



MCL 20<sup>th</sup> Anniversary, Leoben, Austria,  
November 20<sup>th</sup>, 2019

# Uncertainty Quantification for Calphad Models Enabled by Machine Learning

Presented by Dr. Richard Otis

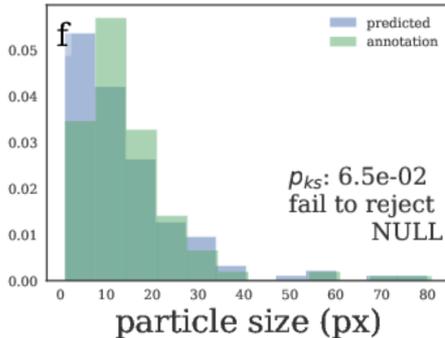
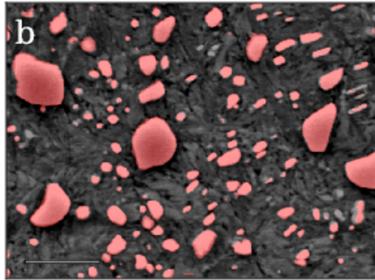


**Jet Propulsion Laboratory**  
California Institute of Technology

Government Sponsorship Acknowledged

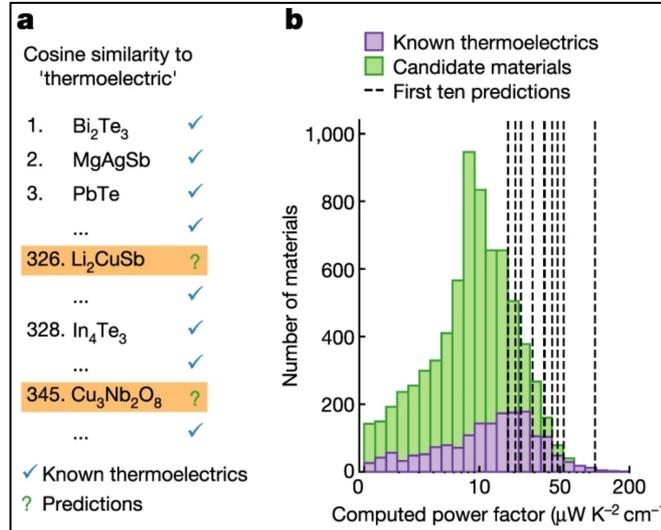
# Why is ML for Materials interesting?

## Convolutional Neural Nets



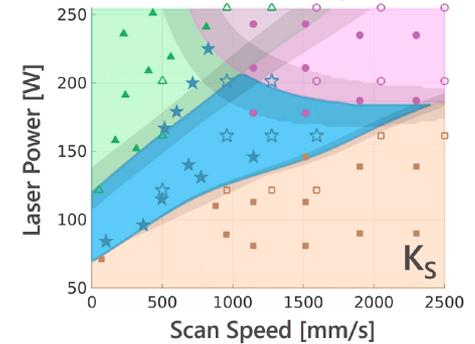
B. L. DeCost, et al., *Microsc. Microanal* **25**(1), 21–29 (2019)  
doi: 10.1017/S1431927618015635

## NLP/Unsupervised word embeddings



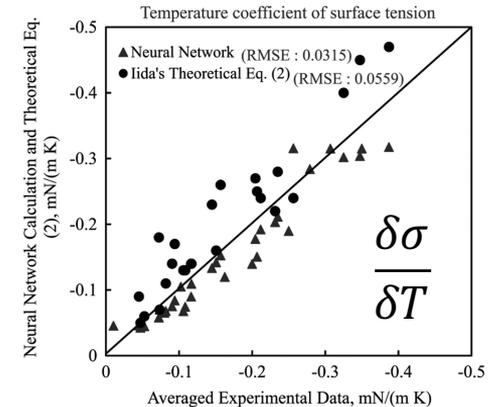
V. Tshitoyan, et al., *Nature* **571**, 95–98 (2019)  
doi: 10.1038/s41586-019-1335-8

