UCONN
UNIVERSITY OF CONNECTICUT

Improvements to EAGO.jl: Consolidation and Performance

Dimitri Alston, Ph.D. Student

Robert Gottlieb, Ph.D. Candidate

Matthew Stuber, P&W Associate Professor in Advanced Systems Engineering

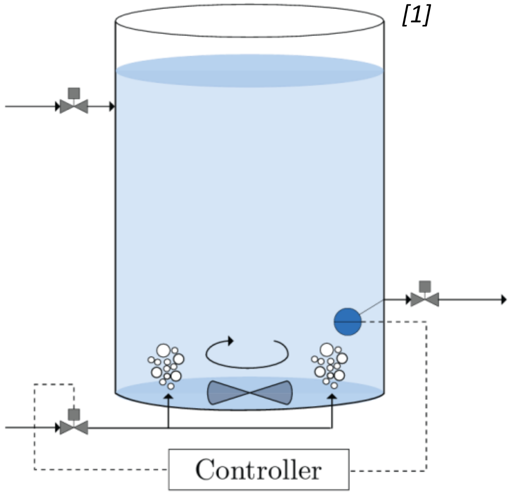
October 28th, 2024

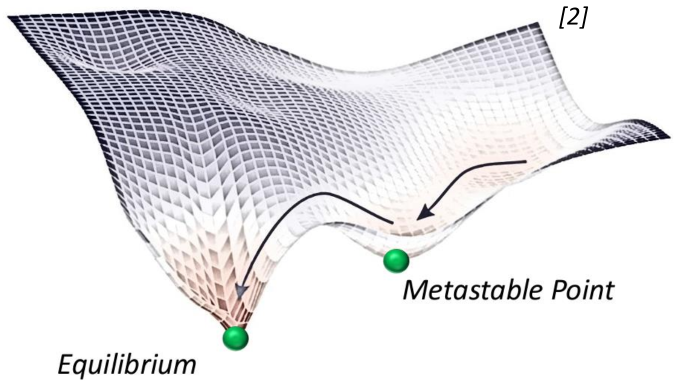
2024 AIChE ANNUAL MEETING

Process Systems and Operations Research Laboratory

Global Optimization

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

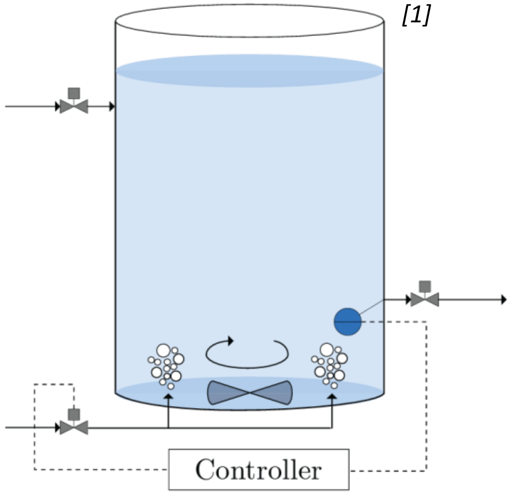


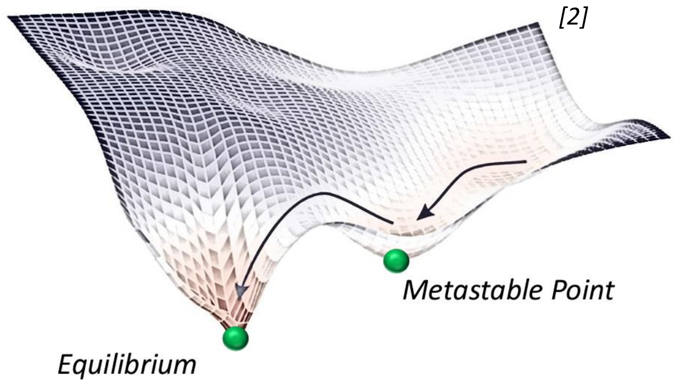


[1] Wang, C., Wilhelm, M.E., and Stuber, M.D. Semi-Infinite Optimization with Hybrid Models. *Industrial & Engineering Chemistry Research.* 61, 5239-5254 (2022).
[2] Grajcarova, L. Simulations of structural phase transitions in crystals using ab initio metadynamics. *INIS-IAEA.* (2013).

