Recent Advances in EAGO.jl: Easy Advanced Global Optimization in Julia

Matthew D. Stuber, Assistant Professor,
stuber@alum.mit.edu
Matthew E. Wilhelm, PhD Candidate

EURO Athens 2021 Greece

Process Systems and Operations Research Laboratory

UCONN UNIVERSITY OF CONNECTICUT

Key Contributor

Matthew Wilhelm

PhD Candidate

PSOR Lab, Dept. of Chemical and Biomolecular Eng.

University of Connecticut

EAGO.jl developer

Outline

- Motivation
  - Reduced-space deterministic global optimization
- EAGO.jl: Deterministic global optimization in Julia
  - Core features
  - Main features for advanced formulations
  - New and near-future additions

- Conclusions

Motivation: Reduced-Space Optimization

Want to solve dynamic optimization problems to guaranteed global optimality:

$$
\begin{aligned}
\phi^* = \min_{\mathbf{p} \in P \subset \mathbb{R}^{n_p}} \ & \phi(\mathbf{x}(\mathbf{p}, t_f), \mathbf{p}) \\
\text{s.t.} \ & \dot{\mathbf{x}}(\mathbf{p}, t) = \mathbf{f}(\mathbf{x}(\mathbf{p}, t), \mathbf{p}, t), \ \forall t \in I = [t_0, t_f] \\
& \mathbf{x}(\mathbf{p}, t_0) = \mathbf{x}_0(\mathbf{p}) \\
& \mathbf{g}(\mathbf{x}(\mathbf{p}, t_f), \mathbf{p}) \leq \mathbf{0}
\end{aligned}
$$

Motivation: Reduced-Space Optimization

Want to solve dynamic optimization problems to guaranteed global optimality:

$$
\begin{aligned}
\phi^* = &\min_{\mathbf{p} \in P \subset \mathbb{R}^{n_p}} \phi(\mathbf{x}(\mathbf{p}, t_f), \mathbf{p}) \\
\text{s.t.}\ & \boxed{\begin{aligned} &\dot{\mathbf{x}}(\mathbf{p}, t) = \mathbf{f}(\mathbf{x}(\mathbf{p}, t), \mathbf{p}, t),\ \forall t \in I = [t_0, t_f] \\ &\mathbf{x}(\mathbf{p}, t_0) = \mathbf{x}_0(\mathbf{p}) \end{aligned}} \\
& \mathbf{g}(\mathbf{x}(\mathbf{p}, t_f), \mathbf{p}) \leq \mathbf{0}
\end{aligned}
$$

Parametric ordinary differential equation initial value problem (ODE-IVP) constraints.

Arise from optimal control, parameter estimation, etc.

Motivation: Reduced-Space Optimization

Want to solve dynamic optimization problems to guaranteed global optimality:

$$\dot{\mathbf{x}}(\mathbf{p},t) = \mathbf{f}(\mathbf{x}(\mathbf{p},t),\mathbf{p},t),\ \forall t \in I = [t_0, t_f]$$
$$\mathbf{x}(\mathbf{p},t_0) = \mathbf{x}_0(\mathbf{p})$$

$$\begin{aligned}
\mathbf{z}_0 &= \mathbf{x}_0(\mathbf{p})\\
\hat{\mathbf{z}}_1 - \mathbf{z}_0 - h\mathbf{f}(\hat{\mathbf{z}}_1,\mathbf{p},t_1) &= \mathbf{0}\\
\vdots \qquad \vdots &\\
\hat{\mathbf{z}}_K - \hat{\mathbf{z}}_{K-1} - h\mathbf{f}(\hat{\mathbf{z}}_K,\mathbf{p},t_K) &= \mathbf{0}
\end{aligned}$$

Discrete-time reformulation (implicit Euler)

P

constraints.

Arise from optimal control, parameter estimation, etc.

Motivation: Reduced-Space Optimization

Want to solve dynamic optimization problems to guaranteed global optimality:

$$\phi^* = \min_{\mathbf{p}\in P\subset\mathbb{R}^{n_p}} \phi(\mathbf{x}(\mathbf{p},t_f),\mathbf{p})$$

$$\text{s.t.}\ \ \dot{\mathbf{x}}(\mathbf{p},t) = \mathbf{f}(\mathbf{x}(\mathbf{p},t),\mathbf{p},t),\ \forall t\in I=[t_0,t_f]$$

$$\mathbf{x}(\mathbf{p},t_0) = \mathbf{x}_0(\mathbf{p})$$

$$\mathbf{g}(\mathbf{x}(\mathbf{p},t_f),\mathbf{p}) \leq \mathbf{0}$$

Dimensionality: n_p

$$\phi^* = \min_{\mathbf{p}\in P,\hat{\mathbf{z}}\in Z} \phi(\hat{\mathbf{z}},\mathbf{p},t_f)$$

$$\text{s.t.}\ \ \mathbf{z}_0 = \mathbf{x}_0(\mathbf{u},\mathbf{p})$$

$$\hat{\mathbf{z}}_1 - \mathbf{z}_0 - h\mathbf{f}(\hat{\mathbf{z}}_1,\mathbf{p},t_1) = \mathbf{0}$$

$$\vdots \qquad \vdots$$

$$\hat{\mathbf{z}}_K - \hat{\mathbf{z}}_{K-1} - h\mathbf{f}(\hat{\mathbf{z}}_K,\mathbf{p},t_K) = \mathbf{0}$$

$$\mathbf{g}(\hat{\mathbf{z}}_K,\mathbf{p}) \leq \mathbf{0}$$

Dimensionality: $n_p \times K$

Motivation: Reduced-Space Optimization

Want to solve dynamic optimization problems to guaranteed global optimality:

$$\phi^* = \min_{\mathbf{p}\in P\subset\mathbb{R}^{n_p}} \phi(\mathbf{x}(\mathbf{p},t_f),\mathbf{p})$$
$$\text{s.t. } \dot{\mathbf{x}}(\mathbf{p},t) = \mathbf{f}(\mathbf{x}(\mathbf{p},t),\mathbf{p},t),\ \forall t\in I=[t_0,t_f]$$
$$\mathbf{x}(\mathbf{p},t_0) = \mathbf{x}_0(\mathbf{p})$$
$$\mathbf{g}(\mathbf{x}(\mathbf{p},t_f),\mathbf{p}) \le \mathbf{0}$$

