# Multiscale Materials Design:
# A model-to-model interface for multiscale materials modeling

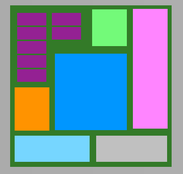
Richard LeSar
Kenneth M. Bryden

Need to use materials more effectively *within* energy systems

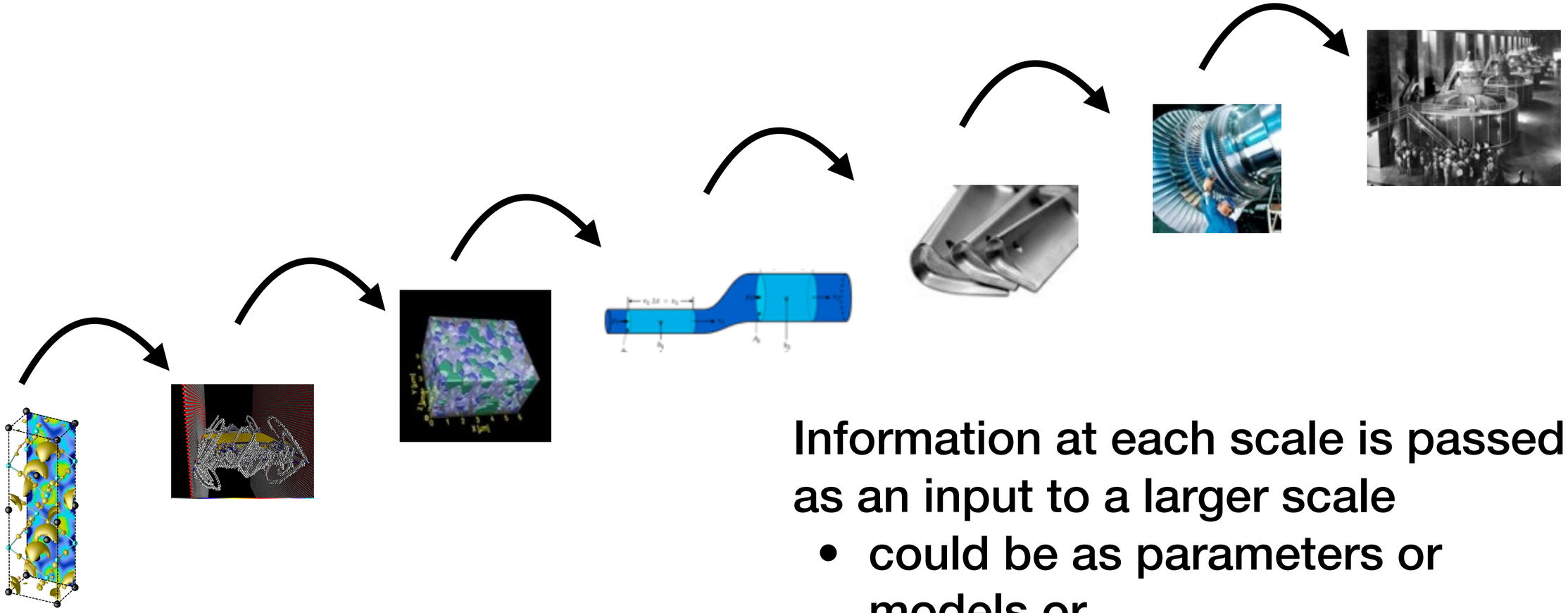
Materials have to be part of the design process

# Energy and environmental

Technology depends on material structures whose properties are governed by both the materials themselves and the interactions between them



Material behavior is governed by physics and chemistry at many scales

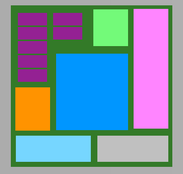


Information at each scale is passed as an input to a larger scale

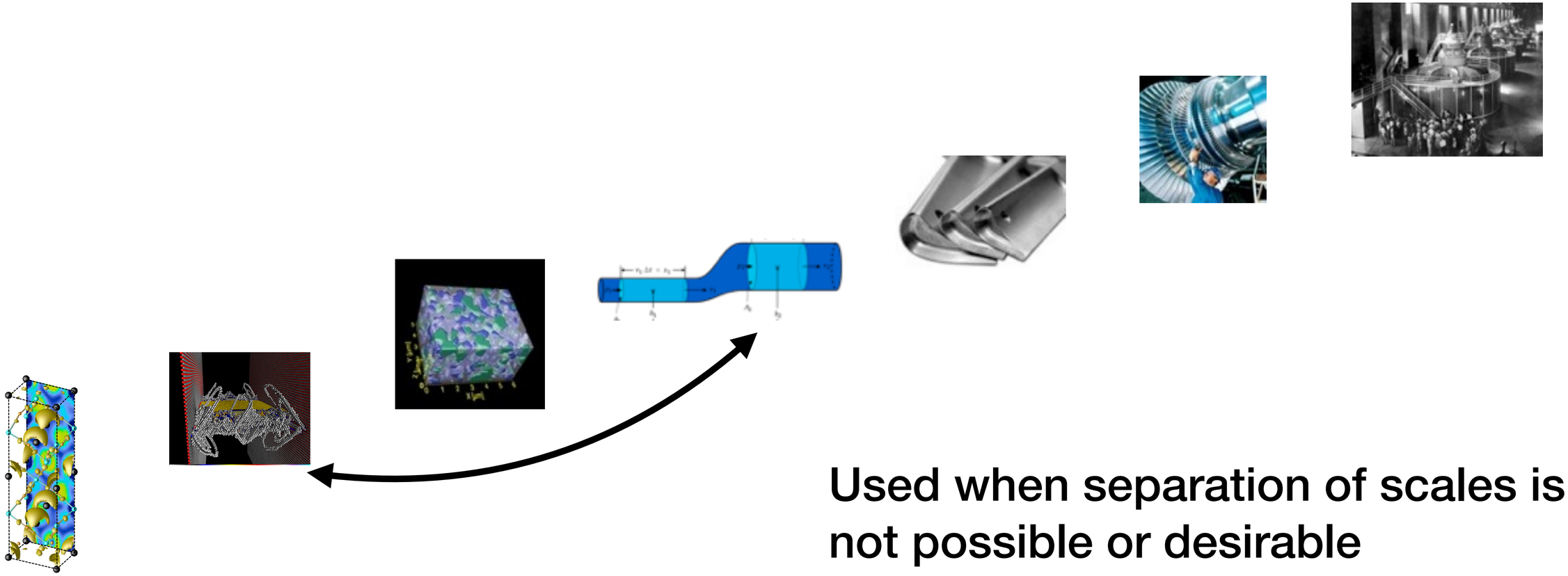
- could be as parameters or models or …

# Sequential multiscale or “information passing”

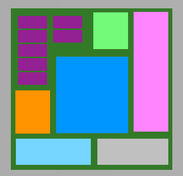
- Assumes a separation of time and length scales that is sometimes not true.
- Generally passes mean quantities.  How do we include distributions and what roles do their tails play?
- There are few theories to guide linking of scales.
- Information is passed in only one direction; we lack inverse models



## Sequential multiscale (information passing)

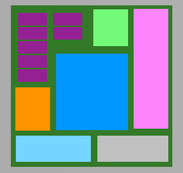


Used when separation of scales is not possible or desirable

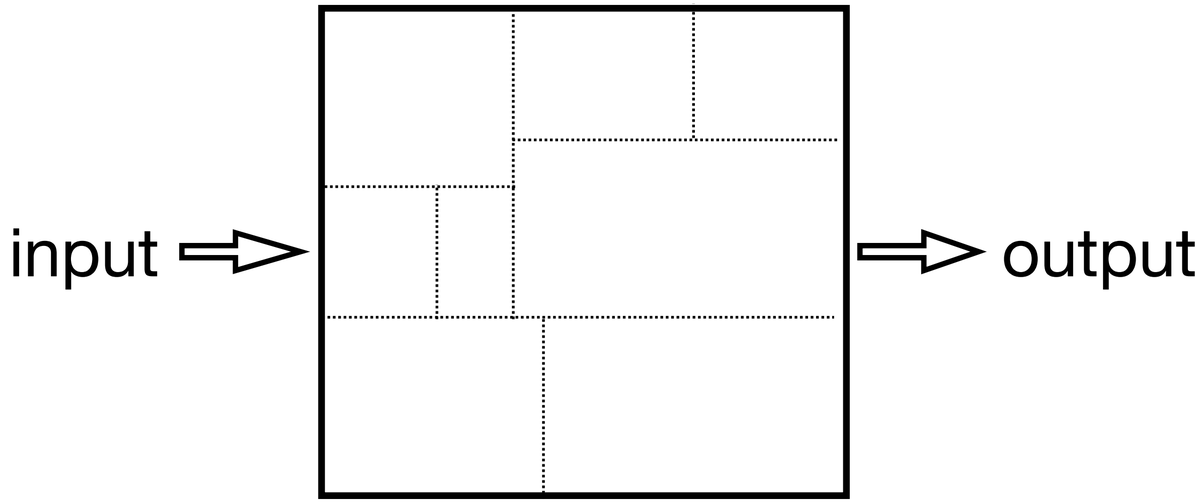


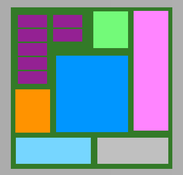
# Concurrent (coupled) multiscale

- How to link models across scales is often not well understood physically or computationally
- We need to explore the “interfaces” between the scales, i.e., the numerical and physical interconnections
- Current codes are static and cumbersome
- We need new computational approaches to link models, which starts with a more flexible way to interconnect them



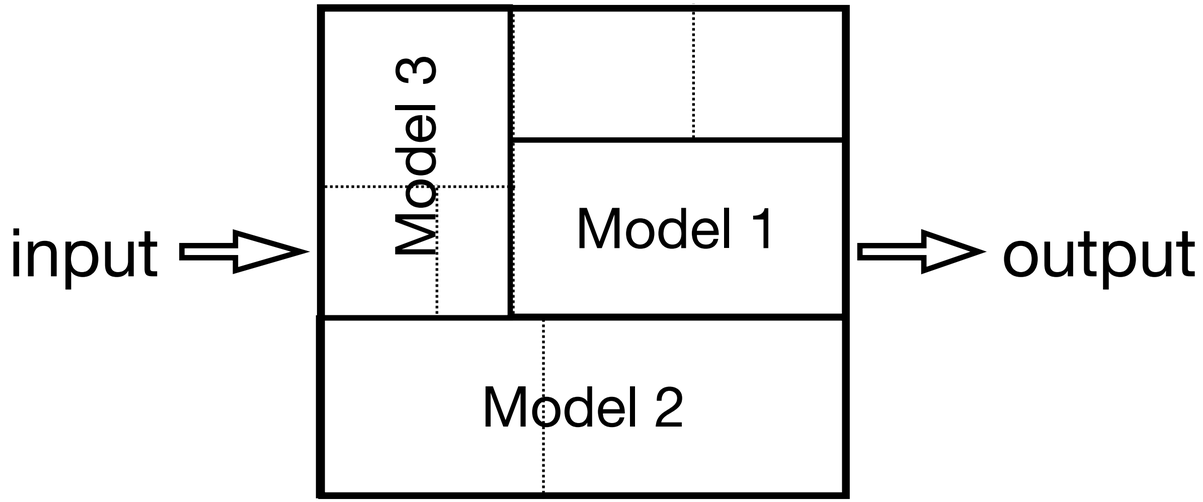
- monolithic codes consist of interconnected “subroutines” with a common data structure
- routines can be aggregated as specific models
- still must have specific interconnections and a common data structure

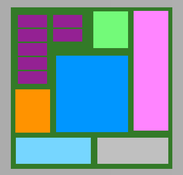


# Monolithic codes

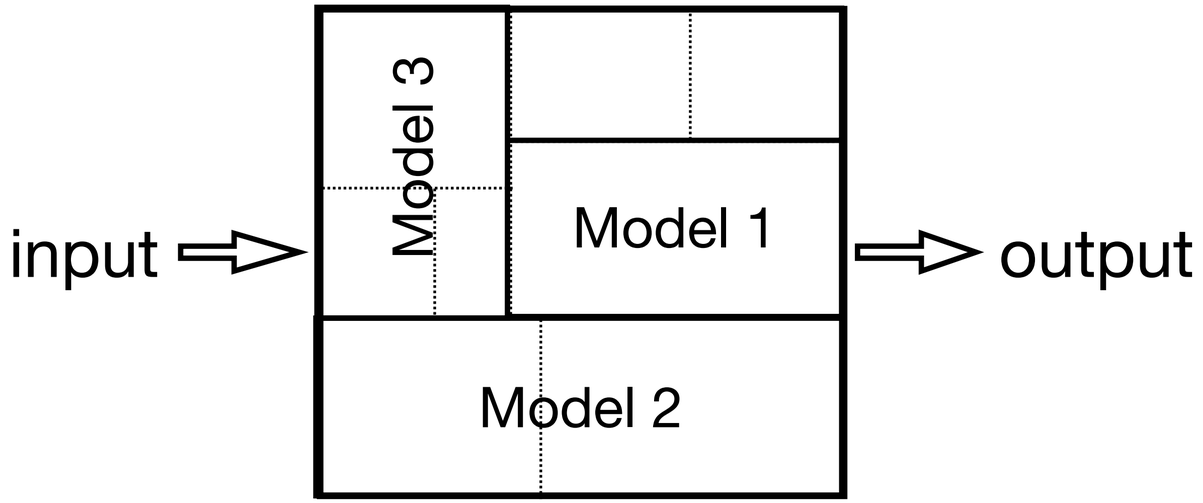
- monolithic codes consist of interconnected “subroutines” with a common data structure
- routines can be aggregated as specific models
- still must have specific interconnections and a common data structure

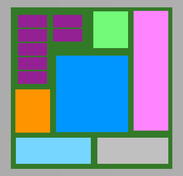


# Monolithic codes

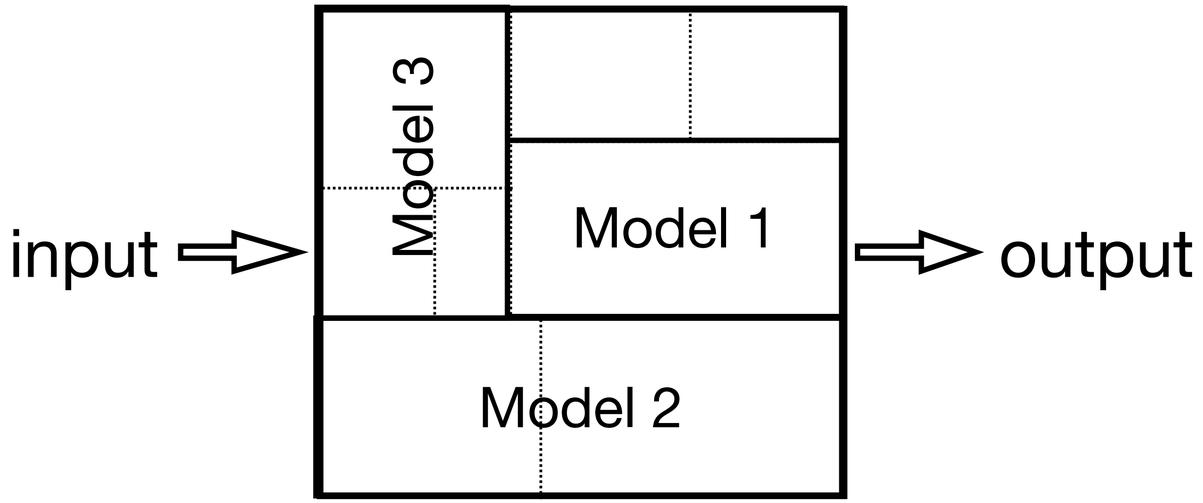
- including a new model would require that model to adopt the data structure of the rest of the code, to create linkages with other routines, …
- using the new model in some other code would require the same type of modifications
- inefficient and a barrier to adopting new models

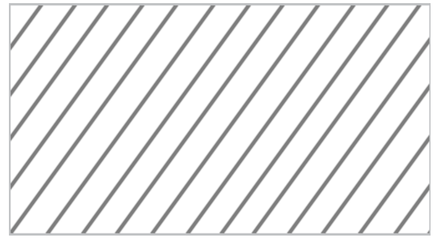


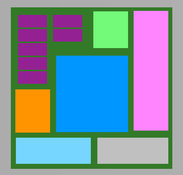



# Monolithic codes: cumbersome to maintain and change

- including a new model would require that model to adopt the data structure of the rest of the code, to create linkages with other routines, …
- using the new model in some other code would require the same type of modifications
- inefficient and a barrier to adopting new models

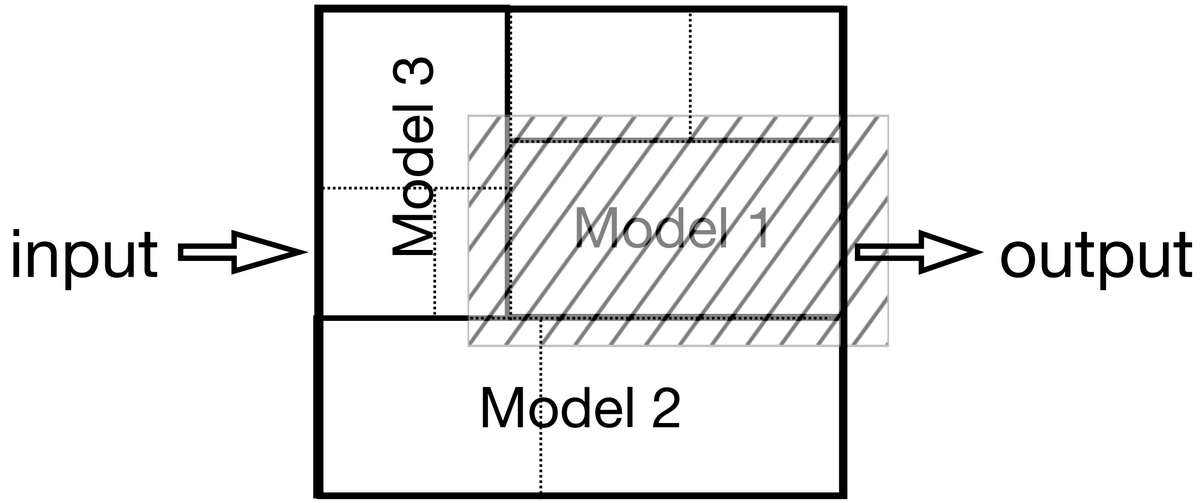


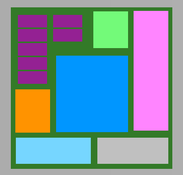





# Monolithic codes: cumbersome to maintain and change

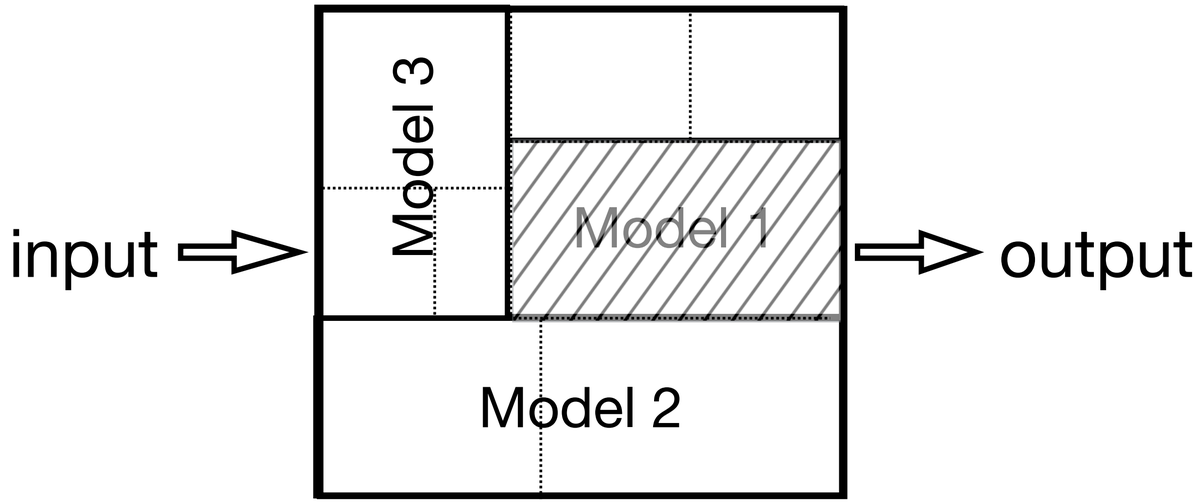
- including a new model would require that model to adopt the data structure of the rest of the code, to create linkages with other routines, …
- using the new model in some other code would require the same type of modifications
- inefficient and a barrier to adopting new models

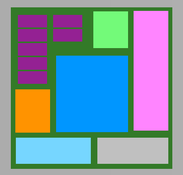


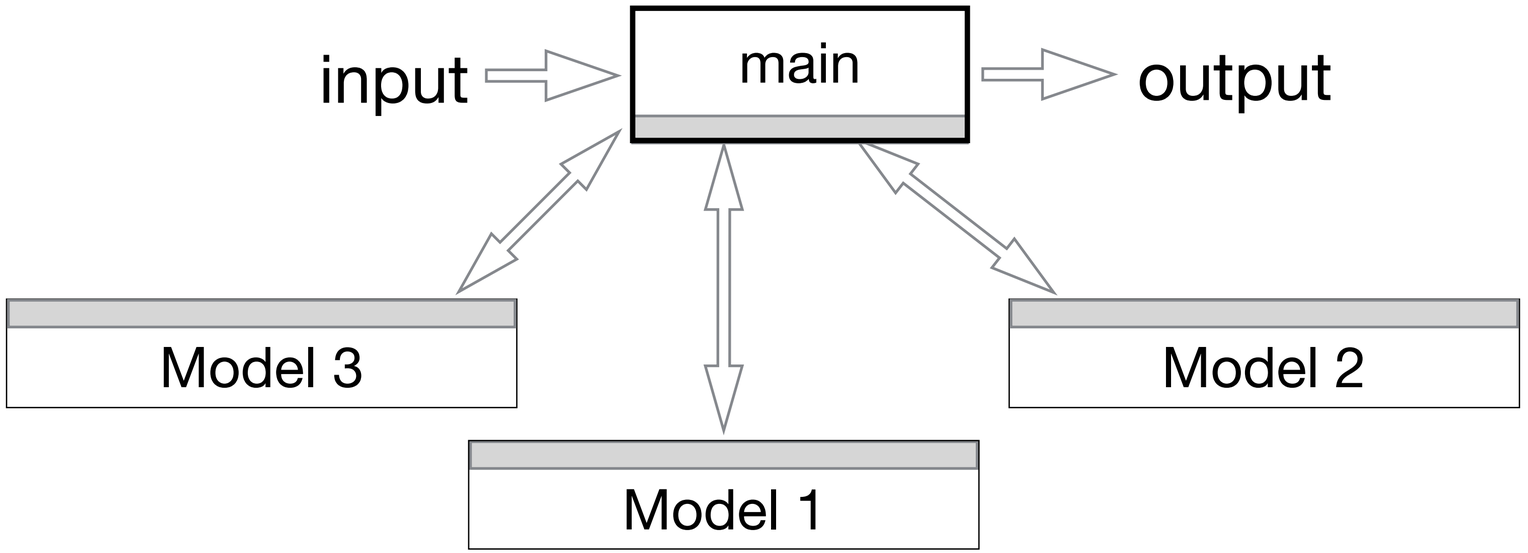
# Monolithic codes: cumbersome to maintain and change

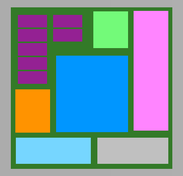
- including a new model would require that model to adopt the data structure of the rest of the code, to create linkages with other routines, …
- using the new model in some other code would require the same type of modifications
- inefficient and a barrier to adopting new models



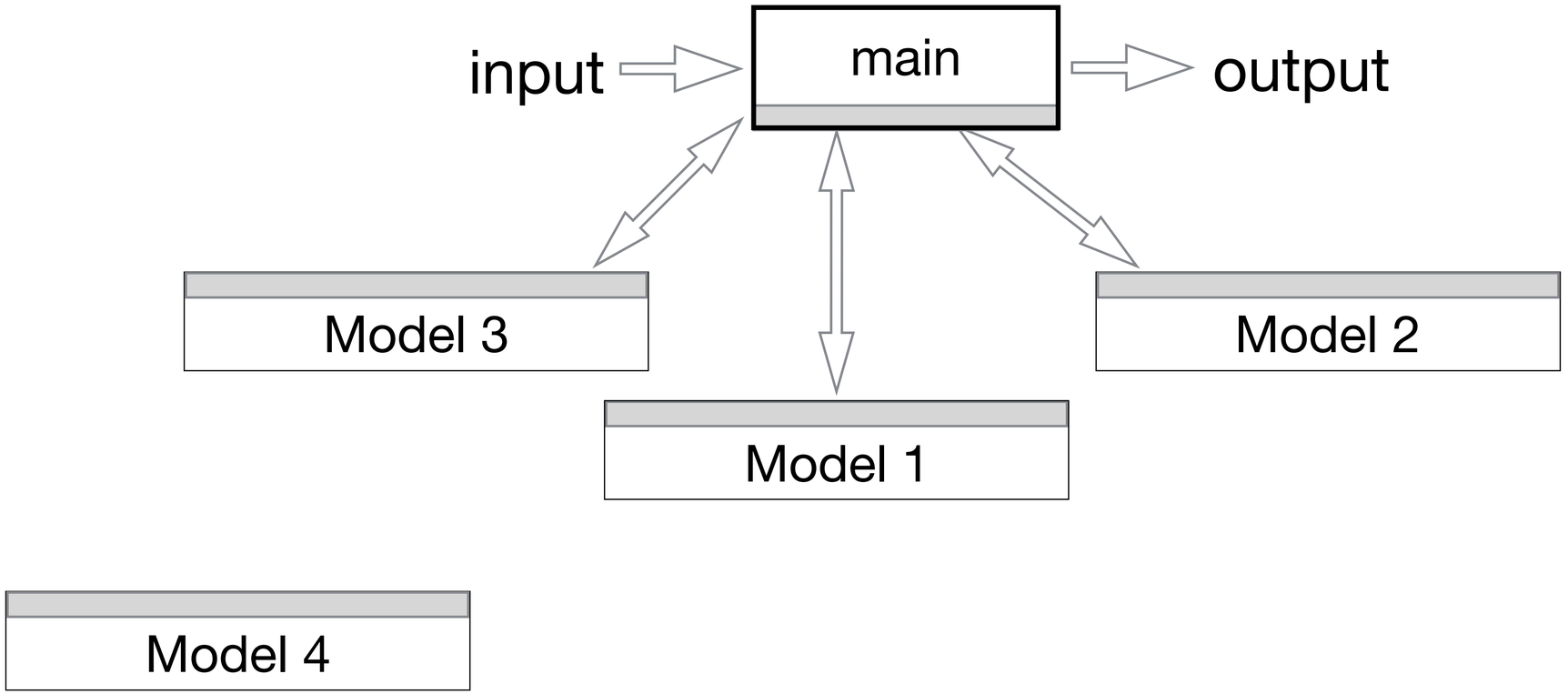



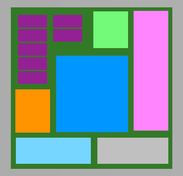
# Monolithic codes: cumbersome to maintain and change



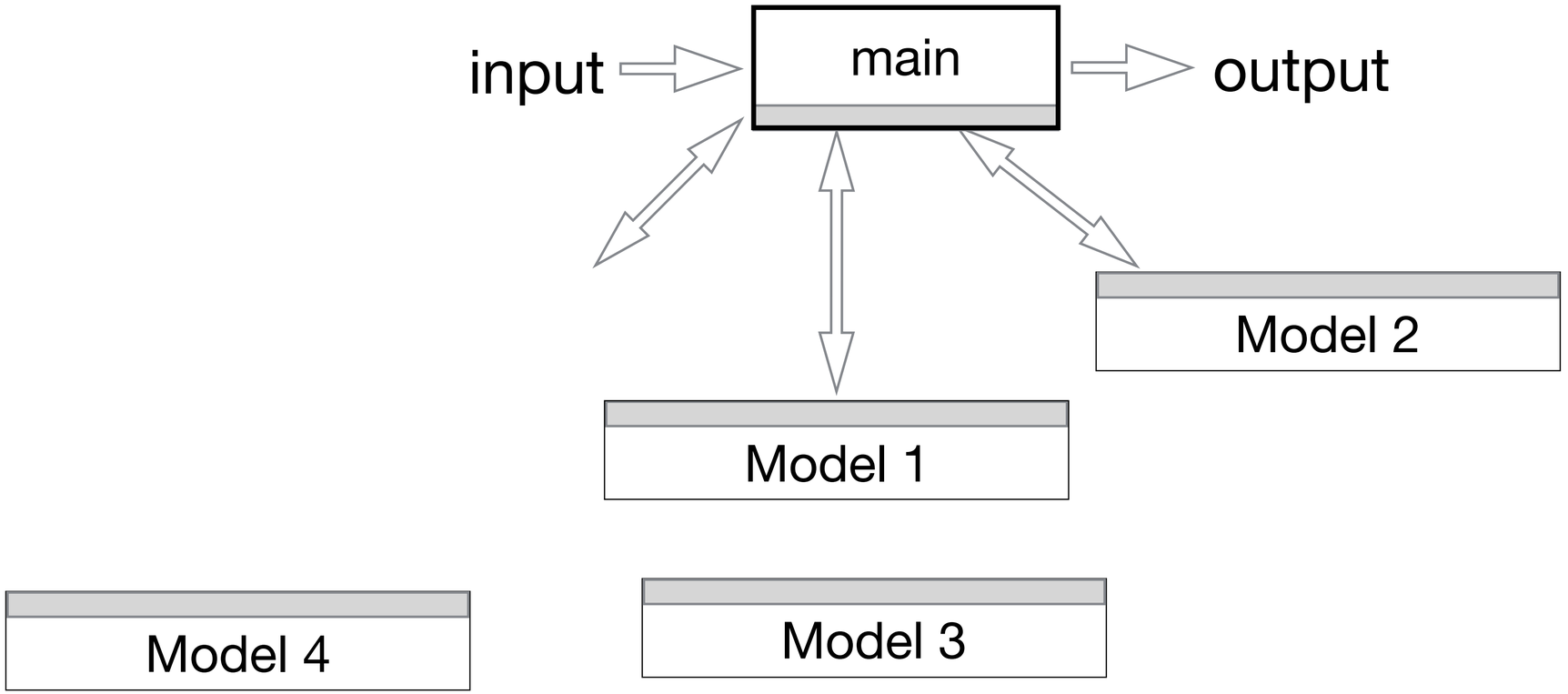



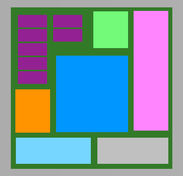
# Our goal: models coupled via a standard interface



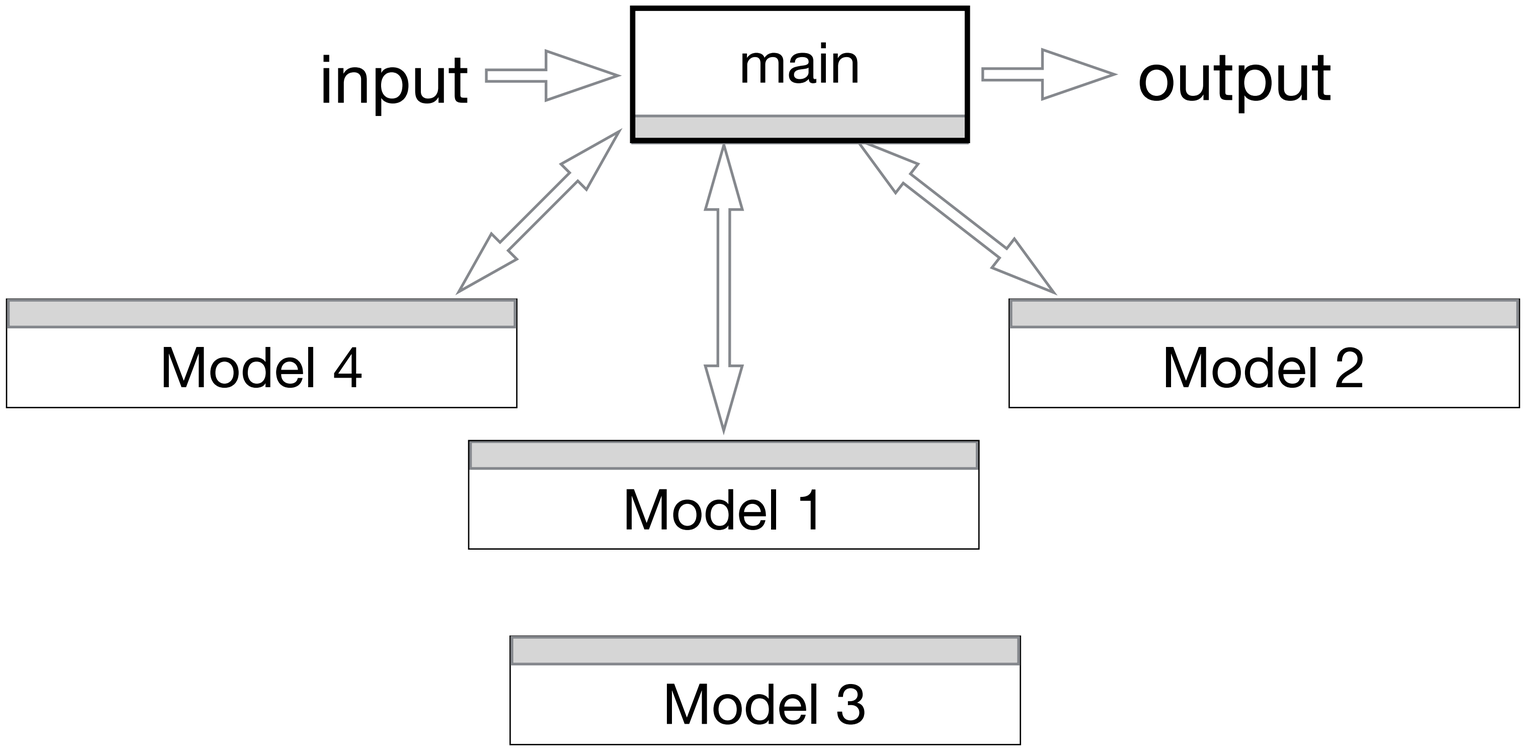



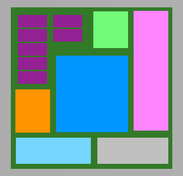
# Our goal: reusable and interchangeable models



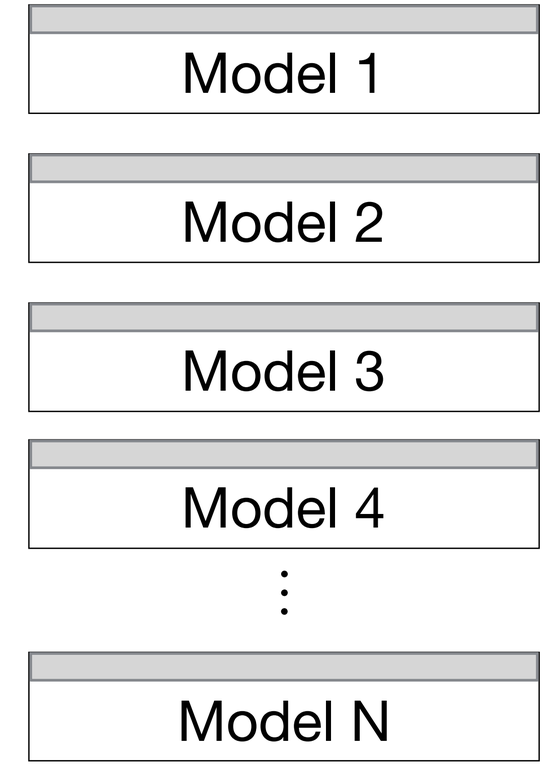
# Our goal: reusable and interchangeable models

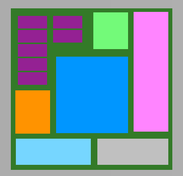


# Our goal: reusable and interchangeable models

- all communication through a standard model-to-model interface
- models are autonomous, with their own data structures, grids, etc.
- models with this interface can be developed by disparate groups
- models can be in any computer language
- models are identified by a DOI with appropriate metadata



Our goal: to create a library of reusable models

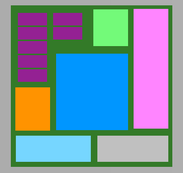
The environmental modeling community faces many of the same issues as does the materials modeling community

- disparate models developed by disparate groups
- linking models

The approach taken by the Community Surface Dynamics Modeling System (CSDMS) centered at the University of Colorado seemed to meet our needs best.

They developed an approach by which their international set of collaborators could make their models available for coupling with other models.

http://csdms.colorado.edu/wiki/Main_Page

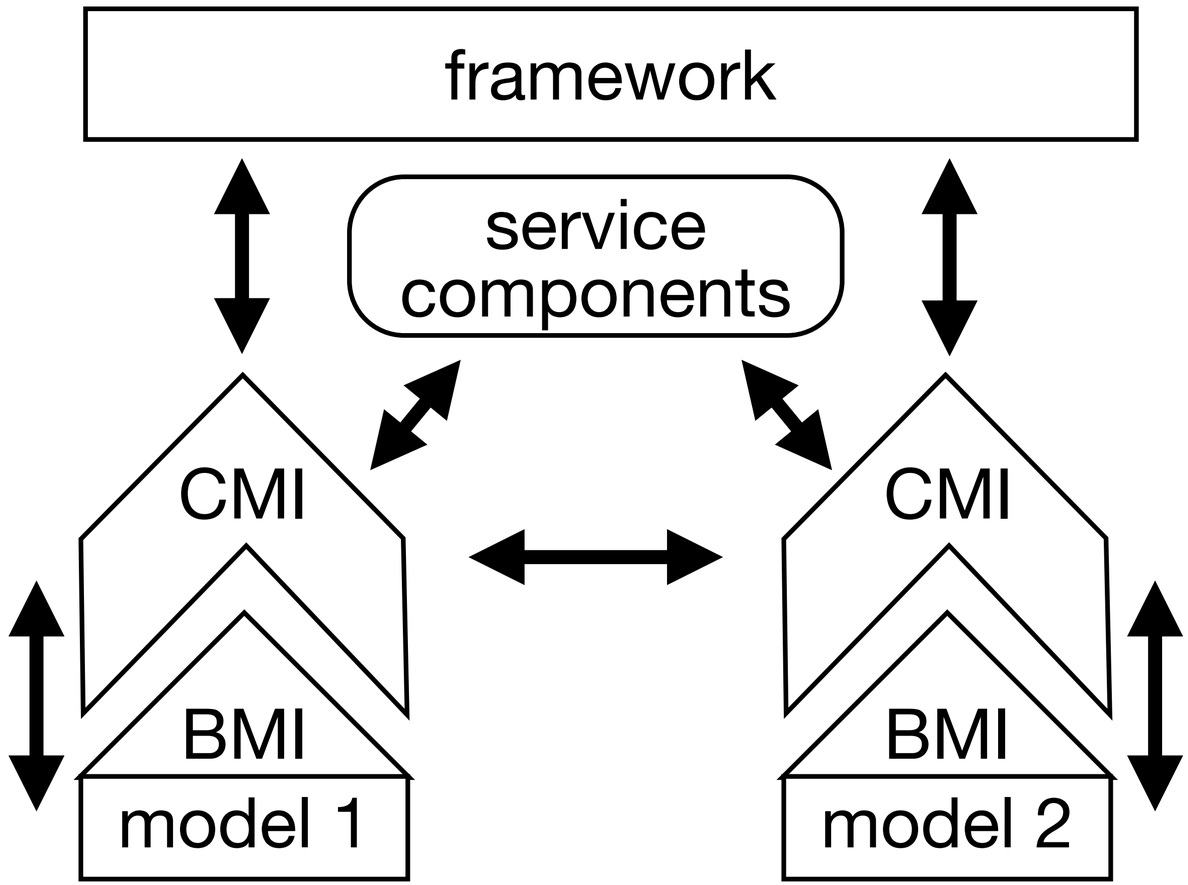


Each model has added to it the Basic Model Interface (BMI)

The Common Model Interface (CMI) automatically handles conversions between languages (with Babel)

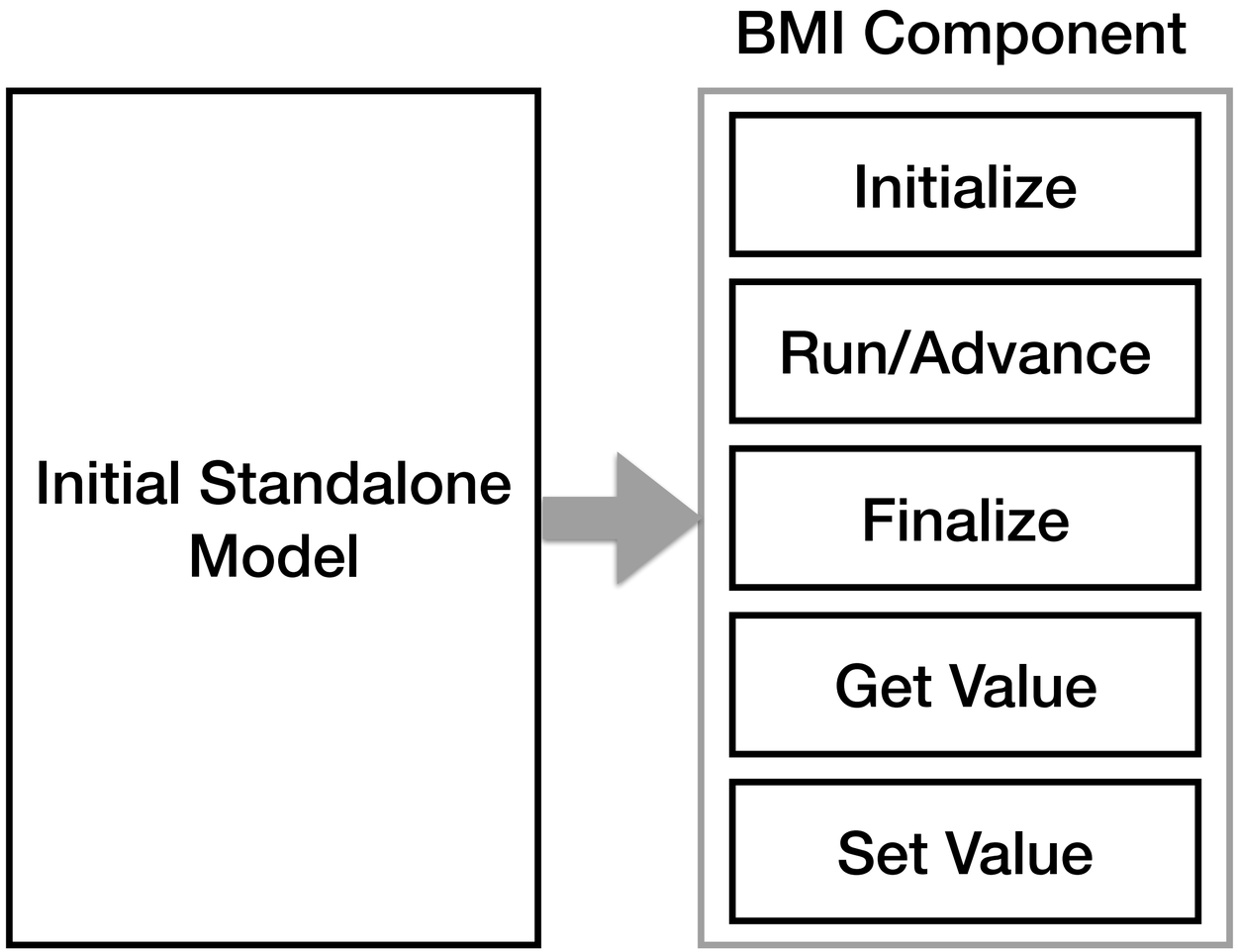
Service components handle such common activities as conversions between grids

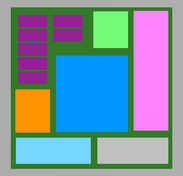
The framework controls the calculation and the communication to/from models



The BMI is a set of common functions (available in a set of computer languages) that handles all communication with the model: input, instructions, output, …

Creating a BMI for materials modeling was our first task.

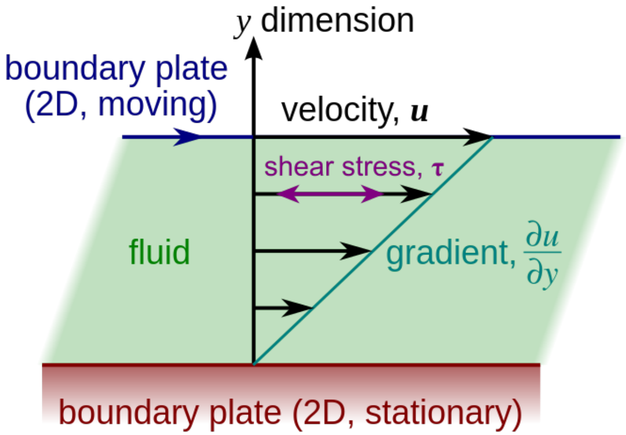


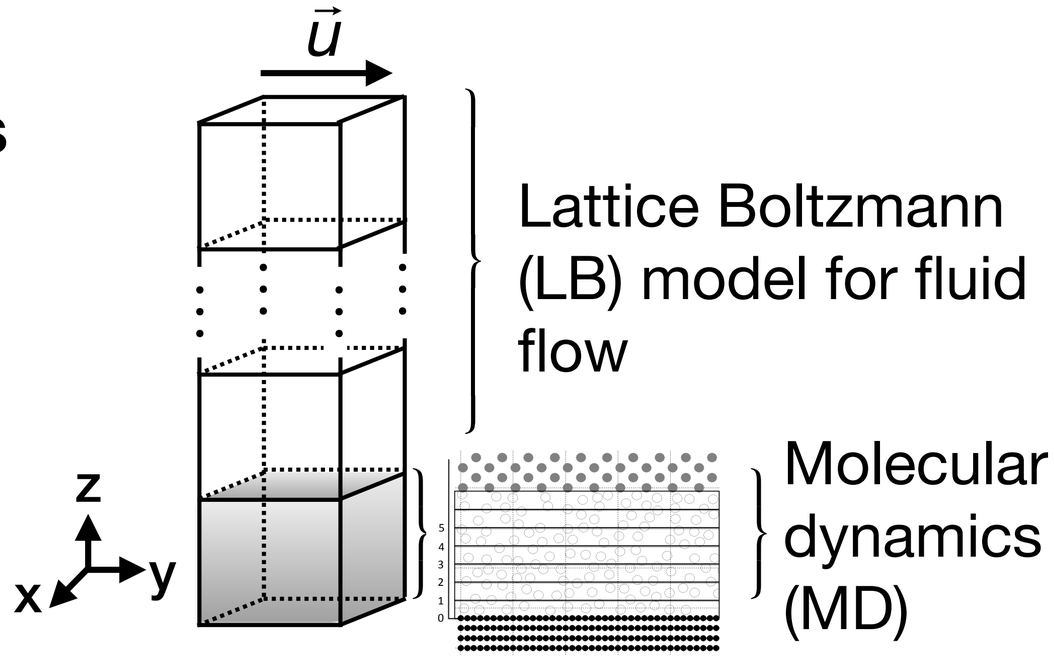



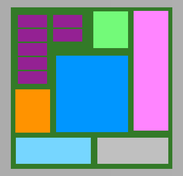
We linked fluid flow calculations to atomistics to calculate slip velocity at bottom surface in Couette flow

Created a BMI for each model and examined

- information transfer between models
- boundaries between models
- convergence of the solution
- stability
- …

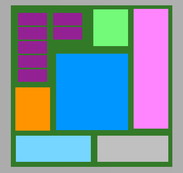


Create a hybrid system to examine key questions

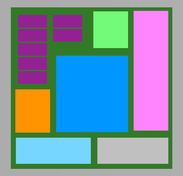
- The Lattice Boltzmann (LB) and molecular dynamics (MD) models were autonomous with their own internal data structure
- Each model had its own internal units.
- Each model was solved with its own time step (very different in size).
- Each model had its own implementation of boundary conditions.
- Each model had its own requirements for convergence.
- Models can be stateless



# Information mediation between LB and MD

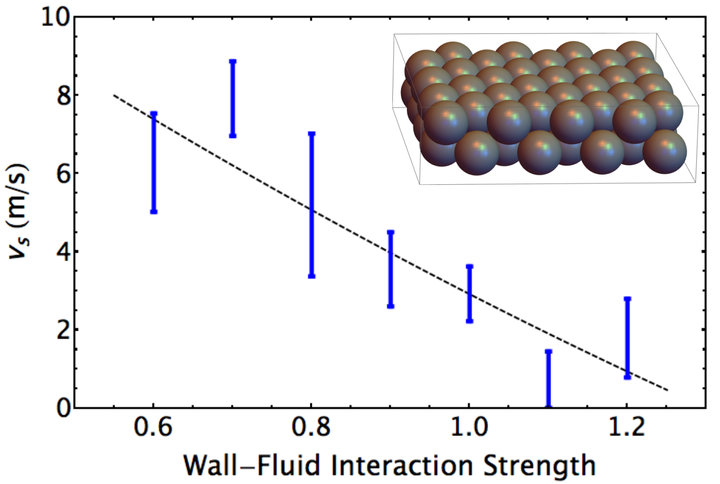
- The Lattice Boltzmann (LB) and molecular dynamics (MD) models were autonomous with their own internal data structure
- Each model had its own internal units.
- Each model was solved with its own time step (very different in size).
- Each model had its own implementation of boundary conditions.
- Each model had its own requirements for convergence.
- Models can be stateless

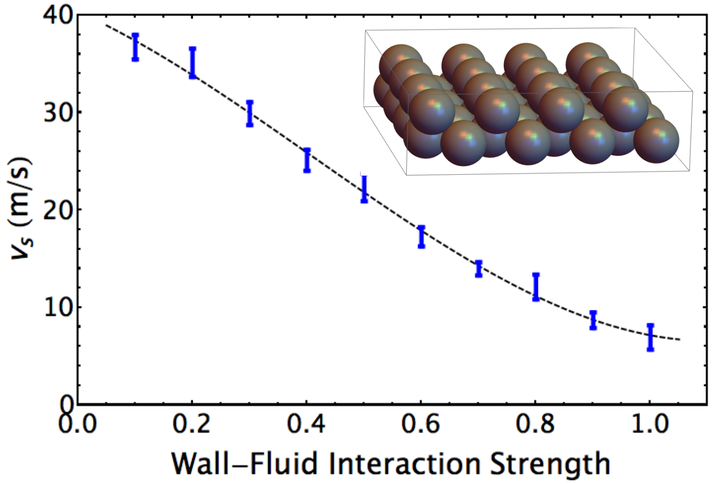
*The BMI handles all communication with the models: input, instructions, output, … through a standard set of functions.*

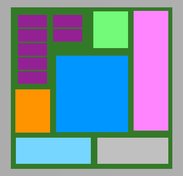


# Information mediation between LB and MD

- Examined slip velocity as a function of solid surface crystal structure and the interactions between fluid atoms and solid atoms
- We saw no change in computational efficiency with the BMI when compared to an optimized monolithic code

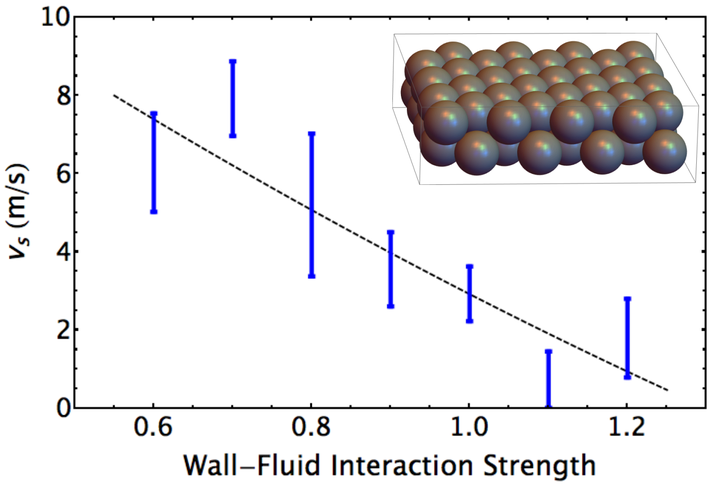


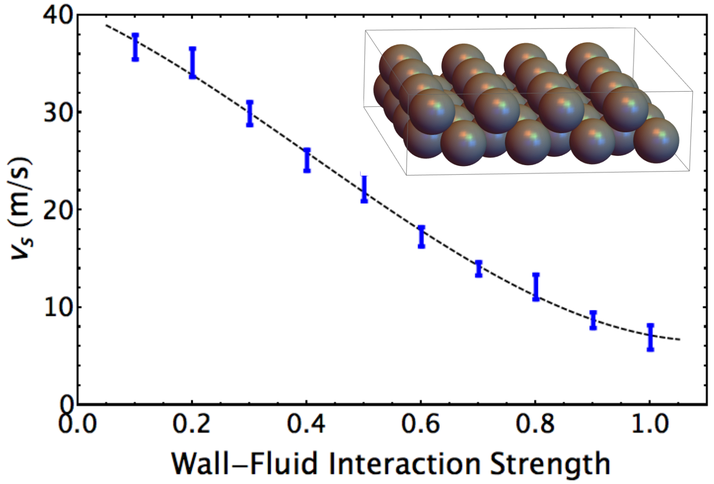



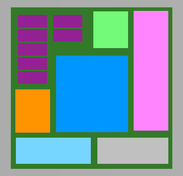



# Results

- Examined slip velocity as a function of solid surface crystal structure and the interactions between fluid atoms and solid atoms
- We saw no change in computational efficiency with the BMI when compared to an optimized monolithic code



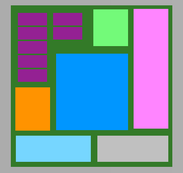



## Results

*"A model-to-model interface for multiscale materials simulations, P. E. Antonelli, K. M. Bryden, and R. LeSar, Computational Materials Science* ***123****, 244-251 (2016).*

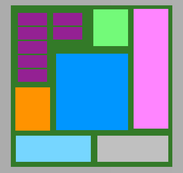
- Straightforward to implement BMI functions within the code
- We use only the BMI functions and include all materials specific information as models (with a BMI)
- We can then directly link to the higher-level structures (CMI) to link codes in different languages, etc.

| function | purpose |
|---|---|
| void initialize(string input_file, string identifier) | allocates memory for model and sets input variables |
| void run(int time_steps) | runs model for number of time steps based on value of time_steps |
| void finalize() | deallocates memory for model and prints output to a file |
| vector <string> get_input_var_names() | returns list of input variables |
| vector <string> get_output_var_names() | returns list of output variables |
| vector <string> get_boundary_condition_names() | returns list of usable boundary conditions |
| vector <string> get_boundary_condition_var_names(string boundary_condition) | returns list of variables to use to enforce given boundary condition |
| string get_var_type(string variable) | returns data type of variable |
| string get_var_units(string variable) | returns units of variable |
| int get_var_rank(string variable) | returns rank of variable |
| double get_0d_double(string) | returns value of a zeroth rank floating point variable |
| vector <double> get_1d_double(string) | returns a first rank floating point variable |
| void set_2d_double_at_index(string, double, int, int) | set the value of a second rank floating point variable at a specified index |
| int get_3d_int_at_index(string, int, int, int) | return the value of a third rank integer variable at a specified index |
| vector < vector < vector < vector < string> > > > get_4d_string(string) | return a fourth rank string variable |



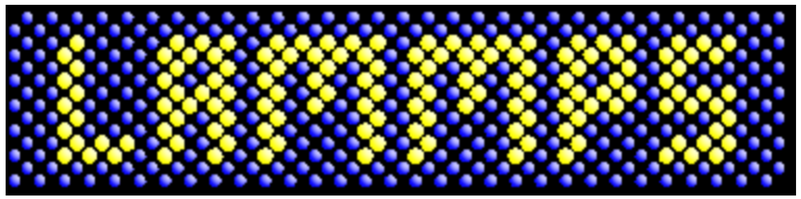
# BMI functions: lessons learned

- Have created BMI-versions of other models (created by us)
  - e.g., a finite difference heat flow code and linked it successfully with our Lattice-Boltzmann fluid flow code
- Goal is to link to general codes in the materials modeling community
  - requirement:  access to the source code
- We wanted to start with a code that has general and proven applicability in materials research and add a BMI to that



## Extension to new codes and methods

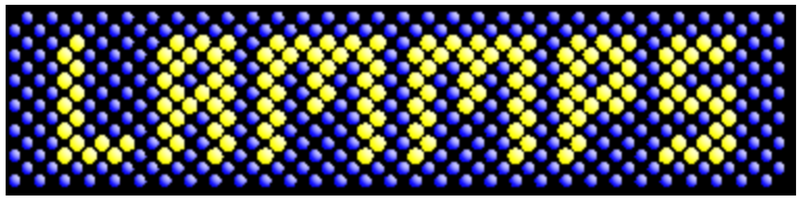
- LAMMPS is an open-source molecular dynamics program created at Sandia National Laboratories
- Used throughout the materials modeling community (original paper cited >9000 times; “LAMMPS” found over 17000 times in Google Scholar (830 in 2017)
- LAMMPS was written to facilitate collaborative additions to its capabilities



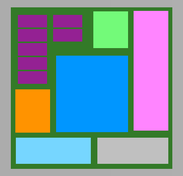
lammps.sandia.gov

# Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS)

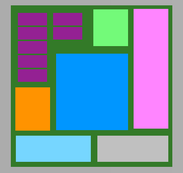
- The open structure of LAMMPS enables a relatively easy implementation of the BMI
- LAMMPS works by creating an object of the class LAMMPS, reading an input file, and translates each line into code to execute — many many options
- Since LAMMPS is already a class, it can be used with the BMI by simply declaring it to be a subclass of the model and adding the BMI functions to it



lammps.sandia.gov

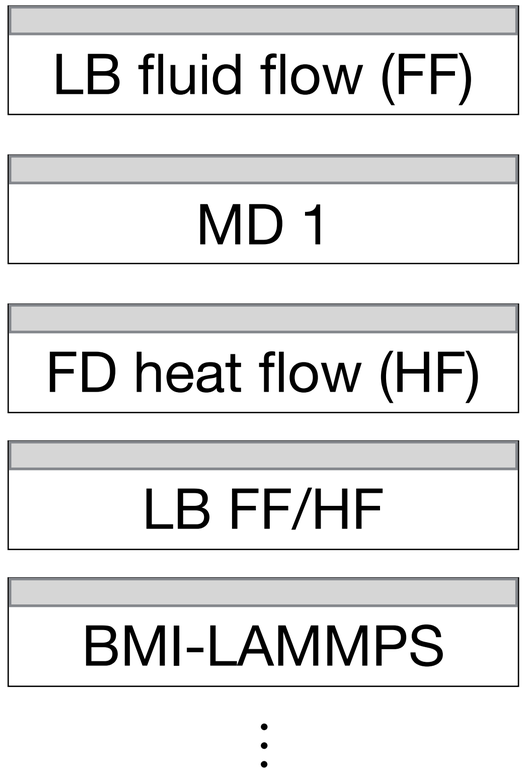


- Added the BMI to LAMMPS — BMI-LAMMPS
- We linked BMI-LAMMPS to our Lattice-Boltzmann code with no additional changes to either code
- We repeated our previous study of Couette flow and found similar, but not identical results (owing to differences in how we implemented thermostats and other details in our code)
- Work is ongoing to extend the BMI to include more of LAMMPS's functionalities (there is an up front cost to adding the BMI)

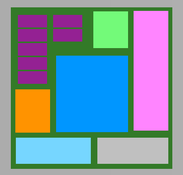


# Application of BMI-LAMMPS

- We now have the beginnings of a library of BMI-based codes
- All of these can be linked together with no changes to the codes themselves.
- By using the BMI on all future projects, we will add to the library over time

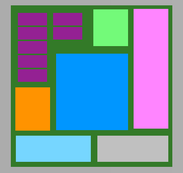



*The first stage of our library of models*



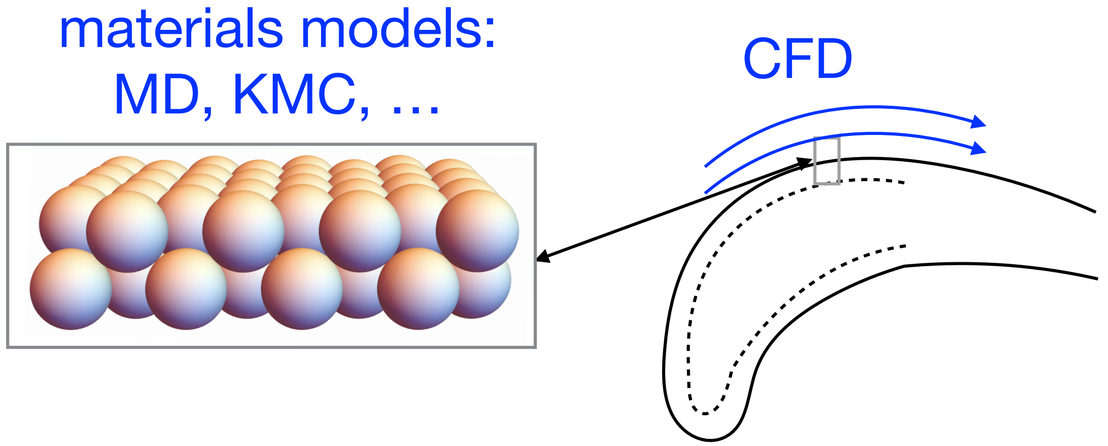
# Model library

- Working with NETL, we have identified ongoing directions
  - physics-based analysis tools to model heat transfer in turbine components to support the development of advanced turbines
  - creating an easy-to-modify model for additive manufacturing (AM) processes to guide development of materials for specific applications

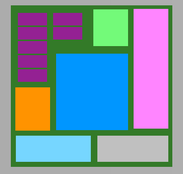


# Current directions (ongoing)

- Objective: better define the fundamental processes that underlie heat transfer and chemistry in these complex physical systems as a guide to improve their usability in high-temperature turbine environments.
- Will link BMI-versions of all models to create multiscale design tool
- Work with Tom Shih (Purdue)
- Identification of models is ongoing

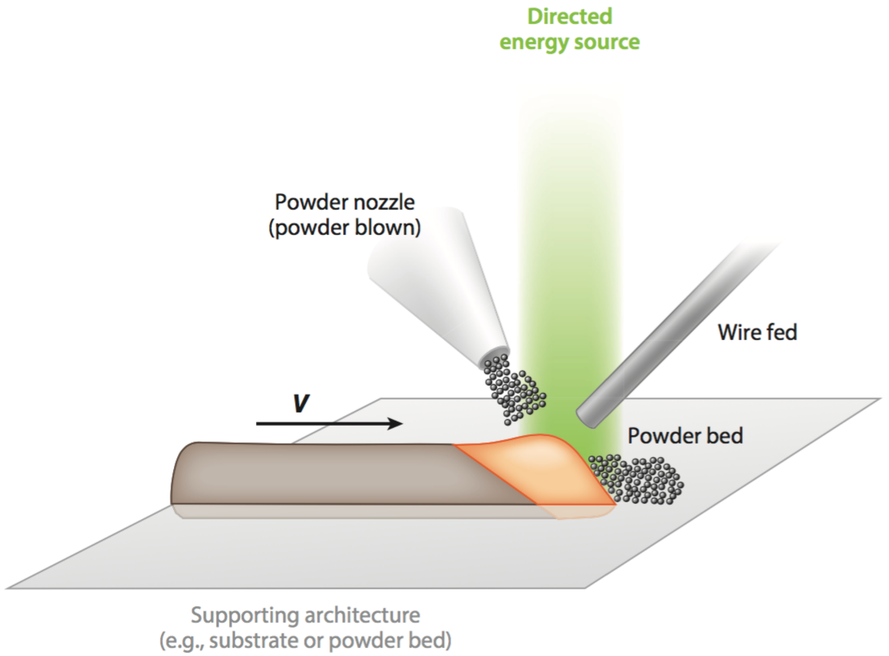


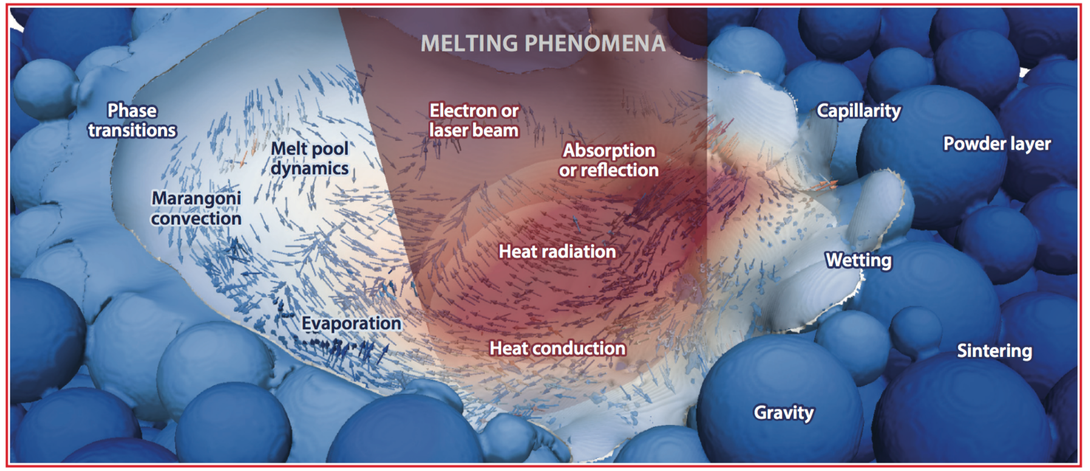

Coupled multiscale models: fluid flow will be modeled with computational fluid dynamics and will be concurrently coupled with detailed materials models such as molecular dynamics (MD: BMI-LAMMPS) and kinetic Monte Carlo (BMI-KMC).

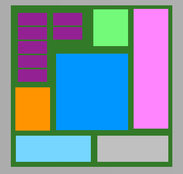


# Future directions: heat transfer in turbine blades

- Additive manufacturing (AM) offers unique opportunities to create structures not easily manufactured by traditional means — explore design space
- However, AM structures have microstructural and surface features that are not always ideal
- The AM process is complicated and it is difficult to know its outcome — modeling can help identify optimal design space that couples material properties and structure

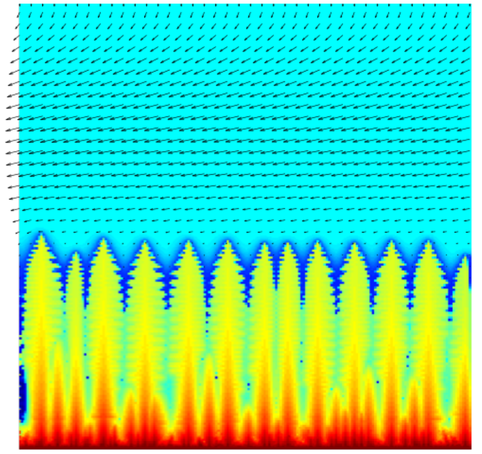


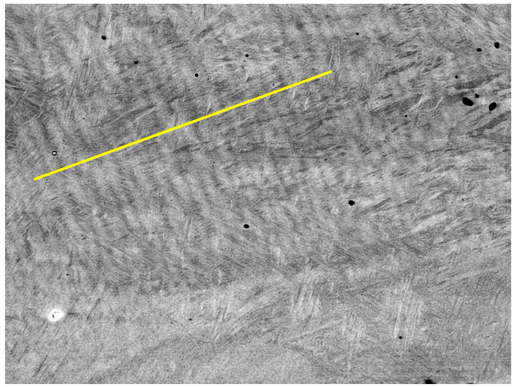



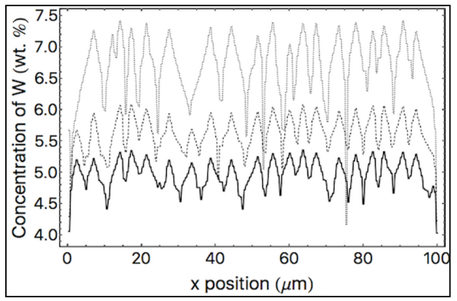



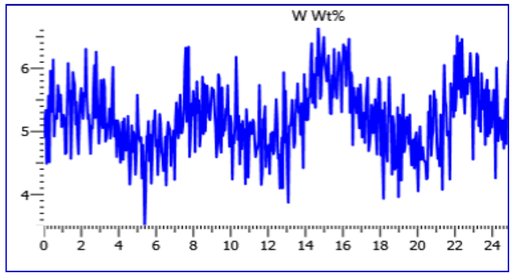
# Future directions: additive manufacturing

- Modeling couples fluid flow, heat flow, and solute flow with models of solidification
- We will created BMI versions of these models to create an easy-to-modify AM simulation package for FE
- We will optimize materials and manufactured structures
- Enables the material and structure to each be part of the design process





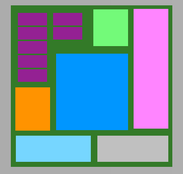



*Model* *Experiment*

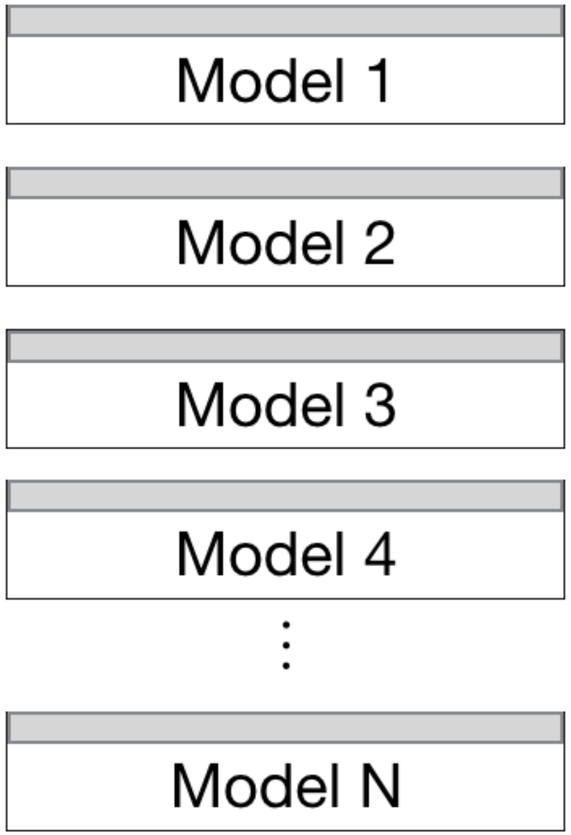
*Calculation and characterization of solute variation across the microstructure.*

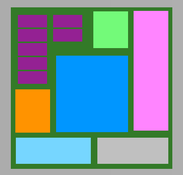
Work submitted to *Metall Mater Trans*



## Future directions: additive manufacturing

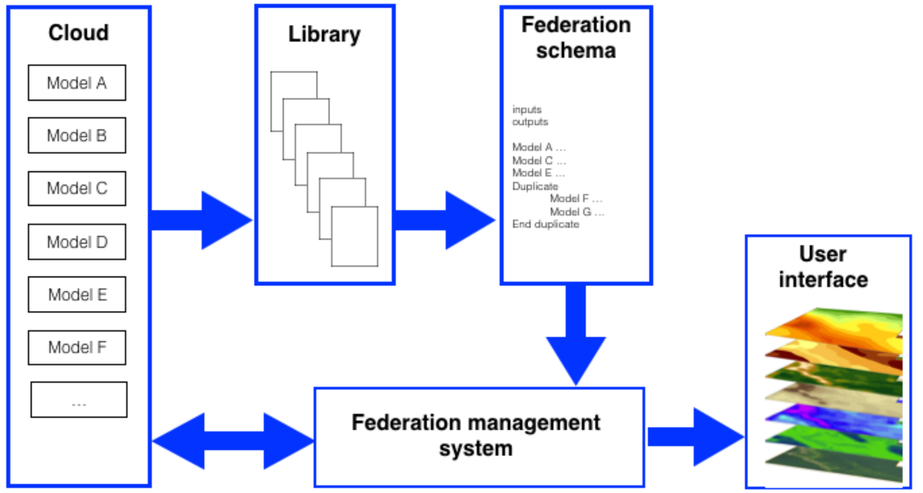
Develop a library of multiscale materials models, created by us and by the FE modeling community, to enable the DOE to create dynamic simulation tools in support of affordable, low carbon, high efficiency, advanced power systems.

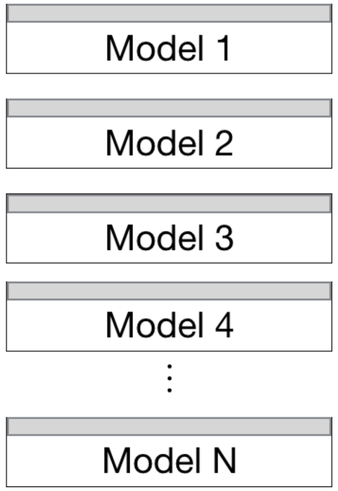


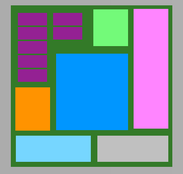



# Vision

- Goal of the ECS program is to create a cloud-based system that acts as a web service for engineering models
- The MMD program will develop a library of materials models that will able to be linked and run on a single computer system to solve specific problems
- We will merge MMD codes into the web-service model

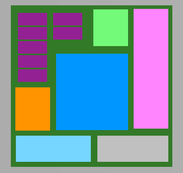


## Connection to Engineering of Complex Systems program

Richard LeSar
515-294-1841
lesar@iastate.edu



# Contact info