Easy Advanced Global Optimization (EAGO)

An Open-Source Platform for Robust and Global Optimization in Julia

Matthew Wilhelm, PhD Student
Matthew Stuber, Assistant Professor (stuber@uconn.edu)

Process Systems and Operations Research Laboratory
Department of Chemical and Biomolecular Engineering and
UTC Institute for Advanced Systems Engineering
University of Connecticut

AIChE Annual Meeting 2017, Minneapolis, MN, October 31st, 2017

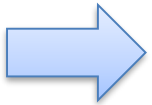
UCONN

Motivation

- Advanced optimization formulations emerge naturally from common system design problems

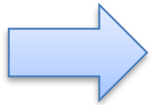
Design Specification

Worst-case constraint satisfaction[1]



Logical Statement[4]

$$\forall \mathbf{p} \in P, \quad \exists \mathbf{u} \in U:$$
$$g(\mathbf{u}, \mathbf{x}, \mathbf{p}) \leq 0$$
$$\mathbf{h}(\mathbf{u}, \mathbf{x}, \mathbf{p}) = \mathbf{0}$$



Semi-infinite Program[4]

$$f^* = \min_{\mathbf{x} \in X} f(\mathbf{x})$$
$$\text{s.t.} \quad g(\mathbf{u}, \mathbf{x}, \mathbf{p}) \leq 0, \forall \mathbf{p} \in P$$
$$\mathbf{h}(\mathbf{u}, \mathbf{x}, \mathbf{p}) = \mathbf{0}, \quad \forall \mathbf{p} \in P$$

- Implementations for recent theoretical advances aren't readily accessible.
  - No publicly available library for differentiable McCormick relaxations[2]
  - No publicly available library for implicit McCormick relaxations[3,4]
  - Prior implementations have an high barrier to usage.

[1] **Worst-case design of subsea production facilities using semi-infinite programming.** Stuber, M.D. et al. (2014) *AIChE Journal*, 60, 2513-2524
[2] **Differentiable McCormick relaxations.** Khan, K. et al. (2017) *Journal Global Optimization*, 67(4), 687-729
[3] **Convex and concave relaxations of implicit functions.** Stuber, M.D. et al. (2015) *Optimization Methods and Software*, 30, 424-460
[4] **Semi-Infinite Optimization with Implicit Functions.** Stuber, M.D., Barton, P.I. (2015) *Industrial & Engineering Chemistry Research*, 54, 307-317

Progression of the Talk

- ☐ Julia Capabilities
- ☐ Developing Robust Optimization for Julia
- ☐ Illustrative Examples
- ☐ Future Directions

Why Julia?

Overview

Ease of Use

- Simplistic Syntax
- Package Management
- Cloud Computing
- Notebook Support

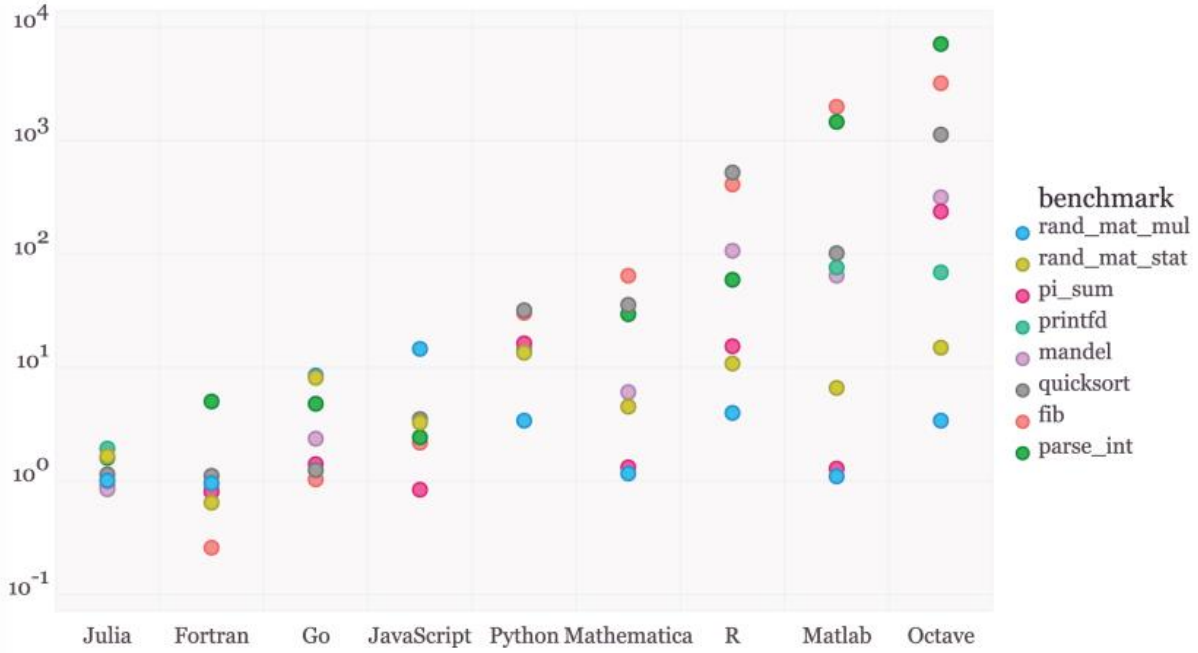
Performance

- Speed of C/Fortran
- Parallelism
- JIT Optimization

Example Julia Code of Solving ODE[4]

```julia
# add packages to library
Pkg.add("DifferentialEquations")
Pkg.add("Plots")
# import packages from library
using DifferentialEquations
using Plots
# setup problem
function lorenz(t,u,du)
 du[1] = 10.0(u[2]-u[1])
 du[2] = u[1]*(28.0-u[3]) - u[2]
 du[3] = u[1]*u[2] - (8/3)*u[3]
end
u0 = [1.0;0.0;0.0]
tspan = (0.0,100.0)
prob = ODEProblem(lorenz,u0,tspan)
# solve problem
sol = solve(prob)
# plot problem
plot(sol,vars=(1,2,3))
```

- Run-time benchmark comparison[5]



Performance comparison of various languages performing simple microbenchmarks. Benchmark execution time relative to C. (Smaller is better; C performance = 1.0.)

[4] DifferentialEquations.jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia. Rackauckas, C. & Nie, Q. (2017). *Journal of Open Research Software,* 5(1), p.15

[5] Julia: A Fresh Approach to Numerical Computing. Jeff Bezanson, Alan Edelman, Stefan Karpinski and Viral B. Shah (2017) *SIAM Review*, 59: 65–98.

Why Julia?