Dimensionality: n_p

$$\phi^* = \min_{\mathbf{p}\in P,\hat{\mathbf{z}}\in Z} \phi(\hat{\mathbf{z}},\mathbf{p},t_f)$$
$$\text{s.t. } \mathbf{z}_0 = \mathbf{x}_0(\mathbf{u},\mathbf{p})$$
$$\hat{\mathbf{z}}_1 - \mathbf{z}_0 - h\mathbf{f}(\hat{\mathbf{z}}_1,\mathbf{p},t_1) = \mathbf{0}$$
$$\vdots \qquad \vdots$$
$$\hat{\mathbf{z}}_K - \hat{\mathbf{z}}_{K-1} - h\mathbf{f}(\hat{\mathbf{z}}_K,\mathbf{p},t_K) = \mathbf{0}$$
$$\mathbf{g}(\hat{\mathbf{z}}_K,\mathbf{p}) \le \mathbf{0}$$

Dimensionality: $n_p \times K$

$$\phi^* = \min_{\mathbf{p}\in P} \phi(\mathbf{z}(\mathbf{p}),\mathbf{p},t_f)$$
$$\text{s.t. } \mathbf{g}(\mathbf{z}_K(\mathbf{p}),\mathbf{p}) \le \mathbf{0}$$

$$\mathbf{z}(\mathbf{p}) = (\mathbf{z}_0(\mathbf{p}),\mathbf{z}_1(\mathbf{p}),\ldots,\mathbf{z}_K(\mathbf{p}))$$

Background: EAGO

How do you get EAGO?

From Julia package manager:

```
(@v1.6) pkg> add EAGO
```

```
julia> using Pkg;

julia> Pkg.add("EAGO")
```

From GitHub:

https://www.github.com/PSORLab/EAGO.jl

Background: EAGO

How do you get EAGO?

From Julia package manager:

```
(@v1.6) pkg> add EAGO
```

```
julia> using Pkg;

julia> Pkg.add("EAGO")
```

From GitHub:

https://www.github.com/PSORLab/EAGO.jl

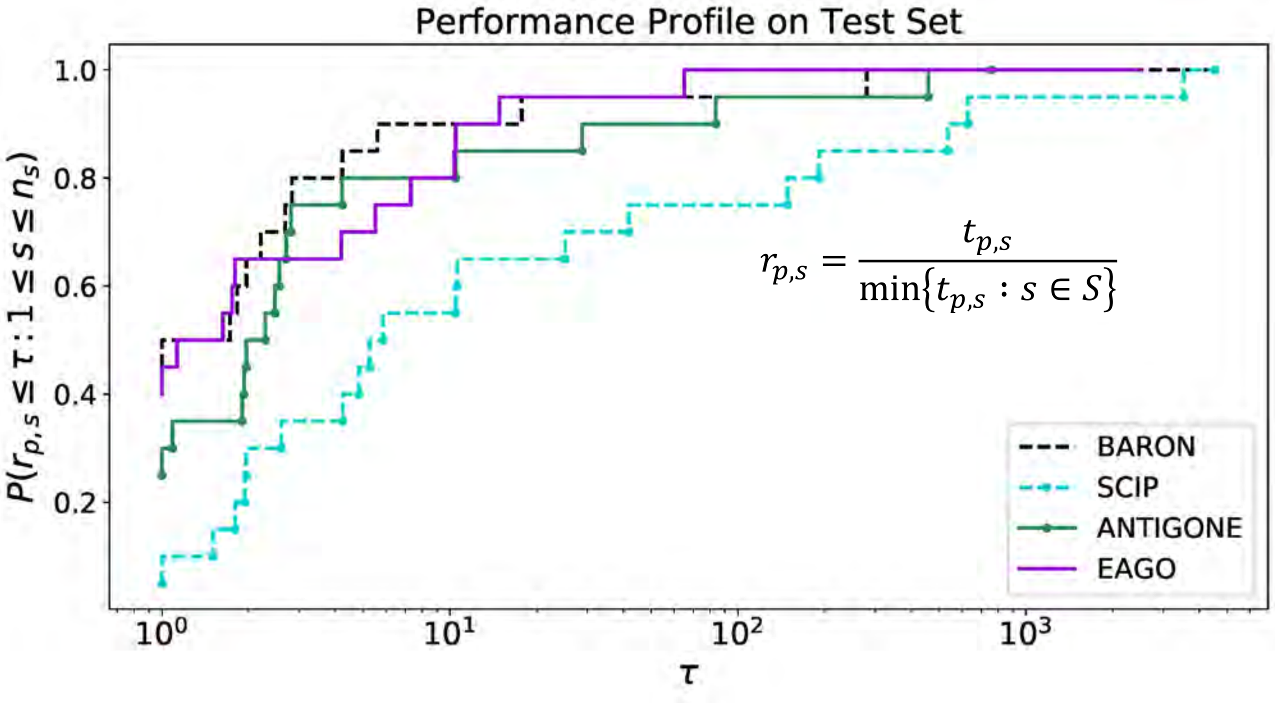
How do you use EAGO?

