Siku Sea Ice Discrete Element Method Model

Principal Investigator: Anton Kulchitsky[1]
Co-Investigators: Jennifer Hutchings[2], Jerome Johnson[1]
Graduate Student: Benjamin Lewis[2]

[1]Institute of Northern Engineering, University of Alaska Fairbanks
[2]College of Earth, Ocean, and Atmospheric Sciences, Oregon State University

FINAL REPORT
OCS Study BOEM 2017-043
June 2017

Coastal Marine Institute
College of Fisheries and Ocean Sciences
University of Alaska Fairbanks
P. O. Box 757220 Fairbanks, AK 99775-7220
Email: CMI@alaska.edu
Phone: 907.474.6782
Fax: 907.474.7204

This study was funded in part by the U.S. Department of the Interior, Bureau of Ocean Energy Management (BOEM) through Cooperative Agreement M13AC00001 between BOEM, Alaska Outer Continental Shelf Region, and the University of Alaska Fairbanks. This report, OCS Study BOEM 2017-043, can be accessed electronically at *http://www.boem.gov/Alaska-Scientific-Publications.*and is also available through the Coastal Marine Institute and select federal depository libraries.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the opinions or policies of the U.S. Government. Mention of trade names or commercial products does not constitute their endorsement by the U.S. Government.

TABLE OF CONTENTS

List of Figures

List of Tables

Appendix

ABSTRACT

Predicting sea ice motion in the Beaufort and Chukchi Seas is important for facilitating marine commercial operations and rescue activities when sea ice is present. This project developed an open-source computational framework and software library for discrete element modeling of sea ice mechanics (“Siku”). Siku uses a global (spherical) reference frame, employing regional ice sheet coverage and local (planar) reference frames, to calculate interactions between ice elements. It incorporates reanalysis surface winds from the National Center for Environmental Prediction/National Center for Atmospheric Research (NCEP/NCAR) as driving forces, but it can also use and combine other sources for wind and current data. Siku modeling allows ice to fail in tension and, when provided appropriate along-shore boundary conditions, can reproduce the locations where leads have been observed to form along the coast. The spacing of large-scale leads originating at the coast can be simulated by defining the along-shore boundary as the minimum observed landfast ice edge in any month. Siku output products are independent of the size of discrete element method (DEM) elements (resolution independence). Comparison of Siku simulations with satellite images indicated good correspondence of the locations and patterns of ice openings in the leads that Siku predicted with the satellite observations for the same wind field. Siku simulations indicated that initiation of arching leads occurs at promontories along the coastline and landfast ice edge and that the pattern of leads is a strong function of wind direction and confinement along the shore boundary. These arches occur in tensile failure, and their initiation is well represented with a linear elastic-viscous-plastic (EVP) representation of contacts between ice elements. Future improvement to Siku will include fine-tuning implementation of shear failure criteria to reproduce shear failure at Point Barrow and the Beaufort Sea shore lead.

INTRODUCTION

Offshore oil and gas exploration, production, and shipping activities in the Beaufort and Chukchi Seas can be significantly and adversely affected by sea ice. For example, in the event of an oil spill, the presence of mobile sea ice complicates the problem of tracking ice/oil trajectories and conducting cleanup operations. In such an incident, it would be useful to forecast the trajectory and dispersion of contaminated ice and to simulate the location of pressured ice (which can hinder transportation and jeopardize marine structures).

In brittle failure, sea ice develops cracks that open into leads and may close forming ridges. In winter, the motion of the ice pack is strongly modified by this ice interaction. The plastic failure dissipates energy from wind and ocean forcing. This is especially true along the coast where ice-coast interactions are known to modify the ice pack drift, which, compared to the interior ice pack, is further away from free drift (Thorndike and Colony 1982). The winter ice drift and dispersion is increased through shear and divergence at leads and shear zones. To simulate ice trajectories and dispersion, we needed to reproduce realistic lead distributions (opening, closing, and shearing rates), the associated horizontal ice deformation (the localization of divergence, convergence, and shear), ice velocity, and ice stress.

Modeling Background

The current state-of-the-art for coupled ocean-ice-atmosphere modeling makes use of a continuum model of sea ice kinematics originally developed by Hibler (1979). Coupled ice-ocean models have not performed well in reproducing observed sea ice strain rates (Kwok and Cunningham 2008). Representations of sea ice in regional, pan-Arctic, and global models that simulate the coupled ice-ocean system (e.g., Regional Arctic System Model, RASM; Wang and Shen 2010) or attempt ice forecasting (e.g., Arctic Cap Now-cast Forecast System) do not account for the brittle failure behavior of the ice pack on the spatial scales these models attempt to resolve (Coon et. al. 2007). The continuum elastic-viscous-plastic (EVP) models used to describe ice constitutive properties do not represent observed internal ice stresses and strain rates (opening and shearing), nor do they reproduce realistic patterns of localized shear zones (Kwok and Cunningham 2008). Therefore, these models cannot simulate the dispersion of sea ice correctly, which limits their utility in forecasting or hind-casting the trajectories of ice.

This project was motivated by the results of Wilchinsky et. al. (2010), who demonstrated the ability of a discrete element method (DEM) sea ice model, under idealized conditions, to simulate fracture patterns with intersection angles and spacing characteristics similar to those observed in Arctic pack ice. Although, to date, no regional model of sea ice has reproduced realistic deformation patterns (Kwok and Cunningham 2008), the DEM approach has been successful in simulating the density of fractures expected in the Beaufort Sea (see Figure 1). The DEM approach directly accounts for discontinuities in the ice pack where failure can occur and stresses concentrate to form cracks. Continuum approaches using an isotropic rheology (such as the CICE ice model, which uses the EVP model) require artificial seeding of stress discontinuities in order to simulate cracks (Hutchings et. al. 2005). As the DEM approach specifies the failure stress of weaknesses (defined as joints or contacts between grains or unit cell floes), control of fracture characteristics is more physically-based in a DEM model. Other approaches developed include anisotropic models that assume embedded failures in the continuum (Wilchinsky and Feltham 2006, Taylor and Feltham 2004) and the elastic de-cohesion model (Schreyer et. al. 2006). There are three particular benefits in using DEM as a base for development of Siku simulations:

(1) High resolution can be achieved around structures or spill sites for direct calculation in the model, avoiding the need for nesting.

(2) Brittle failure of the pack ice can be based on similar physical representations used in continuum models. This allows improvements in continuum models as high-resolution DEM models can be tuned to reproduce observed deformation.

(3) 3-D models would allow development of a landfast component in the regional model through direct simulation of ridge-sediment interaction.

The known deficiencies of EVP models were overcome by using a DEM approach, which has been shown to reproduce the discontinuous dynamics of sea ice that are responsible for shear patterns (Hopkins et. al. 2004, Wilchinsky et. al. 2010). These discontinuous dynamics are not well described using continuum models. Siku model development focused on winter ice, when the ice pack is confined by the Canadian and Alaskan coasts and experiences repeating patterns in its deformation (Eicken et. al. 2006, Mahoney et. al. 2012). Such repeating patterns indicate that the pack ice has preferred modes of failure, which, to date, have not been reproduced in a sea ice model. Since repeating patterns of deformation exist and are observable, it should be possible to constrain a mechanical DEM model to accurately simulate sea ice kinematics (including dispersion) in the Beaufort and Chukchi Seas. A DEM model has the ability to capture modes of discontinuous deformation. For example, Wilchinsky et. al. (2010) demonstrated that lead patterns with realistic intersection angles could be achieved with an idealized DEM model. In their work, they used idealized simulations of the formation of conjugate fault pairs under confining stress, which is one of the types of shear failure observed in the ice pack interior. Siku extends the capabilities of the DEM approach to model ice-coast interaction and the resulting arching and shear zones.

Siku Model

The DEM approach to sea ice modeling was originally developed by Hopkins et. al. (2004) and improved with shear stress local force criteria by Wilchinsky et. al. (2010). This report describes “Siku” (after the Iñupiaq word for sea ice), a new discrete element method (DEM) model of polar sea ice intended to improve the forecasting of ice drift and dispersion.

Siku is the first sea ice DEM model that accounts for the spherical geometry of the Earth and allows simulation ranging from global and basin scale to meter scale. In addition to improved accuracy and compatibility with the inputs from other global models, Siku can solve a much wider range of problems than earlier models including analysis of ice-infrastructure interaction, ice movement on a variety of scales, and ice dynamics of paleo-Earth or other ice covered planets.

Siku was developed to have improved sea ice interaction mechanics compared to previous DEM models. For example, it incorporates a landfast ice boundary condition that ensures modeling of stress build up and ice pack failure (such as promontories along the Alaskan coast where this is observed). It can also reproduce the tensile failure in arches that is observed for many of the repeating lead patterns along the coasts of the Beaufort and Chukchi Seas.

The Siku model is not a computational kernel. Rather, it is a complex system that includes preprocessing, data interpolation, model management, visualization, coordinate system transformations, and other features required for successful simulations and analysis. It includes automatic import of initial ice coverage, coastlines, landfast ice, winds, and currents. Siku is an open source, GPL-licensed model with embedded Python scripting language for setting up simulation scenarios.

METHODS

Basic Concepts

The Siku sea ice DEM model simulates the media using small rigid bodies, the *ice elements*, which we described as convex polygons. The size of the ice elements determines the resolution of the model in a particular area. The adjacent polygons can be fully or partially *coalesced* along a frozen joint. The ice elements cover the ice area determined by *initial conditions* for the sea ice. In simulation, the ice elements experience external *contact forces* from their neighboring ice elements and external *driving forces* from air and water motion, gravity, Coriolis, and any other mass forces. The *boundary conditions* are determined by the *coastline* and, when applicable, by *landfast ice* (ice anchored near the coastline).

The ice element positions and the states of coalesced joints change as initial conditions evolve under external forces that act upon the elements according to the basic laws of physics. Leads and ridges form, and the stress condition in the ice changes. The accuracy of the final pattern of leads and ridges is determined by the accuracy of the external forcing representation, the contact physics model, resolution, and boundary conditions. The model can be validated by comparing the DEM simulated state of the ice, particularly locations of leads and shear zones, with the satellite observations of ice pack deformation.

Coordinates and Reference Frames

Siku uses two types of reference points or "frames" to describe values such as positions or vectors:

- Global frame – connected to the Earth and rotating with the Earth
- Local frames – connected to a particular ice element

Four coordinate systems are used to describe elements in global frame:

- Geographical coordinates (latitude φ and longitude λ) – used only for import or export elements and never used internally. We provide transformation from, and to, geographical coordinates.
- Spherical coordinates ($\phi = \lambda, \theta = \pi/2 - \varphi$).
- 3D extrinsic Cartesian coordinates (*x*', *y*', *z*') with the center of coordinates coinciding with the center of the Earth, *O*; *z*' directed along Earth's axis of rotation towards the north; *x*' directed to the 0° meridian, and a *y*' direction chosen perpendicular to *x*' and *z*' to form a right coordinate system. These coordinates satisfy the constraint $x'^2 + y'^2 + z'^2 = R^2$.
- 3D normalized (dimensionless) extrinsic Cartesian coordinates (*x*, *y*, *z*) defined identically to (*x*', *y*', *z*') but projected onto a unit sphere such that $x^2 + y^2 + z^2 = 1$.

These are the main global coordinates used internally in the computational kernel for all computations.

We use the global frame to transform global coordinates into a local frame for subsequent computations. Global frame is also used for output.

The transformations between extrinsic coordinates and geographic/spherical coordinates are performed using the following formulas:

$$\theta = \frac{\pi}{2} - \varphi, \qquad \phi = \lambda, \qquad x = \sin\theta\cos\phi, \qquad y = \sin\theta\sin\phi, \qquad z = \cos\theta \tag{1}$$

Local coordinates (X, Y, Z) are centered in the center of mass of the ice element. The axes are directed such that they would coincide with global-frame normalized extrinsic coordinates (x, y, z) after an application of the rotation described by the quaternion q linked to the ice element.

Global extrinsic coordinates for any point p are restored from the local coordinates P by applying the rotation matrix restored from the following quaternion:

$$p = R(q)\ \cdot P \tag{2}$$

The rotation matrix, *R*, can be recovered from the quaternion as described in Eberly (2002) and in many other places.

Position and Kinematics of Ice Elements

In Siku, an ice element is a flat, two-dimensional rigid body in the form of a convex polygon moving on a spherical surface. The rigid body motion on a sphere can be described as a motion about a point in three-dimensional (3D) space – the center of the sphere. The position of the rigid body rotating around a fixed point can be described by a single quaternion *q* or, equivalently, by a rotation matrix *R* (see theoretical background in Arribas 2006). The formulae for the quaternion description are very compact and convenient for computer simulations. The quaternion equations have no singularities compared to more traditional spherical coordinates, and the computations usually do not involve time-consuming trigonometric operations. The quaternion representation of ice element position is our novel contribution to DEM modeling of sea ice.