## Gaussian Process Regression



L. Johnson, et al., *Acta Mater.* **176**, 199–210 (2019)  
doi: 10.1016/j.actamat.2019.07.005.

## Neural Networks



J. Yeon, et al., *Calphad* **64**, 267–271 (2019)  
doi: 10.1016/j.calphad.2018.12.008

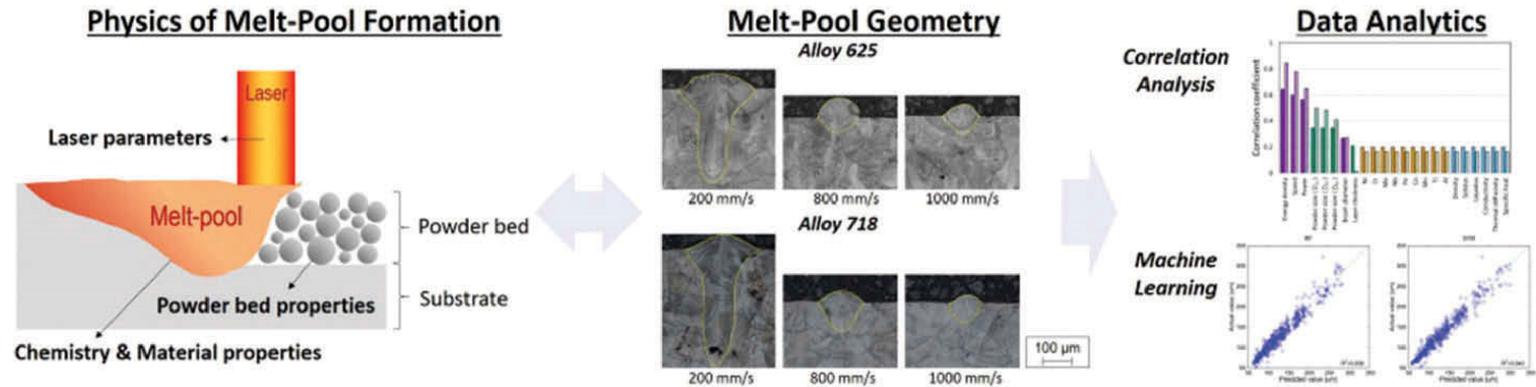
# Teaching Materials Science to AI

- **What We Want:** Given an arbitrary number of heterogeneous observations about the universe, rationalize these observations with a self-consistent worldview, with which we may make predictions about unseen phenomena.
- **“Here are some examples of  $x$  and  $y$ . Can you tell me  $f(x)$ , for  $y = f(x)$ ?”**
  - $x$ : “features”
  - $y$ : “targets”

$x_1$	$x_2$	$x_3$	$x_4$	$y_1$	$y_2$
0.2	-10	“red”	6.8	0.3	-7
...	...	...	...	...	...

# The Importance of Feature Selection

Where Do The x's Come From?



S. Lee, et al., *Sci. Tech. Adv. Mater.* **25(1)**, 972-978 (2019)  
doi: 10.1080/14686996.2019.1671140

## Key Concept

Features in machine learning should come from both experimental observations, and outputs of physics-based models.

