SAND2023-06071C

Sandia National Laboratories

Exceptional service in the national interest

OPTIMIZATION TO GENERATE EQUATIONS OF STATE FOR HYDROGEN PRODUCTION

Intern Summer Symposium – July 20th, 2023

Samantha Yang – Org. 8351

Mathematics and Statistics @ McMaster University, Canada

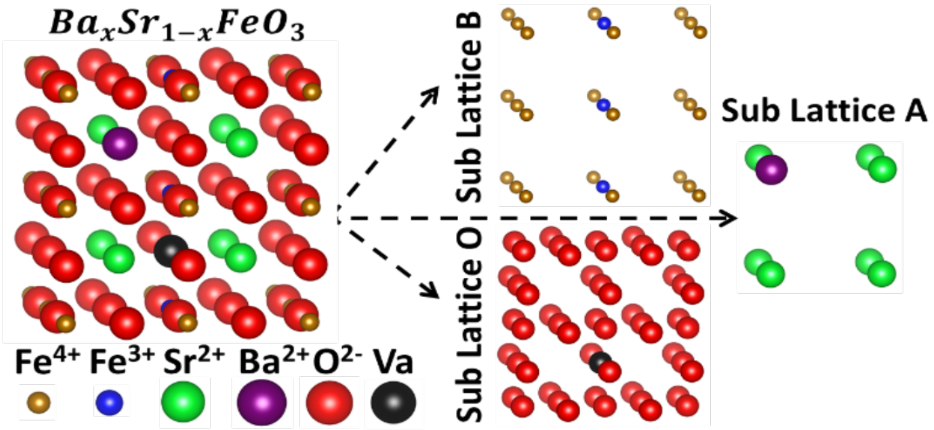
Mentored by Bert Debusschere

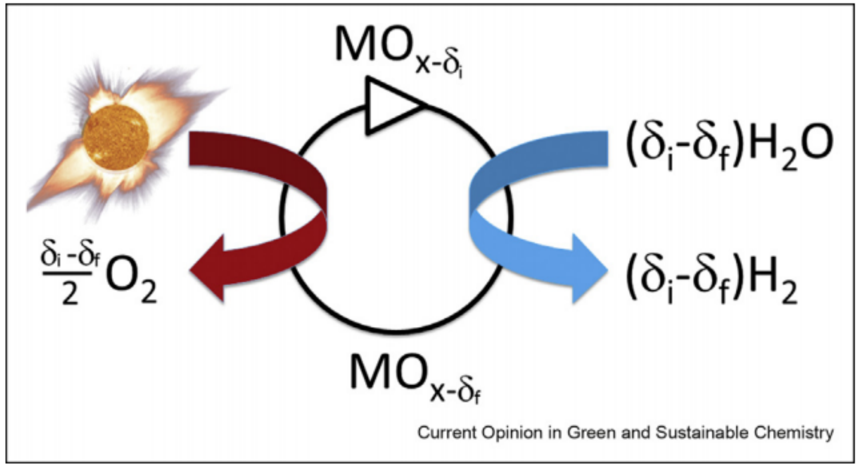
Controlled by:

PROJECT OVERVIEW

PROJECT AIM: DETERMINE CATALYSTS FOR WATER SPLITTING

- Development of software to create an equation of state (EOS) for water splitting catalysts
- Compute efficiency of process
- Perovskites are used
- Goal – to optimize catalyst configuration





Schematic of a general two-step solar powered thermochemical cycle for splitting water using nonstoichiometric metal oxides.

McDaniel, *Current Opinion in Green and Sustainable Chemistry*. 2017

CURRENT CHALLENGES TO THE PROJECT

- Not a lot of experimental data available
- DFT formulation, experimental uncertainty
- Nonlinearities in equations
- Many solutions
- Trouble converging
- Picking the right combination of terms in formulation

Summer project focus – solving equations

EQUATIONS – FULL PROBLEM

$$G_0^{\mathrm{soln}} = G^{\mathrm{endmembers}} - TS_{\mathrm{config}}$$

Thermodynamics to fit endmembers – Gibbs free energy required to get enthalpy and entropy

$$G_0^{\mathrm{soln}} = E_{\mathrm{DFT}}$$

Linear equation

$$\frac{\partial G_0^{\mathrm{soln}}(T,\delta,x)}{\partial \delta} = -\mu_O^{\mathrm{gas,exp}}$$

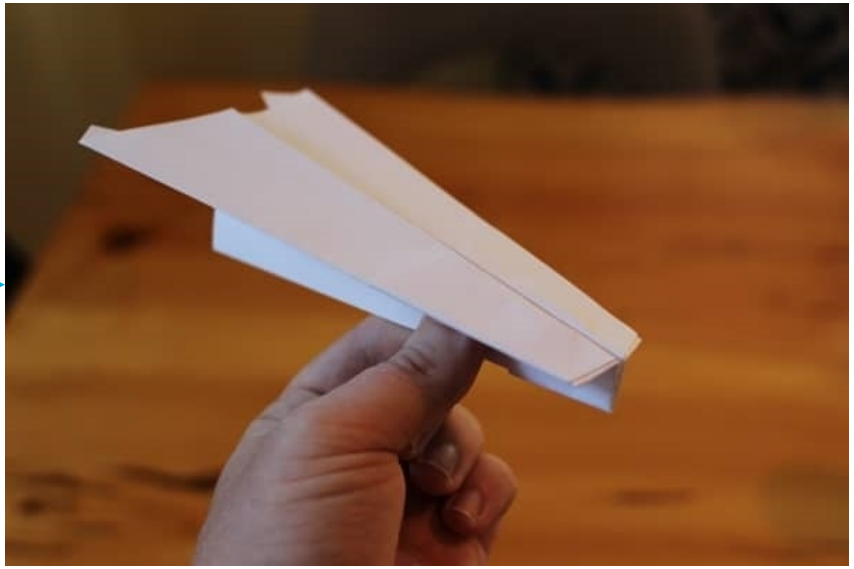
Has nonlinearities, difficult to solve

See also: Wilson SA, Stechel EB, Muhich CL. 2023. Overcoming significant challenges in extracting off-stoichiometric thermodynamics using the compound energy formalism through complementary use of experimental and first principles data: A case study of $Ba_{1-x}Sr_xFeO_{3-\delta}$. Solid State Ionics. Vol. 390.

MY WORK WITH OPTIMIZATION

TOY PROBLEM – SUMMER PROJECT OVERVIEW

- Based on original problem
- Simplified down to the main focus
- Known solution, generated data
- Easily increase complexity as solver gets working



PROBLEM SETUP – OVERALL EQUATIONS

$$\underset{m\times n}{A} \cdot \underset{n\times 1}{c} = \underset{m\times 1}{b}$$

$$Ba_xSr_{1-x}FeO_{3-\delta}$$

m = number of data, n = number of coefficients

Parameters:

- x: scalar, 0-1
- T: vector with dimensions [nData,1]
- δ_0: set to 0.02
- c exact: random vector on the order of 10^{-3} to 10^{-5}
- T_{ref}: reference temperature = 500 K
- $\Delta\delta$: vector with dimensions [nData,1], <0.5

Variables: c guess and δ

EQUATIONS IN MORE DETAIL

Nonlinear constraint: linear in c, nonlinear in δ

$$\frac{\partial G}{\partial \delta}(x, T, \delta_0, c) = -\mu_{O_2, g_{ref}}$$

$$\begin{bmatrix} \frac{\partial G}{\partial \delta} \\ \vdots \end{bmatrix} = A(x, T, \delta) \cdot c = -\mu_{O_2, g_{ref}}$$

$$\underset{m \times n}{A} \cdot \underset{n \times 1}{c} = \underset{m \times 1}{b}$$

$$\delta = \delta_0 + \Delta\delta$$

$$L_i = A_i + B_i \times T + C_i \times T log(T)$$

$$A, B, C = f(x, \delta)$$

A, computed at each data point

ABOUT PYOMO: PYTHON OPTIMIZATION MODELING OBJECTS

- Python-based
- Open source software for optimization models
- Using Ipopt solver – interior point optimizer
  - Compare to the existing two solvers
  - Better functionality, but refactoring code required

PYOMO

PYOMO MODEL COMPONENTS

Variables: changing parts, to be solved for

Parameters: data required for optimization to occur

Constraints: equations, inequalities, or other relations connecting parts of the model

Objective: function to be minimized or maximized

```
def fit_loss(m,c):
    error = 0
    for i in range(m.nData()):
        cABC = dG_dy_matrices(m,m.d+m.dd_values[i],m.T_values[i])
        # value = np.dot(cABC,c)
        value = pyo.sum_product(cABC,c,index=range(3*m.nCoeffs()))
        error += (m.b()[i]-value)**2
    v_lambda = 1.e-3
    error += pyo.sum_product(c,c)*v_lambda
    return error
```

```
m = pyo.ConcreteModel('Toy problem simplified')

m.nData = pyo.Param(
    initialize = 100,
    doc="Number of data points",
    domain=pyo.PositiveIntegers,
    mutable=True
    )
m.x = pyo.Param(
    # initialize=np.random.rand(), # fix this to one value
    initialize=0.5,
    doc="Site fraction, ranges 0-1"
    )
```

```
m.d = pyo.Var( # guess for d0
    initialize = 0.001,
    bounds = (0,0.5),
    doc="delta_0 variable value to solve for"
)
```

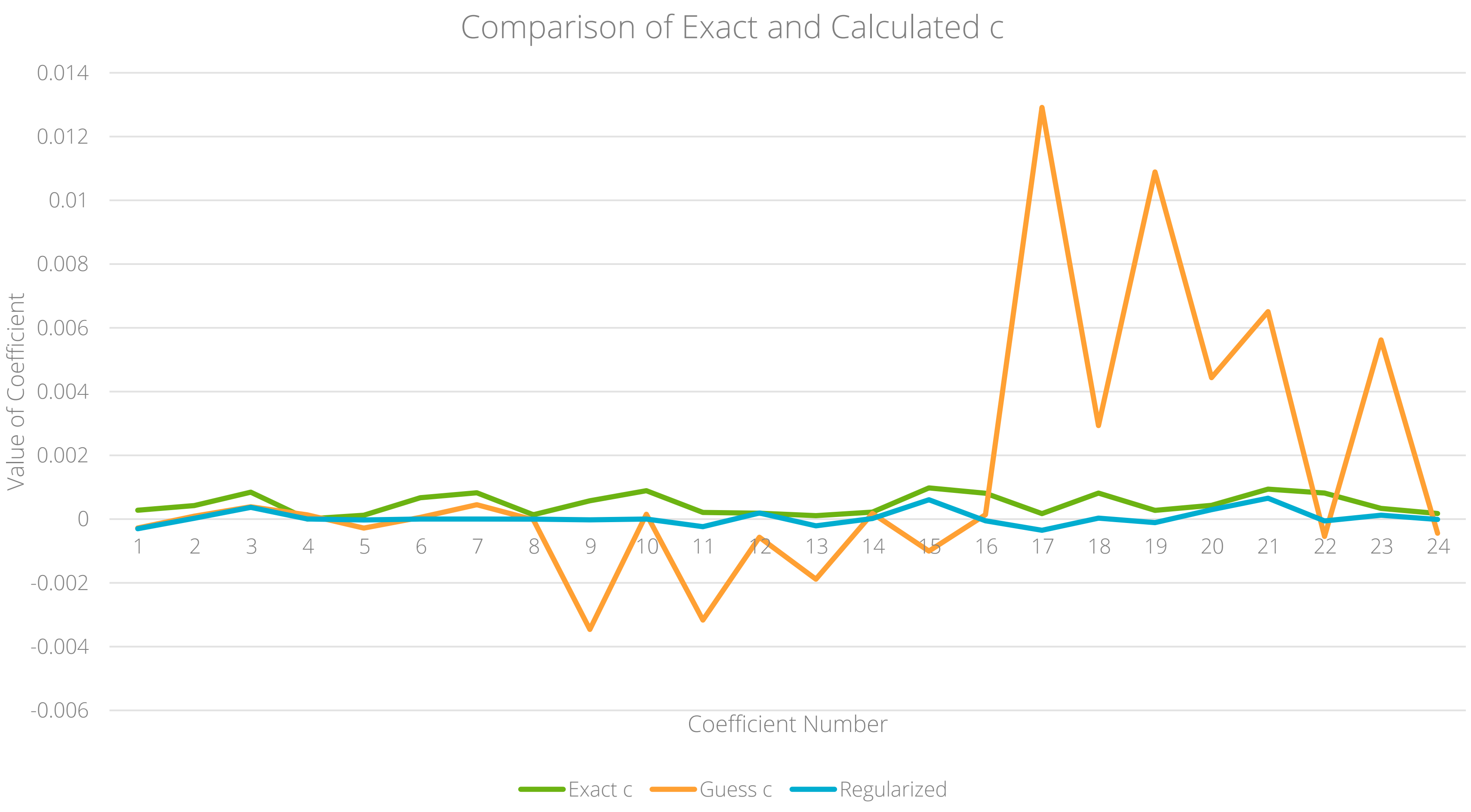
```
eqns = calculate_d0(m)

m.cons = pyo.ConstraintList()
m.cons.add(eqns == 0) # want zero error
```

```
m.obj = pyo.Objective(
    expr=fit_loss(m,m.c_guess),
    sense=pyo.minimize
)
```

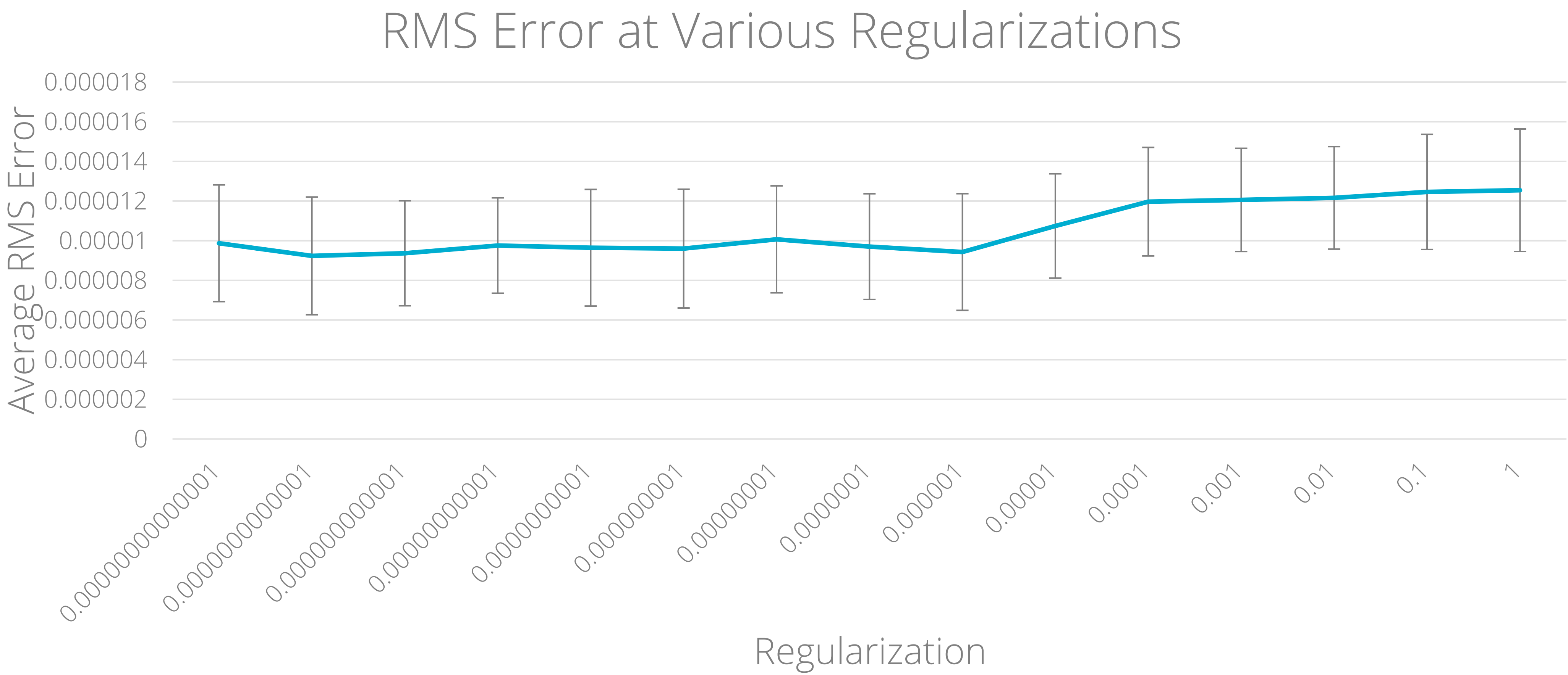
RESULTS: ASSESSING THE SOLVER'S ACCURACY

Using 24 coefficients:

- Low loss function with and without regularization
- Regularization of 1.e-3 added to select the smallest solution



COMPARISON OF VARYING LEVELS OF REGULARIZATION



By repeatedly running the solver, an ideal level of regularization can be selected.
At zero regularization, the average RMS error is 3680417.

NEXT STEPS OF ACTION

- Optimizing the regularization
- Examining the accuracy of $\Delta\delta$ values
- Improving solver accuracy – refining solver parameters
- Lastly – increasing complexity

ACKNOWLEDGEMENTS

Thank you to Bert Debusschere for excellent mentorship and support this summer, and to Tony McDaniel for all the work done for this project. Special thanks to Soraya Rawlings for helping us get started with Pyomo and guiding us along the way.

Funding for this project was provided by the DOE Hydrogen and Fuel Cell Technologies Office.