Modeling emergent Arctic sea ice dynamics using small-scale discrete element methods within macro-scale equations

Andrew D. Davis[1], Dimitris Giannakis[1], Brendan West[2], Nate Frisch[2], Devin O'Connor[2], and Matthew Parno[2]

[1]Courant Institute, New York University
[2]Cold Regions Research and Engineering Laboratory (CRREL), U.S. Army Corps

March 11, 2020

Motivation: What is sea ice?

- Sea ice is **dynamic**!
- Complex physical processes (e.g., deformation & fracture)
- **Ocean- and atmosphere-interactions** link sea ice to the greater Arctic system

No characteristic floe size

Complex fracture patterns

Pressure ridges

Motivation: Why do we care about sea ice?

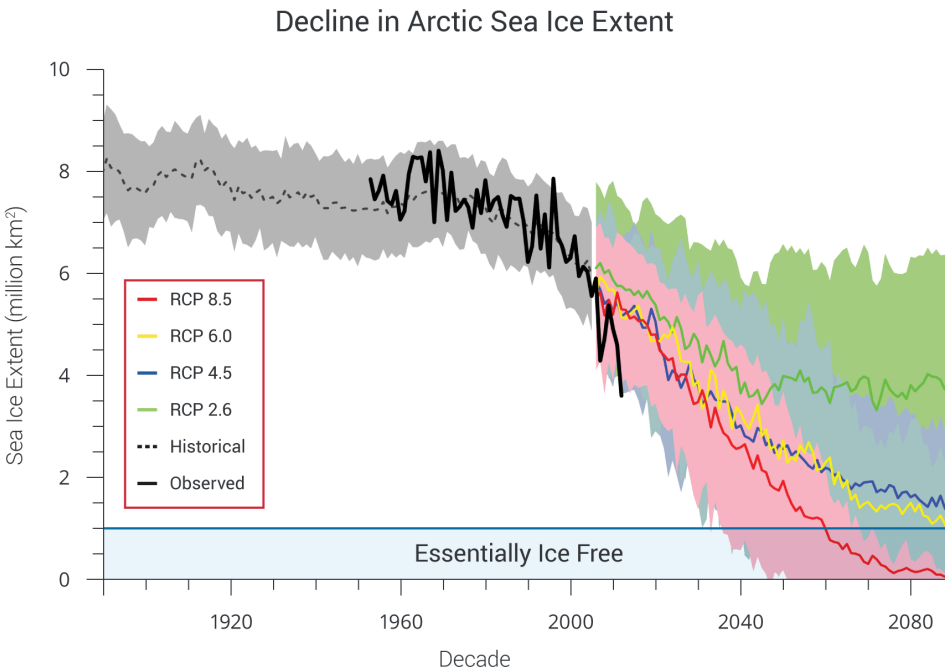


Figure: Nat'l climate assessment, Stroeve et al. (2012)

- Sea ice extent is declining more rapidly than models predict

Implications:

1. Arctic warming affects jet stream dynamics
   Francis et al.
2. Dynamic response across Earth's climate system
3. Commercial and military vessels can navigate an ice-free Arctic
4. Logistics in the Arctic

Figure: Scripps, MOSAiC Cruise

Our modeling goals

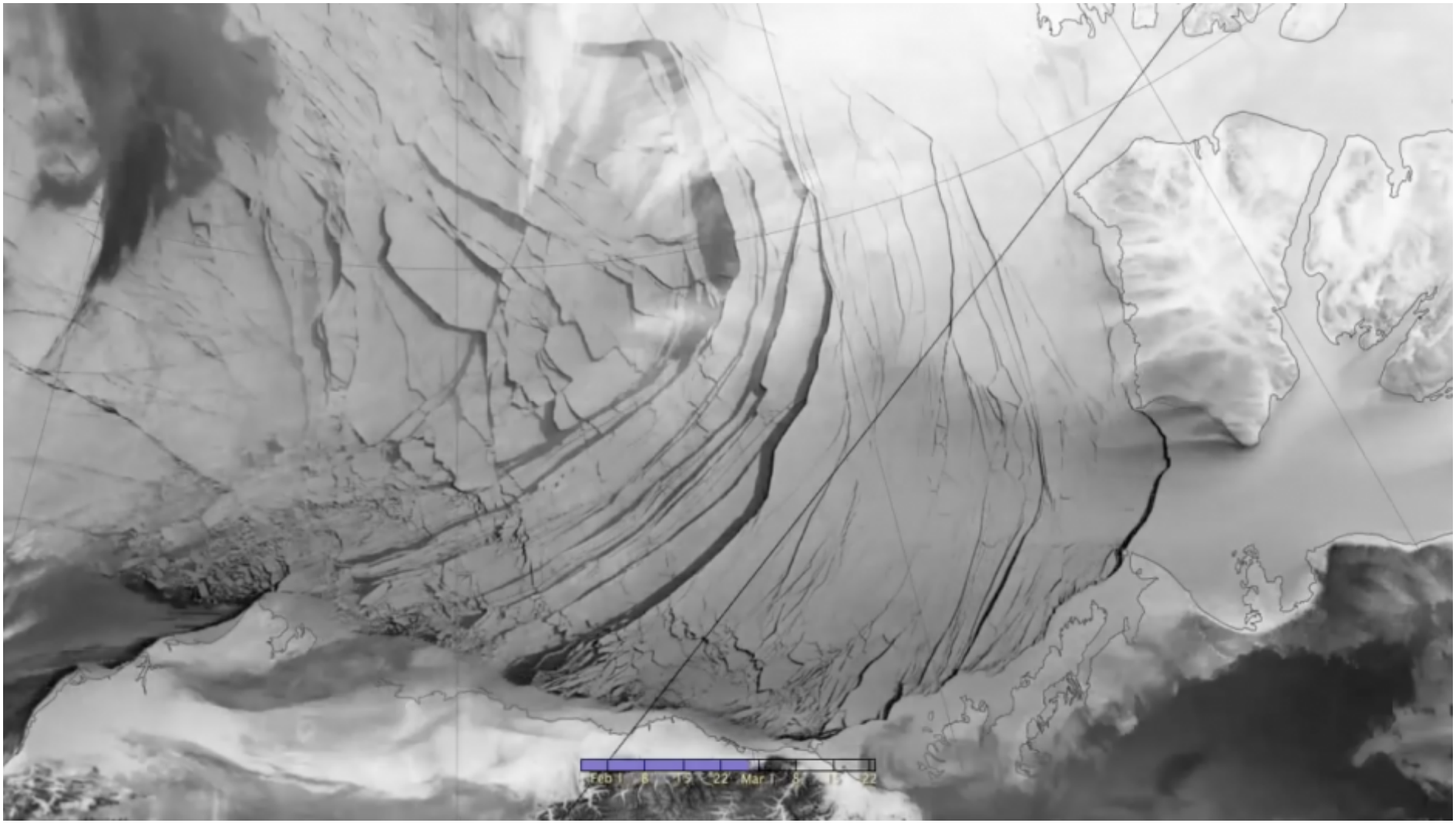


Figure: Satellite image of the Beaufort Gyre, NASA

How do we model **large-scale dynamics**?

Challenges modeling & monitoring the Arctic

Computational feasibility & software engineering:

- Mathematical models are computationally expensive
- Sea ice models require coupling to oceanic & atmospheric models
  - Often developed across different platforms and languages
- We developed a software package (*Particle Level Sets (ParticLS)*)

  Davis et al. (Submitted)

Challenges modeling & monitoring the Arctic

Computational feasibility & software engineering:

- Mathematical models are computationally expensive
- Sea ice models require coupling to oceanic & atmospheric models
  - Often developed across different platforms and languages
- We developed a software package (*Particle Level Sets (ParticLS)*)
  Davis et al. (Submitted)

Unknown physics & mathematical modeling:

- *Continuum mechanics:* deformation, crack formation & propagation
- Complex atmosphere-ice-ocean feedbacks
- Current research: devise a sea ice modeling framework

Challenges modeling & monitoring the Arctic

Computational feasibility & software engineering:

- Mathematical models are computationally expensive
- Sea ice models require coupling to oceanic & atmospheric models
  - Often developed across different platforms and languages
- We developed a software package (*Particle Level Sets (ParticLS)*)
  Davis et al. (Submitted)

Unknown physics & mathematical modeling:

- *Continuum mechanics:* deformation, crack formation & propagation
- Complex atmosphere-ice-ocean feedbacks
- Current research: devise a sea ice modeling framework

Model validation, verification, and calibration:

- Collecting data is expensive & dangerous
  - Remote sensing & in situ observation
- Optimal experimental design helps allocate limited resources
  Huan & Marzouk (2013), Davis & Huan (In prep.)
- Surrogate models make rigorous uncertainty quantification feasible
  Davis et al. (2016 & 2018)

Challenges modeling & monitoring the Arctic

Computational feasibility & software engineering:

- Mathematical models are computationally expensive
- Sea ice models require coupling to oceanic & atmospheric models
  - Often developed across different platforms and languages
- We developed a software package (*Particle Level Sets (ParticLS)*)
  Davis et al. (Submitted)

Unknown physics & mathematical modeling:

- *Continuum mechanics:* deformation, crack formation & propagation
- Complex atmosphere-ice-ocean feedbacks
- Current research: devise a sea ice modeling framework

Model validation, verification, and calibration:

- Collecting data is expensive & dangerous
  - Remote sensing & in situ observation
- Optimal experimental design helps allocate limited resources
  Huan & Marzouk (2013), Davis & Huan (In prep.)
- Surrogate models make rigorous uncertainty quantification feasible
  Davis et al. (2016 & 2018)

Background—Continuum models

- Many models treat sea ice thickness, velocity, and/or concentration as a continuum
- Domain (Arctic basin) is broken into **representative volume elements (RVEs)**
  - Concentration at x is the *average* concentration in the RVE around x
- Basin-scale PDE models solve **mass and energy balance equations** for unknown sea ice concentration/thickness and ice velocity
- **Rheological laws** (models of sea ice "viscosity") govern macro-scale dynamics

  e.g., Dansereau et al. (2017), Feltham (2008)

Background—Discrete element models

What is the thickness, velocity, or concentration at any point in this photo?

- Sea ice is actually made up of discrete "chunks" called **floes**
- Floes range in size from **kilometers to centimeters**
- Floes deform, fracture, and freeze together

A motivating challenge

How do we formulate **one model** that captures **all** of these dynamics?

Particle Level Sets (ParticLS): Overview

- *ParticLS* is an open-source software package for **meshfree methods** (e.g., peridynamics) and **discrete element models**
- Object-oriented framework written in C++
- Not sea ice specific; **designed with sea ice in mind**

Particle Level Sets (ParticLS): Overview

- *ParticLS* is an open-source software package for **meshfree methods** (e.g., peridynamics) and **discrete element models**
- Object-oriented framework written in C++
- Not sea ice specific; **designed with sea ice in mind**

Three components of the software package:

1. **Particle geometries**
   - Particle shapes are defined by signed distance functions
   - The zero level set defines the particle boundary

     Kawamoto et al. (2017)
2. **Contact laws between particles**
   - Discrete contact laws model interacting rigid bodies
   - Continuum contact laws bond particles together to form structures
3. **Supporting algorithms**
   - Particle-particle interaction (e.g., nearest neighbor search)
   - Time integration

The importance of particle shape

- Many existing DEM codes assume simple shapes—typically spheres
  - Computing the distance to neighbors is easy!
  - Results in unrealistic behaviors

- Macro-scale behavior of soil columns requires non-spherical small-scale particles Andrade et al. (many papers)
  - Shape is important!

- Two ways of creating abstract shapes
  1. Clustering simple particles into a larger particle Garcia et al. (2009)
  2. **Represent complex particle geometries using level sets** Kawamoto et al. (2017)

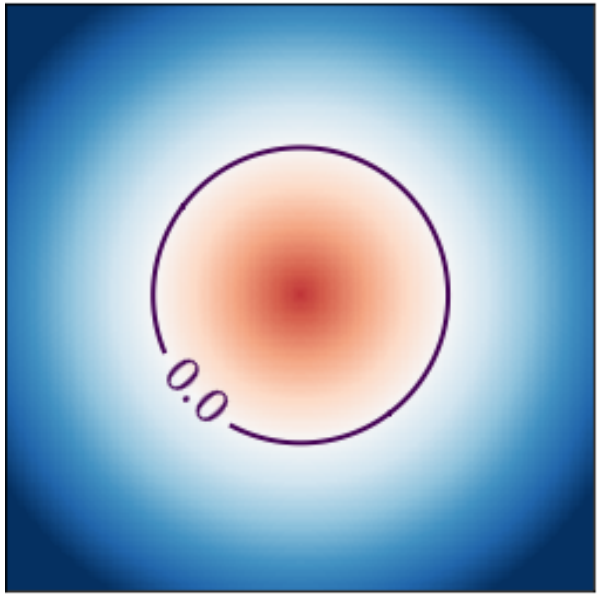
Level set particle representation

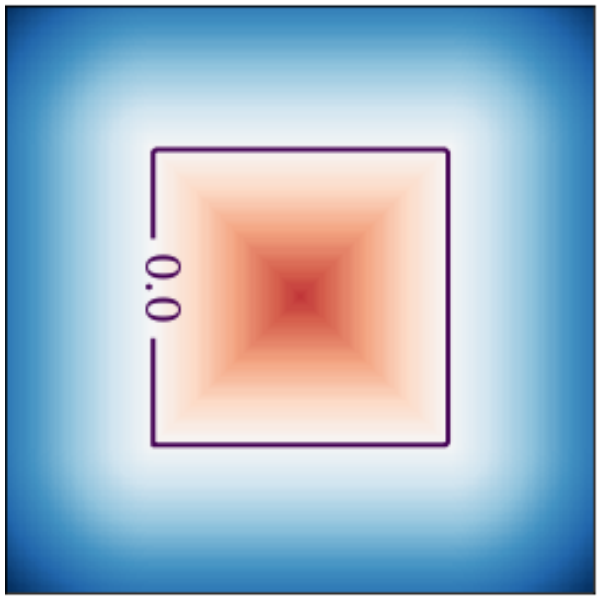
- The **level set function:** $\phi : \mathbb{R}^d \mapsto \mathbb{R}$
  - $\phi(\boldsymbol{X}) = 0$: *on* the particle boundary
  - $\phi(\boldsymbol{X}) < 0$: *inside* the particle
  - $\phi(\boldsymbol{X}) > 0$: *outside* the particle
  - $|\phi(\boldsymbol{X})|$ is the *distance* the the closest point on the boundary
- $\phi(\boldsymbol{X})$ defines a **reference geometry** such that the center of mass is zero

$$\mathbf{0} = \int_{\phi(\boldsymbol{X}) \leq 0} \rho(\boldsymbol{X}) \boldsymbol{X} \, d\boldsymbol{X}$$

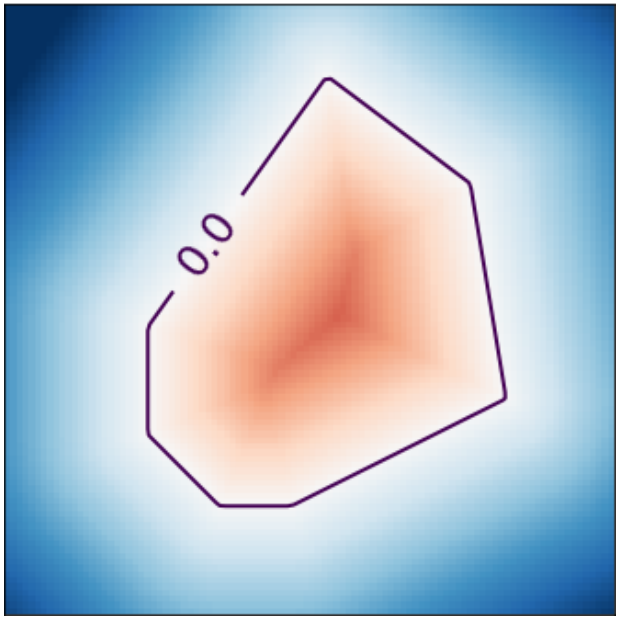
- A hypersphere: $\phi(\boldsymbol{X}) = \|\boldsymbol{X}\|_2 - R$
- A **particle** is defined by its center of mass $\bar{\boldsymbol{x}}$ and a rotation matrix $\boldsymbol{Q}$ such that

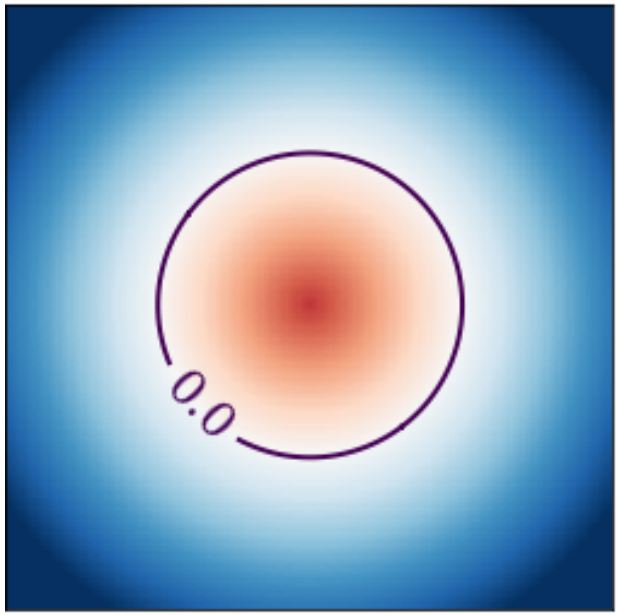
$$\boldsymbol{X} = \boldsymbol{Q}(\boldsymbol{x} - \bar{\boldsymbol{x}})$$

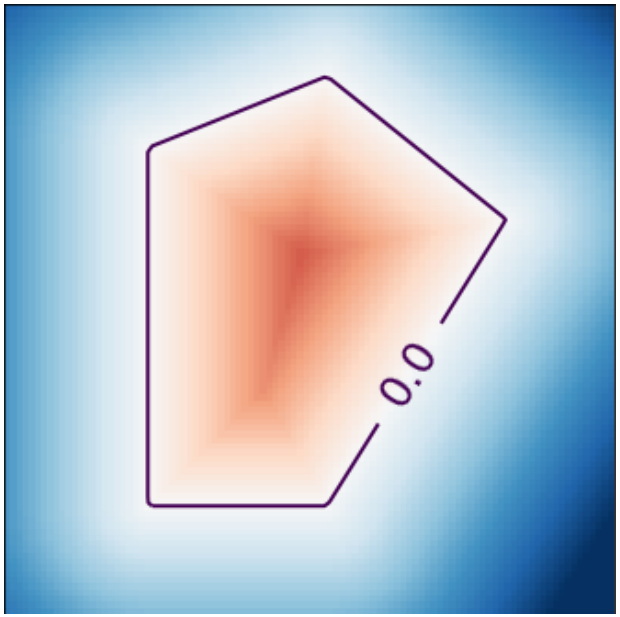


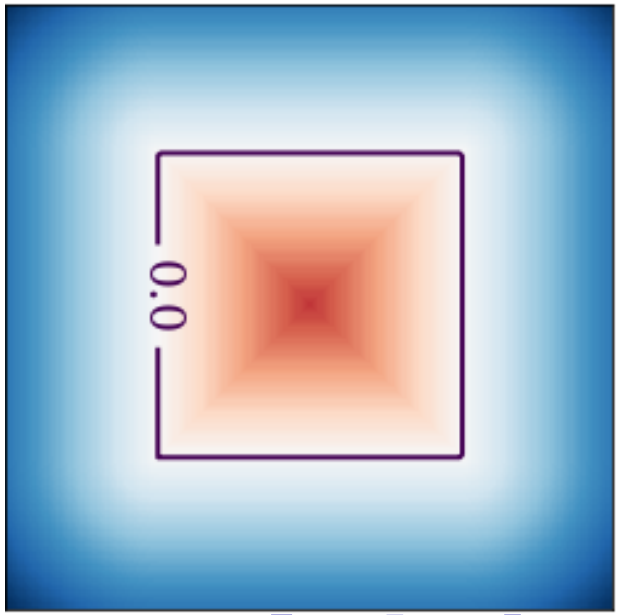


Level set particle examples









Advantages of the level set representation

Computing the contact between particles is **computationally efficient even for abstract shapes**

Advantages of the level set representation

Computing the contact between particles is **computationally efficient even for abstract shapes**

- Redefine the level set in global coordinates $\phi(\boldsymbol{x}) \leftarrow \phi(\boldsymbol{Q}(\boldsymbol{x} - \bar{\boldsymbol{x}}))$
- Let $\phi(\boldsymbol{x})$ (*particle A*) and $\phi'(\boldsymbol{x})$ (*particle B*) be distinct particles

Advantages of the level set representation

Computing the contact between particles is **computationally efficient even for abstract shapes**

- Redefine the level set in global coordinates $\phi(\boldsymbol{x}) \leftarrow \phi(\boldsymbol{Q}(\boldsymbol{x} - \bar{\boldsymbol{x}}))$
- Let $\phi(\boldsymbol{x})$ (*particle A*) and $\phi'(\boldsymbol{x})$ (*particle B*) be distinct particles
- The closest point on *particle A* to *particle B* solves:

Deepest point optimization

$$\boldsymbol{x}^* = \min \phi'(\boldsymbol{x}) \text{ such that } \phi(\boldsymbol{x}) \leq 0$$

- $\min \phi'(\boldsymbol{x})$ minimizes the distance to *particle B* (if $\phi'(\boldsymbol{x}^*) > 0$) **OR** finds the deepest point inside *particle B* (if $\phi'(\boldsymbol{x}^*) \leq 0$)
- The constraint ensures that the optimal point is on or inside *particle A*

Advantages of the level set representation

Computing the contact between particles is **computationally efficient even for abstract shapes**

- Redefine the level set in global coordinates $\phi(\boldsymbol{x}) \leftarrow \phi(\boldsymbol{Q}(\boldsymbol{x} - \bar{\boldsymbol{x}}))$
- Let $\phi(\boldsymbol{x})$ (*particle A*) and $\phi'(\boldsymbol{x})$ (*particle B*) be distinct particles
- The closest point on *particle A* to *particle B* solves:

Deepest point optimization

$$\boldsymbol{x}^* = \min \phi'(\boldsymbol{x}) \text{ such that } \phi(\boldsymbol{x}) \leq 0$$

- $\min \phi'(\boldsymbol{x})$ minimizes the distance to *particle B* (if $\phi'(\boldsymbol{x}^*) > 0$) **OR** finds the deepest point inside *particle B* (if $\phi'(\boldsymbol{x}^*) \leq 0$)
- The constraint ensures that the optimal point is on or inside *particle A*
- If $\phi'(\boldsymbol{x}^*) > 0$, the particles are **not overlapping**
- If $\phi'(\boldsymbol{x}^*) \leq 0$, the particles are **overlapping**

Solving the optimization problem

Deepest point optimization

$$\boldsymbol{x}^* = \min \phi'(\boldsymbol{x}) \text{ such that } \phi(\boldsymbol{x}) \leq 0$$

- $\phi(\boldsymbol{x})$ is **NOT** differentiable (e.g., polyhedra)
- Reformulate

$$\boldsymbol{x}^* = \min \phi'(\boldsymbol{x}) + \begin{cases} 0 & \text{if } \phi(\boldsymbol{x}) \leq 0 \\ \infty & \text{else} \end{cases} = \min \phi'(\boldsymbol{x}) + g(\boldsymbol{x})$$

- Define the proximal operator

$$\text{prox}_{\lambda f}(\boldsymbol{y}) = \arg\min_{\boldsymbol{x} \in \mathbb{R}^d} \left(f(\boldsymbol{x}) + \frac{1}{2\lambda} \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \right)$$

- Note that the proximal operator of g is

$$\text{prox}_{\lambda g}(\boldsymbol{y}) = \begin{cases} \boldsymbol{y} \text{ if } \phi(\boldsymbol{x}) \leq 0 \\ \text{Projection onto the boundary of } \textit{Particle } B \end{cases}$$

Solving the optimization problem

Deepest point optimization

$$\boldsymbol{x}^* = \min \phi'(\boldsymbol{x}) \text{ such that } \phi(\boldsymbol{x}) \leq 0$$

Proximal operator

$$\text{prox}_{\lambda\phi'}(\boldsymbol{y}) = \arg\min_{\boldsymbol{x}\in\mathbb{R}^d} \left(\phi' + \frac{1}{2\lambda}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \right)$$

- Often, the level set function has structure that makes computing (or approximating) the proximal operator easy
- Given the projection and proximal operators, solve the deepest point optimization with **alternating direction method of multipliers**
  Glowinski & Marrocco (1975), Goldstein et al. (2014), and others
- **Special cases:** we forgo solving the optimization problem
  - Sphere-sphere contact is easy!
  - Polyhedra-polyhedra, reformulate as a linear program

Particle-particle interaction

- *ParticLS* leverages the level set function & specialized optimization to decide if particles are in contact (overlapping)
- *ParticLS* includes other heuristics for computational efficiency
  - Symmetry: only need to compute the optimization once per pair
  - Bounding spheres non-overlapping—don't bother with the optimization
- Given a pair of particles, we compute $\boldsymbol{x}$ and $\boldsymbol{x}'$ which represent **bond locations**—the closest (or deepest) point from one particle to another
- $\boldsymbol{\xi} = \boldsymbol{x} - \boldsymbol{x}'$ is the **bond** and $\|\boldsymbol{\xi}\|_2$ is the **penetration depth** (distance apart or overlap distance)
- Bond locations are unique for convex particles
- Local minima are *potential* contact points for non-convex particles

Evolving particle states

- **Particle state** (for particle p_i):
  1. Center of mass $\bar{\boldsymbol{x}}_i$
  2. Velocity $\boldsymbol{u}_i$
  3. Orientation $\boldsymbol{Q}_i$—represented as a unit quaternion q_i
  4. Angular velocity $\boldsymbol{\omega}_i$
- In 3D: 13 unknowns/particle; In 2D: 6 unknowns/particle
- Let $\mathcal{C}_n$ be a **collection** of n particles

Conservation of momentum

$$\frac{d\bar{\boldsymbol{x}}_i}{dt} = \boldsymbol{u}_i \quad \text{and} \quad m_i \frac{d\boldsymbol{u}_i}{dt} = \sum_{j=1}^{n} \underbrace{\boldsymbol{F}(p_i, p_j)}_{\text{Contact laws}} + \underbrace{\boldsymbol{B}(p_i)}_{\text{Body forces}}$$

Conservation of angular momentum

$$\frac{dq_i}{dt} = \frac{1}{2}\boldsymbol{\omega}_i * q_i \quad \text{and} \quad \boldsymbol{I}_i \frac{d\boldsymbol{\omega}_i}{dt} = \sum_{j=1}^{n} \tau(p_i, p_j) - \frac{d\boldsymbol{I}_i}{dt}\boldsymbol{\omega}_i$$

Contact laws between particles

Conservation of momentum

$$\frac{d\bar{\boldsymbol{x}}_i}{dt} = \boldsymbol{u}_i \quad \text{and} \quad m_i \frac{d\boldsymbol{u}_i}{dt} = \sum_{j=1}^{n} \underbrace{\boldsymbol{F}(p_i, p_j)}_{\text{Contact laws}} + \underbrace{\boldsymbol{B}(p_i)}_{\text{Body forces}}$$

- Given particle geometry, the contact laws and body forces completes the model specification
- Two types of contact laws:
  1. **Discrete:** particles act as independent bodies
  2. **Continuum:** particles are bonded together to form a larger body
- In appropriate limits (infinitely small, tightly packed particle) we recover the peridynamic equation Silling et al. (many papers)

$$\rho(\boldsymbol{x}) \frac{d\boldsymbol{u}(\boldsymbol{x})}{dt} = \int_{\delta(\boldsymbol{x})} \eta(\boldsymbol{x}, \boldsymbol{x}')\, d\boldsymbol{x} + \boldsymbol{B}(\boldsymbol{x})$$

Comments on the peridynamic model

$$\rho(\boldsymbol{x})\frac{d\boldsymbol{u}(\boldsymbol{x})}{dt} = \int_{\delta(\boldsymbol{x})} \boldsymbol{\eta}(\boldsymbol{x}, \boldsymbol{x}')\, d\boldsymbol{x} + \boldsymbol{B}(\boldsymbol{x})$$

- Admits discontinuous solution, allowing for explicit fracture modeling
- In the limit $\delta(\boldsymbol{x}) \to 0$, this converges to classic continuum mechanics
  Silling et al. (2008)
- ParticLS allows **multiple contact laws in one simulation**
  - Bonded particles are numerical approximations to the peridynamic equations (**sea ice floes**)
  - Non-bonded particles behave as rigid bodies (**interacting floes**)

Viscoelastic contact

- Consider two **overlapping** particles i and j
- Their **relative velocities** are

$$\boldsymbol{v}_r = (\boldsymbol{u}_i + \boldsymbol{\omega}_i \times (\boldsymbol{x}_i^* - \bar{\boldsymbol{x}}_i)) - (\boldsymbol{u}_j + \boldsymbol{\omega}_j \times (\boldsymbol{x}_j^* - \bar{\boldsymbol{x}}_j))$$

- Define the **normal force** (recall the bond $\boldsymbol{\xi} = \boldsymbol{x}_i^* - \boldsymbol{x}_j^*$)

$$\boldsymbol{F}_n = -\kappa \boldsymbol{\xi} - \gamma \boldsymbol{v}_r$$

- Their **tangential velocities** are $\boldsymbol{v}_a = \boldsymbol{v}_r - (\boldsymbol{v}_r \cdot \boldsymbol{\xi})\boldsymbol{\xi}/\|\boldsymbol{\xi}\|_2$
- The **elastic shear** evolves according to ($\boldsymbol{F}_{e,s}(t=0) = \boldsymbol{0}$)

$$\frac{d\boldsymbol{F}_{e,s}}{dt} = -\kappa_e \boldsymbol{v}_a$$

- The **shear force** is $\boldsymbol{F}_s = \underbrace{\min\left(\mu \boldsymbol{F}_n, \boldsymbol{F}_{e,s}\right)}_{\text{Coulomb criteria}} - \underbrace{\gamma_v \boldsymbol{v}_a}_{\text{Damping}}$
- The **total force** is $\boldsymbol{F}(p_i, p_j) = \boldsymbol{F}_n + \boldsymbol{F}_s$

Finding nearest neighbors

- Elastic shear forces must be evolved for each pair of particles
- Rather that tracking all possible pairs, update the list **pair force** based on the list of **nearest neighbors**
- More generally, only compute the force contribution $\boldsymbol{F}(p_i, p_j)$ if $\|\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}}_j\|_2 < \delta$
- **"Brute force" algorithm:**

```
for i ← 1 : n
    for j ← i : n
        if ‖x̄_i − x̄_j‖_2 < δ:   p_i and p_j are neighbors
```

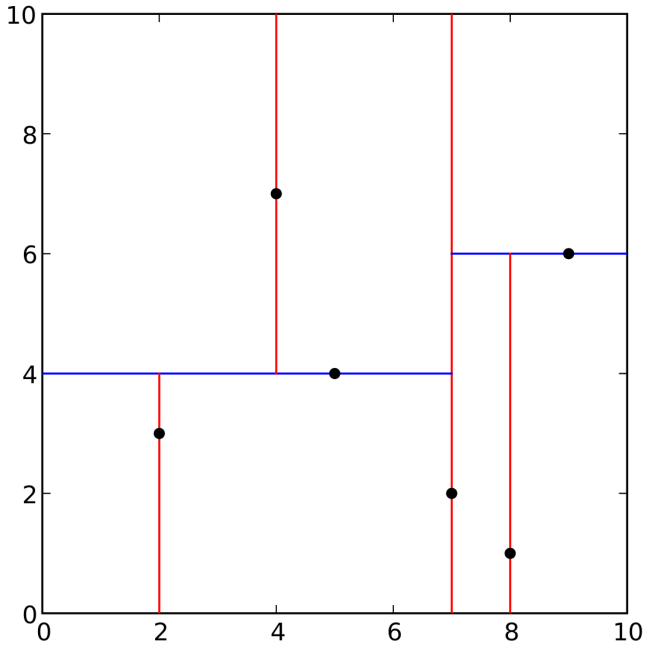
- Complexity $\mathcal{O}(\frac{1}{2}n^2)$—prohibitively slow

k-D search tree

- k-D store each particle as the node on a binary tree
- Left/right subtrees correspond to left/right half-spaces
- Constructing k-D trees costs $\mathcal{O}(n \log n)$
- Finding nearest neighbors costs $\mathcal{O}(\log n)$ per particle
- Total cost: $\mathcal{O}(2n \log n)$

  Bentley (1975)

- *ParticLS* links to the (very fast) *nanoflann* libraries for this functionality



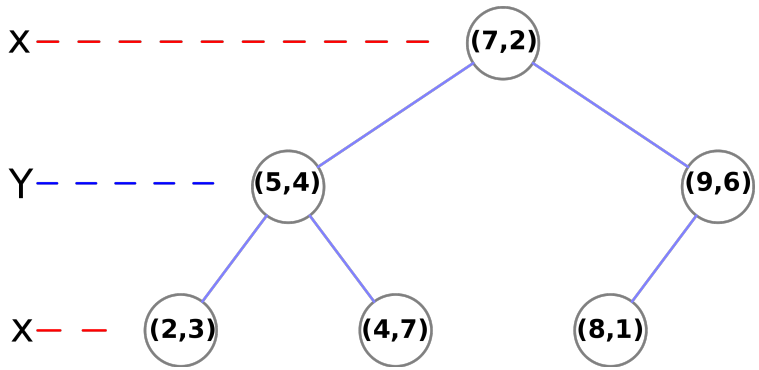


Figure: Wikipedia

Parallel search algorithms

- Partition the domain into overlapping m sub-regions with an equal number of particles
  - k-D tree
- Decide if particle i belongs in sub-region j (cost: $\mathcal{O}(nm)$)

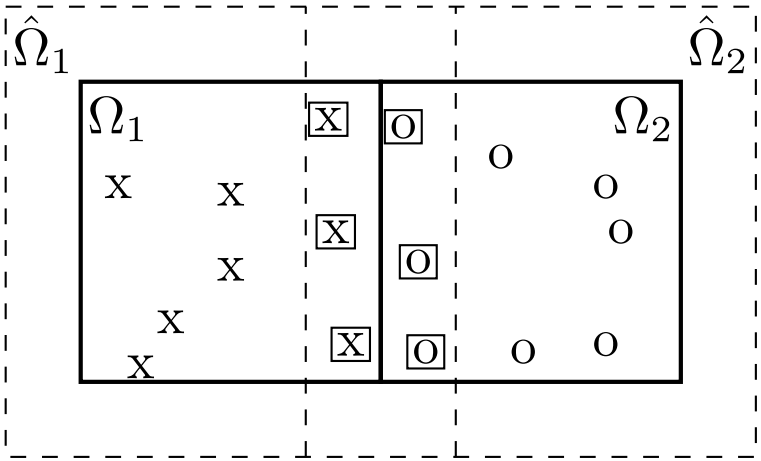


Figure: Davis et al. (Submitted)

- Run search algorithm in search sub-region (n/m particles)
  - Brute force cost: $\mathcal{O}(n^2/(2m^2))$
  - k-D tree cost: $\mathcal{O}(2n/m \log(n/m))$
- Total cost

$$\mathcal{O}\left(nm + \frac{n^2}{2m^2}\right) \quad \text{and} \quad \mathcal{O}\left(nm + 2\frac{n}{m}\log\left(\frac{n}{m}\right)\right)$$

- **Diminishing returns to over-partitioning**
- **Practical observation:** this is not the computational bottle neck

Evolving states in time

- The state of n particles must be integrated in time
  - 6 or 13 dimensional state (2D and 3D)
- k shear forces must also be integrated in time
  - k: number of nearest neighbors pairs we are tracking
  - 2 or 3 dimensional state
- Total: $2k + 6n$ or $3k + 13n$ dimensional state that must be integrated

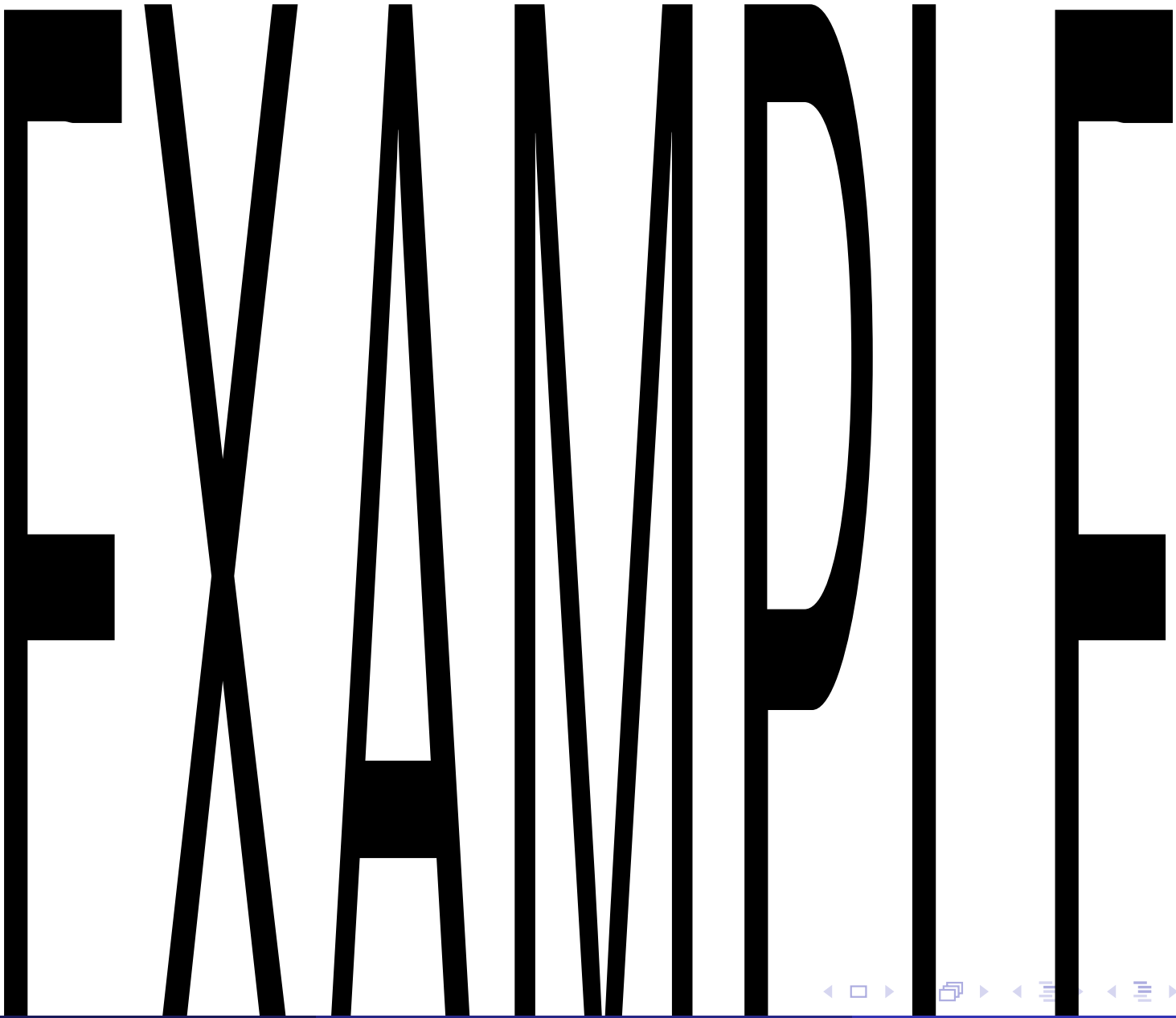
Evolving states in time

- The state of n particles must be integrated in time
  - 6 or 13 dimensional state (2D and 3D)
- k shear forces must also be integrated in time
  - k: number of nearest neighbors pairs we are tracking
  - 2 or 3 dimensional state
- Total: $2k + 6n$ or $3k + 13n$ dimensional state that must be integrated
- Each time steps costs $\mathcal{O}(k + n)$
  - Choose search parameters so that $k \propto n \log n$
  - Total cost of each timestep is $\mathcal{O}(C n \log n + n)$—same as search

Evolving states in time

- The state of n particles must be integrated in time
  - 6 or 13 dimensional state (2D and 3D)
- k shear forces must also be integrated in time
  - k: number of nearest neighbors pairs we are tracking
  - 2 or 3 dimensional state
- Total: $2k + 6n$ or $3k + 13n$ dimensional state that must be integrated
- Each time steps costs $\mathcal{O}(k + n)$
  - Choose search parameters so that $k \propto n \log n$
  - Total cost of each timestep is $\mathcal{O}(C n \log n + n)$—same as search
- High dimensional space requires **explicit time step**
  - *ParticLS* uses Velocity Verlet for particles and a 2^{nd} order Runge Kutte method for other time-dependent states (shear forces)
  - Cannot easily store the linear system required by implicit methods
  - Could leverage sparsity
- Resolving collisions requires **very small timesteps** ($\leq \mathcal{O}(10^{-6})$)

Example: Particle in wind field



Sea ice floes as polyhedra

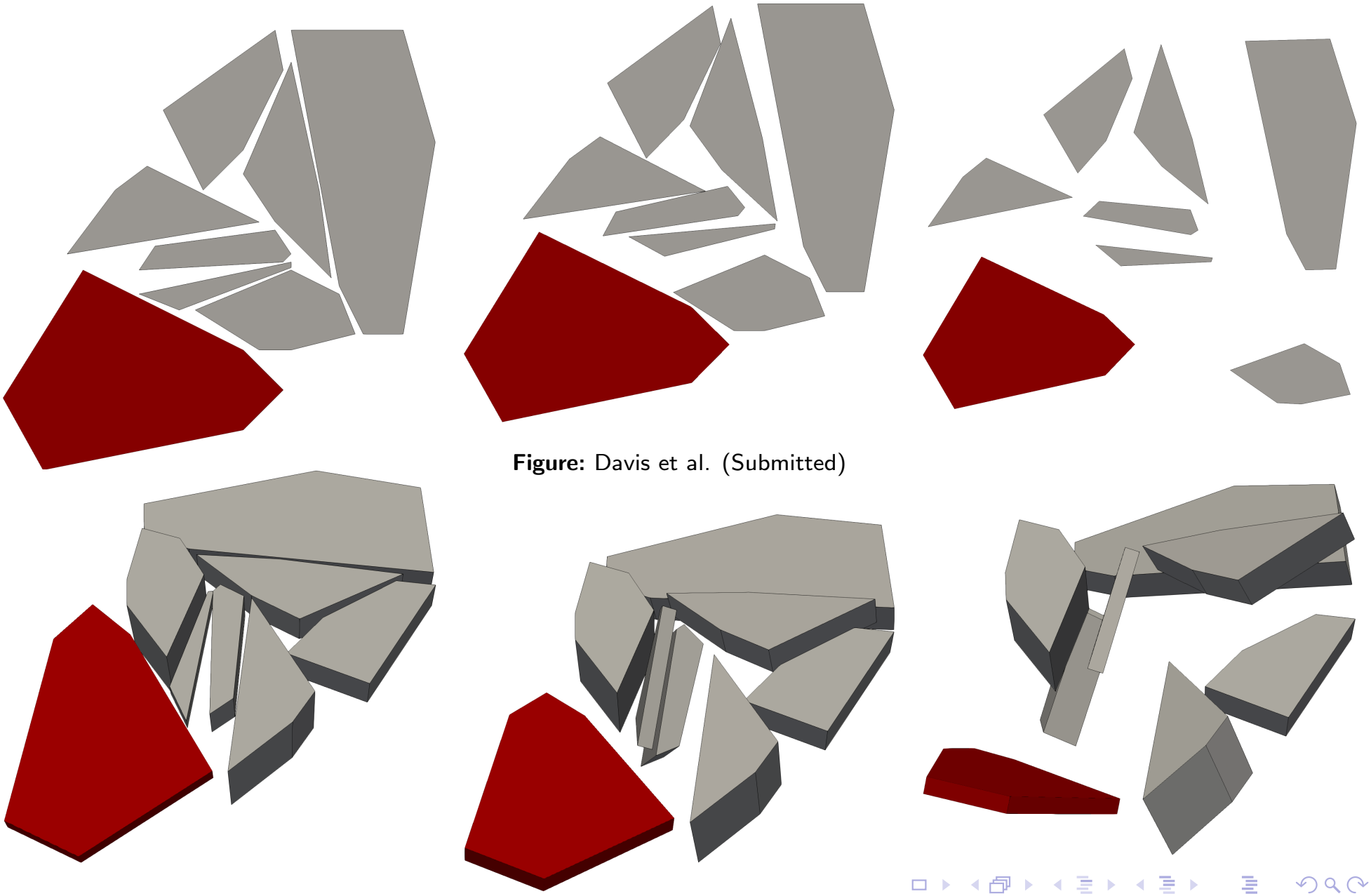


Figure: Davis et al. (Submitted)

Modeling the Nares Strait (MODIS imagery)

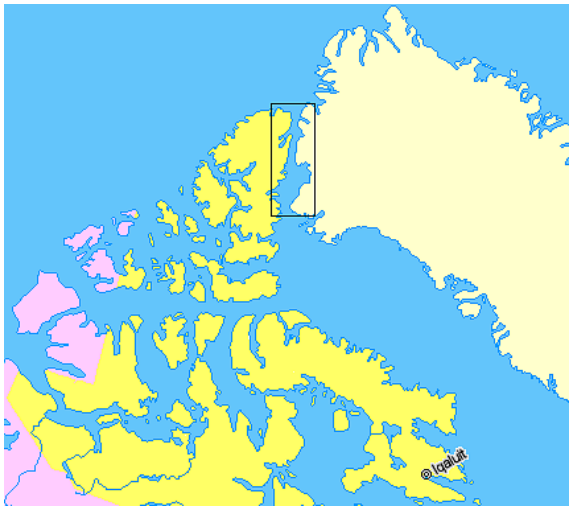


Figure: Wikipedia

- The Nares Strait is located between Canada and Greenland
- Initialize floe particles (polygons) using a **Voronoi tessellation** from MODIS imagery

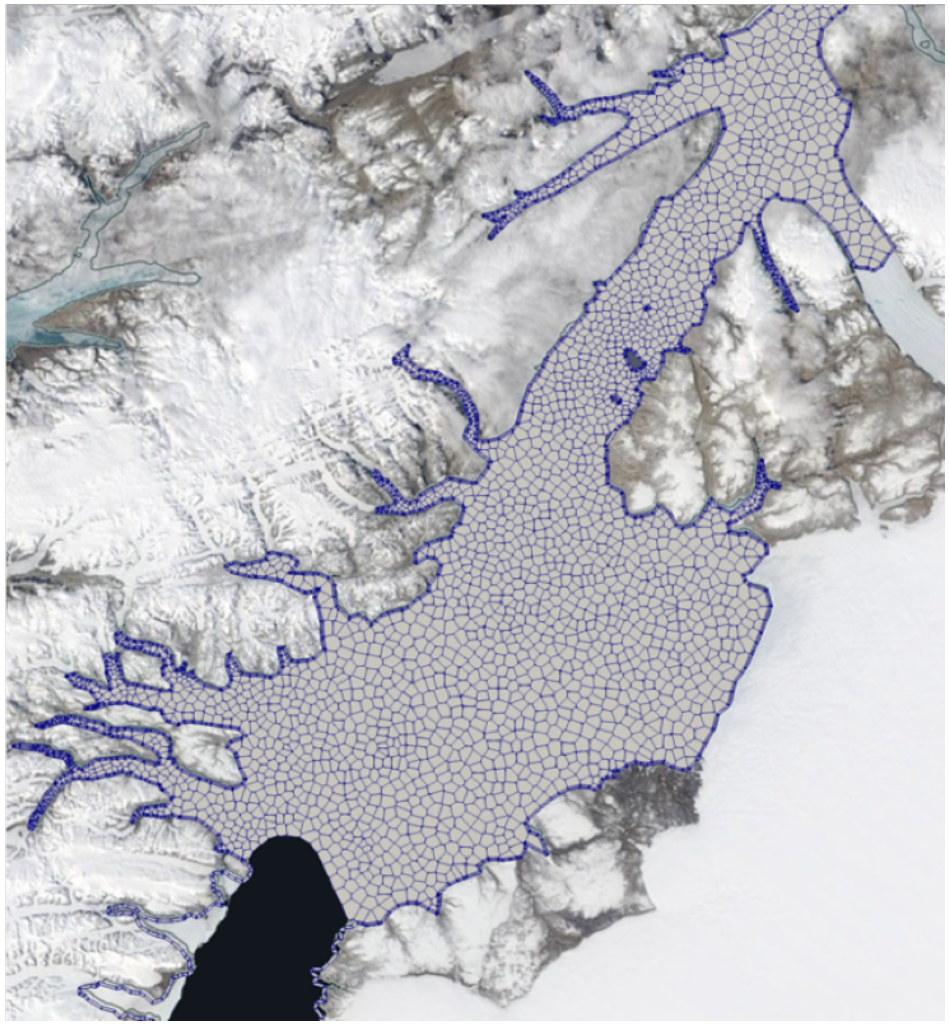


Figure: West et al. (In prep.)

Modeling the Nares Strait (wind data)

- Ice dynamics are primarily driven by wind stresses
  - Ocean velocities are small in the strait
- Wind fields are generated by interpolating observations for nearby weather stations
- Alternatively: generate wind data using a model

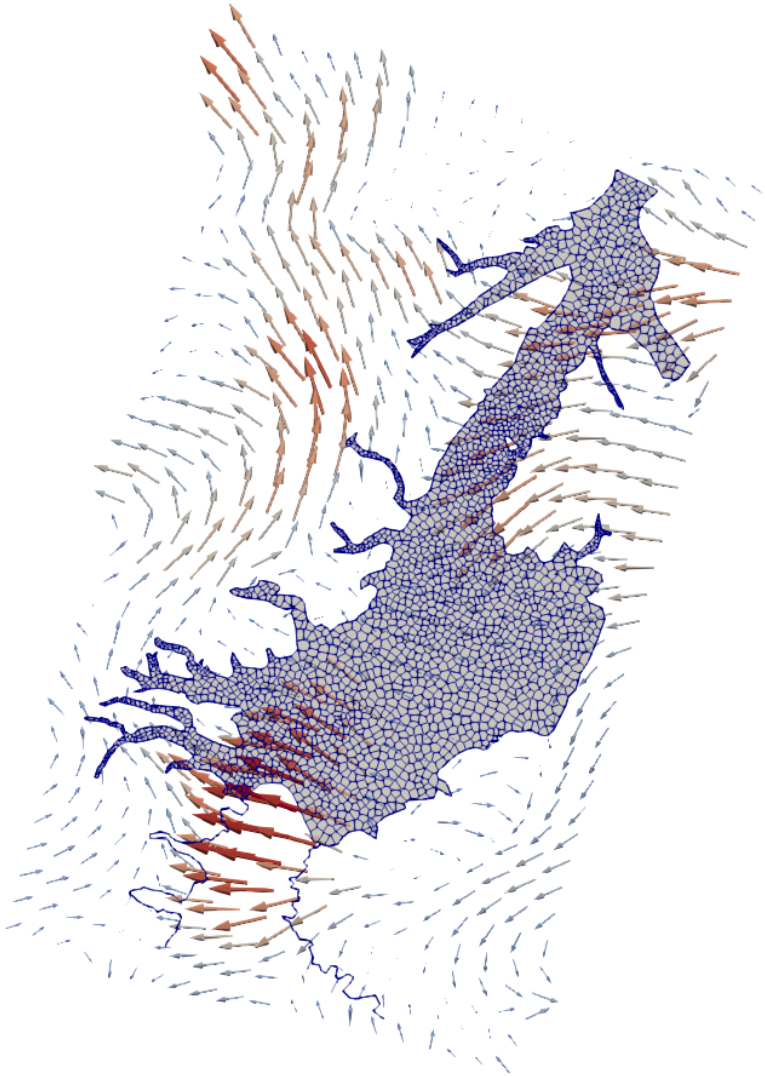


Figure: West et al. (In prep.)

Modeling the Nares Strait (results)

Summary (so far)

- **ParticLS is a robust, open-source software package**
  - C++ (future: Python wrapper)
  - Easily links with existing libraries using `cmake`
  - Object-oriented design Davis et al. (Submited)
- We can *initialize particle simulations from satellite imagery and incorporate existing data & models*
- *ParticLS* allows us to define a variety of contact laws
- Viscoelatic contact acts as proof-of-concept: the software is robust
- **Current research:** which particle geometries and contact laws lead to "sea ice-like" emergent properties?

Back to sea ice: our motivating challenge

How do we formulate **one model** that captures **all** of these dynamics?

Back to sea ice: our motivating challenge

How do we formulate **one model** that captures **all** of these dynamics?

- Continuum models often do not capture the small scale dynamics
  - Some capture macro-scale behavior
  - e.g., Arching in the Nares Strait

    Dansereau et al. (2017)

- Basin scale discrete element models are **computationally infeasible**

- We want to leverage small-scale discrete element methods within macro-scale continuum models
  - Multi-scale models
  - Superparameterization

Two scale modeling framework

1. **Macro-scale model** (e.g., Beaufort Sea $\approx 10^5$ km^2)
   - **GOAL:** Recover important **macro-scale features:**
     - Lead formation, "crack" propagation, ridging
   - Partial differential equations solved using finite element methods
   - Potentially coupled with oceanic & atmospheric models
2. **Small-scale model** (e.g., interacting ice floes ≈ 10 to 10^2 km^2)
   - **GOAL:** Model interaction ice floe dynamics
     - Floe deformation & fracture, interaction between discrete floes
   - *Particle Level Sets (ParticLS):* A package for discrete element methods
     - Explicit timestep, $n \log(n)$ cost per timestep (n particles)
     - Level-sets define particle geometry
     - Multiple contact laws (e.g., viscoelastic contact, cohesive beams)
     - Computationally expensive—not feasible at macro-scale

Macro-scale equations (**constant** ice thickness)

- Compute the ice **concentration** c and **velocity** $\boldsymbol{u}$

Concentration equation (mass balance)

$$\frac{\partial c}{\partial t} + \underbrace{\nabla \cdot (c\boldsymbol{u})}_{\text{Advection}} = \underbrace{Q_c(c, \boldsymbol{u})}_{\text{Freeze/melt}}$$

influx conditions: $c(\boldsymbol{x}, t) = B(\boldsymbol{x}, t)$ at $\boldsymbol{x} \in \{\boldsymbol{x} \in \Gamma : \boldsymbol{u}(\boldsymbol{x}, t) \cdot \boldsymbol{n}(\boldsymbol{x}) < 0\}$

Momentum balance equation

$$\underbrace{M(c)}_{\text{Mass}} \left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} \right) = \underbrace{\boldsymbol{F}(c, \boldsymbol{u})}_{\text{Force}}$$

- Can we model **macro-scale concentration dynamics**?

Macro-scale model: numerics

- Time discretization: **Crank-Nicholson time integration** $\theta = 0.5$
  - $c_{n+1/2} = (1-\theta)c_{n+1} + \theta c_n$
  - $\boldsymbol{u}_{n+1/2} = (1-\theta)\boldsymbol{u}_{n+1} + \theta \boldsymbol{u}_n$

$$\frac{c_{n+1} - c_n}{\Delta t} + \nabla \cdot \left(c_{n+1/2}\boldsymbol{u}_{n+1/2}\right) = Q_c(c_{n+1/2}, \boldsymbol{u}_{n+1/2})$$

- Given a guess for unknown terms let
  - $c_{n+1} = \bar{c} + \Delta c$
  - $\boldsymbol{u}_{n+1} = \bar{\boldsymbol{u}} + \boldsymbol{\Delta u}$
- Linearize the equations around $(1-\theta)\bar{c} + \theta c_n$ and $(1-\theta)\bar{\boldsymbol{u}} + \theta \boldsymbol{u}_n$
- Apply **discontinuous Galerkin (DG) finite element methods**
  - DG is necessary to because the equations are advection-dominated
- Solve for Δc and $\boldsymbol{\Delta u}$
- Update our guesses $\bar{c} \leftarrow \bar{c} + \Delta c$ and $\bar{\boldsymbol{u}} \leftarrow \bar{\boldsymbol{u}} + \boldsymbol{\Delta u}$
- Repeat until converged (**Newton's method**)

Connecting the macro- and small- scale models

Momentum balance equation

$$M(c)\left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\right) = \boldsymbol{F}(c,\boldsymbol{u})$$

- Suppose

$$\boldsymbol{F}(c(\boldsymbol{x},t),\boldsymbol{u}(\boldsymbol{x},t)) = \lim_{\delta(\boldsymbol{x}),\tau\to 0}\int_{\delta(\boldsymbol{x})}\int_{\tau}\eta(\boldsymbol{x},\boldsymbol{x}',t-t')\,dt'\,d\boldsymbol{x}'$$

- $\eta(\boldsymbol{x},\boldsymbol{x}',t-t')$ is the **force density**
- The force is the average force acting on particles in a simulation with domain $\delta(\boldsymbol{x})$ and time horizon τ
- After discretizing the macro-scale equations, δ are the elements of a finite element mesh and τ is the macro-scale timestep size
- Similar to the **peridynamic equations at the macro-scale**

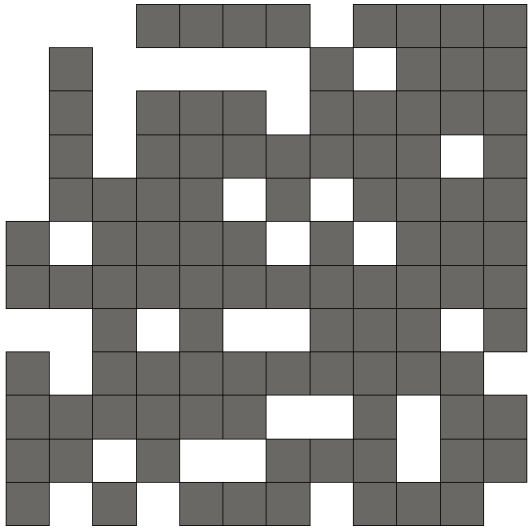
Silling et al. (many papers)

Small-scale model

Momentum balance equation

$$M(c)\left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\right) = \lim_{\delta(\boldsymbol{x}),\tau\to 0}\int_{\delta(\boldsymbol{x})}\int_{\tau}\boldsymbol{\eta}(\boldsymbol{x},\boldsymbol{x}',t-t')\,dt'\,d\boldsymbol{x}'$$

- Finite element method requires us to **integrate over each element**
  - Requires some quadrature rule
  - Evaluate the right hand side at each quadrature point
- Given c and $\boldsymbol{u}$ at the quadrature point, initialize a DEM simulation



- Run the small-scale DEM and return the average force
- Mesh refinement and $\Delta t \to 0$ compute the limit

Small-scale model result

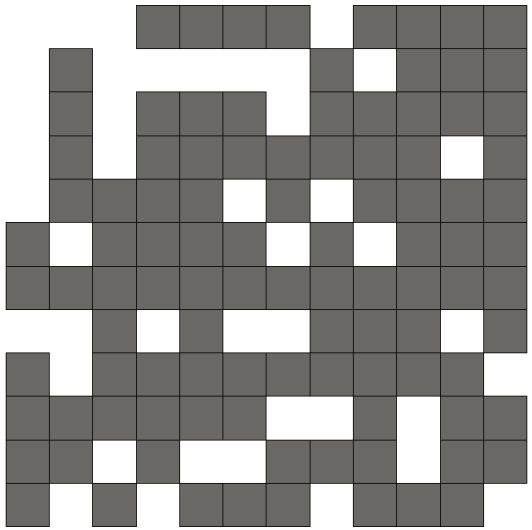


Figure: $t = 0$

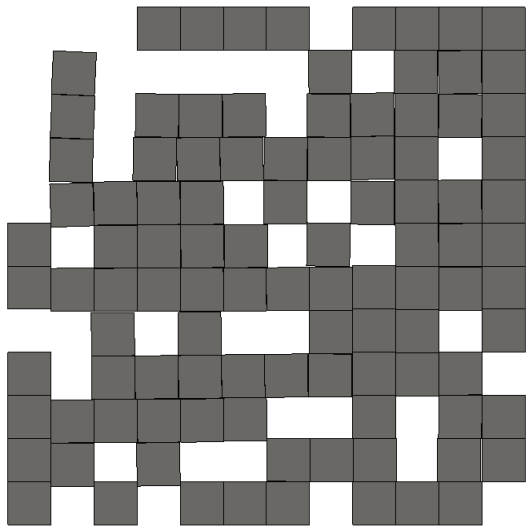


Figure: $t = 1$

- After one macro-scale time step $\Delta t = 10^{-3}$, there is minimal change
- The average force informs the macro-scale model
- We run these **independent** DEM models in parallel
- We require **fewer particles per simulation**—the scale is already small

Two scale model initial results

Figure: $t = 0$

Figure: $t = 2 \times 10^{-2}$

- Use *deal.ii* to implement the finite element level at the macro scale
- Use *ParticLS* to implement the discrete element model at the small-scale
- We successfully move an ice front forward after a small number of timesteps
- This serves as in initial proof-of-concept (very new results)

Current & future work

Currently:

- Developing a software framework called *Super Parameterization sea ICE (SPICE)* that combines the macro- and small- scale models
- Developing **surrogate models** for the energy balance right hand side
  - Gaussian processes
  - Neural networks

Current & future work

Currently:

- Developing a software framework called *Super Parameterization sea ICE (SPICE)* that combines the macro- and small- scale models
- Developing **surrogate models** for the energy balance right hand side
  - Gaussian processes
  - Neural networks

Future:

- Measure **forward uncertainty quantification** using stochastic parameters
- Combine our model with data using **data assimilation and Bayesian inference**
- Using model within **optimal experimental design** to allocate limited resources
- Rigorously develop a **mathematical framework** that combine the macro- and small- scale models

Conclusions

- We have an **open-source software framework** that provides tools to model sea ice
- Initial results show the software is robust and can be combined with data and other models
- We have **formulated a two-scale model** that targets both macro- and small- scale dynamics
- We are currently implementing and calibrating this model