# PyCalphad = Python + CALculation of PHAse Diagrams

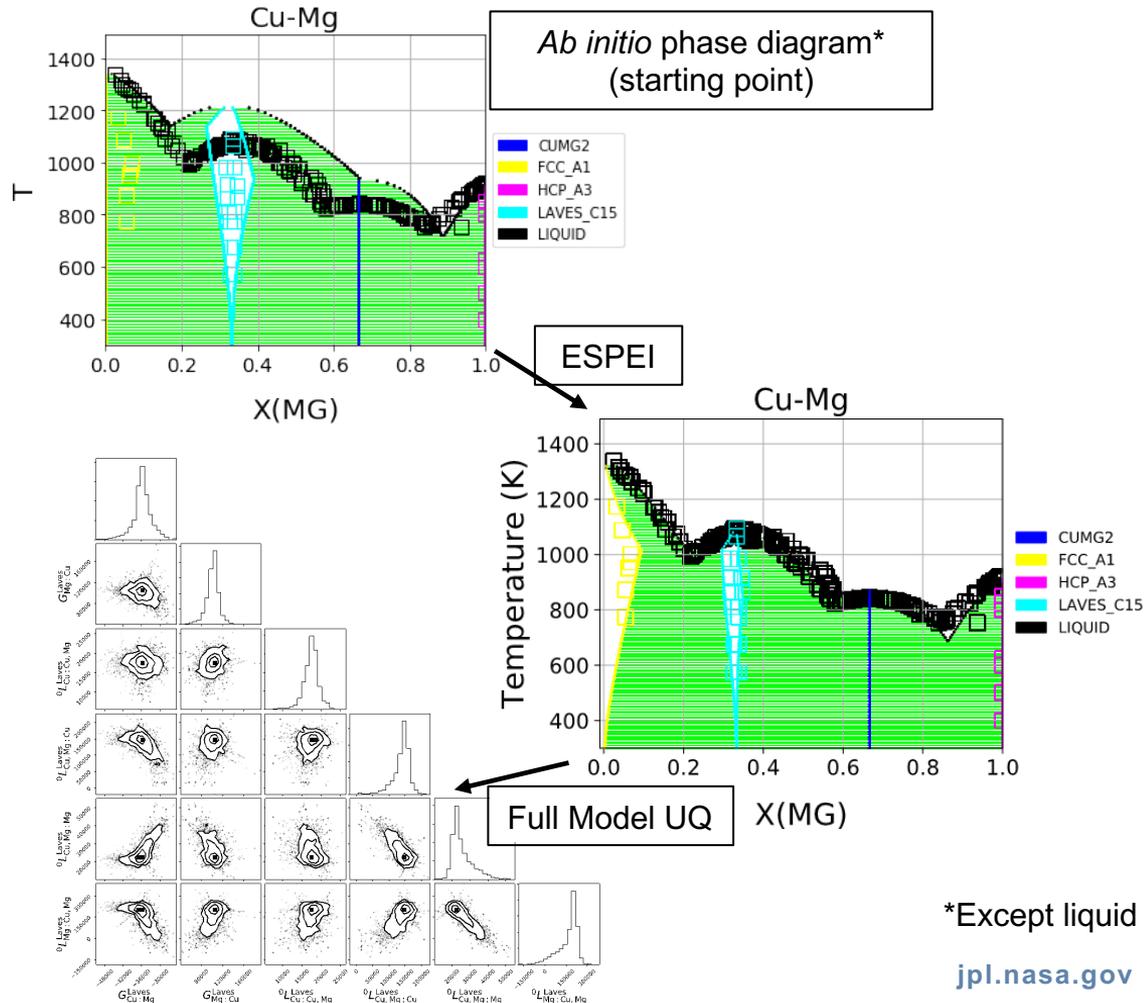
- Using physics-based models, PyCalphad predicts properties of materials, including
  - Transition (e.g., melt) temperatures, speciation, solidification, degradation, corrosion, etc.
  - Chemical processes as a function of temperature, pressure, and composition
  - **Supports uncertainty quantification**
- Free and open source at <https://pycalphad.org>



# PyCalphad and ESPEI

- ESPEI uses PyCalphad as a kernel in a large Monte Carlo simulation
  - Tens of thousands of candidate models on HPC
- Output is new model + full Uncertainty Quantification
- Free and open source at <https://espei.org>

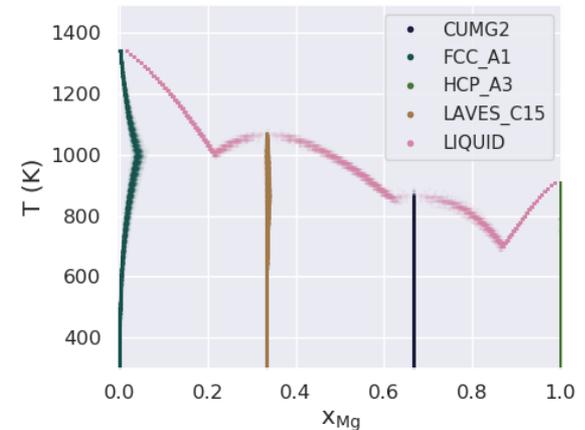
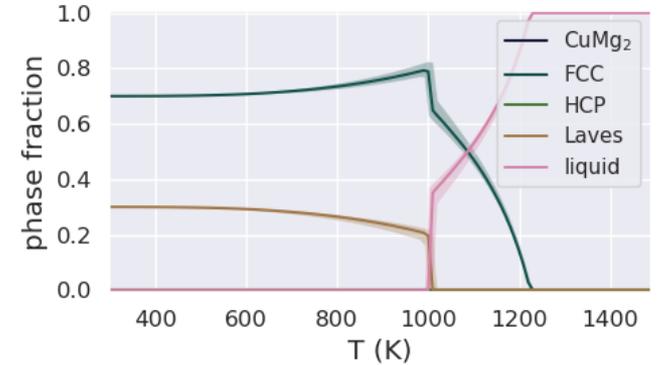
B. Bocklund, et al. "ESPEI for efficient thermodynamic database development, modification, and uncertainty quantification: Application to Cu–Mg." *MRS Communications*, 9(2), 618-627 (2019). doi:10.1557/mrc.2019.59