As a solver in the open-source algebraic modeling language JuMP.

As a stand-alone solver.

Published Results

- EAGO exhibits competitive performance on benchmarking set



$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in S\}}$$

OPTIMIZATION METHODS & SOFTWARE
https://doi.org/10.1080/10556788.2020.1786566

Taylor & Francis
Taylor & Francis Group

Check for updates

EAGO.jl: easy advanced global optimization in Julia

M. E. Wilhelm and M. D. Stuber

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

ABSTRACT

An extensible open-source deterministic global optimizer (EAGO) programmed entirely in the Julia language is presented. EAGO was developed to serve the need for supporting higher-complexity user-defined functions (e.g. functions defined implicitly via algorithms) within optimization models. EAGO embeds a first-of-its-kind implementation of McCormick arithmetic in an Evaluator structure allowing for the construction of convex/concave relaxations using a combination of source code transformation, multiple dispatch, and context-specific approaches. Utilities are included to parse user-defined functions into a directed acyclic graph representation and perform symbolic transformations enabling dramatically improved solution speed. EAGO is compatible with a wide variety of local optimizers, the most exhaustive library of transcendental functions, and allows for easy accessibility through the JuMP modelling language. Together with Julia's minimalist syntax and competitive speed, these powerful features make EAGO a versatile research platform enabling easy construction of novel meta-solvers, incorporation and utilization of new relaxations, and extension to advanced problem formulations encountered in engineering and operations research (e.g. multilevel problems, user-defined functions). The applicability and flexibility of this novel software is demonstrated on a diverse set of examples. Lastly, EAGO is demonstrated to perform comparably to state-of-the-art commercial optimizers on a benchmarking test set.

ARTICLE HISTORY
Received 15 January 2020
Accepted 15 June 2020

KEYWORDS
Deterministic global optimization; nonconvex programming; McCormick relaxations; optimization software; branch-and-bound; Julia

2010 MATHEMATICS SUBJECT CLASSIFICATIONS
90C26; 90C34; 90C57; 90C90

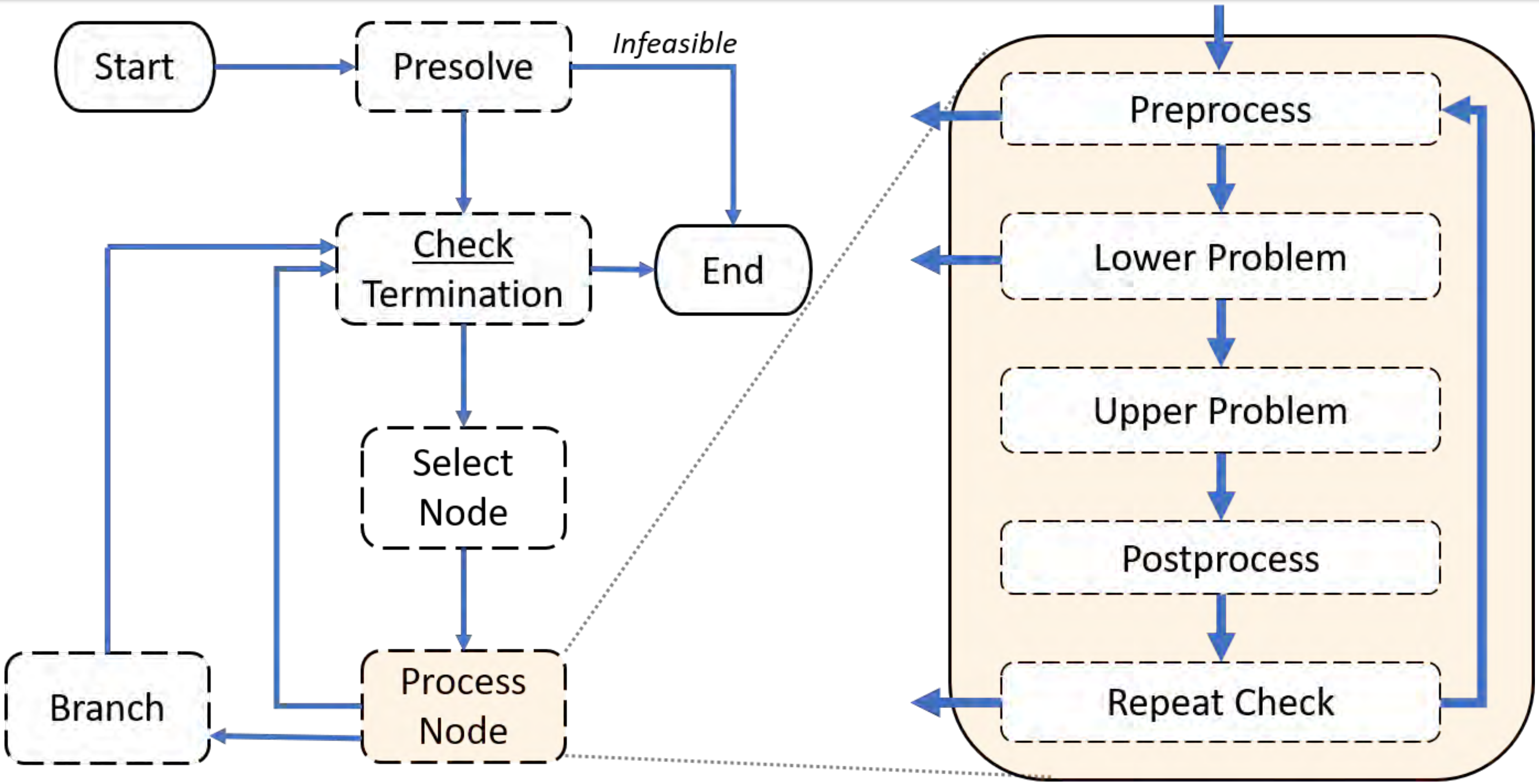
1. Introduction and motivation

Mathematical optimization problems are ubiquitous in scientific and technical fields. Applications range from aerospace and chemical process systems to finance. However, even relatively simple physical processes such as mixing, may introduce significant nonconvexity into problem formulations [60]. As such, nonconvex programs often represent the most faithful representations of the system of interest. Multiple approaches have been developed to address these cases. Heuristics such as evolutionary algorithms, may approximate good solutions for select problems. However, heuristics may fail to guarantee that even a feasible

