. Larger droplets will be distorted and rise more rapidly (Tkalich and Chan 2002). Large droplets are presumed to rise so quickly that little dissolution takes place for them except as part of the dissolution from the surface slick.

A plot of water phase transfer velocity (k_w) in m/s vs. droplet diameter ($\delta_{droplet}$) in µm for an oil-water density difference of $\Delta\rho_{ow} = \{0.1, 0.2, 0.3\}$ is shown in Figure 6.7.

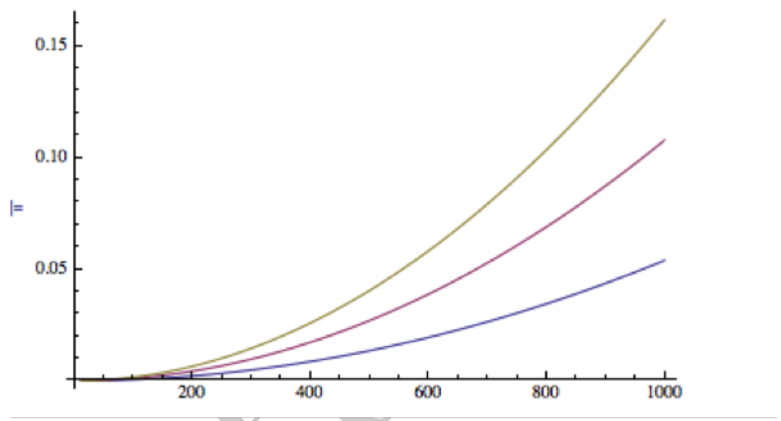


Figure 6.7: Water phase transfer velocity vs. droplet diameter for an oil-water density difference of {0.1, 0.2, 0.3}

6.5.6 Weathering of Surface Oil in Any Time Step

The oil is assumed to be on the surface at the beginning of the time step. The time step is assumed to be long compared to the time between white capping events. When a white cap occurs, the oil volume will be divided into three different pots, as shown in Figure 6.8.

6.5.7 Droplet Size Distribution

The amount of oil in each pot depends on droplet size. Breaking waves will cause the oil in a surface slick to break into droplets based on a distribution function of size ($\mathbb{N}(\delta_{droplet})$). The total volume of oil inserted into the water column (V_{wc}) is then:

$$V_{wc} = \int_{\delta_{min}}^{\delta_{max}} \mathbb{N}(\delta_{droplet}) \cdot \frac{\pi \cdot \delta^3_{droplet}}{6} \cdot d(\delta_{droplet}) \tag{6.73}$$

where δ_{min} and δ_{max} are the smallest and largest diameters of droplets, respectively.

Here, $\delta_{max} = 400$ µm. Minimum droplet size is discussed below. These drops are inserted in a vertically uniform distribution into the water column to a depth of 1.5 the height of the significant wave ($H_{1/3}$) in 1.5x? DAH

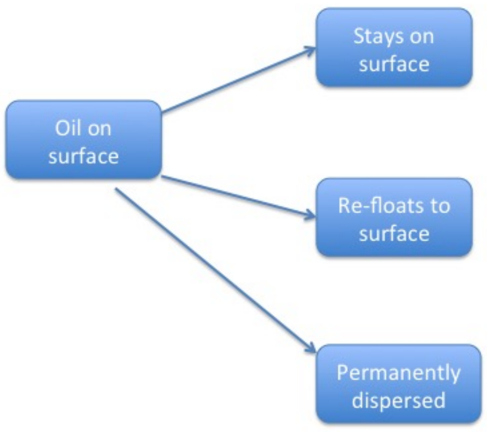


Figure 6.8: Division of oil volume when breaking wave event occurs

m. GNOME assumes a Pierson-Moskowiz spectrum for fully developed, wind-induced, surface waves. This relates significant wave height to the wind speed at 10-meter elevation (U_{10}) in m/s, as:

$$H_{1/3} \simeq 0.22 \cdot \frac{U_{10}^2}{g} \tag{6.74}$$

where g is the gravitational constant.

GNOME requires input regarding the fraction of waves that break. Ding and Farmer (1993) note that the dependence of breaking wave properties on wind speed is variable. According to Delvigne and Sweeney (1988), the fraction of breaking waves (f_{bw}) is given by:

$$f_{bw} = \frac{0.032 \cdot (U_{10} - 5)}{\tau_{peak}} \tag{6.75}$$

where τ_{peak} (in s) is the peak wave period.

Delvigne and Sweeney (1988) assumes no breaking waves for winds less than ten knots. Lehr and Simecek-Beatty (2000) developed a slightly different formula that allowed breaking waves for winds as low as six knots.

All these formulas may be modified as the new Katz experiment results become available and as the GNOME team replaces Delvigne and Sweeney (1988) for natural surface dispersion. The time period between whitecap events (τ_{bw}) in s is given by the wave period divided by the fraction of waves that break:

$$\tau_{bw} = \frac{\tau_{peak}}{f_{bw}} \tag{6.76}$$

Ding and Farmer (1993) note that the duration of the breaking event is about half the wave period, presumably $0.5 \cdot \tau_{peak}$ (in s), although this is unclear from their text. Therefore, the time available for the droplets to refloat (calm period or τ_{calm}) is:

$$\tau_{calm} = \tau_{bw} - 0.5 \cdot \tau_{peak} = (\frac{1}{f_{bw}} - \frac{1}{2}) \cdot \tau_{peak} \tag{6.77}$$

Assuming that the "average" droplet is inserted to a depth of $0.75 \cdot H_{1/3}$, then the average refloat time (τ_{rf}) in s for the droplet is:

$$\tau_{rf} = \frac{3}{4} \cdot \frac{H_{1/3}}{k_w} \tag{6.78}$$

The overall time scale is given by:

$$\tau_{bw} > \tau_{calm} > \tau\{\tau_{rf}, \tau_{peak}\} \tag{6.79}$$

The time fraction that the droplet spends in the water column (f_{wc}) is given by the refloat time (τ_{rf}) divided by the calm period (τ_{calm}) between breaking wave events in s:

$$f_{wc} = \frac{\tau_{rf}}{\tau_{calm}} = \frac{3}{4} \cdot \frac{H_{1/3}}{(\frac{1}{f_{bw}} - \frac{1}{2}) \cdot U_{10}} = \frac{\frac{9}{16} \cdot H_{1/3}}{k_w(\frac{1}{f_{bw}} - \frac{1}{2}) \cdot U_{10}} \tag{6.80}$$

Dissolution from the surface slick will occur only during the calm period.

Droplet sizes and densities change due to dissolution. In addition to recording the volume of oil that dissolves, the program adjusts the size and buoyancy of the new droplet.

If we let V_{dis} be the dissolvable volume of oil and V_{inert} be the inert volume of oil within a subsurface droplet of volume $V_{droplet}$ (all in m^3) such that:

$$V_{droplet} = V_{dis} + V_{inert} \tag{6.81}$$

and:

$$V_{dis} = \sum_j V_j \tag{6.82}$$

where component j is the j^{th} aromatic distillation cut, then aromatics comprise the dissolvable volume of the droplet, which changes over time; all the other SARA cuts (saturates, resins, asphaltenes) make up the inert volume of the droplet, which does not change over time. A necessary assumption is that different volume fractions do not act as solvents for other fractions. This is not strictly true considering, for example, that aromatics can act as a solvent for asphaltenes, so volume may not be conserved in that circumstance.

Similarly, let ρ_{dis} and ρ_{inert} be the respective densities of dissolvable and inert volumes of oil (in kg/m^3), with:

$$\rho_{soluble} = \sum_j \alpha_j \rho_j \tag{6.83}$$

where α_j is the proportion of component j that is soluble.

Should $\rho_{soluble}$ be ρ_{dis}? DAH

Because the inert volume V_{inert} (in m^3) is fixed and does not change with time, we can define a new state variable representing the fraction of dissolvable volume:

$$X(t) = \frac{V_{dis}(t)}{V_{inert}} \tag{6.84}$$

with:

$$V_{droplet} = V_{inert} \cdot (X + 1) \tag{6.85}$$

Then we can calculate mass of the droplet $m_{droplet}$ (in kg) at time t by:

$$m_{droplet}(t) = \rho_{dis}(t) \cdot V_{dis}(t) + \rho_{inert} \cdot V_{inert} \tag{6.86}$$

and:

$$m_{droplet}(t) = V_{inert} \cdot (\rho_{dis} \cdot X + \rho_{inert}) \tag{6.87}$$

Then the mass-rate equation can be written as:

$$\frac{d}{dt} m_{droplet} = \frac{d}{dt}(\rho_{dis} \cdot V_{dis}) = \frac{d}{dt}(\rho_{dis} \cdot X) = -N \cdot A_{droplet} \tag{6.88}$$

where:
N is the mass transfer rate in kg/m^2/s and
$A_{droplet}$ is the surface area of the droplet (in m^2).

The two time-derivative terms, $\frac{d}{dt}\rho_{dis}$ and $\frac{d}{dt}X$, vary for different reasons. The first derivative reflects the fact that the dissolvable volume fraction is changing its composition over time; the second derivative reflects the fractional loss of the total dissolvable volume. In the simplest case, the dissolvable fraction density does not change significantly and its derivative can be neglected. This is the option used by Stevens et al. (2015) in their binary mixture model. This seems a reasonable approximation if one considers the longer-term changes in the oil, since:

$$\int \frac{d\rho_{dis}}{dt} \ll \int \frac{dX}{dX} \tag{6.89}$$

as the unitless, dissolvable fraction (f_{dis}) approaches zero.
The mass transfer rate (N) in kg/m^2/s is a function of the dissolvable oil concentration (c_{dis}):

$$N = \frac{k_w}{K_{ow}} \cdot c_{dis} = \frac{k_w}{K_{ow}} \cdot \rho_{dis} \cdot \frac{X}{X+1} \tag{6.90}$$

Combining Equations 6.88 and 6.90 gives:

Confirm that correct equations are referenced. DAH

$$\frac{d}{dt}X = -\frac{k_w}{K_{ow}} \cdot \frac{X}{X+1} \cdot A_{droplet} \tag{6.91}$$

For an oil droplet assumed to be approximately a perfect sphere, the surface area also can be related to the state variable (X):

$$A_{droplet} = 4\pi \cdot \left(\frac{3}{4} \cdot \frac{1}{\pi}\right)^{2/3} \cdot V_{inert}^{-1/3} \cdot (X+1)^{2/3} \tag{6.92}$$

and:

$$\frac{d}{dt}X(t) = \beta \cdot \frac{X(t)}{(X+1)^{1/3}} \tag{6.93}$$

where:

$$\beta = -5 \cdot \frac{k_w}{K_{ow}} \cdot V_{inert}^{-1/3} < 0 \tag{6.94}$$

6.5.8 Surface Slicks

For surface slicks, the oil droplet size and composition will not change significantly over the model time step (Δt).

The droplets created by the breaking waves will be separated into the appropriate "bin" according to the droplet diameter ($\delta_{droplet}$). The variable n_i represents the number of droplets in bin i ($i = 1, \ldots, i_{max}$), where each bin contains droplets that are the same size and droplets differ in size from one bin to another. The value of n_i is generated from the surface dispersion formulas.

For subsurface droplets, the total oil volume in the water column (V_{wc}) is given by:

$$V_{wc} = \sum_i V_i \cdot n_i \tag{6.95}$$

where V_i is the volume of a droplet in bin i (in m^3).

Note that we assume the SARA composition at the beginning of the time step is the same for each drop, regardless of its size. Then:

$$X_{wc}(t) = X_i(t) \forall i \tag{6.96}$$

so that the fraction volume that is dissolvable in all subsurface droplets is the same as the dissolvable fraction volume in each bin (i).

Define X_{wc} and X_i. DAH

Then:

$$X_i(t + \Delta t) = X_i(t) + f_{wc,i} \cdot \beta_i \cdot \frac{X_{wc}(t)}{(X_{wc}(t) + 1)^{1/3}} \cdot \Delta t \tag{6.97}$$

where β is ...

describe β?

The oil volume dissolved from droplets ($\Delta V_{droplet}$) in the time step (Δt) in m^3 is:

$$\Delta V_{droplet}(\Delta t) = \sum_i n_i \cdot V_{inert,i}[X_i(t + \Delta t) - X_{wc}(t)] \tag{6.98}$$

Volume lost directly from the surface slick (ΔV_{surf}) in m^3 can be approximated as:

$$\Delta V_{surf}(\Delta t) = n_i \cdot \rho_{dis} \cdot \Delta t \tag{6.99}$$

and total volume dissolved (ΔV_{tot}) in m^3 is:

$$\Delta V_{tot} = \Delta V_{droplet} + \Delta V_{surf} \tag{6.100}$$

6.5.9 Subsurface Oil

As t gets larger, $X_i(t)$ is not the same for all droplet sizes.

The boundary condition at the initial time is:

$$X = X_0 \text{ at } t = 0 \tag{6.101}$$

with the limiting condition that:

$$X \leq X_0 \forall t > 0 \text{ and } X \to 0 \text{ as } t \to \infty \tag{6.102}$$

The dynamic equation can be written as:

$$\int_{X_0}^{X} \frac{(X + 1)^{1/3}}{X} dX = \beta t \tag{6.103}$$

An exact, but complex, solution exists for the above equation. However, since its construction already incorporates many idealizations, added complexity probably does not contribute to a more accurate model. Therefore, rather than using the complex solution, an approximate solution can be found by considering the limits of the integrand, where:

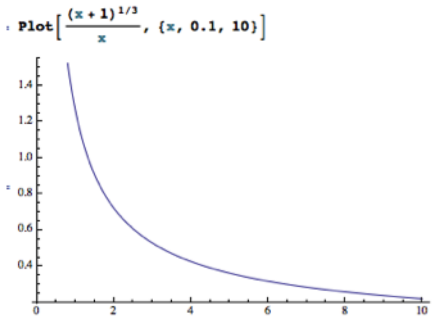
$$\frac{(X + 1)^{1/3}}{X} \to \frac{1}{X} \text{ as } X \to 0 \tag{6.104}$$

and:

$$\frac{(X + 1)^{1/3}}{X} \to \frac{1}{X^{2/3}} \text{ as } X \to \infty \tag{6.105}$$

as shown in Figure 6.9.

If we use:



```
Plot[(x + 1)^1/3 / x, {x, 0.1, 10}]
```

Figure 6.9: Approximate solution for the dynamic equation

$$\frac{(X+1)^{1/3}}{X} \rightarrow \frac{1}{2^{2/3}}(\frac{1}{X}H(1-X) + \frac{1}{X^{2/3}}H(X-1)) \tag{6.106}$$

where $H(X)$ is the Heaviside function, this matches the original and approximate function at $X = 1$.

Then, the equation can be approximately solved for low-solubility oils, for which $X_0 < 1$, as:

$$\int_{X_0}^{X} \frac{(X+1)^{1/3}}{X} dX = \int_{X_0}^{X} \frac{1}{2^{2/3} \cdot X} dX = \frac{1}{X^{2/3}} \cdot ln(\frac{X}{X_0}) \tag{6.107}$$

This yields the solution:

$$X = X_0 e^{2^{2/3}\beta t} = X_0 e^{1.5\beta t} \tag{6.108}$$

If, however, the initial condition is such that the dissolvable volume is larger than the inert volume, as it is for high-solubility oils such as gasoline, then $X_0 > 1$, in which case we have:

$$\int_{X_0}^{X} \frac{(X+1)^{1/3}}{X} dX \simeq \int_{X_0}^{X} X^{-2/3} dX \tag{6.109}$$

This yields the solution:

$$X = (X_0^{1/3} + \frac{1}{5}\beta t)^3 \tag{6.110}$$

For X close to 1, the answer is between the two extremes.

6.6 Biodegradation

For each pseudocomponent j in a single droplet of radius $r_{droplet}$ at a particular time step t, the change in biodegradation mass (m_j) in kg is:

Cite ADIOS3? DAH

Elsewhere in the document, Δt is used for time step. DAH

$$\frac{d}{dt} m_j = -k_j \cdot 4\pi r^2_{droplet}(t) \cdot \frac{m_j(t)}{\sum_i m_i(t)} \tag{6.111}$$

where:
k_j is the biodegradation rate constant for pseudocomponent j (in kg/s),
$4\pi r^2_{droplet}(t)$ is the surface area of the droplet at time t,
$\frac{m_j(t)}{\sum_i m_i(t)}$ is the mass fraction of pseudocomponent j at time t, and
$i = 1, ..., n$ where n is the number of pseudocomponents.

Should the 4 be included in this definition? Change to $A_{droplet}$ here and in equation, as defined in 6.5.5? DAH

The cumulative biodegradation for pseudocomponent j from initial time step t_0 through current time step t_m, using the integrated rate law, is:

$$m_j(t_{m+1}) = \int_{t_0}^{t_m} \frac{d}{dt} m_j(t) = m_j(t_0) \cdot e^{K_{droplet}} \cdot t_{m+1} \tag{6.112}$$

where:

$$K_{droplet} = \frac{-k_j \cdot 4\pi r_{droplet}^2}{\sum_i m_i(t_{m+1})} \tag{6.113}$$

and:
m is the number of time steps taken,
t_{m+1} is the next time step to be taken, and
$i = 1, ..., n$ where n is the number of pseudocomponents.

Define $K_{droplet}$? DAH

6.7 Sedimentation (OPA Formation)

Cite ADIOS2? DAH

Research has indicated that adhesion of oil to sedimentary particles is an important long-term process for removal of stranded oil from marine shorelines (Bragg and Owens 1995). For most open water spills, sedimentation (defined here as adhesion of oil to solid particles in the water column) is usually not an important process for removal of floating oil during the time frame modeled by GNOME. However, in areas of high concentrations of suspended particulate matter, adhesion of the oil to particles that may eventually settle to the bottom or disperse away from the slick may play a key role in the oil slick mass-balance equations.

The physical process of sedimentation is quite complex and has been only fragmentarily researched. Studies by Poirier and Thiel (1941) and Hartung and Klinger (1968) indicated that the process was affected by type and size of the suspended material, salinity of the water, and sulfur content of the oil. Other studies have suggested that oil droplet size (Payne, Kirstein, McNabb, Lambach, Redding, Jordan, Hom, DeOliveira, Smith, Baxter, and Gaegel 1984), which is directly related to oil viscosity and wave energy, plays a significant role. Twomey (1977) has proposed that collection efficiency of particulates is determined by the Stokes number (Stk), with:

ref is not in bibtex file, should it be Payne et al. 1987?

$$Stk = \frac{2r_{droplet}^2 \cdot v_{relative}}{9 \cdot \nu_w \cdot r_{sediment}} \tag{6.114}$$

where:
$r_{droplet}$ is the radius of the oil droplet (in m),
$r_{sediment}$ is the radius of the sediment particle (in m),
$v_{relative}$ is the relative velocity between oil and sediment particles (in m/s), and
ν_w is the water kinematic viscosity (in cSt).

Note that 1 $m^2/s = 10^6$ cSt.

This would imply that collection efficiency is inversely related to the particle size, which is what Poirier and Thiel (1941) found.

GNOME uses a formula proposed by Science Applications International (Payne et al. 1987). It is similar to the Applied Science Associates (1995) formula but incorporates the effects of water turbulence. The mass of oil lost per unit water volume per unit time ($\dot{m}_{oil}$) in kg/m^3/s is:

This paragraph (below) is from Sedimentation from ADIOS2.docx and it uses a different formula - which one should be kept? - I think it is a step on the way to the final formula

$$\dot{m}_{oil} = 1.3\sqrt{\frac{\epsilon}{\nu_w}} \cdot k_a \cdot c_{oil} \cdot c_{sed} \tag{6.115}$$

where:
ϵ is the energy dissipation rate for the surface water (in m^2/s^3),
k_a is the sticking parameter (in m^3/g),
ν_w is the water kinematic viscosity,
c_{oil} is the concentration of oil particles in the water, and

c_{sed} is the concentration of sediment particles in the water.

The energy dissipation rate (ϵ) can be estimated by the breaking wave energy calculations described in the dispersion section (Section 6.4). Typical observed values are of the order 0.1 erg/cm^3/s. The parameter k_a depends on the type and size of the particles. GNOME uses $k_a = 10^{-7}$ m^3/g. To get the sedimentation rate per unit area of the slick, one has to integrate this rate over the depth to which the breaking waves drive the oil droplets. According to Delvigne and Sweeney (1988), this is approximately 1.5 times the height of the breaking wave. Only droplets larger than 70 µm are considered, since droplets smaller than this are assumed not to refloat anyway, and are effectively removed from the slick whether or not they adhere to sediment particles.

Units for ϵ above are listed as m^2/s^3. DAH

The combined oil-sediment particle will typically have different buoyancy than that of either the oil droplet or the sediment particle alone. Typically, the buoyancy will be negative and turbulence will be required to keep the particle from settling to the bottom. The terminal falling velocity of a discrete non-flocculating particle can be determined by Stokes' Law. However, most suspensions of concern with respect to significant sedimentation from the spill will contain particles that are neither discrete nor non-flocculating.

According to Uchrin and Weber (1980), it is impossible to generate a fall velocity deterministically for such a case based only on particle size. Rather than attempt to model something that is clearly site-specific, GNOME considers any of the oil that adheres to the sediment to be effectively removed from the slick, and no further analysis on this portion of the oil is performed.

The bigger droplets will refloat and rejoin the surface slick. Sedimentation operates only on these bigger oil droplets that will refloat. These droplets will have a range of diameters from 70 µm to the thickness of the surface oil slick (δ_{slick}) in meters. GNOME uses an average droplet size (δ_{ave}) in meters of:

This is the "new" text for Sedimentation from Bill – we need to reconcile this with the previous section – there should be only one version! – Caitlin to reconcile

$$\delta_{ave} = 0.613 \cdot \delta_{slick} \tag{6.116}$$

with an average rise velocity (v_{rise}) in m/s given by:

$$v_{rise} = \delta_{ave}^2 \cdot g \cdot \frac{1 - \frac{\rho_{oil}}{\rho_w}}{18 \cdot \nu_w} \tag{6.117}$$

where:
ρ_{oil} is the density of oil (in kg/m^3),
ρ_w is the density of water (in kg/m^3), and
v_{rise} is the rise velocity (in m/s).

A typical rise velocity is on the order of cm/s. If a typical wave height is on the order of a meter, then the droplets will all refloat within about 100 seconds – well within one time step.

The oil droplets are assumed to be dispersed by breaking waves uniformly (uniform distribution in the vertical direction) to a depth of 1.5x the wave height.

The net mass loss rate (Q_{sed}) in kg/s is based on the Payne (SAI) model that was used in ADIOS2:

Ref needed? DAH

$$Q_{sed} = 1.6 \cdot k_a \sqrt{\frac{H_w \epsilon f_{bw}}{\rho_w \nu_w}} \cdot c_{oil} \cdot c_{sed}, \text{ with } Q_{sed} \geq 0 \tag{6.118}$$

where:
$k_a = 0.0001$ m^3/kg, a "sticking" parameter for the oil,
H_w is the wave height (in m),
ϵ is the wave energy dissipation rate per unit area (in J/m^2),
f_{bw} is the fraction of breaking waves (unitless),
$\rho_w \nu_w = 0.001$ N $\cdot$ s/m^2, the dynamic viscosity of water,
$c_{sed} = \{0.5, 0.05, 0.005\}$ kg/m^3 for {muddy river, estuary, open ocean} respectively, the concentration of sediment, and
c_{oil} is the concentration of oil (in kg/m^3) such that:

Different units than those above. DAH

$$\frac{\delta_{slick} f_{cov} \rho_{oil} (1 - Y)}{H_w^2} \cdot (H_w - \Delta t \cdot v_{rise}) \text{ with } c_{oil} \geq 0 \tag{6.119}$$

where:
f_{cov} is the surface fraction coverage (unitless),
Y is the water fraction of emulsion (unitless), and
Δt is the time step (in s).

6.7.1 Density of OSAs

[commented out material in .tex doc]

Add material used in current version of GNOME2 or delete this section. DAH

6.8 Emulsification

For many crude oils and some refined products, weathered oil is likely to reach a stage where water droplets are dispersed into the oil, forming a water-in-oil emulsion or mousse. This is exactly the reverse of dispersion where oil droplets are dispersed into water. Although mousse formation is an important mechanism for oil weathering, the process is poorly understood.

Laboratory and field studies seem to indicate that, in most cases, the amount of wax and asphaltenes in the oil play an important role in whether or not emulsification will occur. Evaporation changes the ratio of waxes to asphaltenes in the slick, so it is common to have an oil that will not emulsify when it is still fresh, but will emulsify after it has partially weathered.

To predict when and if a particular oil will emulsify, GNOME calculates the percentage of oil necessary to evaporate before emulsification begins. This constant is calculated internally and is based on field observations and laboratory studies. The user can override the internal calculation by entering an emulsification constant.

The turbulent energy in the surrounding water can cause small droplets of water to get mixed into the oil, forming a water-in-oil emulsion. The amount of water and water droplet size distribution affects the viscosity and stability of the emulsion. A fully emulsified, stable emulsion may contain eighty to ninety percent water.

Not all oils will emulsify and many crude oils will only emulsify after a certain amount of evaporation or photo-oxidation has occurred. The onset of emulsification is important for cleanup decisions and it is therefore useful for a weathering model to have the capability to forecast this event. The GNOME oil database has observational data on this from actual spills for a handful of oils and some results from lab data from artificially weathered oil experiments. Estimates can be made on new oils by comparing them with tested oils of similar composition, but, in general, this is a difficult parameter to quantify.

For the cases where there is insufficient SARA data in the database for the new algorithm, GNOME uses a first order equation in interfacial area (Eley et al. 1988)

$$\frac{dS}{dt} = k_{emul}\left(1 - \frac{S}{S_{max}}\right) \tag{6.120}$$

where the water uptake coefficient, k_{emul} is sensitive to wave energy. Here S and S_{max} are the oil-water interfacial area and maximum interfacial area, respectively. Water fraction, Y, is then related to interfacial area and maximum emulsion droplet diameter, d_{max}, by the equation

$$Y = \frac{Sd_{\max}}{6 + Sd_{\max}} \tag{6.121}$$

Often a more critical parameter for cleanup decisions is the timing of the onset of emulsification. Not all oils will emulsify and some oils will emulsify only after they have weathered to a certain extent.

For a limited number of oils, the oil database contains experimental data on the onset of emulsification, correlated to the asphaltene fraction. The onset of emulsification for other crude oils is estimated by interpolating the emulsification-asphaltene data for oils where experimental measurements exist. If the asphaltene fraction is not known, an estimate is made based on the oil API value. Alternatively, the user may enter directly a time or evaporative fraction for the initiation of emulsification. The default for most refined products is to assume that emulsification does not occur.

Insert ADIOS2 algorithm here: see Caitlin

6.8.1 New Algorithm

Emulsification is an extremely complex process that generally requires highly detailed information on the chemical structure of the oil and the specific environmental conditions in order to generate accurate behavioral forecasts. Fortunately, response questions can often be answered by using a simplified approach that recognizes and maps the expected uncertainty in model predictions. GNOME uses a simplified emulsification submodel that relies on commonly available oil and environmental data to forecast: conditions when a water-in-oil emulsification is likely to happen, the stability of the resulting emulsion, water fraction of the emulsion, and change in viscosity.

The submodel matches past theoretical studies on emulsification with clustering approach systems developed from a large experimental database on emulsion properties collected by Environment Canada and others. Due to the differences between laboratory measurements and real events, the variation among organizations and methods in classification of hydrocarbons into the appropriate SARA component, and the limited input for real spills, the submodel is designed to provide only expected values or value ranges along with quantitative uncertainty, rather than precise answers. The submodel considers insertion of water droplets into the oil by breaking waves as emulsification onset, which depends principally on the oil hydrocarbon structures and stability of the emulsion.

6.8.2 Introduction

Emulsification has a significant impact on response options for major surface oil spills. Emulsified oil typically has both increased volume and increased viscosity. Standard cleanup techniques (skimming, burning, surfactants) may all be reduced in effectiveness if oil emulsifies. However, not all oils emulsify, and the stability of the formed emulsion is not the same in all cases. Some oils may not emulsify initially but may do so after a period of weathering. Accurately predicting when emulsification will occur and the properties of the emulsion has been a continuing challenge for the modeling community. Both the underlying physical and chemical formulas and the important environmental and petroleum properties needed for modeling remain matters of dispute. Adding to the challenge has been a dearth of reliable and complete laboratory and field data to validate the models. More data and better measurement capabilities will increase the potential for more accurate results.

6.8.3 Mackay Formulation

Most widely used spill models currently employ some version of the Mackay first-order rate law model (Mackay et al. 1980). For example, the previous NOAA model, ADIOS2 (Lehr et al. 2002) calculates water mass fraction of the emulsion (Y) by:

$$\frac{dY}{dt} = k_{emul}\frac{\nu_0}{\nu(t)}U_{10}^2(Y_{max} - Y) \tag{6.122}$$

where:

Define k_{emul}? DAH

ν is the kinematic viscosity,
Y_{max} is the maximum water fraction, based on the oil's initial asphaltene fraction, and
U_{10} is the surface wind speed.

Such an approach does not allow for the common phenomenon of dewatering, seen in unstable and mesostable emulsions. Xie et al. (2007) correct for this by adding a term to account for this process.

The real challenge for traditional models is estimating k_{emul} because this term is a complicated parameter dependent on a large mixture of environmental and oil chemistry variables, many not easily determinable in a real spill incident. SINTEF (e.g., Daling et al. 1990) and others simply have favored direct empirical measurement of different oils under various sea conditions to generate this term. Although this may be practical for a limited geography and sets of oils, it is not the solution for a universal model.

6.8.4 Time Requirements for Emulsion Formation

Breaking waves provide the energy necessary for insertion of water into the slick and for the breakup of large drops into smaller ones. Daling et al. (1990) showed that mechanical energy processes are cumulative in time and contribute not only to water insertion but also to entrained water droplet size distribution, with the droplet diameter tending to decrease in size over time for stable emulsions. This process, while not instantaneous, happens quite rapidly if the oil is capable of emulsification. Fingas and Fieldhouse (2004) calculate the time (in s) for an emulsion to form (Δt_{emul}) as:

$$\Delta t_{emul} = 1630 + \frac{450}{H_w^{3/2}} \tag{6.123}$$

where H_w is the significant wave height in m.

For a one-meter significant wave height, the time to form an emulsion is approximately 30 minutes according to the above formula. Final emulsion stability may take longer. While there are different types of emulsions, this algorithm considers only those models where asphaltenes are the final stabilizing agent, a process that can take some time. Small laboratory experiments by Mohammed et al. (1993) indicate that even for such small experiments, the asphaltene-water layer can take many hours to form. Fingas and Fieldhouse (2004) report that asphaltene migration to the oil-water interface can be an extremely slow process, taking a month or longer in certain cases. In the meantime, resins tend to diffuse first to the interface and later become replaced by asphaltenes (Magual et al. 2005). Bobra (1992) estimated that the water drops will eventually be coated with an asphaltene layer 0.03 µm (approximately ten molecules) thick. This means that the properties of the emulsion during a spill incident are likely to vary over time, with unstable emulsions losing water and stable emulsions increasing in stability and related properties such as viscosity.

6.8.5 Water Droplet Size

Droplet sizes will vary in the emulsion, following a log-normal size distribution (Eley et al. 1988). Johansen et al. (2013) calculate a mean droplet diameter for oil-in-water emulsion based on the Weber and Reynolds number and by assuming that stability in the droplet size distribution would be reached in the inertial subrange of the Kolmogorov scale (Kolmogorov 1949). Boxall et al. (2011), noting the higher viscosity of oil, suggest that droplet size stability for water-in-oil emulsions will be reached only in the viscous subrange. Given that assumption, a median droplet size can be estimated as a function of the Weber and Reynolds number. If we define:

ν_{oil} as kinematic viscosity of oil,
σ_{ow} as oil-water interfacial surface tension,
ϵ as energy dissipation rate from breaking waves,
δ_{th} as average slick thickness (thick part, not sheen), and
u_{*w} as $(\epsilon H_w)^{1/3}$ wave 'turbulent energy' velocity,

Check this for accuracy. DAH

then the Reynolds number (Re) can be defined as:

$$\mathrm{Re} = \frac{u_{*w}\delta_{th}}{\nu_{oil}} \tag{6.124}$$

and the Weber number (We) defined as:

$$We = \frac{\rho_{oil} u_{*w}^2 \delta_{th}}{\sigma_{ow}} \tag{6.125}$$

Recognizing his large contribution to the subject area, GNOME defines the Fingas number (Fi) as:

$$Fi = \frac{\sqrt{\mathrm{Re}}}{We} = \frac{\sigma_{ow}}{\delta_{th}\rho_{oil}\sqrt{\nu_{oil}\epsilon}} \tag{6.126}$$

According to Boxall et al. (2011), the Sauter mean droplet diameter (δ_{50}) can be estimated as:

$$\delta_{50} \simeq \delta_{th} \cdot Fi \tag{6.127}$$

The Fingas number correlates to the reciprocal of the average number of water droplets in a vertical slice of the slick. In general, the smaller the Fingas number, the smaller the Sauter mean diameter and the more stable the emulsion. According to Schramm (1992), stable emulsions will have a water droplet mean size of approximately 5 µm, whereas water droplet mean sizes of less stable emulsions will be at least twice this size. Also, the average droplet size should reduce inversely with the square root of the viscosity, which is consistent with the very limited data of Anisa and Nour (2010).

It is possible to estimate a range for the Fingas number for a spill that has formed a stable emulsion. Assuming a minimum emulsion slick thickness of 100 µm, or a maximum of 10,000 µm, and a Sauter mean between 1 and 10 µm, we get:

$$10^{-4} < Fi < 10^{-1} \tag{6.128}$$

The densities and interfacial surface tension of most oils are similar, so it is also possible to construct an estimate from Fingas limits of an acceptable kinematic viscosity range for an oil likely to form a stable emulsion under normal seas. A result of between 10 and 10,000 cSt is consistent within an order of magnitude with the findings of Fingas and Fieldhouse (2003).

The final droplet size distribution will be log-normal (Eley et al. 1988), but can also be approximated as a Roslin-Rammler distribution (Johansen et al. 2013).

6.8.6 Maximum Water Fraction

According to Desmond and Weeks (2014), if the water droplets are treated as hard spheres randomly assigned, the largest water volume fraction that can be achieved is 0.88. Given the densities of most oils, this would correspond to Y_{max} of 0.89. Because water droplets are not hard spheres, some oils have larger water mass fractions.

While the initial presumption might be that water fraction is highly dependent on Fingas number, the actual data indicate only a very weak dependence. Examination of high-water-content emulsions (Fingas and Fieldhouse 2004) show instead that asphaltene/resin ratio (Z_{ar}) is more important. A preliminary fit yields:

$$Y_{max} = 0.61 + 0.5Z_{ar} - 0.28Z_{ar}^2 \qquad 0 \leq Y_{max} \leq 0.9 \tag{6.129}$$

Figure 6.10 shows the maximum water percentage as a function of Z_{ar}.

remake this figure

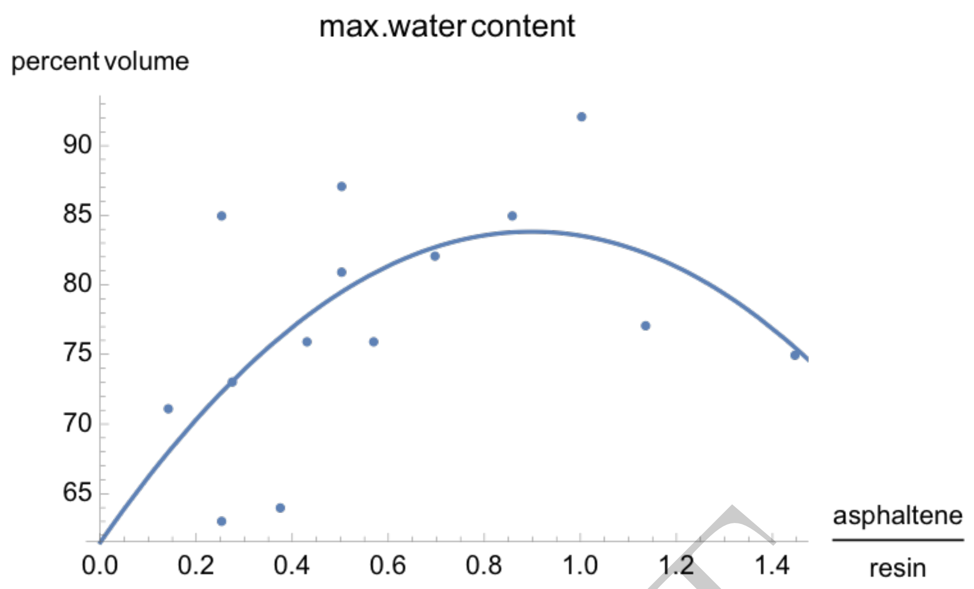


Figure 6.10: Maximum water percentage as a function of asphaltene/resin ratio (Z_{ar})

6.8.7 SARA Formulation

Add notes supplied by Caitlin O'Connor – new algorithm

6.8.8 Emulsion Stability

It is important to separate emulsion stability, which is an operational parameter related to the frequency of water loss, from an emulsion stability prediction parameter that is a function of the chemical structure of the oil. GNOME assumes that the former is directly related to the emulsion rate of water loss in calm conditions. The Environment Canada database records for several oils the water content of a fresh emulsion and for the same emulsion a week later. This standard has been adopted by GNOME. Stability (S_b) is defined as the ratio of entrained water in the oil after one week (τ_{week} = 604,800 s) compared to the initial amount of entrained water when the oil first emulsifies. A completely stable oil will lose no water ($S_b = 1$). An unstable emulsion will lose all its water within the week ($S_b = 0$). Mesostable emulsions will have an S_b value in-between 0 and 1. Consistency with the definitions adopted by Fingas and Fieldhouse (2003) yields the following:

Stable	Mesostable	Unstable
S_b> 0.95	0.95> S_b >0.35	S_b<0.35

6.8.9 Modified Clustering Approach to Stability

Stability is apparently closely related to water fraction as both depend on similar characteristics of the emulsion (Aske et al. 2002; McLean and Kilpatrick 1997). More stable emulsions are usually associated with larger water fractions.

GNOME uses a combination of traditional and clustering models (Lehr 2017). Cluster rules are applied and, to the greatest extent possible, correspond to the underlying science. The following rules for stable emulsions, generally consistent with many cluster rule sets, have been adopted in a modified probabilistic form for the GNOME weathering model. As further development occurs in the model, the number of rules and exact rule limits may change.

- *Rule 1: Fingas number between 0.0001 and 0.1*
  The Fingas number (Lehr 2017) in most cases is strongly related to viscosity. Instability results with viscosities that are either too high or too low because low viscous oils cannot retain water long enough and highly viscous oils prevent water from entering in the first place (Fingas and Fieldhouse 2003).

- *Rule 2: Asphaltene mass fraction (f_{asph}) greater than 0.03 but less than 0.12*
  Asphaltenes play the chief role as indigenous surfactants in stabilizing emulsions (McLean et al. 1998; Issaka et al. 2015).

emulsify but violate the rules, and oils that meet the rules but do not emulsify. Hence, any model forecast will contain significant uncertainty, a common circumstance for complex environmental models. GNOME uses *Rule 4* to determine the on/off switch for weathering oil. The other three rules are then combined into a stability estimation function ($\widetilde{S}_b$), where:

$$\widetilde{S}_b = 1/\big(1 + \exp\big[\frac{10(1-3x)}{3}\big]\big) \tag{6.130}$$

with:

$$x = F(r, 0.6, 2) \cdot F(f_{asph}, 0.9, 2)\sqrt{\frac{Fi_{ref}}{Fi}} \tag{6.131}$$

$$F(a, b, c) \equiv \frac{1}{1 + (a-b)^{2c}} \tag{6.132}$$

Here, Fi_{ref} is a reference constant that depends on slick thickness and sea state. The widely scattered, measured values in Figure 6.11, comparing calculated with measured stability, illustrate the challenge in applying any functional fit to the data.

Is F fetch in equation? Also, if r is the asphaltene/resin ratio, change variable to Z_{ar}. Define $F(a,b,c)$? DAH

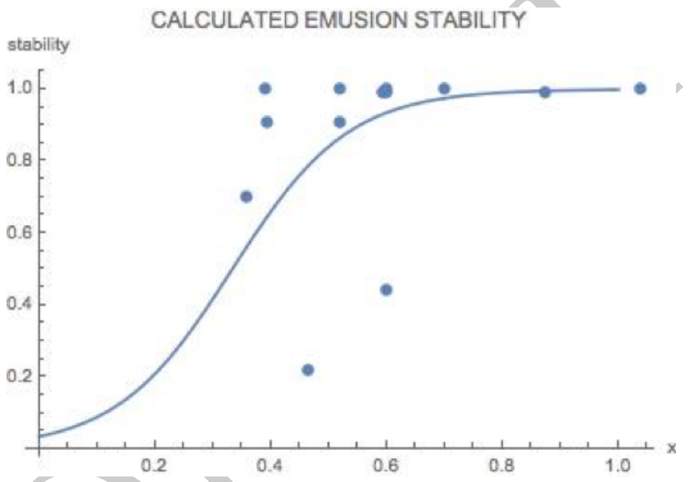


Figure 6.11: Emulsion stability calculated from equation for $\widetilde{S}_b$

remake this figure and correct spelling of "emusion"

6.8.11 Defining the Emulsion Equations

The water-ingestion equation modifies the traditional equation:

$$\frac{dY}{dt} = k_{emul}(Y' - Y)H(Y_{max} - Y) - k_{lw}Y \tag{6.133}$$

with:

$$k_{emul} = \frac{Bw}{\Delta t_{emul}} \quad k_{lw} = \frac{1 - \widetilde{S}_b}{\tau_{week}} \quad Y' = \frac{Y_{max}}{1 - 1/e} \tag{6.134}$$

Define k_{lw}. DAH

where H is the Heaviside function and the choice of the maximum water content (Y') allows the water content to reach a maximum at $t = \Delta t_{emul}$. Bw represents an on-off switch, the value of which depends on *Rule 4*.

Symbolized elsewhere as $H(X)$. DAH

What is the difference between Y' and Y_{max}? DAH

6.8.12 Emulsion Viscosity

Viscosity increases as the oil emulsifies according to the formula in Pal and Rhodes (1989):

$$\nu_{emul} = \nu_0 \cdot \left(1 + \frac{1.15Y}{1.187 - 1.15Y}\right)^{2.49} \tag{6.135}$$

where the viscosity of the emulsion (ν_{emul}) is a function of the unemulsified oil and water fraction.

kinematic viscosity? DAH

6.8.13 Discussion and Disclaimer

The incorporation of rule sets found in the clustering approach does allow a better match with laboratory data than traditional formulas do. A key limitation in this model, and all existing models, is inadequate and inconsistent characterization of oil structure. Significant improvement in our ability to predict and describe emulsification of spilled oil is likely dependent on improvement of our oil properties databases.

Chapter 7

Oil in Ice

7.1 Overview

Oil-ice interactions are very complex, and we do not know enough to model much of them. Even if we did, there is very limited data available from either remote sensing or models – particularly models. So we are limited to a very simple scaling approach.

The models available provide the following parameters:

- Fractional ice coverage
- Ice thickness (sometimes)
- Ice age (sometimes)
- Ice drift velocity

We have to make do with this small amount of information. In theory, we could use ice thickness as a proxy for other parameters that may matter, such as under-ice roughness, but we do not know how to do that yet.

We have an idea of the behavior of oil in ice with full ice coverage, and with no ice at all. So our first-order goal is to have algorithms that behave properly with full coverage and no ice, and provide continuous results in-between. This leads to what we call the 80-20 rule, similar to the approach applied by Sintef in the OSCAR model, and in RPS's OilMap.

Reference needed? DAH

Reference needed? DAH

If there is 20 % or less ice coverage, the oil behaves as it would with no ice coverage. If there is 80 % or more ice coverage, the oil behaves as it would with full ice coverage. If the coverage is in-between 20 % and 80 %, then the process is linearly interpolated between those values.

In some cases, the application of this rule is straightforward and in others, not so much.

7.2 Transport Algorithms

7.2.1 Currents

With 20 % or less ice coverage, the oil moves with the currents. With 80 % or more ice coverage, the oil moves with the ice. For the model to accomplish this, the currents are scaled down according to the ice coverage and the 80-20 rule. The ice movement is also scaled down according to this rule, but in the opposite direction. The net result is that with 50 % ice coverage, the oil moves at the average velocity of the ice and current velocities. At greater than 80 %, it moves with the ice, and at less than 20 % it moves with the water.

The code does no currently support any movement of oil under land fast ice, where there may be substantial currents relative to the ice. A "stripping" velocity approach may be considered in the future.

7.2.2 Wind Drift

The oil drifts with the wind as usual for low-ice coverage, and does not drift with the wind at all with full-ice coverage. At coverage in the middle range, there is an "ice-modified wind," with the velocity of the wind at a given location scaled by the ice coverage according to the 80-20 rule.

7.2.3 Diffusion

Diffusion with no ice is set by the user. With ice, we expect the oil to be "locked in," and diffusion to be very slow or zero.

Diffusion in GNOME is simulated with a random walk algorithm (Section 3.2.3) – at each time step, each element is moved a random amount, computed from the time step and the diffusion coefficient. In the case of ice, the 80-20 rule is implemented by scaling the net movement of the random walk by the ice coverage. So at 80% or more ice coverage, there is zero diffusion, and at less than 20% coverage, the diffusion is the same as open water, with the net movement scaled by linear interpolation between 80% and 20%

7.3 Weathering Processes

For the most part, weathering processes are performed on a per-element basis. It is assumed that with full (or nearly full) ice coverage, the oil is locked in and does not weather.

7.3.1 Spreading

Spreading is really a transport process, it is often included as a weathering process because the spread influences the other weathering processes. As such, while the physical process is spreading, what this component of the model does is predict the area of the slick exposed to weathering processes – notably evaporation.

As usual, it is assumed that no spreading occurs when the oil is "locked in" to the ice, and spreading occurs as usual when there is little ice present. For intermediate quantities of ice, the spreading rate can be modified by the percent coverage according to the 80-20 rule. So for a given time step, an increase in area is computed, and that increase is modified by the ice cover.

However, there may be an additional effect – in practice, the spreading is terminated at an empirical "terminal thickness." But if ice coverage increases, then the area exposed to the atmosphere may decrease – i.e., the exposed area should be about 50 % of the full area with 50 % ice coverage.

This is accomplished in the model with an ice-modified area – the area used by the other algorithms is adjusted according to the ice coverage at the location of the element and the 80-20 rule.

Currently there is no consideration of the spreading of very thick oil under an ice cover, as would occur with a release below the ice.

7.3.1.1 For the Future

In reality, the oil under the ice will get caught up in the pockets since the ice will not be smooth, at least not in the Arctic (although the situation is different in rivers and lakes). We are considering algorithms based on ice roughness and holding capacity, but there currently the ocean-ice models do not predict an ice roughness.

7.3.2 Evaporation

Evaporation is driven by water temperature, wind speed, and exposed area. The temperature is available from the oceanographic model, or provided by the user. The oil is exposed to the same wind when there is partial ice. But we can use the ice-modified area to compute the evaporation. The ice-modified area will be zero with full ice coverage, resulting in zero evaporation, and a reduced area, thus reduced evaporation, with partial ice coverage.

7.3.3 Dispersion

The dispersion is driven by wave climate - no dispersion if full ice cover, and regular dispersion for no ice. We expect no waves in full ice cover, and reduced wave energy in partial ice cover. So the dispersion algorithm is not modified, but rather the wave algorithm is modified to give an ice-modified wave field.

In the future, we may consider the turbulent energy under ice to generate dispersion – particularly if the currents under the ice are substantially different than the ice movement.

7.3.4 Wave Field

In theory, the wave field can be obtained from field measurements and/or a wave model. In either of these cases, the ice effects on the waves would already be taken into account.

However, operationally, the wave field often must be computed from the wind field. This is accomplished by using the ice-modified wind field, and using the same wave estimation algorithms currently used.

Thomson says: "Open water fetch is the strongest indicator of wave energy in the western Arctic Ocean." In the future,the wave module could consider the wind direction and location, look upwind to see how far it is to either ice or land, and use that fetch to compute the waves.

7.3.5 Sedimentation

Sedimentation is driven by dispersion. As the dispersion process is scaled down, sedimentation will simply follow.

7.3.6 Dissolution

Dissolution is similar to sedimentation – the droplet portion is driven by dispersion, so it will get scaled down automatically.

The surface film portion is driven by area and wind speed – the model uses the ice-modified area and ice-modified wind, so surface dissolution will scale down appropriately.

7.3.7 Biodegradation

Biodegradation also takes place when the oil is in droplet form in the water column – so it will be scaled down with the decreased dispersion.

7.3.8 Emulsification

Emulsification is also driven by the wave field (as well as viscosity which will tend to be high at low temperatures). So emulsification under the ice-modified wave field will be reduced similarly to dispersion.

Chapter 8

Releases

[No content in this section]

8.1 Surface Releases

8.2 Blowout Plume

8.3 Pipeline

8.4 Leaking Tank

8.5 Submerged Vessel

Chapter 9

Response/Cleanup Options

GNOME does not attempt to model the cleanup process. However, it does provide some tools that allow the user to input information about cleanup operations, for use in the mass-balance calculations and ICS 209 form inputs.

GNOME uses the same approach as ROC to estimate the amount of oil cleaned up. This involves defining the efficiency of the proposed cleanup procedure. Efficiency in GNOME is based only on sea state (wave height) and uses graphs in the ROC manual (Figures 1 and 5) to estimate default efficiency. GNOME does not use the additional weathering state of the oil parameter as does ROC. The user may modify the default efficiency parameter in the cleanup input screen.

Ref needed? DAH

Ref needed? DAH

9.1 Skimming (Mechanical Cleanup)

9.1.1 Description

Two ways to recover floating oil are by using either mechanical skimming devices or specially designed sorbent material to pick up the slick. Sorbents are materials that are designed to either cause the oil to stick to them or have the oil penetrate into them. They are usually used for small spills and sheens or to clean environmentally sensitive areas. Most commercial sorbents are made from polypropylene fibers but they can be made from other synthetic fibers as well as natural inorganic or organic materials.

Skimming devices are often used in connection with collection booms towed by recovery vessels. The booms help to direct the floating oil into the skimmer. The towing speed of the system must be slow enough such that oil will not be entrained underneath the booms. Typically, this reduces the towing speed to around a knot or less. Skimming efficiency is dependent on oil thickness and sea state. A rough estimate of conditions where skimming is most practical is shown in Figure 9.1.

For Fig. 9.1: Add Allen ref to bib file and correct spelling of "re-comended". DAH

Oil that is collected by skimmers must be stored, separated from any accompanying water, and either disposed of or recycled. All these factors must be considered when estimating the efficiency of mechanical recovery. For large spills, it is unlikely that skimmers will be able to recover more than a small percentage of the floating oil.

9.1.2 Algorithm

Section on 5 categories of skimmers?

The user enters either the rate or the net oil-water volume skimmed and the duration of the cleanup operation. These are related by:

$$V_{skim} = V'_{skim} \cdot \Delta t \tag{9.1}$$

where:
V_{skim} is the oil-water volume skimmed (in m^3), and
V'_{skim} is the gross skim volume rate (in m^3/s).

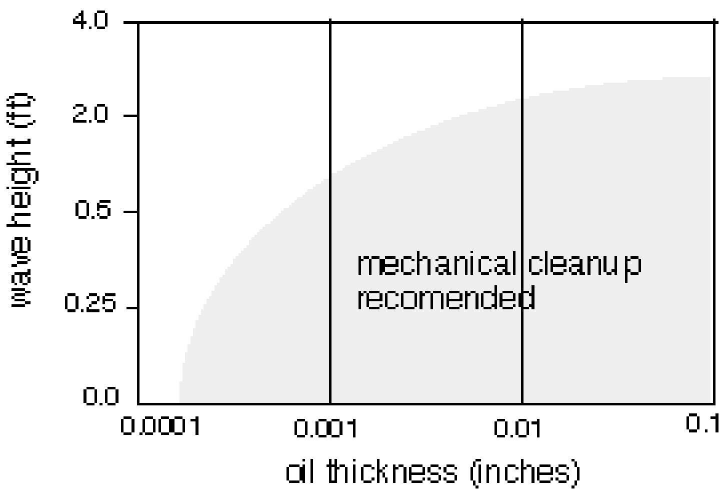


Figure 9.1: Wave height versus oil thickness (adopted from Allen)

Note that this equation represents both oil and water skimmed. GNOME needs to compute Net'_{skim}, the net *oil*-removal rate, not the removal rate of oil and water. This is the quantity that the screen needs to return to the computational core:

$$\text{Net}'_{skim} = \text{eff}_{skim} \cdot (1 - Y)V'_{skim} \tag{9.2}$$

where eff_{skim} (skimmer efficiency) $= 0.9 - 0.35H_w$, if $H_w \leq 2$; otherwise, $\text{eff}_{skim} = 0.2$.

9.2 In-Situ Burning

9.2.1 Description

In-situ burning is a cleanup technique that removes oil by burning it while it is still floating on the water. Generally, the minimum thickness for fresh oil to burn is 2–3 mm. For emulsified oil, the minimum thickness is closer to 5 mm. Oil that is allowed to spread without restriction usually will quickly become too thin to sustain a burn. It therefore becomes necessary to collect the oil by using specially designed fire-proof booms towed through the slick. Figure 9.2 shows a typical boom operation.

GNOME will let you run a burn scenario even under conditions where a real burn would be difficult or impossible. It is difficult to ignite emulsified oil that is more than 25% water unless an emulsion breaker is used first. Towed booms usually will not hold oil if the current normal to the boom is greater than 1 knot. Waves greater than 1–2 m or winds stronger than 40 knots also can make burning impractical.

For mass-balance purposes, GNOME considers all of the boomed oil to be consumed in the burn. In reality, a small percentage of the oil will remain as residue. This residue will have very different characteristics than those of the unburned oil. GNOME does not model properties of the residue.

Add ref? DAH

9.2.2 Algorithm

The user enters the area of boomed oil (A_{burn}) and the initial thickness of the oil ($\delta_{burn}(t_0)$). The total oil (or emulsion) volume available for burning ($V_{burn}(t_0)$) is then:

$$V_{burn}(t_0) = A_{burn} \cdot \delta_{burn}(t_0) \tag{9.3}$$

The oil thickness must be greater than 2 mm. The burning stops when the boomed oil reaches 2 mm. The burn rate (Net'_{burn}) is given by ROC formula 14:

Ref needed? DAH

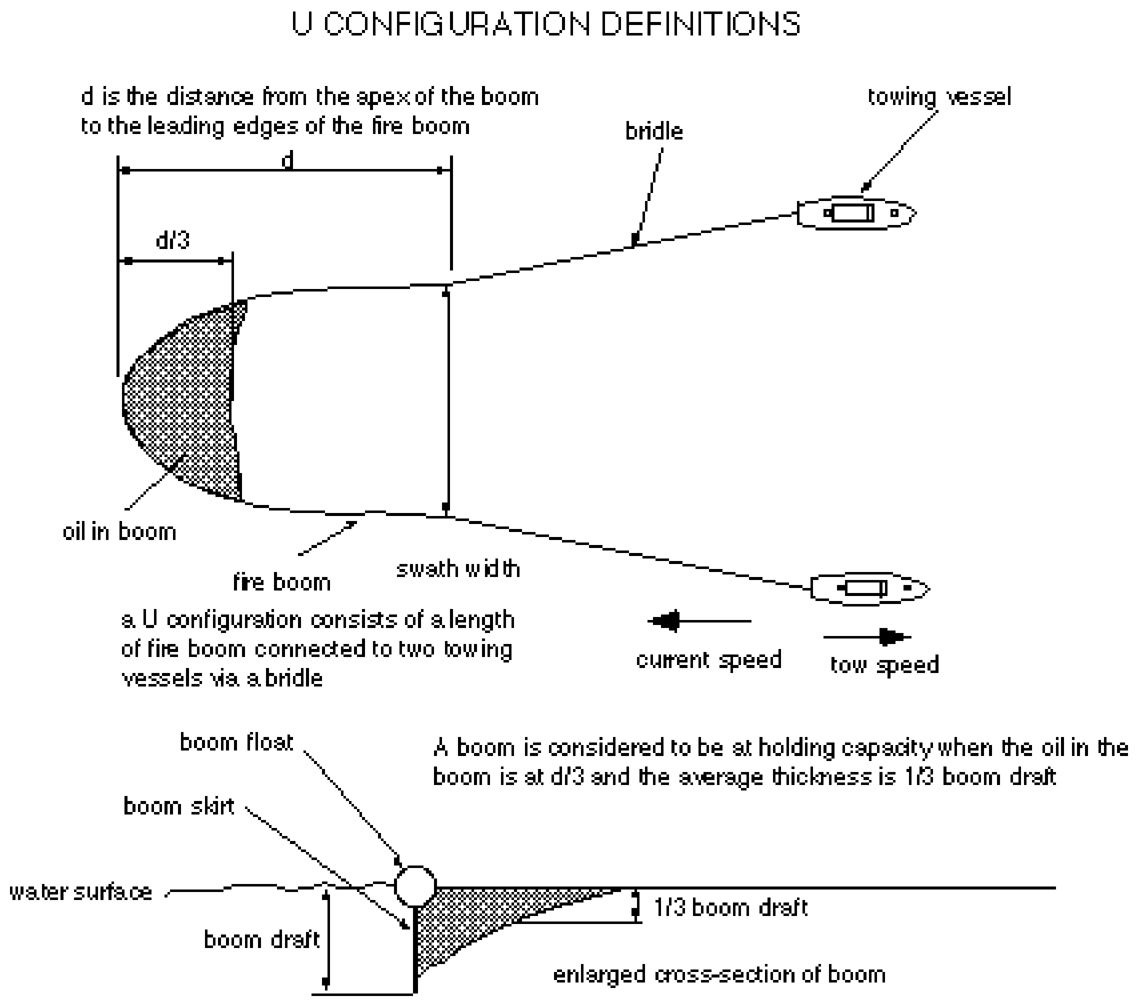

U CONFIGURATION DEFINITIONS
d is the distance from the apex of the boom to the leading edges of the fire boom
d
d/3
bridle
towing vessel
oil in boom
fire boom
swath width
a U configuration consists of a length of fire boom connected to two towing vessels via a bridle
current speed
tow speed
boom float
boom skirt
A boom is considered to be at holding capacity when the oil in the boom is at d/3 and the average thickness is 1/3 boom draft
water surface
boom draft
1/3 boom draft
enlarged cross-section of boom

Figure 9.2: Typical boom operation

$$\text{Net}'_{burn} = 0.000058 \cdot (1 - Y) \cdot A_{burn} \tag{9.4}$$

if $\delta_{burn}(t) > 0.002$; otherwise, $\text{Net}'_{burn} = 0$, with:

$$\delta_{burn}(t) = \frac{V_{burn}(t_0) - t \cdot \text{Net}'_{burn}/(1 - Y)}{A_{emul}} \tag{9.5}$$

where $\delta_{burn}(t)$ is the thickness of the oil at time t, and $0 \leq t \leq \Delta t$.

Cailtin, is this suposed to be Area of emulsion in the numerator?

Or, more simply, for a complete burn:

$$\text{Net}'_{burn} = 0.000058 \cdot (1 - Y) \cdot A_{burn} \tag{9.6}$$

for $0 \leq t \leq \Delta t$, with:

$$\Delta t = \frac{\delta_{burn}(t_0) - 0.002}{0.000058 \cdot (1 - Y)} \tag{9.7}$$

Net'_{burn} and Δt are the quantities that need to be returned to the computational core.

9.3 Dispersant Application

9.3.1 Description

The goal of applying dispersants to an oil slick is to reduce the surface tension between the oil and water. This results in a smaller droplet size distribution when the oil slick is exposed to the turbulence from surface action such as breaking waves. However, modeling this entire process would require a detailed knowledge of a great many processes that are not likely to be well understood:

- The exact distribution of the oil on the water surface
- The spray pattern of the dispersant, and how it relates to the surface oil distribution
- The resulting dispersant/oil ratio (DOR)
- The resulting change in oil-water surface tension
- How this surface tension changes the droplet size distribution

As there are parts in this entire process that are not well understood, and other parts for which the data are unlikely to be available, there is no model for this process in GNOME. Rather, GNOME allows the user to input a percent effectiveness; this value is an estimate of what fraction of the oil was "dispersed" by a given application.

Dispersion occurs naturally but is greatly enhanced by the addition of dispersing chemicals. The argument in favor of dispersion is that breaking the oil into small droplets dispersed into the water column will reduce the oil concentration levels that organisms are exposed to from a surface slick, and may increase natural weathering processes such as biodegradation.

Commercial products typically contain three types of chemicals: surfactants that lower the oil-water interfacial tension, solvents that promote the dissolution of the surfactants into the oil, and additives that may increase biodegradability or long-term stability of the dispersion. The product schedule of the US National Oil and Hazardous Substances Pollution Contingency Plan lists the dispersant chemicals allowed for U.S. waters. Many area contingency plans contain guidelines on when and where dispersants may be used.

Dispersants are sprayed onto the slick from specially equipped planes, helicopters, or boats. Dispersant effectiveness is a function of ocean mixing energy, properties of the spilled oil, and the oil's state of weathering. Increased ocean turbulence will increase both natural and chemically enhanced dispersion. Research studies indicate that the presence of indigenous surfactants in an oil are important factors for its dispersibility. Experience has shown that dispersant effectiveness is reduced as the oil weathers. This is probably due to the increasing viscosity of the slick. Figure 9.3 illustrates a common rule of thumb for the relation of dispersibility to viscosity.

Refs needed? DAH

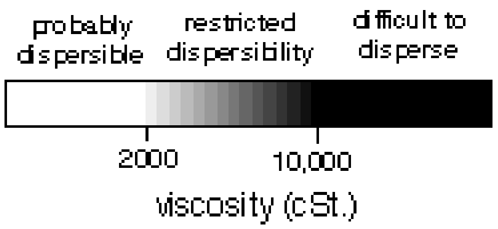


Figure 9.3: Dispersibility versus viscosity

9.3.2 Computing a Droplet Size Distribution from Dispersant Effectiveness

See Section 6.4 on Dispersion for details on how natural dispersion is computed. For the portion of the slick where chemically enhanced dispersion is computed as a separate process, it is added to the natural dispersion. In the surface dispersion calculation, a two-parameter distribution for the droplet sizes is computed. A threshold droplet size is used to determine what fraction of the oil refloats immediately to the surface, and what fraction remains under the water surface, suspended by the turbulence in the near surface (i.e., dispersed). When the user specifies a percent efficiency for the dispersant application, the droplet size distribution is adjusted so that the given percent of the mass is under that threshold.

9.3.3 Algorithm

The user enters his/her estimate of dispersant efficiency (eff_{disp}). However, GNOME displays a suggested default efficiency given by:

$$\mathrm{eff}_{disp} = 0.241 + 0.587x - 0.191x^2 + 0.02616x^3 - 0.0016x^4 - 0.000037x^5 \tag{9.8}$$

where $x = 0.3H_w$. Then:

$$\mathrm{Net}'_{disp} = \mathrm{eff}_{disp} \cdot \frac{V_{disp}}{\Delta t} \tag{9.9}$$

where:
Net'_{disp} is the dispersed volume rate (in m^3/s), and
V_{disp} is the teated oil volume (in m^3). and
Δt is the duration of the dispersant application.

9.3.4 Added Correction for Viscosity

NOTE: The viscosity correction outlined below is a proposal that is not currently implemented in GNOME.

Adjust the dispersant efficiency by a viscosity correction (ν_{cor}; Holder et al. 2015):

$$\nu_{cor}(t_{spray}) = \frac{4}{3} exp\Big(-150 \cdot \nu_{emul}(t_{spray})\Big) \tag{9.10}$$

where:
t_{spray} is the time of the dispersant application and
ν_{emul} is the mass-weighted average (over all LEs) of surface oil emulsion kinematic viscosity (in m^2/s) at that time.

This gives a new net efficiency (eff_{net}). Be sure to check that the efficiency does not exceed 1 (we are not creating new oil).

$$\mathrm{eff}_{net} = \mathrm{eff}_{disp} \cdot \nu_{cor} \tag{9.11}$$

and if $\mathrm{eff}_{net} > 1$, then eff_{net} is set to 1. Then:

$$\mathrm{Net}'_{disp} = \mathrm{eff}_{net} \cdot \frac{V_{disp}}{\Delta t} \tag{9.12}$$

This is the term required by the computational core.

9.4 Code Structure for Cleanup

(Includes suggested changes)
Important notes

May need to be reviewed together with content above

- GNOME is designed to allow oil removal or dispersant spray over a subset of all the LEs.
- Burning and skimming remove a certain amount of surface oil over a certain period of time. Both amount and time are known before the computational run begins.
- Dispersant spraying removes a certain fraction of the surface oil. This fraction is known prior to the computational run. Dispersant operations occur over a single time step.
- If natural weathering, beaching, and cleanup operations remove all surface oil from the best guess scenario at time t, the computational run stops at time t, even if the user specified a longer spill simulation time.

Scenarios where both dispersant and skimming (burn) operations are proposed

A) *Skimming and dispersant operations do not overlap in time.* There should be no conflict here.

B) *Dispersant spraying occurs during a skimming (burning operation).* Suppose skimming starts at t_1 and goes to t_3. Assume dispersant spraying happens at t_2 where $t_1 < t_2 < t_3$. At time step t_2, perform the dispersant operation before the skimming operation.

  i) Define $V_{surf}(t_2)$ as surface oil volume at beginning of time step t_2
  ii) Remove from the surface oil $V_{disp} = f_{disp} \cdot \text{eff}_{disp} \cdot V_{surf}$ where f_{disp} = fraction of surface oil that was sprayed
  iii) Recompute the amount of surface oil as $V_{surf}^{new}(t_2) = V_{surf} - V_{disp}$
  iv) Continue with the skimming operation. Stop the model run if there is no remaining surface oil.

Elsewhere in the document, Δt is used for time step. DAH

Confirm that i-iv belong beneath item B. DAH

Chapter 10

Weathering Uncertainty

In GNOME, the notion of weathering uncertainty is to provide the model user with an indication of the best-case and worst-case scenarios with regard to how the oil might change over time. A user will configure the model using estimated input parameters for a particular situation. Thus, for certain inputs, we assign a level of confidence that we will call "uncertainty."

We have dealt with uncertainty before in other projects. When implementing the oil budget calculator for Deep Water Horizon, we treated each of the major parameters in the weathering algorithms as a stochastic variable and performed Monte Carlo simulations. This involves running a particular model hundreds or even thousands of times, each time varying the inputs to the model by some random factor representing the uncertainty of each input.

This method is very effective for our purposes, and it would be ideal if we could implement this as a feature of GNOME. Unfortunately, the full and complete implementation of this method also involves a very large amount of computational work, and to integrate it into the GNOME architecture, we would need to perform a Monte Carlo suite of operations for every time step in a model run. As a compromise, we have opted for a limited implementation of Monte Carlo that can be run using a reasonable but still large amount of computation.

First, we have limited the number of inputs that can be assigned an uncertainty level. These are surface wind speed (U_{10}), initial spill volume ($V_{oil,0}$), and bullwinkle fraction (f_{bull}). Second, instead of hundreds of random tweaks to the input values, we select (min, max) values based on a distribution that is appropriate for each input. So the number of model iterations can be pared down to the number of uncertain inputs factored by three distribution values each (min, max, plus the unmodified input value):

$$\begin{aligned} NW &= \text{Number of different wind speed iterations} = 3 \\ NV &= \text{Number of different initial spill volumes} = 3 \\ NB &= \text{Number of Bullwinkle fraction options} = 3 \end{aligned}$$

$$\sum_{uncertainty} = \sum_{i=1}^{NW} \sum_{j=1}^{NV} \sum_{k=1}^{NB} \qquad (10.1)$$

Our limited implementation will result in 3^3, or 27 total model iterations. This can feasibly be run in concurrent steps, and it will provide GNOME with an acceptable best-case and worst-cast weathering uncertainty output.

For all inputs in which we define a notion of uncertainty, we will quantify our desired amount of uncertainty δ as a fractional value such that $0 \leq \delta \leq 1$. A minimum value of 0 means we are completely certain, and a value of 1 indicates that we are maximally uncertain, as much as is allowed for that input.

10.1 Wind Speed Weathering Uncertainty

For determining the (min, max) values for wind speed, we will use the distribution that is most commonly used to describe it, the one-parameter Rayleigh distribution (σ). We further assume that the forecasted wind speed (U_{10}) is an average for computing wind speed distribution values.

$$\sigma = \sqrt{\frac{2}{\pi}} \cdot \overline{U}_{10} \tag{10.2}$$

$$PDF:\ f(u_j) = \frac{u_j}{\sigma^2} \cdot e^{(-u_j^2/2\sigma^2)} \tag{10.3}$$

$$CDF:\ f(u_j) = 1 - e^{(-u_j^2/2\sigma^2)} \tag{10.4}$$

Using these distribution functions in combination with our uncertainty factor (δ), we determine our wind speed uncertainty value set. For our purposes, the cumulative distribution function (CDF) will provide a close approximation. Given a wind speed, assumed to be one of the winds that comprise the average, CDF will determine the fraction of wind speeds contained in our distribution that are lower than that speed.

We would like to know at which wind speed we reach a certain percentage of the distribution. In other words, we know the fraction of the distribution that we want, and need to determine the speed immediately above that fraction. So we implement a Quantile function (inverse CDF or percent point function). This allows us to give a fractional value and come up with a wind speed below which that fraction of our statistical speeds should fall.

Define $\overline{U}_{10}$. Elsewhere in this section, U_{10} is used. DAH

Also define PDF, $f(u_j)$, and u_j. DAH

The Rayleigh distribution Quantile function (Q) is:

$$Q(F;\sigma) = \sigma\sqrt{-ln[(1-F)^2]} \tag{10.5}$$

where F is the fraction of distribution.

Our uncertain wind values would be calculated as follows:

$$U_{avg} = U_{10} \tag{10.6}$$

$$\begin{aligned} U_{min} &= Q\left(\left(\frac{1}{2} - \delta\right); \sigma\right) \\ &= \sigma\sqrt{-ln\left[\left(\frac{1}{2} - \delta\right)^2\right]} \end{aligned} \tag{10.7}$$

$$\begin{aligned} U_{max} &= Q\left(\left(\frac{1}{2} + \delta\right); \sigma\right) \\ &= \sigma\sqrt{-ln\left[\left(\frac{1}{2} + \delta\right)^2\right]} \end{aligned} \tag{10.8}$$

where:
U_{avg} is the certain wind speed,
U_{min} is the minimum uncertain wind speed, and
U_{max} is the maximum uncertain wind speed.

Confirm that above definitions are correct. If not, change them here and in list of symbols. DAH

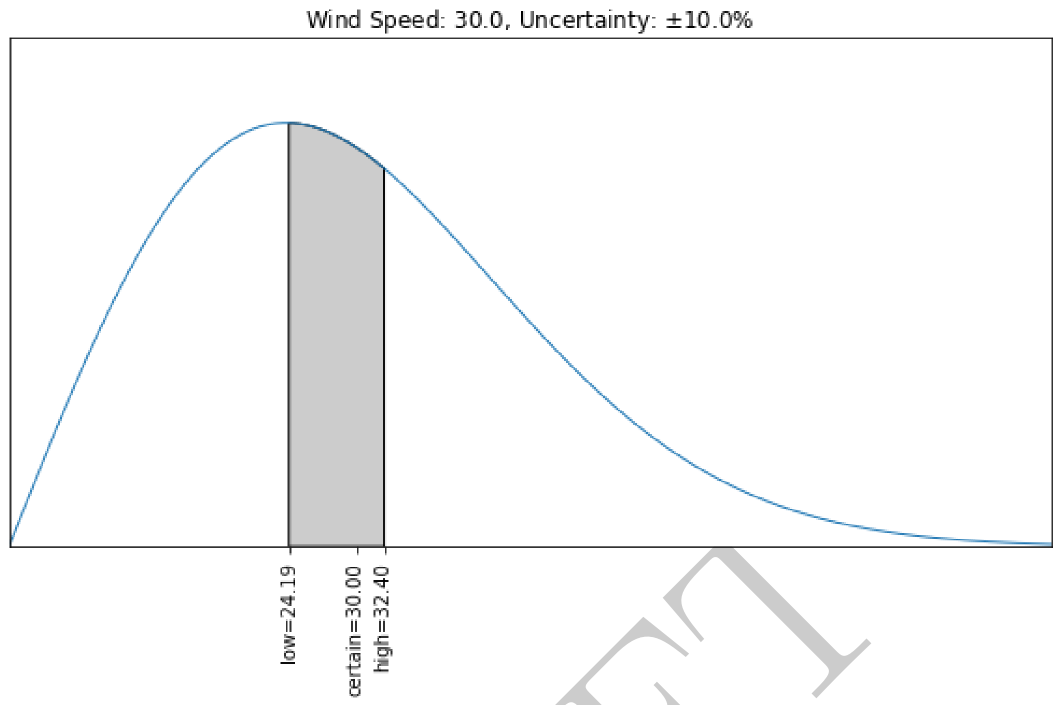


Figure 10.1: Ten percent wind uncertainty visualized along the Rayleigh distribution

10.2 Spill Amount Weathering Uncertainty

The initial spill volume ($V_{oil,0}$) is assumed to be described by a normal distribution where:

$$\overline{V} = V_{oil,0}$$
$$\sigma = \text{amount of uncertainty} \qquad \{\sigma \in \mathbb{R} : 0 \le \sigma \le 1\}$$

Define $\overline{V}$? DAH

where $\sigma = 0$ implies exact knowledge of spill amount, and increases with increasing user uncertainty of spillage size. We use the following equations to describe our minimum uncertain (V_{min}), certain (V_{avg}), and maximum uncertain (V_{max}) values:

$$V_{min} = V_{oil,0} \cdot \left(1 - \frac{2}{3}\sigma\right) \tag{10.9}$$
$$V_{avg} = V_{oil,0} \tag{10.10}$$
$$V_{max} = V_{oil,0} \cdot \left(1 + \frac{2}{3}\sigma\right) \tag{10.11}$$

10.3 Bullwinkle Weathering Uncertainty

Bullwinkle fraction (f_{bull}) is the fraction of the surface oil that must evaporate before emulsification begins. The actual Bullwinkle parameter itself is a logical variable (true = emulsify, false = does not emulsify). The initial model default value of Bullwinkle is false (no emulsification). The user can set a time after spill release to turn on emulsification (Bullwinkle = true) or, alternatively, the model will make Bullwinkle true after the oil has evaporated to the Bullwinkle fraction.

A few SINTEF-supplied oils have a measured value for f_{bull}. For most ADIOS3 oils, we have to estimate it. The user is not allowed to specify the uncertainty estimate of f_{bull}.

Soon, GNOME will define a Bullwinkle uncertainty value (δ) most likely with similar constraints as the other inputs. After discussion with emulsion experts, we will formally define the f_{min}, f_{max} values. The form that our Bullwinkle fraction uncertainty values will take will closely resemble:

Should these be $f_{bull,min}$ and $f_{bull,max}$? DAH

$$f_{bull,min} = f_{bull} \cdot (1 - \sigma_{scale}\sigma) \tag{10.12}$$
$$f_{bull,avg} = f_{bull} \tag{10.13}$$
$$f_{bull,max} = f_{bull} \cdot (1 + \sigma_{scale}\sigma) \tag{10.14}$$

We still need to finalize what we are doing for emulsification uncertainty. JLM

Define $\sigma_{scale}\sigma$? Add $f_{bull,min}$, $f_{bull,avg}$, and $f_{bull,max}$ to list of symbols? DAH

List of Symbols

To see the sections where symbols are used, refer to "Notation List.tex." DAH

Greek Characters

α_b	Uncertainty in backward direction relative to current
α_f	Uncertainty in forward direction relative to current
α_j	Soluble proportion of component j
β	??
β_l	Uncertainty in left direction relative to current
β_r	Uncertainty in right direction relative to current
γ	Eddy scale
$\Delta\rho_{ow}$	Relative difference in density between oil and water (in kg/m^3)
$\Delta V_{droplet}$	Volume of oil dissolved from droplets (in m^3)
ΔV_{surf}	Volume lost directly from surface slick (in m^3)
ΔV_{tot}	Total volume dissolved (in m^3)
ΔW	Range of windage parameters
ΔW_0	Range of reference windage parameters
Δt	Time step (in s)
Δt_0	Persistence
Δt_D	OSSM diffusion time step (in h or cm^2/s)
Δt_{emul}	Time for emulsion to form (in s)
Δt_O	OSSM model time step (in h)
Δx	Range of distances over which LEs are distributed
$\Delta x, \Delta y, \Delta z$	2-D longitude, latitude, and vertical displacement, respectively
Δx_O	Diffusion displacement step (in km)
$\Delta\vec{X}$	Vector displacement $\left(\Delta X\hat{\imath} + \Delta Y\hat{\jmath}\right)$
δ	Dirac delta function (or uncertainty factor)
δ_{50}	Mean diameter of droplets (in µm)

δ_{ave}	Average size of droplet (in m)
$\delta_{burn}(t_0)$	Initial thickness of oil (in m)
$\delta_{burn}(t)$	Thickness of oil at time t (in m)
$\delta_{droplet}$	Diameter of droplet (in µm)
δ_{min}	Minimum thickness of surface oil slick due to spreading (or smallest droplet diameter) (in m or µm)
δ_{max}	Largest droplet diameter (in µm)
δ_{slick}	Thickness of surface oil slick (in m)
δ_{th}	Average slick thickness (thick part, not sheen)
ϵ	Energy dissipation rate (in m^2/s^3 or J/m^2 or $erg/cm^3/s$??)
η_{oil}	Dynamic viscosity of oil (in $kg/(m \cdot s)$)
Θ_j	Temperature of distillation cut (in K)
λ_j	Ratio of molecular weight to density for distillation cut j
ν	Kinematic viscosity (in m^2/s)
ν_0	Initial viscosity (in m^2/s)
ν_{cor}	Viscosity correction
ν_{emul}	Kinematic viscosity of emulsion (in m^2/s)
ν_{max}	Maximum viscosity
ν_{oil}	Kinematic viscosity of oil (in m^2/s)
$\nu_{oil,0}$	Initial kinematic viscosity of oil (in m^2/s)
ν_{ref}	Reference kinematic viscosity (in m^2/s)
ν_T	Viscosity at specified temperature
ν_w	Kinematic viscosity of water (in m^2/s or centistokes (cSt))
$\nu 0_{oil}$	Oil kinematic viscosity at 288.15 K
ρ	Density of fluid (in kg/m^3)
ρ_a	Density of air (in kg/m^3)
ρ_{asph}	Density of asphaltene fractional component
ρ_{dc}	Fixed density of cut
ρ_{dis}	Density of dissolvable volume of oil (in kg/m^3)
ρ_{inert}	Density of inert volume of oil (in kg/m^3)
ρ_j	Aromatic density at distillation cut j (in kg/m^3)
ρ_{oil}	Density of oil (in kg/m^3)
ρ_{ref}	Density of oil at a reference temperature (in kg/m^3)
ρ_{res}	Density of resin fractional component

$\rho_{try,arom,i}$	Trial estimate of aromatic density of component i
$\rho_{try,j}$	Trial estimate of density of component j
$\rho_{try,sat,i}$	Trial estimate of saturate density of component i
ρ_w	Density of water (in kg/m^3 or g/cm^3)
$\rho_w \nu_w$	Dynamic viscosity of water (in N $\cdot$ s/m^2)
$\rho 0_{oil}$	Oil aggregate density at 288.15 K
σ	Rayleigh parameter (or amount of uncertainty ??)
σ^2	Variance of locations of LEs
σ_{ow}	Oil-water interfacial surface tension (in N/m)
$\sigma_{scale}\sigma$	??
τ	Shear stress on water surface
τ_{bw}	Time period between breaking waves (in s)
τ_{calm}	Calm period between breaking waves (in s)
τ_{peak}	Peak wave period (in s)
τ_{rf}	Average refloat time (in s)
τ_{week}	Time in a week (in s)
τ_{wind}	Wind stress factor (in m/s)

Alphabetic Characters

A	Area
A_{burn}	Area of boomed oil (in m^2)
$A_{droplet}$	Surface area of droplet (in m^2)
A_{emul}	Area of emulsion??
A_{slick}	Area for a circular slick (in m^2)
Adh_{oil}	Adhesion of oil (in kg/m^2)
API	API gravity
a	Approximated coefficient
a_{Un}	Random walk length for uncertainty diffusion coefficient
B	??
BP_i	Boiling point of component i (in K)
Bw	On-off switch
b	Approximated coefficient
C	Concentration of material

C_{arom}	Aromatic components
C_{asph}	Asphaltene components
C_D	Coefficient of drag
C_{res}	Resin components
C_{sat}	Saturate components
CDF	Cumulative distribution function
Cf_{dens}	Scaling factor between densities??
$\{c_3, c_2, c_1, c_0\}$	Coefficients in intermediate size range
$c_{\nu 1}$	?? (in K)
c_{dis}	Concentration of dissolvable oil
c_{oil}	Saturation concentration in oil (or concentration of oil particles in water) (in kg/m^3)
c_{sed}	Concentration of sediment particles in water (in kg/m^3)
c_{sol}	Solubility of oil (in kg/m^3)
D	Diffusion coefficient (in cm^2/s or m^2/s?? or $\frac{L^2}{T}$)
D_{ml}	Mixed-layer diffusion coefficient (in cm^2/s)
D_O	OSSM diffusion coefficient
D_x, D_y	Scalar diffusion coefficients in x, y directions (in cm^2/s)
D_z	Vertical diffusion coefficient
DOR	Dispersant/oil ratio
d	Hydrocarbon grouping parameter
dc	Integer index for distillation cuts
eff_{disp}	Chemical dispersion efficiency (from 0–1)
eff_{net}	Net efficiency (from 0–1)
eff_{skim}	Skimmer efficiency (from 0–1)
F	Fetch (or fraction of distribution??) (in m)
$\mathbf{F}$	Mass flux
Fi	Fingas number
Fi_{ref}	Reference constant
f_1, f_2	??
$f_{arom,i}$	Aromatic mass fraction
f_{asph}	Mass fraction of asphaltenes
f_{bull}	Bullwinkle fraction
$f_{bull,adios1}$	ADIOS1-calculated Bullwinkle fraction

f_{bw}	Fraction of waves that break
f_{cov}	Surface fraction coverage
f_{dc}	Mass fraction of cut dc
f_{dis}	Dissolvable fraction of oil
f_{disp}	Fraction of surface oil that was sprayed
f_{evap}	Mass fraction lost to evaporation or dissolution
$f_{evap,i}$	Portion of substance i that was evaporated
$f_{evap,max}$	Maximum amount evaporated from distillation cuts
f_i	Cumulative fraction of oil distilled (or mass fraction of hydrocarbon i??)
f_j	Mass fraction for each component j
f_{lost}	Mass fraction lost to all processes
$f_{mass,i}$	Distillation cut fractional mass??
$f_{mass,j}$	Mass fraction of component j
$f_{mass}0_j$	Fractional mass of j^{th} component
f_n	??
f_{other}	Mass fraction lost to processes other than evaporation or dissolution
f_{ref}	??
f_{res}	Mass fraction of resins
$f_{sat,i}$	Saturate component fractional mass
f_{sul}	Sulphur mass fraction
f_w	Fractional water content in emulsion
$f_{w,max}$	Maximum water fraction of emulsion
f_{wc}	Fraction of time that droplet spends in water column
$f(t_n, y_n)$	Velocity field function
$f(u_j)$	??
g	Gravitational constant
$H_{1/2}$	Significant wave height (fetch limited case) (in m)
$H_{1/3}$	Height of significant wave (in m)
H_w	Wave height (in m)
$H(X)$ (or H??)	Heaviside function
h	Length of the time step
I	Hydrocarbon characterization parameter
iBP	Initial boiling point (in K)

iBP_1	Initial boiling point of first pseudocomponent (in K)
i/n	Cumulative mass fraction of i^{th} pseudocomponent
j	Aromatic distillation cut (or component in aromatic hydrocarbon, or index representing all oil components, or oil fraction??)
j_{max}	Index of last oil component
K_1	Inertial phase constant
K_2	Viscous phase constant
$K_{arom,w}$	Watson characterization factor for aromatics
$K_{droplet}$	??
K_i	Mass transfer coefficient (in m/s)
K_{oil}	Partition coefficient
K_{ow}	(Overall??) partition coefficient
$K_{sat,w}$	Watson characterization factor for saturates
k_ν	Spreading Law coefficient
$k_{\nu 1}$	??
$k_{\nu 2}$	?? (in K)
$k_{\nu 3}$	??
$k_{\nu 4}$	??
$k_{\nu 5}$	?? (in $\sqrt{\text{s}}$/m)
k_a	Sticking parameter for the oil (in m^3/g or m^3/kg??)
k_{emul}	??
k_j	Biodegradation rate constant for pseudocomponent j (in kg/s)
k_{lw}	??
k_{p1}	Coefficient of isobaric thermal expansion
k_w	Water phase transfer velocity (in m/s)
L	Length scale of plume (averaging length for determining turbulence)
LE	Lagrangian element
M	Total mass of original point source
M_{oil}	Molar concentration (in mol/m^3)
MW	Molecular weight (in kg/kmol or kg/mol)
$\overline{MW}$	Average molecular weight for the whole oil
$MW_{arom,i}$	Molecular weight of aromatic component
MW_{asph}	Molecular weight of asphaltene component

MW_i	Molecular weight of pseudocomponent i (or average molecular weight of distillation cut??) (in kg/mol)
MW_j	Molecular weight of component j (in kg/kmol)
MW_{res}	Molecular weight of resin component
$MW_{sat,i}$	Molecular weight of saturate component
m	Number of time steps taken
m_{asph}	Asphaltene mass
$m_{droplet}$	Mass of droplet (in kg)
m_i	Mass of pseudocomponent i (in kg)
m_j	Biodegradation mass of pseudocomponent j (in kg)
m_{res}	Resin mass
mv	Molar volume (in $\mathring{A}^3$)
$\dot{m}_{oil}$	Mass of oil lost per unit water volume per unit time (in kg/m^3/s)
N	Mass transfer rate (in kg/m^2/s)
NB	Number of Bullwinkle fraction options
NV	Number of different initial spill volumes
NW	Number of different wind speed iterations
Net'_{burn}	Net burn volume rate (in m^3/s)
Net'_{disp}	Dispersed volume rate (in m^3/s)
Net'_{skim}	Net skim volume rate (net oil-removal rate) (in m^3/s)
Ni	Nickel content of oil (in PPM)
$\mathbb{N}(\delta_{droplet})$	Distribution function of droplet size
n	Number of pseudocomponents, distillation cuts, or fractions
n_i	Number of droplets in bin i
P_0	Atmospheric pressure (in Pa)
P_i	Vapor pressure at water temperature of pseudocomponent i (in Pa)
$P(x, y, t)$	Displacement probability
PC	Pseudocomponents
PCA	Aromatic components
PCS	Saturates
PDF	??
Pr_{rf}	Probability an LE will refloat
Q	Quantile function
Q_{sed}	Net mass loss rate (in kg/s)

R	Universal gas constant (in J/(K mol))
$R_{(0,1)}$	Random number (from 0–1)
$R_{(n,m)}$	Random number with uniform probability on the interval n, m
Re	Reynolds number
$r_{droplet}$	Radius of droplet (in m)
$r_{sediment}$	Radius of sediment particle (in m)
$r_{slick}(t)$	Radius of slick as a function of time (in m??)
S	Spreading coefficient
S_b	Stability
$\widetilde{S}_b$	Stability estimation function
SG	Specific gravity (in g/cm^3)
SG$_i$	Average specific gravity of distillation cut
SG$_j$	Specific gravity of oil's components
S_i	Aqueous solubility of hydrocarbon i (in kg/m^3)
$S(t)$	Spreading parameter as a function of time
Stk	Stokes number
s	Refractive index of the saturate component at 20°C
T	Temperature (in K)
T_0	Temperature of initial fraction (or lower temperature bound??)
T_1	Temperature of first distillate (or first boiling point??)
T_{b50}	Temperature at which 50% of the oil mass fraction would boil (in K)
T_{cut1}	Boiling point of first distillation cut (in K)
T_{flsh}	Flash point (in K)
T_G	Upper temperature bound
T_i	Temperature of distillate (or distillation cut boiling point??)
T_{i-1}	??
T_n	Last measured boiling point
T_{n+1}	Temperature of final fraction
T_{oil}	Temperature at which oil density is measured
T_{pour}	Pour point of the oil
T_{pp}	Pour point temperature
T_{ref}	Reference temperature (in K)
$T(t)$	Time series

$\vec{T}(t)$	Tidal velocity
T_w	Water temperature (in K)
t	Time (in h or s)
t_0	Initial time step
t_{bp}	ADIOS1 liquid boiling point (bubble point)
t_{drain}	Time to drain vessel (in min)
$t_{drain,0}$	Initial time
t_g	dT/df evaporation
t_m	Current time step
t_{m+1}	Next time step to be taken
t_n	Time at step n
t_r	Release timescale
t_{spray}	Time of dispersant application
t_{wc}	Periodicity of white capping events
tBP	Terminal boiling point (in K)
tBP_i	Terminal boiling point of i^{th} pseudocomponent (in K)
tBP_n	Terminal boiling point of last fraction (in K)
U	Velocity component
$U_{1/3}$	Significant wave height (fetch unlimited case) (in m)
U_{10}	Wind speed at 10 meters above water surface (in m/s)
$\overline{U}_{10}$	??
U_a	Velocity outside of boundary layer
U_{avg}	Certain wind speed
$\vec{U}_{\text{CATS}}$	Spatial pattern from CATS
U_{min}	Minimum uncertain wind speed
U_{max}	Maximum uncertain wind speed
U_o	Velocity of surface layer of water
U_w	Wind velocity
U_z	Wind speed at z meters above water surface (in m/s)
$U(x,y)$	Steady state circulation model
$U(x,y,t)$	Time-dependent model (circulation or wind)
$\vec{U}(x,y)$	Currents at any point
$\vec{U}(x,y)\vec{T}(t)$	Tidal representation

u	East-west
u_{*w}	Wave turbulent energy velocity (in m/s)
u_j	??
V	Vanadium content of oil (in PPM)
V	Velocity component
$\overline{V}$	??
V_{avg}	Certain spill volume
$V_{burn}(t_0)$	Total oil (or emulsion) volume available for burning
V_{dis}	Dissolvable volume of oil (in m^3)
V_{disp}	Treated oil volume (in m^3)
$V_{droplet}$	Volume of subsurface droplet (in m^3)
V_i	Volume of droplet in bin i (in m^3)
V_{inert}	Inert volume of oil (in m^3)
V_{max}	Maximum uncertain spill volume
V_{min}	Minimum uncertain spill volume
V_{oil}	Volume of oil (in m^3)
$V_{oil,0}$	Initial spill volume
V_{skim}	Oil-water volume skimmed (in m^3)
V'_{skim}	Gross skim volume rate (in m^3/s)
V_{surf}	Surface oil volume
$V_{surf}(t_2)$	Surface oil volume at beginning of time step t_2
$V_{surf}^{new}(t_2)$	Amount of surface oil (recomputed)
V_{wc}	Total volume of oil in water column
$\dot{V}_{oil}$	Oil volume spill rate
v	Velocity (in m/s)
$\vec{v}$	Velocity vector ??
$\parallel \vec{v} \parallel$	Magnitude of the original current (or wind) velocity vector (in m/s)
$\vec{v}_\perp$	Vector of the same magnitude as $\vec{v}$ but in an orthogonal direction
$v_{relative}$	Relative velocity between oil and sediment particles (in m/s)
v_{rise}	Rise velocity of oil droplets (in m/s)
W_0	Reference windage
$\overline{W}$	??
$\overline{W}_0$	Mean windage

We	Weber number
w	Vertical component of velocity (or norm)
$w(\delta_{droplet})$	Rise velocity from a droplet distribution
X	State variable
X_i	??
X_{wc}	??
x, y, z	Initial longitude, latitude, and vertical placement, respectively
$\overline{x}(t)$	Mean position
$\overline{x}^2(t)$	Variance
Y	Emulsion water fraction (from 0–1)
Y'	Maximum water fraction??
Y_{max}	Maximum water fraction
y_n	Particle position at step n
Z_{ar}	Asphaltene/resin ratio

References

Abu-Eishah, S. (1999). A new correlation for prediction of the kinematic viscosity of crude oil fractions as a function of temperature, API gravity, and 50% boiling-point temperature. *International Journal of Thermophysics 20*, 1425–1434.

American Society for Testing and Materials (2007). Manual of petroleum measurement standards. Technical Report ASTM D 1250-04, ASTM, International. Addendum 1.

Anisa, A. and A. Nour (2010). Affect of viscosity and droplet diameter on water-in-oil (w/o) emulsions: An experimental study. *International Journal of Chemical and Molecular Engineering 4*(2), 213–216.

Applied Science Associates (1995). Technical manual, spill impact mapping version 3.02. Technical report, ASA, Narragansett, RI.

Aske, N., H. Kallevik, and J. Sjöblom (2002). Water-in-crude oil emulsion stability studied by critical electric field measurements. Correlation to physico-chemical parameters and near-infrared spectroscopy. *Journal of Petroleum Science and Engineering 36*, 1–17.

Banerjee, S., S. Yalkowsky, and S. Valvani (1980). Water solubility and octanol/water partition coefficients of organics. Limitations of the solubility-partition coefficient correlation. *Environmental Science & Technology 14*, 1227–1229.

Banner, M. L. and O. M. Phillips (1974). On the incipient breaking of small scale waves. *Journal of Fluid Mechanics 65*(4), 647–656.

Bobra, M. (1992). A study of water-in-oil emulsification. Technical Report EE-132, Environment Canada, Ottawa, ON.

Bobra, M. and S. Callaghan (1990). A catalogue of crude oil and oil product properties. Technical Report CS-EE–125, Environment Canada, Ottawa, ON.

Boxall, J., C. Koh, E. Sloan, A. Sum, and D. Wu (2011). Droplet size scaling of water-in-oil emulsions under turbulent flow. *Langmuir 28*, 104–110.

Bragg, J. R. and E. H. Owens (1995). Shoreline cleansing by interactions between oil and fine mineral particles. In *International Oil Spill Conference Proceedings*, pp. 219–227.

Coastal Engineering Research Center (1984). *Shore Protection Manual* (1st ed.). Washington, D.C.: U.S. Government Printing Office.

Cohen, Y., D. Mackay, and W. Shiu (1980). Mass transfer rates between oil slicks and water. *The Canadian Journal of Chemical Engineering 58*, 569–575.

Csanady, G. T. (1973). *Turbulent Diffusion in the Environment.* Dordrecht, Holland: Springer Netherlands.

Daling, P., P. Brandvik, D. Mackay, and Ø. Johansen (1990). Characterization of crude oils for environmental purposes. *Oil & Chemical Pollution 7*, 199–224.

Darbyshire, M. and L. Draper (1963). Forecasting wind-generated sea waves. *Engineering 195*, 482–484.

Delvigne, G. A. L. and C. E. Sweeney (1988). Natural dispersion of oil. *Oil & Chemical Pollution 4*, 281–310.

Desmond, K. and E. Weeks (2014). Influence of particle size distribution on random close packing of spheres. *Physical Review E 90*, 022204.

Ding, L. and D. Farmer (1993). Observation of breaking surface wave statistics. *Journal of Physical Oceanography 24*, 1368–1387.

Dodge, F., J. Park, J. Buckingham, and R. Magott (1983). Revisions and experimental verification of the Hazard Assessment Computer System models for spreading, movement, dissolution, and dissipation of insoluble chemicals spilled into water. Technical Report 06-6285, Southwest Research Institute, San Antonio, TX.

Eley, D., M. Hey, and J. Symonds (1988). Emulsions of water in asphaltene-containing oils 1. Droplet size distribution and emulsification rates. *Colloids and Surfaces 32*, 87–101.

Elliott, A. and N. Hurford (1989). The influence of wind and wave shear on the spreading of a plume at sea. *Oil & Chemical Pollution 5*, 347–363.

Elliott, A. J. (1986). Shear diffusion and the spread of oil in the surface layers of the North Sea. *Deutsche Hydrografische Zeitschrift 39*, 117–137.

Eyring, H. (1936). Viscosity, plasticity, and diffusion as examples of absolute reaction rates. *The Journal of Chemical Physics 4*, 283–291.

Fay, J. A. (1969). The spread of oil slicks on a calm sea. In D. P. Hoult (Ed.), *Oil on the Sea*, pp. 53–63. New York, NY: Plenum Press.

Fay, J. A. (1971). Physical processes in the spread of oil on a water surface. In *International Oil Spill Conference Proceedings*, pp. 463–467.

Fingas, M. and B. Fieldhouse (2003). Studies of the formation process of water-in-oil emulsions. *Marine Pollution Bulletin 47*, 369–396.

Fingas, M. and B. Fieldhouse (2004). Formation of water-in-oil emulsions and application to oil spill modelling. *Journal of Hazardous Materials 107*, 37–50.

Fingas, M. and B. Fieldhouse (2012). Studies on water-in-oil products from crude oils and petroleum products. *Marine Pollution Bulletin 64*, 272–283.

Fisher, H. B., E. J. List, R. C. Koh, J. Imberger, and N. Brooks (1979). *Mixing in Inland and Coastal Waters*. Orlando, FL: Academic Press.

Galt, J. A. (1999). Personal Conversation.

Galt, J. A. and R. Overstreet (2014). Development of spreading algorithms for the ROC. Technical report, Genwest Systems, Inc., Edmonds, WA.

Hartung, R. and G. Klinger (1968). Sedimentation of floating oils. *Papers of the Michigan Academy of Science, Arts, and Letters 53*, 23–27.

Holder, E., R. Conmy, and A. Venosa (2015). Comparative laboratory-scale testing of dispersant effectiveness of 23 crude oils using four different testing protocols. *J. of Env. Protection 6*, 628–639.

Huibers, P. and A. Katritzky (1998). Correlation of the aqueous solubility of hydrocarbons and halogenated hydrocarbons with molecular structure. *Journal of Chemical Information and Modeling 38*, 283–292.

Issaka, S., A. Nour, and R. Yunus (2015). Review on the fundamental aspects of petroleum oil emulsions and techniques of demulsification. *Journal of Petroleum & Environmental Biotechnology 6*, 1000214.

Johansen, Ø., P. Brandvik, and U. Farooq (2013). Droplet breakup in subsea oil releases – part 2: Predictions of droplet size distributions with and without injection of chemical dispersants. *Marine Pollution Bulletin 73*, 327–335.

Jones, J. (2010). *Hydrocarbons – Physical Properties and Their Relevance to Utilisation*. Telluride, CO: Ventus Publishing.

Kolmogorov, A. (1949). On the disintegration of drops in a turbulent flow. *Proceedings of the USSR Academy of Sciences 66*, 825–828.

Lee, L., M. Hagwall, J. Delfino, and P. Rao (1992). Partitioning of polycyclic aromatic hydrocarbons from diesel fuel into water. *Environmental Science & Technology 36*, 2104–2110.

Lehr, W. (2001). Review of modeling procedures for oil spill weathering behavior. In C. A. Brebbia (Ed.), *Oil Spill Modelling and Processes*. Southampton, UK: WIT Press.

Lehr, W. (2017). Developing a new emulsification algorithm for spill response models. In *Proceedings of the Fortieth Arctic and Marine Oil Spill Program (AMOP) Technical Seminar*, Ottawa, ON. Environment Canada.

Lehr, W., C. Barker, and D. Simecek-Beatty (1999). New developments in the use of uncertainty in oil spill forecasts. In *Proceedings of the Twenty-Second Arctic and Marine Oil Spill Program (AMOP) Technical Seminar*, Ottawa, ON, pp. 271–284. Environment Canada.

Lehr, W., R. Jones, M. Evans, D. Simecek-Beatty, and R. Overstreet (2002). Revisions of the ADIOS oil spill model. *Environmental Modelling & Software 17*, 191–199.

Lehr, W. and D. Simecek-Beatty (2000). The relation of langmuir circulation processes to the standard oil spill spreading, dispersion and transport algorithms. *Spill Science and Technology Bulletin 6*(3/4), 247–253.

Li, Z., M. Spaulding, D. French McCay, D. Crowley, and J. R. Payne (2017). Development of a unified oil droplet size distribution model with application to surface breaking waves and subsea blowout releases considering dispersant effects. *Marine Pollution Bulletin 114*(1), 247–257.

Li, Z., M. L. Spaulding, and D. French-McCay (2017). An algorithm for modeling entrainment and naturally and chemically dispersed oil droplet size distribution under surface breaking wave conditions. *Marine Pollution Bulletin 119*(1), 145–152.

Loebig, C. (2015). Sinking of crude oil amended-model oil mixtures due to evaporative or dissolution weathering on the surface and submerged in water. Master's thesis, LSU. etd-07132015-093738.

Mackay, D., I. Bruist, R. Mascarenhas, and S. Paterson (1980). *Oil Spill Processes and Models*. Ottawa, ON: Environment Canada.

MacKay, D., W. Shiu, K. Hossain, W. Stiver, D. McCurdy, and S. Paterson (1982). Development and calibration of an oil spill behavior model. Technical Report Report CG-D-27-83, USCG R&D Center, Groton, CT.

Magual, A., G. Horváth-Szabó, and J. Masliyah (2005). Acoustic and electroacoustic spectroscopy of water-in-diluted-bitumen emulsions. *Langmuir 21*, 8649–8657.

McLean, J. and P. Kilpatrick (1997). Effects of asphaltene solvency on stability of water-in-crude-oil emulsions. *Journal of Colloid and Interface Science 189*, 242–253.

McLean, J., P. Spiecker, A. Sullivan, and P. Kilpatrick (1998). The role of petroleum asphaltenes in the stabilization of water-in-oil emulsions. In O. Mullins and E. Sheu (Eds.), *Structures and Dynamics of Asphaltenes*, pp. 377–422. Boston, MA: Springer.

Mohammed, R., A. Bailey, P. Luckham, and S. Taylor (1993). Dewatering of crude oil emulsions 1. Rheological behaviour of the crude oil–water interface. *Colloids and Surfaces A: Physicochemical and Engineering Aspects 80*, 223–235.

Mooney, M. (1951). The viscosity of a concentrated suspension of spherical particles. *Journal of Colloid Science 6*, 162–170.

National Oceanic and Atmospheric Administration (1999). ADIOS2 technical details (draft). Technical report, NOAA, Hazardous Material Response Division. Unpublished.

Nordam, T., R. Nepstad, E. Litzler, and J. Röhrs (2019). On the use of random walk schemes in oil spill modelling. *Marine Pollution Bulletin 146*, 631–638.

Pal, R. and E. Rhodes (1989). Viscosity/concentration relationships for emulsions. *Journal of Rheology 33*, 1021–1045.

Payne, J., B. Kirstein, J. Clayton, C. Clary, R. Redding, D. McNabb, and G. Farmer (1987). Integration of suspended particulate matter and oil transportation study, final report. Technical Report MMS 87-0083, Report to Minerals Management Service.

Payne, J. R., B. E. Kirstein, G. D. McNabb, J. L. Lambach, R. Redding, R. E. Jordan, W. Hom, C. DeOliveira, G. S. Smith, D. M. Baxter, and R. Gaegel (1984). Multivariate analysis of petroleum weathering in the marine environment - sub arctic. Technical Report 21, U.S Department of Commerce, National Oceanic and Atmospheric Administration, and U.S. Department of Interior, Mineral ManagementService.

Poirier, O. and G. Thiel (1941). Deposition of free oil by sediments settling in sea water. *AAPG Bulletin 25*, 2170–2180.

Reed, M. (1989). The physical fates component of the natural resource damage assessment model system. *Oil & Chemical Pollution 5*, 99–123.

Riazi, M. (2005). *Characterization and Properties of Petroleum Fractions*, Volume MNL50. West Conshohocken, PA: American Society for Testing and Materials (ASTM), International.

Richardson, L. F. (1922). *Weather prediction by numerical process*. Cambridge, England: The University Press.

Schramm, L. (1992). *Emulsions: Fundamentals and Applications in the Petroleum Industry*. Washington, DC: American Chemical Society. 428 pp.

Stevens, C., L. Thibodeaux, E. Overton, K. Valsaraj, K.and Nandakamar, A. Rao, and N. Walker (2015). Sea surface oil slick light component vaporization and heavy residue sinking: Binary mixture theory and experimental proof of concept. *Environmental Engineering Science 32*(8), 694–702.

Thibodeaux, L. and E. Overton (2014). ADIOS3 dissolution module for deepwater oil spills. 14 pp. Unpublished.

Thomson, J. (2014). Waves and fetch in the marginal ice zone. Distribution Statement. FY14 report.

Thorpe, S. (2007). *An Introduction to Ocean Turbulence*. New York, NY: Cambridge University Press.

Tkalich, P. and E. Chan (2002). Vertical mixing of oil droplets by breaking waves. *Marine Pollution Bulletin 44*, 1219–1229.

Twomey, S. (1977). *Atmospheric Aerosols*. New York, NY: Elsevier.

Uchrin, C. and W. Weber (1980). Modeling of transport processes for suspended solids and associated pollutants in river-harbor-lake systems. In R. Baker (Ed.), *Contaminants and Sediments*, pp. 407–425. Ann Arbor, MI: Ann Arbor Science Publishers.

Visser, A. W. (1997). Using random walk models to simulate the vertical distribution of particles in a turbulent water column. *Marine Ecology Progress Series 158*, 275–281.

Xie, H., P. Yapa, and K. Nakata (2007). Modeling emulsification after an oil spill in the sea. *Journal of Marine Systems 68*, 489–506.

Zeinstra-Helfrich, M., W. Koops, and A. J. Murk (2017). Predicting the consequence of natural and chemical dispersion for oil slick size over time. *Journal of Geophysical Research: Oceans 122*(9), 7312–7324.

Zheng, L. and P. Yapa (2000). Buoyant velocity of spherical and nonspherical bubbles/droplets. *Journal of Hydraulic Engineering 126*(11), 852–855.

What does the note mean in Loebig 2015? DAH

Appendix A

GNOME Data Formats

A.1 GNOME File Types

A.1.1 File Types

In this document, we describe a number of files that can be used or generated by GNOME. GNOME uses the following files.

Maps –	GNOME uses maps to determine the shoreline where the spilled material (typically oil) beaches. Any area of the model domain not defined as "land" is viewed as "water" by GNOME.
Currents –	Currents move the oil around and are defined on a particular grid. GNOME recognizes finite element, rectangular, and curvilinear grid circulation models in both American Standard Code for Information Interchange (ASCII) and Network Common Data Form (NetCDF) file formats. Finite element models may have either the velocities on the triangles or on the nodes. Both steady state ($U(x, y)$) and time-dependent ($U(x, y, t)$) circulation models are supported. A steady state model "current pattern" may be adjusted ("scaled") and given time dependence through a time series ($T(t)$; see Section A.2.2.3 on Scaling Current Patterns). If there is a gap between the circulation grid and the map, GNOME will assume the gap is water – and other movers, such as wind and diffusion, will be able to move the oil around.
Winds –	Winds may be entered as a constant time series ($T(t)$) or as a regular grid time-dependent model ($U_{\text{wind}}(x, y, t)$). Both NetCDF and ASCII formats are supported for wind model data.

Table A.1 lists the circulation and wind models that we have converted into GNOME format, as well as our recommended format.

GNOME also outputs files. The model particles can be output in NetCDF, kmz, or shapefile format.

Table A.1: Circulation and wind models that have been converted into GNOME format

Model	Grid	GNOME Format
CATS	Triangle, velocities on triangles	CATS
POM	Curvilinear	NetCDF
POM	Regular grid	GridCur; GridCurTime; NetCDF
ROMS	Curvilinear	NetCDF
ADCIRC	Triangle, velocities on nodes	*PTCUR*; NetCDF
FVCOM	Triangle, velocities on triangles	NetCDF
RMA2	Polygons – break down into triangles	*PTCUR*
SWAFS	Regular grid	GridCur; GridCurTime; NetCDF
ARPS	Regular grid	GridCur; GridCurTime; NetCDF
HirLAM	Regular grid	GridCur; GridCurTime; NetCDF
HF Radar	Rotated regular grid – use curvilinear	NetCDF

A.2 GNOME Input File Formats

A.2.1 Maps

Currently, GNOME uses only the boundary file atlas (BNA) map file format.

A.2.1.1 BNA Format

The BNA format consists of a list of lines and polygons that are drawn on the screen. Each feature is preceded by a description line, such as the line shown below, from the example file *simple.bna*.

Include link to example file? DAH

"2","1",18

The first number in quotes represents an identifier for the feature, and is usually unique.

The second number in quotes identifies the type of feature: "1" is a land feature; "2" is a water feature, or a polygon within another larger polygon.

The third number is the number of points in the feature, in order for drawing. A positive number indicates a polygon. Points are defined in a clockwise direction as you trace the land boundary (as though you are walking on an imaginary beach with your left foot on land and your right foot in the water). A negative number defines a line where the start and end points do not connect.

Filename: *simple.bna*

```
"2","1",18
-82.521416,27.278500
-82.552109,27.353674
-82.564636,27.383394
-82.600746,27.500633
-82.576721,27.581442
-82.541473,27.665442
-82.478104,27.725504
-82.443367,27.755222
-82.250000,27.730673
-82.250000,27.685675
-82.250000,27.640678
-82.250000,27.595680
-82.250000,27.505688
-82.250000,27.460690
-82.250000,27.415693
-82.250000,27.370695
```

```
-82.351616,27.278500
-82.453232,27.278500
"2","1",10
-82.250000,27.865969
-82.333580,27.864744
-82.383003,27.879385
-82.479012,27.888107
-82.543144,27.952902
-82.456032,28.066999
-82.405220,28.066999
-82.354408,28.066999
-82.250000,27.977007
-82.250000,27.898989
```

Two special types of polygons are defined for GNOME maps:

1. a map boundary for non-rectangular maps
2. a spillable area

These special polygons are most commonly used in Location Files to help users avoid setting spills in areas where the Location File has not been setup or well calibrated.

A.2.1.1.1 Map Bounds The map bounds define a polygon with a format similar to that shown above. This polygon should be the first polygon in the map file.

```
"Map Bounds","1",7
-121.319176,35.199476
-121.319176,34.197944
-121.218496,34.0
-119.378944,34.0
-119.221448,34.152428
-119.221448,35.199476
-121.319176,35.199476
```

A.2.1.1.2 Spillable Area The spillable area defines a polygon so that the user may not start spills outside the polygon, or over land areas within the polygon. Again, the format is similar to other polygons in the BNA format. This polygon should be the last one defined in the map file.

```
"SpillableArea", "1", 15
-121.319176,35.199476
-121.319176,34.197944
-121.218496,34.0
-120.633640,34.0
-120.445584,34.088112
-120.381776,34.085196
-120.204512,34.026884
-120.066248,34.053124
-119.931528,34.061872
-119.729456,34.015220
-119.534464,34.047292
-119.378944,34.0
-119.221448,34.152428
```

-119.221448,35.199476
-121.319176,35.199476

A.2.2 Currents

GNOME supports steady-state circulation models that produce "current patterns," as well as time-dependent circulation models. With time-dependent models, the data can be contained in a single file or broken into smaller files. GNOME uses both ASCII and NetCDF file formats, though not all grid types are supported in both formats. The types of circulation models that GNOME supports are shown below.

Finite Element –	Velocities on Triangles, Steady State [CATS], or Time Dependent (see Sections A.2.2.1.1 and A.2.2.2.3)
Finite Element –	Velocities on Nodes, Steady State, or Time Dependent [*PTCUR* and NetCDF] (see Sections A.2.2.1.2 and A.2.2.2.3)
Rectangular Grid –	Steady State [GridCur and NetCDF] (see Sections A.2.2.1.3 and A.2.2.2.1)
Rectangular Grid –	Time Dependent [GridCurTime and NetCDF] (see Sections A.2.2.1.4 and A.2.2.2.4)
Curvilinear Grid –	Time Dependent [NetCDF only] (see Section A.2.2.2.2)

Current patterns can be adjusted, or "scaled," and time dependence can be added, connecting the patterns to a time series. Time series used for scaling currents can be of the following types:

- tidal currents
- tidal height (GNOME takes the first derivative)
- wind and hydrological flow volume

Tidal current and tidal height time series can also be represented by the tidal harmonic series. In this case, GNOME calculates the necessary tidal information from the harmonic constants for the start time of the model run. For long simulations or Location Files, this results in using a smaller file size than the full time series.

A.2.2.1 ASCII Formats

A.2.2.1.1 Currents: Finite Element – Velocities on Triangles, Steady State [CATS] For more information about file formats from the NOAA Current Analysis for Trajectory Simulation (CATS) hydrodynamic model, please see the specific documentation for that application.

> **Note**: Beginning with GNOME version 1.2.2, we added the capability to generate a Directional Acyclic Graph (DAG) Tree within GNOME, so that a portion of the current file (i.e., the final section of the file, marked DAGTree) is now optional.

Add link to TinyWillapa file? DAH

Example – Filename: *TinyWillapa SAC.CUR*

```
 DAG 1.0
 Vertices 8
 8              8
-124.018048   46.694592   1.000000
-124.044816   46.668488   1.000000
-124.017968   46.650984   1.000000
-123.992400   46.664772   1.000000
-123.964264   46.646212   1.000000
-123.929744   46.673788   1.000000
-123.956592   46.696068   1.000000
-123.991760   46.683868   1.000000
Topology       6
0      1      7      5      -1     -1     0.502367     -0.298270
1      2      3     -1       5     -1     0.000000     -0.000000
3      4      5     -1       4     -1     0.000000     -0.000000
5      6      7     -1       4     -1     0.588724      0.297317
7      3      5      2       3      5     0.978753      0.205045
7      1      3      1       4      0     0.971727     -0.100222
DAGTree       13
32       1       7
31       2       5
30      -8       3
2        4      -8
0       -8      -8
7        6      -8
6       -8      -8
26       8      11
25      -8       9
12      10      -8
13      -8      -8
18      12      -8
19      -8      -8
```

Annotated Version of the File

```
DAG 1.0
Vertices       8      Number of Vertices
8              8      Number of Vertices, Repeated
Longitude      Latitude     Depth
-124.018048   46.694592   1.000000
-124.044816   46.668488   1.000000
-124.017968   46.650984   1.000000
-123.992400   46.664772   1.000000
-123.964264   46.646212   1.000000
-123.929744   46.673788   1.000000
-123.956592   46.696068   1.000000
-123.991760   46.683868   1.000000
Topology       6      Number of Triangles
```

Confirm data below are under correct headings. DAH

Points in Tri			Adjacent Tri to Seg			Velocity (u,v)		Tri#
0	1	7	5	-1	-1	0.502367	-0.298270	0
1	2	3	-1	5	-1	0.000000	-0.000000	1
3	4	5	-1	4	-1	0.000000	-0.000000	2
5	6	7	-1	4	-1	0.588724	0.297317	3
7	3	5	2	3	5	0.978753	0.205045	4
7	1	3	1	4	0	0.971727	-0.100222	5

DAGTree 13 *Number of Elements in DAGTree*

Seg#	Branch Left	Branch Right	DAGTree Branches
32	1	7	0
31	2	5	1
30	-8	3	2
2	4	-8	3
0	-8	-8	4
7	6	-8	5
6	-8	-8	6
26	8	11	7
25	-8	9	8
12	10	-8	9
13	-8	-8	10
18	12	-8	11
19	-8	-8	12

A.2.2.1.2 Currents: Finite Element – Velocities on Nodes [*PTCUR*] Most finite element circulation models calculate velocities on the triangular grid nodes. The *PTCUR* format can be used to make a single time step "current pattern" or include the full model run time series. The format can be divided into several portions: *header block*, *point definition block*, *topology*, and *time-specific data blocks*. The header block provides basic information about the file, and much of the information is optional. The point definition block includes all the points, organized with the boundary points first. The topology block defines the triangular topology and segment list, and the DAGTree defines how to search through the triangles quickly. These blocks are optional, as GNOME can calculate all this information, although loading the file will take longer without them. The time-specific data blocks make up the velocity data.

> **Note**: There are two different forms of the *PTCUR* data format. The first has velocity values at all of the points, including the boundaries. In the second case, the original circulation model does not specifically define the boundary points, and defining these points may be too time-consuming for the user. In this second case, fake boundary points may be defined that have zero velocity at these nodes. A keyword in the *VERTICES* line notifies GNOME that the first *NumLandPts* have zero velocities, and these points do not show up in the velocity data (i.e., the velocity data start with point *NumLandPts+1*).

The Header Block The header block is made up of lines that are initiated with a reserve word, which is enclosed in square brackets and is all caps, followed by a tab and the corresponding value. Each of these lines provides some global information about the file, and all but the first two are optional. The other lines have default values that GNOME provides. Except for the first line, the order of header lines is not important; however, if the keyword is in the file, a value must follow it, even if it matches the default value. Table A.2 lists the supported header lines.

Table A.2: Lines supported in the header block

Reserve Word	Definition	Data Type	Necessity
[FILETYPE]	"PTCUR"	text	required
[NAME]	"user name for file"	text	optional
[CURSCALE]	multiplicative_scale	float	optional
[UNCERTALONG]	along_axis	float	optional
[UNCERTCROSS]	cross_axis	float	optional
[UNCERTMIN]	min_current_resolution	float	optional
[GRIDTYPE]	vertical model used, bottom bc	text	optional
[MAXNUMDEPTHS]	max depth values	int	optional
[USERDATA]	"non GNOME user info"	text	optional

[FILETYPE] is an identifier for GNOME that identifies the following data as a *PTCUR* file. This is a mandatory first line for all *PTCUR* files.

[NAME] is used to identify the type of file for GNOME and allows the user to supply a name for the resulting current mover that will appear in the GNOME Summary List, in the left section of the window.

[CURSCALE] is used to set a global multiplicative scale for all of the current data in the file. In general, GNOME assumes that all of the current data it receives are in units of m/ sec, but the *PTCUR* mover will allow for a change of units through this global scaling term. If this term is not provided in the file, a value of 1.0 will be assumed. In GNOME's Diagnostic Mode, the associated dialog box allows the user to set or override this value.

[UNCERTALONG] and [UNCERTCROSS] are terms whereby the user can specify a pair of coefficients that GNOME will use to set the uncertainty associated with the *PTCUR* mover. The first coefficient will set the bound on the Monte Carlo uncertainty components added or subtracted to the along-axis component of the current vector, and the second coefficient will be used to Monte Carlo the cross-axis uncertainty of the current vectors. If this term is not provided in the file, values of 0.5 and 0.25 will be assumed. In GNOME's Diagnostic Mode, the associated dialog box allows the user to override these values.

[UNCERTMIN] is currently not implemented, and a value of 0.0 is assumed. When implemented, the Uncertainty Minimum is intended to be used to set a minimum speed resolution that is expected from the model, and is used to Monte Carlo an uncertainty for cases where the predicted currents are very small. If this term is not provided in the file, a value of 0.05 will be assumed. In GNOME's Diagnostic Mode, the associated dialog box allows the user to override these values.

[GRIDTYPE] is an identifier of how the vertical depth data were modeled. If there are no depth data, the keyword "2D" is used. If there is information about the depth of the area being modeled, and the currents are the same at every depth, the keyword "BAROTROPIC" is used (Figure A.1). If the depth is modeled using a sigma coordinate model, the keyword "SIGMA" is used (Figure A.2). If the depth is modeled using a layered system, the keyword "MULTILAYER" is used (Figure A.3). These last two options also require a boundary keyword, either "NOSLIP" or "FREESLIP," where "NOSLIP" also requires a distance in meters to define the boundary layer. This distance is constant throughout the domain. The default is "2D," in which case any depth information will be ignored.

[MAXNUMDEPTHS] gives the maximum number of depth points where horizontal currents are available. In some cases, points within the model may have fewer defined depth points than this number. The sigma model, however, must have data for *MAXNUMDEPTHS* in the water column at every horizontal point. The layered model has data at a maximum of *MAXNUMDEPTHS* in the water column for any horizontal point. The default is 1, which corresponds to surface data only and is the case for both the 2-D and barotropic grid types. Although the latter has depth, it has only one unique current value per horizontal point.

The Point Definition Block The "POINT DEFINITION BLOCK" describes the area covered by the model, including all of the horizontal points where data are available and at which depths the information is specified. This part of the model description also completely defines the topological characteristics of the model domain by specifying the boundary segments that divide the region into "inside" and "outside" portions.

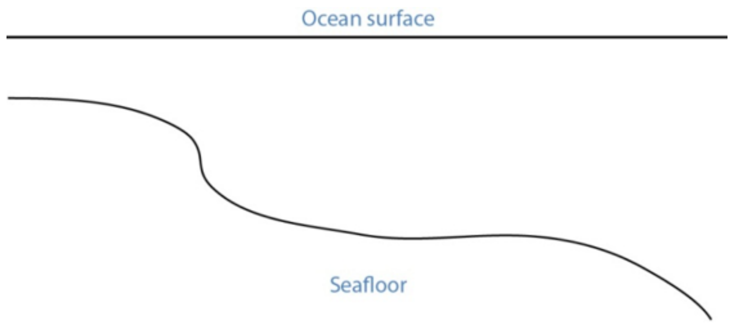


Figure A.1: Barotropic model – single velocity top to bottom

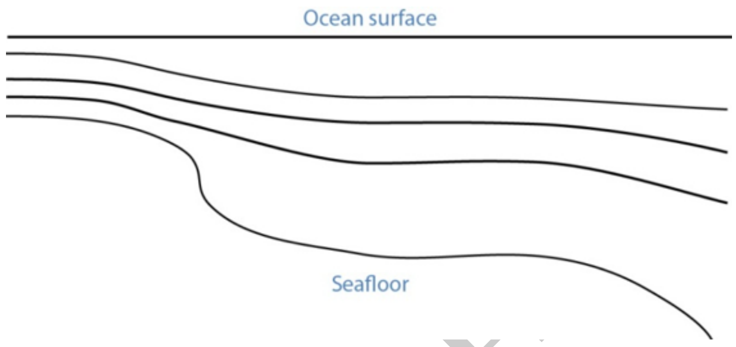


Figure A.2: Sigma model – uniform number of layers, thickness scales to total depth

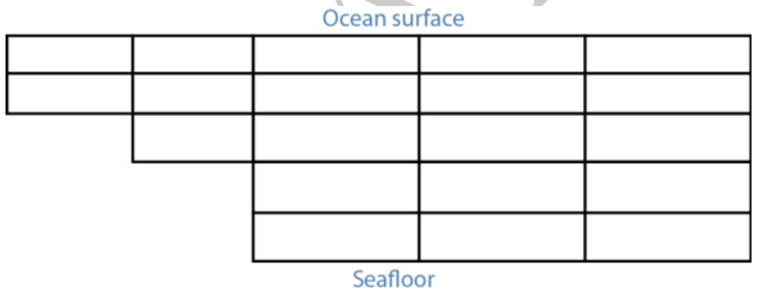


Figure A.3: Gridded model – number of layers and layer thickness may vary from place to place

The first line in the POINT DEFINITION BLOCK is made up of the keyword "Vertices," followed by the total number of horizontal points and the number of land points, separated by white space.

[USERDATA] is a reserved word that can be used (repeatedly, if necessary) by the developer of the *PTCUR* data to record any type of text documentation or metadata that he or she wants to keep associated with the file. This is optional and can be thought of as a comment area in the file format.

Vertices NPTs NumLandPts

The fields are defined as follows:

NPTs –	Gives the total number of horizontal data points (boundary and inside vertices)
NumLandPts –	If data are available at all the horizontal points, this is zero. If there is a separate outer boundary from a land map where current data are not available (assumed to be zero there), the number of these boundary points is given.

The next *NPTs* lines of the POINT DEFINITION BLOCK define the individual horizontal points where data are available. Each line describes a single data point with the following fields, separated by white space:

Pt# Longitude Latitude depth d1 d2 . . . dNdepths

Confirm accuracy of definitions – those of longitude and latitude were flipped. DAH

Each of the fields is defined as follows:

Pt# –	A sequence number, or line number, assigned to each point 1. . . *NPTs*
Longitude –	The longitude of the point, given in decimal degrees ranging from -180 to 180. The Western hemisphere is, by convention, negative longitude.
Latitude –	The latitude of the point, given in decimal degrees ranging from -90 to 90. The Northern hemisphere is, by convention, positive latitude.
depth –	The depth in meters of the modeled ocean at the point. If the grid type is 2-D, this field and the rest of the line will not be read.
d1 d2 . . . dNdepths –	Each of the *d1* through *dNdepths* values will be interpreted as a depth within the water column where current information will be defined. If the grid type is barotropic, these points will not be read and the currents that are given will be assumed to represent the entire water column. For any point where data are available at fewer than the maximum number of depths, the user should enter, in order, all the valid depths and end the line with -1 to mark the end of the data.

The lines that represent data points have two restrictions on the order in which they are entered into the file:

1. All boundary segments must be at the beginning of the file.
2. All boundary segments must have their points entered sequentially in "counter-clockwise" (CC) order.
   a. CC order is defined as follows: If an observer were to travel along the boundary of the model in a direction such that his or her left hand always pointed to the inside of the model, then he or she would encounter the boundary points in CC order.
   b. To build a *PTCUR* file, the user would first enter all of the points in CC order around the outer boundary of the model and follow those by the points in CC order around all the islands (in this case, only one). After the boundary segments are entered in the point list, all other points (the interior ones) can be entered in any order that is convenient.

After a line is entered for each of the model's horizontal data points, the next line contains a single integer value:

Number_of_Boundary_Segments

This is the total number of boundary segments that are needed to define the inside and outside of the model. The first boundary listed is the main outer edge of the model; then each of the islands represented by the model is added. For example, a domain with no islands will have a value of "1," whereas a domain with two islands will have a value of "3."

Following the line that tells GNOME how many boundary segments there are in the model domain will be one line for each of the boundary segments, with the number of the last boundary point on the corresponding segment.

Last_point_in_segment1
Last_point_in_segment2
...

You may want to define the land/water map from the vertices of your domain. This may be preferable to using a high resolution shoreline if your model and the shoreline have significant mismatch. In order to define the map, you need to specify if any of the segments are open water.

```
WaterBoundaries   2   5
3
4
```

The numbers in the header line are the number of water boundaries and the total number of boundaries. The listed points are the indices of the end-points of the water boundary segments.

The Topology Block – Optional From the POINT DEFINITION BLOCK, GNOME will be able to completely reconstruct the topology and geometry of the model domain and develop an interpolation procedure to estimate data between the specified data points. GNOME will also be able to calculate when a pollutant particle has encountered a boundary of the model domain.

Alternatively, the CATS program can be used to determine the topology. The POINT DEFINITION BLOCK is in similar form to a vertex data (VerDat) file and can easily be transformed into one. To do this:

1. Create a separate file with a header line "DOGS",
2. then enter all the points, comma delimited, followed by a line of zeros, and
3. finally enter the boundary information.

Any depth information should be removed and a single z value included for each point (≥ 1.0). The order of the points in the *PTCUR* file must be the same as that in the VerDat file used in CATS. You can then create a fake current pattern, and export the .CUR file. Select the Topology and DAGTree blocks from a CATS .CUR file and paste them into your *PTCUR* file. (The DAGTree is optional. If GNOME doesn't find a DAGTree, it will create it from the topology.) Then GNOME will not have to perform triangularization, which saves time if the same topology will be used repeatedly with different data sets. GNOME will ignore the velocity information given at the end of each topology line from the CATS .CUR file. For more information on using CATS to transform a POINT DEFINITION BLOCK to a VerDat file, see the CATS-specific documentation.

Is this information included in this document? DAH

The Time-Specific Data Blocks The TIME SPECIFIC DATA BLOCK contains the actual current velocity data for a fixed current pattern. If the input data represent a time-stepping pattern, then the block will be repeated as many times as necessary to step through the input information.

The first line in the TIME SPECIFIC DATA BLOCK is the keyword [TIME], followed by the time at which the block of current data was taken.

[TIME]   StartDay   StartMo   StartYr   StartHr   StartMin

The last five fields on this line define the time when the data in the following data block were taken. If these fields have the default value "-1" in them, it will indicate that the model data represent a steady state and that only one TIME SPECIFIC DATA BLOCK will be present.

The next NPTs lines of data in the POINT DEFINITION BLOCK give the current data for each of the points described in the POINT DEFINITION BLOCK. The line of data contains:

Pt#   Ud1   Vd1   Ud2   Vd2   ... UdNdepths VdNdepths

The number of U-V pairs that are given on each line will need to correspond to the data given in the POINT DEFINITION BLOCK. For example, if four different depth data points are specified for a particular point, then four U-V pairs will be expected. This means that different lines of data may be of different lengths, but they will all end with a return sequence, and the actual number of fields for a particular point will be given by the line defining that point in the POINT DEFINITION BLOCK.

If the TIME SPECIFIC DATA BLOCK does not start with a constant time flag, then it may be followed by another TIME SPECIFIC DATA BLOCK that is in the same format, but will have a different time. Each succeeding time block must have a time value that is larger than the one from the previous block. The offsets can vary in size, however.

For very large data sets, where having all the currents in one file would be unwieldy (for example, small time steps or extended time runs, as in Trajectory Analysis Planner [TAP]), there is an alternative format. The [TIME] blocks can be put in separate files, with any number of blocks in each file. In place of these blocks in the header file, the full file pathnames (or partial paths, relative to the GNOME folder), and the start time and end time contained in each file should be listed. The keywords for this are [FILE], [STARTTIME], and [ENDTIME], respectively. If there is a single time in a file, the start time and end time are the same. A constant current can also be handled this way; start time and end time are both a series of negative ones ("-1").

Three annotated example files follow, each starting at the top of a page for easier reading. At this time, only 2-D time-dependent (x, y, t) data are shown in the examples.

Shall I add 0 as placeholder before decimals below or leave as is? DAH

Example 1 – Filename: ***skipptcur.cur***

```
[FILETYPE] PTCUR
[NAME] skip_ps_ptcur2D
[CURSCALE] 2.0
[UNCERTALONG] .3052
[UNCERTCROSS] .127
[UNCERTMIN] .01
[MAXNUMDEPTHS] 1
[GRIDTYPE] 2-D
[USERDATA] hi fred
[USERDATA]how are you ?
 VERTICES    5056    3150
 1           -122.548000   47.588500   1.000
 2           -122.547000   47.585500   1.000
 3           -122.548000   47.581300   1.000
 4           -122.547000   47.578700   1.000
 5           -122.543000   47.578200   1.000
 6           -122.544000   47.576000   1.000
 7           -122.546000   47.574000   1.000
 8           -122.549000   47.572400   1.000
 9           -122.550000   47.570600   1.000
 10          -122.545000   47.568500   1.000
 . . .
 . . .
 5054        -122.447000   47.582600   1.000
 5055        -122.437000   47.583300   1.000
 5056        -122.427000   47.583600   1.000
 Topology    6986
 4      5045   5046   4162   2612   2613   -2.536220   3.269671
 2      3      4      -1     2      -1     0.662334    0.724430
 4      1      2      -1     1      2612   1.187206    0.244548
 5      10     5045   8      2613   7      -0.668295   1.037525
 6      9      10     -1     7      6      -0.174680   -0.246778
 7      8      9      -1     6      -1     -0.130291   -0.090753
 . . .
 . . .
 4958   4942   4959   6981   6975   6960   -0.899112   5.741174
 4442   4441   4419   6922   6966   6985   0.818613    0.580789
 4420   4421   4442   6969   6966   6967   0.626956    0.418461
 4442   4461   4441   6980   6983   6970   0.641757    0.720018
 DAGTree    15270
 34236      1        7758
 10448      2        3922
 32803      3        1906
 23762      4        1005
 13772      5        492
 34118      6        260
 . . .
 . . .
 2448       -8       -8
 2455       15268    15269
 2454       -8       -8
 2466       -8       -8
[TIME] 14 2 00 10 00      Day Month Year Hour Minute
 0.889853    0.737729
 2.14009     1.90379
 2.84519     2.40390
 2.84138     2.89028
 2.33662     3.00912
 0.0381742   1.07280
 1.23144     2.63017
 1.02648     2.13573
 . . .
 -1.96154    9.09004
 1.30358     -7.58093
 0.697695    -6.05114
[TIME] 14 2 00 11 00      Day Month Year Hour Minute
 0.605738    0.502185
 1.38961     1.23618
 0.982804    0.830371
 -0.529060   -0.538164
 -1.72646    -2.22335
 0.403527    -0.554565
 -1.38999    -2.69251
 . . .
 2.66105     -11.2783
 -0.714619   3.31164
 2.13874     -12.4378
 -0.351095   3.04506      Velocity information ends the file
```

Example 2 – Filename: *ptCurMap.cur*

```
[FILETYPE] PTCUR
[NAME] PtCur : Negative currents
[CURSCALE] 2.0
[UNCERTALONG] .3052
[UNCERTCROSS] .127
[UNCERTMIN] .01
[MAXNUMDEPTHS] 1
[GRIDTYPE] 2-D
[USERDATA] comments here
[USERDATA]
Vertices 9      0
  1   -124.360000   48.574744   1.000000
  2   -124.959368   48.563896   1.000000
  3   -125.104952   48.182896   1.000000
  4   -124.534720   48.210148   1.000000
  5   -124.360000   48.288996   1.000000
  6   -124.702840   48.452732   97.000000
  7   -124.863320   48.383372   60.000000
  8   -124.739872   48.299656   102.000000
  9   -124.545448   48.400108   75.000000
BoundarySegments      1
5
WaterBoundaries   2   5    (optional section to generate land water map)
3
4
[TIME] 14 2 00 10 00
  0.041327   0.001107
  0.079485   -0.004495
  0.036132   0.002556
  0.053070   0.035451
  0.086580   0.005730
  0.045369   0.012076
  0.031629   -0.002985
  0.039163   0.009258
  0.023545   -0.000079
[TIME] 14 2 00 11 00
  0.041327   0.001107
  0.079485   -0.004495
  0.036132   0.002556
  0.053070   0.035451
  0.086580   0.005730
  0.045369   0.012076
  0.031629   -0.002985
  0.039163   0.009258
  0.023545   -0.000079
[TIME] 14 2 00 12 00
```

```
  0.041327    0.001107
  0.079485    -0.004495
  0.036132    0.002556
  0.053070    0.035451
  0.086580    0.005730
  0.045369    0.012076
  0.031629    -0.002985
  0.039163    0.009258
  0.023545    -0.000079
[TIME] 14 2 00 13 00
  0.041327    0.001107
  0.079485    -0.004495
  0.036132    0.002556
  0.053070    0.035451
  0.086580    0.005730
  0.045369    0.012076
  0.031629    -0.002985
  0.039163    0.009258
  0.023545    -0.000079
[TIME] 14 2 00 14 00
  0.041327    0.001107
  0.079485    -0.004495
  0.036132    0.002556
  0.053070    0.035451
  0.086580    0.005730
  0.045369    0.012076
  0.031629    -0.002985
  0.039163    0.009258
  0.023545    -0.000079
[TIME] 14 2 00 15 00
  0.041327    0.001107
  0.079485    -0.004495
  0.036132    0.002556
  0.053070    0.035451
  0.086580    0.005730
  0.045369    0.012076
  0.031629    -0.002985
  0.039163    0.009258
  0.023545    -0.000079
[TIME] 14 2 00 16 00
  0.041327    0.001107
  0.079485    -0.004495
  0.036132    0.002556
  0.053070    0.035451
  0.086580    0.005730
  0.045369    0.012076
  0.031629    -0.002985
  0.039163    0.009258
  0.023545    -0.000079
[TIME] 14 2 00 17 00
```

0.041327	0.001107
0.079485	-0.004495
0.036132	0.002556
0.053070	0.035451
0.086580	0.005730
0.045369	0.012076
0.031629	-0.002985
0.023545	-0.000079
0.027216	0.003247

Example 3 – Filename: *ptCurNoMap.cur*

```
[FILETYPE] PTCUR
[NAME] PtCur : Negative currents
[CURSCALE] 2.0
[UNCERTALONG] .3052
[UNCERTCROSS] .127
[UNCERTMIN] .01
[MAXNUMDEPTHS] 1
[GRIDTYPE] 2-D
[USERDATA] comments here
Vertices 9      0
  1   -124.360000   48.574744   1.000000
  2   -124.959368   48.563896   1.000000
  3   -125.104952   48.182896   1.000000
  4   -124.534720   48.210148   1.000000
  5   -124.360000   48.288996   1.000000
  6   -124.702840   48.452732   97.000000
  7   -124.863320   48.383372   60.000000
  8   -124.739872   48.299656   102.000000
  9   -124.545448   48.400108   75.000000
  BoundarySegments    1
  5                       (Note that the Water Boundaries section is missing)
[TIME] 14 2 00 10 00
  0.041327   0.001107
  0.079485   -0.004495
  0.036132   0.002556
  0.053070   0.035451
  0.086580   0.005730
  0.045369   0.012076
  0.031629   -0.002985
  0.039163   0.009258
  0.023545   -0.000079
[TIME] 14 2 00 11 00
  0.041327   0.001107
  0.079485   -0.004495
  0.036132   0.002556
  0.053070   0.035451
  0.086580   0.005730
  0.045369   0.012076
  0.031629   -0.002985
  0.039163   0.009258
  0.023545   -0.000079
[TIME] 14 2 00 12 00
  0.041327   0.001107
  0.079485   -0.004495
  0.036132   0.002556
  0.053070   0.035451
  0.086580   0.005730
  0.045369   0.012076
  0.031629   -0.002985
  0.039163   0.009258
  0.023545   -0.000079
[TIME] 14 2 00 13 00
```

```
  0.041327    0.001107
  0.079485    -0.004495
  0.036132    0.002556
  0.053070    0.035451
  0.086580    0.005730
  0.045369    0.012076
  0.031629    -0.002985
  0.039163    0.009258
  0.023545    -0.000079
[TIME] 14 2 00 14 00
  0.041327    0.001107
  0.079485    -0.004495
  0.036132    0.002556
  0.053070    0.035451
  0.086580    0.005730
  0.045369    0.012076
  0.031629    -0.002985
  0.039163    0.009258
  0.023545    -0.000079
[TIME] 14 2 00 15 00
  0.041327    0.001107
  0.079485    -0.004495
  0.036132    0.002556
  0.053070    0.035451
  0.086580    0.005730
  0.045369    0.012076
  0.031629    -0.002985
  0.039163    0.009258
  0.023545    -0.000079
[TIME] 14 2 00 16 00
  0.041327    0.001107
  0.079485    -0.004495
  0.036132    0.002556
  0.053070    0.035451
  0.086580    0.005730
  0.045369    0.012076
  0.031629    -0.002985
  0.039163    0.009258
  0.023545    -0.000079
[TIME] 14 2 00 17 00
  0.041327    0.001107
  0.079485    -0.004495
  0.036132    0.002556
  0.053070    0.035451
  0.086580    0.005730
  0.045369    0.012076
  0.031629    -0.002985
  0.023545    -0.000079
  0.027216    0.003247
```

A.2.2.1.3 Currents: Rectangular Grid – Steady State [GridCur] The GridCur file should contain velocity information in the x and y directions on a rectangular grid. The first eight lines contain header information that defines the file type, grid size, and grid location. The remaining lines contain the current data. The keywords are the words shown in capital letters below. They must appear exactly as shown. This documentation consists of two example files followed by an explanation of each of the file components. You can set the range of the data by providing:

a) the upper left corner position and the increment size of Δ_x and Δ_y, or

b) the bounding latitude and longitude box.

Should x and y be subscripted after Δ? DAH

If you would like to try either of these current patterns, you will also need the *GridCur.bna* file.

Note: If you have missing values, you may simply skip those grid points in the data file.

Example 1 – Filename: *GridCurExA.cur*
In this first example, *GridCurExA.cur*, position information is given from a starting latitude and longitude and an increment.

Include link to example file? DAH

```
[GRIDCUR]
NUMROWS 100
NUMCOLS 100
STARTLAT 33.8
STARTLONG -120.4
DLAT .008
DLONG .01
row col u v
1 1 .10 .10
1 2 .10 .10
1 3 .10 .10
1 4 .10 .10
1 5 .10 .10
1 6 .10 .10
. . .
```

Example 2 – Filename: *GridCurExB.cur*
In this second example, *GridCurExB.cur*, the grid location is given by bounding latitudes and longitudes.

Include link to example file? DAH

```
[GRIDCUR]
NUMROWS 100
NUMCOLS 100
LOLAT 33.4
HILAT 35
LOLONG -120.4
HILONG -119
row col u v
1 1 .10 .10
1 2 .10 .10
1 3 .10 .10
1 4 .10 .10
1 5 .10 .10
. . .
```

Explanation of File Components The first line of the file is a flag identifying the file as an outside current file:

NUMROWS nrows
NUMCOLS ncols

Lines 4 through 7 give the grid bounds and can be specified in either of two ways:

(1) by the latitude and longitude of the grid origin (assumed to be the upper-left corner) and the increment size:

STARTLAT lat
STARTLON long
DLAT dlat
DLONG dlong

(2) by low and high latitude and longitude ranges:

LOLAT lolat
HILAT hilat
LOLONG lolong
HILON hilong

In the former case, the velocities are assumed to be given at the grid points, and in the latter case, the velocities are assumed to be in the center of the grid rectangles.

Line 8 is designed to be a header, identifying the columns of data. It is read, but not used.

row col u v

This header information is followed by NROWS × NCOLS lines of current data. Each line consists of four elements, corresponding to the items in Line 8. These are the point's location in the grid, given by a row and column, and its velocity components in the x and y directions, assumed to be in m/s. The file must contain a line for each of the NROWS × NCOLS grid points.

A.2.2.1.4 Currents: Rectangular Grid – Time Dependent [GridCurTime] If you have a rectangular grid time-dependent model, you can use this data format to create the time series of currents for GNOME. Large models and/or long time series can produce large files of output fields. You have the option to store all your data in one file, or in a series of files. We have been successful in obtaining daily forecasts in separate files, archiving them, and then "stringing" them together to create a time series for a single incident.

Data in a Single File GNOME accepts rectangular grid models in a simple file format, similar to the single current pattern described above. The header now indicates with [GRIDCURTIME] that time has been added, and the time of the first time step has been given in the [TIME] line.

> **Note**: As in the rectangular GridCur data format, if you have missing values, you may simply skip those grid points in the data file. You may also create a constant current pattern by setting all the time references to -1.

Example – Filename: *gridcurTime.cur*
[GRIDCURTIME]
NUMROWS 100
NUMCOLS 100
LOLAT 34.0
HILAT 34.4

LOLONG -120.8
HILONG -119.2
[TIME] 14 2 00 10 00 *day month year hour minute*
1 1 .10 .10
1 2 .10 .10
1 3 .10 .10
1 4 .10 .10
. . .
Each succeeding time step is simply appended to the bottom:
. . .
100 97 .10 .10
100 98 .10 .10
100 99 .10 .10
100 100 .10 .10
[TIME] 14 2 00 11 00 *next timestep information*
1 1 .20 .20
1 2 .20 .20
1 3 .20 .20
1 4 .20 .20
1 5 .20 .20
1 6 .20 .20

Data in Multiple Files

With larger time series of current data, it may be useful to break the current time series into separate files that make up a long time series all together. In that case, GNOME supports multi-file data with a header file that indicates data and hard drive location information, and the subsequent files. The format is similar to that of the header on the regular *GridCurTime* format; however, rather than including the time information and the data, this header includes the filename and location, and the start and end times for each of the files. GNOME uses linear interpolation between time steps within and across files. The references to the locations of the different current files on the computer's hard drive can be given in two ways: a full path description of the directory or a relative description of the directory.

Merge this paragraph with similar "Data in Multiple NetCDF Files" below? DAH

> **Note**: A constant current can be created using a single record with all the time indicators set to -1. A single time step is acceptable in a file with the start and end times listed as the same time in the header file.

The following four files are provided as examples with full path descriptions:

gridcurtime_hdr.cur
gridcurtime_hdrA.cur
gridcurtime_hdrB.cur
gridcurtime_hdrC.cur

Example 1 – Filename: *gridcurtime_hdr.cur*

The first file, *gridcurtime_hdr.cur*, contains the header information, and the three subsequent files comprise the data.

Include link to example file? DAH

[GRIDCURTIME]
NUMROWS 78
NUMCOLS 92
LOLAT 34.
HILAT 34.4
LOLONG -120.8

```
HILONG -119.2
[FILE] C:\GridCurTime\gridcurtime_hdrA.cur
[STARTTIME] 30 1 2002 1 0
[ENDTIME] 30 1 2002 2 0
[FILE] C:\GridCurTime\gridcurtime_hdrB.cur
[STARTTIME] 30 1 2002 3 0
[ENDTIME] 30 1 2002 5 0
[FILE] C:\GridCurTime\gridcurtime_hdrC.cur
[STARTTIME] 30 1 2002 6 0
[ENDTIME] 30 1 2002 8 0
```

Example 2 – Filenames: *gridcurtime_hdrA.cur, gridcurtime_hdrB.cur* and *gridcurtime_hdrC.cur*

In the next example, the paths start with a colon (:) to indicate that they are relative paths.

Include links to example files? DAH

```
[GRIDCURTIME]
NUMROWS 78
NUMCOLS 92
LOLAT 34
HILAT 34.4
LOLONG -120.8
HILONG -119.2
[FILE] :gridcurtime_hdrA.cur
[STARTTIME] 30 1 2002 1 0
[ENDTIME] 30 1 2002 2 0
[FILE] :gridcurtime_hdrB.cur
[STARTTIME] 30 1 2002 3 0
[ENDTIME] 30 1 2002 5 0
[FILE] :gridcurtime_hdrC.cur
[STARTTIME] 30 1 2002 6 0
[ENDTIME] 30 1 2002 8 0
```

Each subsequent file contains only the data, with no header information:

```
[TIME] 30 1 2002 1 0
  1   1     0.00000   0.00000
  1   2     0.00000   0.00000
  1   3     0.00000   0.00000
  1   4     0.00000   0.00000
  1   5     0.00000   0.00000
  1   6     0.00000   0.00000
  1   7     0.00000   0.00000
  1   8     0.00000   0.00000
  1   9     0.00000   0.00000
  1   10    0.00000   0.00000
  1   11    0.00000   0.00000
```

A.2.2.2 NetCDF Formats

Currently, GNOME can read in NetCDF files for rectangular, curvilinear, and triangular grids. This section includes examples of the three formats currently in use and some descriptions of the required information. Note that the NetCDF formats described here are presently undergoing revision to conform to the new Climate & Forecast unstructured grid data model, to be adopted in future releases of GNOME.

A.2.2.2.1 NetCDF Rectangular Grid Below is an example of the regular grid format for NetCDF files. The global attribute *grid_type = REGULAR* is the default. Time units can be hours, minutes, seconds, or days. A separate map will be needed in order to set a spill.

```
NetCDF MacintoshHD:Desktop Folder:test {
dimensions:
        lat = 16 ;
        lon = 20 ;
        time = UNLIMITED ;          (85 currently)
variables:
        double lat(lat) ;
                lat:long_name = "Latitude" ;
                lat:units = "degrees_north" ;
                lat:point_spacing = "even" ;
        double lon(lon) ;
                lon:long_name = "Longitude" ;
                lon:units = "degrees_east" ;
                lon:point_spacing = "even" ;
        double time(time) ;
                time:long_name = "Valid Time" ;
                time:units = "minutes since 1999-11-25 00:00:00" ;
        float water_u(time, lat, lon) ;
                water_u:long_name = "Eastward Water Velocity" ;
                water_u:units = "m/s" ;
                water_u:_FillValue = -9.9999e+32f ;
                water_u:scale_factor = 1.f ;
                water_u:add_offset = 0.f ;
        float water_v(time, lat, lon) ;
                water_v:long_name = "Northward Water Velocity" ;
                water_v:units = "m/s" ;
                water_v:_FillValue = -9.9999e+32f ;
                water_v:scale_factor = 1.f ;
                water_v:add_offset = 0.f ;
global attributes:
                :grid_type = "REGULAR" ;

data:

    lat = 51.144606, 51.234386, 51.324167, 51.413944, 51.503722, 51.5935, 51.683275, 51.77305, 51.862825,
51.952594, 52.042364, 52.132133, 52.2219, 52.311664, 52.401425, 52.491186 ;

    lon = 2.3155722, 2.4583139, 2.6010833, 2.743875, 2.8866917, 3.0295306, 3.1723917, 3.3152694, 3.4581667,
3.6010833, 3.7440139, 3.8869583, 4.0299167, 4.1728861, 4.3158667, 4.4588583, 4.6018583, 4.7448639, 4.887875,
5.0308917 ;

    time = 7020, 7080, 7140, 7200, 7260, 7320, 7380, 7440, 7500, 7560, 7620, 7680, 7740, 7800, 7860, 7920,
7980, 8040, 8100, 8160, 8220, 8280, 8340, 8400, 8460, 8520, 8580, 8640, 8700, 8760, 8820, 8880, 8940, 9000,
9060, 9120, 9180, 9240, 9300, 9360, 9420, 9480, 9540, 9600, 9660, 9720, 9780, 9840, 9900, 9960, 10020, 10080,
10140, 10200, 10260, 10320, 10380, 10440, 10500, 10560, 10620, 10680, 10740, 10800, 10860, 10920, 10980,
11040, 11100, 11160, 11220, 11280, 11340, 11400, 11460, 11520, 11580, 11640, 11700, 11760, 11820, 11880,
11940, 12000, 12060 ;
```

A.2.2.2.2 NetCDF Curvilinear Grid Below is an example of the curvilinear format for NetCDF files. The global attribute *grid_type = CURVILINEAR* is required (the default is *grid_type = REGULAR*). In addition to *x* and *y*, there are several other dimension name options for latitude and longitude. The dimension names need to start with *X, Y* or *LAT*, *LON* to be recognized. The variable names must appear as shown. The velocities can be short, float, or double precision numbers. Time units can be hours, minutes, seconds, or days. The land-mask is required if you want to use the grid boundary as the shoreline: *0* is land, *1* is water. If no map is available, the mask is used to identify land points (land = *0*, water = *1*) and a boundary map is created. The first sigma value is used, although currently GNOME is being extended to handle 3-D currents. The topology can be saved the first time and reloaded.

```
netcdf 20040726_11z_HAZMAT {
dimensions:
        x = 73 ;
        y = 163 ;
        sigma = 3 ;      optional
        time = UNLIMITED ;      (12 currently)
variables:
        float time(time) ;
                time:long_name = "Time" ;
                time:base_date = 2004, 1, 1, 0 ;
                time:units = "days since 2004-01-01 0:00:00 00:00" ;
                time:standard_name = "time" ;
        float lon(y, x) ;
                lon:long_name = "Longitude" ;
                lon:units = "degrees_east" ;
                lon:standard_name = "longitude" ;
        float lat(y, x) ;
                lat:long_name = "Latitude" ;
                lat:units = "degrees_north" ;
                lat:standard_name = "latitude" ;
        float mask(y, x) ;
                mask:long_name = "Land Mask" ;
                mask:units = "nondimensional" ;
        float depth(y, x) ;      optional
                depth:long_name = "Bathymetry" ;
                depth:units = "meters" ;
                depth:positive = "down" ;
                depth:standard_name = "depth" ;
        float sigma(sigma) ;      optional
                sigma:long_name = "Sigma Stretched Vertical Coordinate at Nodes" ;
                sigma:units = "sigma_level" ;
                sigma:positive = "down" ;
                sigma:standard_name = "ocean_sigma_coordinate" ;
                sigma:formula_terms = "sigma: sigma eta: zeta depth: depth" ;
        float u(time, sigma, y, x) ;
                u:long_name = "Eastward Water Velocity" ;
                u:units = "m/s" ;
                u:missing_value = -99999.f ;
                u:_FillValue = -99999.f ;
                u:standard_name = "eastward_sea_water_velocity" ;
        float v(time, sigma, y, x) ;
                v:long_name = "Northward Water Velocity" ;
                v:units = "m/s" ;
                v:missing_value = -99999.f ;
                v:_FillValue = -99999.f ;
                v:standard_name = "northward_sea_water_velocity" ;
```
global attributes:
```
                :file_type = "Full_Grid" ;
                :Conventions = "COARDS" ;
                :grid_type = "curvilinear" ;
                :z_type = "sigma" ;
                :model = "POM" ;
                :title = "Forecast: wind+tide+river" ;

data:

time = 208.4688, 208.4792, 208.4896, 208.5, 208.5104, 208.5208, 208.5312, 208.5417, 208.5521, 208.5625,
208.5729, 208.5833,,;

sigma = 0, .5, 1.;
}
```

A.2.2.2.3 NetCDF Triangular Grid

Below is an example of the triangular grid format for NetCDF files with velocities on the nodes. The global attribute *grid_type = TRIANGULAR* is required (the default is *grid_type = REGULAR*). The first depth value is used. Time units can be hours, minutes, seconds, or days. A map will be created using the boundary data. The topology can be saved the first time and reloaded.

Subheading "Example – Triangular Grid Format with Velocities on the Nodes" was removed. Was that intentional? DAH

The NetCDF header description for finite element model:
```
    NetCDF MacintoshHD:Desktop Folder:testFile {
    dimensions:
        node = 7258 ;
        nele = 13044 ; not currently used
        nbnd = 1476 ;
        nbi = 4 ;
        sigma = 11 ; optional
        time = UNLIMITED ; (12 currently)
    variables:
        short bnd(nbnd, nbi) ;
                bnd:long_name = "Boundary Segment Node List" ;
                bnd:units = "index_start_1" ;
        float time(time) ;
                time:long_name = "Time" ;
                time:units = "days since 2003-01-00 0:00:00 00:00" ;
                time:base_date = 2003, 1, 0, 0 ;
        float lon(node) ;
                lon:long_name = "Longitude" ;
                lon:units = "degrees_east" ;
        float lat(node) ;
                lat:long_name = "Latitude" ;
                lat:units = "degrees_north" ;
        float sigma(sigma) ; optional
                sigma:long_name = "Stretched Vertical Coordinate" ;
                sigma:units = "sigma_level" ;
                sigma:positive = "down" ;
        float u(time, sigma, node) ;
                u:long_name = "Eastward Water Velocity" ;
                u:units = "m/s" ;
                u:missing_value = -99999.f ;
                u:_FillValue = -99999.f ;
        float v(time, sigma, node) ;
                v:long_name = "Northward Water Velocity" ;
                v:units = "m/s" ;
                v:missing_value = -99999.f ;
                v:_FillValue = -99999.f ;

    global attributes:
                :file_type = "FEM" ;
                :Conventions = "COARDS" ;
                :grid_type = "Triangular" ;
data:

time = 26.95833, 27, 27.04167, 27.08333, 27.125, 27.16667, 27.20833, 27.25, 27.29167, 27.33333, 27.375,
27.41667 ;

sigma = 1, 0.9807215, 0.9306101, 0.83061, 0.6807215, 0.5, 0.3192785, 0.1693899, 0.06938996, 0.01927857, 0 ;
}
```

Notes:

1. The boundary list is an array of dimension *bnd(nbnd, 4)*. It consists of node numbers of the line segments, with a digit to indicate which land or island the segment is a part of, and a digit to indicate whether a boundary is land or water.

node1	node2	island	land/water (0/1)	
1	2	0	0	*1 is usually the continent and outer water BC*
2	5	0	0	
5	23	0	1	
...				
3568	1	0	1	*The last segment joins up with the first.*
551	552	1	0	*next island*
552	567	1	0	
...				
677	551	1	0	
789	388	2	0	
...				*next island, etc.*

2. Only the first sigma level is used, although GNOME is currently being extended to handle 3-D currents.

Example – Triangular Grid Format with Velocities on the Triangles Below is an example of the triangular grid format for NetCDF files with velocities on the triangles. The global attribute *grid_type = TRIANGULAR* is required (the default is *grid_type = REGULAR*). The first depth value is used. Time units can be hours, minutes, seconds, or days. A map will be created using the boundary data. The topology must be included in the file.

```
netcdf FVCOM_example {
dimensions:
        node = 32649 ;
        nele = 60213 ;
        nbnd = 5099 ;
        nbi = 4 ;
        time = UNLIMITED ; // (1 currently)
        three = 3 ;
variables:
        int bnd(nbnd, nbi) ;
        float time(time) ;
                time:units = "days since 1978-11-17 00:00:00 0:00" ;
                time:long_name = "time" ;
                time:time_zone = "UTC" ;
                time:format = "modified julian day (MJD)" ;
        float lon(node) ;
        float lat(node) ;
        float u(time, nele) ;
                u:units = "meters s-1" ;
                u:long_name = "Eastward Water Velocity" ;
                u:grid = "fvcom_grid" ;
                u:type = "data" ;
        float v(time, nele) ;
                v:units = "meters s-1" ;
                v:long_name = "Northward Water Velocity" ;
                v:grid = "fvcom_grid" ;
                v:type = "data" ;
        int nbe(three, nele) ;
        int nv(three, nele) ;

// global attributes:
                :grid_type = "Triangular" ;
data:

time = 11452 ;
}
```

Notes:

1. The boundary list is an array of dimension *bnd(nbnd, 4)*, same as above.
2. The triangle vertices are contained in *nv* and the neighboring triangles in *nbe*.

A.2.2.2.4 Data in Multiple NetCDF Files: When Your NetCDF Files Start to Get Too Big

Longer simulations require more model data, and that can cause problems when putting the entire time series into one data file. GNOME allows you to break the time series into separate files using a master file to identify all the pieces of the time series in order. This also makes possible using a series of nowcasts and forecasts strung together to make a time series. This technique worked well during the 2002 T/V *Prestige* incident in Spain.

Merge info in this section with very similar paragraph above, "Data in Multiple Files"? DAH

First create a text master file with the list of file pathnames (relative to the GNOME directory) in order. Next supply the full path name if the files are not in the same directory as GNOME, or in a subdirectory. The file will also need the header line, *NetCDF Files*.

When loading the currents in GNOME, load your master file (see example below). GNOME will use this as the list of files for the time series.

Include link to Mymasterfileex.txt? DAH

Example 1 – Filename: *MyMasterFileEx.txt*

```
NetCDF Files
[FILE] :day1.nc
[FILE] :day2.nc
[FILE] :day3.nc
[FILE] :day4.nc
```

[FILE] :day5.nc
[FILE] :day6.nc

A.2.2.3 Scaling Current Patterns

Since the current patterns created in CATS indicate only the direction of the current and the relative speeds, these current patterns need to be scaled in order to be useful with the trajectory model. For example, consider a fictitious current pattern with only two triangles: A and B. The velocity in triangle A is 1.2 to the east and the velocity in triangle B is 1.8 to the north. Observations indicate that the velocity in triangle A should be 3.0 knots to the east, so we must scale the current pattern by the ratio of these velocities in triangle A ($1.2 \cdot 2.5$ knots = 3.0 knots). That is, multiplying the velocity in triangle A in the current pattern (1.2) by the scale factor (2.5 knots) yields the observed velocity (3.0 knots). The direction did not change. To find the velocity in triangle B, we multiply the velocity in triangle B in the current pattern (1.8) by the scale factor (2.5 knots) to get a velocity of 4.5 knots. The velocity in triangle B is still to the north because the direction does not change in the current pattern.

GNOME is quite helpful in scaling current patterns. At a given reference point in the current pattern, GNOME tells you what the flow is. You then input into GNOME what you would like the velocity to be at the reference point, and GNOME calculates the scaling coefficient for the pattern for you.

The direction of the flow in the current fields in GNOME can reverse by multiplying the pattern by a negative scaling coefficient. The ebb and flow of tides are simulated this way, through a time series of positive and negative scaling values. You can scale currents with either a constant value or a time series. The acceptable file formats for time series are outlined below.

A.2.2.3.1 Time Series File Formats Current patterns in GNOME can be scaled to be time dependent with two different file types:

(1) a time series of current magnitude, or

(2) a "SHIO mover" that contains data for GNOME to use in calculating tidal current magnitudes.

All data in this section were created by the NOAA SHIO application ("shio" is Japanese for "tide").

The South Bend, Washington, U.S.A. station on the Willapa River was chosen for all the examples in this section. Below is the information found in the *SouthBend.text* file to illustrate the information GNOME needs to calculate the tidal currents at this station. This particular file is not a data file for GNOME. Those data are represented in data files presented later in this discussion.

Example – Filename: *SouthBend.text*
Tidal currents at South Bend, Willapa River, WASHINGTON COAST
Station No. CP1009
Meter Depth: n/a

Latitude: 46°0′ N
Longitude: 123°47′ W

Maximum Flood Direction: 90°
Maximum Ebb Direction: 270°

Time offsets	Hour:Min
Min Before Flood	0:19am
Flood	0:20am
Min Before Ebb	0:24am
Ebb	-0:06am

Flood Speed Ratio: 0.6
Ebb Speed Ratio: 0.5

	Speed(kt)	Direction(deg.)
Min Before Flood	00.0	n/a
Flood	01.2	090
Min Before Ebb	00.0	n/a
Ebb	01.4	270

Based on Grays Harbor Ent.

Local Standard Time

Mon, Aug 24, 1998	Sunrise – 6:09am	Sunset – 7:55pm	
	0:38am	+01.2	Max Flood
	3:31am	+00.0	Min Before Ebb
	6:29am	-01.6	Max Ebb
	9:59am	+00.0	Min Before Flood
	1:08pm	+01.4	Max Flood
	4:12pm	+00.0	Min Before Ebb
	6:56pm	-01.4	Max Ebb
	10:17pm	+00.0	Min Before Flood

Time Series of Current Magnitude Time series files for currents have the format:

dd,mm,yy,hr,min,$|U|$,0.0

where *dd* is the day, *mm* is the month, *yy* is the year, *hr* is the hour, *min* is the minute, $|U|$ is the magnitude of the velocity, and 0.0 is a number to indicate that the file is in a magnitude format rather than a *U,V* format. The direction is left blank because the current pattern supplies the individual current vectors. There is an optional header:

- The first line lists the station name.
- The second line lists the station position.
- The third line provides the units.

For example, the *SouthBend.ossm* file contains one day of tidal information for South Bend, Washington, U.S.A.

Example – Filename: *SouthBend.ossm*

```
South Bend
-123.78,46
knots
24, 8, 98, 0, 37, 1.2, 0.0
24, 8, 98, 3, 30, 0.0, 0.0
24, 8, 98, 6, 28, -1.6, 0.0
24, 8, 98, 9, 58, 0.0, 0.0
24, 8, 98, 13, 7, 1.4, 0.0
24, 8, 98, 16, 11, 0.0, 0.0
24, 8, 98, 18, 55, -1.4, 0.0
24, 8, 98, 22, 16, 0.0, 0.0
```

Annotated Version of the File

Day,	Month,	Year,	Hour,	Min.,	Speed,	Direction (Dummy Value)
24,	8,	98,	0,	37,	1.2,	0.0
24,	8,	98,	3,	30,	0.0,	0.0
24,	8,	98,	6,	28,	-1.6,	0.0
24,	8,	98,	9,	58,	0.0,	0.0
24,	8,	98,	13,	7,	1.4,	0.0
24,	8,	98,	16,	11,	0.0,	0.0
24,	8,	98,	18,	55,	-1.4,	0.0
24,	8,	98,	22,	16,	0.0,	0.0

SHIO Movers: Using Tidal Constituents GNOME can use both **tidal height** and **tidal current constituent data** to scale current patterns. In the case of tidal height station data (see below), GNOME will take the time derivative of the tidal heights, and request that the user scale that derivative to calculate the tidal currents. For the tidal current station data (see below), GNOME will use the calculated currents directly. The constituent record data are rather complex, so we have provided information about the data fields and then provided an example file.

Tidal Heights Constituent Record

Line 1	[StationInfo]	
Line 2	Type=H	*station type for heights is "H"*
Line 3	*staName*	*station name*
Line 4	Latitude=*latStr*	*decimal degrees*
Line 5	Longitude=*longStr*	*decimal degrees*
Line 6	[Constituents]	
Line 7	DatumControls.datum=*datum*	*mean sea level*
Line 8	DatumControls.FDir=0	*bug. Type as seen. Will be fixed in 1.2.7*
Line 9	DatumControls.EDir=0	*bug. Type as seen. Will be fixed in 1.2.7*
Line 10	DatumControls.L2Flag=0	*bug. Type as seen. Will be fixed in 1.2.7*
Line 11	DatumControls.HFlag=0	*bug. Type as seen. Will be fixed in 1.2.7*
Line 12	DatumControls.RotFlag=0	*bug. Type as seen. Will be fixed in 1.2.7*
Lines 13-17	*constituent amplitudes in order*	*M2, S2, N2, K1, M4, O1, M6, MK3, S4, MN4, NU2, S6, MU2, 2N2, OO1, LAM2, S1, M1, J1, MM, SSA, SA, MSF, MF, RHO, Q1, T2, R2, 2Q1, P1, 2SM2, M3, L2, 2MK3, K2, M8, MS4 [feet]*
Lines 18-23	*constituent phases in order*	*see above, lines 13-17 [degrees]*
Line 24	[Offset]	
Line 25	HighTime=*highTime 1*	*high water time adjustment (minutes)*
Line 26	LowTime=*lowTime 1*	*low water time adjustment*
Line 27	HighHeight_Mult=*highHeightScalar 1*	*high water height multiplier*
Line 28	HighHeight_Add=*highHeightAdd 1*	*high water height addend*
Line 29	LowHeight_Mult=*lowHeightScalar 1*	*low water height multiplier*
Line 30	LowHeight_Add=*lowHeightAdd 1*	*low water height addend*

Notes:

Exceptions for Lines 13–17: For the cases of ANCHORAGE, CALETA PERCY, DAEHUKSAN DO, LISBON, MUKHO HANG, and SOKCHO HANG, there are more than 37 constituents.

Lines 25–30 use a second integer to indicate to GNOME whether there are valid data in the field. "0" indicates no data, so GNOME skips the calculation. "1" indicates that valid data exist (that may be zero).

Example – Filename: *HornIslandPass.shio.txt*

```
[StationInfo]
Type=H
Name=Horn Island Pass
Latitude=30.2167
Longitude=-88.483333
[Constituents]
DatumControls.datum=0.620000
DatumControls.FDir=0
DatumControls.EDir=0
DatumControls.L2Flag=0
DatumControls.HFlag=0
DatumControls.RotFlag=0
H=0.066000 0.022000 0.013000 0.468000 0.000000 0.460000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.020000 0.000000 0.000000 0.033000 0.036000 0.000000 0.120000 0.299000 0.000000 0.000000 0.018000 0.099000 0.000000 0.000000 0.012000 0.139000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
kPrime=358.500000 355.700012 0.000000 327.000000 0.000000 324.200012 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 329.799988 0.000000 0.000000 325.500000 328.399994 0.000000 32.099998 151.800003 0.000000 0.000000 323.000000 314.100006 0.000000 0.000000 321.299988 331.899994 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
[Offset]
HighTime=-0.516667 1
LowTime=-0.883333 1
HighHeight_Mult=1.300000 1
HighHeight_Add=0.000000 1
LowHeight_Mult=1.300000 1
LowHeight_Add=0.000000 1
```

Tidal Currents Constituent Record

Line 1	[StationInfo]	
Line 2	Type=C	*station type for currents is "C"*
Line 3	*staName*	*station name*
Line 4	Latitude=*latStr*	*decimal degrees*
Line 5	Longitude=*longStr*	*decimal degrees*
Line 6	[Constituents]	
Line 7	DatumControls.datum=*datum*	*datum*
Line 8	DatumControls.FDir=*floodDirection*	*flood direction*
Line 9	DatumControls.EDir=*ebbDirection*	*ebb direction*
Line 10	DatumControls.L2Flag=*L2Flag*	*L2Flag*
Line 11	DatumControls.HFlag=*hydraulicFlag*	*hydraulic flag*
Line 12	DatumControls.RotFlag=*0*	*For non-rotary tides, use "0". For rotary tides defined relative to North or East, use "1". For rotary tides defined by major and minor axes, use "2".*
Lines 13-17	*constituent amplitudes in order*	*M2, S2, N2, K1, M4, O1, M6, MK3, S4, MN4, NU2, S6, MU2, 2N2, OO1, LAM2, S1, M1, J1, MM, SSA, SA, MSF, MF, RHO, Q1, T2, R2, 2Q1, P1, 2SM2, M3, L2, 2MK3, K2, M8, MS4 [knots]*
Lines 18-23	*constituent phases in order*	*see above, lines 13-17 [degrees]*
Line 24	[Offset]	
Line 25	MinBefFloodTime= *minBefFloodTime 1*	*minimum before flood time adjustment*
Line 26	FloodTime= *floodTime 1*	*flood time adjustment*
Line 27	MinBefEbbTime= *minBefEbbTime 1*	*minimum before ebb time adjustment*
Line 28	EbbTime= *ebbTime 1*	*ebb time adjustment*
Line 29	FloodSpdRatio=*floodSpeedRatio 1*	*flood speed ratio*
Line 30	EbbSpdRatio=*ebbSpeedRatio 1*	*ebb speed ratio*
Line 31	MinBFloodSpd=*minBefFloodAvgSpeed 0*	*average speed - minimum before flood*
Line 32	MinBFloodDir=*minBefFloodAvgDir 0*	*average direction - minimum before flood*
Line 33	MaxFloodSpd=*maxFloodAvgSpeed 0*	*average speed - flood*
Line 34	MaxFloodDir=*maxFloodAvgDir 0*	*average direction - flood*
Line 35	MinBEbbSpd=*minBefEbbAvgSpeed 0*	*average speed - minimum before ebb*
Line 36	MinBEbbDir=*minBefEbbAvgDir 0*	*average direction - minimum before ebb*
Line 37	MaxEbbSpd=*maxEbbAvgSpeed 0*	*average speed – ebb*
Line 38	MaxEbbDir=*maxEbbAvgDir 0*	*average direction – ebb*

Notes:

Exceptions for Lines 13–17: For the cases of ANCHORAGE, CALETA PERCY, DAEHUKSAN DO, LISBON, MUKHO HANG, and SOKCHO HANG, there are more than 37 constituents.

Lines 25–38 use a second integer to indicate to GNOME whether there are valid data in the field. "0" indicates no data, so GNOME skips the calculation. "1" indicates that valid data exist (that may be zero).

Example – Filename: *StJohnsRiver.shio.txt*

```
[StationInfo]
Type=C
Name=ST. JOHNS RIVER ENT. (between jetties)
Latitude=30.400000
Longitude=-81.383333
[Constituents]
DatumControls.datum=-0.350000
DatumControls.FDir=275
DatumControls.EDir=100
DatumControls.L2Flag=0
DatumControls.HFlag=0
DatumControls.RotFlag=0
H=1.993000 0.333000 0.404000 0.216000 0.293000 0.174000 0.092000 0.000000 0.000000 0.000000 0.078000 0.000000 0.000000 0.054000 0.000000 0.014000 0.000000 0.012000 0.014000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.034000 0.020000 0.000000 0.000000 0.071000 0.000000 0.000000 0.054000 0.000000 0.091000 0.044000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
kPrime=227.199997 244.399994 208.800003 98.800003 131.100006 122.699997 238.699997 0.000000 0.000000 0.000000 211.199997 0.000000 0.000000 190.300003 0.000000 235.199997 0.000000 110.699997 86.800003 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 134.600006 244.600006 0.000000 0.000000 99.199997 0.000000 0.000000 245.699997 0.000000 244.000000 100.800003 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
[Offset]
MinBefFloodTime=0.000000 1
FloodTime=0.000000 1
MinBefEbbTime=0.000000 1
EbbTime=0.000000 1
FloodSpdRatio=1.000000 1
EbbSpdRatio=1.000000 1
MinBFloodSpd=0.000000 0
MinBFloodDir=0.000000 0
MaxFloodSpd=0.000000 0
MaxFloodDir=0.000000 0
MinBEbbSpd=0.000000 0
MinBEbbDir=0.000000 0
MaxEbbSpd=0.000000 0
MaxEbbDir=0.000000 0
```

Example – Filename: *Edmonds.shio.txt*

```
[StationInfo]
Type=C
Name=Edmonds, 2.7 miles WSW of
Latitude=47.800000
Longitude=-122.450000
[Constituents]
DatumControls.datum=-0.500000
DatumControls.FDir=180
DatumControls.EDir=5
DatumControls.L2Flag=0
DatumControls.HFlag=0
DatumControls.RotFlag=0
H=1.954000 0.460000 0.402000 0.847000 0.000000 0.421000 0.000000 0.000000 0.000000 0.000000 0.078000
0.000000 0.000000 0.054000 0.018000 0.013000 0.000000 0.030000 0.033000 0.000000 0.000000 0.000000
0.000000 0.000000 0.016000 0.081000 0.028000 0.000000 0.000000 0.280000 0.000000 0.000000 0.055000
0.000000 0.125000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
kPrime=66.400002 84.099998 39.400002 72.500000 0.000000 66.199997 0.000000 0.000000 0.000000 0.000000
43.000000 0.000000 0.000000 12.300000 78.800003 74.599998 0.000000 69.300003 75.800003 0.000000 0.000000
0.000000 0.000000 0.000000 63.500000 63.000000 84.400002 0.000000 0.000000 73.199997 0.000000 0.000000
93.400002 0.000000 83.400002 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
[Offset]
MinBefFloodTime=0.733333 1
FloodTime=0.100000 1
MinBefEbbTime=0.216667 1
EbbTime=0.316667 1
FloodSpdRatio=0.100000 1
EbbSpdRatio=0.200000 1
MinBFloodSpd=0.000000 1
MinBFloodDir=0.000000 0
MaxFloodSpd=0.200000 1
MaxFloodDir=170.000000 1
MinBEbbSpd=0.000000 1
MinBEbbDir=0.000000 0
MaxEbbSpd=0.500000 1
MaxEbbDir=0.000000 1
```

Hydrology Time Series Hydrology time-series files for currents have the format shown in the bulleted list below. An example of a hydrology time-series file, *Hillsbourgh.HYD*, is also provided below.

- The first line lists the station name.
- The second line contains the reference point position for scaling the current pattern with the hydrology volume transport time series.
- The third line provides the units for the volume transport:
  - cubic feet per second (CFS)
  - kilo cubic feet per second (KCFS)
    - Defined as 1,000 cubic feet of water passing a given point for an entire second
  - cubic meters per second (CMS)
  - kilo cubic meters per second (KCMS)
    - Defined as 1,000 cubic meters of water passing a given point for an entire second

The data are given in the same time-series format as that for the currents, except that the magnitude of the current is changed to the volume transport.

Example – Filename: *Hillsbourgh.HYD*

```
HILLSBOURGH STATION
28.029534,-82.688080
CMS
01,10,2002,0,0,432,0
02,10,2002,0,0,309,0
03,10,2002,0,0,310,0
04,10,2002,0,0,312,0
05,10,2002,0,0,311,0
06,10,2002,0,0,287,0
07,10,2002,0,0,234,0
08,10,2002,0,0,235,0
09,10,2002,0,0,232,0
10,10,2002,0,0,177,0
```

Include link to Hills-bourgh.HYD? DAH

Annotated Version of the File

HILLSBOURGH STATION *Station Name*
28.029534,-82.688080 *Position (latitude, longitude)*
CMS *Units*

Day,	Month,	Year,	Hour,	Min.,	Transport,	Direction (Dummy Value)
01,	10,	2002,	0,	0,	432,	0
02,	10,	2002,	0,	0,	309,	0
03,	10,	2002,	0,	0,	310,	0
04,	10,	2002,	0,	0,	312,	0
05,	10,	2002,	0,	0,	311,	0

A.2.3 Winds

GNOME uses winds, in addition to the currents and diffusion, to move the oil. For small-scale uses, a single point forecast, *[OSSM]*, is sufficient for trajectories. In this case, the time series can be created in GNOME using the variable or constant wind dialog boxes, or loaded as a file. You will find it useful to load winds as a file if you are downloading archived wind observations, or if you are creating a blended time series, with archived observations combined with a wind forecast.

For large-scale areas, you may use winds generated by atmospheric circulation model data. Be careful in mapping latitude and longitude of the grid points with the proper projection of the model.

GNOME supports both ASCII and NetCDF formats for wind. The rectangular grid wind model in a time-series format is *[GridWindTime]*. The gridded output time-series format is the same as that for currents, except the starting keyword is different. This is to prevent the user from accidentally loading wind as current, and vice versa. The Finite Element ASCII Format (for winds at the model nodes) is *[ptWind]*. GNOME also supports a NetCDF file structure for rectangular grid models. If your particular atmospheric model is not supported, please let the GNOME Wizard know (ORR.GNOME@noaa.gov); we are always interested in adding new grids to our collection.

A.2.3.1 Winds: Single Point, Time Series [OSSM]

If you input an On-Scene Spill Model (OSSM) wind file into GNOME, you can specify the units when the file is loaded, or there is an optional header:

- The first line lists the station name.
- The second line lists the station position.
- The third line provides the units.

A.2.3.1.1 Example – Filename: *OSSM Format.WND*

```
Inchon
-126.63,37.5
knots
8,4,99,01,00,10,S
8,4,99,05,00,10,S
8,4,99,09,00,10,S
8,4,99,11,00,10,S
8,4,99,15,00,10,SW
8,4,99,21,00,10,SW
9,4,99,01,00,10,SW
9,4,99,05,00,10,SW
9,4,99,09,00,10,SW
9,4,99,11,00,10,SW
9,4,99,15,00,10,SW
9,4,99,21,00,10,SW
10,4,99,01,00,10,SW
10,4,99,05,00,05,S
10,4,99,09,00,05,S
10,4,99,11,00,05,S
10,4,99,15,00,05,S
10,4,99,21,00,05,S
11,4,99,01,00,10,SW
11,4,99,05,00,10,SW
11,4,99,09,00,10,SW
11,4,99,11,00,10,W
11,4,99,15,00,10,W
11,4,99,21,00,10,W
12,4,99,01,00,25,NW
12,4,99,05,00,25,NW
12,4,99,09,00,25,NW
12,4,99,11,00,25,NW
12,4,99,15,00,25,NW
12,4,99,21,00,25,NW
```

Annotated Version of the File

Day,	Month,	Year,	Hour,	Min.,	Speed,	Direction
8,	4,	99,	01,	00,	0,	S
8,	4,	99,	05,	00,	10,	S
8,	4,	99,	09,	00,	10,	S
8,	4,	99,	11,	00,	10,	S
8,	4,	99,	15,	00,	10,	SW

. . .

Note: Direction can also be in degrees.

A.2.3.2 Winds: Rectangular Grid, Time Series [GridWindTime]

The rectangular grid surface wind model formats are very similar to the *GridCurTime* formats for rectangular grid time-dependent currents. The only difference is that the first line of the file changes from *[GRIDCURTIME]* to *[GRIDWINDTIME]*.

A.2.3.2.1 Example – Filename: *GridWindTime.wnd*

```
[GRIDWINDTIME]
NUMROWS 19
NUMCOLS 26
LOLAT 36.6
HILAT 47.8
LOLONG -15.4
HILONG -4.3
[TIME] 19 11 02 1 00
1 1 0.15 -.16
1 2 0.16 -.17
1 3 0.17 -.19
1 4 0.19 -.21
. . .
```

A.2.3.3 Winds: NetCDF Rectangular Grid, Time Series

The NetCDF rectangular grid surface wind model formats are very similar to those of the NetCDF rectangular grid current. The only difference is that *air_u* and *air_v* are used instead of *water_u* and *water_v* for the *U* and *V* velocity components.

```
netcdf pwsWind2004080904 {
dimensions:
        lon = 155 ;
        lat = 150 ;
        time = UNLIMITED ; (49 currently)
variables:
        float time(time) ;
                time:long_name = "Time in AST" ;
                time:units = "hours since 2004-08-09 00:00:00" ;
        float lon(lon) ;
                lon:long_name = "Longitude" ;
                lon:units = "degrees_East" ;
                lon:point_spacing = "even" ;
        float lat(lat) ;
                lat:long_name = "Latitude" ;
                lat:units = "degrees_North" ;
                lat:point_spacing = "even" ;
        float air_u(time, lat, lon) ;
                air_u:valid_range = -30.f, 30.f ;
                air_u:long_name = "Eastward Air Velocity" ;
                air_u:units = "m/s" ;
                air_u:_FillValue = -9.9999e+32f ;
                air_u:scale_factor = 1.f ;
                air_u:add_offset = 0.f ;
        float air_v(time, lat, lon) ;
                air_v:valid_range = -30.f, 30.f ;
                air_v:long_name = "Northward Air Velocity" ;
                air_v:units = "m/s" ;
                air_v:_FillValue = -9.9999e+32f ;
                air_v:scale_factor = 1.f ;
                air_v:add_offset = 0.f ;
```

```
    global attributes:
              :experiment = "PWS-NFS" ;
              :grid_type = "REGULAR" ;
              :base_date = 2004, 8, 9 ;
data:

    time = 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52 ;

    lon = -148.72, -148.7, -148.68, -148.66, -148.64, -148.62, -148.6, -148.58, -148.56, -148.54, -148.52, -148.5,
-148.48, -148.46, -148.44, -148.42, -148.4, -148.38, -148.36, -148.34, -148.32, -148.3, -148.28, -148.26, -148.24,
-148.22, -148.2, -148.18, -148.16, -148.14, -148.12, -148.1, -148.08, -148.06, -148.04, -148.02, -148, -147.98,
-147.96, -147.94, -147.92, -147.9, -147.88, -147.86, -147.84, -147.82, -147.8, -147.78, -147.76, -147.74, -147.72,
-147.7, -147.68, -147.66, -147.64, -147.62, -147.6, -147.58, -147.56, -147.54, -147.52, -147.5, -147.48, -147.46,
-147.44, -147.42, -147.4, -147.38, -147.36, -147.34, -147.32, -147.3, -147.28, -147.26, -147.24, -147.22, -147.2,
-147.18, -147.16, -147.14, -147.12, -147.1, -147.08, -147.06, -147.04, -147.02, -147, -146.98, -146.96, -146.94,
-146.92, -146.9, -146.88, -146.86, -146.84, -146.82, -146.8, -146.78, -146.76, -146.74, -146.72, -146.7, -146.68,
-146.66, -146.64, -146.62, -146.6, -146.58, -146.56, -146.54, -146.52, -146.5, -146.48, -146.46, -146.44, -146.42,
-146.4, -146.38, -146.36, -146.34, -146.32, -146.3, -146.28, -146.26, -146.24, -146.22, -146.2, -146.18, -146.16,
-146.14, -146.12, -146.1, -146.08, -146.06, -146.04, -146.02, -146, -145.98, -145.96, -145.94, -145.92, -145.9,
-145.88, -145.86, -145.84, -145.82, -145.8, -145.78, -145.76, -145.74, -145.72, -145.7, -145.68, -145.66, -145.64 ;

    lat = 59.79, 59.8, 59.81, 59.82, 59.83, 59.84, 59.85, 59.86, 59.87, 59.88, 59.89, 59.9, 59.91, 59.92, 59.93,
59.94, 59.95, 59.96, 59.97, 59.98, 59.99, 60, 60.01, 60.02, 60.03, 60.04, 60.05, 60.06, 60.07, 60.08, 60.09, 60.1,
60.11, 60.12, 60.13, 60.14, 60.15, 60.16, 60.17, 60.18, 60.19, 60.2, 60.21, 60.22, 60.23, 60.24, 60.25, 60.26,
60.27, 60.28, 60.29, 60.3, 60.31, 60.32, 60.33, 60.34, 60.35, 60.36, 60.37, 60.38, 60.39, 60.4, 60.41, 60.42,
60.43, 60.44, 60.45, 60.46, 60.47, 60.48, 60.49, 60.5, 60.51, 60.52, 60.53, 60.54, 60.55, 60.56, 60.57, 60.58,
60.59, 60.6, 60.61, 60.62, 60.63, 60.64, 60.65, 60.66, 60.67, 60.68, 60.69, 60.7, 60.71, 60.72, 60.73, 60.74,
60.75, 60.76, 60.77, 60.78, 60.79, 60.8, 60.81, 60.82, 60.83, 60.84, 60.85, 60.86, 60.87, 60.88, 60.89, 60.9,
60.91, 60.92, 60.93, 60.94, 60.95, 60.96, 60.97, 60.98, 60.99, 61, 61.01, 61.02, 61.03, 61.04, 61.05, 61.06, 61.07,
61.08, 61.09, 61.1, 61.11, 61.12, 61.13, 61.14, 61.15, 61.16, 61.17, 61.18, 61.19, 61.2, 61.21, 61.22, 61.23, 61.24,
61.25, 61.26, 61.27, 61.28 ;
}
```

A.2.3.4 Winds: NetCDF Curvilinear Grid

The NetCDF curvilinear grid surface wind model format is very similar to the NetCDF curvilinear grid current format. The only differences are that (1) *air_u* and *air_v* are recommended instead of *u* and *v* for the *U* and *V* velocity components, and (2) the land mask is not used. The dimension names need to start with *X*, *Y* or *LAT*, *LON* to be recognized. The variable names must appear as shown. The topology can be saved the first time and reloaded.

```
netcdf 20040726_11z_HAZMAT {
dimensions:
        x = 73 ;
        y = 163 ;
        time = UNLIMITED ; (12 currently)
variables:
        float time(time) ;
                time:long_name = "Time" ;
                time:base_date = 2004, 1, 1, 0 ;
                time:units = "days since 2004-01-01 0:00:00 00:00" ;
                time:standard_name = "time" ;
        float lon(y, x) ;
```

```
                lon:long_name = "Longitude" ;
                lon:units = "degrees_east" ;
                lon:standard_name = "longitude" ;
        float lat(y, x) ;
                lat:long_name = "Latitude" ;
                lat:units = "degrees_north" ;
                lat:standard_name = "latitude" ;
        float air_u(time, y, x) ;
                air_u:long_name = "Eastward Air Velocity" ;
                air_u:units = "m/s" ;
                air_u:missing_value = -99999.f ;
                air_u:_FillValue = -99999.f ;
                air_u:standard_name = "eastward_wind" ;
        float air_v(time, y, x) ;
                air_v:long_name = "Northward Air Velocity" ;
                air_v:units = "m/s" ;
                air_v:missing_value = -99999.f ;
                air_v:_FillValue = -99999.f ;
                air_v:standard_name = "northward_wind" ;

global attributes:
                :file_type = "Full_Grid" ;
                :Conventions = "COARDS" ;
                :grid_type = "curvilinear" ;
                :title = "Forecast: wind+tide+river" ;
data:

    time = 208.4688, 208.4792, 208.4896, 208.5, 208.5104, 208.5208, 208.5312, 208.5417, 208.5521, 208.5625,
208.5729, 208.5833,,;

    }
```

A.2.3.5 Data in Multiple Files: When Your NetCDF Files Start to Get Too Big

This section is virtually identical to A.2.2.2.4. Delete this and refer reader to that section? DAH

Longer simulations require more model data, and that can cause problems when putting the entire time series into one data file. GNOME allows you to break the time series into separate files using a master file to identify all the pieces of the time series in order. This also makes possible using a series of nowcasts and forecasts strung together to make a time series. This technique worked well during the 2002 T/V *Prestige* incident in Spain.

First create a text master file with the list of file pathnames (relative to the GNOME directory), in order. The full pathname is needed if the files are not in the same directory as GNOME, or in a subdirectory. The file will also need a header line, *NetCDF Files*.

To load the winds in GNOME, load the master file shown below. GNOME will use this as the list of files for the time series.

A.2.3.5.1 Example – Filename: *MyMasterFileEx.txt*

```
NetCDF Files
[FILE] :day1.nc
[FILE] :day2.nc
[FILE] :day3.nc
[FILE] :day4.nc
[FILE] :day5.nc
[FILE] :day6.nc
```

A.3 GNOME Output File Formats

A.3.1 NetCDF LE Output File Format

Below is an example of the NetCDF format for outputting the model LEs. GNOME produces a single file either for a particular time or for the whole model run. It is important to note that each of these attributes grows along the data axis, which is distinguished from the time axis. For example, the attribute *particle_count*, which grows along the time axis, describes the number of particles being tracked at every time step. If the model has been configured for uncertainty, an additional file will be created for Minimum Regret.

In sections below, single quotes are used, but double quotes are used everywhere else in this Appendix. Leave as is? DAH

A.3.1.1 Example – Contents of NetCDF LE File from Whole 2-D Model Run

```
Format:
     classic
Global Attributes:
     comment = 'Particle output from the NOAA GNOME model'
     creation_date = '2012-09-11 09:38:00'
     source = 'GNOME version 1.3.5'
     references = 'http://response.restoration.noaa.gov/gnome'
     feature_type = 'particle_trajectories'
     institution = 'NOAA Emergency Response Division'
     conventions = 'CF-1.6'
Dimensions:
     time = 241
     data = 241000 (UNLIMITED)
Variables:
     time
          Size: 241x1
          Dimensions: time
          Datatype: double
          Attributes:
               units = 'seconds since 2012-09-11 09:00:00'
               long_name = 'time'
               standard_name = 'time'
               calendar = 'gregorian'
     particle_count
          Size: 241x1
          Dimensions: time
          Datatype: int32
          Attributes:
               units = '1'
               long_name = 'number of particles in a given timestep'
               ragged_row_count = 'particle count at nth timestep'
     longitude
          Size: 241000x1
          Dimensions: data
          Datatype: single
          Attributes:
               long_name = 'longitude of the particle'
               units = 'degrees_east'
     latitude
          Size: 241000x1
          Dimensions: data
          Datatype: single
```

```
        Attributes:
            long_name = 'latitude of the particle'
            units = 'degrees_north'
    mass
        Size: 241000x1
        Dimensions: data
        Datatype: single
        Attributes:
            units = 'grams'
    age
        Size: 241000x1
        Dimensions: data
        Datatype: int32
        Attributes:
            description = 'from age at time of release'
            units = 'seconds'
    flag
        Size: 241000x1
        Dimensions: data
        Datatype: int8
        Attributes:
            long_name = 'particle status flag'
            valid_range = [0.00e+00 5.00e+00]
            flag_values = [1.00e+00 2.00e+00 3.00e+00 4.00e+00]
            flag_meanings = 'on_land off_maps evaporated below_surface'
    id
        Size: 241000x1
        Dimensions: data
        Datatype: int32
        Attributes:
            Description = 'particle ID'
            units = '1'
```

A.3.1.2 Example – Contents of NetCDF LE File from Whole (Pseudo)3-D Model Run

```
Format:
    classic
Global Attributes:
    comment = 'Particle output from the NOAA GNOME model'
    creation_date = '2012-09-11 10:18:00'
    source = 'GNOME version 1.3.5'
    references = 'http://response.restoration.noaa.gov/gnome'
    feature_type = 'particle_trajectories'
    institution = 'NOAA Emergency Response Division'
    conventions = 'CF-1.6'
Dimensions:
    time = 289
    data = 229803 (UNLIMITED)
Variables:
    time
        Size: 289x1
        Dimensions: time
        Datatype: double
        Attributes:
            units = ‘seconds since 2010-01-24 00:00:00’
            long_name = ‘time’
            standard_name = ‘time’
            calendar = ‘gregorian’
    particle_count
        Size: 289x1
        Dimensions: time
        Datatype: int32
        Attributes:
            units = ‘1’
            long_name = ‘number of particles in a given timestep’
            ragged_row_count = ‘particle count at nth timestep’
    longitude
        Size: 229803x1
        Dimensions: data
        Datatype: single
        Attributes:
            long_name = ‘longitude of the particle’
            units = ‘degrees_east’
    latitude
        Size: 229803x1
        Dimensions: data
        Datatype: single
        Attributes:
            long_name = ‘latitude of the particle’
            units = ‘degrees_north’
    depth
        Size: 229803x1
        Dimensions: data
        Datatype: single
        Attributes:
            long_name = ‘particle depth below sea surface’
            units = ‘meters’
            axis = ‘z positive down’
    mass
        Size: 229803x1
        Dimensions: data
        Datatype: single
        Attributes:
            units = ‘grams’
    age
        Size: 229803x1
        Dimensions: data
        Datatype: int32
        Attributes:
            description = ‘from age at time of release’
            units = ‘seconds’
    flag
        Size: 229803x1
        Dimensions: data
        Datatype: int8
        Attributes:
            long_name = ‘particle status flag’
            valid_range = [0.00e+00 5.00e+00]
            flag_values = [1.00e+00 2.00e+00 3.00e+00 4.00e+00]
            flag_meanings = ‘on_land off_maps evaporated below_surface’
    id
        Size: 229803x1
        Dimensions: data
        Datatype: int32
        Attributes:
            Description = ‘particle ID’
            units = ‘1’
```

Appendix B

Simulating Diffusion in a Lagrangian Element Model

B.1 Background Theory

This section will provide a summary of the background theory for simulating diffusion. For a complete explanation, see Csanady (1973).

The diffusion equation can be derived from Brownian motion or Fick's law:

$$\mathbf{F} = -D\nabla C \tag{B.1}$$

where:
F is the mass flux,
D is the diffusion constant (a tensor, or for an isotropic material, a scalar), and
C is the concentration of a material.

The result is the classical diffusion equation (for a constant D):

$$\frac{\partial C}{\partial t} = D\nabla^2 C \tag{B.2}$$

or, in 2-D Cartesian coordinates:

$$\frac{\partial C}{\partial t} = D_x\frac{\partial^2 C}{\partial x^2} + D_y\frac{\partial^2 C}{\partial y^2} \tag{B.3}$$

In this case, D_x and D_y are the scalar diffusion coefficients in the x and y directions.

For turbulent diffusion, D is the turbulent mass diffusion coefficient. It has the units of $\frac{L^2}{T}$.

The solution of Equation B.3 for a point source is:

$$C(x,y,t) = \frac{M}{2\pi t\sqrt{(D_x D_y)}}e^{(-\frac{x^2}{4D_x t}-\frac{y^2}{4D_y t})} \tag{B.4}$$

where M is the total mass of the original point source (taken as one from this point forward). This is a standard bivariate normal distribution, with the variances in the two directions given by:

$$\sigma_x^2 = 2D_x t \qquad \text{and} \qquad \sigma_y^2 = 2D_y t \tag{B.5}$$

Thus, the variance in the x and y directions grows linearly with time.

For $Dx = Dy = D$, the solution is:

$$C(x,y,t) = \frac{M}{4\pi Dt}e^{\frac{-x^2-y^2}{4Dt}} \tag{B.6}$$

B.2 Approximating Diffusion with a Random Walk

For a random walk with a displacement probability of $P(x, y, t)$, the mean position $(\overline{x}(t))$ remains zero, but the variance $(\overline{x}^2(t))$ grows linearly with time (Csanady 1973). A long series of random steps will converge to a Gaussian distribution with variance growing linearly with time. Csanady (1973) says:

> "Note also that the precise (Gaussian) form of the transition probability distribution is irrelevant, as long as its second moment is $2D\Delta t$, . . ."

The transition probability distribution is the distribution of displacements at each random walk step, and D is the diffusion coefficient in the diffusion equation. So, diffusion can be simulated with a random walk with any distribution, with the resulting diffusion coefficient being one half the variance of the distribution of each step divided by the time step:

$$D_x = \frac{1}{2}\frac{\sigma_x^2}{\Delta t} \tag{B.7}$$

B.3 Particular Random Walks

In HAZMAT software, we have used a variety of forms for the random walk in simulating diffusion in Lagrangian element (LE) models.

B.3.0.1 The OSSM Method

The OSSM method computes a $\Delta x, \Delta y$ from an input diffusion coefficient, and at each diffusion time step, randomly places each LE at $x \pm \Delta x$ and $y \pm \Delta y$. Currently, $\Delta x = \Delta y$, but this condition is not strictly required, and we might want to allow anisotropic diffusion in the future. The result is the distribution shown in Figure B.1.

I tried fixing the position of delta x in the figure to no avail. The original pdf might need to be altered. DAH

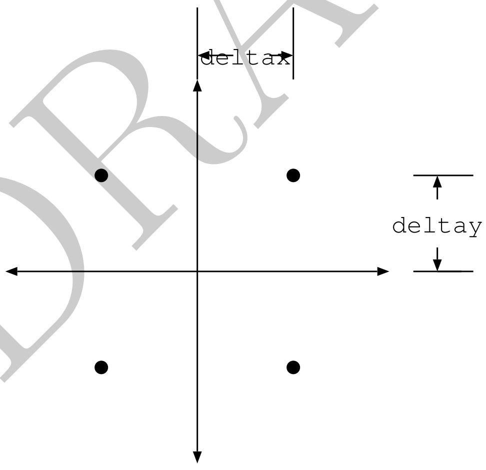


Figure B.1: Schematic of the OSSM method for computing a random walk

The variance of this distribution is:

$$\sigma_x^2 = \int_{-\infty}^{\infty} x^2 \left(\frac{1}{2}\delta(x - \Delta x) + \frac{1}{2}\delta(x + \Delta x) \right) dx = \Delta x^2 \tag{B.8}$$

and similarly for σ_y^2, with δ being the Dirac delta function. From Equation B.7:

$$\Delta x = \sqrt{2D_x \Delta t} \qquad \text{and} \qquad \Delta y = \sqrt{2D_y \Delta t} \tag{B.9}$$

B.3.0.2 The GNOME Method

The GNOME method computes a $\Delta x, \Delta y$ from an input diffusion coefficient, and at each diffusion time step, chooses a dx and dy randomly from a uniform distribution, such that $-\Delta x \leq dx \leq \Delta x$ and $-\Delta y \leq dy \leq \Delta y$. As with OSSM, $\Delta x = \Delta y$. The result is the distribution shown in Figure B.2.

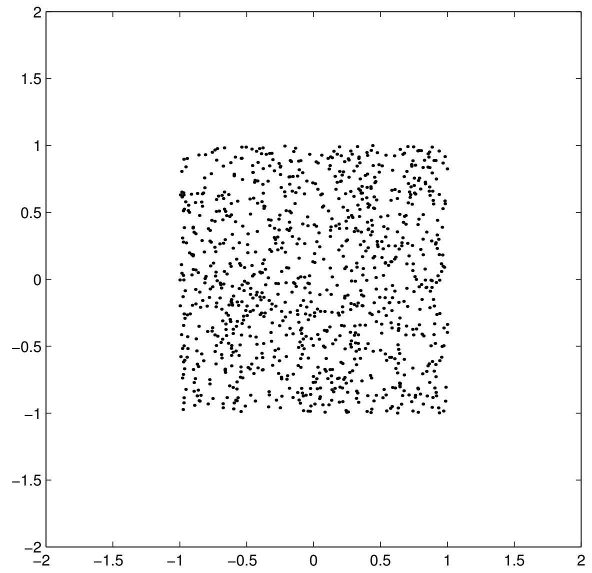


Figure B.2: Schematic of the GNOME method for computing a random walk

The variance of this distribution is:

$$\sigma_x^2 = \int_{-\Delta x}^{\Delta x} \frac{x^2}{2\Delta x} dx = \frac{\Delta x^2}{3} \tag{B.10}$$

and similarly for σ_y^2. From Equation B.7:

$$\Delta x = \sqrt{6 D_x \Delta t} \qquad \text{and} \qquad \Delta y = \sqrt{6 D_y \Delta t} \tag{B.11}$$

B.3.1 Experimental Results

To confirm the accuracy of our math, we did a few computational experiments. We computed the movement of many LEs using random walk approaches, and compared the results to LEs computed from the analytical solution. The results were comparable.

Refer to one or both random walk sections? DAH

In Figures B.3 to B.6, the top three plots show the LEs themselves, and the bottom three plots are the cumulative distribution of the x-coordinates of the LEs, plotted on top of the cumulative distribution of the analytical solution.

After 24 time steps (Figure B.6), the cumulative distribution of all the solutions approximates the analytical solution. In the OSSM method, the representation of LEs themselves does not appear as well matched because the LEs have all been moved by multiples of Δx, so many of them are on top of one another. When running a complete trajectory, the variable windage will spread out the LEs, so that the results are better matched.

In the first few time steps, the OSSM solution does not approximate the analytical solution, but the GNOME solution is well matched after only two steps (Figure B.4).

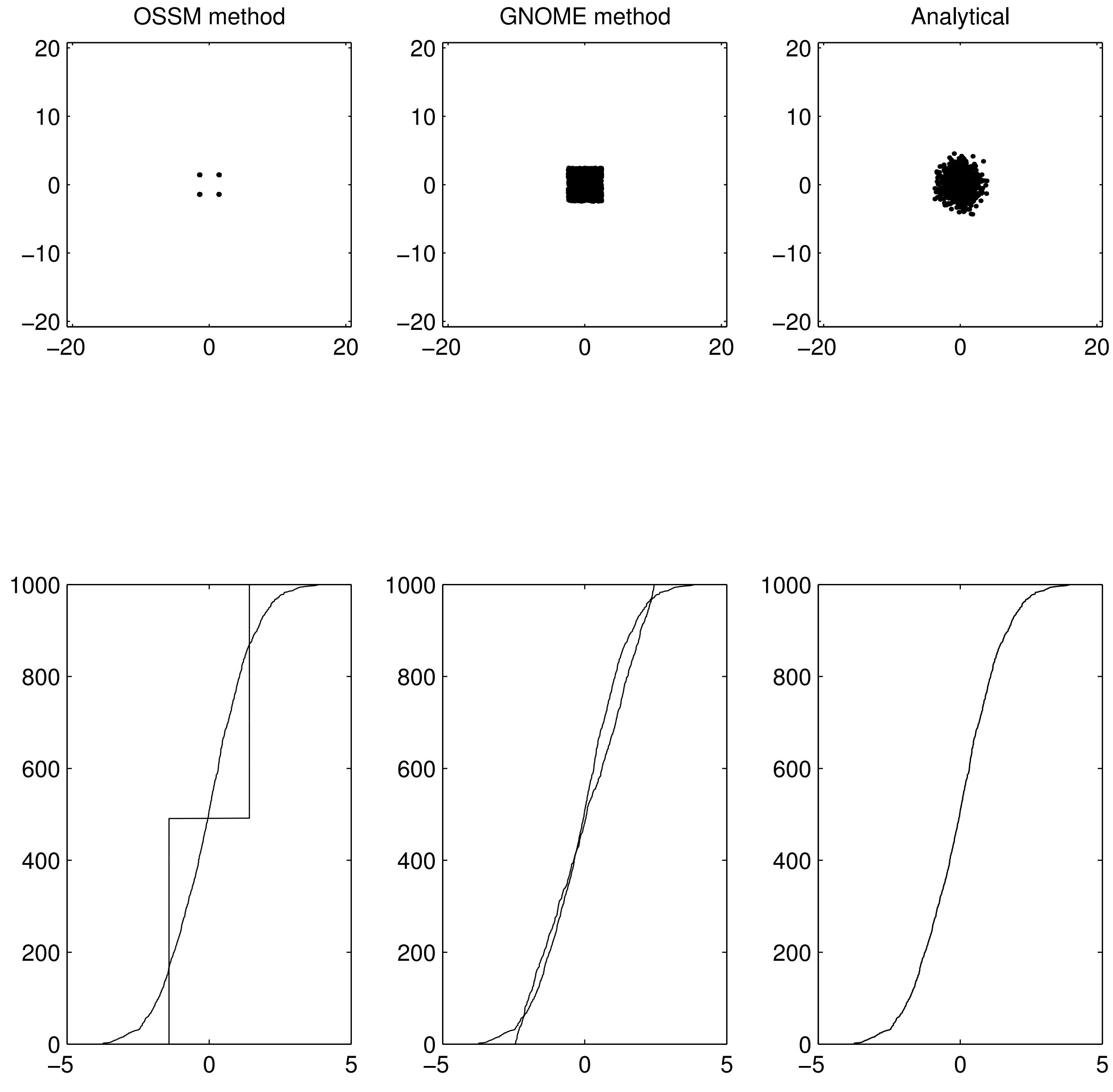


Figure B.3: Comparison of results of computing a random walk after 1 step

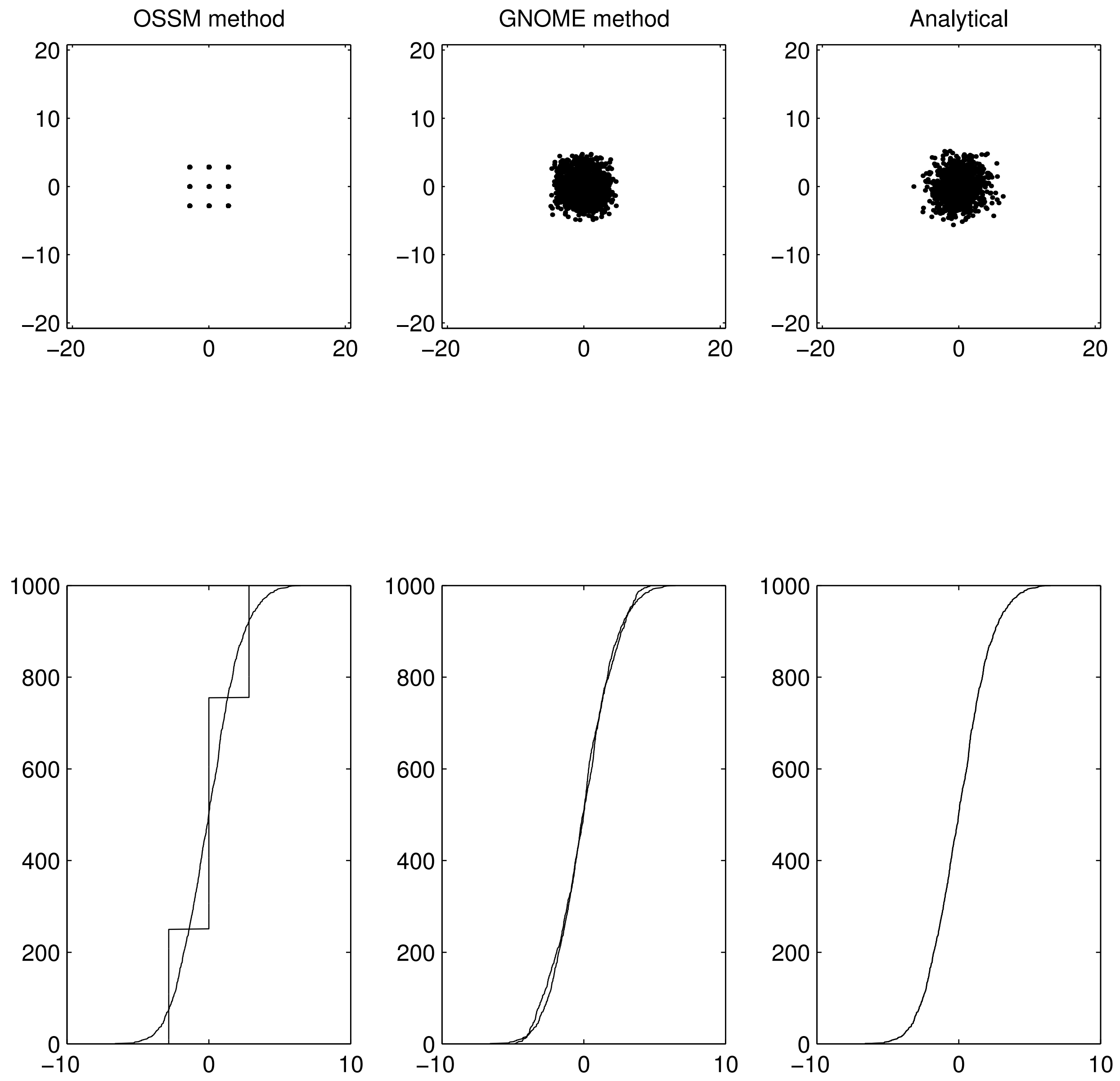


Figure B.4: Comparison of results of computing a random walk after 2 steps

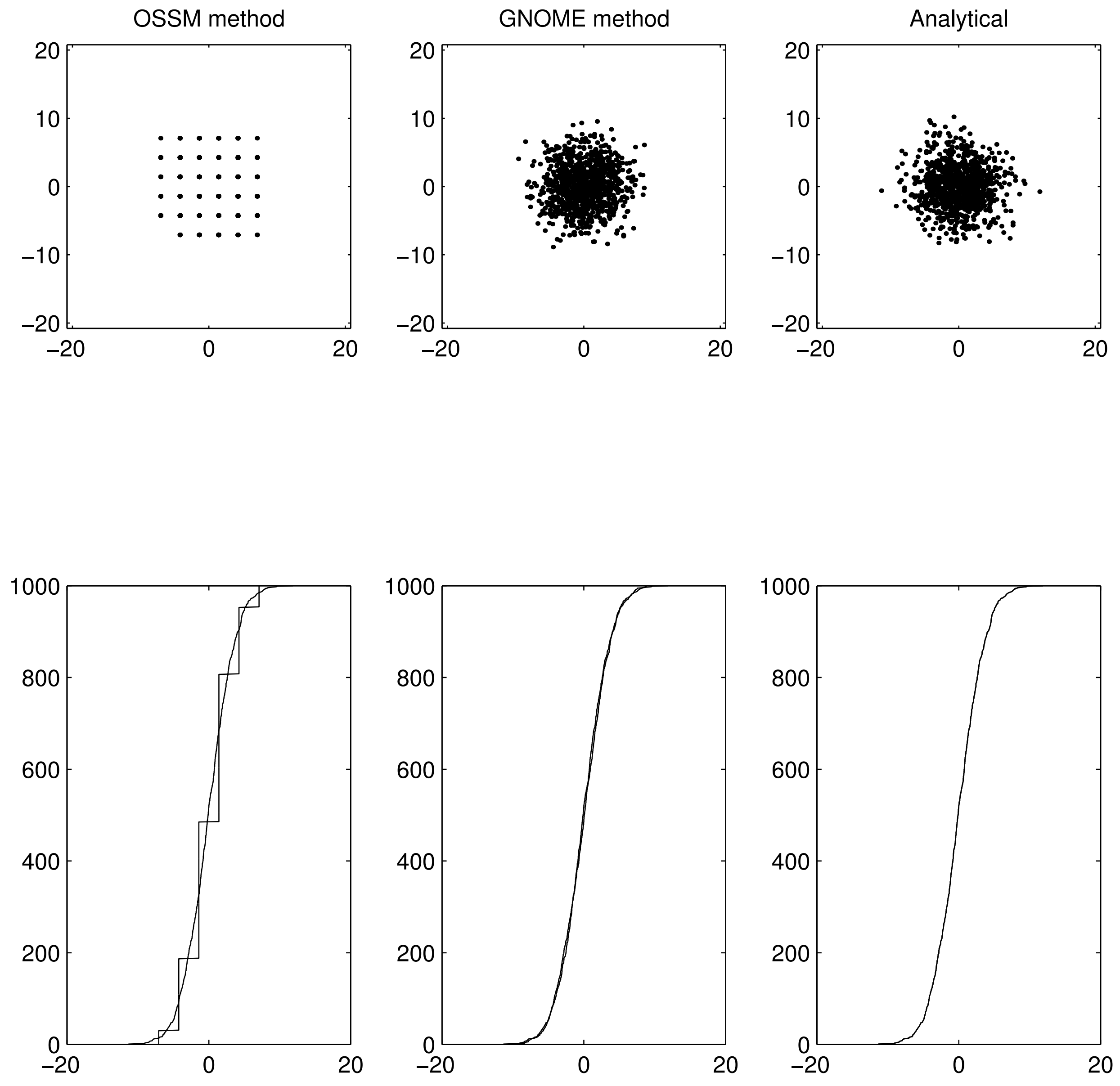


Figure B.5: Comparison of results of computing a random walk after 5 steps

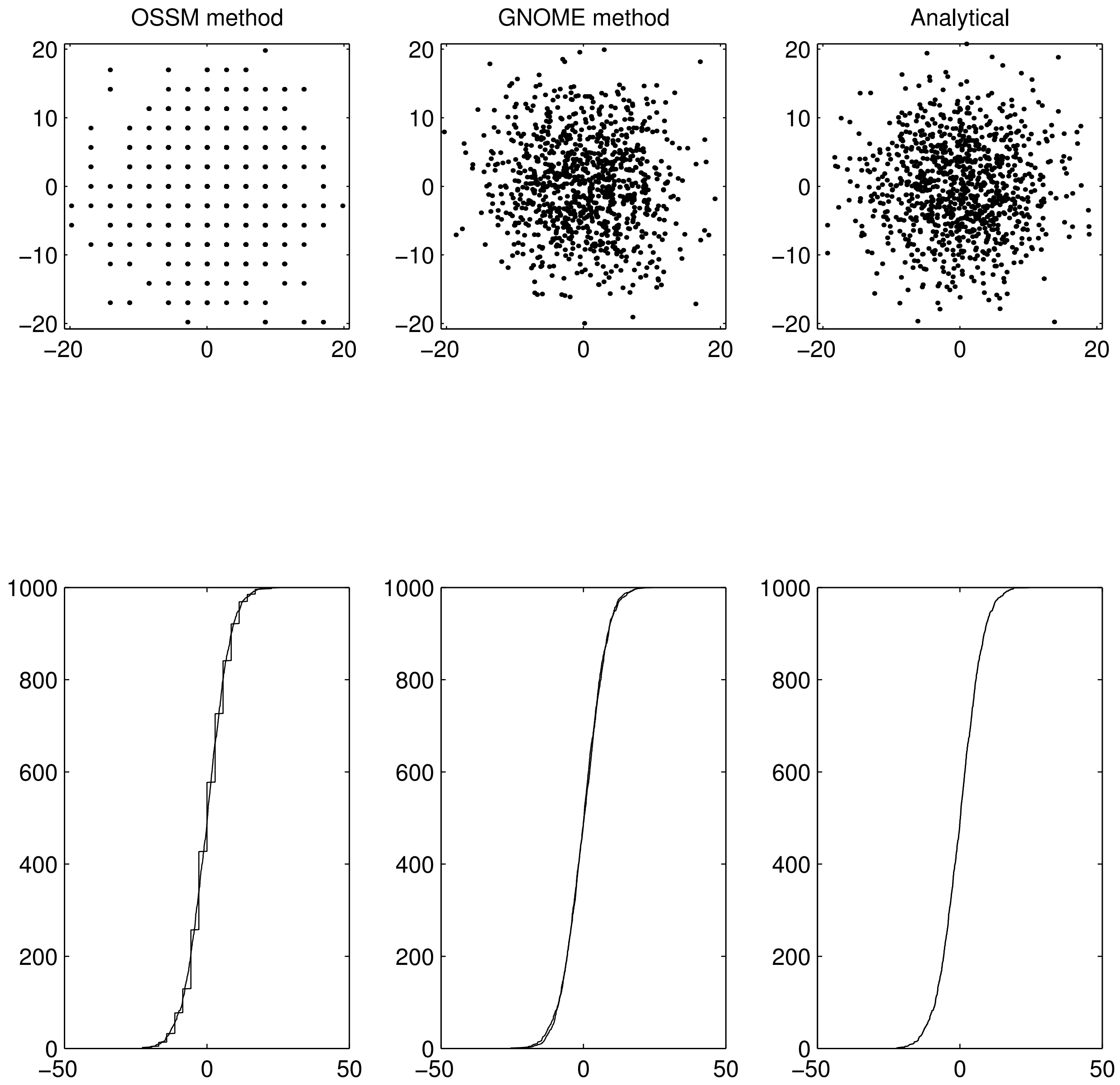


Figure B.6: Comparison of results of computing a random walk after 24 steps

B.3.2 Actual Model Results

The previous method tested how the algorithms in the models work. As a final test to confirm that the actual model output is correct, we ran both OSSM and GNOME with only diffusion, and compared the resulting LE files.

B.3.2.1 What the Code is Supposed to Do

We examined the OSSM code and consulted with Bushy . OSSM is written to implement the algorithm outlined above, but with Δx computed as:

$$\Delta x = \sqrt{D_O \Delta t} \tag{B.12}$$

Bushy? DAH

with D_O being the OSSM diffusion coefficient.

GNOME uses the algorithm outlined above, with Δx computed as in Equation B.11. The result is that:

$$D_O = 2D \tag{B.13}$$

where D is the familiar diffusion coefficient from the diffusion equation, and the one used in GNOME.

B.3.2.2 The Model Results

Figures B.7 to B.10 are the results of the model runs, compared to the analytical solution. The diffusion coefficient for these runs was 1×10^5 cm^2/s. The time step was 1 h for both the model and the diffusion; the two time steps have to be the same for GNOME, but in OSSM, the diffusion time step can be shorter. Figure B.11 is the variance and standard deviation of the X position of the LE locations for the GNOME and OSSM output, compared to the analytical solution. It is clear from these figures that the OSSM is underpredicting the spread of the LEs.

In Figs. B.7 through B.16, please change spelling of "timesteps" to two words. DAH

Figures B.12 to B.15 represent the corrected diffusion coefficient in the GNOME and analytical solutions, according to Equation B.13, setting D to 5×10^4 cm^2/s. Clearly the results are now essentially the same (Figure B.16).

Confirm that correct figure is referenced (B.16) DAH

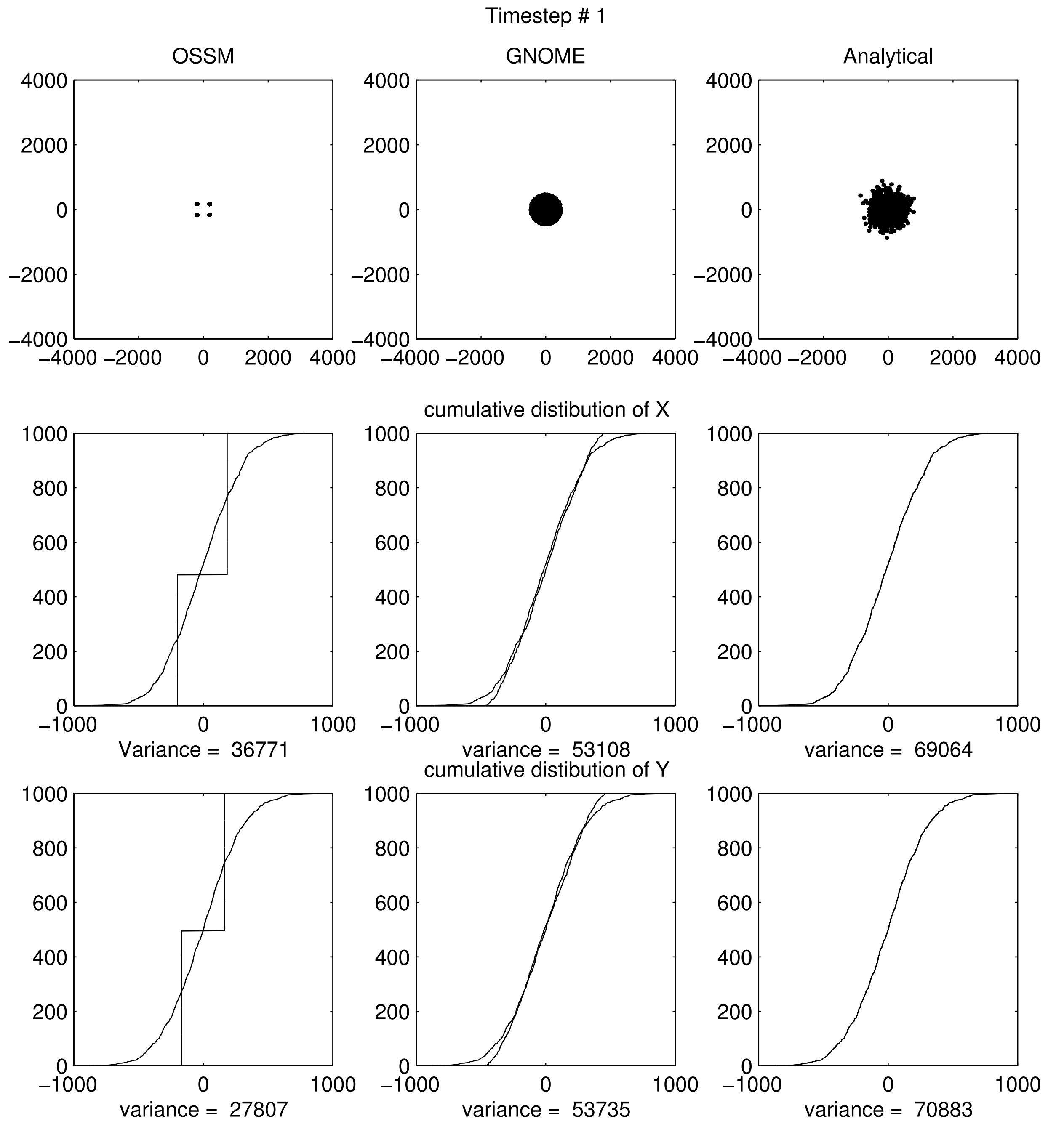


Figure B.7: Comparison of results of OSSM and GNOME after 1 step

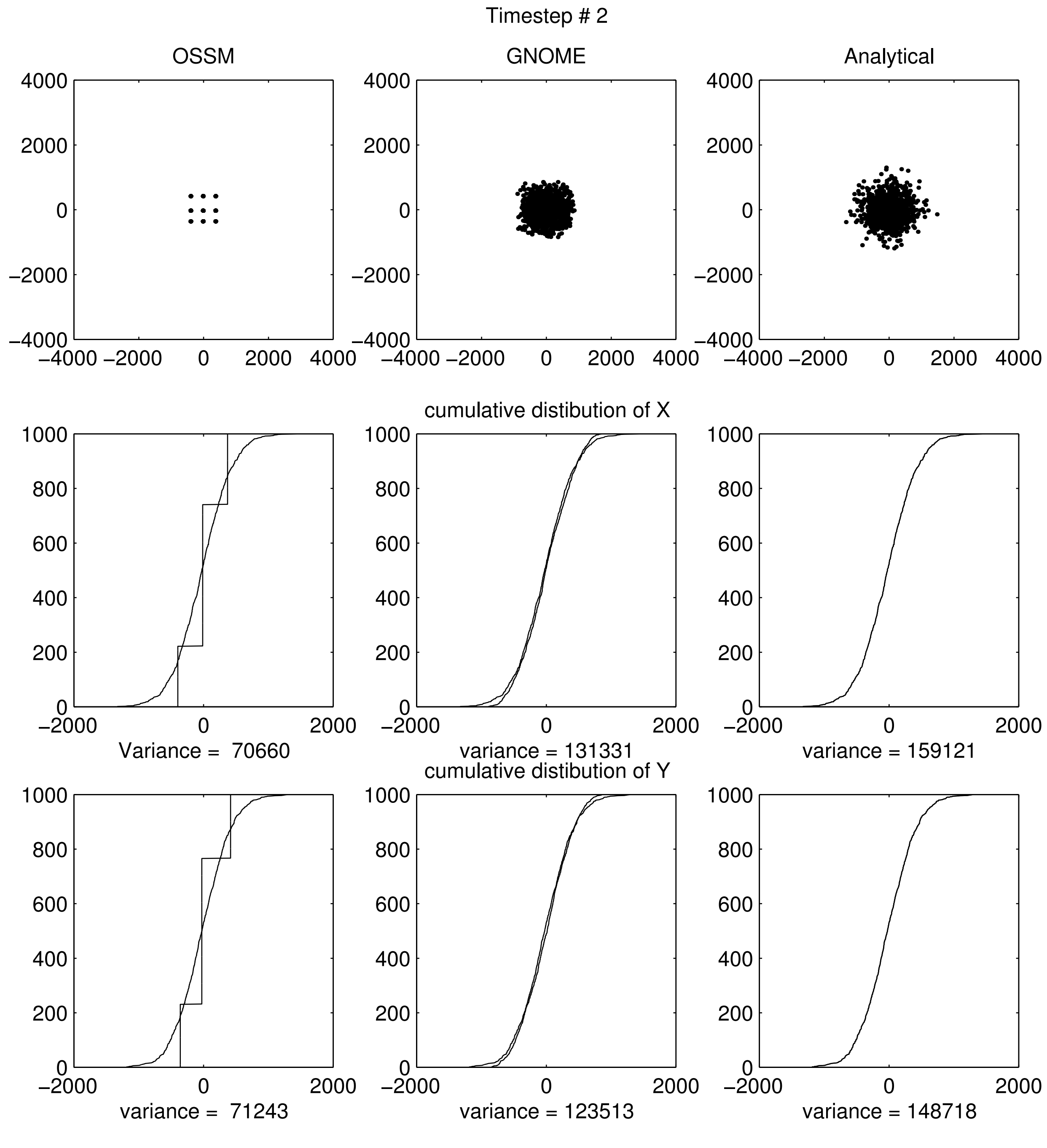


Figure B.8: Comparison of results of OSSM and GNOME after 2 steps

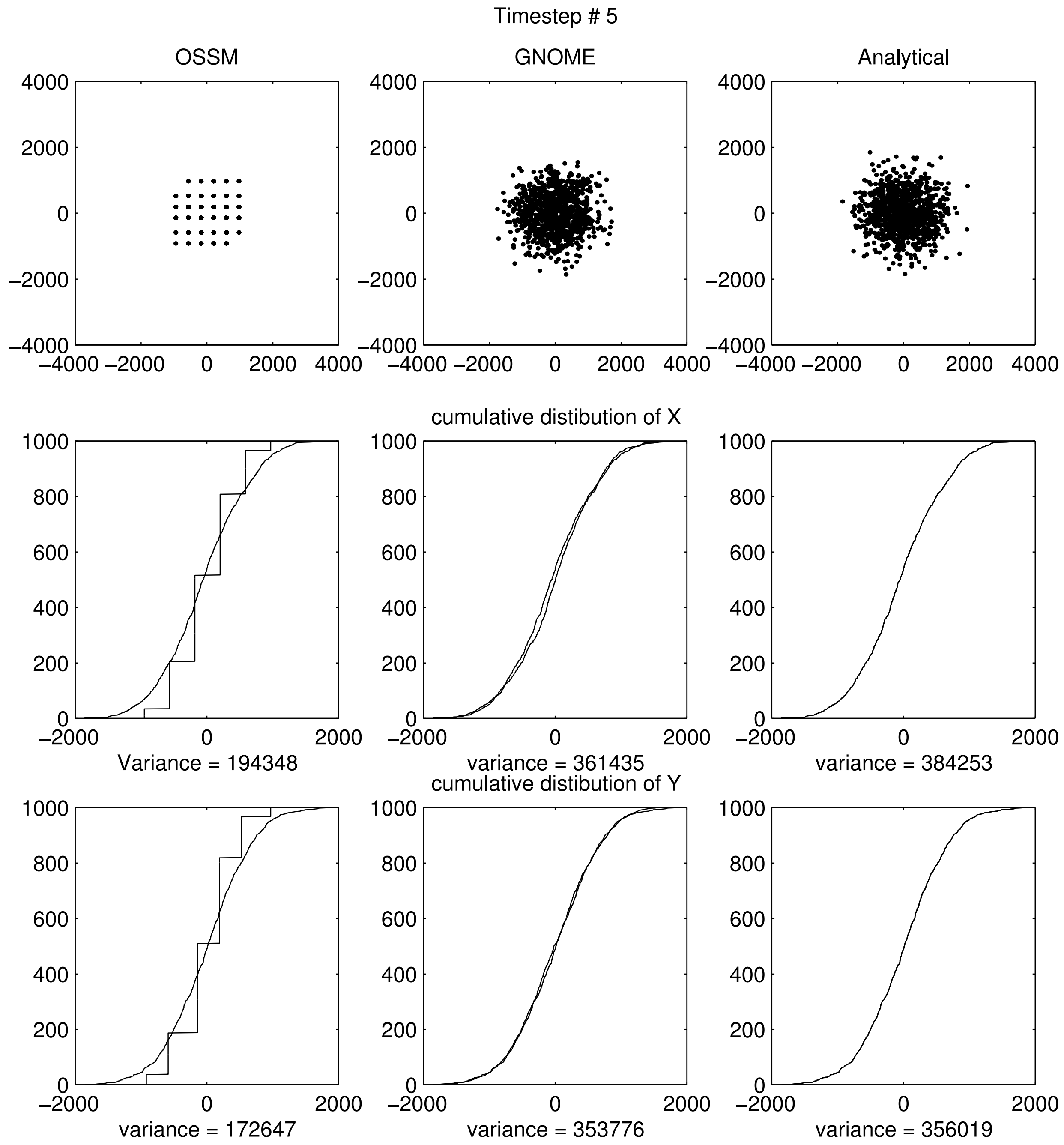


Figure B.9: Comparison of results of OSSM and GNOME after 5 steps

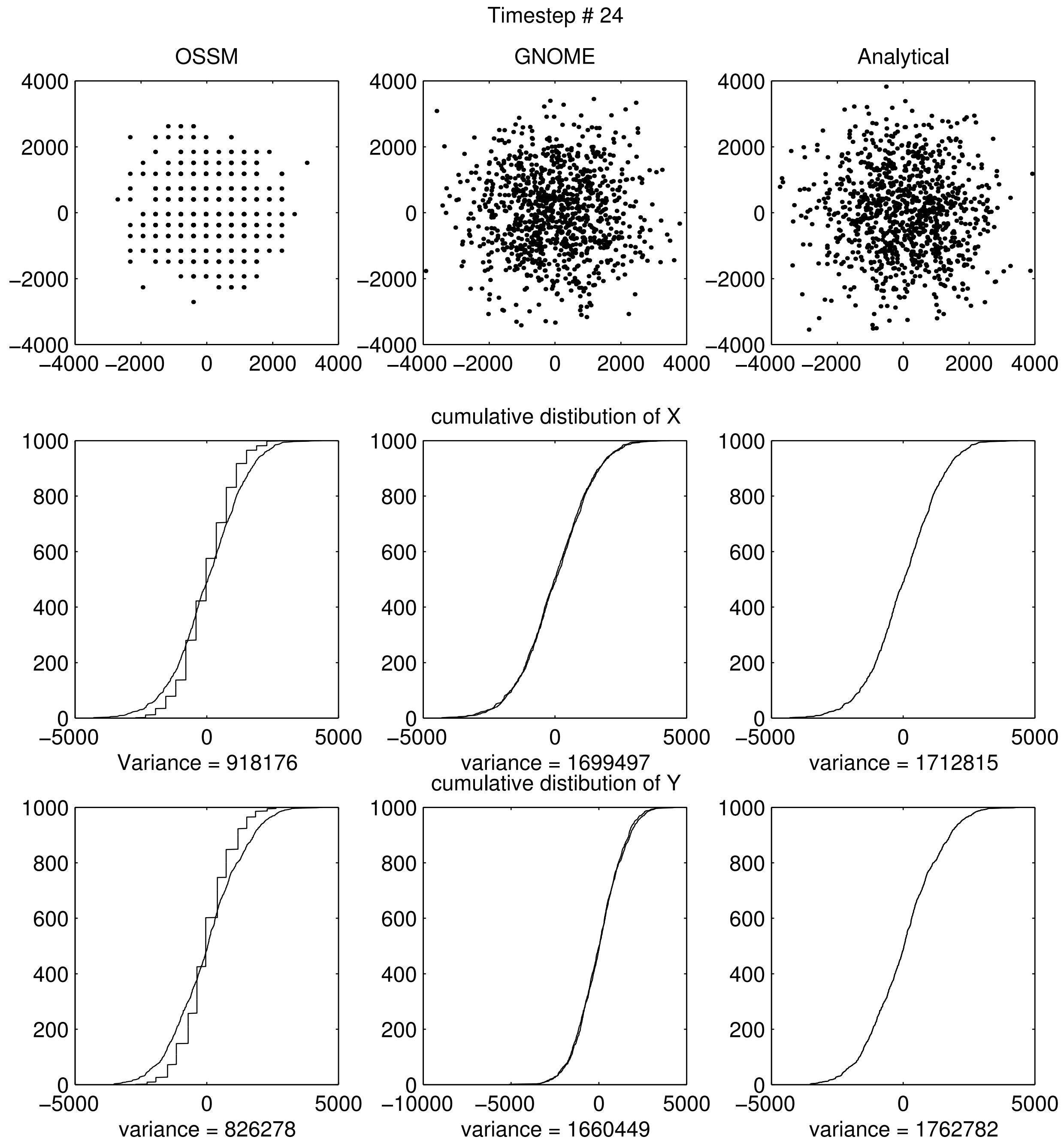
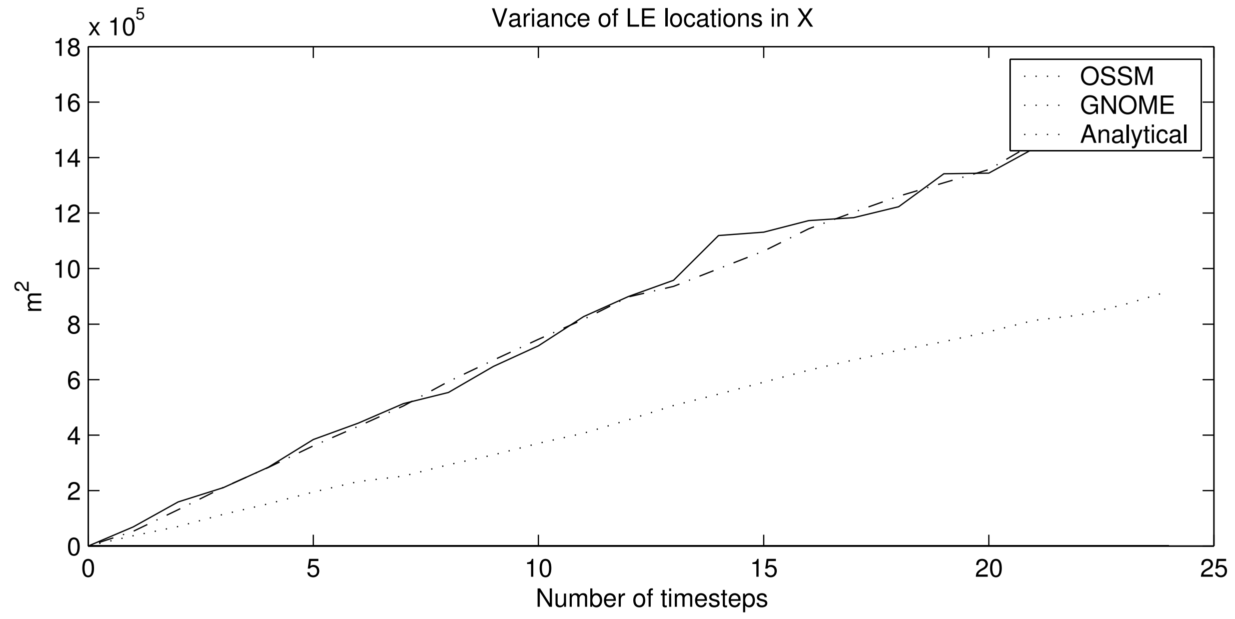


Figure B.10: Comparison of results of OSSM and GNOME after 24 steps



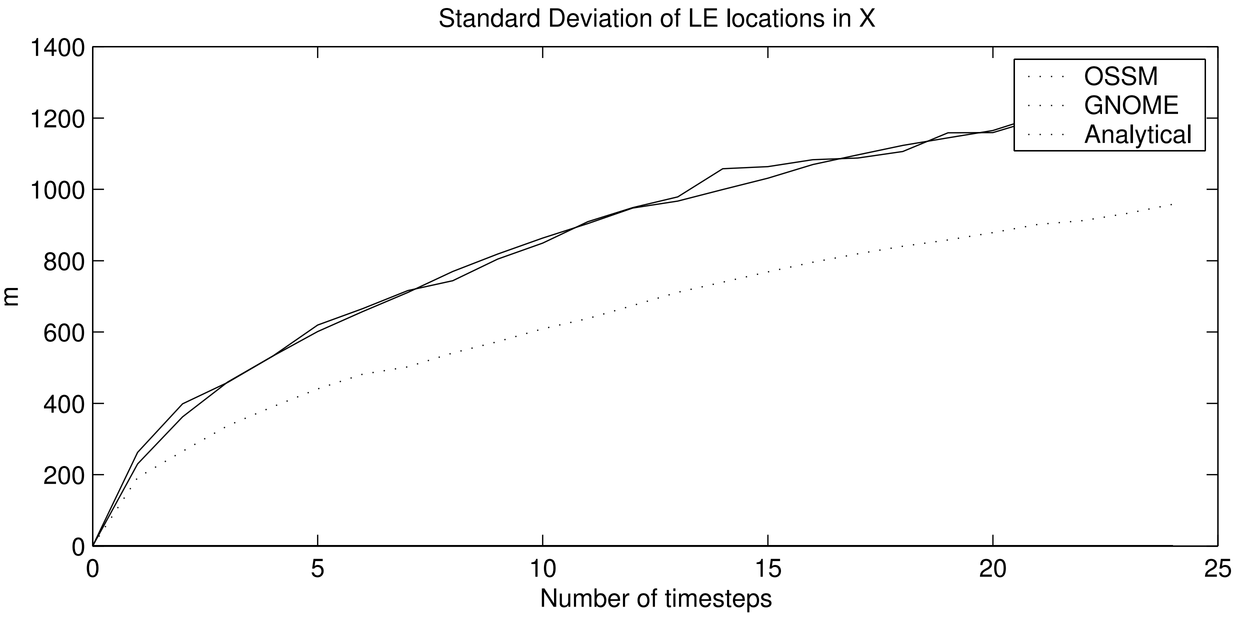


Figure B.11: Comparison of the variance and standard deviation of the X position of the LEs from OSSM, GNOME, and analytical solutions

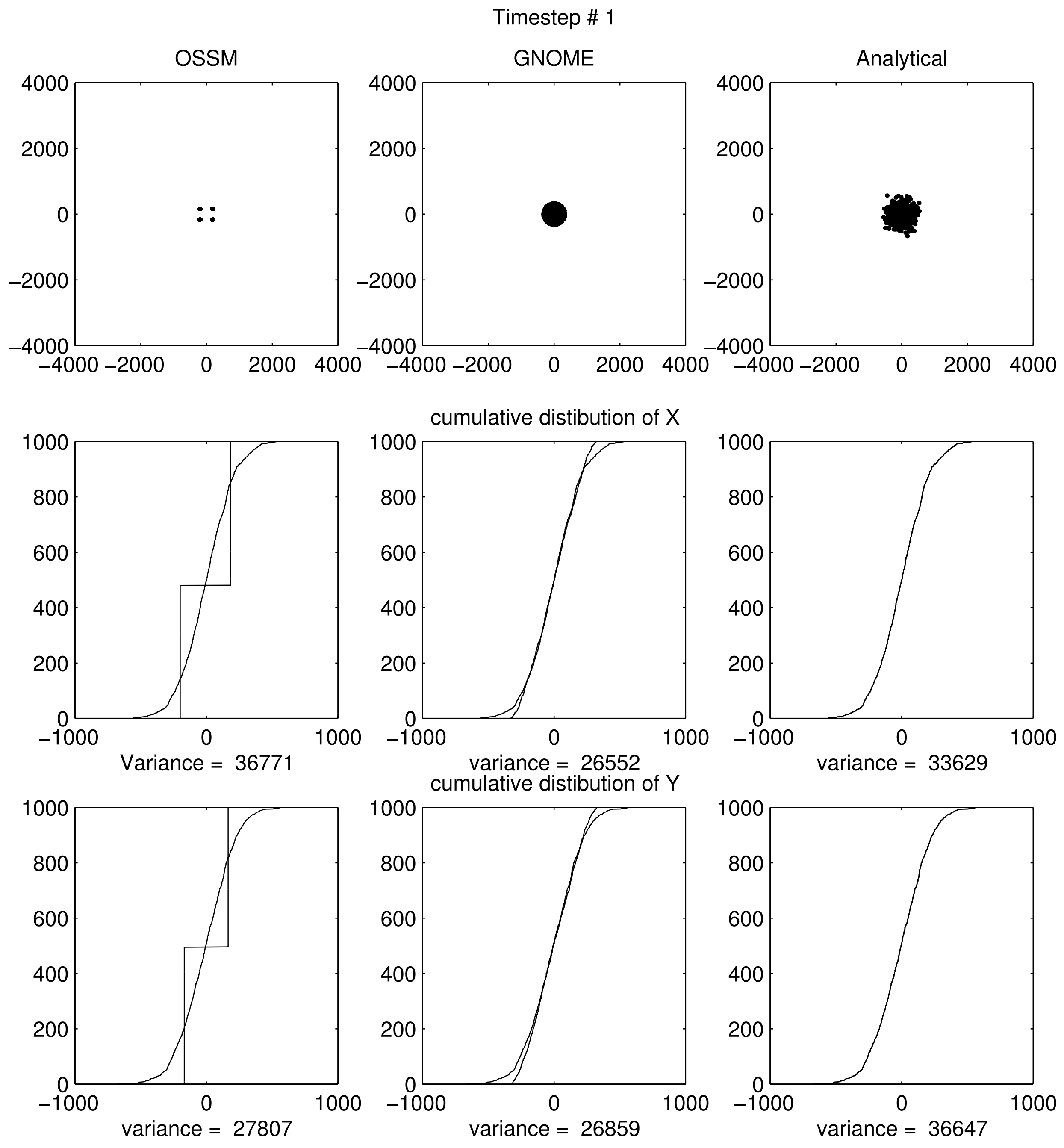


Figure B.12: Comparison of results of OSSM and GNOME after 1 step, with the GNOME D corrected to match the OSSM D_O

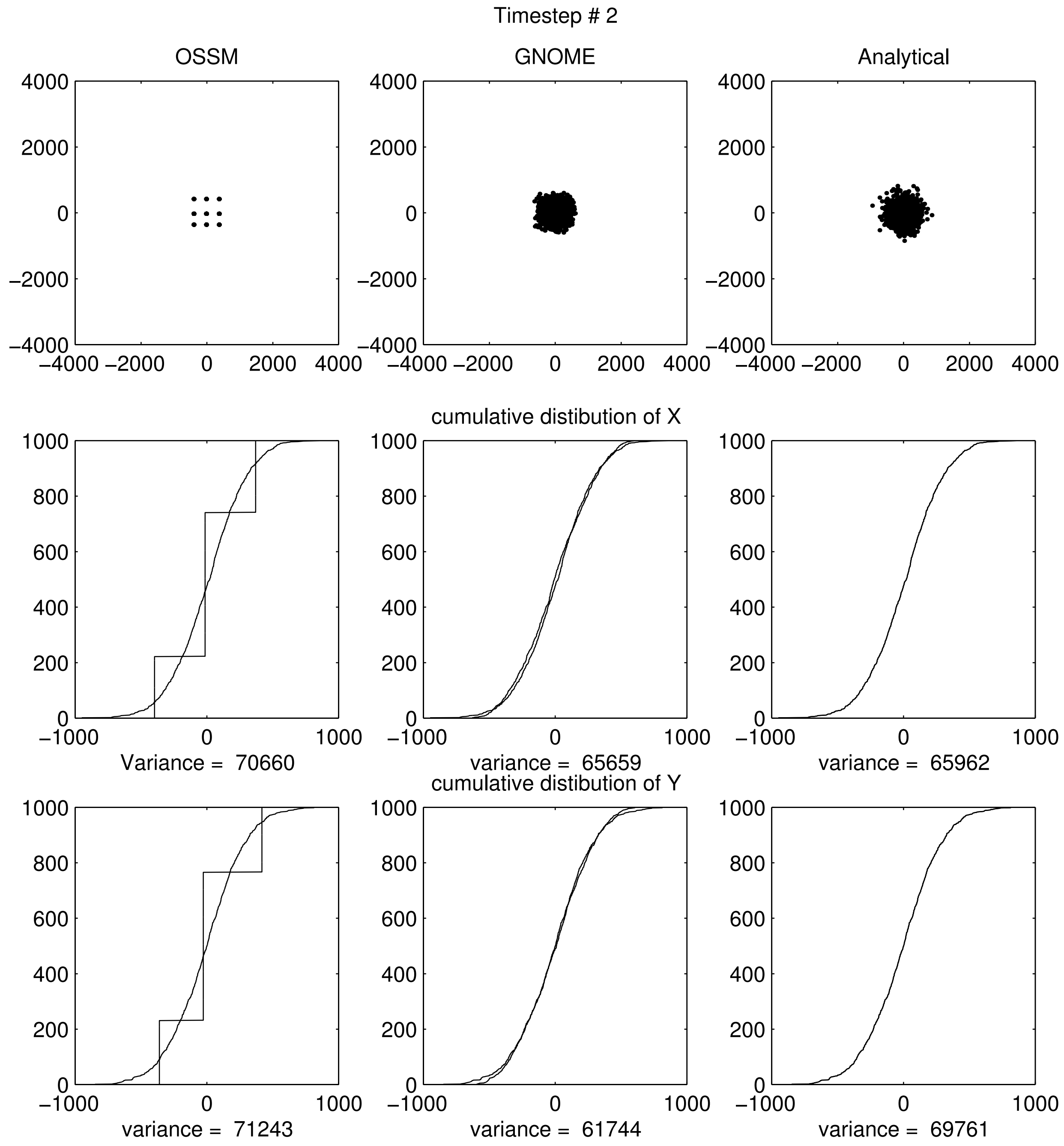


Figure B.13: Comparison of results of OSSM and GNOME after 2 steps, with the GNOME D corrected to match the OSSM D_O

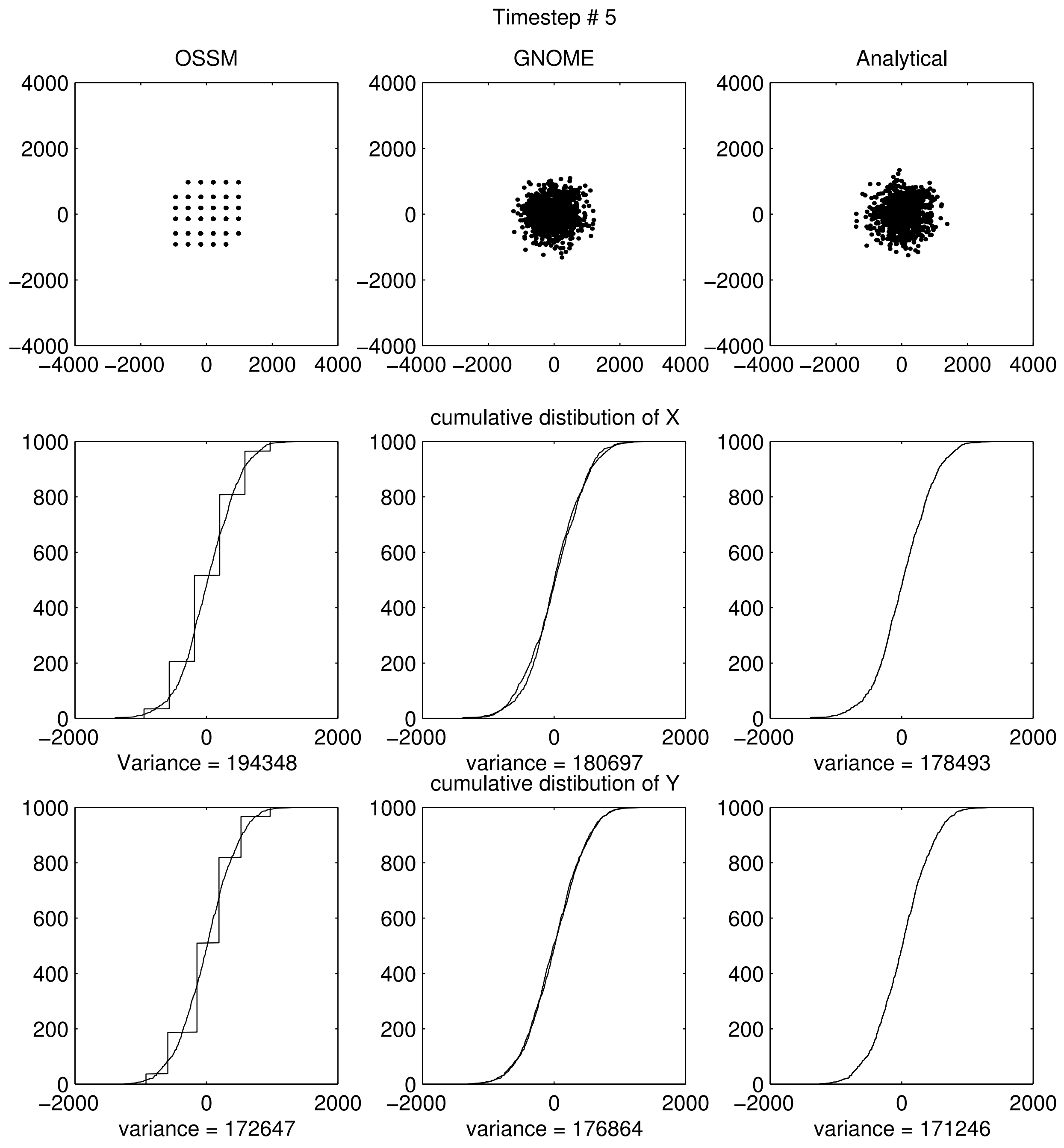


Figure B.14: Comparison of results of OSSM and GNOME after 5 steps, with the GNOME D corrected to match the OSSM D_O

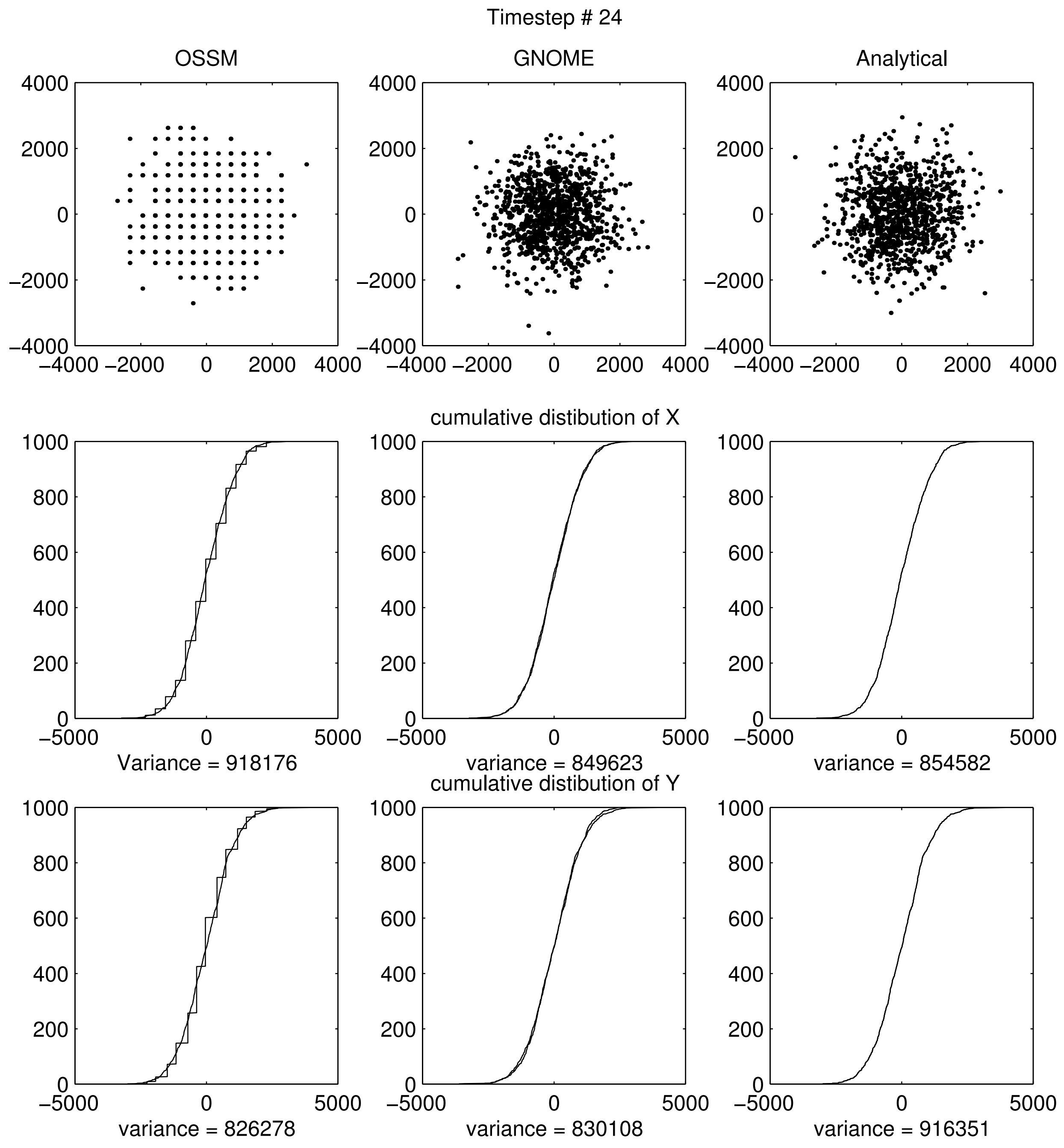
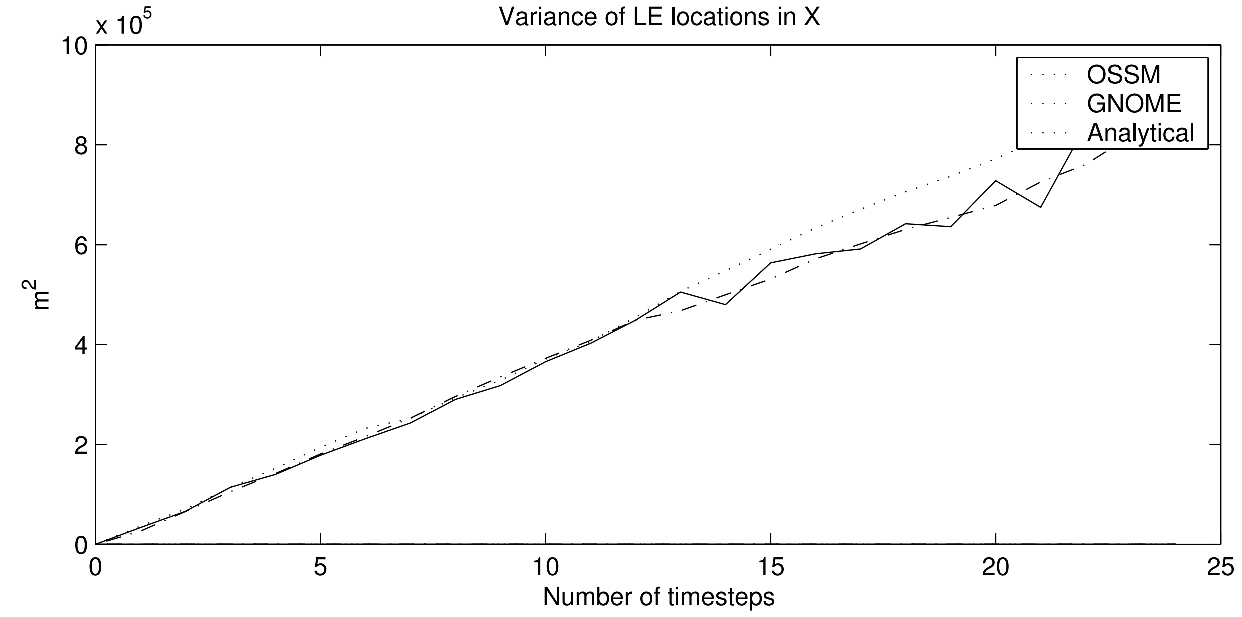


Figure B.15: Comparison of results of OSSM and GNOME after 24 steps, with the GNOME D corrected to match the OSSM D_O



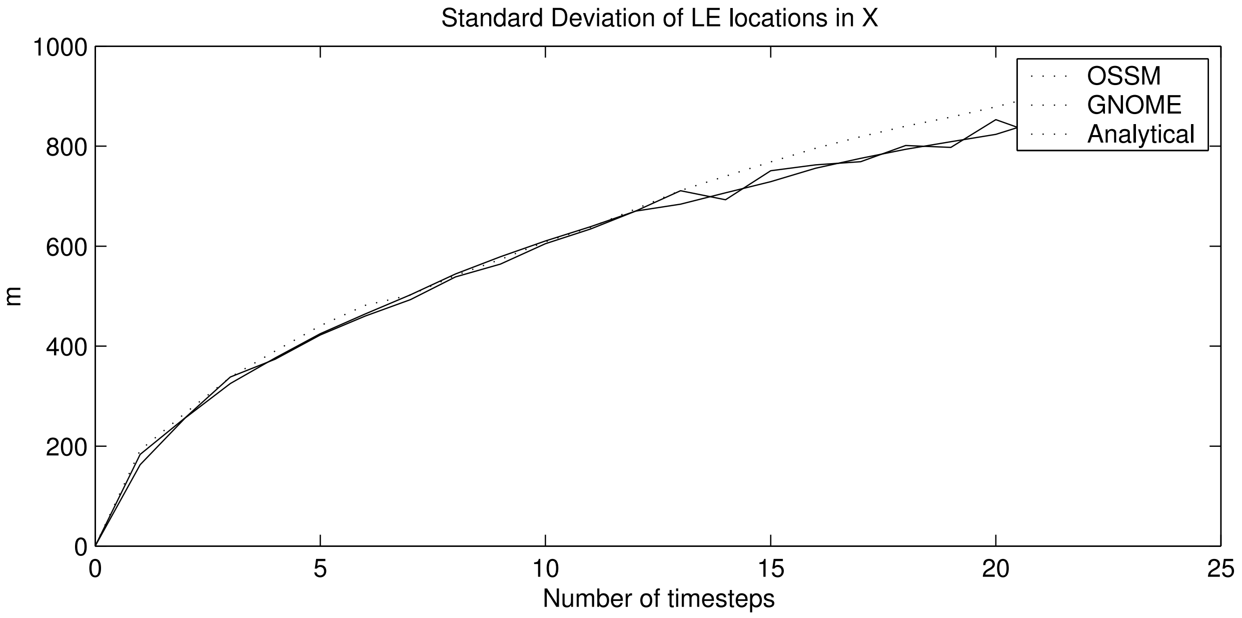


Figure B.16: Comparison of the variance and standard deviation of the X position of the LEs from OSSM, GNOME, and analytical solutions, with D corrected in GNOME and analytical solutions to match the OSSM solution

B.3.2.3 Complications

After running these experiments, it appeared that the conflict between GNOME and OSSM diffusion was resolved. Unfortunately, after running more experiments, another discrepancy was discovered. The problem is that OSSM allows the diffusion time step and the model time step to be set independently, so the model time step is not necessarily an integer multiple of the diffusion step. The diffusion code contains fixes to correct for this. The result of that code, however, changes the effective diffusion coefficient under certain circumstances. The relevant OSSM code is on the following page.

```
C This routine is called once every computational time step for each LE.

SUBROUTINE DODIFF(UDIF,VDIF,ALAT,JNMAP)
      INCLUDE "Run.inc"
      INCLUDE "Diff.inc"
      INCLUDE "IO.inc"
      Real*4 displacement

C       NDIFST is the number of diffusion steps per hour
C       XINT is the computational time step in hours
C       ADIF is the random displacement every diffusion time step
C       UDIF is the X net displacement
C       VDIF is the net Y displacement

      UDIF=0.0
      VDIF=0.0
      IF(NDIFST.EQ.0) GOTO 999

C Note here that NTIMES is the computational time step rounded down to
C integer hours
C If it is less than 1, then it is set to 1.

      NTIMES=INT(ABS(XINT))
      IF(NTIMES.LT.1) NTIMES=1

C This is the displacement per diffusion step scaled by the computational
C time step divided by the rounded time step. If your computational time
C step is in whole hours, then displacement = ADIF.  If your computational
C time step is less than an hour then XINT/NTIMES = XINT.

      displacement = ADIF*XINT/NTIMES

C The outer loop is the number of hours your computational time step is
C with a minimum of 1.
C The inner loop is the number of diffusion steps per hour.

      DO 150 I=1,NTIMES

C Note that we ALWAYS loop through NDIFST times.
C This is the number of steps per hour even if your computational
C time step is less than 1 hr. Displacement was calculated to
C compensate by scaling down the displacement by XINT/NTIMES

      DO 100 K=1,NDIFST
        AEW=1
        ANS=1
        R1=RANF()
        R2=RANF()
        IF(R1.GT.0.5) AEW=-1.0
        IF(R2.GT.0.5) ANS=-1.0
        UDIF=UDIF+AEW*displacement
        VDIF=VDIF+ANS*displacement
 100  CONTINUE
 150  CONTINUE

C We now convert from km to lat and lon degrees and return.
          UDIF=XEW(UDIF,ALAT,JNMAP)
          VDIF=YNS(VDIF,JNMAP)
 999  RETURN
      END
```

This routine is called once for each LE, for each model time step. The diffusion time step is defined as "steps per hour" (`NDIFST`), so the OSSM time step is truncated to integer hours. This sets the time step to zero if it started at less than one hour, so it is rounded up to 1 hour. This ensures that the defined number of steps per hour is in fact the minimum number of diffusion steps per OSSM step.

The code has changed the diffusion time step, so it must also change the displacement step to compensate. This is done with the line:

```
displacement = ADIF*XINT/NTIMES
```

Unfortunately, this compensation is incorrect, and the result is a change in the resulting diffusion, so that the spreading predicted by OSSM is a function of the input diffusion coefficient, and of both the model and diffusion time steps.

From Equation B.9, Δx is proportional to the square root of the time step, so the displacement should be adjusted by the square root of the ratio of the new time step to the old time step:

```
displacement = SQRT(ADIF*XINT/NTIMES)
```

In the interest of protecting old files, the code will stay the same, and this correction will not be added. A new "GNOME compatible" mode will use the same algorithm as GNOME, except that it will still allow a smaller diffusion time step than model time step. The resulting spreading will match the analytical solution, regardless of chosen time steps (within numerical limits).

B.3.2.4 The Corrections

To match GNOME output to OSSM output that uses the old algorithm, these are the corrections required for the diffusion coefficient:

- If the OSSM model time step (Δt_O) is an integer number of hours, the same correction as presented earlier applies:

$$D = \frac{D_O}{2} \quad \text{or} \quad D = \frac{(\Delta x_O \cdot 1 \times 10^5)^2}{2 \cdot \Delta t_D \cdot 3600} \tag{B.14}$$

  where:
  D_O is the OSSM diffusion coefficient,
  Δx_O is the diffusion displacement step saved and displayed by OSSM (in km), and
  Δt_D is the OSSM diffusion time step (in h), with the result converted to cm^2/s.

- If Δt_O (in h) is less than one hour:

$$D = \frac{D_O \cdot \Delta t_O}{2} \quad \text{or} \quad D = \frac{(\Delta x_O \cdot 1 \times 10^5)^2 \cdot \Delta t_O}{2 \cdot \Delta t_D \cdot 3600} \tag{B.15}$$

- If Δt_O is greater than one hour:

$$D = \frac{D_O \cdot \Delta t_O}{2 \cdot \text{floor}(\Delta t_O)} \quad \text{or} \quad D = \frac{(\Delta x_O \cdot 1 \times 10^5)^2 \cdot \Delta t_O}{2 \cdot \Delta t_D \cdot 3600 \cdot \text{floor}(\Delta t_O)} \tag{B.16}$$

  where the floor() function rounds down to the nearest integer hour.

Add Δt_O inside parentheses? DAH

B.3.3 Conclusions

- Diffusion can be simulated in a Lagrangian element model by a random walk, with virtually any distribution for the random steps taken. Whatever distribution is used, the variance of that distribution should be:

$$\sigma^2 = 2D\Delta t \tag{B.17}$$

  where D is the diffusion coefficient in the diffusion equation, which is equal to the eddy diffusivity in a turbulent diffusion model.

- Both the GNOME and OSSM methods work well, with GNOME's method giving results closer to the analytical solution in fewer time steps.
- The diffusion coefficient used in GNOME is the same one users are familiar with from the diffusion equation.
- The diffusion coefficient used in OSSM is different than that used in GNOME, and the effective diffusion is a function of both OSSM time step and the diffusion time step. The effective diffusion coefficient for integer-hour OSSM time steps is one half the input coefficient. For non integer-hour OSSM time steps, the effective coefficient can be computed from the formulas given in Section B.3.2.4.

Appendix C

LE Windage Math

C.1 Background

For many years at NOAA, the movement of oil by wind has been described as 1–4% of the wind speed. This is based on a combination of an analytically derived factor (3% of wind speed), and on the empirical observation that oil tends to spread in the direction of the wind. After much experimentation and observation, using 1–4% of wind speed was shown to work well, and could be adjusted when overflights indicated that the oil was spreading in the direction of the wind more or less so than the model runs predicted.

To understand the physics of this phenomenon, imagine an oil droplet being pushed up and down from the surface of the water by wave action etc. so it moves faster or slower depending on how close to the surface it is at any moment. There has been mpore recent work better quantifying this phenomenon (Galt and Overstreet 2014; Zeinstra-Helfrich, Koops, and Murk 2017), but it has not been included in GNOME at this time.

C.2 The 3 Percent Rule

¡¡¡¡¡¡ HEAD ¡¡¡¡¡¡ HEAD Figure C.1 represents the time-averaged velocities above and below the water surface, in a reference frame moving with the "undisturbed" current below the water surface (i.e., the current without the wind effect). Ua is the undisturbed "air" velocity, In this case U_w would be 0, and U_o is the velocity of the (oil on the) water surface.

Figure C.1 represents the time-averaged velocities above and below the water surface, in a reference frame moving with the "undisturbed" current below the water surface (i.e., the current without the wind effect). U_a is the undisturbed "air" velocity (the velocity outside of the boundary layer), and U_o is the velocity of the (oil on the) water surface. In this case, wind velocity (U_w) would be 0.

C.2.1 The Analysis

Assuming a turbulent boundary layer and steady state conditions, the shear stress on the surface of the water (τ) is given by:

$$\tau = C_D \rho U^2 \tag{C.1}$$

where:
C_D is a coefficient of drag, and
ρ is the density of the fluid, and
U is the velocity difference between air and watter surface.

The stress of the water on the air must be the same as that of the air on water, so:

$$C_D \rho_a (U_a - U_o)^2 = C_D \rho_w (U_o)^2 \tag{C.2}$$

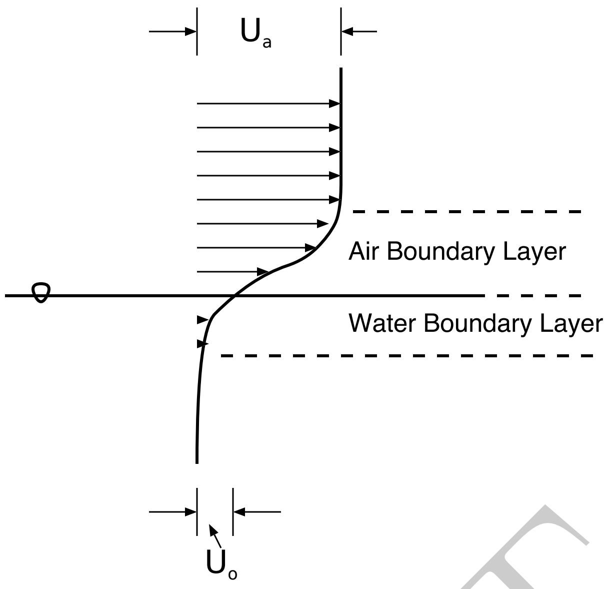


Figure C.1: Air and water boundary layers for a wind-driven surface current

where:
ρ_a is the density of air,
ρ_w is the density of water,
U_a is the velocity outside of the boundary layer, and
U_o is the velocity of the surface layer of water.

We do not need an estimate for C_D because it is the same on both sides of the equation. Rearrangement results in:

$$\frac{\rho_a}{\rho_w}U_a^2 = (1 - \frac{\rho_a}{\rho_w})U_o^2 + 2\frac{\rho_a}{\rho_w}U_a U_o \tag{C.3}$$

Because $\rho_a \ll \rho_w$, a few terms can be neglected:

$$(1 - \frac{\rho_a}{\rho_w}) \Longrightarrow 1 \tag{C.4}$$

$$2\frac{\rho_a}{\rho_w}U_a U_w \Longrightarrow 0 \tag{C.5}$$

resulting in:

$$U_o = U_a\sqrt{\frac{\rho_a}{\rho_w}} \tag{C.6}$$

Using the values of $\rho_a = 1.2$ kg/m^3 and $\rho_w = 1000$ kg/m^3 produces:

$$U_w = 0.03U_a \tag{C.7}$$

otherwise known as "the 3% rule."

To confirm our scaling simplifications, the exact solution is:

$$\frac{\frac{\rho_a}{\rho_w} - \sqrt{\frac{\rho_a}{\rho_w}}}{\frac{\rho_a}{\rho_w} - 1}U_a \tag{C.8}$$

resulting in:

$$U_w = 0.03065 \ldots U_a \tag{C.9}$$

C.3 Computation

In order to compute the windage, both OSSM and GNOME choose a random number between the range of windage values for each LE, and move it according to that number at each time step. The result is that the LEs spread in the direction of the wind. This method is very similar to the method used to compute diffusion, except that the spreading occurs only in the direction of the wind. The amount of spreading is given by:

$$\frac{d\sigma^2}{dt} = S(t) \tag{C.10}$$

where:
σ^2 is the variance of the locations of the LEs, and
$S(t)$ is a spreading parameter that is a function of time (because wind velocity is a function of time).

For a constant wind, the spreading coefficient (S) would be constant and:

$$\sigma^2 = St \tag{C.11}$$

The variance of the particles grows linearly in time, the same as does diffusion with a constant diffusion coefficient.

Change St to $S(t)$ in equation? DAH

C.3.1 The Problem

With the algorithm used by OSSM and GNOME, at any given time step:

$$S = \frac{\Delta W^2 \cdot U^2 \cdot \Delta t}{3} \tag{C.12}$$

where: ¡¡¡¡¡¡¡ HEAD ¡¡¡¡¡¡¡ HEAD
ΔW is the range of the windage parameters, and
U is the wind velocity at that time step.
=======
ΔW is the range of the windage parameters, and
U_w is the wind velocity at that time step.
¿¿¿¿¿¿¿ 656045eea63295441aeccaacff609b5e47af8664 =======
ΔW is the range of the windage parameters, and
U_w is the wind velocity at that time step.

Should U be U_w? And should S in equation be $S(t)$? DAH

Should S in equation be $S(t)$? DAH

Should S in equation be $S(t)$? DAH

¿¿¿¿¿¿¿ c9c2ff12a42e2dbc72aead8b2f190f8ef4263b6e

It is immediately apparent from this equation that the degree of spreading is a function of both the wind speed, which it should be, and the time step, which it should not. Experiments with GNOME have demonstrated that the spreading does indeed vary with time step.

Why is this the case? If we imagine that any given particle that is moved from the surface travels more slowly for a period of time and then travels faster as it moves back up, the time step is the persistence of this process (i.e., the amount of time a particle spends traveling at a given velocity). At the extremes: 1) infinite persistence would have each particle moving at a different speed, while maintaining that speed the entire time, resulting in much spreading, and 2) infinitely short persistence would involve each particle varying its speed constantly between the extremes, resulting in all the particles moving at the mean velocity, and no spreading.

We concluded that the above situation was undesirable and should be fixed. A model should behave consistently, as much as possible, when the time step is changed.

C.3.2 The Solution

The best way to describe our solution is in terms of persistence. After giving GNOME a range of windage percentages and a persistence time step, it moves the LEs appropriately to achieve the desired amount of spreading. Specifying a windage range is similar to what we have always done, and the persistence can be either fixed at an appropriate number (currently GNOME defaults to a time step of 0.25 h), or it can be adjusted by the user.

C.3.2.1 The Math

We are using a uniform distribution at each time step, so the resulting spreading coefficient (S) is given by:

$$S = \frac{\Delta x^2}{3\Delta t} \tag{C.13}$$

where Δx is the range of distances over which the LEs are distributed. This is computed by multiplying the windage by the wind speed, resulting in Equation C.12.

If the reference windage (W_0) is defined as being between A_0 and B_0 times the wind speed, with a persistence of Δt_0, we have:

$$\Delta W_0 = B_0 - A_0 \tag{C.14}$$

and the mean windage ($\overline{W}_0$) is:

$$\overline{W}_0 = \frac{B_0 + A_0}{2} \tag{C.15}$$

From Equation C.12, we want S to be constant, so we can compute the values for a given time step from:

$$\frac{\Delta W^2 \cdot U^2 \cdot \Delta t}{3} = \frac{\Delta W_0^2 \cdot U^2 \cdot \Delta t_0}{3} \tag{C.16}$$

with:

$$\overline{W} = \overline{W}_0 \tag{C.17}$$

Define $\overline{W}$. DAH

so that:

$$\Delta W = \Delta W_0 \sqrt{\frac{\Delta t_0}{\Delta t}} \tag{C.18}$$

and:

$$A = \overline{W} - \frac{\Delta W}{2}$$
$$B = \overline{W} + \frac{\Delta W}{2} \tag{C.19}$$

Using the new A and B in the same code as before works well.

Define A and B. DAH

Appendix D

ADIOS Oil Datbase

Contents

D.1 Introduction

The Automated Data Inquiry for Oil Spills (ADIOS) database is a software library and database of oil properties that can be used along with GNOME for oil spill modeling, for providing data for other models, as a source of response oil data, or for research and development needs.

Although fully documented elsewhere, this chapter contains the documentation for process of building a "Gnome Oil" that can be used in the model from the data available in the database.

Add an actual reference at some point

D.2 Minimum Requirements

Ideally, an oil will be very well characterized to be used with GNOME, but there is the ability to estimate various properties if they have not been measured. What can be estimated depends somewhat on what kind of oil is being modeled, e.g. crude oils have fairly consistent distillation curves, but refined products do not, so the minimum requirements depend on the product type.

The ADIOS Database defines the following product types: Crude Oil NOS, Tight Oil, Condensate, Bitumen Blend, Bitumen, Refined Product NOS, Fuel Oil NOS, Distillate Fuel Oil, Residual Fuel Oil, Refinery Intermediate, Solvent, Bio-fuel Oil, Bio-Petro Fuel Oil, Natural Plant Oil, Lube Oil, Dielectric Oil, Hydraulic Fluid, Other,

D.2.0.0.1 Minimum requirement for all product types:

- Density at at least one temperature near 15 Celsius
- Viscosity at at least one temperature near 15 Celsius

D.2.0.0.2 Additional requirements for specific product types

Crude Oil NOS
- Nothing

Tight Oil
- At least three distillation cuts up to 400 celsius

Condensate
- At least three distillation cuts up to 400 celsius

Bitumen Blend
- At least three distillation cuts up to 400 celsius

Bitumen
- Can not be modeled

Bitumen Blend
- At least three distillation cuts up to 400 celsius

Refined Product NOS
- At least three distillation cuts up to 400 celsius

Distillate Fuel Oil
- At least three distillation cuts up to 400 celsius

Residual Fuel Oil
- At least three distillation cuts up to 400 celsius

Refinery Intermediate
- At least three distillation cuts up to 400 celsius
- Interfacial tension

Solvent
- At least three distillation cuts up to 400 celsius
- Interfacial tension

Bio-fuel Oil
- At least three distillation cuts up to 400 celsius
- Interfacial tension

Bio-Petro Fuel Oil
- At least three distillation cuts up to 400 celsius
- Interfacial tension

Natural Plant Oil
- At least three distillation cuts up to 400 celsius
- Interfacial tension

Natural Plant Oil
- At least three distillation cuts up to 400 celsius
- Interfacial tension

Lube Oil
- At least three distillation cuts up to 400 celsius
- Interfacial tension

Dielectric Oil
- At least three distillation cuts up to 400 celsius
- Interfacial tension

Hydraulic Fluid
- At least three distillation cuts up to 400 celsius
- Interfacial tension

D.3 Bulk Properties

In order to support its weathering algorithms, GNOME requires density and viscosity. Density and viscosity both change with temperature, so any data available for density and viscosity at various temperatures is used.

Density and Viscosity both also change as the oil weathers – if an oil record contains data about lab-processed data that includes these properties, those are included as well.

D.4 Building Pseudocomponents

As discussed in 6.2, the GNOME weathering model uses a pseudo component (PC) approach: modeling the oil as a finite set of components that share similar properties. An oil assay does not directly have the data for the pseudo components – they must be constructed from the available measured data.

D.4.0.1 Distillation Curves

As oil is a mixture of many compound with a range of boiling points, different fractions will boil off at different temperatures. This distillation process is used to refine petroleum into useful products, and thus a distillation curve is a commonly produced as part of an oil analysis.

The distillation cuts can be presented in two ways: either broken down into specific temperature ranges, or specific fractions. But either way, there is fraction as a function of temperature – i.e. a cumulative distribution.

add plots here?

There are multiple methods to produce the curve, including actual distillation and "simulated distillation", in which the boiling points of the fractions are computed by gas chromatography. In all methods, the result is a finite set of "cuts", or mass fractions with given boiling points that can be interpolated to yield a cumulative distribution of boiling point.

Caution: while all method produce a cumulative distribution, there are some caveats on how to intepret the data:

D.4.0.1.1 Mass vs Volume Fraction Traditional distillation measured the volume of product that was boiled off at each temperature. Simulated distillation usually produces a mass fraction. Since the densities of the fractions are not the same, the two are not equivalent. This is further complicated by the fact that volume is not necessarily conserved when mixing different components. The ADIOS DB code currently assumes mass fraction – and the difference is probably within the error of oil weathering calculations. Adjustments for volume fraction may be made in future versions.

D.4.0.1.2 recoveredfraction Some simulated distillation methods only work up to a certain temperature. This means that only the fraction below a certain boiling point is included, known as the "recovered fraction". If the recovered fraction is known, the properties of the unrecovered fraction are unknown, but at least it is known that it is very high boiling point compounds. If the recovered fraction is unknown, then there is no way to assign a mass fraction to the high boiling point PCs, and it is impossible to provide accurate modeling of the fate of that oil.

D.4.0.1.3 Interpolating the Distillation Curve The Psuedo components used in GNOME will not generally line up with the distillation cuts available in an oil record. So we need a continuous curve from which to derive the properties of the PCs.

For crude oils, the distillation curve is usually fairly linear, but for "odd" oils, and particularity refined products, that is not the case. So with no known structure to the curve, linear interpolation is about the best we can do. This is a straightforward calculation between the known data points, but there are complications at the ends of the data – it is often not known what the smallest or initial boiling point (IBP) is, or the maximum boiling point is of the heavy fraction.

D.4.0.1.4 Extrapolating the IBP Robert / Caitlin: how are we doing this???

D.4.0.1.5 Extrapolating the max boiling point

D.4.0.2 No distillation data available

In some cases, it is desired to model oils where there is no distillation data available. In the case of refined products, it is impossible to know what the curve might be. In the case of crude oils, it is probably better to find a substitute oil will similar properties and origins, but the ADIOS DB does provide a way to estimate the distillation curves for crude oils.

When no distillation data are available, a linear relationship is assumed between mass fraction and boiling point. The initial boiling point (iBP) is based on a correlation found for a large subset of crude oils:

NOTE: can we reframe this in terms of density rather than API?

$$iBP = \frac{10}{9}(591.3728 - 3.6637 \cdot \mathrm{API}) - \frac{1015\,\mathrm{K}}{9} \tag{D.1}$$

and the terminal boiling point (tBP) is:

$$tBP_n = 1015\,\mathrm{K} \tag{D.2}$$

D.5 Everything below is old and probably out of date.

D.5.1 Vapor Pressure

Two of the key parameters of a PC are the vapor pressure and molecular weight – these drive the evaporation rates, one of the most significant weathering processes.

Primary PCs for crude oils can be constructed based on distillation data, if available, or API if no distillation data are available. We recommend that primary PCs have equal mass; there are n primary PCs.

Data condition 1:	distillation data available (crudes and refined products)
INPUTS:	cumulative mass distilled, true boiling points

If two or more vapor temperature distillation data points are available, then these data are used to determine the boiling points of the PCs. If distillation data are measured at a reduced pressure, they must be adjusted to atmospheric pressure. The data are stored as pairs of values: the temperature of the distillate (T_i) and the cumulative fraction of oil distilled (f_i).

The initial boiling point of the first PC (iBP_1 in K) is found by extrapolating the first two points:

$$iBP_1 = T_1 - f_1\left(\frac{T_2 - T_1}{f_2 - f_1}\right) \tag{D.3}$$

Define f_1 and f_2? DAH

and the terminal boiling point of the first **final** fraction (tBP_n in K) is:

$$tBP_n = 1015\,\mathrm{K} \tag{D.4}$$

The terminal boiling point of the i^{th} PC (tBP_i) is based on a linear fit to the distillation data that bracket or are closest to the cumulative mass fraction of the i^{th} PC (i/n). If tBP_i found this way is greater than tBP_n (the terminal boiling point of the last fraction) then tBP_i is set equal to tBP_n.

Data condition 2:	distillation data not available (crudes only)
INPUTS:	API or density

When no distillation data are available, assume a linear relationship between mass fraction and boiling point. The initial boiling point of the first PC (iBP_1) is based on a correlation found for a large subset of crude oils:

$$iBP_1 = \frac{10}{9}(591.3728 - 3.6637 \cdot \mathrm{API}) - \frac{1015\,\mathrm{K}}{9} \tag{D.5}$$

and the terminal boiling point of the first **final** fraction (tBP_n in K) is:

$$tBP_n = 1015\,\mathrm{K} \tag{D.6}$$

The fit is:

Cut Temperature at 10 % = 519.3728 – 3.6637· API

Cut_temperat
image here

D.5.2 Molecular Weight

Riazi1 image
here

Riazi2 image
here

D.5.3 Vapor Pressure

The vapor pressure (P_i) of each PC is based on Antoine's equation as discussed by Lyman et al.

TJB-bibtex citation here?

$$ln\frac{P_i}{P_0} = \frac{\Delta S_i(BP_i - C_{2,i})^2}{\Delta Z_b R_{cal} BP_i}\left[\frac{1}{BP_i - C_{2,i}} - \frac{1}{T_w - C_{2,i}}\right] \quad \text{(D.7)}$$

where:
$C_{2,i} = 0.19 \cdot BP_i - 18$,
$\Delta S_i = 8.75 + 1.987 \cdot ln(BP_i)$,
$\Delta Z_b = 0.97$,
$R_{cal} = 1.987$,
P_0 is the atmospheric pressure (101325 Pa),
T_w is the water temperature (in K), and
BP_i is the boiling point of component i (in K).

This seems to be all duplicated?

D.5.4 Construction of the Pseudocomponents

Data condition 1:	distillation data available (crudes and refined products)
INPUTS:	cumulative mass distilled, true boiling points

If two or more vapor temperature distillation data points are available, then these data are used to determine the properties of the pseudocomponents (PC). If distillation data are measured at a reduced pressure, they must be adjusted to atmospheric pressure. The data are stored as pairs of values: the temperature of the distillate (T_i) and the cumulative fraction of oil distilled (f_i).

The temperature of the initial fraction (T_0) is found by extrapolating the first two points:

$$T_0 = T_1 - f_1\left(\frac{T_2 - T_1}{f_2 - f_1}\right) \quad \text{(D.8)}$$

The temperature of the final fraction (T_{n+1}) is found by extrapolating the first and last distillation points:

$$T_{n+1} = T_n + (1 - f_n)\left(\frac{T_n - T_1}{f_n - f_1}\right) \quad \text{(D.9)}$$

Define f_n? DAH

or a linear least squares fit to the distillation data can be used.

Five PCs, each containing 20 % of the oil by mass, are constructed as follows:

- The boiling point of the first PC is the average of the lower and upper cut temperatures. The lower temperature is the temperature of the initial fraction; upper temperature is found through a linear interpolation of the points that bracket 20 % distilled.
- The boiling points of the second, third, and fourth PCs are found in a similar manner, where the lower temperature is the upper temperature of the previous PC.
- The boiling point of the fifth PC is the average of the upper temperature of the fourth PC and the temperature of the final fraction.

Data condition 2:	distillation data not available (crudes only)
INPUTS:	API or density

As a first-order approximation, T_0 and dT/df are based on the API from an empirical correlation using all the DOE oils in ADIOS:

GNOME? DAH

$$\frac{d}{dt} = 1356.7 - 247.36 \cdot ln(\text{API}) \quad \text{(D.10)}$$

and:

$$T_0 = 457.16 - 3.3447 \cdot \text{API} \quad \text{(D.11)}$$

Constants A, B, C, and D are based on average properties of oils in the ADIOS library.

These letters are not in equations above and should be in italics if kept here. DAH

GNOME? DAH

Five PCs, initially of equal mass, are constructed and their boiling points (BP_i) are determined using T_0 and dT/df:

$$BP_i = T_0 + \frac{dT}{df}\left(\frac{(i-\frac{1}{2})}{5}\right) \tag{D.12}$$

The cumulative fraction of each PC is found with:

$$f_i = \frac{i}{5} \tag{D.13}$$

Properties of PCs for data conditions 1 and 2:

Is something missing here? DAH

Each pseudocomponent is split into saturates (PCS) and aromatic components (PCA). The vapor pressure of each PC is based on Antoine's equation as discussed by Lyman et al.

TJB-bibtex citation here?

$$ln\frac{P_i}{P_0} = \frac{\Delta S_i(BP_i - C_{2,i})^2}{\Delta Z_b R_{cal} BP_i}\left[\frac{1}{BP_i - C_{2,i}} - \frac{1}{T_w - C_{2,i}}\right] \tag{D.14}$$

where:
$C_{2,i} = 0.19 \cdot BP_i - 18$,
$\Delta S_i = 8.75 + 1.987 \cdot ln(BP_i)$,
$\Delta Z_b = 0.97$,
$R_{cal} = 1.987$,
P_0 is the atmospheric pressure (101325 Pa),
T_w is the water temperature (in K), and
BP_i is the boiling point of component i (in K).

The effective molecular weight (MW_i) of each pseudocomponent i is based on correlations with alkanes and aromatic compounds:

$$MW_i^S = 0.04132 - (1.985 \cdot 10^{-4} \cdot BP_i) + (9.494 \cdot 10^{-7} \cdot (BP_i)^2) \tag{D.15}$$

and:

$$MW_i^4 = E - (F \cdot BP_i) + (G \cdot (BP_i)^2) \tag{D.16}$$

where the constants E, F, and G are based on a correlation to be determined.

D.5.5 SARA fractions

D.5.6 Solubility

D.6 Estimating Properties

Many times, particularly in response, a complete oil assay is not available for a given oil. In this case, ADIOS includes a number of algorithms that are used to estimate missing data, in order to generate a complete oil record for the GNOME model.

D.6.1 Distillation Cuts

D.6.1.1 Crude Products

ADIOS requires at least Density and Viscosity measurements for a crude oil. From these, the rest of the properties can be estimated. However, the uncertainty will be very high if all properties have been estimated.

If there are no distillation cut data (or fewer than three temperature ranges), the distillation is estimated using a linear fit, as a function of API. Note that this approach is only appropriate for crude oils – the distillation cuts of a refined product cannot be expected to follow a simple pattern.

Most common crude oils have a fairly linear distillation...

fill in more here

D.6.1.2 Refined Products

For refined products, at least three distillation cuts are required, because refined products with similar bulk physical properties can vary widely in other characteristics. It is advised that a generic product be used as a reference if specific data about the product in question are lacking.

Confirm that these edited sentences convey what was meant. DAH

D.6.2 Bulk SARA

The GNOME oil library accepts a much larger amount of oil data, if available. This data set is more inclusive than that used for the NOAA weathering model, ADIOS2 . For example, the GNOME oil library now includes information on the structural nature of the hydrocarbon groups that make up the oil. While not ideal for describing the weathering processes that affect spilled oil, GNOME has adopted the SARA categorization common in the industry. SARA represents...

reference the ADIOS2 paper

Incomplete sentence. DAH

Saturates – Nonpolar oil molecules without double bonds that including linear, branched, and cyclic saturated hydrocarbons

Aromatics – Hydrocarbon molecules that contain one or more benzene rings

Resins – Large hydrocarbon molecules with one to three sulfur, oxygen, or nitrogen atoms per molecule. Resins can dissolve in oil.

Asphaltenes – Very large hydrocarbon molecules with one to three sulfur, oxygen, or nitrogen atoms per molecule. Asphaltenes do not dissolve in oil.

D.7 Other Properties

There are number of other oil properties not strictly used by the GNOME oil-weathering model, but that are useful to know during a response. These include:

Flash point – At a certain temperature, vaporizing oil or oil product may reach a vapor concentration large enough to sustain combustion; this temperature is called the flash point.

Pour point – The temperature below which the oil loses its flow characteristics; this is the oil equivalent of freezing point.

Oil-water interfacial surface tension – The attractive force exerted upon the surface molecules of a liquid by the molecules beneath to make the liquid have the least interfacial surface area. A typical oil-water surface tension is 30 dynes/cm. Chemical dispersants reduce surface tension.

Adhesion strength – Beached oil will have different levels of stickiness on the shoreline. This dimensionless measure is called adhesion. Adhesion equals zero if the oil will not stick at all, and will equal one if it adheres completely.

Metal content – Although it consists mostly of hydrocarbons, oil may have certain trace metal components that will affect its behavior.

Emulsification constants – Emulsification (water-in-oil) of spilled oil increases its volume and viscosity. Not all oils emulsify and some will emulsify only after a certain degree of evaporative losses. At present, there does not appear to be a reliable formula to estimate emulsification (Lehr, 2018) constants for all oils but some emulsification properties that trigger emulsification have been measured. These include the fraction of the oil that must evaporate before emulsification begins and the maximum water fraction in any stable oil.

use my 2018 AMOP paper, Chris has a copy