Abstract Syntax Tree

- *Syntactic macros*[5]
  - Work on Abstract Syntax Tree
  - Preserve lexical structure reliably
  - Increased capability relative to text substitution macros in C++
- *Multiple Dispatch Capable*
- *AST representation allows for graph level optimization and modification as alternative to operator overloading*
- *Allows for "overdubbing" instead of overloading to generate context unaware code with context dependent behavior*

Code Generating Smooth McCormick Objects

```
# Generate SMC objects for increasing concave functions
CC_List = [:acosh,:log,:log2,:log10,:sqrt]
for j in CC_List
    eval(quote
          @inline $j(x,xL,xU;relax="cv") = line_seg($j,x,xL,xU)
          @inline $j(x,xL,xU;relax="cc") = $j(x)
          function $j(x::SMC)
            cc = $j(mid(x.cc,x.cv,x.Intv.hi),x.Intv.lo,x.Intv.hi;relax="cc")
            cv = $j(mid(x.cc,x.cv,x.Intv.lo),x.Intv.lo,x.Intv.hi;relax="cv")
            return SMC(cc,cv,$j(x.Intv))
          end
        end)
end
```

Compiler Interaction via Macros

```
x = [1:100]; y = [2:2:200]
# Devectorization of Array Operations
@devec r = exp(x + y) .* sum(x)
# SIMD & Eliminate Bound Checking
@simd for i=1:length(x)
    @inbounds s += x[i]*y[i]+x[i]
end
```

[5] Julia: A Fresh Approach to Numerical Computing. Jeff Bezanson, Alan Edelman, Stefan Karpinski and Viral B. Shah (2017) *SIAM Review*, 59: 65–98.

Why Julia?
Design Objectives

Current Capabilities:

- Native use of LAPACK, SuiteSparse
- JuMP/JuMPeR[6] (Julia AML)
  - Low syntax, User Defined Functions
  - Automatic Derivative Calculations
  - Callback Support
- ValidatedNumerics.jl (Interval Library)
- ForwardDiff.jl[8] (Automatic Differentiation)

Future Capabilities:

- Future Compatibility for Overdubbing Approaches to Source-Code Transformation

Objective: *Open-Source*

Create a toolbox of modules that end-users can modify

Objective: *Intuitive User Interface*

Provide a graphical interface and interface to AMPL

Objective: *Easily Accessible Routines*

Provide pre-built routines for that can be used without advanced subject area knowledge

[6] **JuMP: A Modeling Language for Mathematical Optimization** Iain Dunning and Joey Huchette and Miles Lubin (2017) *SIAM Review*, 59: 295–320.
[7] **ValidatedNumerics.jl** [Computer Software]. Retrieved from https://github.com/JuliaIntervals/ValidatedNumerics.jl.
[8] **Forward-Mode Automatic Differentiation in Julia.** (2016) Revels, J., Lubin, M., Paramarkou, T. arXiv:1607.07892

Progression of the Talk

☑ Julia Capabilities

☐ Developing Robust Optimization for Julia

Build global solver*

- **Nonconvex Semi-Infinite Programming**
- **Branch-and-Bound Library**
- **Smooth McCormick Relaxation Library**
- **Implicit Function Bounding Routine**
- **Constraint Propagation via DAG**
- **Global Solver**

Analytic Tools

- **Directed Graph Toolbox**
- Parametric Interval Methods

User Interface

- Flow-sheeting Interface
- Solution Visualization
- Data Import/Export/Storage
- AML Integration

☐ Illustrative Examples

☐ Future Directions

* Support for validated calculations
* Special routines for imbedded implicit functions

SIP Solvers

- **EAGOSemiInfinite** implements implicit and explicit meta-algorithms

Explicit SIP

$$f^* = \min_{\mathbf{x}\in X} f(\mathbf{x})$$

$$\text{s.t. } g(\hat{\mathbf{y}}, \mathbf{x}, \mathbf{p}) \le 0, \forall (\mathbf{y}, \mathbf{p}) \in Y \times P$$

$$\mathbf{h}(\hat{\mathbf{y}}, \mathbf{x}, \mathbf{p}) = \mathbf{0}, \ \forall (\mathbf{y}, \mathbf{p}) \in Y \times P$$

Implicit SIP

$$f^* = \min_{\mathbf{x}\in X} f(\mathbf{x})$$

$$\text{s.t. } g(\mathbf{y}(\mathbf{x}, \mathbf{p}), \mathbf{x}, \mathbf{p}) \le 0, \quad \forall \mathbf{p} \in P$$

- Solved generally via restriction of right-hand side method[9,4]
  - *Discretization of uncertainty set for upper/lower bounds.*
  - *Finite convergence (if Slater point arbitrarily near minimizer).*

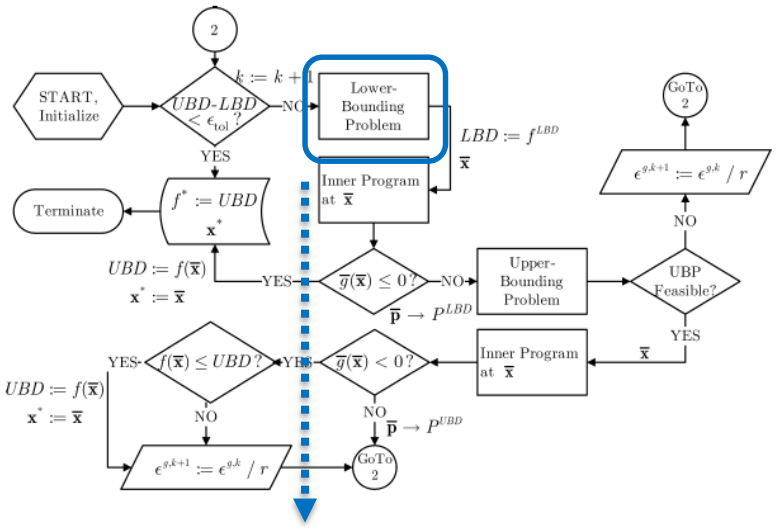
- Problem size and complexity dramatically increased with equality constraints[4,9,10]

[9] **Global optimization of semi-infinite programs via restriction of the right-hand side**. Mitsos, A. (2011). *Optimization* 60:10-11, 1291-1308

[4] **Semi-Infinite Optimization with Implicit Functions.** Stuber, M.D., Barton, P.I. (2015) *Industrial & Engineering Chemistry Research*, 54, 307-317

[10] **Evaluation of process systems operating envelopes.** Stuber, M.D., (2013) Ph.D Thesis.

Flowchart for restriction of RHS method[4]



Lower Bounding Problem (explicit) for restriction of RHS method[4]

$$f^{LBD} = \min_{\mathbf{x}\in X} f(\mathbf{x})$$

$$\text{s.t. } g_{SIP}(\hat{\mathbf{y}}, \mathbf{x}, \mathbf{p}) \le 0 \quad \forall (\mathbf{y}, \mathbf{p}) \in Y^{LBD} \times P^{LBD}$$

$$\mathbf{h}(\hat{\mathbf{y}}, \mathbf{x}, \mathbf{p}) = \mathbf{0} \quad \forall (\mathbf{y}, \mathbf{p}) \in Y^{LBD} \times P^{LBD}$$

Branch & Bound Library

- **Configuration**
  - BNB type (option storage)
  - BNBObject type (node storage)
- **Fully customizable**
- **Natively for common schemes**
  - Best-first, depth-first, breadth-first search
  - Absolute/relative midpoint bisection

Install Packages

```
using EAGOBranchBound
using ValidatedNumerics
```

Initialize B&B Objects

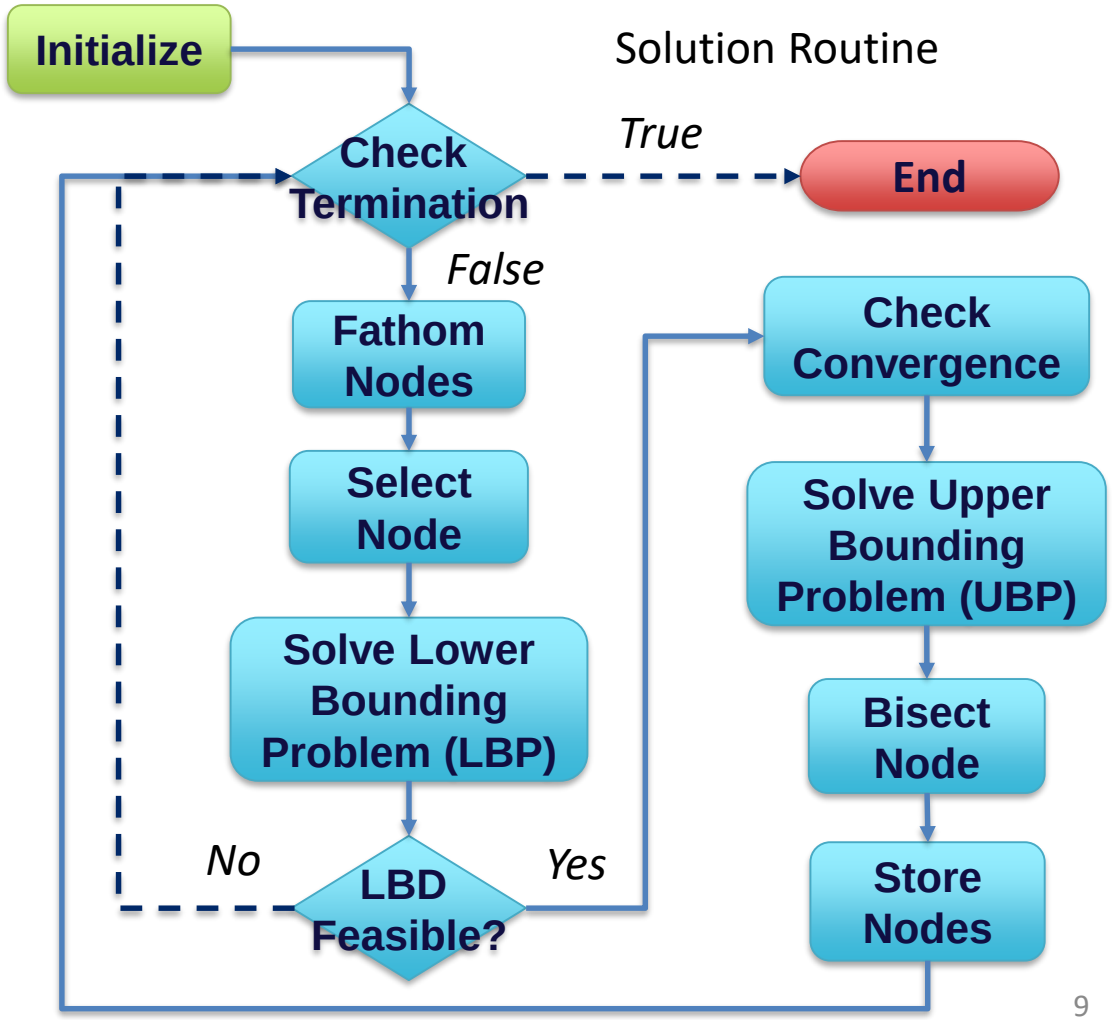
```
b = IntervalBox(-1..1,2..9)
a = EAGOBranchBound.BnB(b)
C = EAGOBranchBound.BnBObject(b)
EAGOBranchBound.set_to_default!(a)
```

Define and Set UBP/LBP

```
function ex_LBP(X::IntervalBox,k,opt)
  ex_LBP_int = @interval X[1]+X[2]^2
  return ex_LBP_int.lo, mid.(X), true
end
function ex_UBP(X::IntervalBox,k,opt)
  ex_UBP_int = @interval X[1]+X[2]^2
  return ex_UBP_int.hi, mid.(X), true
end
a.Lower_Prob = ex_LBP
a.Upper_Prob = ex_UBP
```

Solve Problem

```
outy = EAGOBranchBound.solve(a,C)
```



McCormick Relaxations

Implementation

Smooth McCormick Object[11]

```
struct SMC <: AbstractMC
  cc
  cv
  Intv
  function SMC(a,b,c)
    if ~(typeof(c)<:Interval)
      error("SMC of form SMC($a,$b,$c) not allowed.
             Last argument must be an interval.")
    elseif (a<b)
      error("SMC of form SMC($a,$b,$c) not allowed.
             Must have cv <= cc.")
    end
    new SMC(a,b,c)
end
```

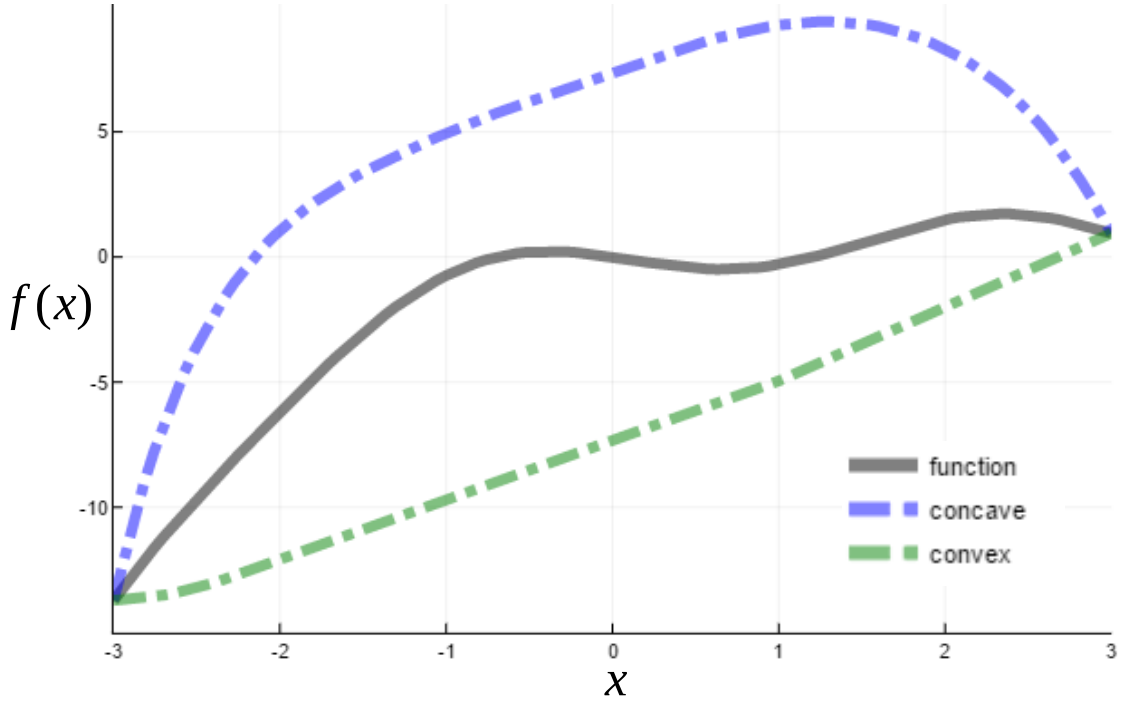
Expressions Supported:

+, -, /, *, ^, exp, exp2, exp10, log, log2, log10, sqrt, pow, sin, cos, tan, asin, acos, atan, sinh, cosh, tanh, asinh, acosh, atanh, min, max, abs, step, sign

Gradients of Relaxations:

- Absent from SMC to allow for flexibility
- Can be readily calculated via automatic differentiation
- Existing packages (JuMP[6], ForwardDiff[8]) outperform naïve implementation

$$f(x) = \exp(-x/2)x - \sin(2x)$$



Performance:

Comparable in speed to MC++ library[11] +/-30% runtime in benchmarks

[2] **Differentiable McCormick relaxations.** Khan, K. et al. (2017) *Journal Global Optimization*, 67(4), 687-729
[6] **JuMP: A Modeling Language for Mathematical Optimization.** Dunning, I. Huchette, J. and Lubin, M. (2017) *SIAM Review*, 59, 295–320.
[8] **Forward-Mode Automatic Differentiation in Julia.** (2016) Revels, J., Lubin, M., Paramarkou, T. arXiv:1607.07892
[11 **MC++: A versatile library for McCormick relaxations and Taylor models**. B. Chachuat. http://www3.imperial.ac.uk/people/b.chachuat/research
[12] **McCormick-based relaxations of algorithms.** Mitsos et al. (2009) *SIAM Journal on Optimization*, SIAM, 2009, 20, 73-601

Abstract Syntax Tree

Directed Graphs & Interval Constraint Propagation

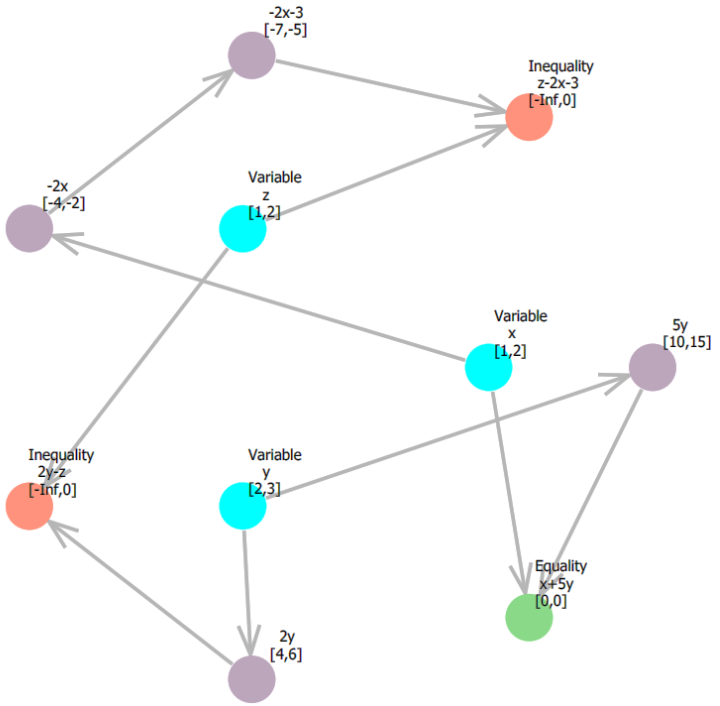
EAGODAGContractor Package

- Methods for interval constraint propagation via natural interval extensions and interval contractors[1]
- Supports visualization of directed acyclic graphs (DAG) via Gladfly integration
- Integrates with common-subexpression reduction routines (@cse) in Julia

Expressions Supported:

+, -, /, *, ^, exp, exp2, exp10, log, log2, log10, sqrt, pow, sin, cos, tan, asin, acos, atan, sinh, cosh, tanh, asinh, acosh, atanh, min, max, abs, step, sign

DAG Visualization



[1] **Worst-case design of subsea production facilities using semi-infinite programming.** Stuber, M.D. et al. (2014) *AIChE Journal*, 60, 2513-2524

[7] **ValidatedNumerics.jl** [Computer Software]. Retrieved from https://github.com/JuliaIntervals/ValidatedNumerics.jl.

Global Solver[2,3]

Solver Options

- **IPOPT is standard solver for LBP/UBP**
  - Open-source
  - Compiled version via Package manager
  - Sparsity Pattern Detected Automatically
  - Forward-Mode AD Calculation of Hessian/Jacobian/Gradients Used
- **Alternative LBD Solver:**
  - SNOPT
  - Proximal-Bundle
- **Alternative UBP Solver:**
  - Any JuMP NLP solver
- **Affine Relaxations Also Available**
  - JuMP LP solvers

Global Solution Scheme

- **Default settings:**
  - Best-first search
  - Relative-width bisection
  - Validated interval calculations
- **Preprocessing Step**
  - Forward-Reverse Interval Constraint Propagation
  - (Optional): Interval Parametric Tests

- **Lower Bounding Problem***

$$f^{LBD} = \min_{\mathbf{p}\in P} f^{cv}(\mathbf{x}(\mathbf{p}),\mathbf{p})$$

$$\text{s.t. } g^{cv}(\mathbf{x}(\mathbf{p}),\mathbf{p}) \le 0$$

**Bounds calculated via implicit-function based fixed point iteration[3] using smooth McCormick relaxations[2].*

- **Upper Bounding Problem**

$$f^{UBD} = \min_{\mathbf{z}\in Z,\mathbf{p}\in P} f(\mathbf{z},\mathbf{p})$$

$$\text{s.t. } g(\mathbf{z},\mathbf{p}) \le 0$$

$$\mathbf{h}(\mathbf{z},\mathbf{p}) = \mathbf{0}$$

[2] **Differentiable McCormick relaxations.** Khan, K. et al. (2017) *Journal Global Optimization*, 67(4), 687-729
[3] **Convex and concave relaxations of implicit functions.** Stuber, M.D. et al. (2015) *Optimization Methods and Software*, 30, 424-460

Progression of the Talk

- ☑ Julia Capabilities
- ☑ Developing Robust Optimization for Julia
- ☐ Illustrative Examples
- ☐ Future Directions

Case Study I

Kinetic Parameter Estimation[3,12,13,14]

Problem Statement

- Determine the forward rate constants (k_{2f}, k_{3f}, k_4) of oxygen addition to cyclohexadienyl radicals from transient flash photolysis data.
- Mass-balance ODEs system discretized via implicit Euler.
- Minimize residual sum-of-squares for intensity.

Equality Constraints

$$\boldsymbol{h}(\boldsymbol{z},\boldsymbol{p}) = \begin{cases} \boldsymbol{c}_{\boldsymbol{A}}^{i-1} - \boldsymbol{c}_{\boldsymbol{A}}^{i} + \Delta t\left(k_1 \boldsymbol{c}_{\boldsymbol{Y}}^{i}\boldsymbol{c}_{\boldsymbol{Z}}^{i} - C_{O_2}(k_{2f}+k_{3f})\boldsymbol{c}_{\boldsymbol{A}}^{i} + \frac{k_{2f}}{K_2}\boldsymbol{c}_{\boldsymbol{D}}^{i} + \frac{k_{3f}}{K_3}\boldsymbol{c}_{\boldsymbol{B}}^{i} - k_5 \boldsymbol{c}_{\boldsymbol{A}}^{i\,2}\right) = 0 \\ \boldsymbol{c}_{\boldsymbol{B}}^{i-1} - \boldsymbol{c}_{\boldsymbol{B}}^{i} + \Delta t\left(k_{3f}\boldsymbol{c}_{\boldsymbol{A}}^{i} C_{O_2} - \left(\frac{k_{3f}}{K_3} + k_4\right)\boldsymbol{c}_{\boldsymbol{B}}^{i}\right) = 0 \\ \boldsymbol{c}_{\boldsymbol{D}}^{i-1} - \boldsymbol{c}_{\boldsymbol{D}}^{i} + \Delta t\left(k_{2f}\boldsymbol{c}_{\boldsymbol{A}}^{i} C_{O_2} - \frac{k_{2f}}{K_2}\boldsymbol{c}_{\boldsymbol{D}}^{i}\right) = 0 \\ \boldsymbol{c}_{\boldsymbol{Y}}^{i-1} - \boldsymbol{c}_{\boldsymbol{Y}}^{i} + \Delta t\left(-k_{1s}\boldsymbol{c}_{\boldsymbol{Y}}^{i}\boldsymbol{c}_{\boldsymbol{Z}}^{i}\right) = 0 \\ \boldsymbol{c}_{\boldsymbol{Z}}^{i-1} - \boldsymbol{c}_{\boldsymbol{Z}}^{i} + \Delta t\left(-k_1 \boldsymbol{c}_{\boldsymbol{Y}}^{i}\boldsymbol{c}_{\boldsymbol{Z}}^{i}\right) = 0 \end{cases}$$

Objective Function

$$\min_{\boldsymbol{p}\in P} \sum_{i=1}^{n} \left(\boldsymbol{I}^{i} - \boldsymbol{I}_{\boldsymbol{data}}^{i}\right)^2$$

$$s.t. \quad \boldsymbol{I}^{i} = \boldsymbol{c}_{A}^{i} + \tfrac{2}{21}\boldsymbol{c}_{B}^{i} + \tfrac{2}{21}\boldsymbol{c}_{D}^{i}$$

Uncertain Variables

$$\boldsymbol{p} = (k_{2f}, k_{3f}, k_4)$$

State Variables

$$\boldsymbol{z} = (\ldots, \boldsymbol{c}_{\boldsymbol{A}}^{i}, \boldsymbol{c}_{\boldsymbol{B}}^{i}, \boldsymbol{c}_{\boldsymbol{D}}^{i}, \boldsymbol{c}_{\boldsymbol{Y}}^{i}, \boldsymbol{c}_{\boldsymbol{Z}}^{i}, \ldots)$$

$$s.t. \quad i = 1 \ldots n$$

Parameters

$$k_1, k_{1s}, k_5, K_2, K_3, c_{O2}, \Delta t, n$$

[3] **Convex and concave relaxations of implicit functions.** Stuber, M.D. et al. (2015) *Optimization Methods and Software*, 30, 424-460
[12] **McCormick-based relaxations of algorithms.** Mitsos et al. (2009) *SIAM Journal on Optimization*, SIAM, 2009, 20, 73-601
[13] **Direct measurement of the fast, reversible addition of oxygen to cyclohexadienyl radicals in nonpolar solvents**, J. W. Taylor, et al. *Phys. Chem. A*, 108 (2004), pp. 7193–7203.
[14] **Global dynamic optimization for parameter estimation in chemical kinetics** A. B. Singer et al., *J. Phys. Chem. A*, 110 (2006), pp. 971–976.

Case Study II

Kinetic Parameter Estimation

Read Data in Julia

```
using ExcelReaders, DataFrames, IntervalArithmetic, EAGOGlobalSolver

xls_data = readxl("C:/Users/matt/Desktop/KineticParameterData.xlsx",
                  "Sheet1!A1:B5")
data = convert(Array{Float64},xls_data[:,2])
```

Define objective

```
eval( quote function f(x,p)
                I_calc = x[1:5:996]+(2/21)*x[2:5:997]+(2/21)*x[3:5:998]
                return sum((I_calc-data).^2) end end)
```

Define equality constraints

```
delt = 0.01; T = 273; k1 = 53; k1s = k1*1E-6; k5 = 1.2E-3
K2 = 46*exp(6500/T-18); K3 = 2*K2; cO2 = 2E-3; n = 200
eval( quote function h(z,p)
  h = zeros(typeof(z),n*5)
  for i=0:($n-1)
    h[5*i+1] = z[5*(i+1)+1] - z[5*i+1] + $delt*($k1*z[5*i+4]*z[5*i+5]
             - (p[1]+p[2])*z[5*i+1]*$cO2  + (p[1]/$K2)*z[5*i+4]
             + (p[2]/$K3)*z[5*i+4] - $k5*z[5*i+1]^2)
    h[5*i+2] = z[5*(i+1)+2] - z[5*i+2] + $delt*(p[2]*z[5*i+1]*$cO2
             - (p[2]/$K3+p[3])*z[5*i+2])
    h[5*i+3] = z[5*(i+1)+3] - z[5*i+3] + $delt*(p[1]*z[5*i+1]*$cO2
             - (p[1]/$K2))
    h[5*i+4] = z[5*(i+1)+4] - z[5*i+4] + $delt*(-$k1s*z[5*i+4]*z[5*i+5])
    h[5*i+5] = z[5*(i+1)+5] - z[5*i+5] + $delt*(-$k1*z[5*i+4]*z[5*i+5])
  end
return h
end end)
```

Setup Box Constraints for Problem

```
# creates intervalboxes
Xt = []
for i=1:1000
  mod(i,4)==0 ? push!(Xt,Interval(0,140)) : push!(Xt,Interval(0,0.4))
end
X = IntervalBox(tuple(Xt)...)
P = IntervalBox(10..1200,10..1200,(0.001)..40)
```

Call EAGO to Solve Problem

```
# creates initial interval box, #4 = [0,140], #1-3,5 = [0,0.4]
solution = EAGO_Global_Implicit(f,[],h,X,P,[],hbox,[])
```

Finishes with certificate of global optimality

- *Optimal point:* (789.02, 423.85,12.969)
- *Optimal value:* 16796.04
- Solution consistent with the literature

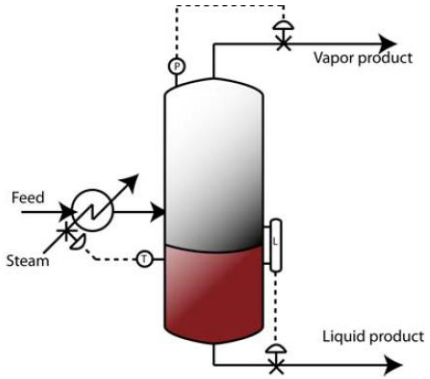
Case Study II

Flash Separation of Benzene and Toluene[14]

Problem Statement

We wish to ensure a cut-fraction, α, of less than or equal to 0.7 in the flash separation of a benzene-toluene mixture subject to uncertain temperature, T, by controlling the pressure, P.

Flash Separator



SIP Formulation

$$\max_{\tau \in T,\, p \in P} \eta$$

$$s.t. \quad \eta \le \alpha(\tau, p) - 0.7, \quad \forall p \in P$$

$$\tau \in T = [80,110]$$

$$p \in P = [90,100]$$

$$\alpha \in A = [0,1]$$

Process Model

$$h(\alpha,\tau,p) = \sum_{i=1}^{2} \frac{z_i(K_i(\tau,p)-1)}{(K_i(\tau,p)-1)\alpha+1}, \quad K_i(\tau,p) = \frac{p_i^{sat}(\tau)}{p}, \quad \log(p_i^{sat}(\tau)) = A_i - \frac{B_i}{C_i+\tau}$$

JuMP/Julia Code for Formulating/Solving Model

```
# specifies model and variables
m = RobustModel()
@uncertain(m, 90.0 <= P <= 100) @variable(m, 80.0 <= T <= 110)
@variable(m, nu) @variable(m, 0<= alpha <= 1)
# sets parameters
@NLparameter(m, z == 0.5)
n = 2 # number of species
data = [6.95087 6.87987; 1342.31 1936.01; 219.187 258.451]
for j in [[A,1],[B,2],[C,3]]
  @NLparameter(m, j[i,1][i=1:n] == data[i,j[i,2]])
end
# defines constraints
Psat = @NLexpression(m,[i=1:n],10^(A[i]-B[i]/(C[i]+tau)))
K = @NLexpression(m,[i=1:n],Psat[i]/P)
summand = @NLexpression(m,[i=1:n],z*(K[i]-1)/((K[i]*alpha-1)))
@NLconstraint(m, sum(summand[i] for i=1:n) == 0.0)
@NLconstraint(m, nu <= alpha-0.7)
@NLobjective(m, Max, nu)
value,point,info = SolveSIP(m)
```

Solving the SIP, we find that no pressure setting exists which is robust to the full range of temperature variation.

[14] **Robust simulation and design using semi-infinite programs with implicit functions.** Stuber, M. D. & Barton, P. I. (2011) *International Journal of Reliability and Safety*, 5, 378-397

Recap

- ☑ Julia Capabilities
- ☑ Developing Robust Optimization for Julia
- ☑ Illustrative Examples
- ☐ Future Directions

Case Study III

Worst-Case Subsea Separator Design[1]

Problem Statement (Original)

For any inlet gas fraction (*stream 1*) within typical bounds, is there a control setting that will prevent the effluent gas fraction of the liquid-liquid separator (*stream 7*) from exceeding the specification and damaging the pump.

Problem Statement (Semi-infinite Program)

$$\mathbf{h}(\mathbf{z},\mathbf{u},\mathbf{p})=\mathbf{0}\longrightarrow\mathbf{z}=\mathbf{x}(\mathbf{u},\mathbf{p})$$

$$\eta^{*}=\max_{\mathbf{p}\in P,\mathbf{u}\in U}\eta$$

$$\eta\le g(\mathbf{x}(\mathbf{u},\mathbf{p}),\mathbf{u},\mathbf{p}),\forall\mathbf{u}\in U$$

Equality constraints specified by mass balances, energy balances, and unit operation relations from previous flowsheet.

Semi-infinite constraint:

$$g(\mathbf{x}(\mathbf{u},\mathbf{p}),\mathbf{u},\mathbf{p})\equiv x_{G7}(\mathbf{u},\mathbf{p})-0.05$$

Uncertain Parameter (Inlet Gas Fraction)

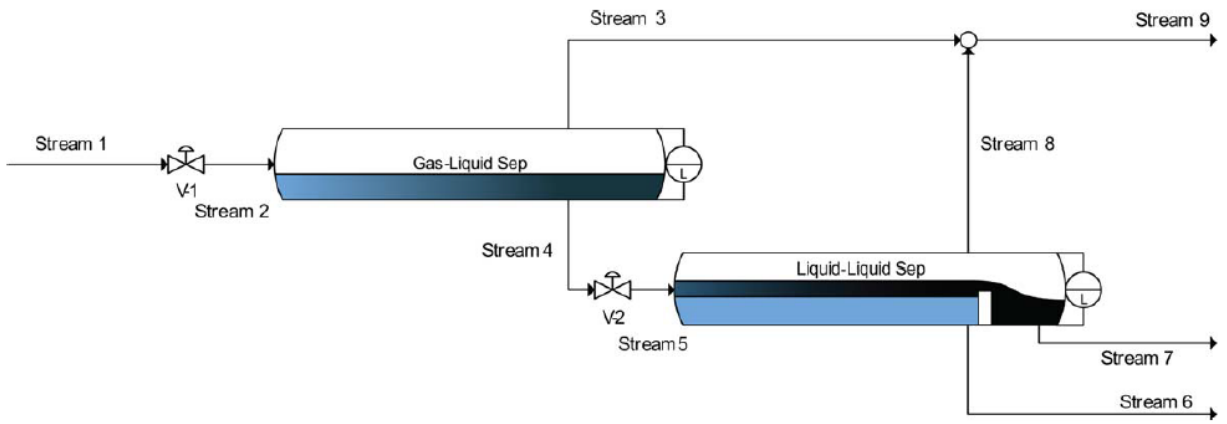
$$\mathbf{p}=(\xi_{G1})\in[0.35,0.5]$$

Control Settings (Valve Position)

$$\mathbf{u}=(u_1,u_2)\in[0,1]\times[0,1]$$

State Variables

$$\mathbf{z}=\left(\xi_{G4},\xi_{W4},\xi_{O4},\xi_{G7},\xi_{O7},H_{GLS},\dot{m}_3,\dot{m}_4,\dot{m}_6,\dot{m}_7,\dot{m}_8\right)$$



[1] **Worst-case design of subsea production facilities using semi-infinite programming.** Stuber, M.D. et al. (2014) *AIChE Journal*, 60, 2513-2524

Case Study III

Worst-Case Subsea Separator Design[1]

$$SG_O = \frac{141.5}{131.5 + API}$$

Inlet Specification

$$SG_{mix} = (\xi_{G1}/SG_G + \xi_{W1}/SG_W + \xi_{O1}/SG_O)^{-1}$$

$$\xi_{G1} + \xi_{W1} + \xi_{O1} = 1$$

Gas Mixer

$$\xi_4 = \xi_5$$

$$\dot{m}_9 = \dot{m}_3 + \dot{m}_8$$

$$P_9 = \min(P_3, P_8)$$

$$P_5 = P_{LLS}$$

$$\xi_4 = \xi_5$$

Valve 2

$$\dot{m}_4 = \dot{m}_5 = u_2 C_{v2} \sqrt{\frac{P_4 - P_5}{\rho_4 / \rho_W{}^o}}$$

$$\xi_6 = (0,1,0), \quad \xi_8 = (1,0,0), \quad \xi_{G7} + \xi_{O7} = 1$$

$$\dot{m}_5 = \dot{m}_6 + \dot{m}_7 + \dot{m}_8, \quad \xi_{G5}\dot{m}_5 = \xi_{G8}\dot{m}_8 + \xi_{G7}\dot{m}_7, \quad \xi_{W5}\dot{m}_5 = \xi_{W6}\dot{m}_6$$

$$\xi_{G7} = \xi_{G5} \exp\left[-k_{LLS} \frac{V_{oil}}{\dot{m}_7 / \rho_7}\right]$$

$$\rho_7 = \rho_W{}^o (\xi_{G7} / SG_G + \xi_{O7} / SG_O)^{-1}, \quad P_8 = P_{LLS}$$

Liquid-Liquid Separator

$$V_{oil} = V_{LLS}\left(\frac{\dot{m}_7}{\rho_7}\frac{\rho_5}{\dot{m}_5}\right)$$

$$V_{LLS} = L_{LLS}\left((H_{LLS} - R_{LLS})\sqrt{2R_{LLS}H_{LLS} - H_{LLS}{}^2} + R_{LLS}{}^2 \cos^{-1}\left[1 - \frac{H_{LLS}}{R_{LLS}}\right]\right)$$

$$P_1 = P_{well}$$

$$P_2 = P_{GLS}$$

$$\xi_1 = \xi_2$$

Valve 1

$$\dot{m}_1 = \dot{m}_2 = u_1 C_{v1} \sqrt{\frac{P_1 - P_2}{SG_{mix}}}$$

$$\xi_3 = (1,0,0), \quad \xi_{G4} + \xi_{O4} + \xi_{W4} = 1$$

$$\dot{m}_2 = \dot{m}_3 + \dot{m}_4, \quad \xi_{G2}\dot{m}_2 = \xi_{G3}\dot{m}_3 + \xi_{G4}\dot{m}_4, \quad \xi_{W2}\dot{m}_2 = \xi_{W4}\dot{m}_4$$

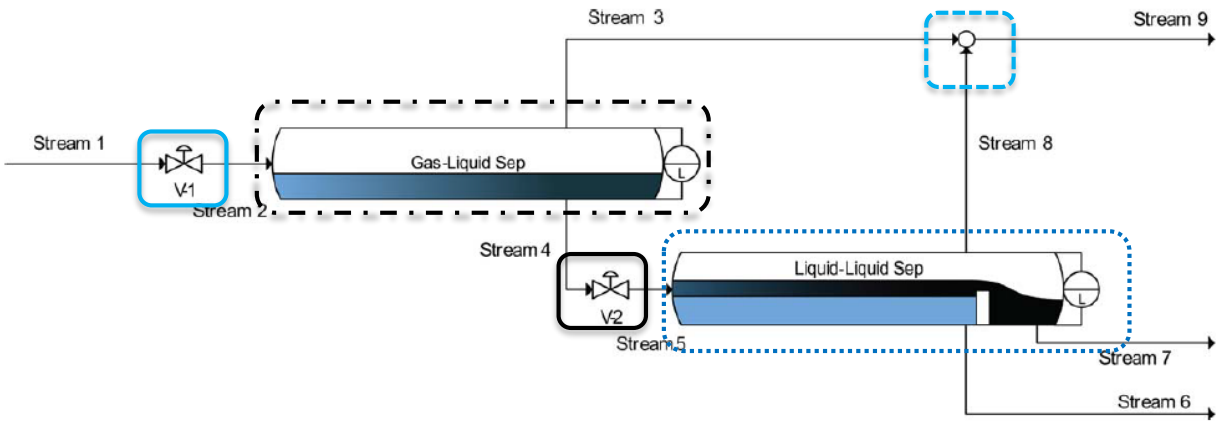
$$\xi_{G4} = \xi_{G2} \exp\left[-k_{GLS} \frac{V_{GLS}}{\dot{m}_4 / \rho_4}\right]$$

Gas-Liquid Separator

$$\rho_4 = \rho_W{}^o (\xi_{G4} / SG_G + \xi_{W4} / SG_W + \xi_{O4} / SG_O)^{-1}, \quad P_3 = P_2, \quad P_4 = P_3 + \rho_4 g_a H_{GLS}$$

$$V_{GLS} = L_{GLS}\left((H_{GLS} - R_{GLS})\sqrt{2R_{GLS}H_{GLS} - H_{GLS}{}^2} + R_{GLS}{}^2 \cos^{-1}\left[1 - \frac{H_{GLS}}{R_{GLS}}\right]\right)$$

$\dot{m}_2$ & ξ_2 *wellhead – specification*



Case Study III

Worst-Case Subsea Separator Design[1]

$$P_5 = P_{LLS}$$
$$\xi_4 = \xi_5$$
$$\dot{m}_4 = \dot{m}_5 = u_2 C_{v2} \sqrt{\frac{P_4 - P_5}{\rho_4 / \rho_W^{\,o}}}$$

- **Each of the four design case in the literature were run for Subsea Separator Design**
- **Good agreement with published results in all cases.**

```
using EAGOSemiInfinite
using EAGOFlowsheet
m = Flowsheet()
@define_parameters(m,
Cv2 = 1.67E-2,
PLLS = 4E6)
@define_equality(
$P5-$PLLS,
$x1-$x51,
$x2-$x52,
$x3-$x53,
$x7-$u2*$Cv2*sqrt(($P4-$PLLS)/($rho4/$pwo))
)
@define_state(x1,x2,x3,x7)
@define_internal(P5,rho4,pwo)
@define_control(u2)
@define_uncertain(u2)

###### INPUT FOR OTHER BLOCKS FOR PROCESS MODEL
###### ....

model = @generate_model(m)
value,point,info = SolveSIP(model)
```

Accessing EAGO & Future Plans

An intuitive graphical user interface - *EAGOFlowsheet*

- Equation-Oriented Flow-sheeting
- Graphical Manipulation of Directed Graphs
- Template Registration of New Relaxations
- AMPL Compatibility for Benchmarking

Explore and Contribute to the EAGO Package Library at

https://github.com/MatthewStuber/EAGO

Acknowledgements

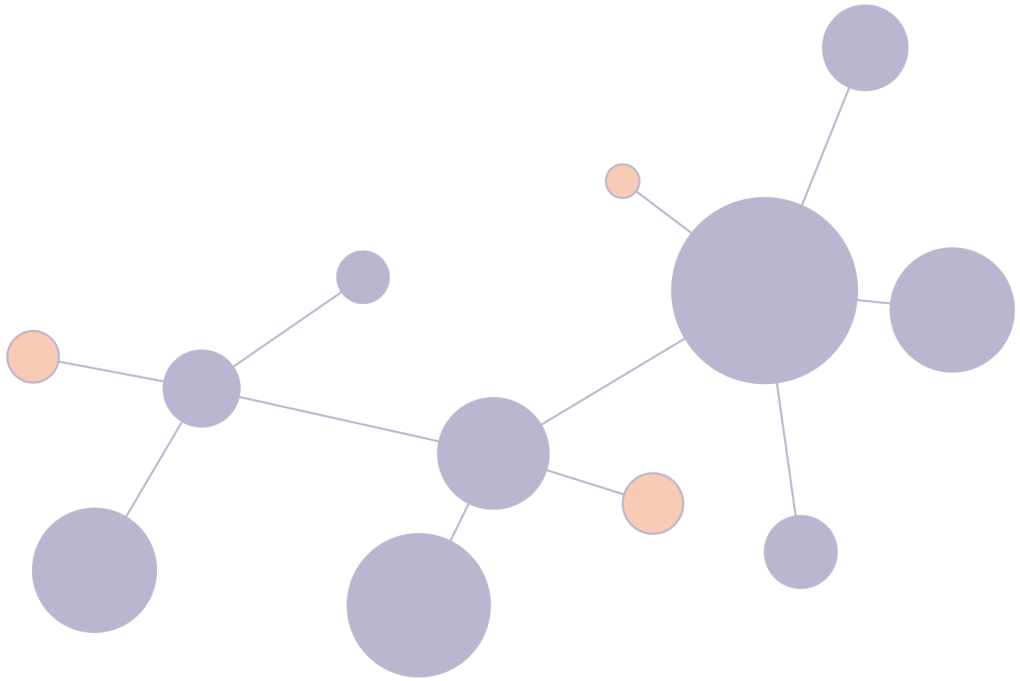
PSOR lab at UCONN (https://psor.uconn.edu/)

- Robert Ernst

Julia Development Community

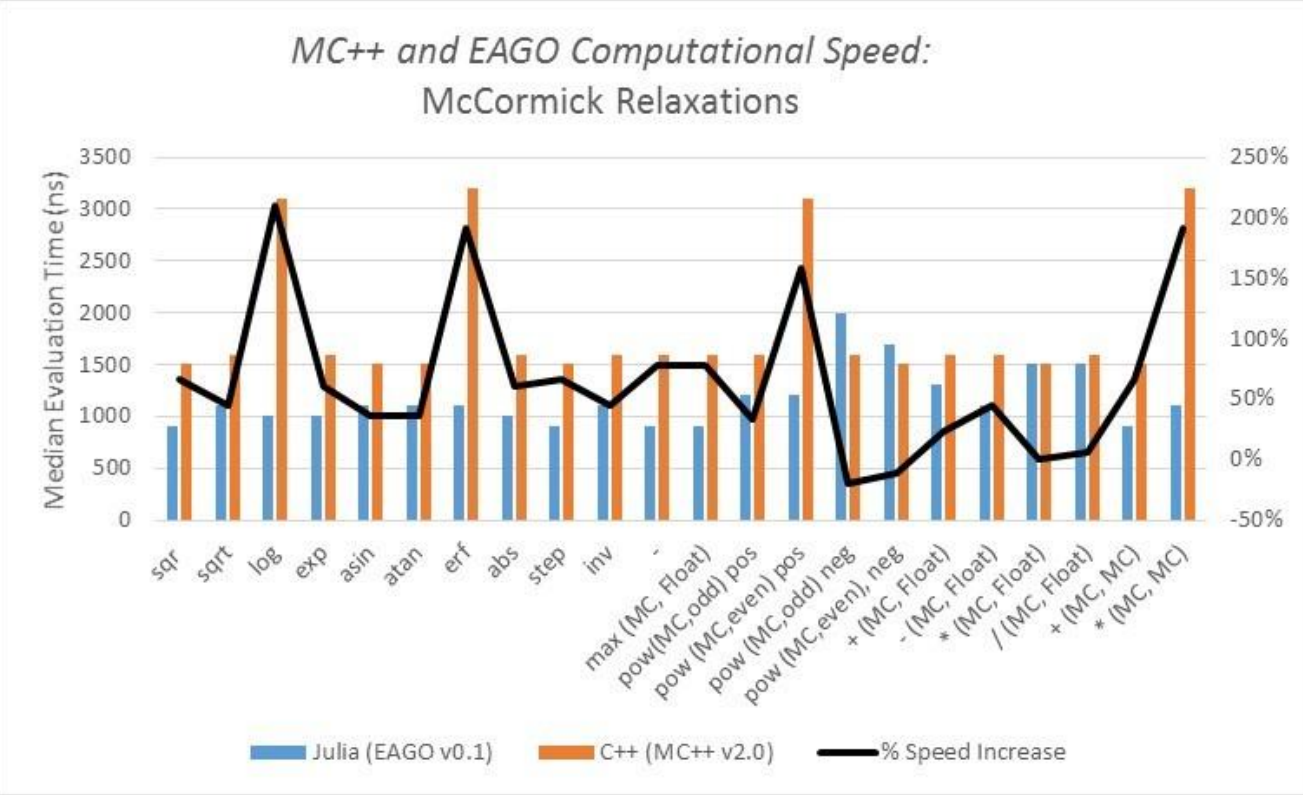
- Mike Innes
- David Saunders

University of Connecticut for funding this research



The End

Julia MC vs. MC++



- Using 64bit, Intel Xeon E3-1270 v5 @ 3.60 GHz, 32GB RAM
- Non-validated interval calculations
- Ubuntu/Linux 16.04 LTS and Julia v0.6
- Profiling 10,000 evaluations with recommend settings to limit clock noise