The coordinate system used for polygon position and dynamics description is shown in Figure 1. The quaternion *q* describes a rotation (around some axis *e* that passes through the center of the sphere *O)* that would move the polygon and match local coordinate lines *x', y', z'* with global-frame coordinate lines *x, y, z*.

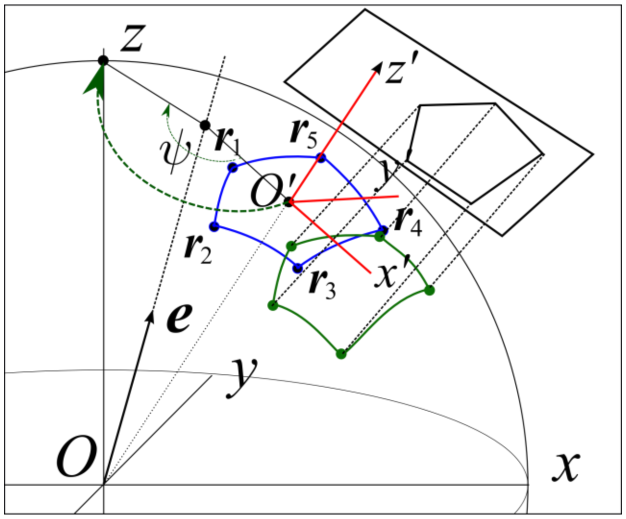

Figure 1. Coordinates and projection.

The position update is described by a standard differential equation for rigid body rotation as

$$\frac{dq}{dt} = \frac{1}{2} q \circ \Omega, \tag{3}$$

where Ω is the 3D angular velocity in the local ice element frame. The efficient integration scheme used for these placement equations (as first described in Kulchitsky and Hutchings 2014) provides a significant improvement over integration formulas derived by Walton and Braun (1993) and trigonometric approaches similar to those described in Harada (2008).

Let $q^n = q(t)$, $q^{n+1} = q(t + \Delta t)$, then the 2nd order scheme for Eq. (1) takes the form

$$q^{n+1} = q^n + \frac{\Delta t}{4}(q^n + q^{n+1}) \circ \Omega, \tag{4}$$

where vector Ω is taken from the previous time step. Collecting all values calculated at (n+1) step on the left, and taking into account that quaternion product is not commutative, we have

$$q^{n+1} \circ \left(1 - \frac{\Delta t}{4}\Omega\right) = q^n \circ \left(1 + \frac{\Delta t}{4}\Omega\right). \tag{5}$$

Using relation $q \circ q^* = |q|^2$, we can multiply both sides of Eq. (5) by $\left(1 + \frac{\Delta t}{4}\Omega\right)$ and, after trivial mathematical operations, we get

$$q^{n+1} = q^n \circ \frac{\left(1 - \frac{\Omega^2 \Delta t}{16} + \frac{\Delta t}{2}\Omega\right)}{1 + \frac{\Omega^2 \Delta t^2}{16}} = q^n \circ p. \tag{6}$$

It is easy to see that $|p| = 1$; therefore, the formula does not change the absolute value of the unit quaternion.

Dynamics of Ice Elements

The dynamics equations define a 3D "pendulum" with angular velocity Ω in the ice element reference frame. This vector can be calculated exactly using 3D dynamics of a rigid body with constraints. While the rigid ice element on a sphere is curved, a simple, yet precise approach is to assume the ice element is flat, since its size is much smaller than the radius of the Earth. The reference frame for an ice element is a standard Cartesian coordinate system, so the velocity and rotation rate of a flat (2D) rigid body can easily be calculated by the following dynamics equations:

$$\frac{dV_x}{dt} = \frac{F_x}{m}, \qquad \frac{dV_y}{dt} = \frac{F_y}{m}, \qquad \frac{d\Omega_z}{dt} = \frac{N}{I}, \tag{7}$$

where (F_x, F_y) are external forces that include all driving forces, integral forces from interactions with other ice-elements, and Earth Coriolis force; m is the total mass of the ice element; I is a moment of inertia around the Z axis; and N is the total torque calculated in the center of mass C, which includes a torque from interaction with other ice elements and a possible torque that can be calculated from the driving forces.

V_x is directed normal to Z, thereby, "rotating" the body around Y axis with a rate of V_x/R. This gives us exactly Ω_y. Using the same consideration for the x-axis, $\Omega_x = -V_y/R$. Thus, the dynamics equations are described by evolution of the 3D angular velocity as follows:

$$\frac{d\Omega}{dt} = H = \left(-\frac{F_y}{mR}, \frac{F_x}{mR}, \frac{N}{I}\right), \tag{8}$$

where R is the radius of the Earth. The integration scheme for this equation is a straightforward linear integration scheme.

Ice Generation

Ice element generation defines the resolution of the model. The size of ice elements is varied depending on their location and the need to have different resolutions in different areas. For sea ice DEM models, it is conventional to use Voronoi tessellation of a semi-random point set to build the set of elements. The Voronoi diagram provides the set of convex polygons suitable for sea ice modeling.

Generation of the initial ice elements describing the sea ice sheet is done in the following stages:

1. Generate points on a sphere with different distances between the points depending on the desired resolution in particular areas as shown in Figure 2 (B). The points are generated randomly with a blue-noise condition that constrains the distance between any two points such that it cannot be less than some minimum value.

2. Generate Voronoi tessellation of the point set or “seed” from stage 1. The Voronoi area for a particular seed is a region consisting of all points on a sphere closer to that seed than to any other seed in the predefined set. These areas form a set of convex polygons as shown in Figure 2 (C).

3. Mark as boundary polygons those polygons that intersect the coastline provided by the Global Self-consistent, Hierarchical, High-resolution Geography (GSHHG) database (described by Wessel and Smith 1996) or landfast ice (detailed below). All polygons that are inside the land or are landfast are removed from the system. The final set of polygons (representing sea ice) used for our case studies testing Siku is shown in Figure 2 (D). Boundary polygons from the coastline and landfast ice do not move. Rather, they interact with sea ice using different contact mechanics to prevent sea ice elements from crossing the shoreline.

The described functionality in the Siku model is implemented in Python 3 as a library that is called from a “scenario” simulation script. The STRIPACK library by Renka (1997) was adopted for Voronoi tessellation on a sphere.

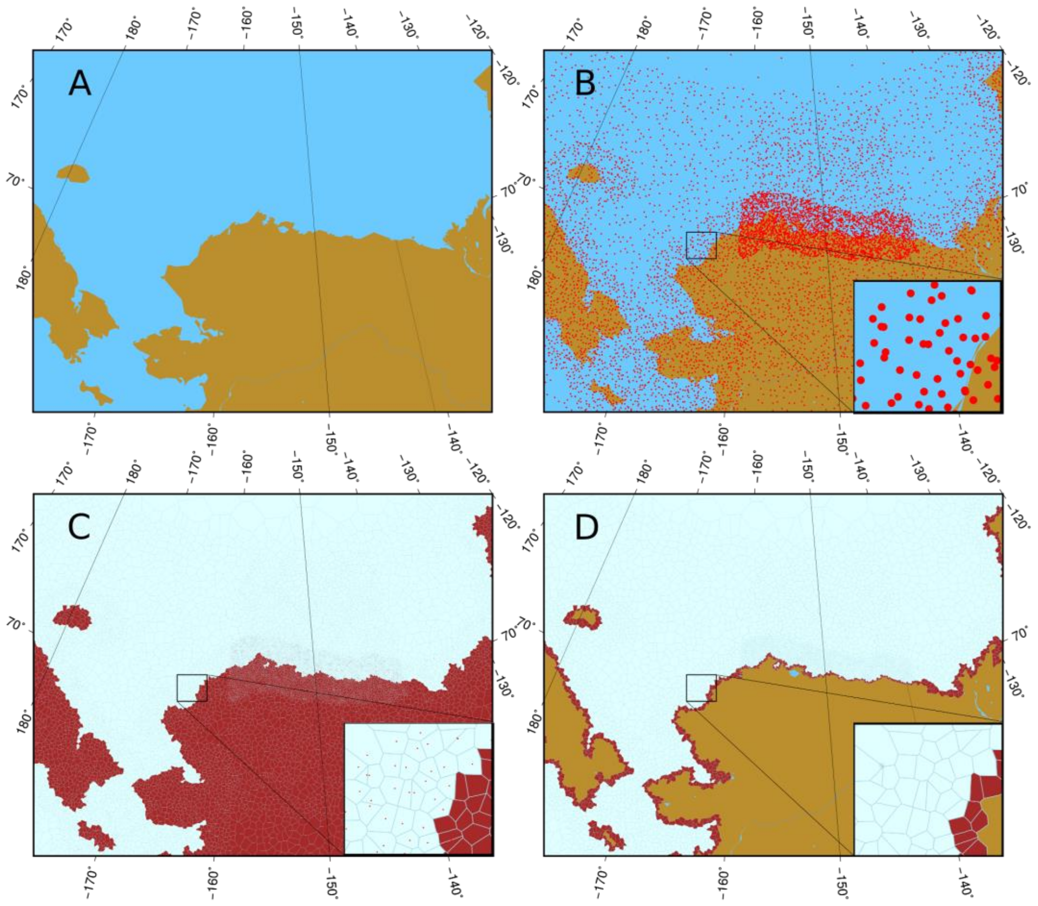

Figure 2. Ice elements creation stages: (A) initial contours, (B) blue noise ice element kernels generation with resolution depending on the area, (C) Voronoi tessellation on a sphere based on the kernels, and (D) ice and boundary elements ready to be imported to the computational kernel.

Contact Detection

The key feature that distinguishes a discrete system from a continuum system is that any pair of elements in contact in the discrete system may separate and a separated pair of particles may connect. Thus, discrete element models need to keep track of all contacts and their evolution. Searching for contacting particles occupies a significant portion of simulation time. Checking the distance between every pair of discrete objects in the model is a straightforward but naive approach to this problem. It would require comparison of every pair and poses a $O(N^2)$ time complexity problem, where N is the number of elements in the system. It becomes a computationally prohibitive algorithm for applications with many elements.

To avoid excessive computations for finding neighboring elements, a broad-phase contact detection stage is used to avoid comparisons between objects located far apart from each other. At this stage, every element in the model is surrounded by a bounding circle. Since intersection of bounding circles is a necessary condition for intersection of the elements, it can be used to filter out the majority of impossible contacts. The model utilizes a sweep and prune contact detection algorithm (Ericson 2004). The model makes projections of the bounding circles on x direction in the global frame Cartesian coordinates, forming a set of intersecting segments. The one dimensional array of these segments is then sorted by the segment starting point coordinate and the sorted list traversed to find all projected segment intersections. Projected intersections are checked to determine if they are actual intersections before adding to the contact list.

Contact Physics

Contact physics is an essential part of the model. There are two principle modes of contacts, one for coalesced elements and one for independent elements. The coalesced elements have a single frozen joint and are "glued" to each other along it. Depending on the stresses or deformation it experiences, the frozen joint may accumulate damage with time and break. If the frozen joint breaks, the elements become independent and may still collide and interact with each other (Figure 3). The Siku code implements two physical models for coalesced elements, a distributed spring model and a modification of the Hopkins-Frankenstein (HF, Hopkins et. al. 2004) and Wilchinsky-Feltham-Hopkins (WFH, Wilchinsky et. al. 2010) models.

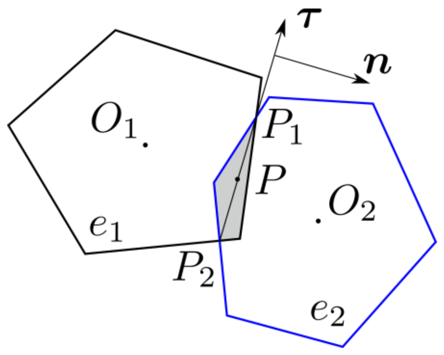

Figure 3. Polygon intersection and interaction in the case of independent polygons.

Coalesced elements: distributed spring model

The forces in the deformed ice region are being calculated as pairwise contact forces between elements that are connected with joints. The basic assumption of the distributed springs model is that the ice behaves like a homogeneous viscous-elastic substance for small deformations, but it breaks if the deformation exceeds some threshold. The interaction force is calculated for each pair of elements that either overlap or were previously marked as a joint by a “freezing” function. This freezing function is commonly called immediately after the polygons are loaded for identifying joints that are coalesced. Joints may also be manually added in the Python script.

