UCONN
UNIVERSITY OF CONNECTICUT

Automatic Generation of JuMP.jl Constraints from ModelingToolkit.jl Models

Dimitri Alston, Joseph Choi, Pengfei Xu, Robert Gottlieb
Matthew Stuber, P&W Associate Professor in Advanced Systems Engineering

November 18th, 2025

JUMP

Process Systems and Operations Research Laboratory

Motivation

Dynamic Optimization

$$\min_{\mathbf{p}} \phi(\mathbf{p},t) = \sum_{i=0}^{N} (I_i^{calc} - I_i^{exp})^2$$

$$\text{s.t.} \;\; \mathbf{p} \in [\mathbf{p}^L, \mathbf{p}^U]$$

$$I_i^{calc} = x_{A,i} + \frac{2}{21} x_{B,i} + \frac{2}{21} x_{D,i}$$

$$\frac{dx_A}{dt} = k_1 x_Z x_Y - c_{O_2}(k_{2f} + k_{3f}) x_A + \frac{k_{2f}}{K_2} x_D + \frac{k_{3f}}{K_3} x_B - k_5 x_A^2$$

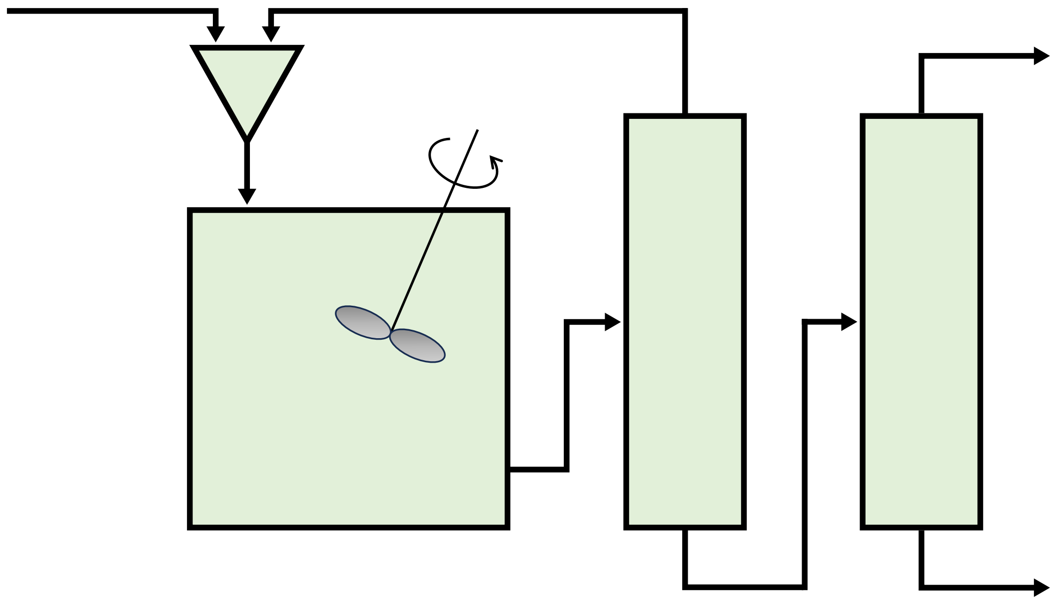
$$\frac{dx_B}{dt} = c_{O_2} k_{3f} x_A - \left(\frac{k_{3f}}{K_3} + k_4 \right) x_B$$

$$\frac{dx_D}{dt} = c_{O_2} k_{2f} x_A - \frac{k_{2f}}{K_2} x_D$$

$$\frac{dx_Y}{dt} = -k_{1s} x_Z x_Y$$

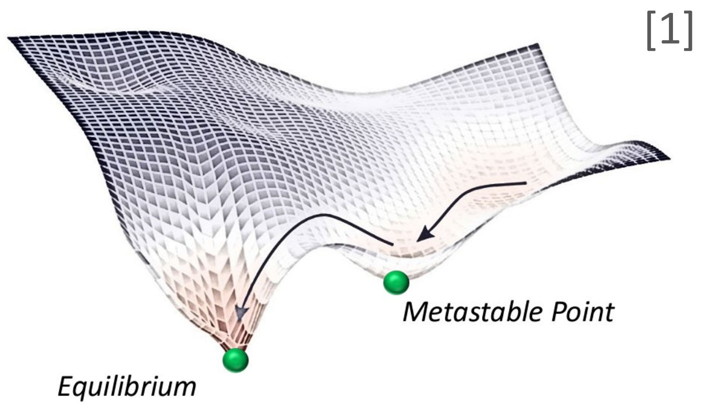
$$\frac{dx_Z}{dt} = -k_1 x_Z x_Y$$

Process Flow Sheets



Deterministic Global Optimization

- Nonconvex problems arise in many applications
  - Thermodynamic stability
  - Kinetic parameter estimation
  - Advanced control systems
  - Design under uncertainty
  - Etc.



$$
\begin{aligned}
&\min_{\mathbf{p}\in P} \phi(\mathbf{x}(\mathbf{p},t_f),\mathbf{p}) \\
&\text{s.t.} \quad \dot{\mathbf{x}}(\mathbf{p},t) = \mathbf{f}(\mathbf{x}(\mathbf{p},t),\mathbf{p},t) = \mathbf{0} \quad \forall t \in I = [t_0, t_f] \\
&\qquad \mathbf{x}(\mathbf{p},t_0) = \mathbf{x_0}(\mathbf{p}) \\
&\qquad \mathbf{g}(\mathbf{x}(\mathbf{p},t),\mathbf{p}) \le \mathbf{0} \\
&\qquad P = \{\mathbf{p} \in \mathbb{R}^m : \mathbf{p}^L \le \mathbf{p} \le \mathbf{p}^U\}
\end{aligned}
$$

[1] Grajcarova, L. Simulations of structural phase transitions in crystals using ab initio metadynamics. INIS-IAEA. (2013).

Kinetic Parameter Estimation

$$\min_{\mathbf{p}} \phi(\mathbf{p},t) = \sum_{i=0}^{N} (I_i^{calc} - I_i^{exp})^2$$

$$\text{s.t.} \;\; \mathbf{p} \in [\mathbf{p}^L, \mathbf{p}^U]$$

$$I_i^{calc} = x_{A,i} + \frac{2}{21} x_{B,i} + \frac{2}{21} x_{D,i}$$

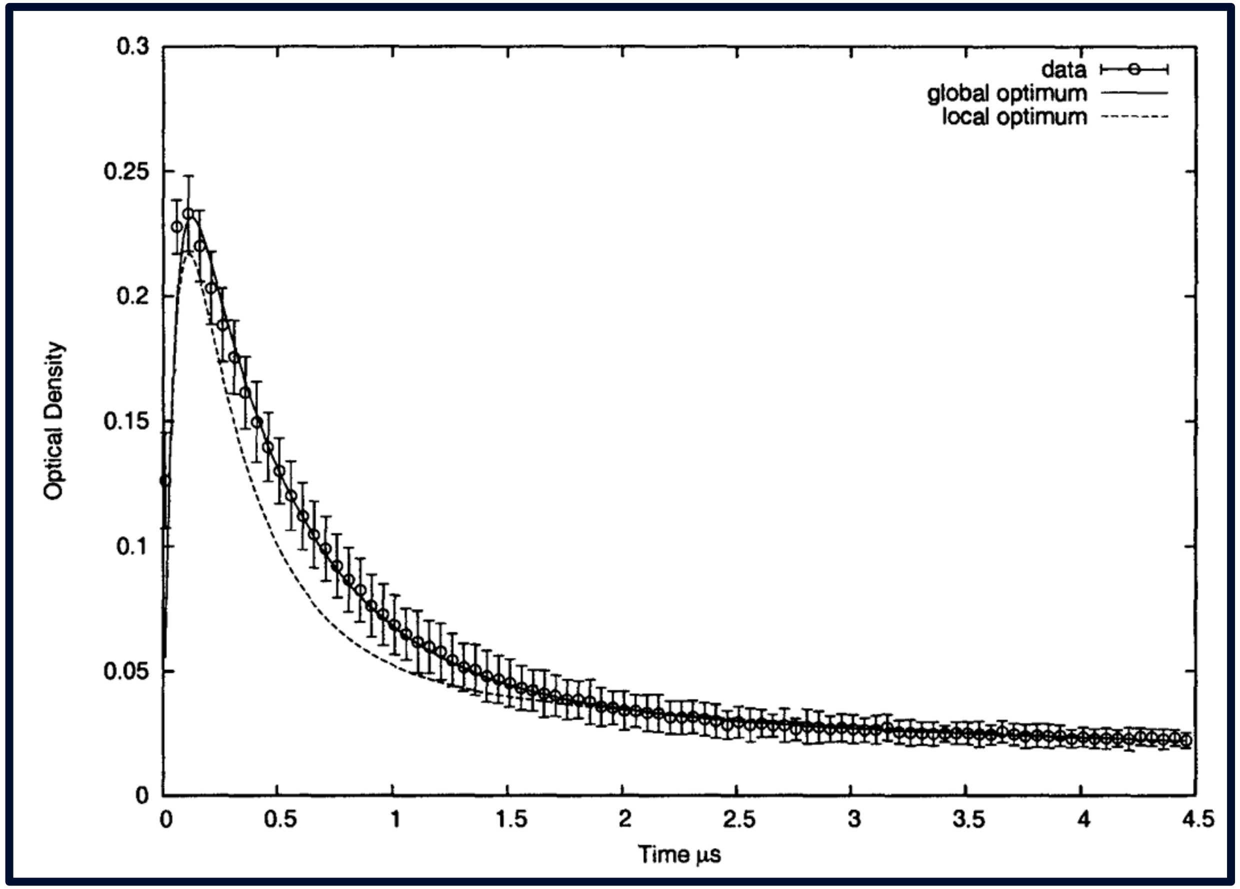
$$\frac{dx_A}{dt} = k_1 x_Z x_Y - c_{O_2}(\boxed{k_{2f}} + \boxed{k_{3f}}) x_A + \frac{\boxed{k_{2f}}}{K_2} x_D + \frac{\boxed{k_{3f}}}{K_3} x_B - k_5 x_A^2$$

$$\frac{dx_B}{dt} = c_{O_2} \boxed{k_{3f}} x_A - \left(\frac{\boxed{k_{3f}}}{K_3} + \boxed{k_4} \right) x_B$$

$$\frac{dx_D}{dt} = c_{O_2} \boxed{k_{2f}} x_A - \frac{\boxed{k_{2f}}}{K_2} x_D$$

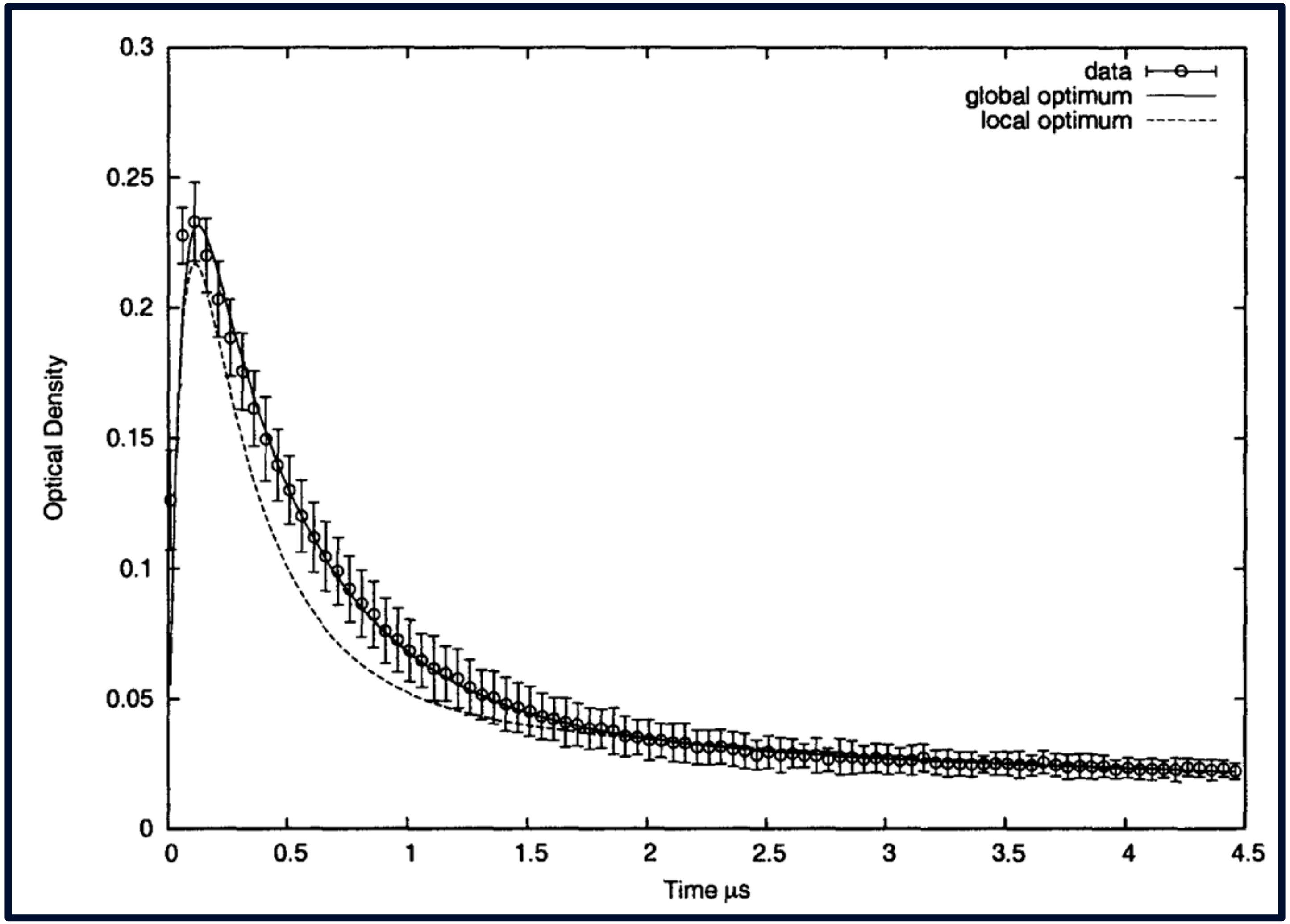
$$\frac{dx_Y}{dt} = -k_{1s} x_Z x_Y$$

$$\frac{dx_Z}{dt} = -k_1 x_Z x_Y$$



[2] Taylor, J.W., et al. Direct measurement of the fast, reversible addition of oxygen to cyclohexadienyl radicals in nonpolar solvents, The Journal of Physical Chemistry A. 108, 7193-7203 (2004).

Kinetic Parameter Estimation



[2] Taylor, J.W., et al. Direct measurement of the fast, reversible addition of oxygen to cyclohexadienyl radicals in nonpolar solvents, The Journal of Physical Chemistry A. 108, 7193-7203 (2004).

EAGO.jl

Easy Advanced Global Optimization

- Open-source deterministic global solver for nonconvex MINLPs
  - Semi-infinite programs (SIPs)
  - Dynamic optimization
  - User-defined functions
- Uses branch-and-bound (B&B) to guarantee global optimality or infeasibility
- Applies McCormick-based relaxations for convex lower-bounding problems
- Designed in conjunction with JuMP

[3] Wilhelm, M.E. and Stuber, M.D. EAGO.jl: Easy Advanced Global Optimization in Julia. Optimization Methods & Software. (2020).

Recent Advances in EAGO.jl

Easy Advanced Global Optimization

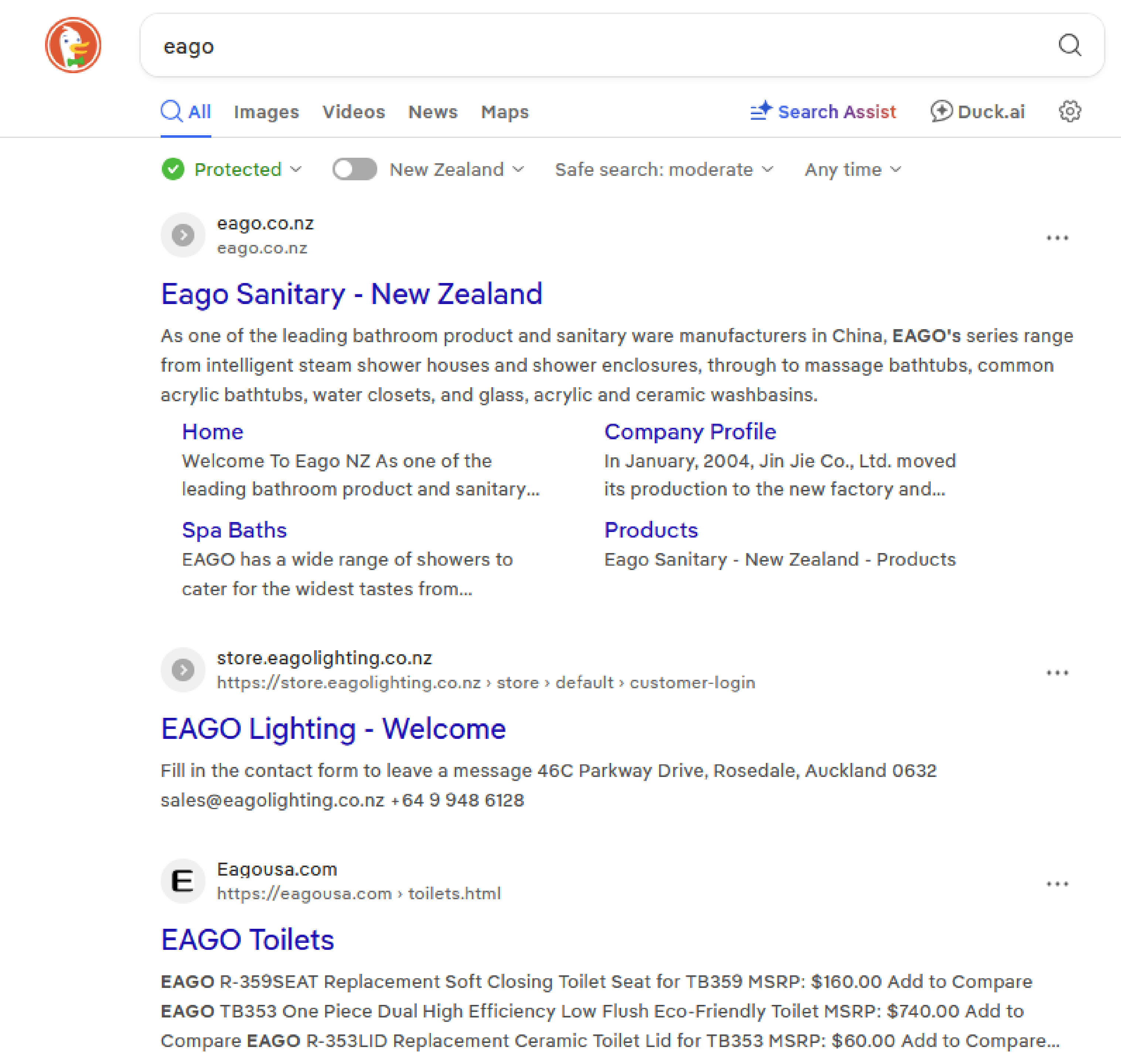
- Improved bilinear relaxations [4]
- Optimization of ANNs [5]
- Custom implementation of PDLP on GPUs

[4] Wilhelm, M.E. and Stuber, M.D. Improved Convex and Concave Relaxations of Composite Bilinear Forms. Journal of Optimization Theory and Applications. 197, 174-204 (2023).
[5] Wang, C., Wilhelm, M.E., and Stuber, M.D. Semi-Infinite Optimization with Hybrid Models. Industrial & Engineering Chemistry Research. 61, 5239-5254 (2022).

Recen

Easy Advanc

- Improved
- Optimizati
- Custom in



[4] Wilhelm, M.E. and Stuber, M.D. ... 23).
[5] Wang, C., Wilhelm, M.E., and St

Should you use JuMP?

When should you not use JuMP?

JuMP supports a broad range of optimization classes. However, there are still some that it doesn't support, or that are better supported by other software packages.

You want to optimize a complicated Julia function

Packages in Julia compose well. It's common for people to pick two unrelated packages and use them in conjunction to create novel behavior. JuMP isn't one of those packages.

If you want to optimize an ordinary differential equation from DifferentialEquations.jl or tune a neural network from Flux.jl, consider using other packages such as:

- Optim.jl
- Optimization.jl
- NLPModels.jl
- Nonconvex.jl

[6] https://jump.dev/JuMP.jl/stable/should_i_use/#When-should-you-not-use-JuMP?

Motivation

Dynamic Optimization

$$\min_{\mathbf{p}} \phi(\mathbf{p},t) = \sum_{i=0}^{N} (I_i^{calc} - I_i^{exp})^2$$

$$\text{s.t. } \mathbf{p} \in [\mathbf{p}^L, \mathbf{p}^U]$$

$$I_i^{calc} = x_{A,i} + \frac{2}{21} x_{B,i} + \frac{2}{21} x_{D,i}$$

$$\frac{dx_A}{dt} = k_1 x_Z x_Y - c_{O_2}(k_{2f} + k_{3f}) x_A + \frac{k_{2f}}{K_2} x_D + \frac{k_{3f}}{K_3} x_B - k_5 x_A^2$$

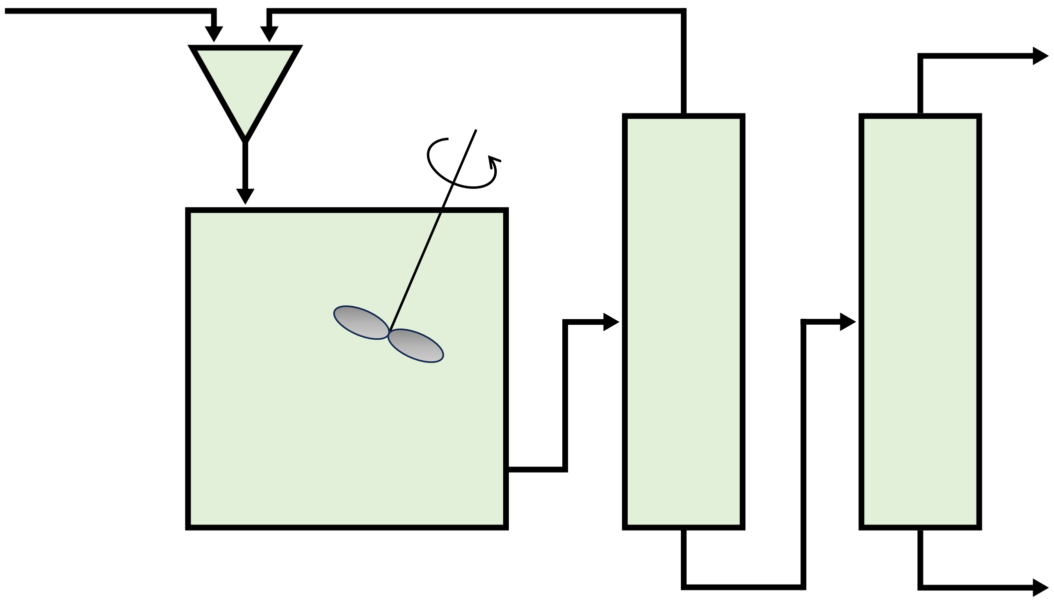
$$\frac{dx_B}{dt} = c_{O_2} k_{3f} x_A - \left(\frac{k_{3f}}{K_3} + k_4\right) x_B$$

$$\frac{dx_D}{dt} = c_{O_2} k_{2f} x_A - \frac{k_{2f}}{K_2} x_D$$

$$\frac{dx_Y}{dt} = -k_{1s} x_Z x_Y$$

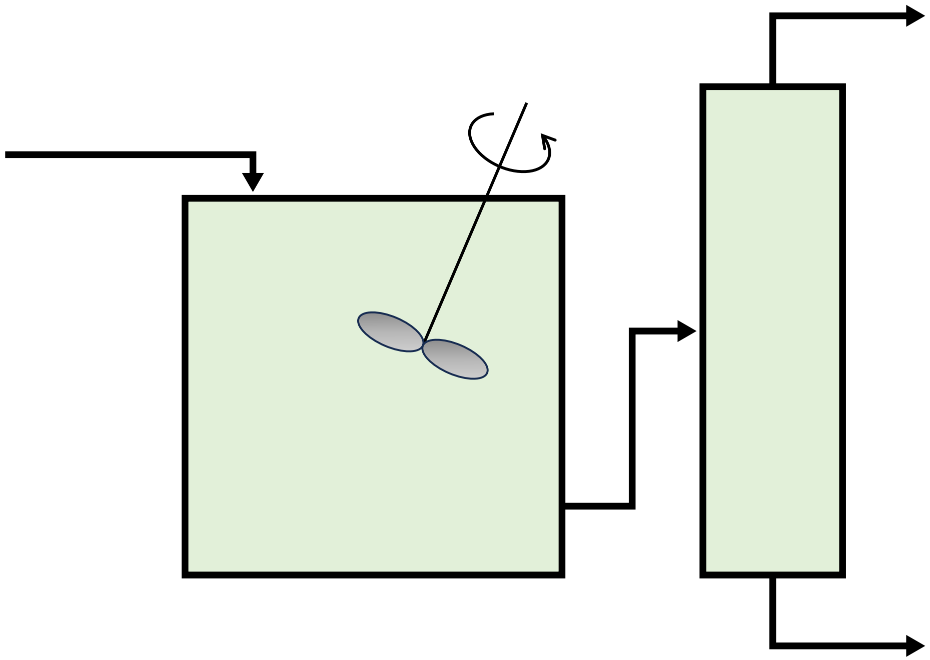
$$\frac{dx_Z}{dt} = -k_1 x_Z x_Y$$

Process Flow Sheets



Causal vs Acausal Modeling

Causal/Sequential-Modular



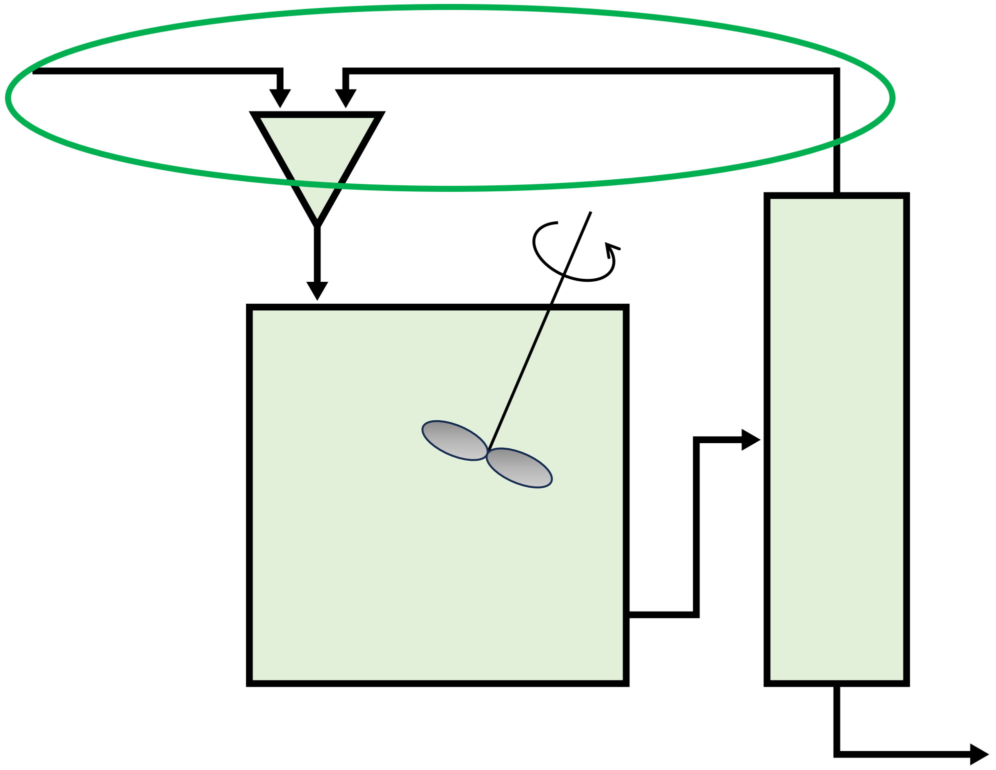
Acausal/Equation-Oriented

$$y_{2,\mathrm{A}} + y_{2,\mathrm{B}} + y_{2,\mathrm{C}} = 1$$
$$y_{2,\mathrm{B}} F = y_{1,\mathrm{B}} + V r_{\mathrm{B}}$$
$$y_{2,\mathrm{C}} F = y_{1,\mathrm{C}} + V r_{\mathrm{C}}$$
$$y_{3,\mathrm{A}} = y_{1,\mathrm{A}} - V r_{\mathrm{A}}$$
$$y_{4,\mathrm{B}} + y_{4,\mathrm{C}} = y_{2,\mathrm{B}} + y_{2,\mathrm{C}}$$

[7] Schweiger, G. et al. Modeling and simulation of large-scale systems: A systematic comparison of modeling paradigms. *Applied Mathematics and Computation.* 365, 124713 (2020).

Causal vs Acausal Modeling

Causal/Sequential-Modular



Acausal/Equation-Oriented

$$y_{3,\mathrm{A}} + y_{3,\mathrm{B}} + y_{3,\mathrm{C}} = 1$$

$$y_{3,\mathrm{B}} F = y_{2,\mathrm{B}} + V r_{\mathrm{B}}$$

$$y_{3,\mathrm{C}} F = y_{2,\mathrm{C}} + V r_{\mathrm{C}}$$

$$y_{4,\mathrm{A}} = y_{2,\mathrm{A}} - V r_{\mathrm{A}}$$

$$y_{5,\mathrm{B}} + y_{5,\mathrm{C}} = y_{3,\mathrm{B}} + y_{3,\mathrm{C}}$$

$$y_{2,\mathrm{A}} = y_{1,\mathrm{A}} + y_{4,\mathrm{A}}$$

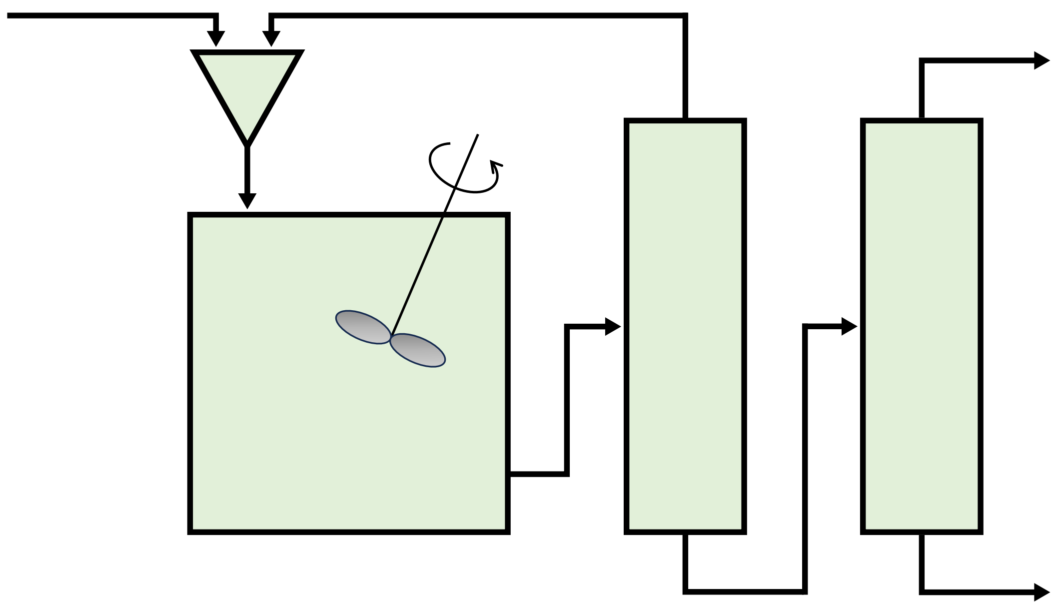
[7] Schweiger, G. et al. Modeling and simulation of large-scale systems: A systematic comparison of modeling paradigms. *Applied Mathematics and Computation.* 365, 124713 (2020).

ModelingToolkit.jl

- Open-source, acausal modeling framework in Julia
- Supports a broad range of system types
  - ODEs, SDEs, PDEs
  - Nonlinear systems
  - Optimization problems
- Automatically composes, transforms, and reduces models
  - Dimensionality reduction through algebraic simplification

[8] Ma, Y. et al. Modelingtoolkit: A composable graph transformation system for equation-based modeling. arXiv preprint arXiv:2103.05244 (2021).

Componentized Model



Should you use JuMP?

When should you not use JuMP?

JuMP supports a broad range of optimization classes. However, there are still some that it doesn't support, or that are better supported by other software packages.

You want to optimize a complicated Julia function

Packages in Julia compose well. It's common for people to pick two unrelated packages and use them in conjunction to create novel behavior. JuMP isn't one of those packages.

If you want to optimize an ordinary differential equation from DifferentialEquations.jl or tune a neural network from Flux.jl, consider using other packages such as:

- Optim.jl
- Optimization.jl
- NLPModels.jl
- Nonconvex.jl

[6] https://jump.dev/JuMP.jl/stable/should_i_use/#When-should-you-not-use-JuMP?

Optimization.jl

Modeling Optimization Problems

ModelingToolkit.jl is not only useful for generating initial value problems (`ODEProblem`). The package can also build optimization systems.

> **Note**
>
> The high level `@mtkmodel` macro used in the getting started tutorial is not yet compatible with `OptimizationSystem`. We thus have to use a lower level interface to define optimization systems. For an introduction to this interface, read the programmatically generating Systems tutorial.

[9] https://docs.sciml.ai/ModelingToolkit/stable/tutorials/optimization/

Optimization in Julia

SciML
[10]

- 25+ libraries, 100+ solvers
- No deterministic global optimization support

JuMP
[11]

- Algebraic modeling language designed for mathematical optimization

[10] Dixit, V.K. and Rackauckas C. Optimization.jl: A Unified Optimization Package. (2023).
[11] Lubin, M., Dowson, O., Garcia, J.D. et al. JuMP 1.0: recent improvements to a modeling language for mathematical optimization. *Math. Prog. Comp*. 15, 581–589 (2023).

Optimization in Julia

SciML [10]

- 25+ libraries, 100+ solvers
- No deterministic global optimization support

EOptInterface
Abstraction layer

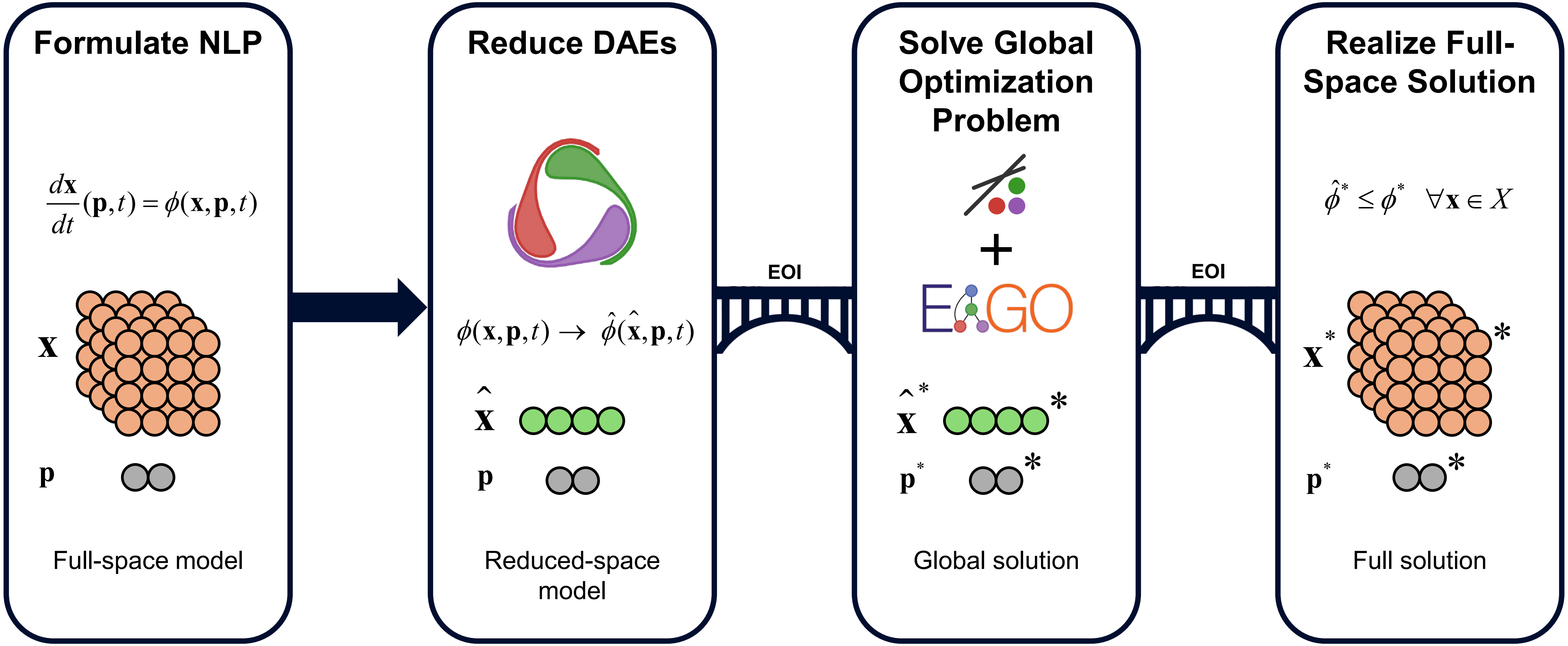
JuMP [11]

- Algebraic modeling language designed for mathematical optimization

[10] Dixit, V.K. and Rackauckas C. Optimization.jl: A Unified Optimization Package. (2023).
[11] Lubin, M., Dowson, O., Garcia, J.D. et al. JuMP 1.0: recent improvements to a modeling language for mathematical optimization. *Math. Prog. Comp*. 15, 581–589 (2023).

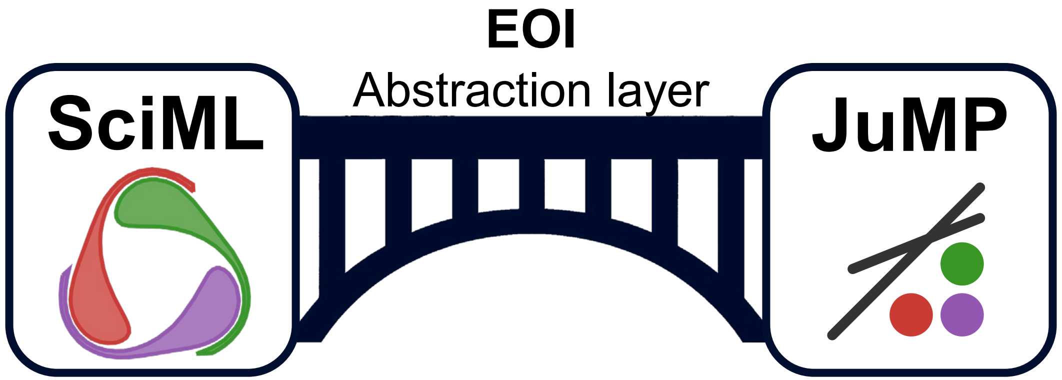
EOptInterface.jl

Equation-oriented Optimization Interface (EOI)



EOptInterface Features

- Automatically generate JuMP constraints from ModelingToolkit models
- Display decision variables
- Detect and directly transcribe ODEs
- Calculate full-space solutions



Case Studies

1) Dynamic Kinetic Parameter Estimation

2) Steady-State Process Flow Sheet

3) Nonlinear Model Predictive Control (NMPC)

Dynamic Kinetic Parameter Estimation

$$\min_{\mathbf{p}} \phi(\mathbf{p}, t) = \sum_{i=0}^{N} (I_i^{calc} - I_i^{exp})^2$$

$$\text{s.t.} \ \ \mathbf{p} \in [\mathbf{p}^L, \mathbf{p}^U]$$

$$\boxed{\begin{aligned}
I_i^{calc} &= x_{A,i} + \frac{2}{21} x_{B,i} + \frac{2}{21} x_{D,i} \\
\frac{dx_A}{dt} &= k_1 x_Z x_Y - c_{O_2} (k_{2f} + k_{3f}) x_A + \frac{k_{2f}}{K_2} x_D + \frac{k_{3f}}{K_3} x_B - k_5 x_A^2 \\
\frac{dx_B}{dt} &= c_{O_2} k_{3f} x_A - \left(\frac{k_{3f}}{K_3} + k_4 \right) x_B \\
\frac{dx_D}{dt} &= c_{O_2} k_{2f} x_A - \frac{k_{2f}}{K_2} x_D \\
\frac{dx_Y}{dt} &= -k_{1s} x_Z x_Y \\
\frac{dx_Z}{dt} &= -k_1 x_Z x_Y
\end{aligned}}$$

[12] Mitsos, A., Chachuat, B., and Barton, P.I. McCormick-based relaxations of algorithms. *SIAM Journal on Optimization, SIAM.* 20:2, 573-601 (2009).

Dynamic Kinetic Parameter Estimation

$$\min_{\mathbf{p}} \phi(\mathbf{p},t) = \sum_{i=0}^{N} (I_i^{calc} - I_i^{exp})^2$$

$$\text{s.t.} \;\; \mathbf{p} \in [\mathbf{p}^L, \mathbf{p}^U]$$

$$I_i^{calc} = x_{A,i} + \frac{2}{21} x_{B,i} + \frac{2}{21} x_{D,i}$$

$$\frac{dx_A}{dt} = k_1 x_Z x_Y - c_{O_2}(k_{2f} + k_{3f}) x_A + \frac{k_{2f}}{K_2} x_D + \frac{k_{3f}}{K_3} x_B - k_5 x_A^2$$

$$\frac{dx_B}{dt} = c_{O_2} k_{3f} x_A - \left(\frac{k_{3f}}{K_3} + k_4 \right) x_B$$

$$\frac{dx_D}{dt} = c_{O_2} k_{2f} x_A - \frac{k_{2f}}{K_2} x_D$$

$$\frac{dx_Y}{dt} = -k_{1s} x_Z x_Y$$

$$\frac{dx_Z}{dt} = -k_1 x_Z x_Y$$

```julia
@mtkmodel KineticParameterEstimation begin
    @parameters begin
        T = 273.0
        K_2 = 46.0*exp(6500.0/T - 18.0)
        K_3 = 2.0*K_2
        k_1 = 53.0
        k_1s = k_1*1e-6
        k_5 = 1.2e-3
        c_O2 = 2e-3

        k_2f
        k_3f
        k_4
    end
    @variables begin
        x_A(t) = 0.0
        x_B(t) = 0.0
        x_D(t) = 0.0
        x_Y(t) = 0.4
        x_Z(t) = 140.0
        I(t)
    end
    @equations begin
        D(x_A) ~ k_1*x_Z*x_Y - c_O2*(k_2f + k_3f)*x_A + k_2f/K_2*x_D + k_3f/K_3*x_B - k_5*x_A^2
        D(x_B) ~ c_O2*k_3f*x_A - (k_3f/K_3 + k_4)*x_B
        D(x_D) ~ c_O2*k_2f*x_A - k_2f/K_2*x_D
        D(x_Y) ~ -k_1s*x_Z*x_Y
        D(x_Z) ~ -k_1*x_Z*x_Y
        I ~ x_A + 2/21*x_B + 2/21*x_D
    end
end
```

[12] Mitsos, A., Chachuat, B., and Barton, P.I. McCormick-based relaxations of algorithms. *SIAM Journal on Optimization, SIAM.* 20:2, 573-601 (2009).

Dynamic Kinetic Parameter Estimation

- EOptInterface

```
if integrator == "Explicit Euler"
    @constraint(model, [j in 1:V, i in 1:(N-1)], xs[j,i+1] == xs[j,i] + t_step*dx[j](xs[:,i]..., ps...))
elseif integrator == "Implicit Euler"
    @constraint(model, [j in 1:V, i in 1:(N-1)], xs[j,i+1] == xs[j,i] + t_step*dx[j](xs[:,i+1]..., ps...))
```

Dynamic Kinetic Parameter Estimation

- EOptInterface

```
if integrator == "Explicit Euler"
    @constraint(model, [j in 1:V, i in 1:(N-1)], xs[j,i+1] == xs[j,i] + t_step*dx[j](xs[:,i]..., ps...))
elseif integrator == "Implicit Euler"
    @constraint(model, [j in 1:V, i in 1:(N-1)], xs[j,i+1] == xs[j,i] + t_step*dx[j](xs[:,i+1]..., ps...))
```

- InfiniteOpt

```
problem = JuMPDynamicOptProblem(system, [u0_map; p_map], (t_start, t_end); dt = 0.001)
solution = solve(problem, JuMPCollocation(Ipopt.Optimizer, constructRK4()))

problem = InfiniteOptDynamicOptProblem(system, [u0_map; p_map], t_span; steps = 100)
solution = solve(problem, InfiniteOptCollocation(Ipopt.Optimizer))
```

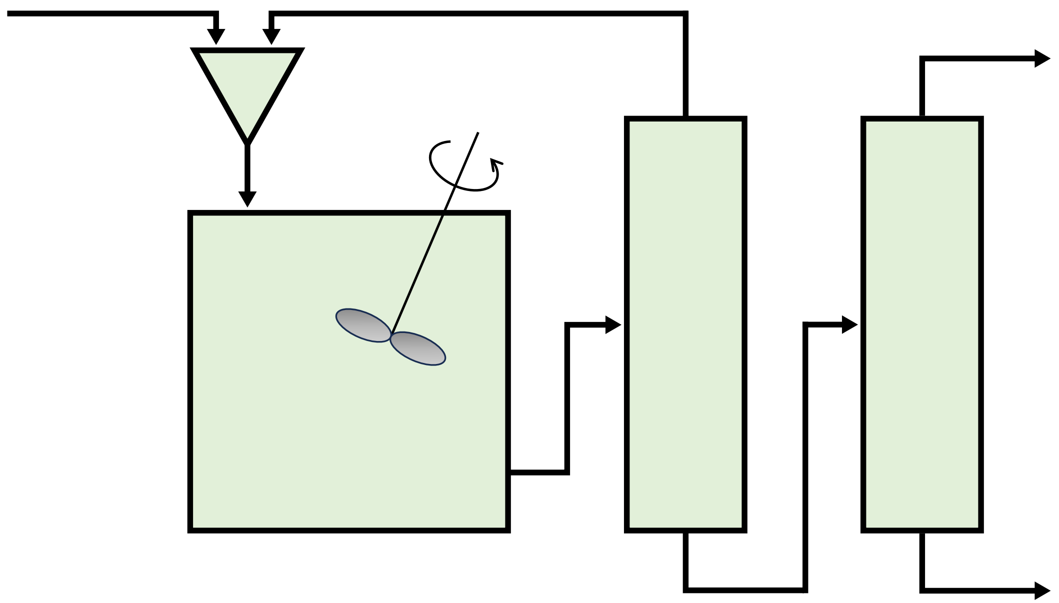
Case Studies

1) Dynamic Kinetic Parameter Estimation
    - Direct transcription of ODEs
2) Steady-State Process Flow Sheet
3) Nonlinear Model Predictive Control (NMPC)

Steady-State Process Flow Sheet

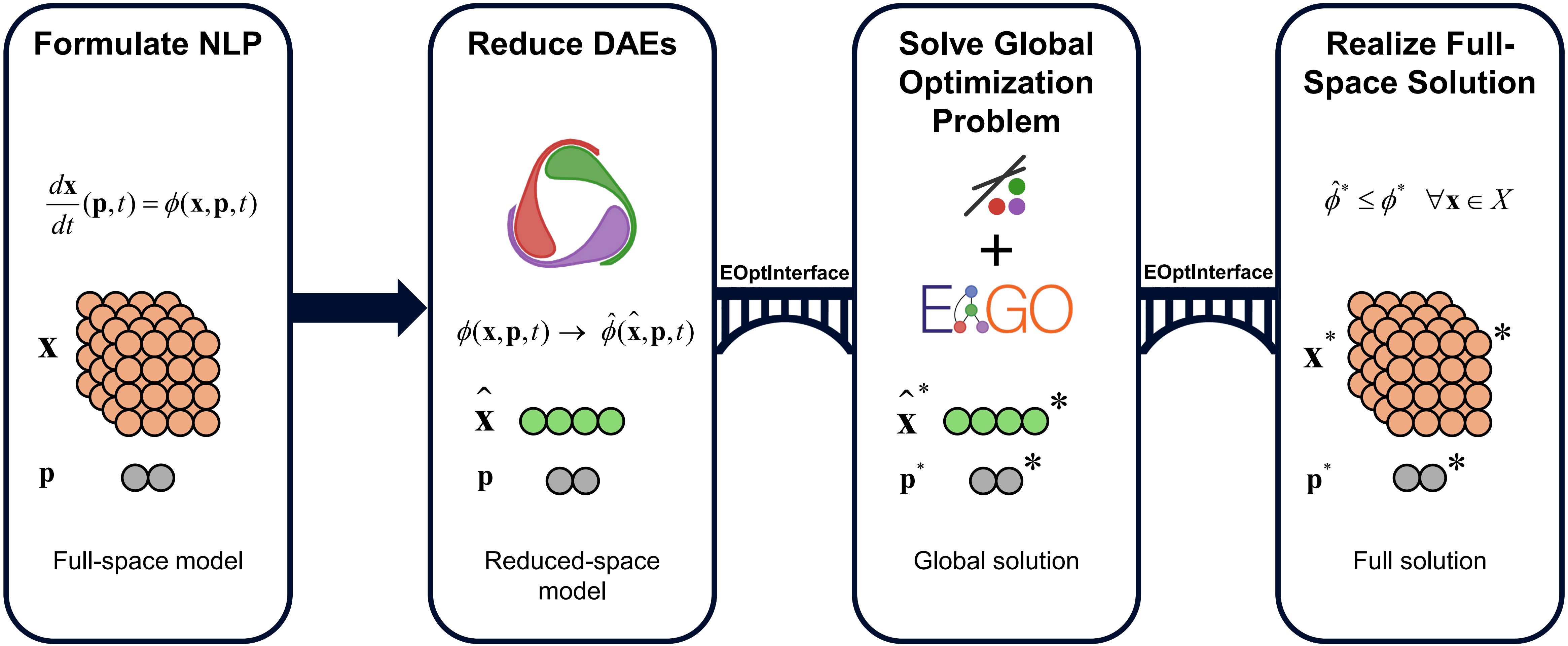
- Minimize total annualized costs
- Design variables:
  - Feed flowrate
  - Reactor volume

$$A \rightarrow B \rightarrow C$$



[13] Kokossis, A. and Floudas, C.A. Synthesis of isothermal reactor-separator-recycle systems. *Chemical Engineering Science*. 46, 1361-1383 (1991).

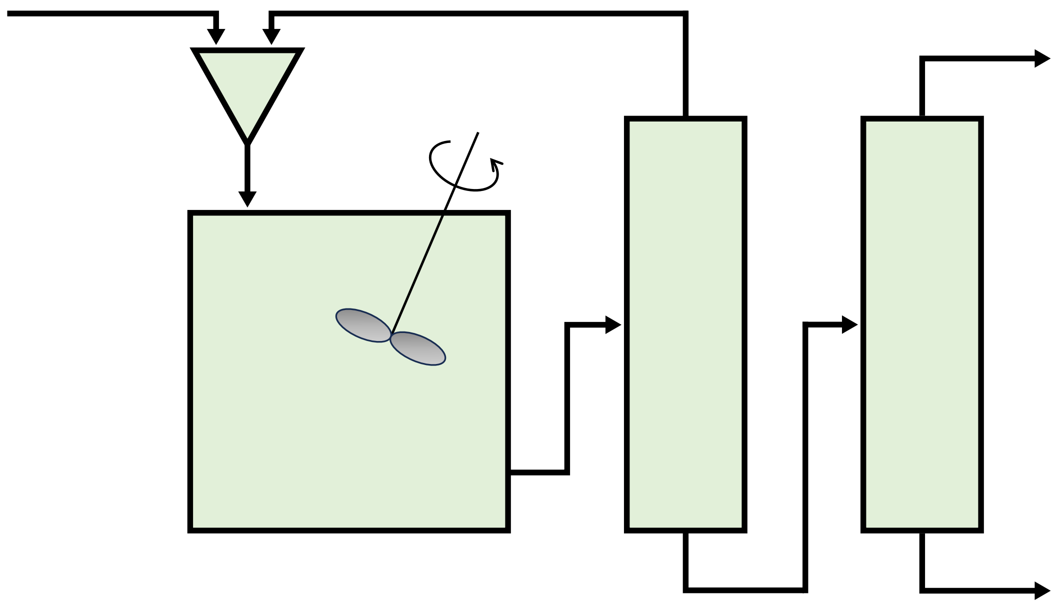
EOI Workflow



Steady-State Process Flow Sheet

- Minimize total annualized costs
- Design variables:
  - Feed flowrate
  - Reactor volume

$$A \rightarrow B \rightarrow C$$



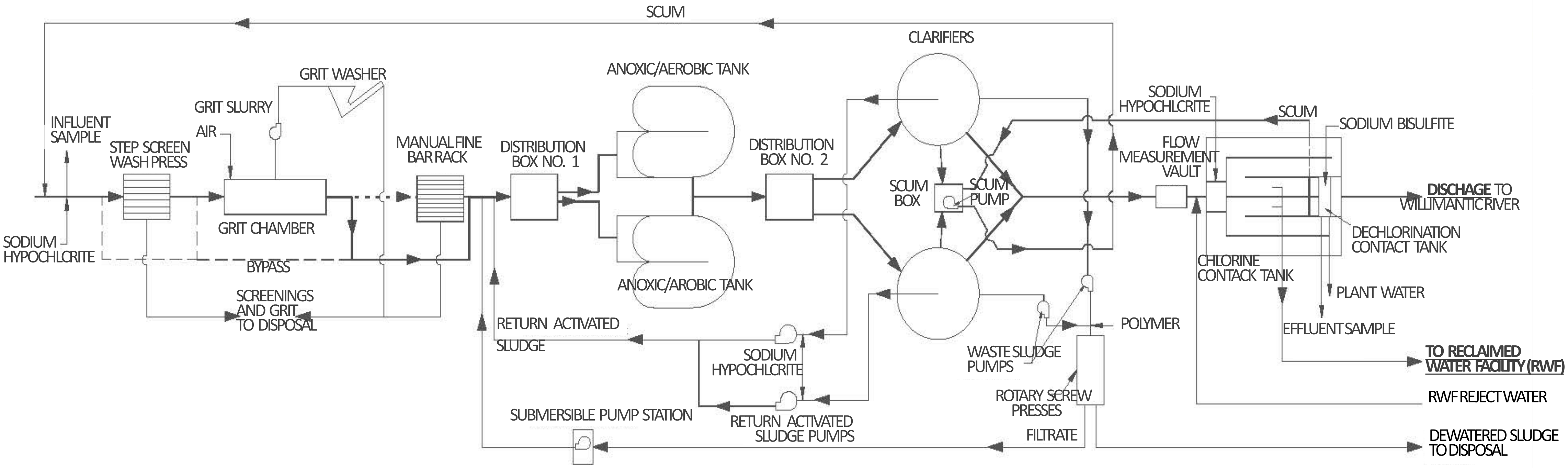
Model	Number of variables	Solve Time (s)
Full-space	50	202.0
Reduced-space	6	3.441

[13] Kokossis, A. and Floudas, C.A. Synthesis of isothermal reactor-separator-recycle systems. *Chemical Engineering Science*. 46, 1361-1383 (1991).

Case Studies

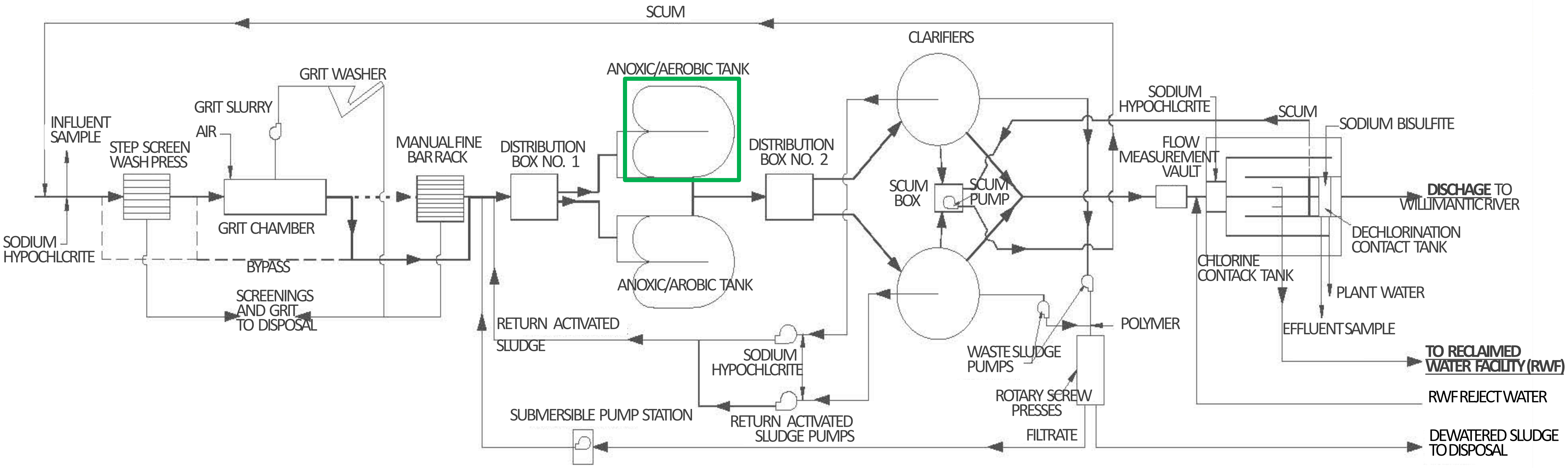
1) Dynamic Kinetic Parameter Estimation
   - Direct transcription of ODEs
2) Steady-State Process Flow Sheet
   - Exploit dimensionality reduction from ModelingToolkit for optimization
3) Nonlinear Model Predictive Control (NMPC)

Nonlinear Model Predictive Control



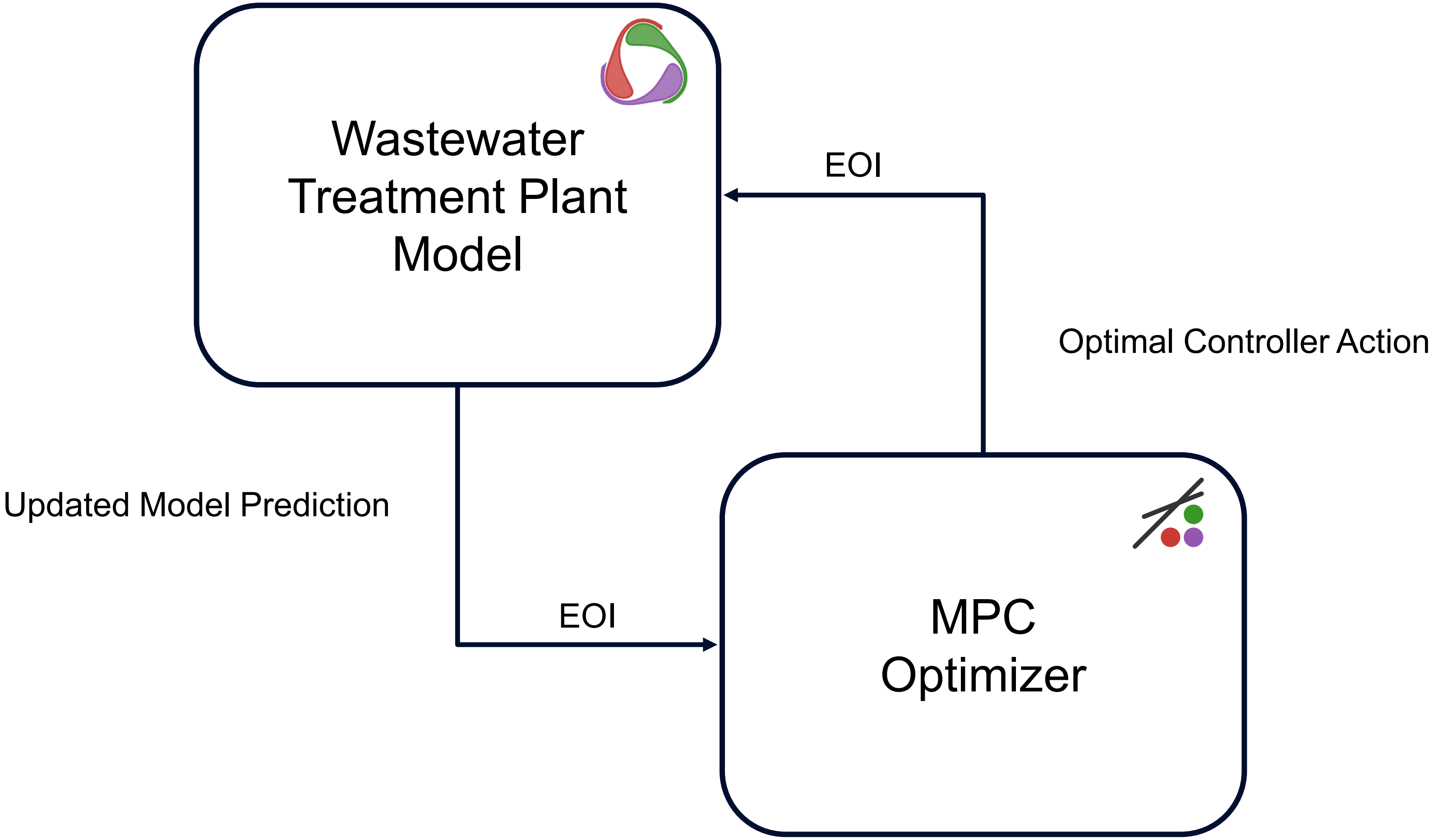
UConn Water Resource Recovery Facility

Nonlinear Model Predictive Control



UConn Water Resource Recovery Facility

Nonlinear Model Predictive Control

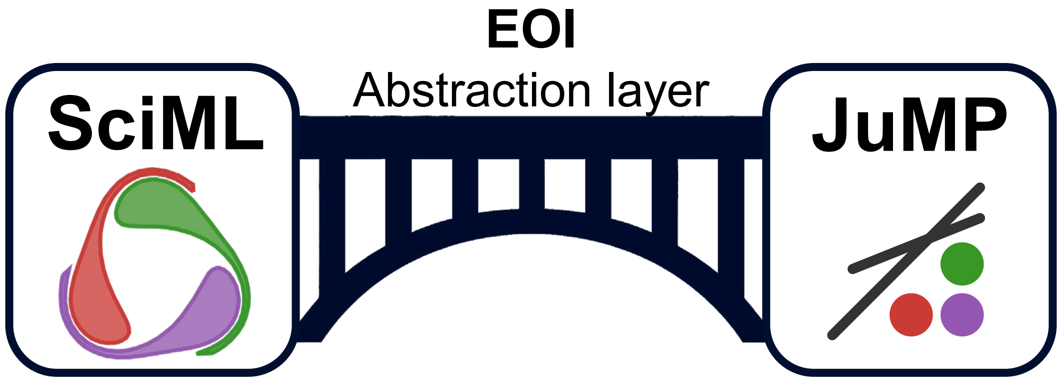


Case Studies

1) Dynamic Kinetic Parameter Estimation
   - Direct transcription of ODEs
2) Steady-State Process Flow Sheet
   - Exploit dimensionality reduction from ModelingToolkit for optimization
3) Nonlinear Model Predictive Control (NMPC)
   - Fully integrated real-time closed-loop NMPC

Conclusion

- **EOptInterface.jl**
  - Seamlessly integrates modeling and simulation with formal optimization
  - Contains multiple discretization methods for ODEs
  - You can use your favorite solver in JuMP



- **Future Work**
  - EAGODynamicOptimizer.jl

Acknowledgements

Members of the Process Systems and Operations Research Laboratory at the University of Connecticut (https://www.psor.uconn.edu)

UCONN
UNIVERSITY OF CONNECTICUT

Process Systems and Operations Research Laboratory

Pratt & Whitney
Institute for Advanced Systems Engineering
UNIVERSITY OF CONNECTICUT

JUMP

Funding:

Department of Energy, Award No.: DE-EE0010991

National Science Foundation, Award No.: 2051084

Pratt & Whitney Endowed Professorship

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or the United States Government.

Questions?

Process Systems and Operations Research Laboratory

The PSOR Laboratory at UConn develops numerical analysis methods and software for process systems engineering applications.

22 followers   University of Connecticut, Storrs, CT, ...   https://psor.uconn.edu   stuber@uconn.edu

https://psor.uconn.edu

https://github.com/PSORLab