# Uncertainty Propagation for Materials

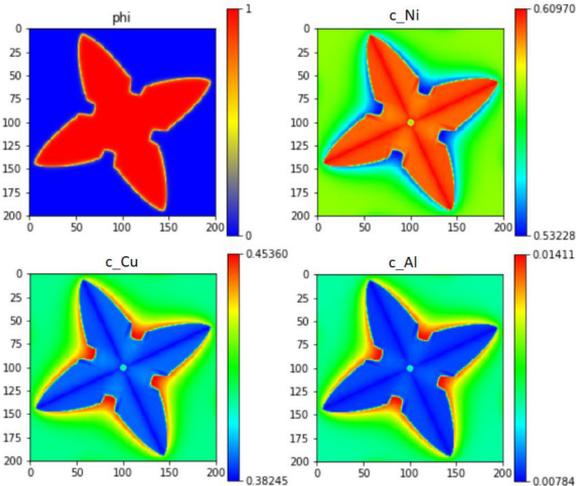
- With PyCalphad/ESPEI-powered UQ, all chemical predictions inherit “error bars”
- Model uncertainty is based on the experimental measurements
- Uncertainty propagation work in collaboration with Argonne National Laboratory
- Free and open source at <https://pduq.readthedocs.io/>

Paulson, N. H., Bocklund, B. J., Otis, R. A., Liu, Z. K., & Stan, M. “Quantified Uncertainty in Thermodynamic Modeling for Materials Design,” *Acta Materialia*, 174 (2019) 9-15.



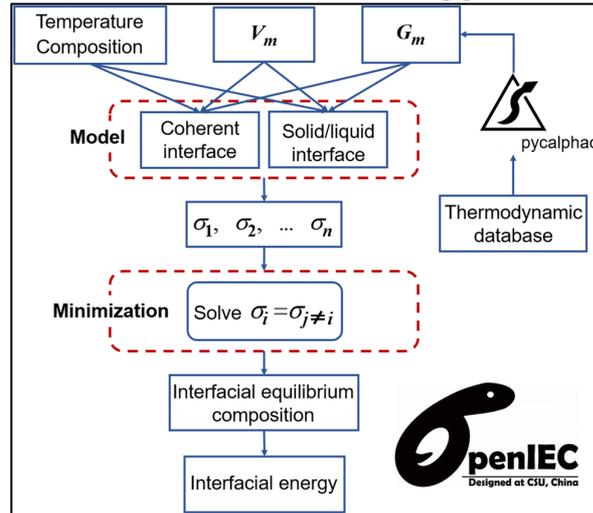
# PyCalphad Community Use Cases Beyond UQ/UP

## Phase-Field



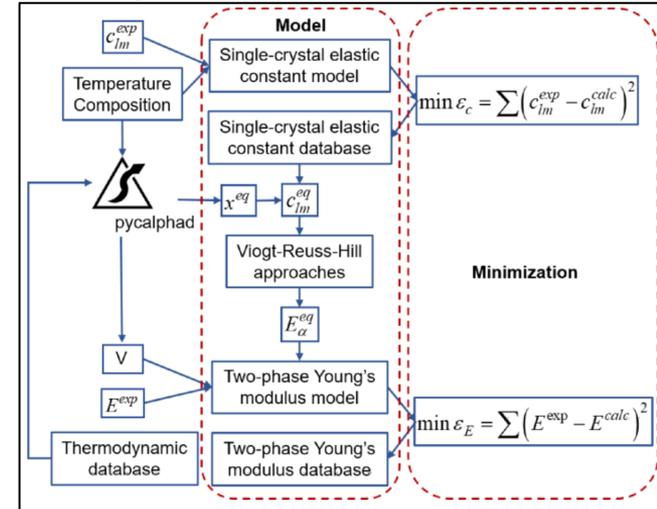
Scott Peters and Prof. Dan Lewis  
Rensselaer Polytechnic Institute  
<https://github.com/AdditiveModeling/SolidificationMicrostructure>

## Interfacial Energy



S. Yang, et al., *J. Mat. Sci.* **54**, 10297–10311 (2019)  
doi: 10.1007/s10853-019-03639-w

## Two-phase Elastic Constants



Y. Shang, et al., *Materialia* **8**, 100500 (2019)  
doi: 10.1016/j.mtla.2019.100500

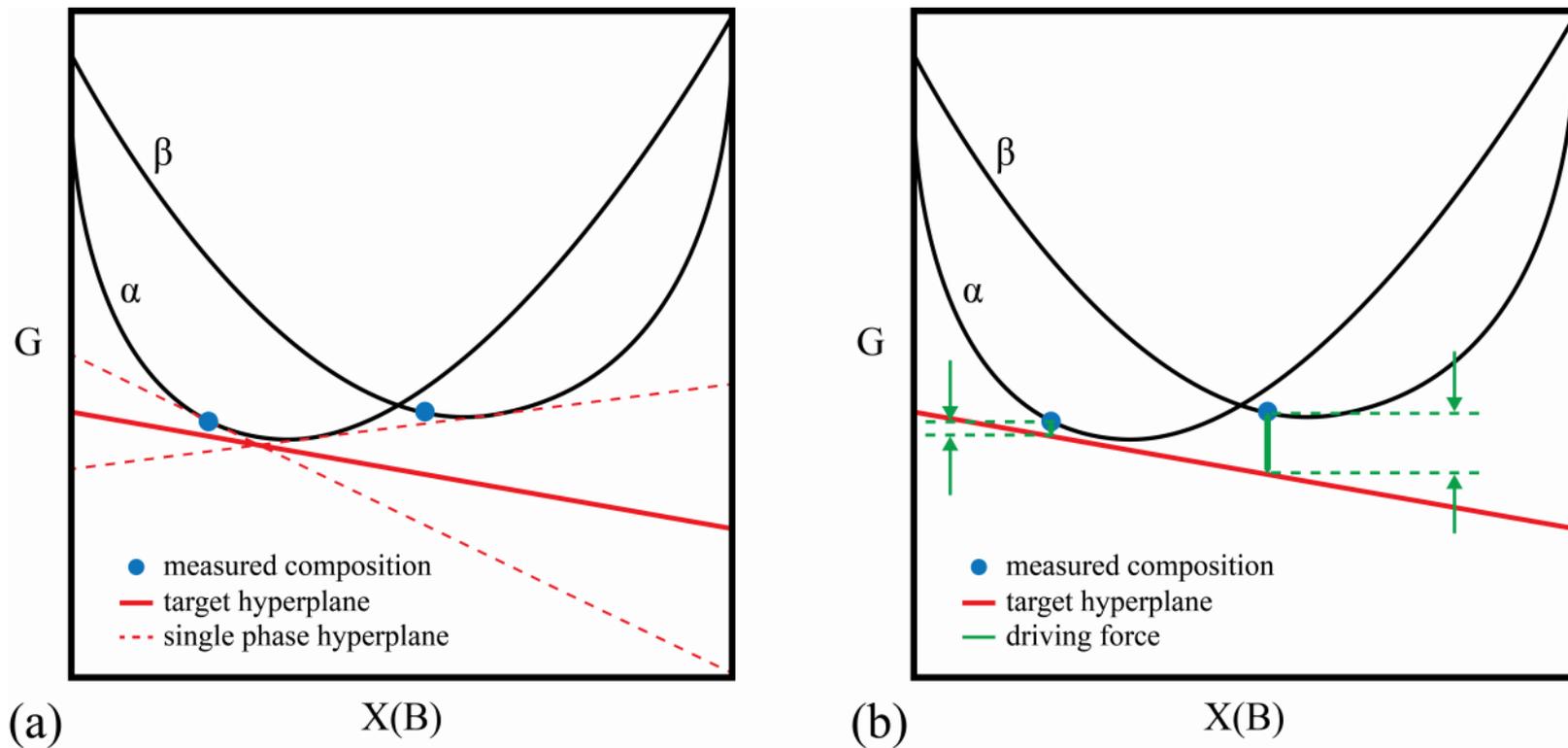
# Motivating Sensitivity for Calphad Modeling

- **Q:** For a given Calphad model, what experiments should be performed to improve the prediction quality?
- In principle, there are maximally-informative experiments for every degree of freedom (the “parameters”) in a Calphad model.
- Quantitative sensitivity estimates tell us how to reduce uncertainty.