The interaction force is calculated as a composition of two different mechanisms. The first mechanism (“Collision” in code) is a basic repulsion of overlapping elements. It works for any pair of overlapping elements, whether jointed or independent. As with any other force, collision has elastic and viscous components. The elastic component is a simple linear spring where the stiffness of homogeneous spring depends on its size and elastic modulus, and the force is stiffness multiplied by deformation.

$$F = KAn, \qquad K = \frac{h_1E_1h_2E_2}{h_1E_1r_2 + h_2E_2r_1}, \tag{9}$$

where K is the effective stiffness of material, A is the area of the overlap, and n is the direction of force. Effective stiffness is a general property of each pair of elements. Area is the product of spring's width and deformation. The direction of the force is either the perpendicular vector to the interaction segment P_1P_2 in a regular case (Figure 4a) or a vector (Figure 4b) in more complicated interactions.

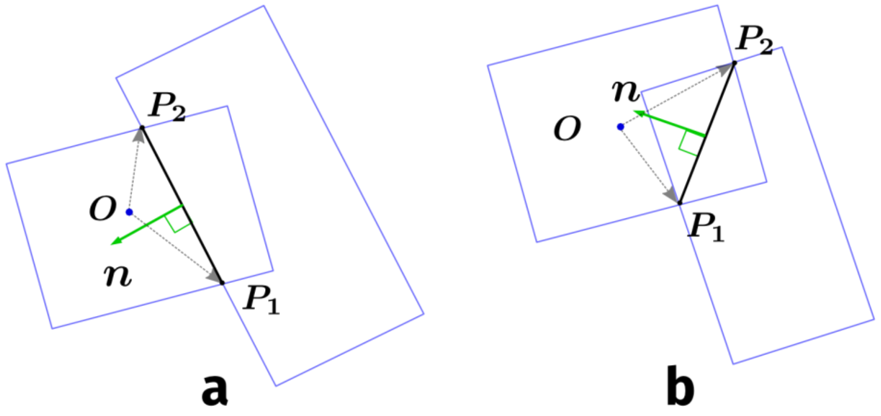

Figure 4. Normal vector computation for intersecting polygons for two basic cases: (a) side to side intersection and (b) corner to corner intersection.

$$s = (P_{1y} - P_{2y}, P_{2x} - P_{1x}), \qquad n = s/|s| \tag{10}$$

Viscous component is a simple liquid viscosity of two surfaces moving relatively to each other:

$$F_i = -A\eta(v_{O1} - v_{O2}), \tag{11}$$

where A is the overlap area, η is a global viscosity parameter, and $(v_{O1} - v_{O2})$ is the vector of relative velocity of overlapping zones. The torque caused by viscosity due to relative angular velocity of elements is ignored.

The second mechanism is the traction of jointed edges. At the beginning of the modeling process, the elements are "frozen" together as described in section "Freezing joints." For each frozen pair, both elements have four points (two points for each element in its local coordinates) that mark the ends of the interaction front along the joint. This joint represents the edge between two (potentially separate) floes, regardless of actual polygons shapes. At the same time, the joint determines the behavior of the "distributed spring." In an equilibrium position, the joints of both elements have the same global position (the spring is not deformed). When the elements have some relative displacement, the segments become shifted and a force appears. The mechanical properties of the spring are determined in the same way as the Collision mechanism.

The width of the spring (w) is the length of the interacting joint rather than some multiplier of the overlap area because there may be no overlap. The effective stiffness is calculated in the same way as in Eq. (9) and viscosity is the same global parameter as before.

The force and torque calculation mechanism is more complicated than in the "Collision" mechanism. When the elements are not in an equilibrium position, the interaction segments do not match. The displacements of the corresponding points, which are the ends of joints, form two displacement vectors. These vectors carry all of the information needed to calculate the force and the torque generated by the distributed spring. The total influence of the spring may be decomposed into two effects as shown in Figure 5.

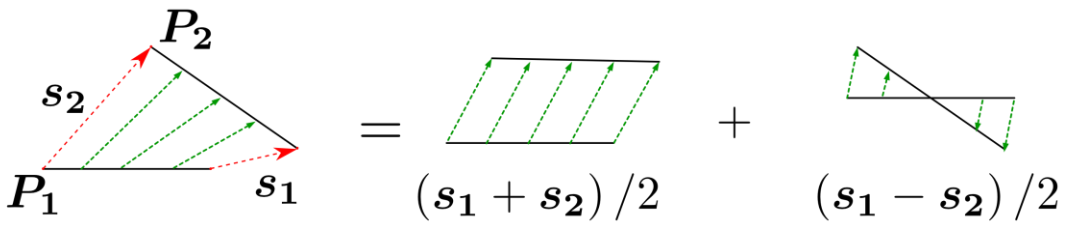

Figure 5. Distributed spring force model decomposition.

The arithmetic mean of the displacement vectors gives the raw traction force applied to a single point – the middle of the interaction segment. The cross-product of this force and the vector to the corresponding element's mass center gives a "traction torque." The force acts in any direction and works for both compression and tangential displacement. The second effect is a stress-torque.

The cross-product of the directed interaction segment and arithmetical difference of displacement vectors with the proper coefficient gives the torque produced by the distributed spring. The force, F_n, and the torque, N, can be written as

$$F_n = \frac{Kw(s_1 + s_2)}{12}\,, \qquad N = \frac{K\big((s_1 - s_2) \times w\big)}{12}, \tag{12}$$

Coalesced elements: modification of HF and WFH stress models

The first DEM constitutive model for regional pack ice was developed by Hopkins (1996) and used in subsequent work (Hopkins et. al. 2004, Hopkins and Thorndike 2006). Hopkins later model allows contact stresses between elements to increase as the displacement at the contacts increase. When the contact stress reaches the failure criteria for compression and tension failure, the contact between elements starts to fail (i.e., the elastic modulus is decreased) with further displacement at the contact. The stress at the contact decreases (weakening of ice strength) linearly until it reaches zero to create a local crack. The crack is allowed to grow to 95% of the contact length and then fails completely to form a lead (in tension) or a ridge (compression). Wilchinsky et. al. (2010) updated this model to allow shear deformation. This is achieved by applying a velocity-dependent Mohr-Coulomb shear-strength failure criterion along the contact between ice elements to determine when rupture of the contact occurs.

where s_1 and s_2 are the displacement vectors at end points of the joints as shown in Figure 5, and w is the directed interaction segment or "width" vector defined as a mean value of both interacting edge vectors.

In addition to the elastic spring, we add viscous damping force similar to the Eq. (11). As the viscous force depends on the relative velocity of the elementary area, and the velocity depends on the distance from the rotation axis, then the integration by the area of interaction gives a formula similar to the moment of inertia as follows:

$$F_i = -A\eta(v_{O1} - v_{O2})\,, \qquad N_i = \eta A^2(\Omega_1 - \Omega_2) \tag{13}$$

Ice elements are rigid such that deformation at the contacts is determined by the normal and tangential displacement along the length of the contact as shown in Figure 6. The contacts between the DEM ice elements support both normal (compressive and tensile) and shear (tangential) stresses and fail when the stresses exceed specified failure conditions.

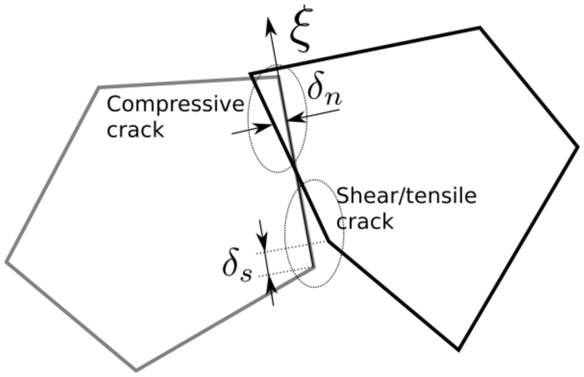

Figure 6. Coalesced ice element interaction with compressive and tensile areas along the frozen joint and with shear deformation.

Note that, without the Wilchinsky et. al. 2010 update, models only considered tensile and compressive rupture, resulting in unrealistic rectilinear lead patterns in cases where the mode of failure should be predominantly shear. The Mohr-Coulomb criteria allows the ratio of shear strength to compressive strength of the ice to be varied through a friction angle parameter, ϕ, which can be used to control intersection angles between active leads. Wilchinsky et. al. 2010 assume that joints between ice elements maintain some tensile strength and have a large compressive strength. We will consider a Mohr-Coulomb failure criterion, following Wilchinsky et. al. 2010 or its modifications as shown in Figure 7. Unlike the Hopkins group, neither Siku nor Wilchinsky et. al. 2010 include stress weakening in their model; rather, they assume that a crack forms once stresses reach the failure stress.

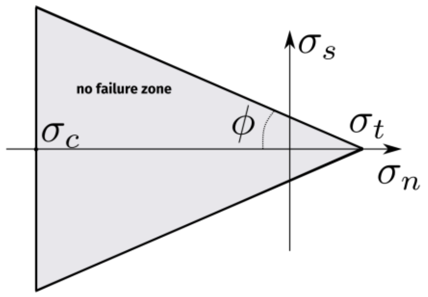

Figure 7. Mohr-Coulomb failure criteria.

The normal, $\sigma_{n,}$ and shear stresses, σ_s, are calculated as

$$\sigma_n = \frac{hE}{L}\delta_n, \qquad \sigma_s = \frac{hG}{L}\delta_s, \tag{14}$$

where h is the mean ice thickness, L is the length of contact, and the normal and tangential displacement along the contact are δ_n and δ_s, respectively, as shown in Figure 6. Failure occurs when normal stress exceeds the compressive/tensile failure criteria or the shear failure criteria (see Table 1) are met.

Where only existing connections contribute to the contact forces, Siku computes the local forces on the contacts as

$$F_n = L\int_0^1 \sigma_n\, \theta(\sigma_n, \sigma_s)\, d\zeta\, n\,, \qquad F_s = \int_0^1 \sigma_s\, \theta(\sigma_n, \sigma_s)\, d\zeta\, \tau, \tag{15}$$

where $\theta(\sigma_n, \sigma_s) = 1$ if (σ_n, σ_s) are inside the yield curve and 0 if the failure criteria is locally met, and ζ is a parameter along the joint between two polygons equal to 0 at one vertex and equal to 1 at another $(0 < \zeta < 1)$.

For each pair of polygons, the torques at a contact have the form

$$N_j = L\int_0^1 r_j \times (\sigma_n n + \sigma_s \tau)\, \theta(\sigma_n, \sigma_s)\, d\zeta\,, \qquad j = 1{,}2 \tag{16}$$

where r_j is the vector from the center of mass of a polygon to the point at the edge that depends on ζ. The edges are split in segments and integrals are approximated by sums. Each sub-polygon contributes to the total forces if the stress is inside the yield curve.

Table 1. Material properties required in calculating contact stress and failure criteria. Sources of excepted values or ranges are provided.

Young's modulus	E = 1 GPa 0.25 < E < 4 GPa	Hopkins and Thorndike (2006)
*S*hear modulus	$G = \frac{E}{2(1+\nu)}$	Elastic model
Poisson's ratio	$\nu = 0.3$	Wilchinsky et. al. (2010)
Compression failure criteria	$\sigma_c = 1285\, h^{\frac{2}{3}}_{\min}$ kPa/m	Kovacs and Sodhi (1980)
Tensile failure criteria (cohesion)	$\sigma_t = 0.1\sigma_c$	Wilchinsky et. al. (2010)
Shear failure criteria	$\tau_f = \tan\phi\, \lvert\sigma_t - \sigma_n\rvert$	Coulomb (1776)
Friction angle	$13^o < \phi < 18^o$ $(0 < \phi < 53^o)$	Erlingsson (1991), Wang (2007) discuss the larger range, but most observational studies cluster around Erlingsson's results.

Independent elements

When elements are independent, they can either collide as two rigid bodies or slide one above another. The Hopkins et. al. 2004 model does not restrict the ice elements in intersecting. We added an option to allow the ice elements to collide and resist at the contact point. This physics can be turned off or freely adjusted during the simulations.

For elements that are not coalesced we assume interactions are viscous elastic with a normal force linearly dependent on the square root of the overlap area between the polygons similar to Matuttis et.al (2000). The damping part of the force depends on local velocities over the contact area. The independent element interaction is essentially the same as repulsive mechanics in coalesced distributed spring model described above.

Frozen Joints Description (Initialization)

The continual ice force model requires a list of ice elements coalesced along the common frozen joint. These joints contain the information about the original relative position of the interacting elements, physical and geometrical parameters of the ice in the area at the contact, and some additional information required for the force computations.

Different physical models require different sets of parameters so joints need to be generated with respect to the selected physical model. The user can also determine that only some ice elements remain connected with each other while the rest become separate floes, despite their initial position and properties. This can be done, for example, for modeling the marginal ice zones. In order to fulfill that requirement, Siku implements a mechanism that allows the different options.

The user may place the list of required contacts into the Python script in variable 'siku.settings.links' (the format is a list of pairs of tuples of two elements indexes, implemented as a list of Python). When such a list of contacts exists at the beginning of calculations, Siku will load it to be the initial list of contacts. If the user does not provide the predefined contacts list, or the list length equals to zero, then Siku uses its own method to form the list of pairs of elements potentially close enough to be connected along a common side.

The user selects a joint model and Siku generates a list of actual joints with sufficient information for the subsequent force calculations. There are two different joint models that satisfy existing force models.

The first function is a single spring. To apply this, polygons are temporally enlarged by a 'tolerance' (1% for a predefined links list, 10% for an auto-defined links list) before searching for each polygon’s intersection area. If such an area is not zero, its center becomes a point of the spring connection. Joint width and the size of joined elements define the stiffness of the spring.

The second function uses a distributed spring and our modification of HF and WFH methods. It enlarges the polygons and finds intersections in the same way. The only difference in this function is that, after the intersection is found, Siku defines the vertices of the original polygons that match the ends of the common edges. As these vertices may be slightly displaced relative to each other, Siku calculates mean positions for each pair of the vertices, respectively, and the resulting length of the common edge. These points are saved as the ends of distributed spring or interaction segment as required for the selected physical model.

After all frozen joints are identified and a joint model assigned to each, the list of these joints becomes the list of contacts that is updated and used each time step.

Driving Forces

In the summer, when arctic ice concentration is low, long-term mean ice motion results from the balance of four forces on the ice (Arctic Ice Dynamics Joint Experiment AIDJEX; Hunkins 1975; McPhee 1979). In free drift, the significant forces on the ice are wind stress, ocean stress, the Coriolis force, and acceleration down the sea surface tilt. Inertial terms, found to be an order of magnitude smaller than these on daily or longer time scales (Thorndike 1986) are, however, important on shorter time scales, and the accumulative impact of the inertial motion of the ocean and ice may be significant in the sea ice mass balance (Heil and Hibler 2002). The residual term in the measured force balance is found to be comparable to the Coriolis force and an order of magnitude smaller than the wind stress (McPhee 1979). This residual is attributed to interactions between ice floes. In areas of ice convergence, resistance to compression becomes important and ice can no longer be considered to be in free drift. For climatological studies, small time scale forcing may be ignored. Contributions to the force balance from tidal motion, pressure gradients, and inertial terms are considered to be small when averaged over time scales greater than a day (McPhee 1979). For short-term ice drift estimation, and in regions with strong tides, the tidal and inertial terms may be important and included in our future work. For this model development, we focused on the largest driving forces: wind, ocean drag, and ice interaction.

In Siku, the external forces acting on the ice element and used in Eq. (17) are given as

$$F = S(\tau_a + \tau_w - mfk \times U) + F_i, \tag{18}$$

where S is the surface area of the ice element, τ_a and τ_w are the wind stress (“air”) and ocean stress (“water”) on the ice, respectively, f is the Coriolis parameter, k is a unit vector normal to the surface, and F_i is the net force from interaction with other ice elements.
The wind stress τ_a and ocean stress τ_w are calculated by the air velocity U_a at a known height in the atmosphere and by the ocean current velocity U_w, usually below the surface Ekman layer. They can be calculated as

$$\tau_a = \rho_a C_a |U_a| (U_a \cos\theta_a + k \times U_a \cos\theta_a) \text{ and} \tag{19}$$

$$\tau_w = \rho_w C_w |U_w - U| ((U_w - U)\cos\theta_w + k \times (U_w - U)\sin\theta_w), \tag{20}$$

where ρ_a is the density of air, ρ_w is the density of water, C_a and C_w are the drag coefficients for the air and water interfaces, respectively, and θ_a and θ_w are the turning angles across the boundary layers. The magnitudes of the drag parameters and turning angle are dependent on the height (depth) that winds (currents) are assigned. We used the following drag coefficients and turning angles for our model testing:

$$C_w = 0.0045, \qquad \theta_w = 0, \qquad C_a = 0.0016, \qquad \theta_a = 0. \tag{21}$$

Winds Import and Interpolation

Currently, Siku uses the National Centers for Environmental Prediction and National Center for Atmospheric Research (NCEP/NCAR) reanalysis data for winds (Kalnay et. al. 1996). Alternatively, winds can be provided from a text file. NetCDF4 files provided by NOAA need to be manually downloaded from the *www.esrl.noaa.gov* website for the select time period of computations. The model verifies data dates and extracts only data that match the selected computation period. The data is represented as latitude and longitude physical components of the wind in a regular grid.

Ice elements are not generally located at the wind grid point and are higher density than wind data. Therefore, the wind data needs to be interpolated at the position of ice elements. As the grid is regular, we use the simplest method of bilinear interpolation at each ice element. The following simple procedure is currently implemented:

1. Find a grid cell that includes the current position.
2. Find the projections of the current element position on the borders of the cell.
3. Use linear interpolation to receive wind data in the projection points.
4. Find the weighted linear average using the projection points.

For more accurate interpolation, which takes into account differential properties of the wind field such as divergence and curl, a gradient-preserving, yet accurate procedure needs to be used. For example, Fuselier and Wright (2009) described the method to reconstruct the vector field into divergence-free and curl-free parts and accurately preserve differential properties of interpolated vector fields. However, for the ice sheet stress representation, a simple bilinear interpolation suffices in most cases, especially when taking into account uncertainties in the input data.

The architecture and inner representation of the grids were created for NCEP/NCAR reanalysis data format, although any actual source format may be processed with some modifications.

The data is imported in the following order:

1. The data is loaded and preprocessed by the Python scenario. The scenario uses the utility module 'nmc.py' to load the raw gridded data from the specified .nc files: East and North components of the velocity field. That module is designed for the NCEP/NCAR reanalysis surface winds.
2. The 'wnd.py' module initiates actual surface velocity grid from two 'nmc' instances. The data is being processed and adapted to match the kernel input format.
3. The Siku variable 'wind' is assigned to the wind grid prepared in step 2. The kernel will attempt to read the vector field from that variable.
4. The flag 'settings.wind_source_type' is assigned to the type of imported data source. For example, “siku.settings.wind_source_type = siku.GRID_SOURCES['NMC'] ” is used for the standard NCEP/NCAR reanalysis field.

5. The kernel requests the scenario if it is time to update the grid before each simulation time step (in 'pretimestep' callback function). The kernel tries to load the grid from the 'wind' variable and use it in calculations if the wind was updated.

If no grid is given, the kernel loads an empty grid with all the values set to 0.

The grid itself may either cover the entire globe or a particular area. Excessive data is ignored, and the areas with no data are set to 0 if the grid covers a region that is smaller than the globe. In the future, when the data are lacking, the forces in the regions will be adjusted to avoid artificial gradients.

The Python utility libraries allow some reasonable testing values to be placed in the grid by loading the grid from a file and using the 'make_test_field' method.

The ocean currents are processed in the same way as the winds but with different flag and variable names:

- siku.settings.water_source_type = siku.GRID_SOURCES['NONE']
- siku.water = wnd.NMCSurfaceVField(…)

Coastal and Landfast Ice Boundary Conditions

Ice element interaction is controlled by the force balance on elements, which includes interaction with the coastal boundary. The winter ice pack is confined by landfast ice and the coastline. Therefore, to properly model ice-coast interaction and the coastal shear zone, we need to represent the landfast ice edge as a fixed boundary in our model. This ice edge can be identified as the extent of stationary ice contingent with the coast; however, this edge varies depending on the time scales considered. A more robust definition of the landfast ice edge is the ice that is held in place by grounded ridges, or the strength of the ice pack bridging between islands or shoals with grounded ridges. We identify a landfast ice mask for the Beaufort Sea as the minimum landfast ice extent observed in any month (Mahoney et. al. 2007). In the absence of landfast ice, Siku uses the GSHHG database from NOAA for coastline representation. We confined our experiments to the Chukchi and Beaufort Seas because landfast ice products are not currently available for the entire Arctic. We will be able to expand the scope of Siku simulations to the pan-Arctic if landfast ice extent datasets become available for other Arctic regions.

Visualization and Post-processing

Siku provides maps of the ice elements. All visualization is performed in the Generic Mapping Tools (Wessel et. al. 2013). This is a powerful command-line controlled package that uses shapefiles to process, visualize, and export a wide range of different external data.

The command line and complex syntax of the GMT interface make it difficult to use or embed it. Siku provides utility modules to automate the visualization. The 'gmt_Plotter' Python module contains a class that processes scenario settings and generates a file with simplified command strings for further usage. At each step of simulation, various callback functions export required data from the kernel into the files. As soon as the files and the command strings are prepared, ‘gmt_Plotter’ calls 'gmt_Drawer' module. The 'gmt_Drawer' module reads previously prepared command strings, extends them to full GMT commands, and calls the GMT to execute those commands one by one.

Most of the settings required to draw the picture are set in '…conFigurepy' files.

The Siku automates the following output features:

- .nc wind grids: external or dumped from the kernel
- various surface grids: ocean currents, directions, polygons (regardless of their actual source)
- interpolated winds: interpolation may be performed internally by gmt_Plotter module
- polygons: dumped from the core with optional coloring (for example stress coloring);
- shapefiles: as the GMT work with shapefiles, drawing of several layers in one picture is straightforward.

The format of the output files is encapsulated postscript vector format (eps) by default. However, that can be changed with minor adjustments of the code. These eps files may be converted to gif, pdf or any other format by calling an external utility like “convert” from ImageMagick package.

Program Structure and Workflow

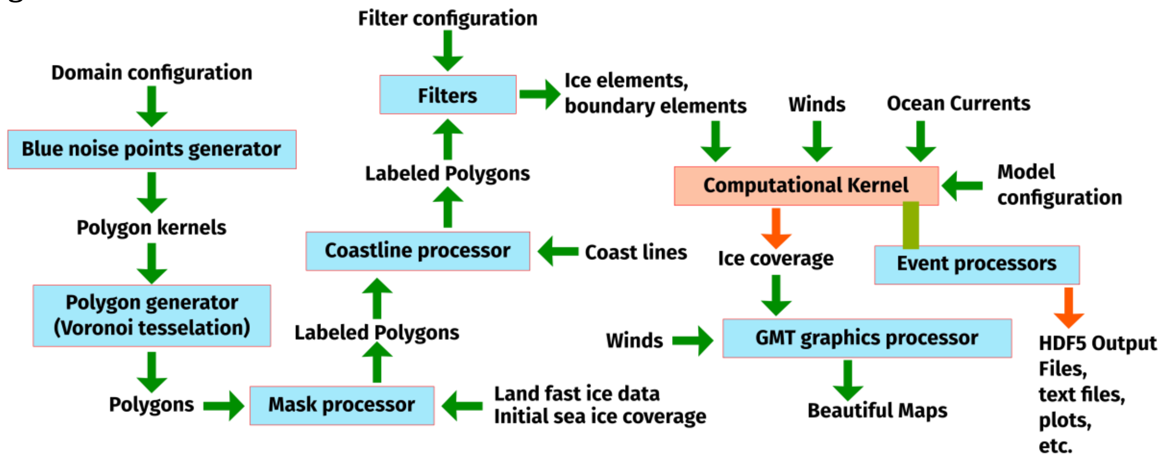

Figure 8. Data flow and Siku DEM model structure.

Python Interface

The model incorporates the Python 3 interpreter to set up numerical experiments. The Python interface allows a user to define the domain, time range, and sources and to monitor the runs. The Siku kernel is running the simulations and interacts with user’s script to exchange the data, create plots, and diagnose problems if they occur. A sample Python 3 Siku script can be found in the Appendix.

Simulation Preparation and Run

Any particular case simulation consists of a pre-processing stage, computational stage, and post-processing stage. The following workflow describes a typical simulation in Siku:

1. Coastline boundary preparation. A special Siku Python module (border_gen) is created to accumulate user data and provide boundary creation. The module creates a list of contours by different inputs. The inputs can be shape files from GSHHG, user text files, or a list of latitude and longitude pairs. After all contour points are added, the minimal angular distance between them can also be used to filter them.
2. Adding vertices. Randomly generated high-density points close to desired high-resolution borders need to be added. Points are generated using the same boarder generation module.
3. Borders generation. As soon as all contours and vertices are ready, the borders need to be generated. The border points are labeled to form boundary ice elements in the future.
4. Sea ice points generation. The globe needs to be covered with another set of randomly generated points. The points are generated using Siku tools used for border vertices generation or by using user's file with latitude and longitude pairs of points generated elsewhere. Different parts of the globe can be covered with different resolution and different patches of points using a built-in Siku module.
5. Filtering and merging. When the border and covering files are ready, they have to be cleared to avoid conflicts between close points in patches overlapping areas. This step is required for the Voronoi diagram generator to work properly on the whole point set.
6. Points file finalizing. When all borders are merged and all point coverings are filtered, all points are saved in one file for Voronoi point processing and ice element generation.
7. Voronoi tessellation. The merged file is passed to a Fortran Voronoi tessellation program that produces two files containing the data of generated polygons.
8. Polygons upload and marking. The polygon files are loaded into Siku Python scenario file by Siku poly_voronoi module. Polygons can be marked; for example, to form landfast ice or marginal ice zones or to filter by some mask such as whether generated in open water or on the land.
9. Scenario file creation and debugging. Siku Python scenario file also contains the time period, time step, wind or currents source location, material properties, and call back functions that define output and other parameters. The Python script can be debugged first by passing to a Python 3 interpreter.
10. Model run. Siku kernel runs with the created scenario.
11. Post-processing. The HDF5 files with output can be processed at this stage using built-in Siku Python 3 scripts that use GMT for visualization.

Model Verification

We first verified that the physical model Siku uses worked as we expected. We checked that the physical laws were satisfied and equations were solved correctly. We started with free drift of an ice element and independence of the results on the model resolution.

Free drift

The free drift test is the initial simulation that verifies that ice elements move as expected according to the force balance imposed on them. This is in the absence of any contact physics. We created a set of tests with a separate ice floe moving along a meridian, equator, or any other big circle on the globe, with or without rotation. Accurate checks showed the correctness of the dynamic derivation and implementation.

Forcing and ice pack failure tests

We performed a series of ice pack failure experiments as demonstrated in Figure 9. The ice pack was created using rectangular ice elements and then a wind pattern was applied. The deformation and failure process was observed and compared to that qualitatively expected due to the stresses in the ice pack. These verification tests only checked for the correctness of the model implementation and are not validation of the ice mechanics.

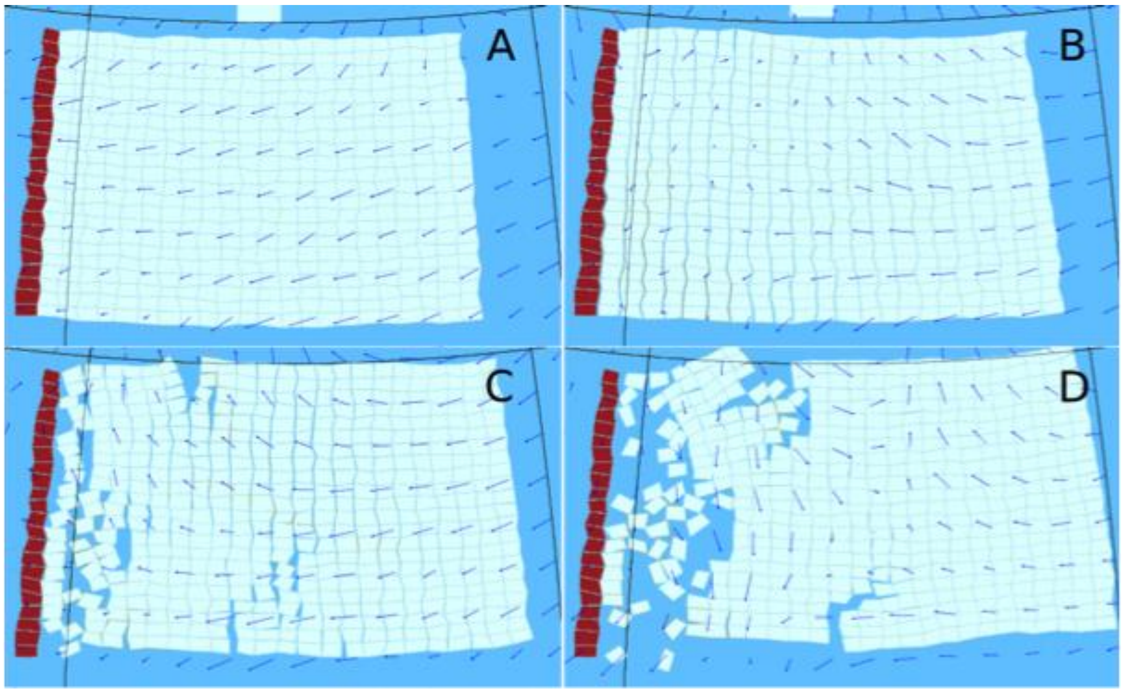

Figure 9. Wind tearing test of a rectangular ice pack with elastic bonds. The boundary shoreline is shown on the left with brown polygons. Panels (A-B) demonstrate the pack with all bonds unbroken. Panels (C-D) show the evolution of the ice pack with some ice elements disconnected when the local damage exceeds a threshold.

Resolution independence

We verified that our mathematical and computational model is resolution-independent up to the model resolution accuracy. It is important to ensure that ice representation and discretization in DEM does not change the physical outcome except to improve the accuracy of results. A series of tests with a 1500 km by 450 km ice pack built from different sized ice floes were performed. An ice pack was stretched or compressed, or a sheared deformation was applied to an edge of the pack as shown in Figure 10. Stress maps were compared to ensure that results did not depend on the representation of the ice pack.

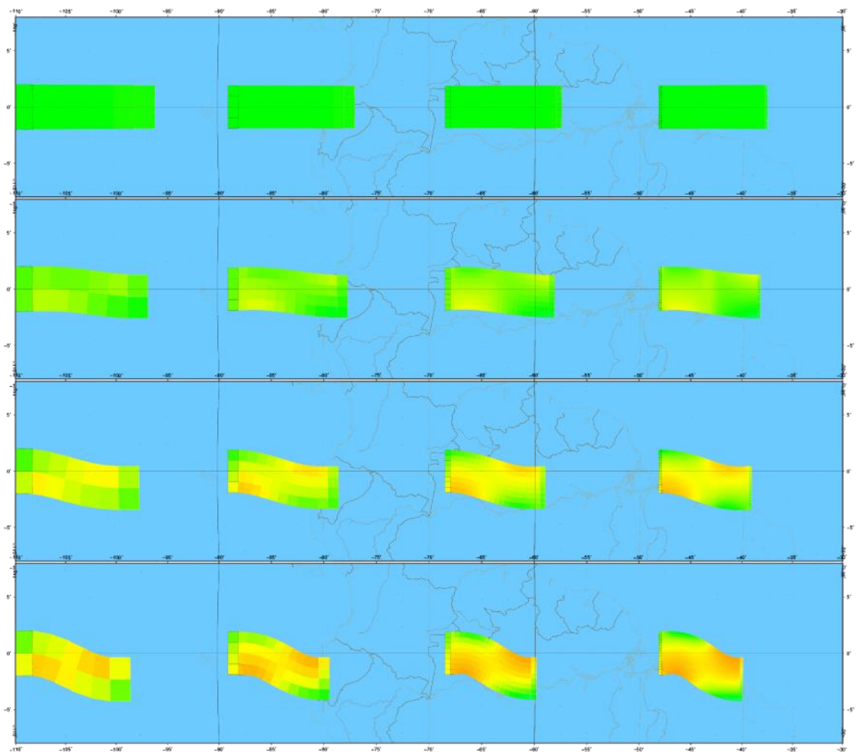

Figure 10. Resolution independence test with shearing deformation applied to a large 1500 km by 450 km ice pack with distributed spring model. The tests verified that the model does not change the integral behavior depending on the size of its ice elements.

RESULTS

Lead Formation

Lead patterns in the Beaufort and Chukchi Seas are observed to have repeating patterns (Mahoney et. al. 2012). The coastal-originating leads form in the same locations and take a couple of forms. They can be formed in tensile failure as arches, originating from promontories along the landfast and coast edges, with shear stress or cohesion defining the arch shape (Sohdi 1977). They can also follow shear zones that also originate at these promontories, but at more acute angles to the wind. We have extended the classification of lead patterns offered by Mahoney et. al. (2012) (Lewis in prep.). Several of the lead patterns were used to test the ability of Siku to simulate failure of the ice pack into leads under different failure mechanisms and confining stresses. As of all these patterns occur in the consolidated ice pack between late December and May, we confined our validation simulations to winter.

Our first test was to constrain the time period over which lead formation occurs. The simulation area was initialized with a 100% consolidated ice cover and resulted in the leads forming while the ice is breaking under the stresses produced by wind pushing the ice along the coast. Figure 11 shows how leads initiate along the shoreline and propagate into the ice pack at locations where land protrudes. The full pattern forms within about 15-18 hours of the model time and then evolves depending on the wind pattern. This is within the time period of satellite images that bound the actual lead formation observed on the day of this case study.

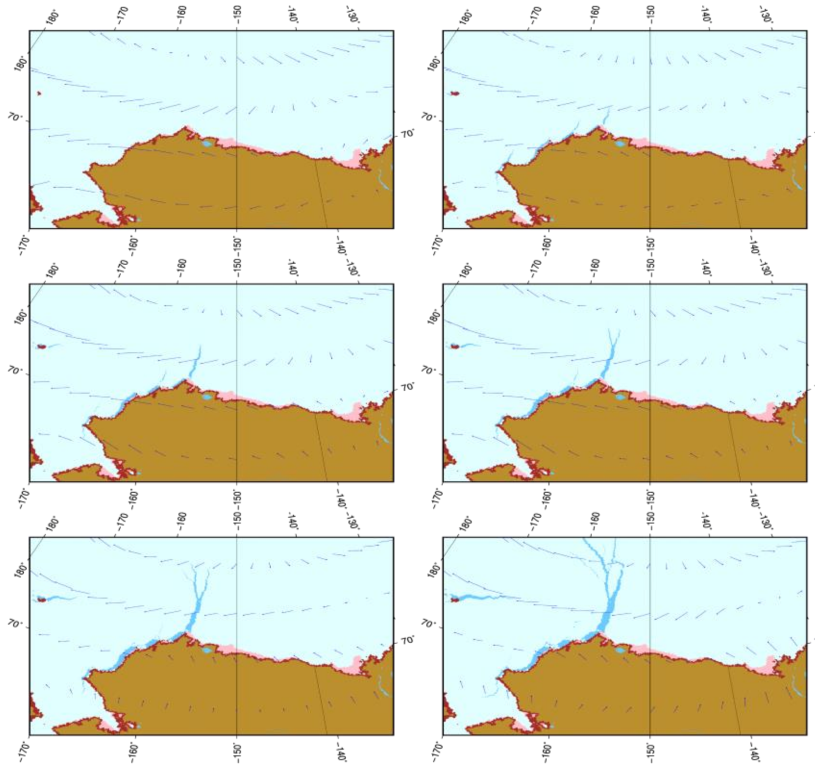

Figure 11. The evolution of the sea ice from unbroken ice to a fully formed lead pattern within 24 hours. This case study is for February 9, 2000.

The protruding features of the shoreline, or landfast ice, produce stress concentrations that initialize ice failure. The crack grows into the ice pack and it fails under tension, forming a lead. As the leads form, the stress is localized around the opening crack tip and they continue to grow and open as winds blow ice away from the arch.

Case Studies

Leads with widths greater than 250 m (Mahoney et. al. 2012) are identifiable in cloud-free Advanced Very High Resolution Radiometer (AVHRR) thermal images, which have been collected on the polar pathfinder satellites since 1979. Cloud-free AVHRR satellite images from 1994-2010 were collated (one per day) for the case study region of the Beaufort Sea. With consecutive cloud free images, it is possible to identify lead formation times to within the bracketing time of the images. For the case study model initial validation, we chose a set of daily AVHRR images that showed significant and distinct lead patterns. The following series of cases were used to validate and tune the current model.

1. February 17, 1994. This particular lead formation pattern was studied by Lewis et. al. (2016) and simulated in the Siku DEM model by Kulchitsky et. al. (2016). The case study period considered by Lewis et. al. (2016) is from 00:00 UTC on February 16, 1994 until 18:00 UTC on February 21 1994, during which time a high-pressure system forms and moves across the Beaufort Sea from the NW towards the SE. As the anticyclone transits the region, associated leads, are observed. Cloud free satellite images during this time-frame allow for opening leads to be identified within the constraint of satellite overpass timing. February 17, 1994 is a convenient day to compare the results of the model and observations.

2. February 8-9, 2000. Similar to the case 1, but with an almost perfectly parabolic shape of the lead. Both case study 1 and 2 are leads that terminate in the center of a high-pressure weather system (anticyclone) that is zonally aligned with Point Barrow. This weather is associated with easterly winds along the Beaufort coast, and the promontory of Point Barrow is where stress can accumulate in the ice pack. Hence a crack can initiate here and an arch forms as the ice breaks in tensile failure. The radius of the arch is related to the cohesion of the ice pack (Sodhi 1977), which can be adjusted through the stiffness of springs in the spring based contact physics models or by the cohesion in the Wilchinsky et. al. (2010) model.

3. March 6-7, 2000. This is a clear example of an arch forming off Point Barrow, which is related to an anticyclone centered above Point Barrow.

4. April 6, 2001. This is a tangent lead, formed at more an easterly angle from Point Barrow than the previous arch examples. The anticyclone associated with this lead is in a location that has also been found to be associated with arches forming off Point Barrow, Richards Island and Hershel Island as it traverses west to east. Further research is required to identify why on some occasions a tangent lead forms and on other occasions arches.