CONTACT M. D. Stuber stuber@alum.mit.edu Process Systems and Operations Research Laboratory, Department of Chemical and Biomolecular Engineering, University of Connecticut, 191 Auditorium Rd, Unit 3222, Storrs, CT 06269-3222, USA

Supplemental data for this article can be accessed here. https://doi.org/10.1080/10556788.2020.1786566

EAGO.jl: Core Optimizer



EAGO.jl: Core Optimizer

Key Improvements to Global Optimization Routine:

- Heuristics to ensure numerically safe affine relaxations for lower-bounding problems
- More computationally efficient approach to optimization-based bounds tightening
- No-overhead user-defined subroutines (lower-bounding problem, etc.)
- Improved parameter tuning

Other High-Level Improvements to EAGO's Global Optimizer:

- Bridging + configuration for a large variety of subsolvers
- Preliminary support for integer-variables (MINLP problem forms)
- Detection of specialized problem forms (LP, MILP, convex)
- Support for additional semi-infinite programming (SIP) routines

Improvements to MINLP solution algorithm currently under development.

EAGO.jl: McCormick Relaxations

$$\mathbf{y} = \mathbf{f}(\mathbf{g}(\mathbf{x}), \dots, \mathbf{h}(\mathbf{x}))$$

Relaxations of g(x) at $\mathbf{x} \in \mathbf{X}$

• • •

Relaxations of h(x) at $\mathbf{x} \in \mathbf{X}$

Apply **f** *composite relaxation rules*

Relaxations of $\mathbf{f}(\cdot, \dots, \cdot)$ at $\mathbf{x} \in \mathbf{X}$

- **EAGO generates relaxations of complicated nonlinear expressions using a McCormick relaxation methodology**

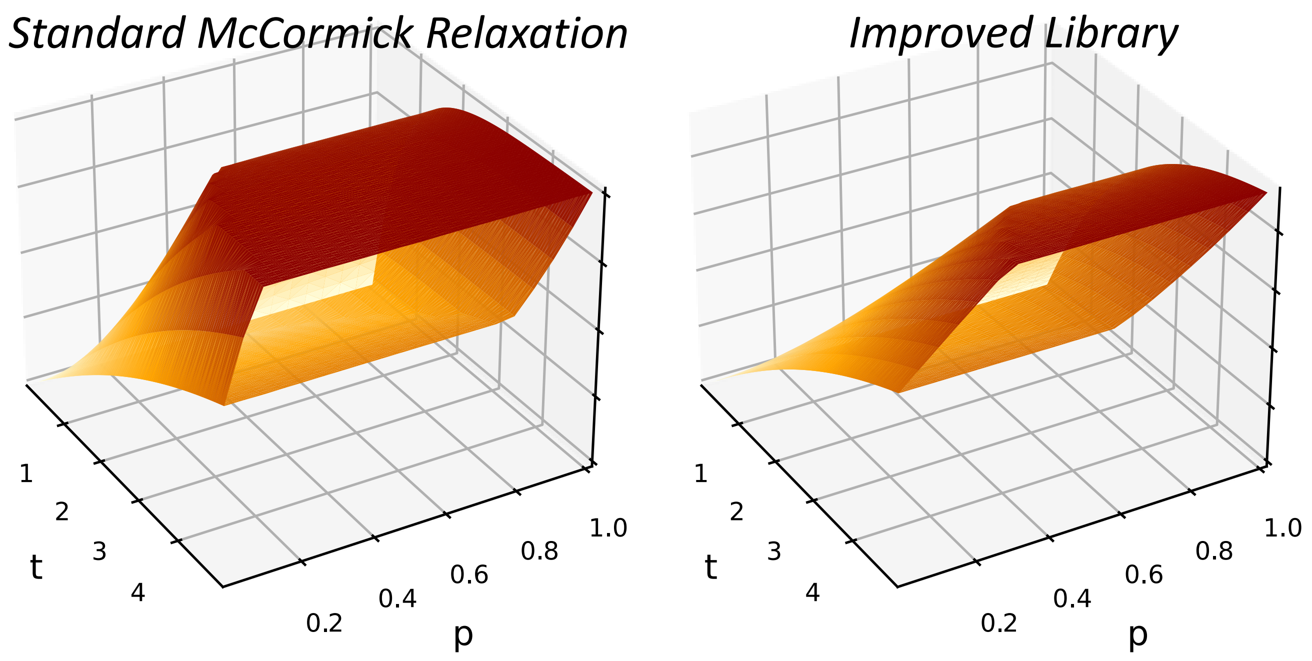
- **Improvements EAGO's intrinsic library of relaxations which include**
  - Improved relaxations (tighter) of composite bilinear and trilinear terms*
  - Commonly encountered subexpressions: x*log(x), x*exp(x), erf(x), and more…
  - Envelopes for common activation functions**

*In preparation ** Under Review

EAGO.jl: Novel Relaxations

Improved relaxation subroutine performance due to intrinsic relaxation library upgrade:

- Relaxation of implicit functions[10]
- Relaxations of ODEs[11]
- Reverse propagation of relaxations[12]



Simple ODE Relaxation

$$\frac{dx}{dt} = \exp(p)\sin(x)(2-x),$$

$$x(0) = 1, \quad p \in [0.01,1], \quad t \in [0,5]$$

10. Stuber, MD et al. **Convex and concave relaxations of implicit functions.** *Optimization Methods and Software* (2015), 30, 424-460
11. Scott, Joseph K., and Paul I. Barton. **Improved relaxations for the parametric solutions of ODEs using differential inequalities.** *Journal of Global Optimization* 57.1 (2013): 143-176.
12. Wechsung, Achim, et al. **Reverse propagation of McCormick relaxations.** *Journal of Global Optimization* 63.1 (2015): 1-36.

EAGO.jl: Main Features

- ***Embedded Machine Learning (ML) Models***
- Semi-infinite Programming
- Dynamic Optimization

… Composability thereof

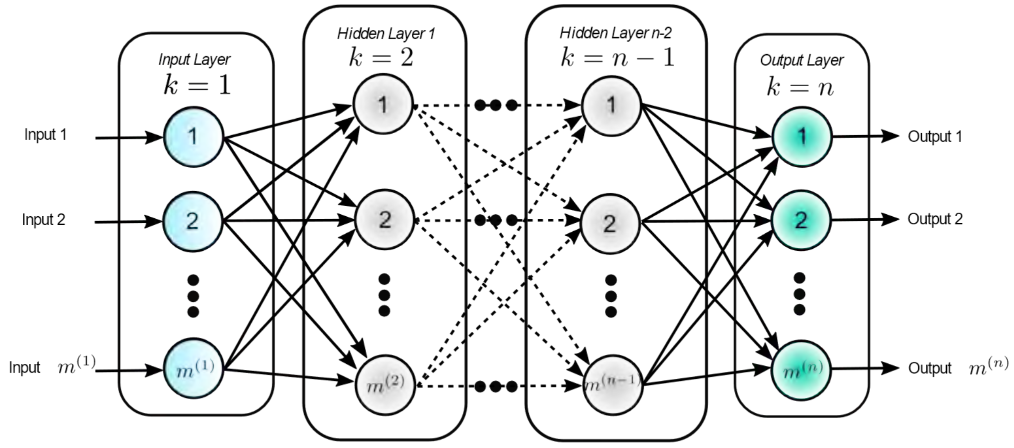
EAGO.jl: Embedded ML-models

- **Support for embedded general feedforward neural networks**

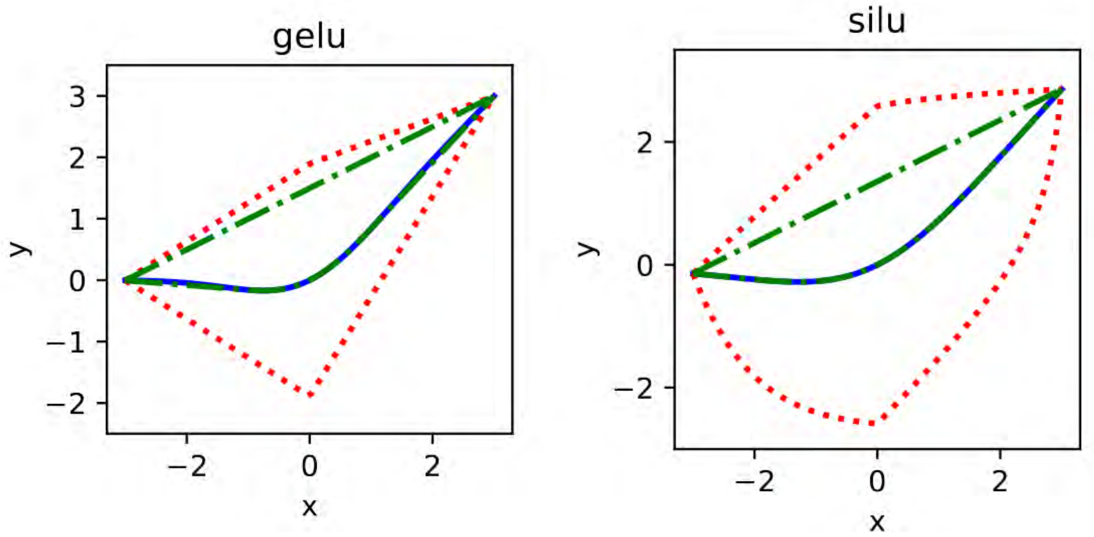
  Multilayer Perceptron, Deep Residual Networks, etc.

- **Incorporate many common Flux.jl trained models**

Multilayer Perceptron



Envelopes of Relaxation Functions



EAGO.jl: Embedded ML-models

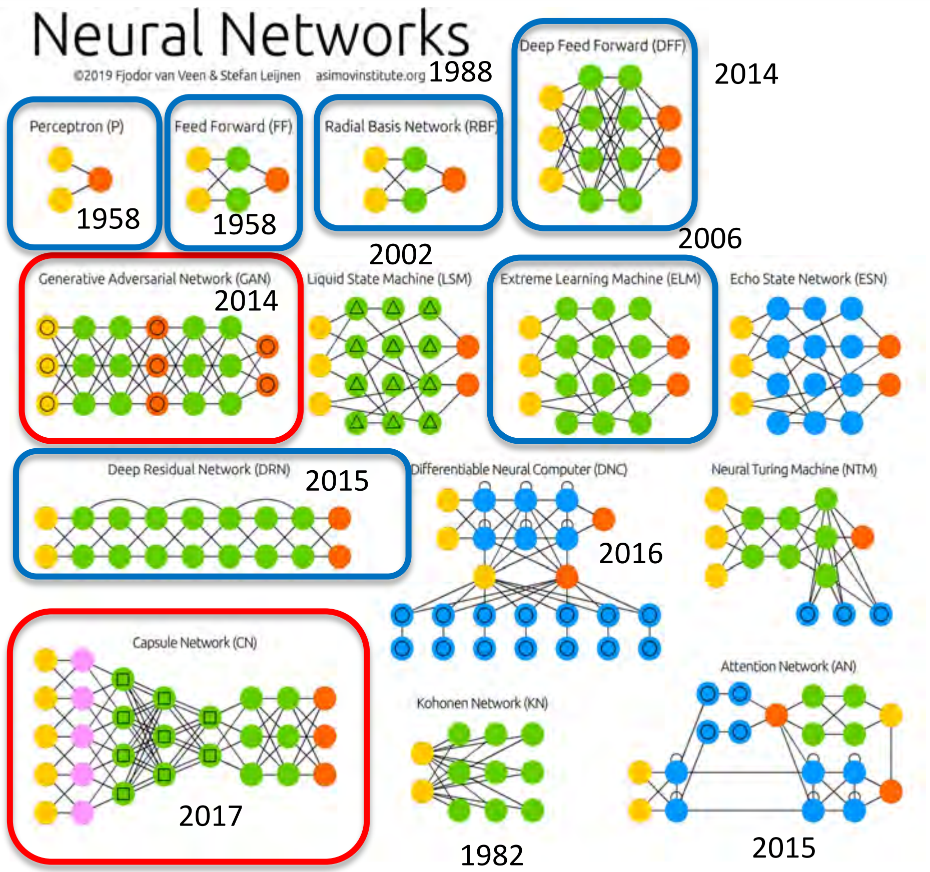
- Future support for specialized surrogate models
- Implicit function centered machine learning architectures

  i. Deep equilibrium networks
  ii. Implicit deep learning

$$\hat{y}\,(u) = Cx + Du$$
$$x = \phi(Ax + Bu)$$

  iii. Fixed-point equation-based networks

Enabled by core optimizer development.
Expected by end of August 2021.



Adapted from https://www.asimovinstitute.org/neural-network-zoo/

https://arxiv.org/pdf/1908.06315.pdf, https://arxiv.org/abs/1909.01377,

EAGO.jl: Main Features

- Embedded Machine Learning (ML) Models
- ***Semi-infinite Programming***
- Dynamic Optimization

… Composability thereof

Solving Nonconvex SIPs

- EAGO.jl supports general nonconvex SIPs:

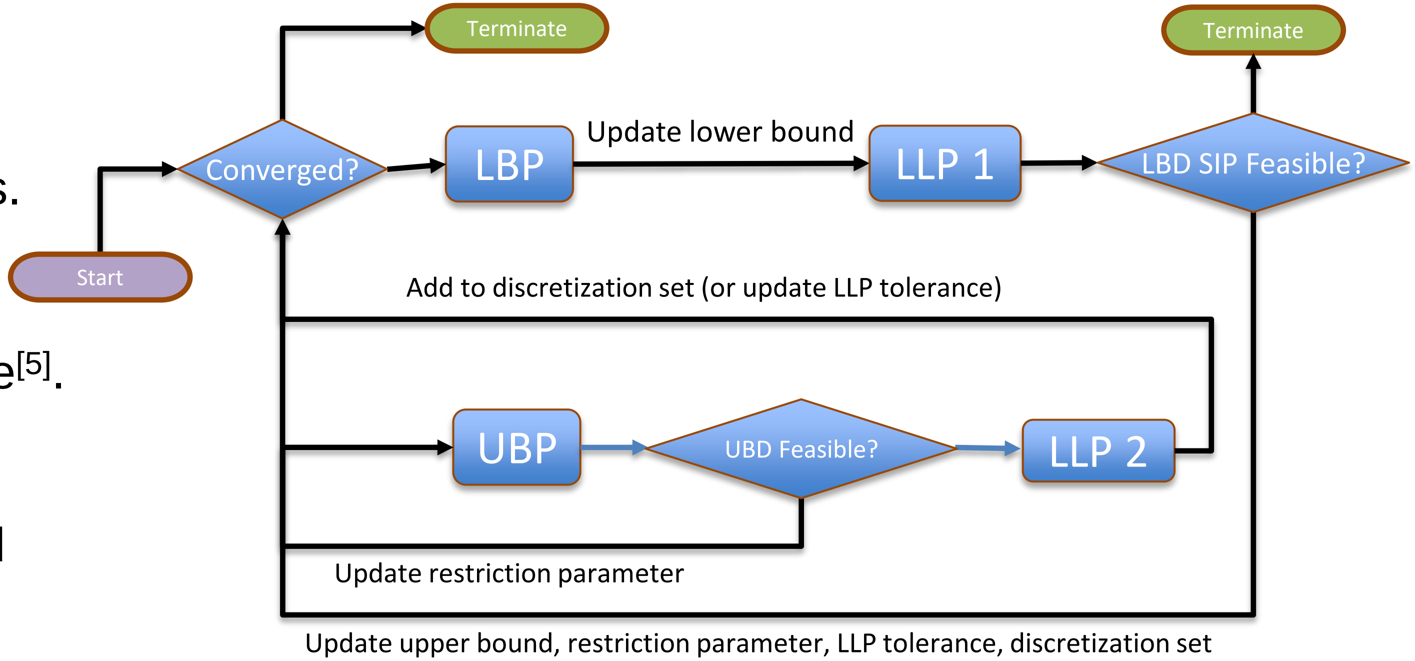
$$
\begin{aligned}
f^* = &\min_{\mathbf{x}\in X} f(\mathbf{x}) \\
\text{s.t.}\quad &g(\mathbf{x},\mathbf{p}) \le 0, \forall(\mathbf{p}) \in P
\end{aligned}
$$

- Composable with ML/dynamic relaxations.

New Features:

- Added new hybrid-oracle SIP routine[5].
- Automatic subproblem tolerance specification (SIPResRev SIPHybrid algorithms)[5].
- User-extendable SIP subproblems.

SIPres algorithm[4]



3. B. Bhattacharjee, P. Lemonidis, W.H. Green Jr, and P.I. Barton. **Global solution of semi-infinite programs.** *Math. Program*. 103 (2005), pp. 283–307.
4. Mitsos, Alexander. **Global optimization of semi-infinite programs via restriction of the right-hand side.** *Optimization* 60.10-11 (2011): 1291-1308.
5. Djelassi, Hatim, and Alexander Mitsos. **A hybrid discretization algorithm with guaranteed feasibility for the global solution of semi-infinite programs.** *Journal of Global Optimization* 68.2 (2017): 227-253.

EAGO.jl: Main Features

- Embedded Machine Learning (ML) Models
- Semi-infinite Programming
- ***Dynamic Optimization***

… Composability thereof

Dynamics, Relaxation

- New support included for general nonlinear parametric ordinary differential equations.

- Incorporation into global optimizer:
  - Relaxations & Domain Reduction:
    - Interval bounds
    - Relaxations & (sub)gradients
  - Local NLP solver:
    - Automatic differentiation for upper-bounding problem

Continuous-Time Relaxations[10,11,12]

$$\dot{\mathbf{x}}^{cv}(t,\mathbf{p}) = \mathbf{f}^{cv}(t,\mathbf{p},\mathbf{x}^{cv}(t,\mathbf{p}),\mathbf{x}^{cc}(t,\mathbf{p})), \quad \mathbf{x}^{cv}(t_0,\mathbf{p}) = \mathbf{x}_0^{cv}(\mathbf{p})$$
$$\dot{\mathbf{x}}^{cc}(t,\mathbf{p}) = \mathbf{f}^{cc}(t,\mathbf{p},\mathbf{x}^{cv}(t,\mathbf{p}),\mathbf{x}^{cc}(t,\mathbf{p})), \quad \mathbf{x}^{cc}(t_0,\mathbf{p}) = \mathbf{x}_0^{cc}(\mathbf{p})$$

Discrete-Time Relaxations[13,14,15]

$$\mathbf{x}(\tau_{q+1},\mathbf{p}) \in \underbrace{\mathbf{x}(\tau_q,\mathbf{p}) + \sum_{j=1}^{p} \frac{h^j}{j!}\mathbf{f}^{(j)}(\mathbf{x}(\tau_q,\mathbf{p}),\mathbf{p})}_{\text{Taylor Series}} + \underbrace{\frac{h^{p+1}}{(p+1)!}\mathbf{f}^{(p+1)}(\mathrm{X}(\tau_q),\mathrm{P})}_{\text{Remainder Bound}}$$

10. Scott, Joseph K., and Paul I. Barton. **Improved relaxations for the parametric solutions of ODEs using differential inequalities.** *Journal of Global Optimization* 57.1 (2013): 143-176.
11. Scott, Joseph K., and Paul I. Barton. **Bounds on the reachable sets of nonlinear control systems.** *Automatica* 49.1 (2013): 93-100.
12. Scott, Joseph K., Benoit Chachuat, and Paul I. Barton. **Nonlinear convex and concave relaxations for the solutions of parametric ODEs.** *Optimal Control Applications and Methods* 34.2 (2013): 145-163.
13. Sahlodin, Ali M., and Benoit Chachuat. **Discretize-then-relax approach for convex/concave relaxations of the solutions of parametric ODEs.** *Applied Numerical Mathematics* 61.7 (2011): 803-820.
14. Berz, Martin, and Georg Hoffstätter. **Computation and application of Taylor polynomials with interval remainder bounds.** *Reliable Computing* 4.1 (1998): 83-97.
15. Sahlodin, Ali Mohammad, and Benoit Chachuat. **Convex/concave relaxations of parametric ODEs using Taylor models.** *Computers & Chemical Engineering* 35.5 (2011): 844-857.

Dynamics, Implementation

Core Algorithms

- **DynamicBoundspODEsDiscrete.jl**
  – Discrete time approaches
- **DynamicBoundspODEsIneq.jl**
  -- Continuous time approaches

Abstraction Layer

DynamicBounds.jl[34]

- **DynamicBounds.jl**
- **DynamicBoundsBase.jl**

Extendable Global Optimizer[35]

EAGO

EAGODynamicOptimizer.jl[36]

Future Work: Integrate with JuMP-based frontends (e.g., InfiniteOpt.jl[37])

34. Wilhelm, M. E., **DynamicBounds.jl**, (2020), GitHub repository, https://github.com/PSORLab/DynamicBounds.jl
35. Wilhelm, M. E., and M. D. Stuber. **EAGO. jl: easy advanced global optimization in Julia.** *Optimization Methods and Software* (2020): 1-26.
36. Wilhelm, M. E., **EAGODynamicOptimizer.jl**, (2020), GitHub repository, https://github.com/PSORLab/EAGODynamicOptimizer.jl
37. Pulsipher, J.L., et al. A Unifying Modeling Abstraction for Infinite-Dimensional Optimization, https://arxiv.org/abs/2106.12689

EAGO.jl: Main Features

- Embedded Machine Learning (ML) Models
- Semi-infinite Programming
- Dynamic Programming

… Composability thereof

Robust Dynamic Optimization

Dynamic SIP Formulation

$$\Phi^* = \min_{\mathbf{u}} \Phi(\mathbf{u})$$ ← Objective

$$\text{s.t. } g(\mathbf{x}(\mathbf{u},\mathbf{p},\mathrm{t}_f),\mathbf{u},\mathbf{p}) \leq 0$$ ← Performance Constraint(s)

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(\mathbf{u},\mathbf{p},\mathrm{t}),\mathbf{u},\mathbf{p})$$ ← Parametric ODEs

$$\mathbf{x}(\mathbf{u},\mathbf{p},\mathrm{t}_0) = \mathbf{x}_0(\mathbf{u},\mathbf{p})$$ ← Initial Condition

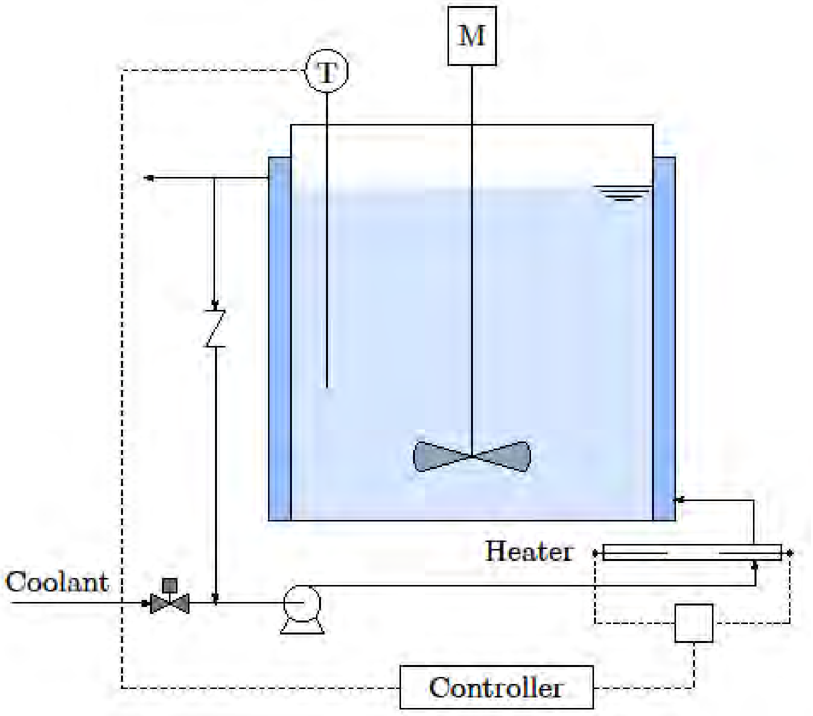
$$t \in I = [t_0, t_f], \forall \mathbf{p} \in P$$

- Design under worst-case realization of uncertainty.
- Safety-critical systems and high-impact defect elimination.

1. Puschke, Jennifer, et al. **Robust dynamic optimization of batch processes under parametric uncertainty: Utilizing approaches from semi-infinite programs.** *Computers & Chemical Engineering* 116 (2018): 253-267.
2. Puschke, Jennifer, and Alexander Mitsos. **Robust feasible control based on multi-stage eNMPC considering worst-case scenarios.** *Journal of Process Control* 69 (2018): 8-15.

Robust Dynamic Optimization

Batch MMA Polymerization Reaction



Adequate cooling at maximum temperature to withstand sensor fault?

Robust Operation SIP

$$\gamma^* = \max_{p \in P, \gamma \in \Gamma} \gamma$$
$$\text{s.t. } \gamma \leq T(t_f, \mathbf{u}) - p, \ \forall \mathbf{u} \in U$$

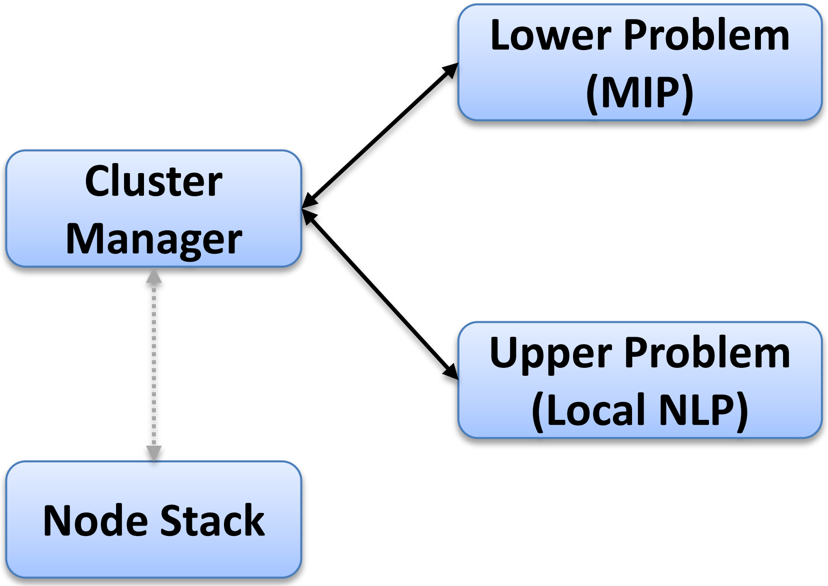
- ❑ Nonconvex semi-infinite program
- ❑ Embedded dynamic system
- ❑ Complex chemical kinetics (hybrid model desirable)

EAGO.jl: Distributed Computing

Overall Framework:

Currently, under development. Expected by end of 2021.

- ