Towards Enabling Predictive Scale-Bridging Simulations through Active Learning

LDRD Project 20190005DR

Presenter: Jeff Haack (CCS-2)

PI: Tim Germann (T-1)
Co-PI: Daniel Livescu (CCS-2)
Co-PI: Hari Viswanathan (EES-16)

April 16, 2020

Our Team (PI: Tim Germann, T-1)

Hydrocarbon extraction application

- Hari Viswanathan (EES-16), co-PI
- Qinjun Kang (EES-16)
- *Yu Chen (EES-16)
- *Soyoun Son (EES-17)
- *Animesh Agarwal (T-6)
- *Mohamed Mehana (EES-16)

ICF application

- Daniel Livescu (CCS-2), co-PI
- Jeff Haack (CCS-2)
- Brett Keenan (XCP-6)
- Irina Sagert (CCS-2)
- *Abdourahmane Diaw (CCS-2)
- +Austin Gilbert (CCS-2)

Machine learning

- Kipton Barros (T-1), task lead
- Nick Lubbers (CCS-3)
- *David Rosenberger (T-1/T-CNLS)
- *Kun Wang (T-3/T-CNLS)
- +David Sanchez-Alcantarilla (T-1/T-CNLS)
- +Aleksandra Pachalieva (T-CNLS)

Computer science

- Christoph Junghans (CCS-7), task lead
- Satish Karra (EES-16)
- Ying Wai Li (CCS-7)
- Mike McKerns (CCS-3)
- Robert Pavel (CCS-7)

*Postdoc +Student

Outline of Talk

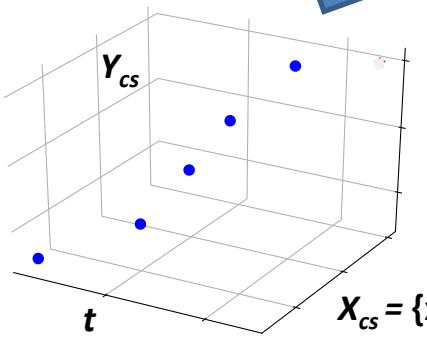
- Overview of the project
- Active Learning approach
- Mixing and transport in Inertial Confinement Fusion simulations
- Computer science infrastructure challenges
- Preliminary numerical results

Conventional (sequential) upscaling

Experiments

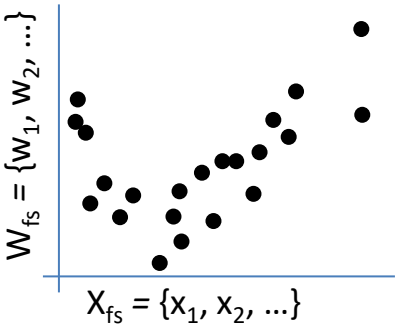
Theory

Simulations



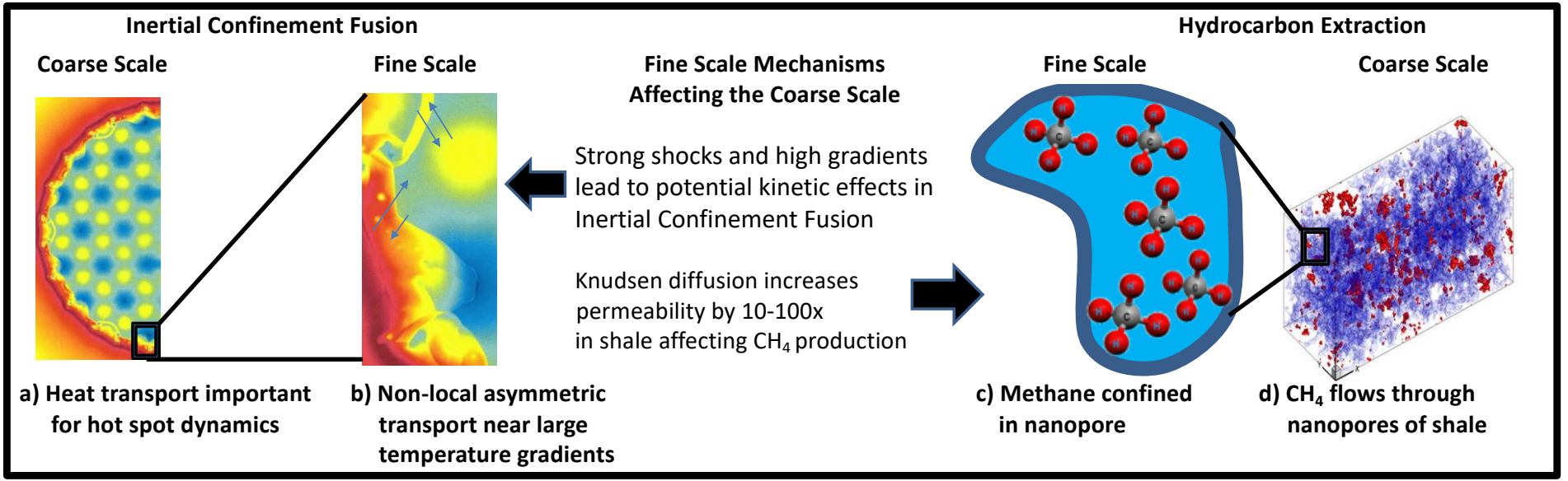
$X_{cs} = \{x_1, x_2, \ldots; w_1, w_2, \ldots\}$

Coarse Scale (CS) Model



Fine Scale (FS) Model

Examples: including molecular-scale effects in continuum-scale models

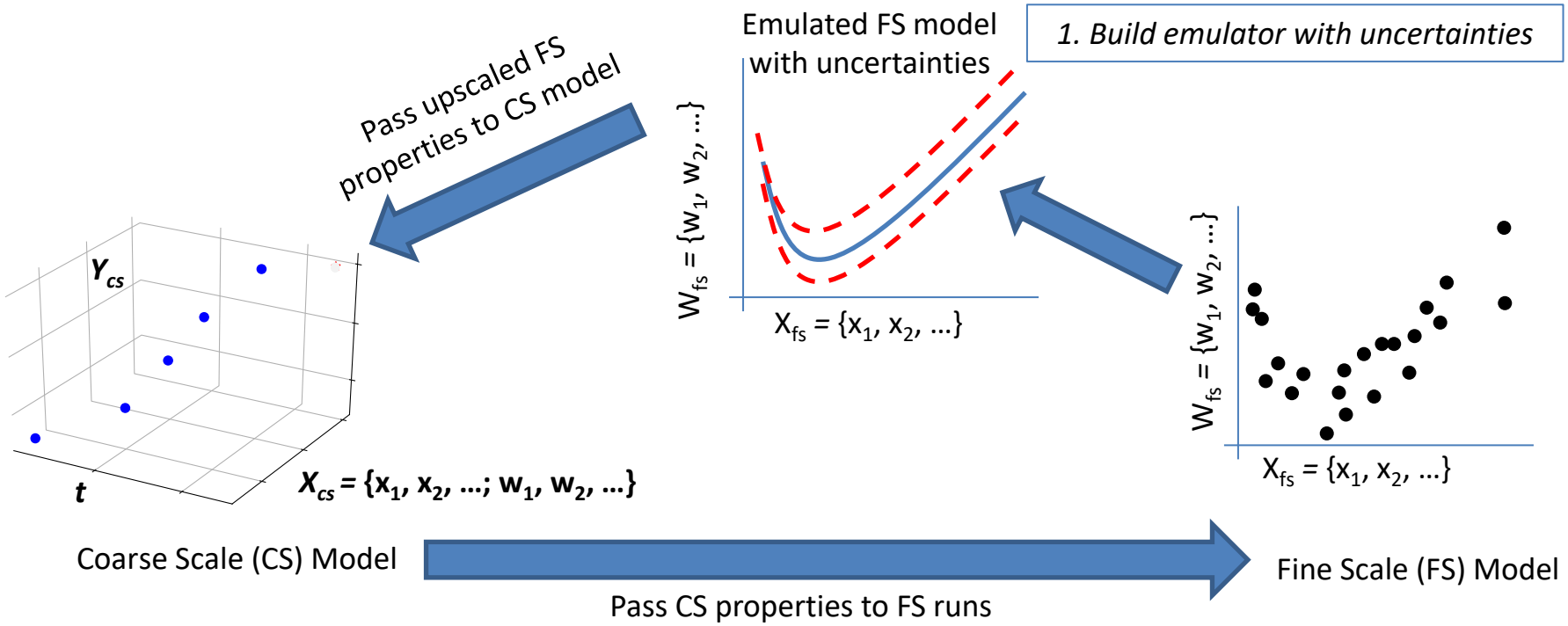


Other ICF issues that can be addressed

- Sparse data on multicomponent mixtures, ad hoc mixing rules
- Hydro needs smooth data to avoid spurious waves
- Hydro can break down e.g in shocks – spawn kinetic regions?
- Kinetic models parameter unclear moderate coupling regimes

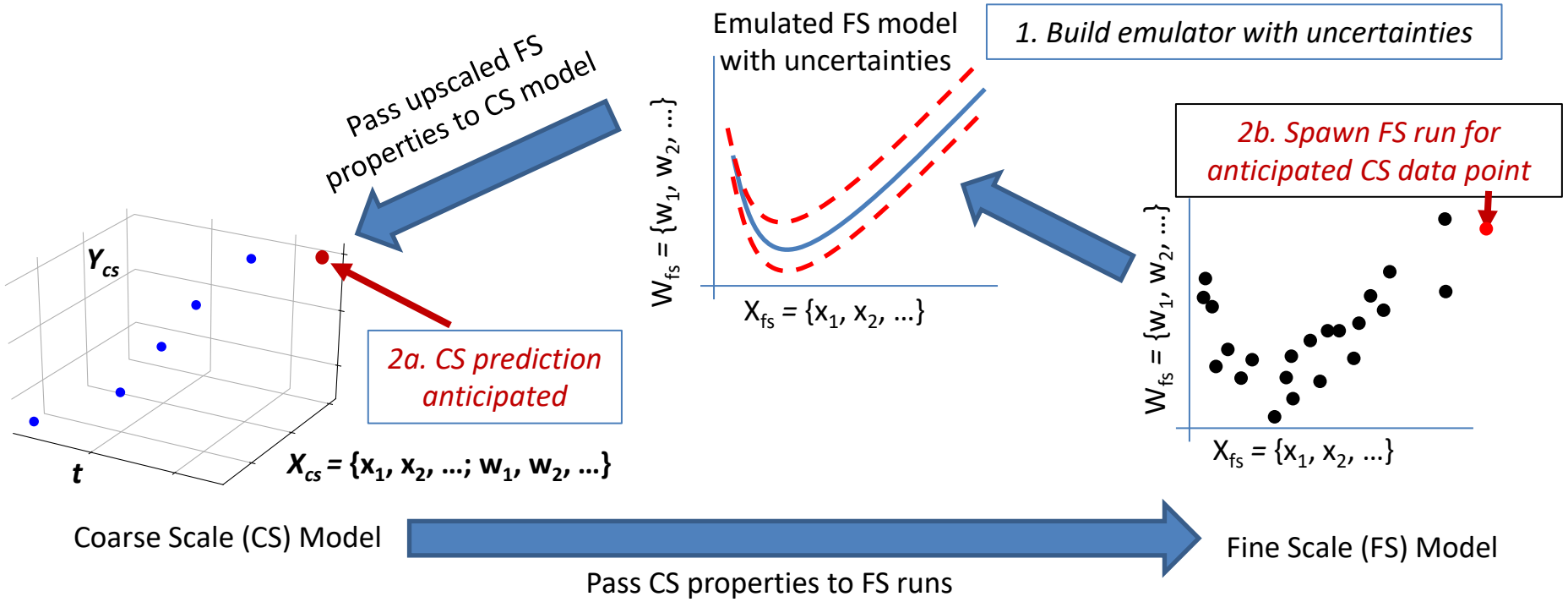
Our Active Learning strategy consists of 3 parts

1. **Construct a surrogate model to predict both upscaled parameters *and* uncertainties**



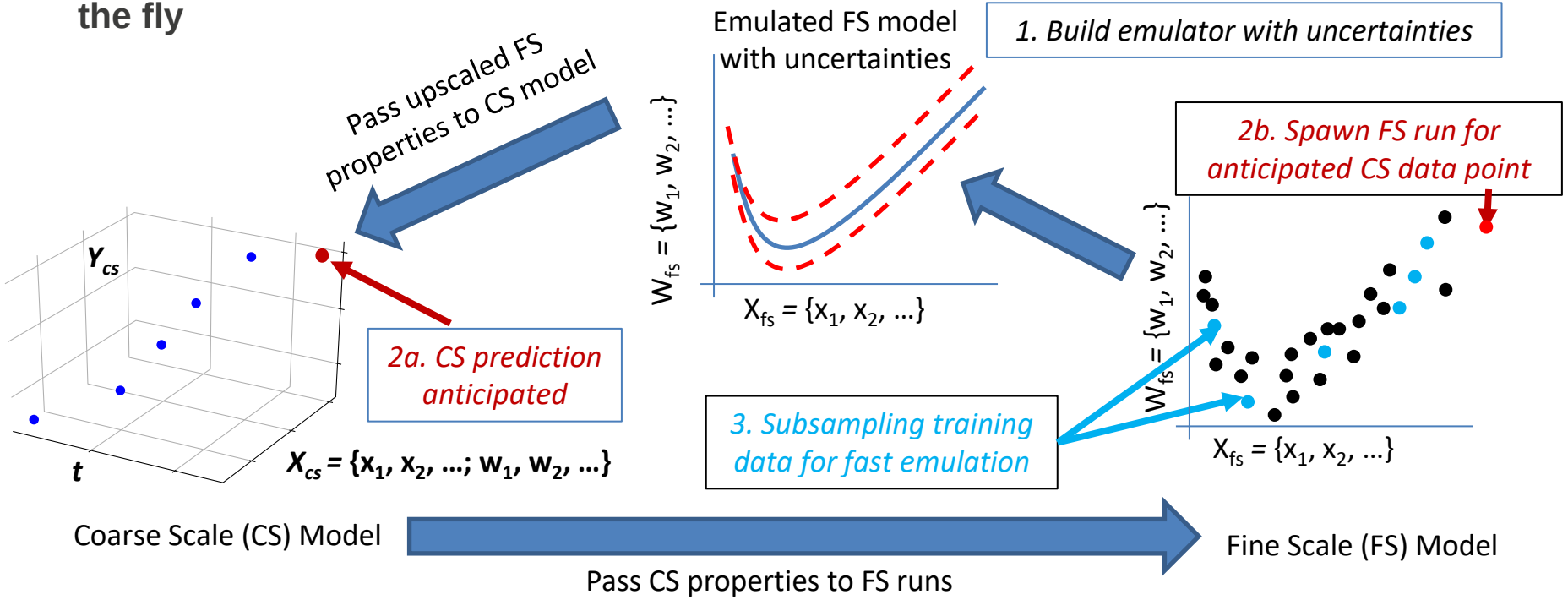
Our Active Learning strategy consists of 3 parts

1. **Construct a surrogate model to predict both upscaled parameters *and* uncertainties**
2. **Anticipate the data to be required by the coarse scale (CS) simulation**



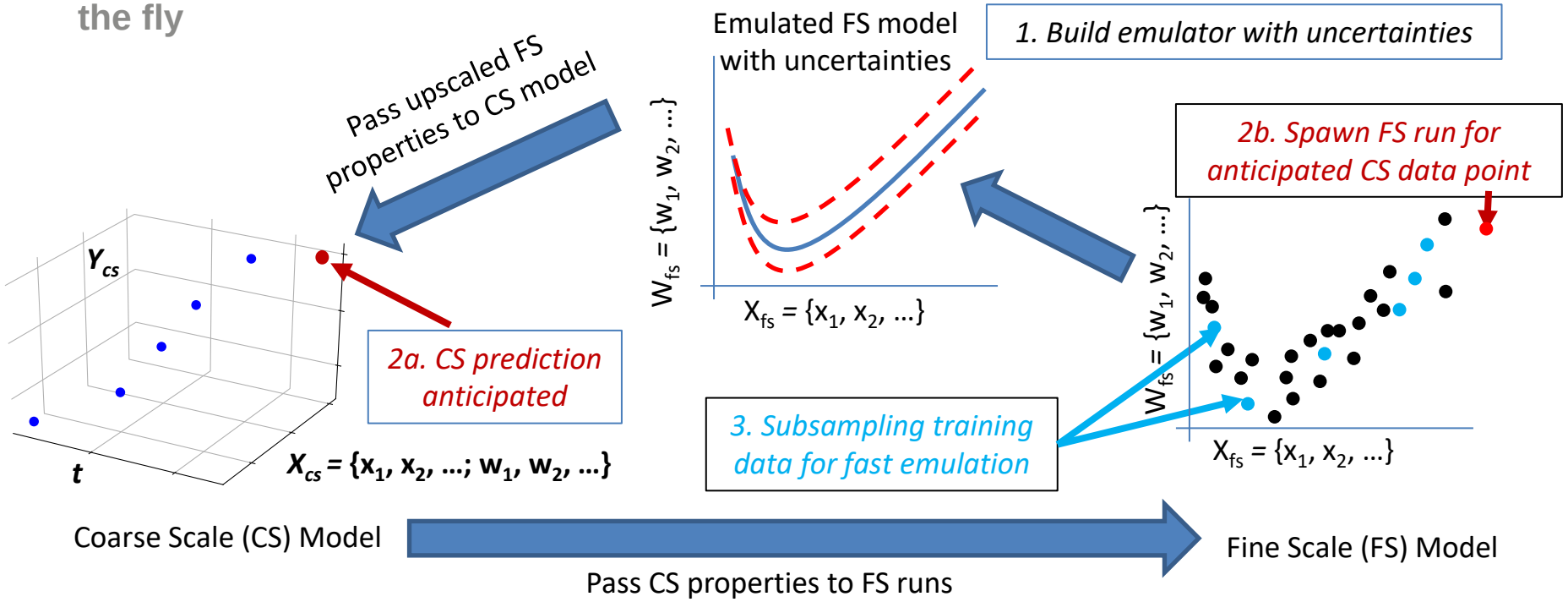
Our Active Learning strategy consists of 3 parts

1. **Construct a surrogate model to predict both upscaled parameters *and* uncertainties**
2. **Anticipate the data to be required by the coarse scale (CS) simulation**
3. **Dynamically update surrogate models with new fine scale (FS) data generated on the fly**



Active Learning approaches: mid-project status

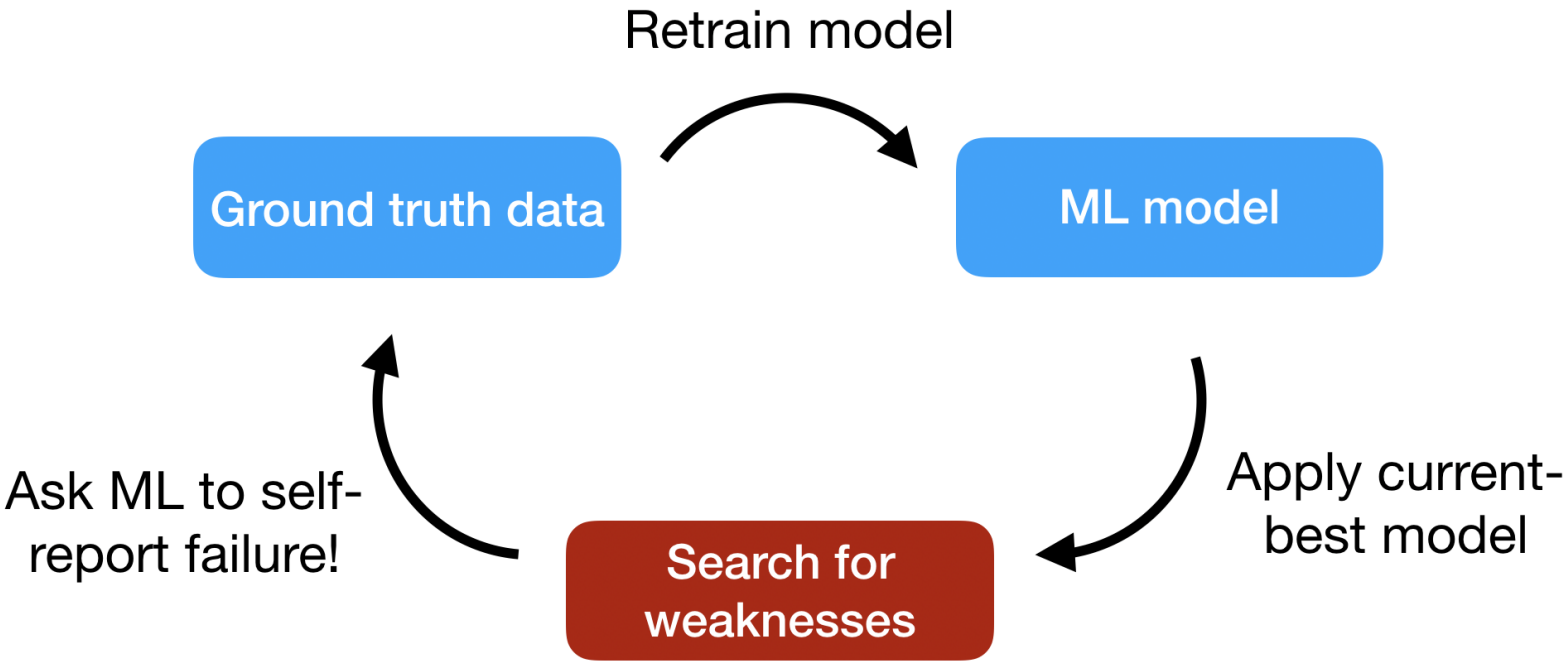
1. **Construct a surrogate model to predict both upscaled parameters *and* uncertainties**
2. **Anticipate the data to be required by the coarse scale (CS) simulation**
3. **Dynamically update surrogate models with new fine scale (FS) data generated on the fly**



Machine Learning Details

Active learning approach – quick sketch

ML assists in collecting ***good data***!



Machine Learning: Current implementation

- We use Deep Neural Nets
  - Parametric models scale better for large data
  - Unlike e.g. Kriging/GPR
    - Used in a previous scale-bridging project here
  - Can also bootstrap these to provide some variance

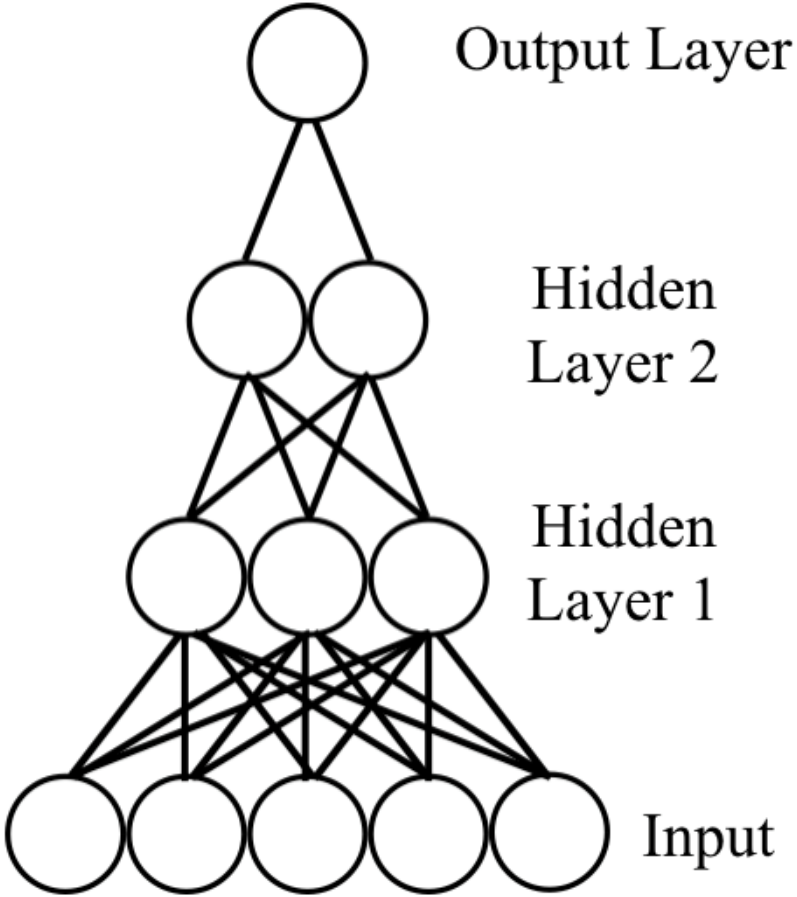
- We also need a sense of the uncertainty when we query
  - Go get new data if we decide it is bad
  - Retrain the model as we get new data points

Quick ’n’ dirty neural net intro

- A neural net (NN) is a layered network of neurons
  - Each neuron has TBD weight parameters

$$x' = f(W \cdot x + b)$$

  - Find parameters by minimizing loss function over training data
  - Run batches through 50 at a time (an epoch)
  - Usually takes a few hundred epochs to converge
  - Implemented in PyTorch



How good is our model?

- UQ currently provided by ensemble of Neural Nets
  - "Surrogate modeling by committee"
  - We return the average prediction of all the models
  - Ensemble variance = model uncertainty
  - Neural net parameters (depth) chosen to encourage disagreement
  - Many other approaches exist but this is our first cut

UQ details

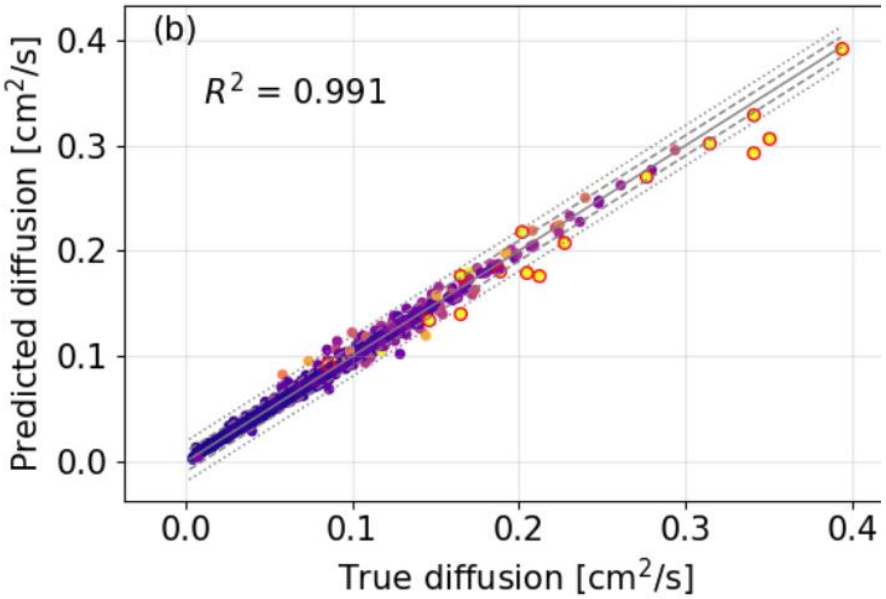
- For ensemble building
  - Withhold a different random 10% as validation data for each committee member – each produces a different model

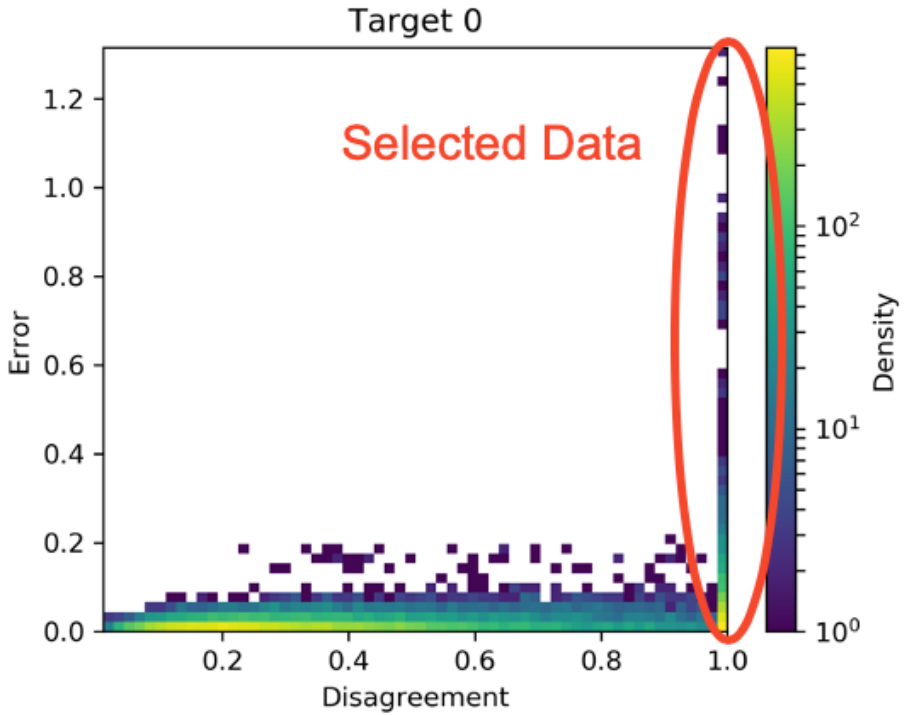
  Reject models with $R^2 < 0.7$

- Estimating the uncertainty
  - Calculate the variance of the predictions of the ‘committee members’
  - Calibrate vs the individual model errors and variance of the full model
  - Set a disagreement threshold for when one should go get new data
    - This is tunable

Example: predicting interdiffusion

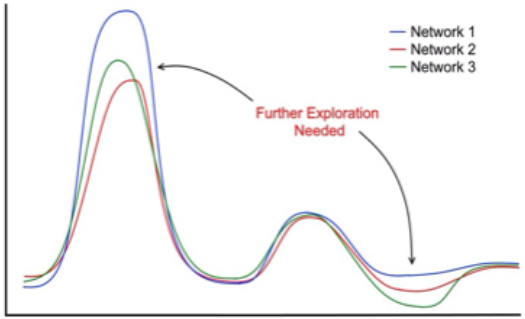
- Plot of entire data set vs the model
- Red circles = disagreement tells us that we need to run MD





Approach two – next steps

1. Ensemble variance



2. Direct prediction of error

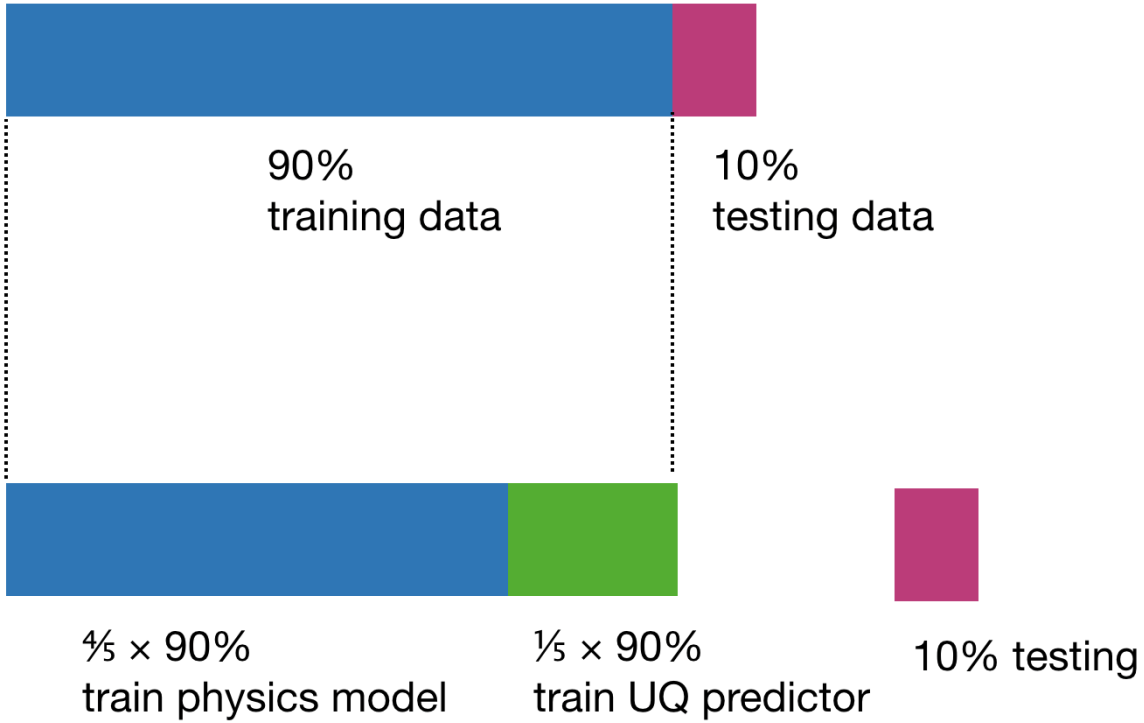
$$x \mapsto \hat{y} \pm \delta\hat{y}$$

neural net 1

neural net 2

Active learning approach – eventual plan

We are trying to learn the model and its error bars at the same time



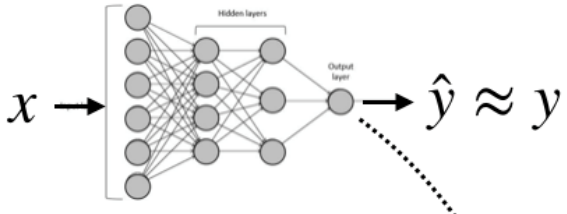
$$\hat{y}(x) \approx y \qquad \delta\hat{y}(x) \approx |\hat{y} - y|$$

Learning error bars

Idea: transfer learning

Step 1. Train $\hat{y} \approx y$

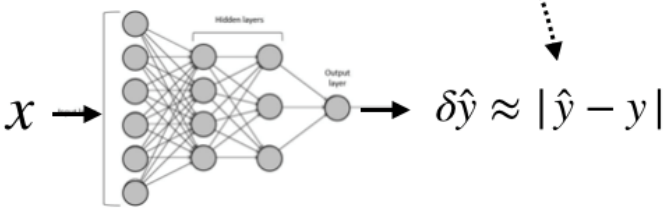
using data 1



Step 2. Copy weights

Step 3. Train $\hat{\delta y}$

using data 2



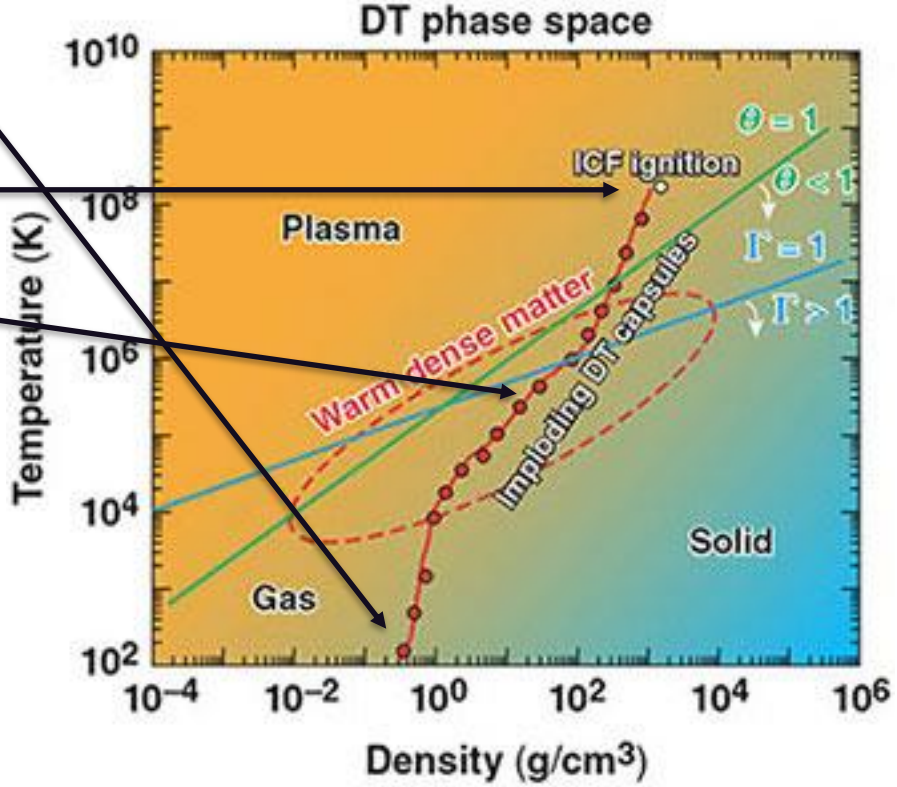
Future stuff: MD

- Other UQ definitions can be plugged in to the AL framework
  - Neural net for error bars (above)
  - Mystic UQ framework
- Could incorporate error bars from the MD results as well
- Use uncertainty to set up a priority-based MD
  - not just a yes or no question whether to run MD!

Next: Applying this to an ICF problem

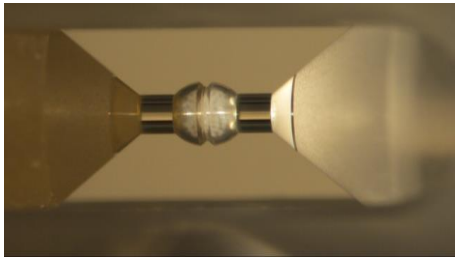
Motivation: Inertial Confinement Fusion

- ICF experiments span many orders of magnitude
  - Low T, high dens: Solid Mechanics
  - High T: Plasma physics
  - ”Warm” T: ????
- Mixing during warm phase can have huge effects on hot phase
  - How we combine material properties is often done in an ad-hoc fashion
- Shock structure important – kinetic upscaling?

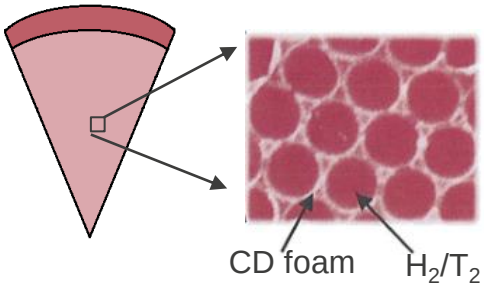


Motivating experiment: MARBLE

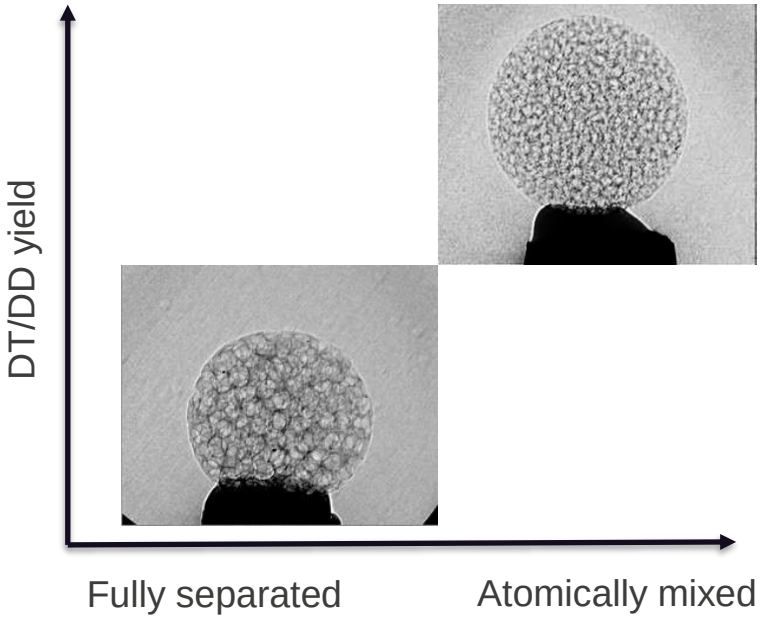
NIF capsule size~ 2mm



Si-doped CH ablator filled with deuterated foam



Engineered foams:
- 0.1-1μm pores for baseline
- 30-100μm macro pores



- MARBLE is an experimental implosion campaign using engineered foams with prescribed pore sizes to study the impact of mix on a burning plasma.
- Complex flow and problem size require continuum simulations, but description breaks near sharp gradients (e.g. shock waves, material interfaces) -> inaccurate predictions.
- Is preheat washing out the pores?

Multiscale connections

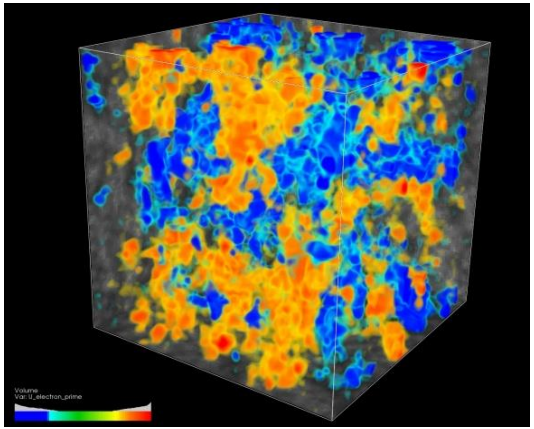
Three opportunities for multi-scale connections

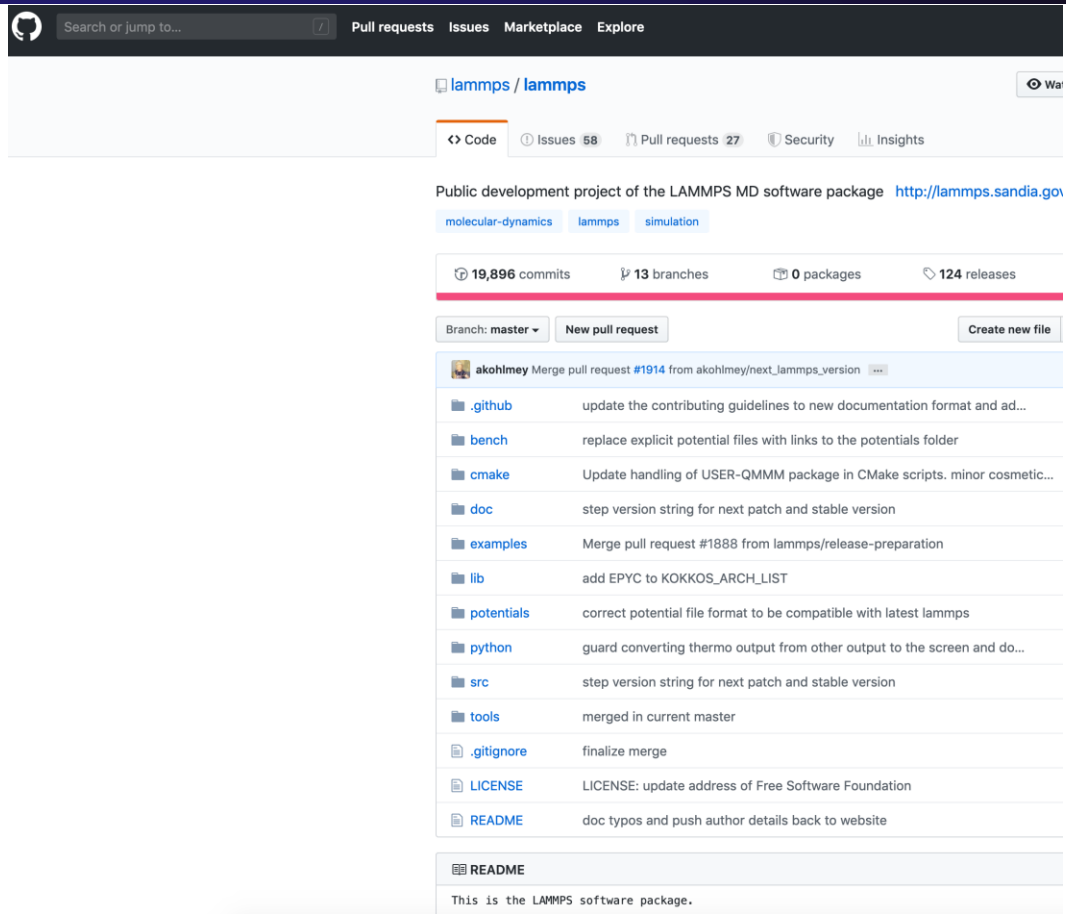
- Hydro and Molecular Dynamics
  - Get diffusion coefficients, viscosity, conductivity
  - Affects how turbulence/instabilities develop
- Hydro and Kinetic Theory
  - Hydro models break down in shocks
  - Get the correct shock width = better shock yield predictions
- Kinetic Theory and Molecular dynamics
  - Get accurate collision rates in warm dense matter regime
  - Current models only valid in weak coupling regime

We will use the active learning framework to link three existing codes

Atomistic description: LAMMPS

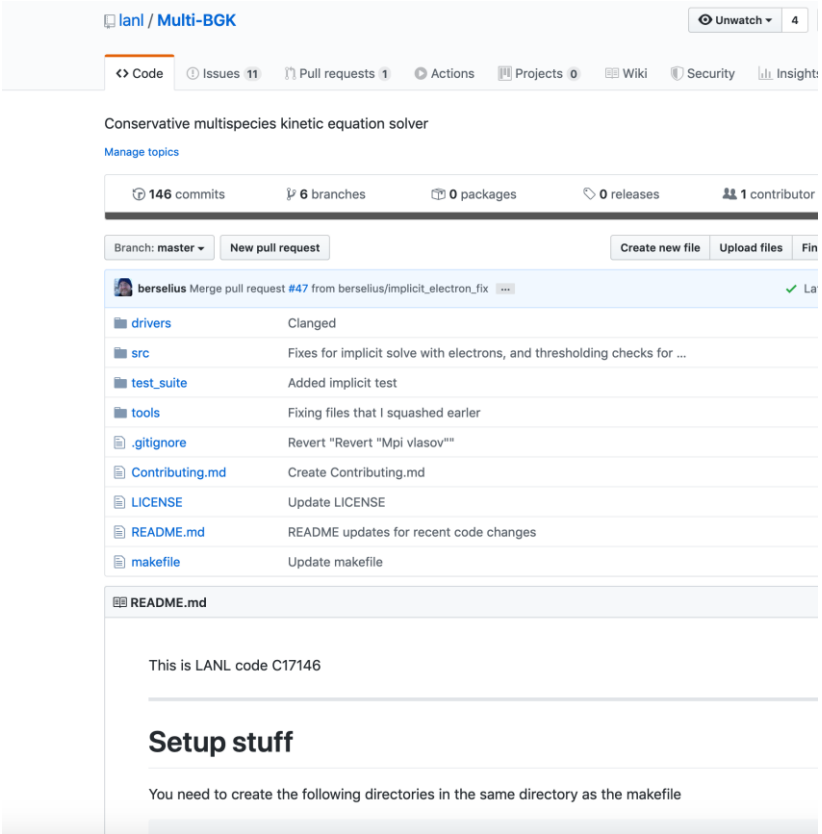
- Massively parallel molecular dynamics code
- Open sourced, on github
- Primarily developed at Sandia





Kinetic description: Multi-BGK

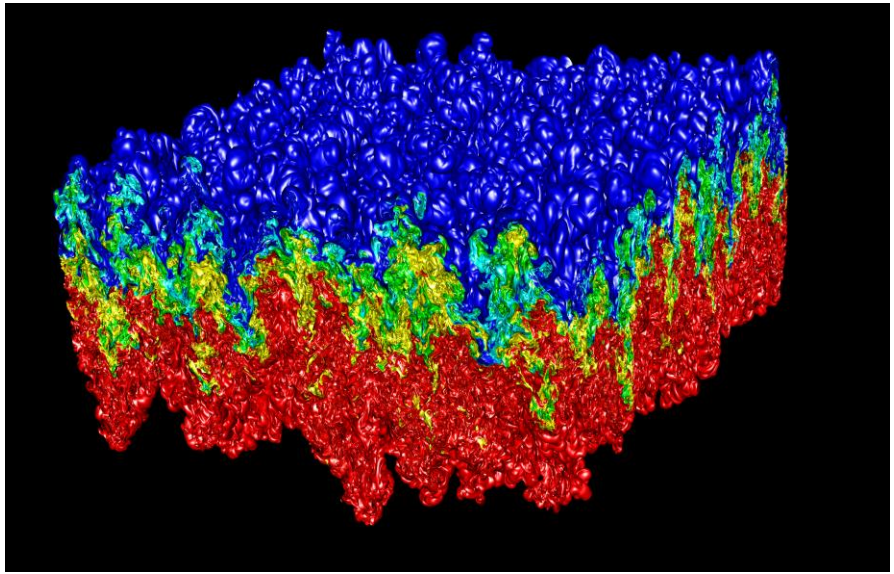
- Developed at LANL
- Open sourced, can be found at github.com/lanl/Multi-BGK
- Scalable using OpenMP + MPI
- Geared towards WDM regime
  - Partial ionization
  - Degnerate electric fields
- The model when hydrodynamics breaks down
  - Shocks
  - Interfaces
  - Low density/high temperature



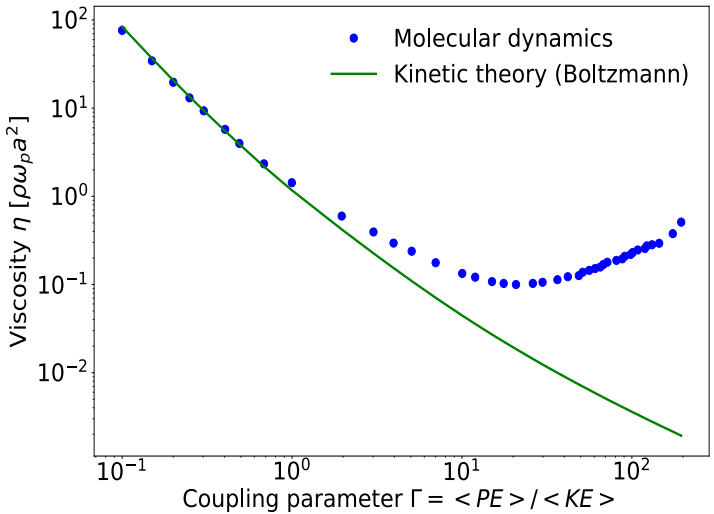
Continuum description: multispecies Navier-Stokes solver

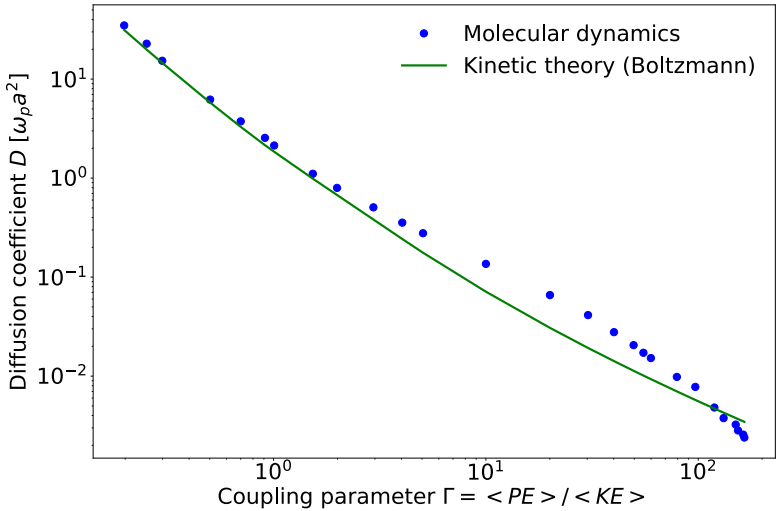
- Code used: CFDNS (Livescu et al LA-CC-09-100).
- FFTs for triply periodic problems, mixed FFTs - 6th order compact finite differences for inhomogeneous problems.
- Three-dimensional domain decomposition for parallelization
- Code was used to generate some of the largest turbulence simulations to date on
  - 2-D simulations (up to 16,384^2) of Rayleigh-Taylor instability (Trinity)
  - 3-D simulations (up to 4096^3) of RTI, shock-turbulence interaction, mixing layers, etc. performed on Sequoia, LLNL; Trinity, LANL; MIRA, ANL up to 1,500,000 compute cores.
- Other features
  - Coupled to Stanton-Murillo transport model for weak coupling

Needs transport information in WDM regime



Unlike existing theories, MD provide rates for plasma across coupling regimes



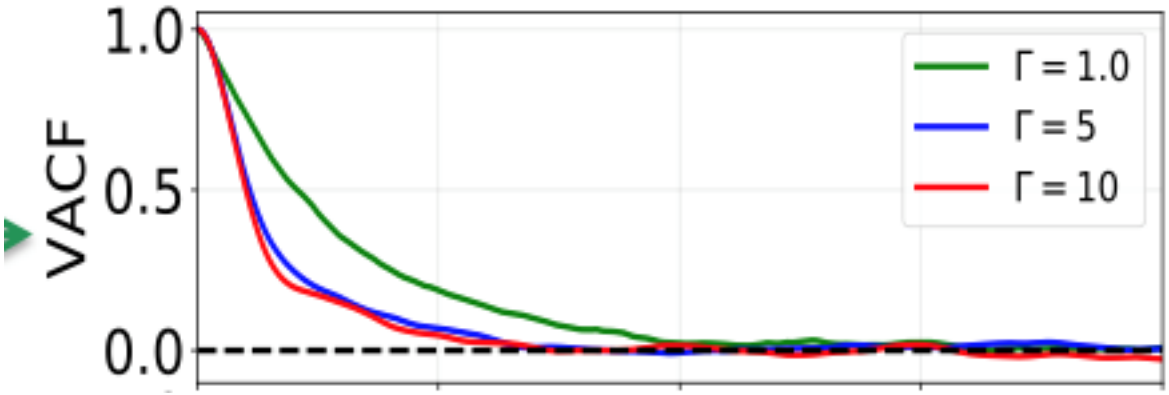


KT models do not properly describe the rates for strongly coupled systems while MD does.

We bridge this gap by coupling MD directly to the coarse-grained framework using HMM.

MD-Fluid hybrid approach

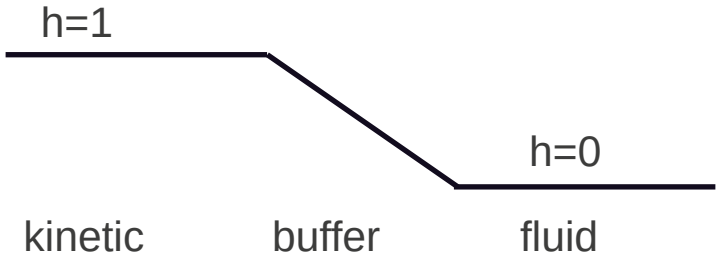
- What hydro needs is the **transport coefficients**
  - Diffusion, viscosity, thermal conductivity
  - A function of all species in the cell
  - Compute these with MD via Green-Kubo



- Also exploring doing this with EOS, but the above was our initial focus
- This is still WIP due to FORTRAN….fun…. but close

Kinetic-Fluid Hybrid Approach

- Introduce a *buffer zone* for a gradual transition from the kinetic to the fluid model, using a transition function *h* that goes from 1 in the kinetic region to 0 in the fluid region.
- No boundary conditions are needed at the edges of the buffer zone.
- Single species idea: Degond-Jin-Mieussens JCP 2005



Start with $\partial_t f + v \cdot \nabla_x f = \frac{1}{\varepsilon} Q(f)$ and decompose f:

Domain Decomposition

$$f_F = (1-h)f, f_K = hf$$

$$(\partial_t + hv \cdot \nabla_x) f_K + hv \cdot \nabla_x E(\Omega) = \frac{h}{\varepsilon} Q(f_K + E(\Omega))$$

$$\partial_t \Omega + (1-h)\nabla_x F + (1-h)hv \cdot \nabla_x \langle vmf_K \rangle = 0$$

- solve the fluid model in the fluid domain,
- solve the kinetic model in the kinetic domain,
- solve the sum of both in the buffer region.

Micro-Macro Decomposition

$$f = M + g, g_F = (1-h)g, g_K = hg$$

Resulting coupled system

$$(\partial_t + hv \cdot \nabla_x) g_K = \frac{h}{\varepsilon} Q(g_K + M) - h(\partial_t + v \cdot \nabla_x) M$$

$$\partial_t \Omega + \nabla_x F + \nabla_x \langle vmg_K \rangle = 0$$

- solve the fluid model in the entire domain,
- localized kinetic upscaling that corrects the fluid model wherever necessary.

MD-Kinetic hybrid approach

Kinetic theory models non-equilibrium physics

Connection to microscopic physics: collision rate models

- Fokker-Planck / Boltzmann
  - Expensive
  - Parameters built on weak coupling assumption
- Bhatnagar-Gross-Krook (BGK) model
  - Simple to compute
  - Extends well to multispecies
  - Only valid for near-equilibrium situations

$$\left.\frac{\partial f}{\partial t}\right|_{\text{coll}} = \nu\left(\mathcal{M}[f] - f\right)$$

MD-Kinetic hybrid approach

- The microscopic physics is contained in the collision rates ν_{ij}
- These are still a free parameter at this point that we need to constrain
- A Chapman-Enskog analysis of the Multi-BGK model gives a relationship between diffusion and collision rates

$$D_{12} = \frac{nT}{\rho^2} \frac{\rho_1 \nu_{12} + \rho_2 \nu_{21}}{\nu_{12} \nu_{21}}$$

(More than two species - matrix solve)

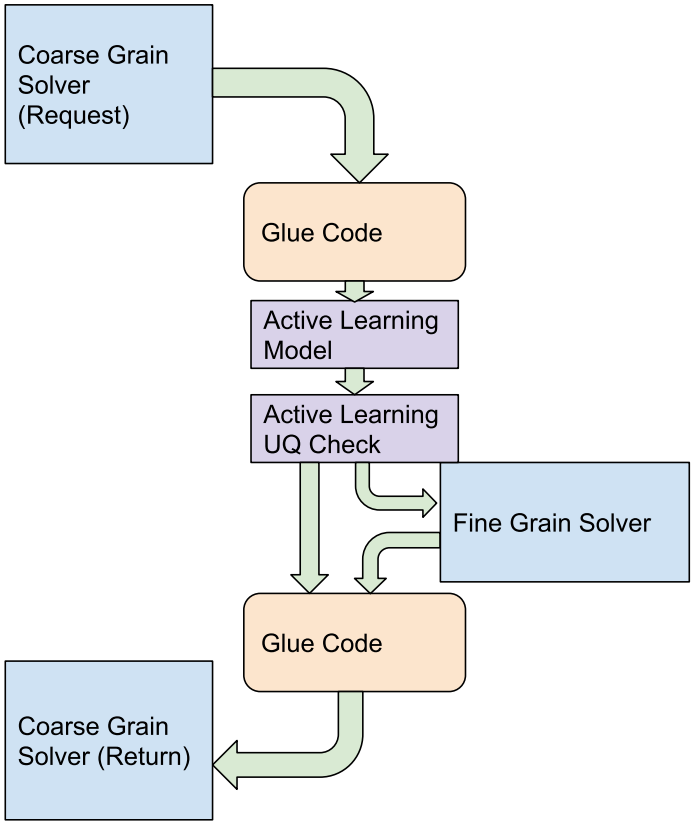
- For weakly coupled plasmas, theory for D_{ij} is mature
- For moderate to strongly coupled plasmas, we need high fidelity data from MD
- Bonus – can use same framework that we use for MD-DNS coupling

Building the infrastructure

How do we link these existing codes?

Development of an interface code to connect scales with machine learning:

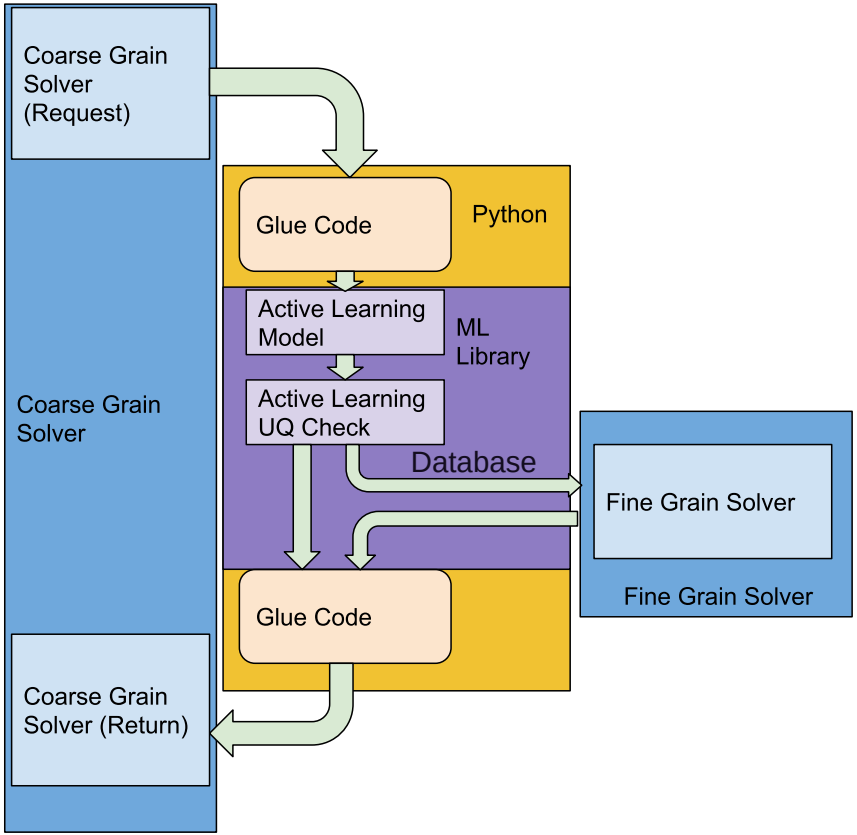
Generic Learning User Enablement (GLUE) Code



Implementation

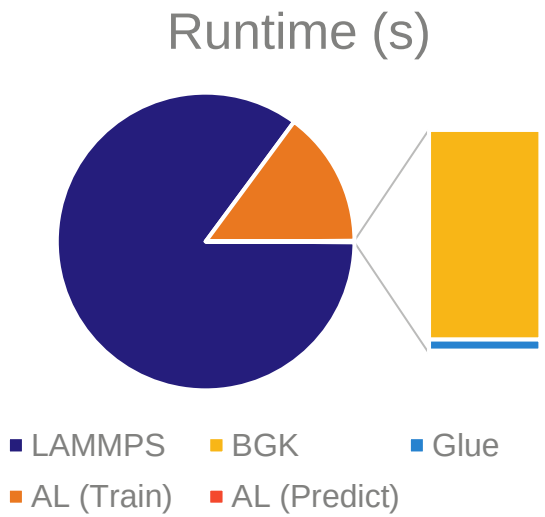
- Initial version based fully off commodity software
  - Templated C++ and ISO-C Bindings (for Fortran) to rapidly couple with existing solver codes and iterate on requirements
  - Python to interface with Slurm and Popular Machine Learning Frameworks
  - SQL database used for communication and resilience
    - Atomic communication between processes
    - Aggregation of “ground truth” training data
    - SQLite integrated in every modern Python distribution
- Proof of concept implementation meant to work on any (modern) platform
- Task-based design allows hot-swapping of backend implementations

Implementation Diagram – current workflow



Timing and Costs

- LAMMPS: ~7min 50s per call on a 36 core skylake
- Multi-BGK time per timestep: ~0.75s
- Glue and AL Time
  - 730 GNDs
    - Took 81.8s for 730 GNDs
    - Took 0.008s to predict
  - Time in Glue is 0.04s per request



Technical Challenges: Scheduling

- Job footprint changes drastically during the course of a run
  - Spawned Fine Scale Call
  - Re-training machine learning model
- Need to dynamically adjust resources
  - No point in idle nodes
- Will use some for generating training data to fill perceived gaps for ML
  - But still a very burst heavy load

Technical Challenges: Scheduling

- Initial Solution relies on separate slurm jobs for each task
  - Effective but highly inefficient due to wait time in queue
  - Vulnerable to system load
  - Bonus stress test: almost all open HPC was down the weekend we wanted to run the demo data…
- Next step is batching of tasks
  - Increases efficiency
  - Still vulnerable to system load
- Also investigating more HPC-oriented solutions and frameworks
  - LLNL's Flux seems designed for this kind of problem
  - Will need to balance efficiency versus job footprint
- Usage of persistent fine-scale scale task

Nuts + Bolts - Required Changes to BGK and CFDNS

- **Minimal (<200 LOC) Changes**
  - Modification to build system to link in glue code library
  - Additional option/code path to call glue code
  - Function calls to interface with glue code

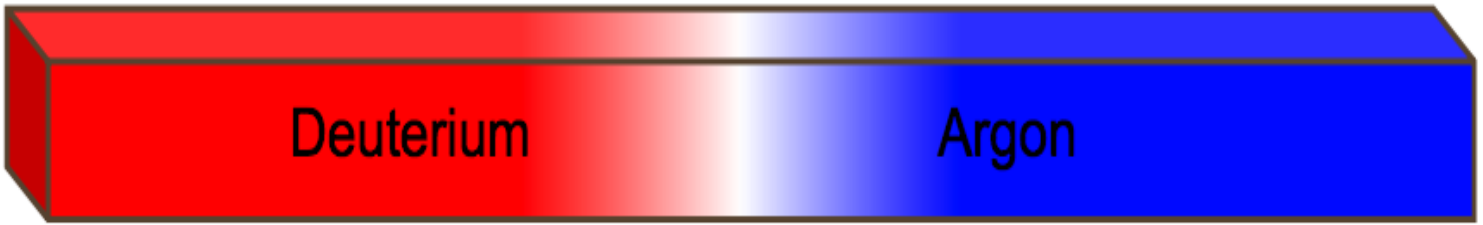
```
281     input.temperature = Tmix;
282
283     for(int sp=0; sp < nspec; sp++) {
284       input.density[sp] = n[sp];
285       input.charges[sp] = Z[sp];
286     }
287
288     output = bgk_req_single(input, 0, tag, db);
289
290     D_ij[0][0] = output.diffusionCoefficient[0];
291     D_ij[0][1] = output.diffusionCoefficient[1];
292     D_ij[0][2] = output.diffusionCoefficient[2];
293     D_ij[0][3] = output.diffusionCoefficient[3];
294     D_ij[1][1] = output.diffusionCoefficient[4];
295     D_ij[1][2] = output.diffusionCoefficient[5];
296     D_ij[1][3] = output.diffusionCoefficient[6];
297     D_ij[2][2] = output.diffusionCoefficient[7];
298     D_ij[2][3] = output.diffusionCoefficient[8];
299     D_ij[3][3] = output.diffusionCoefficient[9];
```

Prototype demonstration

Demo Problem: 1D atomic diffusion of two species

- Mixing of cold fuel and ablator materials into hot fuel can spoil fusion burning in HED experiments.
- Mixing can occur on disparate length and time scales
- MARBLE – will pores collapse due to atomic / hydro mix?
- We study impact of atomic scale mixing of a Deuterium/Argon interface across coupling.
- Each species seeded with 1% trace of the other

$$\text{Temperature} = 100 \text{ eV}$$
$$\text{Total density} = 10^{25} \ [1/cc]$$
$$\text{Domain size} = 100 \ \mu m$$
$$\text{Timestep} = 0.2 \text{ ps}$$



Results of the MD-Kinetic coupling

- **All-MD Run: 256 cells X 200 steps = up to 51200 MD Calls (actual ~10k)**

```
Initial condition setup complete
At time 0 of 2e-10
1 :0 is using values from LAMMPS
BGK request data for this cell: n: 4.91076e+24 7.2503e+22 0 0 Z: 0.896396 8.38153 0 0 T: 99.9211
Output returned from Glue code:  D[0]: 0.3268  D[1]: 0.0173107  D[2]: 0  D[3]: 0  D[4]: 0.0127414
1 :1 is using values from LAMMPS
BGK request data for this cell: n: 5.00429e+24 5.00202e+22 0 0 Z: 0.896396 8.38153 0 0 T: 100.073
Output returned from Glue code:  D[0]: 0.340595  D[1]: 0.022848  D[2]: 0  D[3]: 0  D[4]: 0.019672
```

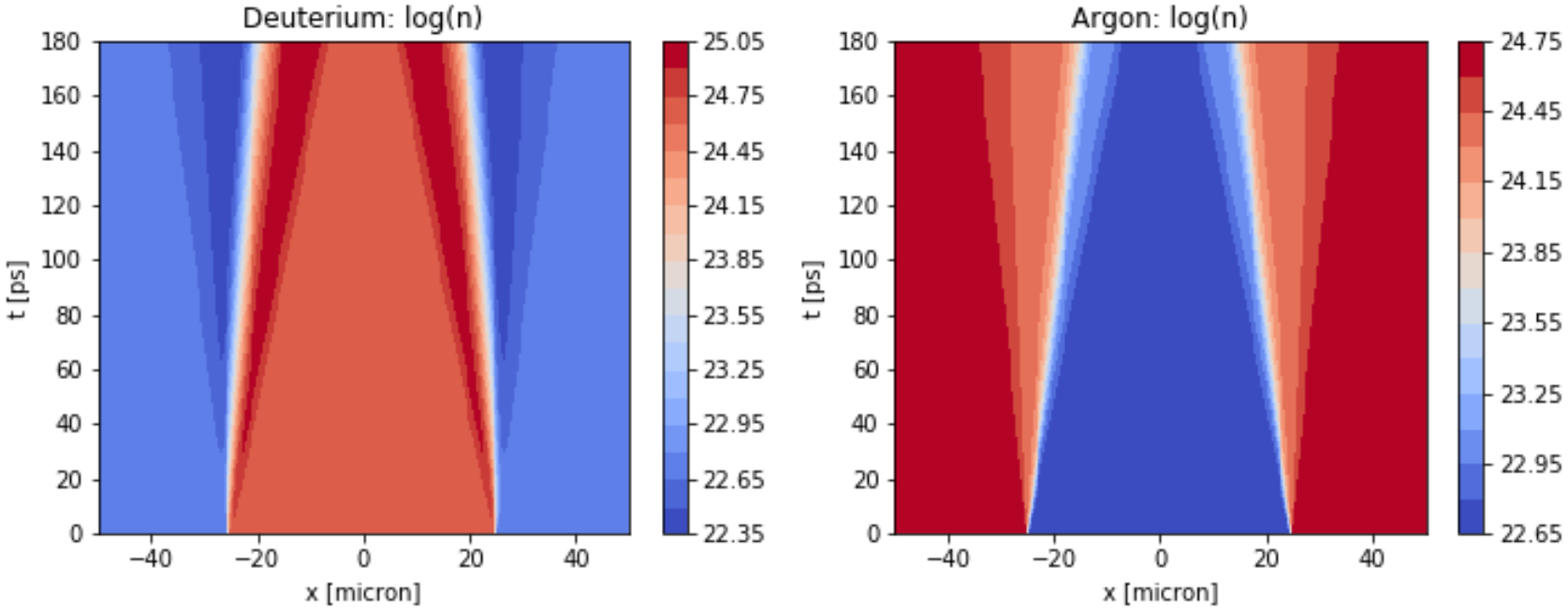
```
Loop time of 38.8306 on 36 procs for 100000 steps with 15277 atoms

95.8% CPU use with 36 MPI tasks x 1 OpenMP threads
```

- **ML Run for A: Compare with MD**
  - Seeded with 1000 random training points
- **ML Run for B: Temperature ramp**
  - Expect to dynamically spawn more MD at later time as it sees holes in data

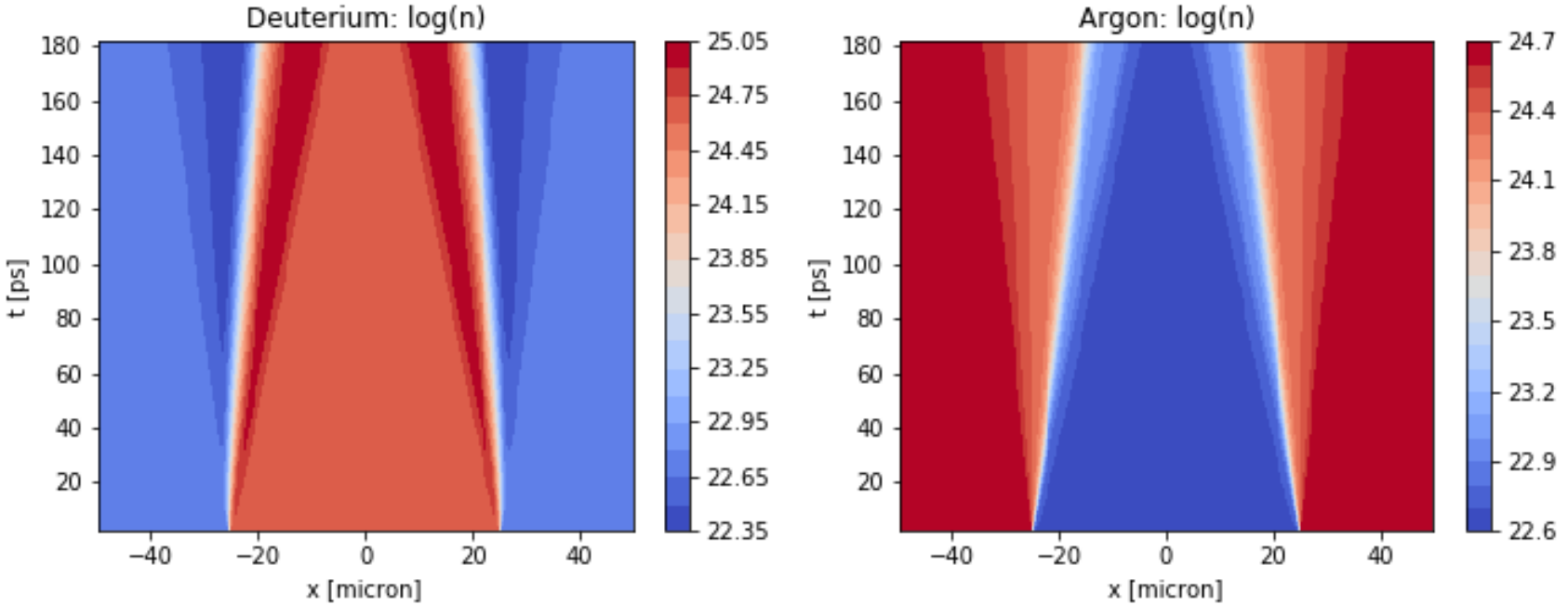
All-MD run: result

- Argon pushes in on Deuterium
- Atomic mixing layer forms at interface
- Expect some asymmetry due to different mass/charges
- ~10k MD calls = 333 node-hrs



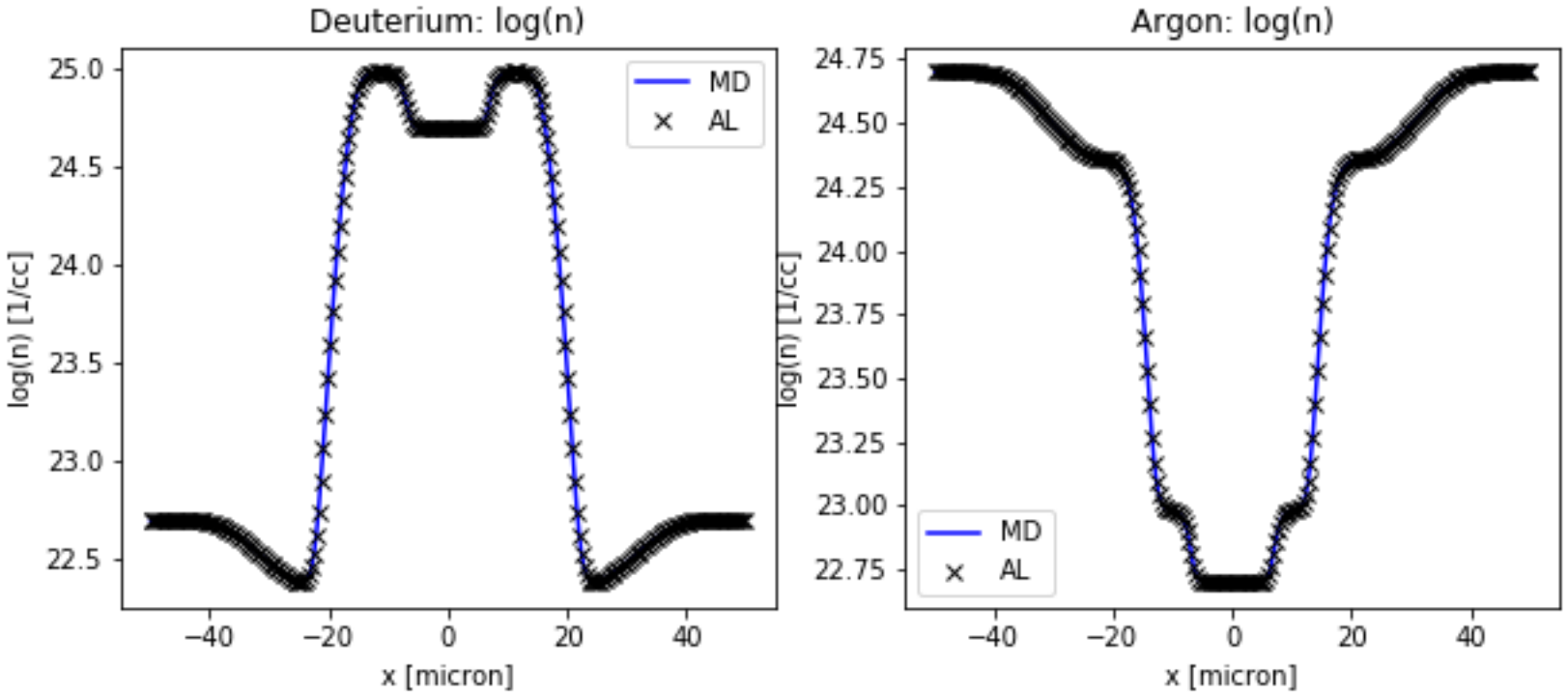
Solution with AL: case A

- Al trained with 1000 training points
- This stayed in the training data domain – used AL entirely
- ~45 minutes



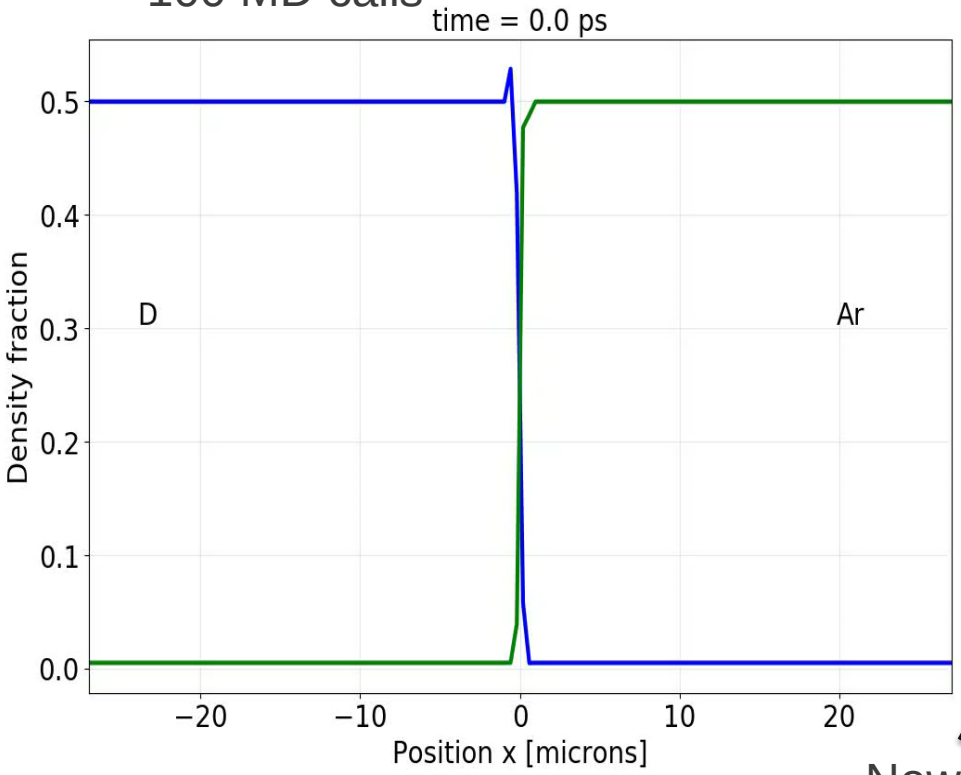
Solution with AL: comparison to MD run

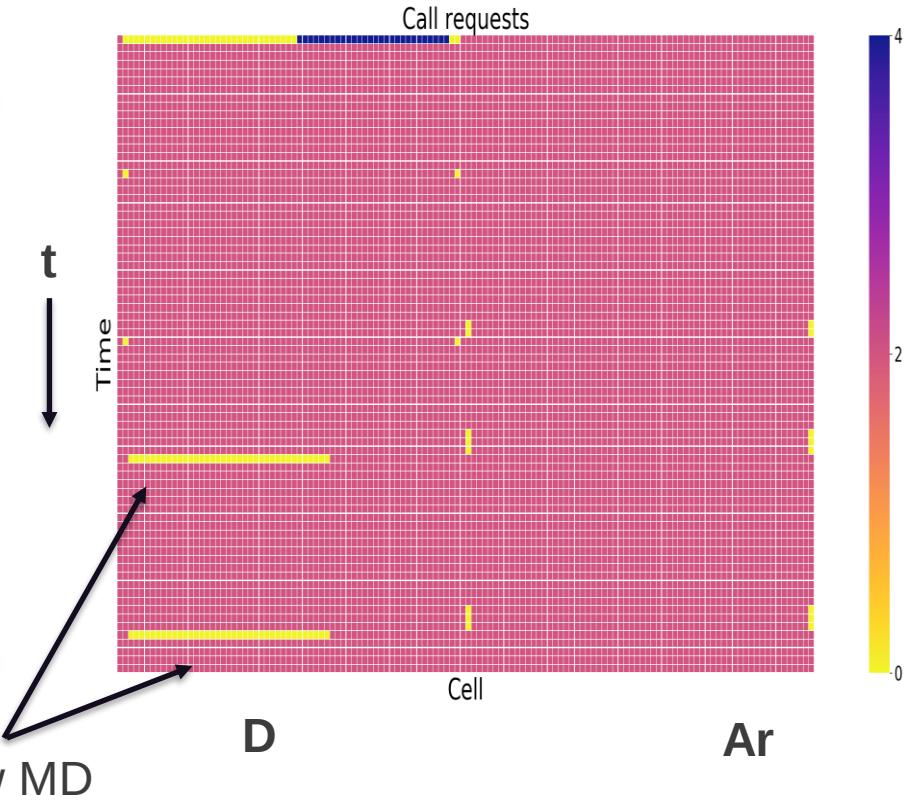
- Density profiles at final time
- Solution from learner matches MD very well



Case B: Stressing the ML to make new calls

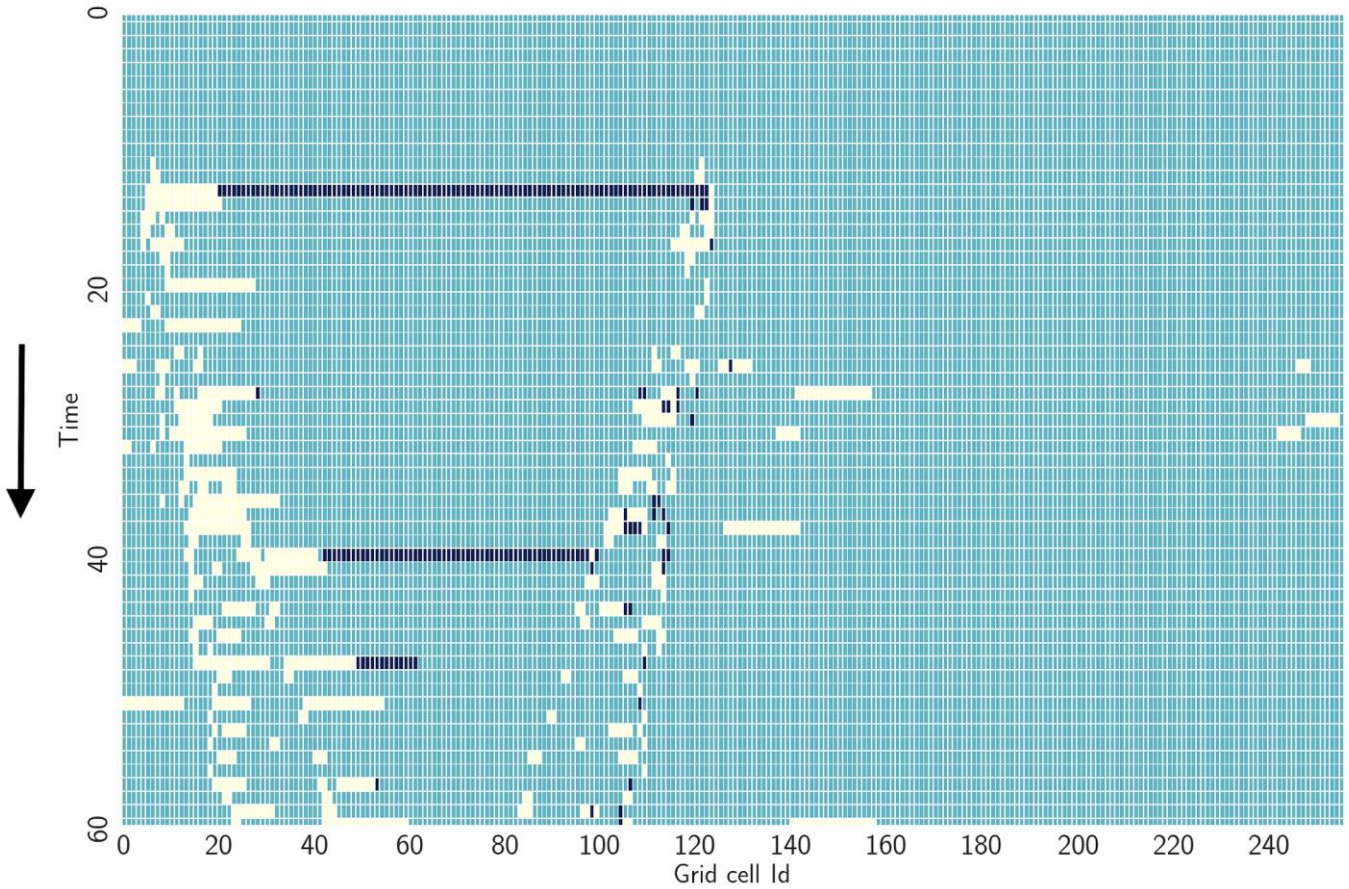
- Electron temperature ramped from 100 to 200eV over the simulation
- ~100 MD calls





What if we run out of the training set entirely?

No extrapolation needed



Overall Status

- Currently implemented:
  - Solver Codes
    - Full coupling with Multi-BGK
    - Currently debugging CFDNS coupling
  - Machine Learning and Optimization Frameworks
    - Fully coupled with Ensemble Neural Nets via PyTorch
    - Preliminary coupling to Mystic and UQ Foundation Framework
- Future tasks:
  - Coupling with Flux framework for simplified approach
  - Improving queuing of tasks
    - Prioritize important tasks
    - Improve resource utilization on heterogeneous platforms
- Current Challenges:
  - Support for passing geometries for shale application

Questions?