# Defining Local Sensitivity of the Model Error

$$S_k = \Delta p_k \frac{\partial L}{\partial p_k}$$

- Sensitivity has two components
  - Parameter Variance: “easy” to guess, or calculate
  - Error Gradient: requires low-level access to Calphad solver



**Figure 1.** Schematic procedure for calculating the error in phase equilibrium data.  $\alpha$  and  $\beta$  are

# Definition of Chemical Potentials

- Hillert (2008, p. 77) derived expressions for the chemical potentials in a few special cases, but not for the general case
- We can consider the Lagrangian definition, where the chemical potentials are the Lagrange multipliers of the mass balance constraints
- For a feasible design point, the first-order optimality condition is obeyed:  $A\mu = \nabla g$  ;  $A$  is the “constraint Jacobian” and  $\nabla g$  is the Gibbs energy gradient
- **How is  $A$  defined?**

## Definition of Chemical Potentials (2)

- Each column of  $A$  relates to a constraint
- Each row of  $A$  relates to a degree of freedom (P,T,yi)
- Example: Consider two-sublattice phase of form (A,B):(A,B)

- $$A^T = \begin{bmatrix} 1 & 1 & & \\ & & 1 & 1 \\ 1 & & 1 & \\ & 1 & & 1 \end{bmatrix}; \mu = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \mu_A \\ \mu_B \end{bmatrix}; (\tau_i \text{ are discarded})$$

- In general,  $A$  can include other constraints, e.g., charge balance and phase amount balance
- $A$  is a function of  $x$  in the case of vacancies or two-sublattice ionic liquids

## Definition of Chemical Potentials (3)

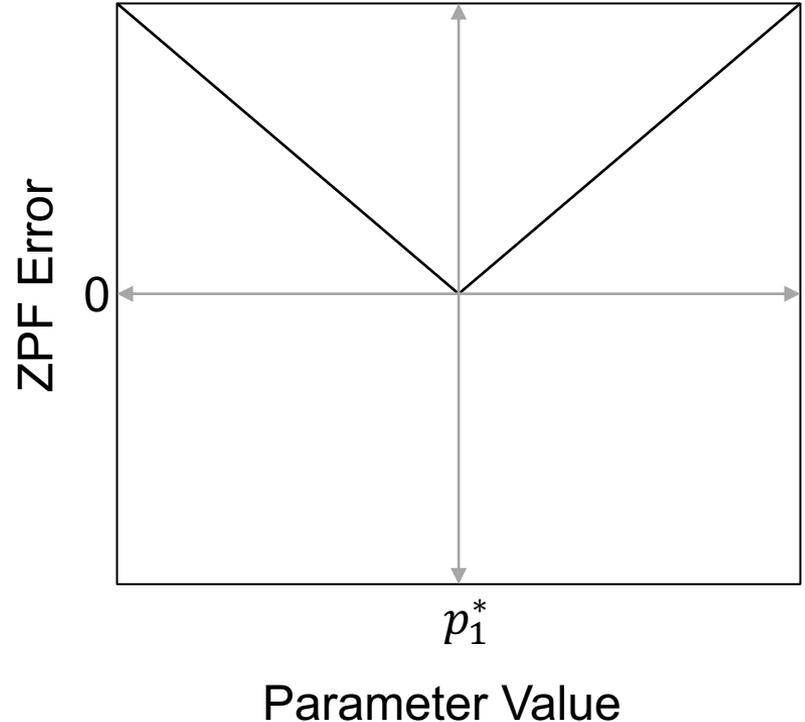
- In general,  $A\mu = \nabla g$  can be an over-determined system if some constraints are redundant
- Choose least-squares solution:  $\mu = (AA^T)^{-1}A\nabla g$
- Assume  $A$  is independent of  $p$  (true for all published models)
- Then  $\frac{\partial\mu}{\partial p} = (AA^T)^{-1}A \frac{\partial(\nabla g)}{\partial p}$
- Defined for every **feasible** design point

# Chemical Potential Gradient

- If we know the equilibrium solution, then we know  $\frac{\partial \mu}{\partial p}$
- This enables expressions for the full likelihood gradient,  $\frac{\partial L}{\partial p_k}$
- $S_k = \Delta p_k \frac{\partial L}{\partial p_k}$  ; can set  $\Delta p_k$  as standard deviation
- Sensitivity is then defined as the amount the error changes if a parameter is increased by one standard deviation

# Discussion of the Tie-line/ZPF Error

- $\frac{\partial \mu}{\partial p} = (AA^T)^{-1} A \frac{\partial(\nabla g)}{\partial p}$  (for constant  $A$ )
- For fixed X-T-P,  $\frac{\partial(\nabla g)}{\partial p}$  will often be a constant
  - $g = [\dots] + (p_1 + p_2 T)x_A x_B + [\dots]$
- In many cases, the error gradient for a tie-line will be a constant, with a discontinuity at zero
- For ternaries, the behavior is the same



# Conclusion

- CALPHAD model sensitivity estimates can be calculated today, using existing commercial or open-source packages, for simple systems
- In many cases, the sensitivity will have a simple linear form
- Analytic error gradients will enable faster (but still global) database optimization with MCMC
- Sensitivity calculations may help build more robust CALPHAD models in the future



**Jet Propulsion Laboratory**  
California Institute of Technology

---

[jpl.nasa.gov](https://jpl.nasa.gov)

# The Path to AI for Materials

- We have an approach for optimizing materials models, but this is not yet “AI” – a lot of human decision-making is required for each model
- Model structural decisions are discrete actions, with complex interdependencies
  - Sometimes called “researcher degrees of freedom”
- Meta-learning (Wade Shen, Data-Driven Discovery of Models, 2019)

<https://www.youtube.com/watch?v=wNIudAkInvQ>

# AlphaGo (2016)

## ARTICLE

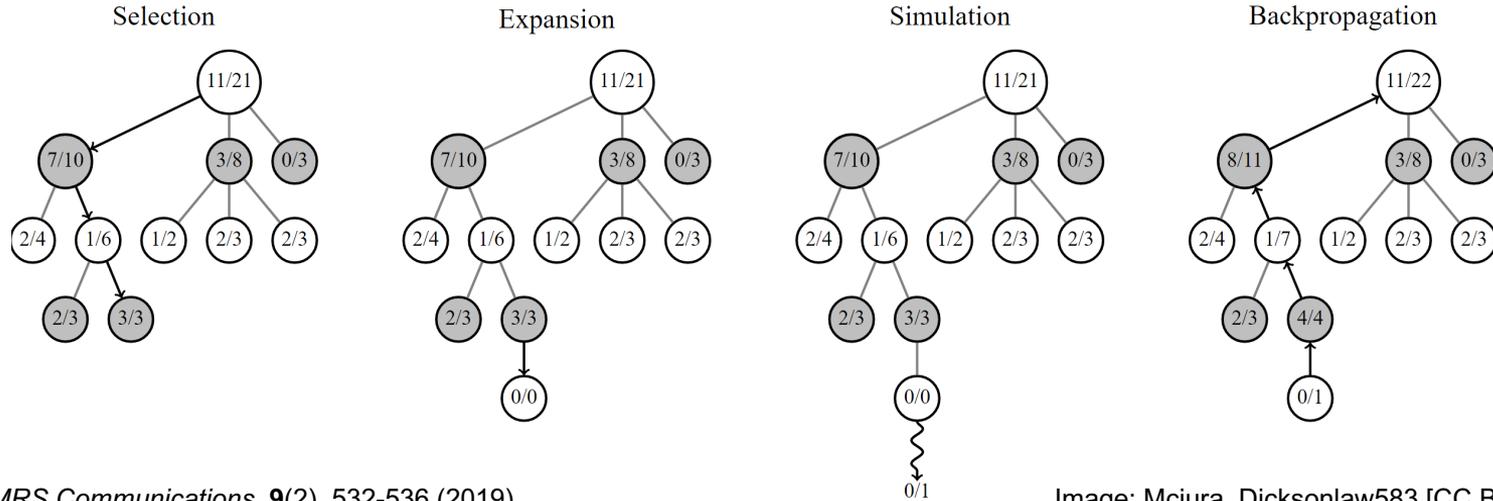
doi:10.1038/nature16961

# Mastering the game of Go with deep neural networks and tree search

David Silver<sup>1\*</sup>, Aja Huang<sup>1\*</sup>, Chris J. Maddison<sup>1</sup>, Arthur Guez<sup>1</sup>, Laurent Sifre<sup>1</sup>, George van den Driessche<sup>1</sup>, Julian Schrittwieser<sup>1</sup>, Ioannis Antonoglou<sup>1</sup>, Veda Panneshelvam<sup>1</sup>, Marc Lanctot<sup>1</sup>, Sander Dieleman<sup>1</sup>, Dominik Grewe<sup>1</sup>, John Nham<sup>2</sup>, Nal Kalchbrenner<sup>1</sup>, Ilya Sutskever<sup>2</sup>, Timothy Lillicrap<sup>1</sup>, Madeleine Leach<sup>1</sup>, Koray Kavukcuoglu<sup>1</sup>, Thore Graepel<sup>1</sup> & Demis Hassabis<sup>1</sup>