5. March 17, 1996. This is one of the inverse leads that instead of forming from a high pressure system with winds blowing clockwise, results from a low pressure with counter clockwise winds. However, it still forms a typical arch shape, but with the opposite concavity of the anticyclone associated . This is a good test for the model as it is representing fundamentally the same physics as other cases but in reverse and potentially with stronger winds due to the steeper pressure gradient in cyclones.

6. April 2, 2003. This pattern is very common, and occurs when the ice pack is moving clockwise around the entire Beaufort Sea. It is related to an anticyclone that is centered above the eastern Beaufort Sea. The anticyclone associated with this lead opening formed on March 30, and persisted until April 2nd after which it moved over land.

All of the following simulations were performed with the distributed spring model, with spring stiffness K calculated by ice Young's modulus of 0.8 GPa and global viscosity parameter $\eta = 0.001$, see Eqs. (9)-(11).

February 17, 1994

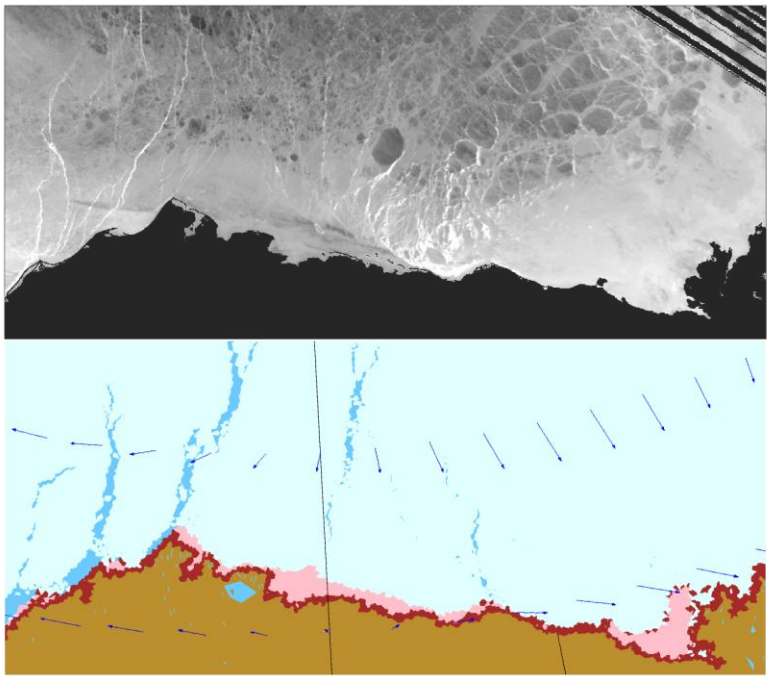

Figure 12. Observations and simulation comparison for February 17, 1994 lead formation.

February 8-9, 2000

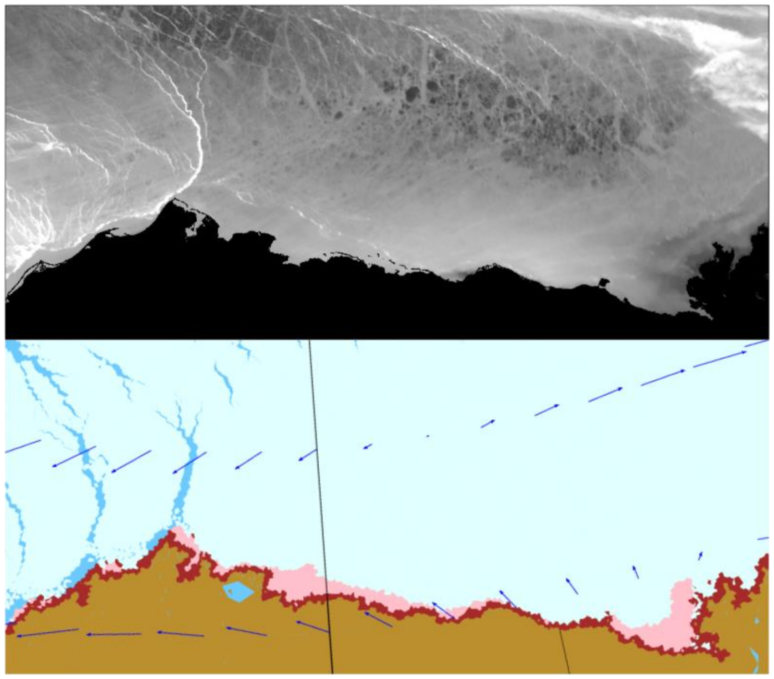

Figure 13. Observations and simulation comparison for February 8, 2000 lead formation.

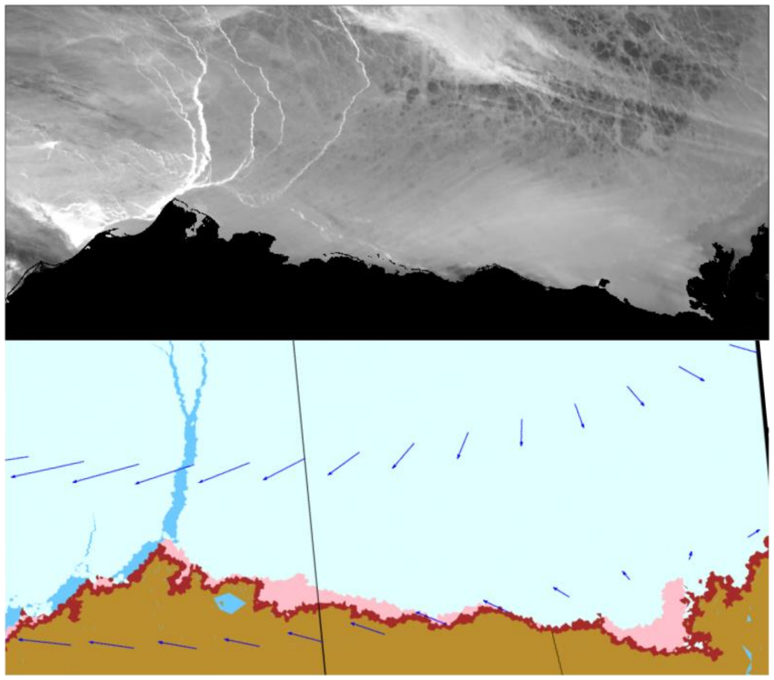

Figure 14. Observations and simulation comparison for February 9, 2000 lead formation.

March 6-7, 2000

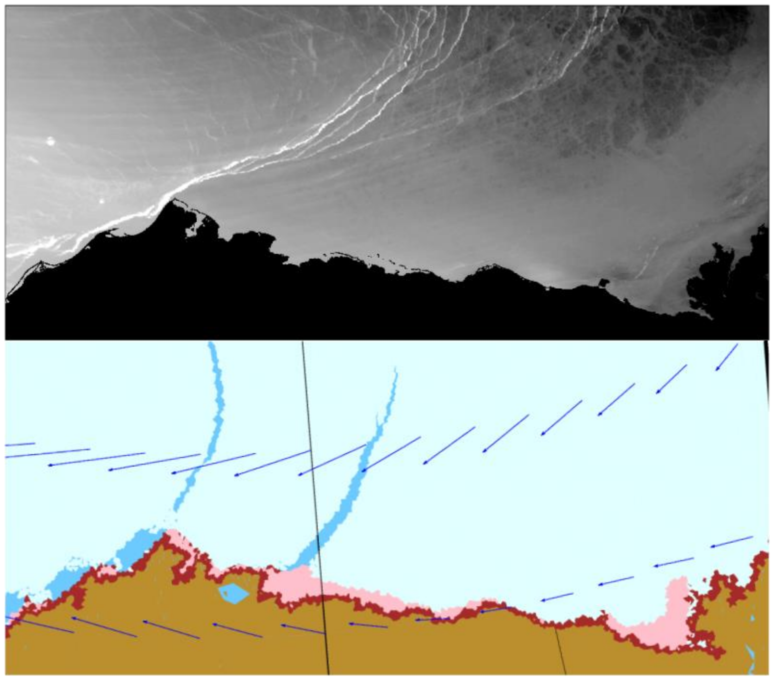

Figure 15. Observations and simulation comparison for March 6, 2000 lead formation.

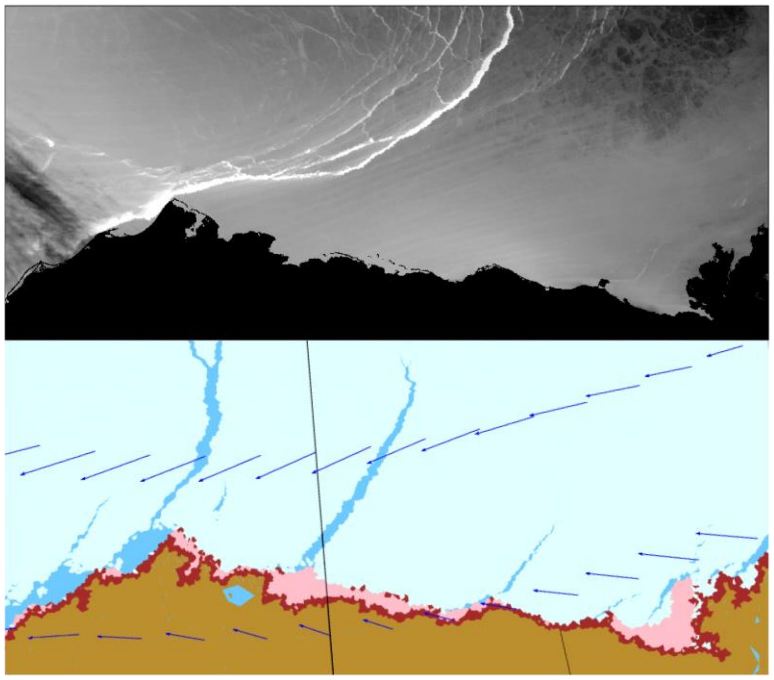

Figure 16. Observations and simulation comparison for March 7, 2000 lead formation.

April 6, 2001

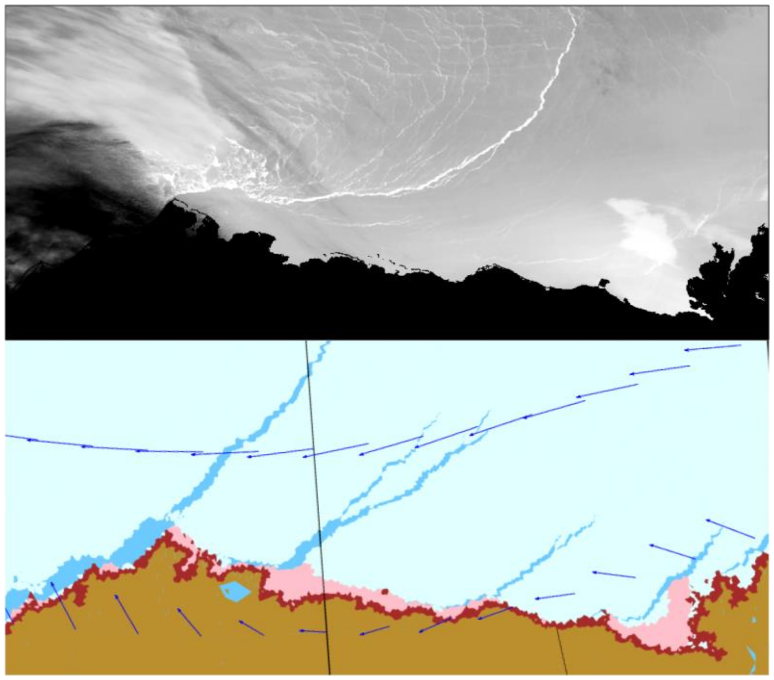

Figure 17. Observations and simulation comparison for April 6, 2001 lead formation.

March 17, 1996

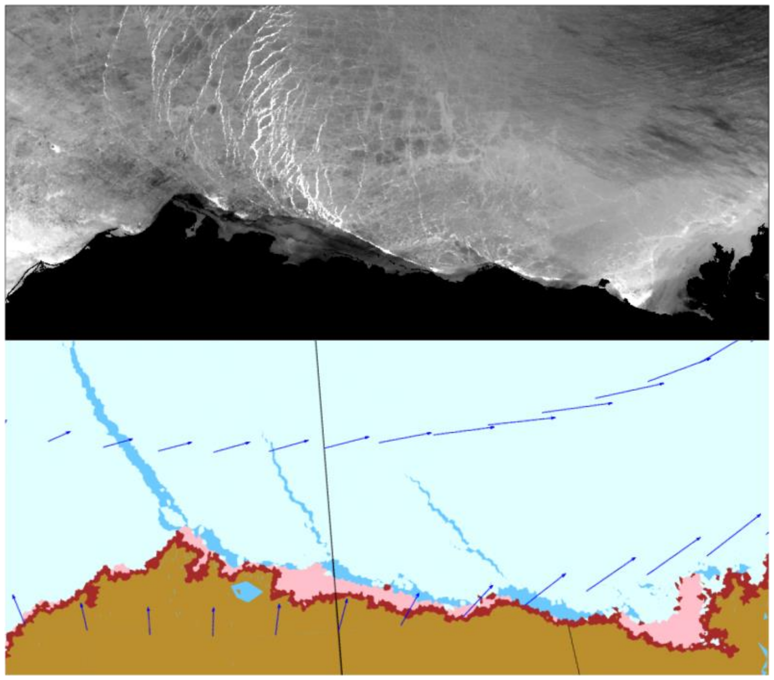

Figure 18. Observations and simulation comparison for March 17, 1996 lead formation.

April 2, 2003

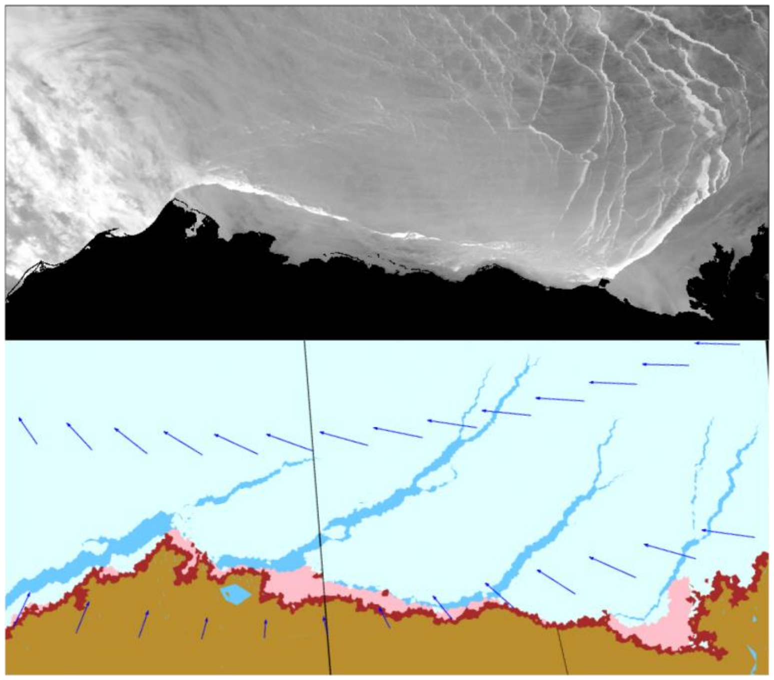
Figure 19. Observations and simulation comparison for April 2, 2003 lead formation.

Initial results from the first set of realistic simulations performed by Siku demonstrate that the model is capable of reproducing the nucleation of cracks along the Beaufort Sea landfast ice edge. These simulations are an improvement on earlier simulations without landfast ice, where the coastal boundary included all the spatial heterogeneity of the actual coastline. We found leads nucleated at many locations along the coast that are not associated with known lead patterns.

Improvements to the contact physics will improve these simulations. With the distributed spring model, arches are not simulated with sufficient curvature as the model does not allow for cohesion. The extension fractures that formed off point Barrow fractures in the March 6-7, 2001, and April 6, 2001, case studies cannot be simulated. In these cases, second arches form at a promontory to the east of Point Barrow. Both of these failings suggest that cohesion needs to be represented in the contact physics. In the case of arch curvature, increased cohesion of the ice pack will increase the curvature. In the case of the extension (or tangent leads following Mahoney et. al. (2012) naming convention) to the Barrow fracture pattern, cohesion of the ice pack with the landfast ice, which can essentially be viewed as extensions of the landfast ice, would create a boundary along which these leads form. For this reason, we have not pursued further sensitivity study and validation of the spring model. Instead, we have focused on developing a contact physics model that includes cohesion and shear failure. Future work will include running the case studies with the new contact physics.

DISCUSSION

The Siku model is a new framework for DEM sea ice modeling with a powerful ability to add extensions. It can be used for any area of the Earth (at any time of its history) or any other spherical planet. Although the current physical model is immature, as shown in validation section, improving the contact models and input quality will significantly improve the predictive power of the model.

Siku is implemented in C++ but has a built-in Python 3 interpreter and a set of Python modules that allow users to prepare the case studies without recompiling or changing the kernel code. Python interface provides the features of a powerful, all-purpose programming language to extend the model, do pre- and post-processing, and write interfaces with other software. The Python scripting language makes the model accessible to specialists who are not very familiar with programming and allows the community to write extensions.

CONCLUSION

Including landfast ice is important for accurate simulation of the deformation of the ice pack located within 500 miles of the coast. It is not possible to model the correct location of coastal-originating leads without defining the pinning points against which the ice pack is constrained and stress concentrates. We found that the stable landfast ice edge needs to be represented for accurate simulation of lead spacing. This can be defined as the minimum observed landfast ice edge in any month (following Mahoney et. al. 2012).

We found that a simple elastic-brittle model is suitable for modeling the propagation and extent of these leads; however, it does not adequately simulate the curvature of arches. We will address this in future efforts by including cohesion in the model. Shear leads cannot be simulated with these contact physics and, in the future, these cases can be used to constrain a version of Siku that includes Mohr-Coulomb contact physics.

Our simulations are the first time that these coastal leads have been simulated in a numerical model. Further improvements to the contact physics will result in a model that is better constrained for tracking ice drift, deformation, and dispersion along the Alaskan coast in winter.

ACKNOWLEDGEMENTS

This study was sponsored by the University of Alaska Coastal Marine Institute. Study collaboration and funding were provided by the U.S. Department of the Interior, Bureau of Ocean Energy Management, Environmental Studies Program under Agreement Number M13AC00001. Cost-share funding was provided by the University of Alaska Fairbanks Institute of Northern Engineering and Oregon State University.

STUDY PRODUCTS

This project supported graduate student, Ben Lewis, whose Master's thesis explores the role of repeating lead patterns in the Beaufort Sea in the mean Beaufort Gyre motion. Mr. Lewis identified the case studies for testing Siku.

The web site http://siku.ceoas.oregonstate.edu describes the model and contains the links to the open source program code. The source code repository is hosted at Bitbucket: https://bitbucket.org/coupi/siku with wiki page and bug tracker.

Kulchitsky, A. V., Hutchings, J. K., & Johnson, J. B. (2014, December). Siku: A Sea Ice Discrete Element Method Model on a Spherical Earth. In: *AGU Fall Meeting Abstracts* (Vol. 1, p. 0358).

Kulchitsky, A. V., Hutchings, J. K., Johnson, J. B., & Velikhovskiy, G. (2016) Siku Discrete Element Method Sea Ice Model. In: *Proceedings of the 23d IAHR International Symposium on Ice, Ann Arbor, Michigan, USA.*

Kulchitsky, A. V., Hutchings, J. K., Johnson, J. B., & Velikhovskiy, G. (2016, December). Siku DEM Simulations of Beaufort Sea ice Fracture Pattern. In: *AGU Fall Meeting Abstracts*.

REFERENCES

Arribas, M., Elipe, A., & Palacios, M. (2006). Quaternions and the rotation of a rigid body. Celestial Mechanics and Dynamical Astronomy, 96(3/4):239–251.

Coon, M., Kwok, R., Levy, G., Pruis, M., Schreyer, H., & Sulsky, D. (2007). Arctic Ice Dynamics Joint Experiment (AIDJEX) assumptions revisited and found inadequate. Journal of Geophysical Research: Oceans, 112(C11).

Coulomb, C. A. (1776). An attempt to apply the rules of maxima and minima to several problems of stability related to architecture. Mémoires de l'Académie Royale des Sciences, 7:343–382.

Eberly, D. (2002). Rotation representations and performance issues. Magic Software, Inc., Chapel Hill, NC.

Eicken, H., Shapiro, L. H., Gaylord, A. G., Mahoney, A., & Cotter, P. W. (2006). Mapping and characterization of recurring spring leads and landfast ice in the Beaufort and Chukchi seas. US Department of Interior, Minerals Management Service (MMS), Alaska Outer Continental Shelf Region, Anchorage.

Ericson, C. (2004). Real-time collision detection. CRC Press.

Erlingsson, B. (1991). The propagation of characteristics in sea-ice deformation fields. Annals of Glaciology, 15(1):73–80.

Fuselier, E. J., & Wright, G. B. (2009). Stability and error estimates for vector field interpolation and decomposition on the sphere with RBFs. SIAM Journal on Numerical Analysis 47(5):3213–3239.

Harada, T. (2008). Real-time rigid body simulation on GPUs. In Nguyen, H., Ed., GPU Gems 3, pp. 611–631. Addison-Wesley, NVIDIA.

Heil, P., & Hibler III, W. D. (2002). Modeling the high-frequency component of Arctic sea ice drift and deformation. Journal of Physical Oceanography, 32(11):3039-3057.

Hibler III, W. D. (1979). A dynamic thermodynamic sea ice model. Journal of Physical Oceanography, 9(4):815–846.

Hopkins, M. A. (1996). On the mesoscale interaction of lead ice and floes. Journal of Geophysical Research: Oceans, 101(C8):18315-18326.

Hopkins, M. A., Frankenstein, S., & Thorndike, A. S. (2004). Formation of an aggregate scale in arctic sea ice. Journal of Geophysical Research: Oceans, 109(C1).

Hopkins, M. A., & Thorndike, A. S. (2006). Floe formation in Arctic sea ice. Journal of Geophysical Research: Oceans, 111(C11).

Hunkins, K. (1975). The oceanic boundary layer and stress beneath a drifting ice floe. Journal of Geophysical Research, 80(24):3425-3433.

Hutchings, J. K., Heil, P., & Hibler III, W. D. (2005). Modeling linear kinematic features in sea ice. Monthly Weather Review, 133(12):3481-3497.

Kalnay, E., Kanamitsu, M., Kistler, R., Collins, W., Deaven, D., Gandin, L., Iredell, M., & et. al. (1996). The NCEP/NCAR 40-year reanalysis project. Bulletin of the American Meteorological Society, 77(3):437–471.

Kovacs, A., & Sodhi, D. S. (1980). Shore ice pile-up and ride-up: Field observations, models, theoretical analyses. Cold Regions Science and Technology 2:210–288.

Kulchitsky, A. V., Hutchings, J. K., Johnson, J. B., & Velikhovskiy, G. (2016, December). Siku DEM Simulations of Beaufort Sea-Ice Fracture Pattern. In: AGU Fall Meeting Abstracts.

Kulchitsky, A., & Hutchings, J. (2014). Development of an accurate model of the Beaufort and Chukchi ice drift and dispersion for forecasting spill trajectories and providing decision support for spill response. In: Annual Report No. 21. OCS Study BOEM 2015-015. University of Alaska Coastal Marine Institute and USDOI, BOEM Alaska OCS Region, 52p.

Kwok, R., & Cunningham, G. F. (2008). ICES at over Arctic sea ice: Estimation of snow depth and ice thickness. Journal of Geophysical Research: Oceans, 113(C8).

Lewis, B., Hutchings, J. K., & Mahoney, A. (2016). Repeating Beaufort lead patterns and their relation to weather. In Proceedings of the 23d IAHR International Symposium on Ice, Ann Arbor, Michigan, USA.

Lewis, B. (in preparation). Master's Thesis, College of Earth Ocean and Atmospheric Sciences, Oregon State University.

Mahoney, A. R., Eicken, H., Gaylord, A. G., & Shapiro, L. (2007). Alaska landfast sea ice: Links with bathymetry and atmospheric circulation. Journal of Geophysical Research: Oceans, 112(C2).

Mahoney, A., H. Eicken, L. Shapiro, R. Gens, T. Heinrichs, F. Meyer, and A. Graves-Gaylord. 2012. Mapping and Characterization of Recurring Spring Leads and Landfast Ice in the Beaufort and Chukchi Seas. Final Report. OCS Study BOEM 2012-067, University of Alaska Fairbanks and USDOI, BOEM Alaska OCS Region, 154 p.

Matuttis, H. G., Luding, S., & Herrmann, H. J. (2000). Discrete element simulations of dense packings and heaps made of spherical and non-spherical particles. Powder Technology, 109(1): 278–292.

McPhee, M. G. (1979). The effect of the oceanic boundary layer on the mean drift of pack ice: Application of a simple model. Journal of Physical Oceanography, 9(2):388–400.

Renka, R. J. (1997). Algorithm 772: STRIPACK: Delaunay triangulation and Voronoi diagram on the surface of a sphere. ACM Transactions on Mathematical Software (TOMS), 23(3):416–434.

Schreyer, H. L., Sulsky, D. L., Munday, L. B., Coon, M. D., & Kwok, R. (2006). Elastic-decohesive constitutive model for sea ice. Journal of Geophysical Research: Oceans, 111(C11).

Sodhi, D. S. (1977). Ice arching and the drift of pack ice through restricted channels (No. CRREL-77-18). Cold Regions Research and Engineering Lab, Hanover, NH, USA.

Taylor, P. D., & Feltham, D. L. (2004). A model of melt pond evolution on sea ice. Journal of Geophysical Research: Oceans, 109(C12).

Thorndike, A. S. (1986). Kinematics of sea ice. In: The Geophysics of Sea Ice, Springer, US. pp. 489–549.

Thorndike, A. S., & Colony, R. (1982). Sea ice motion in response to geostrophic winds. Journal of Geophysical Research: Oceans, 87(C8), 5845-5852.

Walton, O. R., & Braun, R. L. (1993). Simulation of rotary-drum and repose tests for frictional spheres and rigid sphere clusters. In: 5-th Joint DOE/NSF workshop on flow of particulates and fluids, Ithaca, NY, USA.

Wang, K. (2007). Observing the yield curve of compacted pack ice. Journal of Geophysical Research: Oceans, 112(C5).

Wang, R., & Shen, H. H. (2010). Gravity waves propagating into an ice-covered ocean: A viscoelastic model. Journal of Geophysical Research: Oceans, 115(C6).

Wessel, P., & Smith, W. H. (1996). A global, self-consistent, hierarchical, high-resolution shoreline. Journal of Geophysical Research, 101:8741–8743.

Wessel, P., Smith, W. H., Scharroo, R., Luis, J., & Wobbe, F. (2013). Generic mapping tools: Improved version released. Eos, Transactions American Geophysical Union, 94(45):409–410.

Wilchinsky, A. V., & Feltham, D. L. (2006). Anisotropic model for granulated sea ice dynamics. Journal of the Mechanics and Physics of Solids, 54(6):1147–1185.

Wilchinsky, A. V., Feltham, D. L., & Hopkins, M. A. (2010). Effect of shear rupture on aggregate scale formation in sea ice. Journal of Geophysical Research: Oceans (1978–2012), 115(C10).

APPENDIX

Example of Siku scenario script

Below is an example of a Siku scenario script written in Python. It includes an example of February 16, 1994 case study run. This scenario uses previously generated ice elements and uses built-in plotting functions to visualize the results during the runs using GMT library. This example script is located in Siku model code in “samples” directory together with many other examples that can be used with the model. The script shows how a Python program allows the user to initialize the model with the data, program the output, and monitor the runs. The scripts interact with the Siku kernel by Siku Callback functions defined in the script and provide the input using Siku namespace.

```
'''Siku scenario

   Sea ice Beaufort sea ice high resolution example
   for February 16, 1994 case study

   Be sure that siku module is in your PYTHONPATH.
   Use python3 for checking. It is not compatible with python2.x

   (c)2014-2017 UAF
   GPLv3 or later license (same as siku)

'''

import subprocess
import os
import math
import sys
import datetime
import mathutils
import numpy

# Siku modules import
import siku
from   siku import polygon
from   siku import element
from   siku import material
from   siku import geocoords
from   siku import regrid
from   siku import gmt_Plotter
GMT_Plotter = gmt_Plotter.GMT_Plotter
from   siku import poly_voronoi
PolyVor = poly_voronoi.PolyVor
from   siku import h5load
hload = h5load.Loader
from   siku import wnd

def main():

    # ---------------------------------------------------------------------
    # Define material
    # ----------------------------------------------------------------------

    ice = material.Material()       # default ice values, 10 thicknesses
    ice.name = 'ice'                # prefer to use our own name instead
                                    # of default

    siku.materials.append( ice )    # list of all materials

    # table of material names for convenience
    matnames = {
        'ice': 0,
    }

    # ----------------------------------------------------------------------
    #  Wind initializations (NMC grid example)
    # ----------------------------------------------------------------------

    siku.uw = wnd.NMCVar( 'u1994.nc', 'uwnd' )
    siku.vw = wnd.NMCVar( 'v1994.nc', 'vwnd' )

    start =  datetime.datetime  ( 1994, 2, 16, 00, 00, 00 )
    for i in range(len( siku.uw.times )):
        if siku.uw.times[i] >= start:
            break
    st_t_ind = i
    siku.time.update_index = i - 1
    print( 'start time: ' + str( start ) + ' at position: ' + str( i ) + \
           ' of ' + str( len( siku.uw.times ) ) + '\n\n' )

    siku.wind = wnd.NMCSurfaceVField( siku.uw, siku.vw, st_t_ind )

    siku.settings.wind_source_type = siku.WIND_SOURCES['NMC']
    siku.settings.wind_source_names = [ 'u1994.nc', 'v1994.nc' ]

    # ----------------------------------------------------------------------
    # date/time settings
    # ----------------------------------------------------------------------

    siku.time.dts      = datetime.timedelta ( seconds = 600 )
    hour = datetime.timedelta ( minutes = 60 )

    ## time inits by NMC grid times
    siku.time.start = siku.uw.times[st_t_ind]
    siku.time.last = siku.uw.times[st_t_ind]
    siku.time.last_update = siku.time.last
    siku.time.finish = siku.uw.times[st_t_ind] + hour * 90
    #siku.time.dt = datetime.timedelta ( milliseconds = 1 )
    siku.time.dt = ( siku.time.finish - siku.time.start ) / 3600

    # ----------------------------------------------------------------------
    # elements
    # ----------------------------------------------------------------------

    coords = []
```

```
    siku.elements = []

    # ---------------------- voronoi initialization ------------------------
    print('\nLoading polygons')
    ## North cap
    PV = PolyVor( 'alaska.voronoi.xyz', 'alaska.voronoi.xyzf' )
    ## Channel (handmade)
##    PC = PolyVor( 'alaska.voronoi.xyz', 'alaska.voronoi.xyzf' )

    PV.filter_( 0, 360, 60, 90 )
##    PC.filter_( 179, 187, 54, 60 )
    print('Deleting land polygons')
    PV.clear_the_land()

    coords = PV.coords

    siku.tempc = coords # for debug

    ### Initializing elements with polygon vertices
    for c in coords:
        siku.P.update( c )

        # Element declaration
        E = element.Element( polygon = siku.P, imat = matnames['ice'] )
        E.monitor = "drift_monitor"
        gh = [ 0.2, 0.2, 0.4, 0.2, 0.0,
               0.0, 0.0, 0.0, 0.0, 0.0 ]
        E.set_gh( gh, ice )

        # all elements in the list
        siku.elements.append( E )

    ## Core will mark polygons, those contain at leas one point from next
    ## file as 'static'
    siku.settings.border_mark = 1
    siku.settings.borders = 'contours.ll'

    print('Marking borders with GMT')
    bor = PV.get_border_by_gmt()
    for b in bor:
        siku.elements[ b ].flag_state = element.Element.f_static
    print('Done\n\n')

    ## Plotter initialization
    siku.plotter = GMT_Plotter( 'beaufort94_plot.py' )

    ### period of picturing
    siku.diagnostics.monitor_period = 30
    siku.drift_monitor = drift_monitor
    siku.diagnostics.step_count = 0

    siku.settings.contact_method = siku.CONTACT_METHODS['sweep']
    siku.settings.force_model = \
                    siku.CONTACT_FORCE_MODEL['distributed_spring']
    # name of file to load from
    siku.settings.loadfile = 'save_test.h5'

    siku.settings.phys_consts = { 'rigidity' : 10.0,#10,
                                  'viscosity' : 1.0,#1.0,#1
                                  'rotatability' : 0.750,#0.75
                                  'tangency' : -0.00003,#-0.00003

                                  'elasticity' :-50000000.0,#-5000000.0,
                                  'bendability' : 1.0,#1.0,
                                  'solidity' : 0.05,#0.05,
                                  'tensility' : 0.30,#0.615,

                                  'anchority' : 0.0005,
                                  'windage': 0.05, #0.05
                                  'fastency' : 0.50, #0.5

                                  'sigma' : 1.0,        # -//- rigidity
                                  'etha' : 1.0          # -//- viscosity
                                  }

    # ---------------------------------------------------------------------
    #  Diagnostics function for the winds
    # ------------------------------abs2( e.V )---------------------------
##    # We create a grid and append it to monitor grids
##    siku.diagnostics.wind_counter = 0
##    rg = regrid.Regrid()
##    mesh_01 = rg.globe_coverage( 5.0 )
##    siku.diagnostics.meshes.append( mesh_01 )
##    siku.diagnostics.wind.append(
##        ( winds_diag, 0, siku.time.start, 2*siku.time.dt ) )

    # ---------------------------------------------------------------------
    #  Settings
    # ---------------------------------------------------------------------

    # ---------------------------------------------------------------------
    #  Callback flag-mask generator
    # ---------------------------------------------------------------------

    siku.callback.pretimestep = pretimestep
    siku.callback.aftertimestep = aftertimestep
    siku.callback.conclusions = conclusions
    siku.callback.initializations = initializations
    siku.callback.updatewind = updatewind

    ##
    siku.callback.presave = presave

    siku.err_test = {}

    return 0

def presave( t, n, ns ):
    '''no saving at all'''
    return

# -----------------------------------------------------------------------------

def initializations( siku, t ):
    subprocess.call(["gmtset", "PS_MEDIA=Custom_24cx20c"])

# -----------------------------------------------------------------------------

def conclusions( siku, t ):

    with open("err_time.txt", 'w') as erf:
        for i in siku.err_test:
            erf.write( str(i) + ' : ' )#+ ':\n' )
            erf.write( str( len( siku.err_test[i] ) ) )
            erf.write( '\n' )

    print('creating .gif')
    subprocess.call( "nice convert -density 300 -delay 10 beaufort*.eps beaufort.gif",
\
                     shell=True )

# -----------------------------------------------------------------------------

def pretimestep( t, n, ns ):
    status = siku.MASK['NONE']
    siku.diagnostics.step_count = n

    siku.local.poly_f = open( 'Polygons.txt', 'w' )

    # step by NMC own time step
    if t >= siku.uw.times[siku.time.update_index + 1]: # siku.time.last: #
        status += siku.MASK['WINDS']
        siku.time.last = t# siku.time.finish#

    # and change the winds here
    # ~!wind is changed with another call

    # and save the current time in a structure
    # ~!current time is saved in siku.time.last
    return status

# -----------------------------------------------------------------------------

def updatewind( siku, t ):
    siku.time.update_index += 1
    siku.time.last_update = t
    siku.wind = \
            wnd.NMCSurfaceVField(siku.uw, siku.vw, siku.time.update_index)
    print( str( t ) + '\n' )
    pass

# -----------------------------------------------------------------------------

def aftertimestep( t, n, ns ):
    siku.local.poly_f.close()
    if siku.diagnostics.step_count % siku.diagnostics.monitor_period == 0:
        pic_name = 'beaufort%03d.eps' % \
            (siku.diagnostics.step_count / siku.diagnostics.monitor_period)
        print('drawing ' + str( pic_name ) )

        siku.plotter.plot( pic_name, siku.time.update_index, siku.wind )

    #siku.local.poly_f.close()
    return 0

# ---------------------------------------------------------------------------

def drift_monitor( t, Q, Ps, st, index, ID, W, F, N, m, I, i, A, a_f, w_f ):

    # create actual quaternion
    q = mathutils.Quaternion( Q )
    C = mathutils.Vector( (0,0,1) )

    # get latitude and longitude of center of mass (0,0,1)
    R = q.to_matrix()
    c = R * C

    # appending vertices to plotting list
    if siku.diagnostics.step_count % siku.diagnostics.monitor_period == 0:
        Pglob = [ R*mathutils.Vector( p ) for p in Ps ]
        vert = [ geocoords.lonlat_deg(mathutils.Vector( p ) ) for p in Pglob ]

        poly = siku.local.poly_f

    return

# -------------------------------------------------------------------------
def winds_diag( t, winds ):

    mesh = siku.diagnostics.meshes[0]
    ez = mathutils.Vector( (0,0,1) )

    return

# ------------------------------------------------------------------------
# Calling main function at the end
# ------------------------------------------------------------------------

siku.main = main()

if __name__ == '__main__':
    sys.exit( siku.main )
```

The Department of the Interior Mission

As the Nation’s principal conservation agency, the Department of the Interior has responsibility for most of our nationally owned public lands and natural resources. This includes fostering the sound use of our land and water resources, protecting our fish, wildlife and biological diversity; preserving the environmental and cultural values of our national parks and historical places; and providing for the enjoyment of life through outdoor recreation.The Department assesses our energy and mineral resources andworks to ensure that their development is in the best interests of all our people by encouraging stewardship and citizen participation in their care. The Department also has a major responsibility for American Indian reservation communities and for people who live in island communities.

The Bureau of Ocean Energy Management

The Bureau of Ocean Energy Management (BOEM) works to manage the exploration and development of the nation's offshore resources in a way that appropriately balances economic development, energy independence, and environmental protection through oil and gas leases, renewable energy development and environmental reviews and studies.