Global Optimization

- Complex problems arise in many applications
  - Advanced control systems
  - Thermodynamic stability
  - Kinetic parameter estimation
  - Design under uncertainty
  - Etc.
- Certificate of optimality

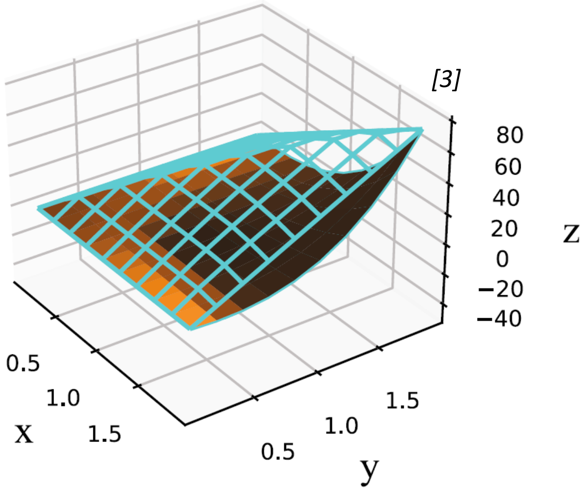




[1] Wang, C., Wilhelm, M.E., and Stuber, M.D. Semi-Infinite Optimization with Hybrid Models. *Industrial & Engineering Chemistry Research.* 61, 5239-5254 (2022).
[2] Grajcarova, L. Simulations of structural phase transitions in crystals using ab initio metadynamics. *INIS-IAEA.* (2013).

Easy Advanced Global Optimization

- Open-source deterministic global solver for nonconvex MINLPs
  - Semi-infinite programs (SIPs)
  - Dynamic optimization
  - User-defined functions
- Applies McCormick-based relaxations for convex lower-bounding problems
- Designed in Julia for performance and extensibility
  - Improved user experience through JuMP.jl



[3] 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).

Parameter Estimation Example

- Oxidation of cyclohexadienyl
- Dynamic optimization problem

$$\min_{\mathbf{p}} \phi(\mathbf{p},t) = \sum_{i=0}^{N} (I_i^{calc} - I_i^{exp})^2 \quad [3]$$

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

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

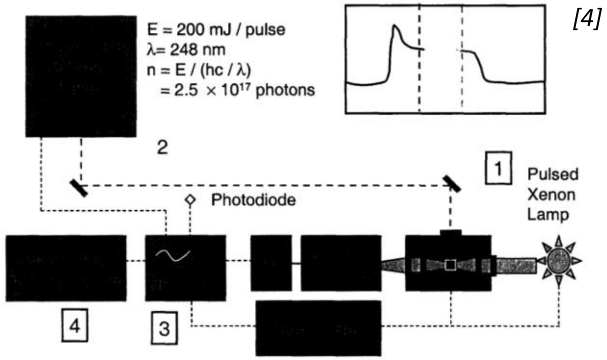
$$\frac{dx_A}{dt} = k_1 x_Z x_Y - c_{O_2}(k_{2f} + k_{3f}) x_A + \frac{k_{2f}}{K_2} x_D + \frac{k_{3f}}{K_3} x_B - k_5 x_A^2$$

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

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

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

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



[4]

[3] 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).
[4] 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).

Parameter Estimation Example

- Oxidation of cyclohexadienyl
- Dynamic optimization problem
  - Simplify using explicit Euler

[3]

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

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

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

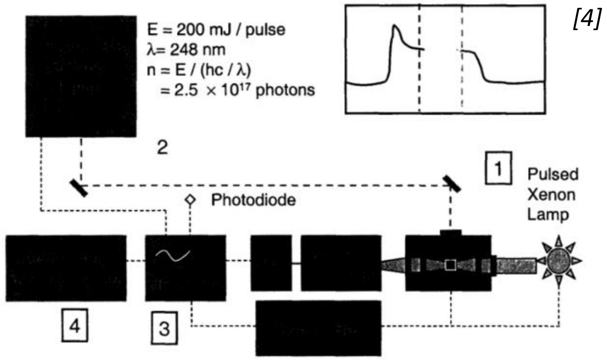
$$x_{A,i+1} = x_{A,i} + \Delta t \left(k_1 x_{Z,i} x_{Y,i} - c_{O_2} (k_{2f} + k_{3f}) x_{A,i} + \frac{k_{2f}}{K_2} x_D^i + \frac{k_{3f}}{K_3} x_{B,i} - k_5 (x_{A,i})^2 \right)$$

$$x_{B,i+1} = x_{B,i} + \Delta t \left(c_{O_2} k_{3f} x_{A,i} - \left(\frac{k_{3f}}{K_3} + k_4 \right) x_{B,i} \right)$$

$$x_{D,i+1} = x_{D,i} + \Delta t \left(c_{O_2} k_{2f} x_{A,i} - \frac{k_{2f}}{K_2} x_{D,i} \right)$$

$$x_{Y,i+1} = x_{Y,i} + \Delta t \left(-k_{1s} x_{Z,i} x_{Y,i} \right)$$

$$x_{Z,i+1} = x_{Z,i} + \Delta t \left(-k_1 x_{Z,i} x_{Y,i} \right)$$



[4]

[3] 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).
[4] 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).

Parameter Estimation Example

```julia
using CSV, DataFrames, EAGO, HiGHS, JuMP

data = CSV.read(joinpath(@__DIR__, "kinetic_intensity_data.csv"), DataFrame)

pL = [10.0, 10.0, 0.001];
pU = [1200.0, 1200.0, 40.0];

intensity(xA, xB, xD) = xA + (2/21)*xB + (2/21)*xD

function explicit_euler_integration(p)⋯
end

function objective(p::Vector{VariableRef})
    x = explicit_euler_integration(p)
    SSE = 0.0
    for i = 1:200
        SSE += (intensity(x[5i-4], x[5i-3], x[5i-2]) - data[!,:intensity][i])^2
    end
    return SSE
end

factory = () -> EAGO.Optimizer(SubSolvers(; r = HiGHS.Optimizer()))
model = Model(factory)
@variable(model, pL[i] <= p[i=1:3] <= pU[i])
@objective(model, Min, objective(p))
JuMP.optimize!(model)
```

$$\min_{\mathbf{p}} \phi(\mathbf{p},t) = \sum_{i=0}^{N} (I_i^{calc} - I_i^{exp})^2$$ [3]

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

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

$$x_{A,i+1} = x_{A,i} + \Delta t \left(k_1 x_{Z,i} x_{Y,i} - c_{O_2} (k_{2f} + k_{3f}) x_{A,i} + \frac{k_{2f}}{K_2} x_D^i + \frac{k_{3f}}{K_3} x_{B,i} - k_5 (x_{A,i})^2 \right)$$

$$x_{B,i+1} = x_{B,i} + \Delta t \left(c_{O_2} k_{3f} x_{A,i} - \left(\frac{k_{3f}}{K_3} + k_4 \right) x_{B,i} \right)$$

$$x_{D,i+1} = x_{D,i} + \Delta t \left(c_{O_2} k_{2f} x_{A,i} - \frac{k_{2f}}{K_2} x_{D,i} \right)$$

$$x_{Y,i+1} = x_{Y,i} + \Delta t \left(-k_{1s} x_{Z,i} x_{Y,i} \right)$$

$$x_{Z,i+1} = x_{Z,i} + \Delta t \left(-k_1 x_{Z,i} x_{Y,i} \right)$$

[3] 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).
[4] 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).

Parameter Estimation Example

```julia
using CSV, DataFrames, EAGO, HiGHS, JuMP

data = CSV.read(joinpath(@__DIR__, "kinetic_intensity_data.csv"), DataFrame)

pL = [10.0, 10.0, 0.001];
pU = [1200.0, 1200.0, 40.0];

intensity(xA, xB, xD) = xA + (2/21)*xB + (2/21)*xD

function explicit_euler_integration(p)...
end

function objective(p::Vector{VariableRef})
    x = explicit_euler_integration(p)
    SSE = 0.0
    for i = 1:200
        SSE += (intensity(x[5i-4], x[5i-3], x[5i-2]) - data[!,:intensity][i])^2
    end
    return SSE
end

factory = () -> EAGO.Optimizer(SubSolvers(; r = HiGHS.Optimizer()))
model = Model(factory)
@variable(model, pL[i] <= p[i=1:3] <= pU[i])
@objective(model, Min, objective(p))
JuMP.optimize!(model)
```

$$\min_{\mathbf{p}} \phi(\mathbf{p},t) = \sum_{i=0}^{N} (I_i^{calc} - I_i^{exp})^2 \quad [3]$$

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

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

$$x_{A,i+1} = x_{A,i} + \Delta t \left(k_1 x_{Z,i} x_{Y,i} - c_{O_2} (k_{2f} + k_{3f}) x_{A,i} + \frac{k_{2f}}{K_2} x_D^i + \frac{k_{3f}}{K_3} x_{B,i} - k_5 (x_{A,i})^2 \right)$$

$$x_{B,i+1} = x_{B,i} + \Delta t \left(c_{O_2} k_{3f} x_{A,i} - \left(\frac{k_{3f}}{K_3} + k_4 \right) x_{B,i} \right)$$

$$x_{D,i+1} = x_{D,i} + \Delta t \left(c_{O_2} k_{2f} x_{A,i} - \frac{k_{2f}}{K_2} x_{D,i} \right)$$

$$x_{Y,i+1} = x_{Y,i} + \Delta t \left(-k_{1s} x_{Z,i} x_{Y,i} \right)$$

$$x_{Z,i+1} = x_{Z,i} + \Delta t \left(-k_1 x_{Z,i} x_{Y,i} \right)$$

[3] 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).
[4] 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).

Parameter Estimation Example

```julia
using CSV, DataFrames, EAGO, HiGHS, JuMP

data = CSV.read(joinpath(@__DIR__, "kinetic_intensity_data.csv"), DataFrame)

pL = [10.0, 10.0, 0.001];
pU = [1200.0, 1200.0, 40.0];

intensity(xA, xB, xD) = xA + (2/21)*xB + (2/21)*xD

function explicit_euler_integration(p) ···
end

function objective(p::Vector{VariableRef})
    x = explicit_euler_integration(p)
    SSE = 0.0
    for i = 1:200
        SSE += (intensity(x[5i-4], x[5i-3], x[5i-2]) - data[!,:intensity][i])^2
    end
    return SSE
end

factory = () -> EAGO.Optimizer(SubSolvers(; r = HiGHS.Optimizer()))
model = Model(factory)
@variable(model, pL[i] <= p[i=1:3] <= pU[i])
@objective(model, Min, objective(p))
JuMP.optimize!(model)
```

$$\min_{\mathbf{p}} \phi(\mathbf{p},t) = \sum_{i=0}^{N} (I_i^{calc} - I_i^{exp})^2$$ [3]

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

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

$$x_{A,i+1} = x_{A,i} + \Delta t \left(k_1 x_{Z,i} x_{Y,i} - c_{O_2} (k_{2f} + k_{3f}) x_{A,i} + \frac{k_{2f}}{K_2} x_D^i + \frac{k_{3f}}{K_3} x_{B,i} - k_5 (x_{A,i})^2 \right)$$

$$x_{B,i+1} = x_{B,i} + \Delta t \left(c_{O_2} k_{3f} x_{A,i} - \left(\frac{k_{3f}}{K_3} + k_4 \right) x_{B,i} \right)$$

$$x_{D,i+1} = x_{D,i} + \Delta t \left(c_{O_2} k_{2f} x_{A,i} - \frac{k_{2f}}{K_2} x_{D,i} \right)$$

$$x_{Y,i+1} = x_{Y,i} + \Delta t \left(-k_{1s} x_{Z,i} x_{Y,i} \right)$$

$$x_{Z,i+1} = x_{Z,i} + \Delta t \left(-k_1 x_{Z,i} x_{Y,i} \right)$$

[3] 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).

[4] 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).

Parameter Estimation Example

```
using CSV, DataFrames, EAGO, HiGHS, JuMP

data = CSV.read(joinpath(@__DIR__, "kinetic_intensity_data.csv"), DataFrame)

pL = [10.0, 10.0, 0.001];
pU = [1200.0, 1200.0, 40.0];

intensity(xA, xB, xD) = xA + (2/21)*xB + (2/21)*xD

function explicit_euler_integration(p)···
end

function objective(p::Vector{VariableRef})
    x = explicit_euler_integration(p)
    SSE = 0.0
    for i = 1:200
        SSE += (intensity(x[5i-4], x[5i-3], x[5i-2]) - data[!,:intensity][i])^2
    end
    return SSE
end

factory = () -> EAGO.Optimizer(SubSolvers(; r = HiGHS.Optimizer()))
model = Model(factory)
@variable(model, pL[i] <= p[i=1:3] <= pU[i])
@objective(model, Min, objective(p))
JuMP.optimize!(model)
```

$$\min_{\mathbf{p}} \phi(\mathbf{p},t) = \sum_{i=0}^{N} (I_i^{calc} - I_i^{exp})^2$$ [3]

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

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

$$x_{A,i+1} = x_{A,i} + \Delta t \left(k_1 x_{Z,i} x_{Y,i} - c_{O_2} (k_{2f} + k_{3f}) x_{A,i} + \frac{k_{2f}}{K_2} x_D^i + \frac{k_{3f}}{K_3} x_{B,i} - k_5 (x_{A,i})^2 \right)$$

$$x_{B,i+1} = x_{B,i} + \Delta t \left(c_{O_2} k_{3f} x_{A,i} - \left(\frac{k_{3f}}{K_3} + k_4 \right) x_{B,i} \right)$$

$$x_{D,i+1} = x_{D,i} + \Delta t \left(c_{O_2} k_{2f} x_{A,i} - \frac{k_{2f}}{K_2} x_{D,i} \right)$$

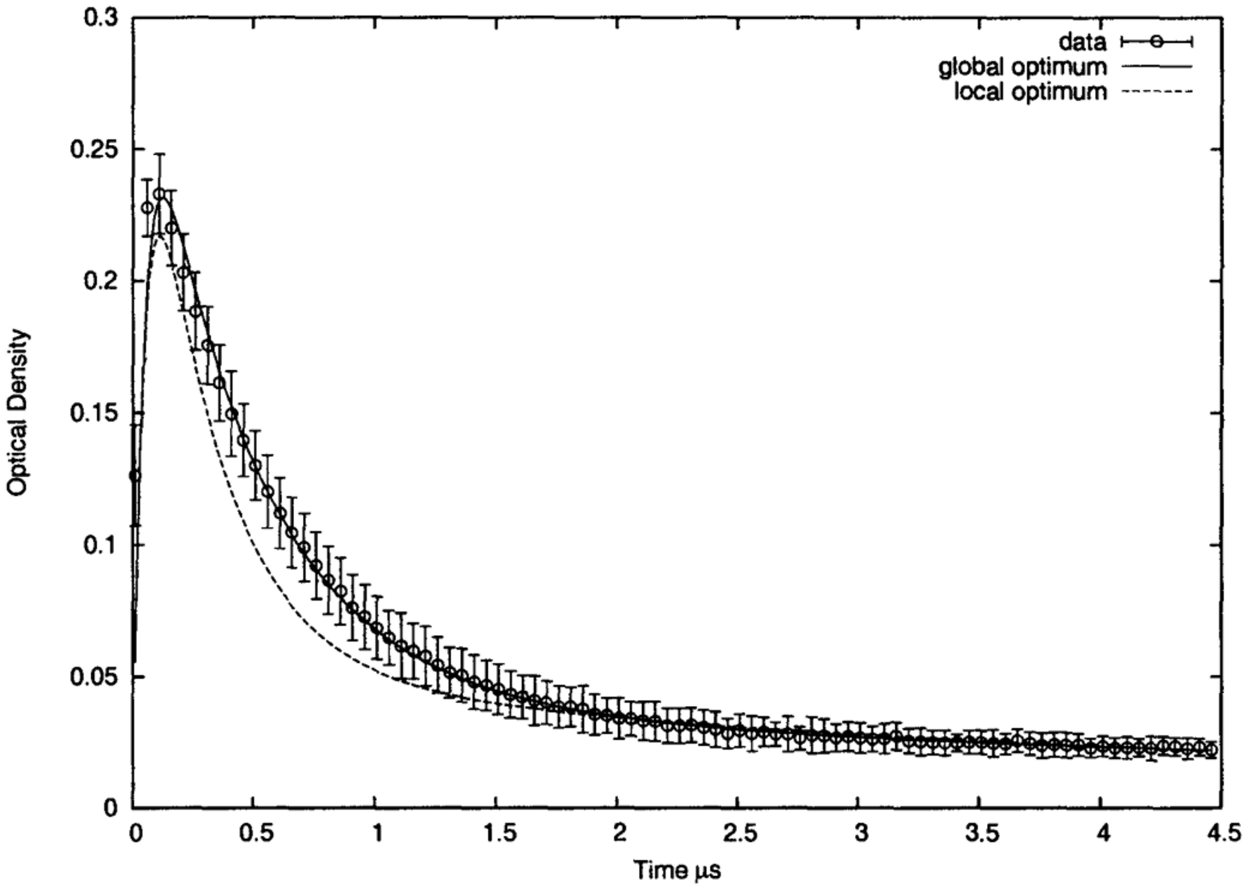
$$x_{Y,i+1} = x_{Y,i} + \Delta t \left(-k_{1s} x_{Z,i} x_{Y,i} \right)$$

$$x_{Z,i+1} = x_{Z,i} + \Delta t \left(-k_1 x_{Z,i} x_{Y,i} \right)$$

[3] 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).
[4] 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).

Parameter Estimation Example

```
using CSV, DataFrames, EAGO, HiGHS,

data = CSV.read(joinpath(@__DIR__, "

pL = [10.0, 10.0, 0.001];
pU = [1200.0, 1200.0, 40.0];

intensity(xA, xB, xD) = xA + (2/21)*

function explicit_euler_integration(
end

function objective(p::Vector{Variabl
    x = explicit_euler_integration(p
    SSE = 0.0
    for i = 1:200
        SSE += (intensity(x[5i-4], x
    end
    return SSE
end

factory = () -> EAGO.Optimizer(SubSo
model = Model(factory)
@variable(model, pL[i] <= p[i=1:3] <
@objective(model, Min, objective(p))
JuMP.optimize!(model)
```



$$k_{3f})x_{A,i} + \frac{k_{2f}}{K_2}x_D^i + \frac{k_{3f}}{K_3}x_{B,i} - k_5(x_{A,i})^2$$

$$4)x_{B,i}$$

[3] 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).
[4] 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).

Notable Updates

EAGO v0.8.x

- Updated nonlinear code to account for JuMP's major refactor

Notable Updates

EAGO v0.8.x

- Updated nonlinear code to account for JuMP's major refactor

EAGO v0.8.2

- Improved user experience in setting up optimization models

$$@\texttt{NLconstraint} \rightarrow @\texttt{constraint}$$
$$@\texttt{NLobjective} \rightarrow @\texttt{objective}$$

Notable Updates

EAGO v0.8.x

- Updated nonlinear code to account for JuMP’s major refactor

EAGO v0.8.2

- Improved user experience in setting up optimization models

@NLconstraint → @constraint

@NLobjective → @objective

Legacy

```
@NLconstraint(model, e1, -x[1]*(1.12 + 0.13167*x[8] - 0.00667*(x[8])^2) + x[4] == 0.0)
@constraint(model, e2, -x[1] + 1.22*x[4] - x[5] == 0.0)
@NLconstraint(model, e3, -0.001*x[4]*x[9]*x[6]/(98.0 - x[6]) + x[3] == 0.0)
@NLconstraint(model, e4, -(1.098*x[8] - 0.038*(x[8])^2) - 0.325*x[6] + x[7] == 57.425)
@NLconstraint(model, e5, -(x[2] + x[5])/x[1] + x[8] == 0.0)
@constraint(model, e6, x[9] + 0.222*x[10] == 35.82)
@constraint(model, e7, -3.0*x[7] + x[10] == -133.0)
```

→

Modern

```
@constraint(model, e1, -x[1]*(1.12 + 0.13167*x[8] - 0.00667*(x[8])^2) + x[4] == 0.0)
@constraint(model, e2, -x[1] + 1.22*x[4] - x[5] == 0.0)
@constraint(model, e3, -0.001*x[4]*x[9]*x[6]/(98.0 - x[6]) + x[3] == 0.0)
@constraint(model, e4, -(1.098*x[8] - 0.038*(x[8])^2) - 0.325*x[6] + x[7] == 57.425)
@constraint(model, e5, -(x[2] + x[5])/x[1] + x[8] == 0.0)
@constraint(model, e6, x[9] + 0.222*x[10] == 35.82)
@constraint(model, e7, -3.0*x[7] + x[10] == -133.0)
```

Notable Updates

EAGO v0.8.x

- Updated nonlinear code to account for JuMP's major refactor

EAGO v0.8.2

- Improved user experience in setting up optimization models

$$\texttt{@NLconstraint} \rightarrow \texttt{@constraint}$$
$$\texttt{@NLobjective} \rightarrow \texttt{@objective}$$

Legacy

```
@NLconstraint(model, e1, -x[1]*(1.12 + 0.13167*x[8] - 0.00667*(x[8])^2) + x[4] == 0.0)
@constraint(model, e2, -x[1] + 1.22*x[4] - x[5] == 0.0)
@NLconstraint(model, e3, -0.001*x[4]*x[9]*x[6]/(98.0 - x[6]) + x[3] == 0.0)
@NLconstraint(model, e4, -(1.098*x[8] - 0.038*(x[8])^2) - 0.325*x[6] + x[7] == 57.425)
@NLconstraint(model, e5, -(x[2] + x[5])/x[1] + x[8] == 0.0)
@constraint(model, e6, x[9] + 0.222*x[10] == 35.82)
@constraint(model, e7, -3.0*x[7] + x[10] == -133.0)
```

→

Modern

```
@constraint(model, e1, -x[1]*(1.12 + 0.13167*x[8] - 0.00667*(x[8])^2) + x[4] == 0.0)
@constraint(model, e2, -x[1] + 1.22*x[4] - x[5] == 0.0)
@constraint(model, e3, -0.001*x[4]*x[9]*x[6]/(98.0 - x[6]) + x[3] == 0.0)
@constraint(model, e4, -(1.098*x[8] - 0.038*(x[8])^2) - 0.325*x[6] + x[7] == 57.425)
@constraint(model, e5, -(x[2] + x[5])/x[1] + x[8] == 0.0)
@constraint(model, e6, x[9] + 0.222*x[10] == 35.82)
@constraint(model, e7, -3.0*x[7] + x[10] == -133.0)
```

Notable Updates

EAGO v0.8.2

- Updated documentation and examples

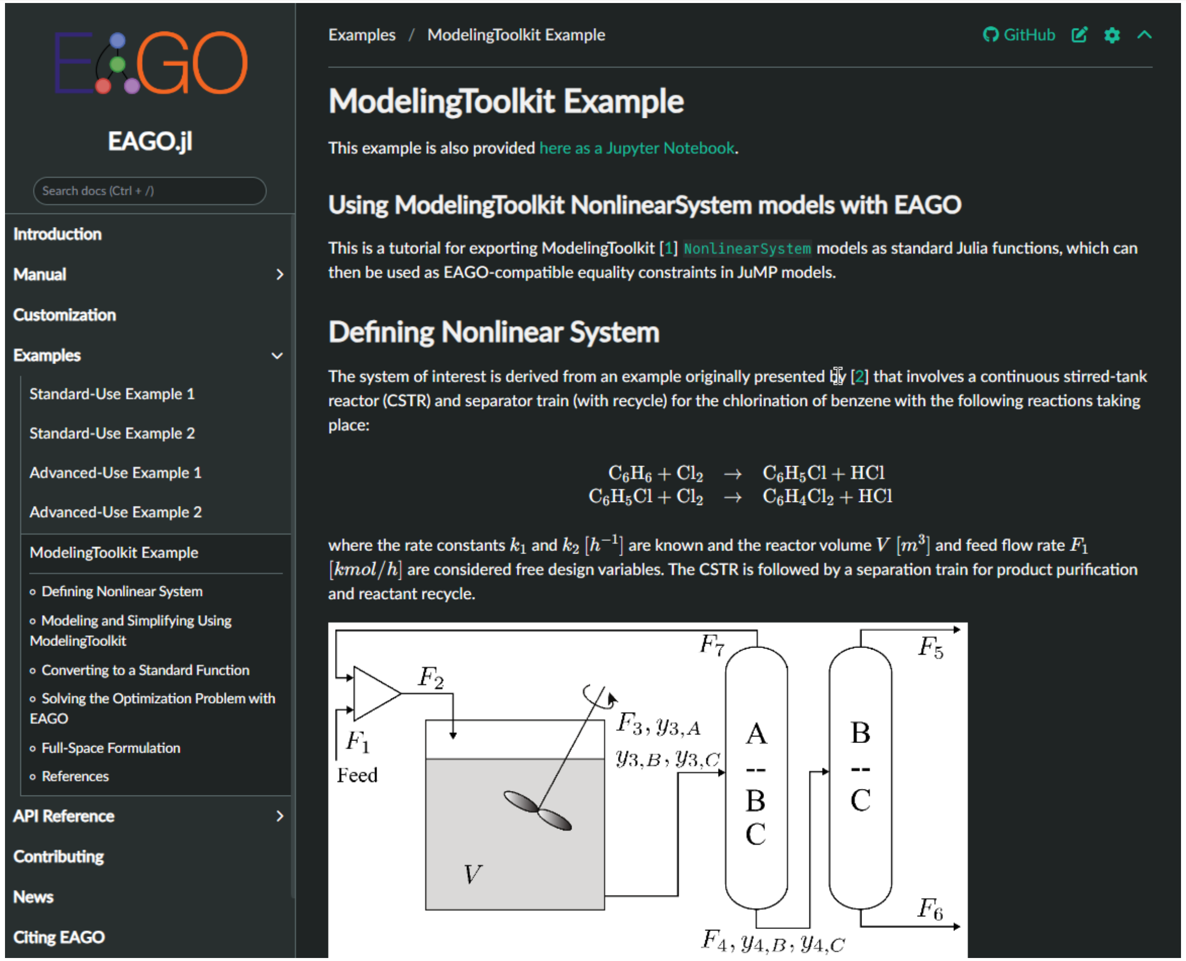
Notable Updates

EAGO v0.8.2

- Updated documentation and examples

Automatic Generation of Reduced-Space Optimization Formulations of Process Systems for Faster Deterministic Global Optimization in Julia

Joseph Choi



Active Projects

- Integrate GPU-based methods

[5]

OPTIMIZATION METHODS & SOFTWARE
https://doi.org/10.1080/10556788.2024.2396297

Taylor & Francis
Taylor & Francis Group

Check for updates

Automatic source code generation for deterministic global optimization with parallel architectures

Robert X. Gottlieb, Pengfei Xu and Matthew D. Stuber

Process Systems and Operations Research Laboratory, Department of Chemical and Biomolecular Engineering, University of Connecticut, Storrs, CT, USA

ABSTRACT
Trends over the past two decades indicate that much of the performance gains of commercial optimization solvers is due to improvements in x86 hardware. To continue making progress, it is critical to consider alternative/specialized massively parallel computing architectures. In this work, we detail the development of an open-source source code transformation approach built using `Symbolics.jl` to construct McCormick-based relaxations of functions that enables their effective parallelized evaluation. We then apply this approach in a novel parallelized branch-and-bound routine that offloads lower- and upper-bounding problems to a GPU. The effectiveness of this new approach is demonstrated on three nonconvex problems of interest, where it yields convergence time improvements of 22–118x compared to an equivalent serial CPU implementation and in two cases outperforms vanilla branch-and-bound versions of existing state-of-the-art solvers that use tighter bounding techniques. This work exemplifies how deterministic global optimizers using alternative hardware architectures can compete with—or eventually outclass—even the most powerful serial CPU implementations, and to the best of the authors' knowledge, represents the first successful demonstration of deterministic global optimization using a GPU.

ARTICLE HISTORY
Received 31 March 2023
Accepted 13 August 2024

KEYWORDS
Dynamical systems; parameter estimation; factorable programing; open-source software; McCormick relaxations

2020 MATHEMATICS SUBJECT CLASSIFICATIONS
90C26; 90-04; 65G30; 26B25; 65Y05

[5] Gottlieb, R.X., Xu, P., and Stuber, M.D. Automatic Source Code Generation for Deterministic Global Optimization With Parallel Architectures. *Optimization Methods & Software*. (2024).

Active Projects

- Integrate GPU-based methods
- Update advanced functionality
  - SIP algorithms
  - Dynamic optimizer
  - Implicit routines

[5]

OPTIMIZATION METHODS & SOFTWARE
https://doi.org/10.1080/10556788.2024.2396297

Taylor & Francis
Taylor & Francis Group

Check for updates

Automatic source code generation for deterministic global optimization with parallel architectures

Robert X. Gottlieb, Pengfei Xu and Matthew D. Stuber

Process Systems and Operations Research Laboratory, Department of Chemical and Biomolecular Engineering, University of Connecticut, Storrs, CT, USA

ABSTRACT
Trends over the past two decades indicate that much of the performance gains of commercial optimization solvers is due to improvements in x86 hardware. To continue making progress, it is critical to consider alternative/specialized massively parallel computing architectures. In this work, we detail the development of an open-source source code transformation approach built using `Symbolics.jl` to construct McCormick-based relaxations of functions that enables their effective parallelized evaluation. We then apply this approach in a novel parallelized branch-and-bound routine that offloads lower- and upper-bounding problems to a GPU. The effectiveness of this new approach is demonstrated on three nonconvex problems of interest, where it yields convergence time improvements of 22–118x compared to an equivalent serial CPU implementation and in two cases outperforms vanilla branch-and-bound versions of existing state-of-the-art solvers that use tighter bounding techniques. This work exemplifies how deterministic global optimizers using alternative hardware architectures can compete with—or eventually outclass—even the most powerful serial CPU implementations, and to the best of the authors' knowledge, represents the first successful demonstration of deterministic global optimization using a GPU.

ARTICLE HISTORY
Received 31 March 2023
Accepted 13 August 2024

KEYWORDS
Dynamical systems; parameter estimation; factorable programing; open-source software; McCormick relaxations

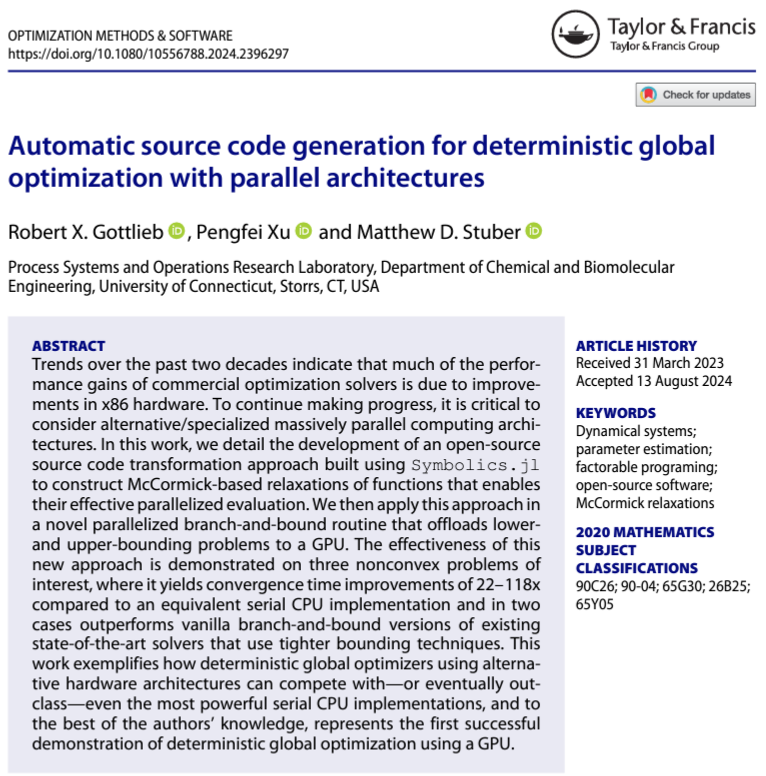
2020 MATHEMATICS SUBJECT CLASSIFICATIONS
90C26; 90-04; 65G30; 26B25; 65Y05

[5] Gottlieb, R.X., Xu, P., and Stuber, M.D. Automatic Source Code Generation for Deterministic Global Optimization With Parallel Architectures. *Optimization Methods & Software*. (2024).

P S O R

Active Projects

- Integrate GPU-based methods
- Update advanced functionality
  - SIP algorithms
  - Dynamic optimizer
  - Implicit routines
- Continue to update documentation and examples



OPTIMIZATION METHODS & SOFTWARE
https://doi.org/10.1080/10556788.2024.2396297

Taylor & Francis
Taylor & Francis Group

Check for updates

Automatic source code generation for deterministic global optimization with parallel architectures

Robert X. Gottlieb, Pengfei Xu and Matthew D. Stuber

Process Systems and Operations Research Laboratory, Department of Chemical and Biomolecular Engineering, University of Connecticut, Storrs, CT, USA

ABSTRACT

Trends over the past two decades indicate that much of the performance gains of commercial optimization solvers is due to improvements in x86 hardware. To continue making progress, it is critical to consider alternative/specialized massively parallel computing architectures. In this work, we detail the development of an open-source source code transformation approach built using `Symbolics.jl` to construct McCormick-based relaxations of functions that enables their effective parallelized evaluation. We then apply this approach in a novel parallelized branch-and-bound routine that offloads lower- and upper-bounding problems to a GPU. The effectiveness of this new approach is demonstrated on three nonconvex problems of interest, where it yields convergence time improvements of 22–118x compared to an equivalent serial CPU implementation and in two cases outperforms vanilla branch-and-bound versions of existing state-of-the-art solvers that use tighter bounding techniques. This work exemplifies how deterministic global optimizers using alternative hardware architectures can compete with—or eventually outclass—even the most powerful serial CPU implementations, and to the best of the authors' knowledge, represents the first successful demonstration of deterministic global optimization using a GPU.

ARTICLE HISTORY
Received 31 March 2023
Accepted 13 August 2024

KEYWORDS
Dynamical systems; parameter estimation; factorable programing; open-source software; McCormick relaxations

2020 MATHEMATICS SUBJECT CLASSIFICATIONS
90C26; 90-04; 65G30; 26B25; 65Y05

[5]

[5] Gottlieb, R.X., Xu, P., and Stuber, M.D. Automatic Source Code Generation for Deterministic Global Optimization With Parallel Architectures. *Optimization Methods & Software*. (2024).

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

NSF

2024 AIChE ANNUAL MEETING

Funding:

National Science Foundation, Award No.: 2330054

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?

EAGO

https://psorlab.github.io/EAGO.jl/dev/

EAGO-notebooks

https://github.com/PSORLab/EAGO-notebooks