The game of Go has long been viewed as the most challenging of classic games for artificial intelligence owing to its enormous search space and the difficulty of evaluating board positions and moves. Here we introduce a new approach to computer Go that uses ‘value networks’ to evaluate board positions and ‘policy networks’ to select moves. These deep neural networks are trained by a novel combination of supervised learning from human expert games, and reinforcement learning from games of self-play. Without any lookahead search, the neural networks play Go at the level of state-of-the-art Monte Carlo tree search programs that simulate thousands of random games of self-play. We also introduce a new search algorithm that combines Monte Carlo simulation with value and policy networks. Using this search algorithm, our program AlphaGo achieved a 99.8% winning rate against other Go programs, and defeated the human European Go champion by 5 games to 0. This is the first time that a computer program has defeated a human professional player in the full-sized game of Go, a feat previously thought to be at least a decade away.

# Monte Carlo Tree Search (MCTS)



T. Dieb, et al. *MRS Communications*, 9(2), 532-536 (2019)  
doi:10.1557/mrc.2019.40

Image: Mciura, Dicksonlaw583 [CC BY-SA 4.0]  
[https://en.wikipedia.org/wiki/Monte\\_Carlo\\_tree\\_search](https://en.wikipedia.org/wiki/Monte_Carlo_tree_search)

## Key Concepts

**MCMC:** Continuous action space (parameter values)

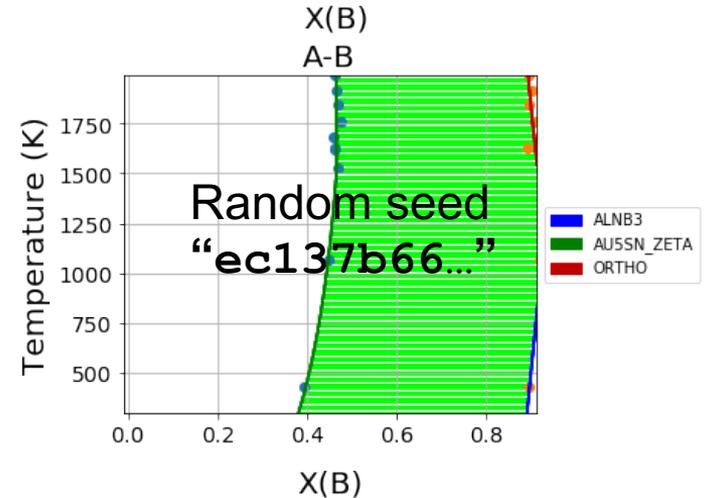
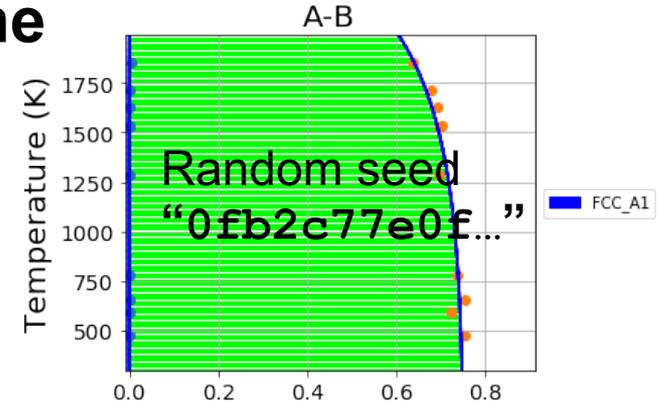
**MCTS:** Discrete action space (game moves)

**“Policy function”:** How the AI (the “agent”) decides

# “TDB Mixer”: How to Teach the Game

1. Uses statistics from TDBDB dataset [\*]
2. Randomly chooses phases, sublattice model, and parameters
3. Generates a TDB
4. Calculates the phase diagram
5. Adds noise to the calculated predictions
6. Writes ESPEI JSON

[\*] A. van de Walle, et al., *Calphad* **61**, 173-178 (2018)  
doi: 10.1016/j.calphad.2018.04.003



# A Framework for Materials AI

- Contemporary Approach
  - Single-shot optimization without model search
- Proposed
  - Design “game” corresponding to modeling task
  - Train on a class of synthetic problems
  - MCTS for model search
  - Apply policy for starting model (fast)
  - MCMC to be quantitative, and to get UQ
  - For future problems, repeat only last two steps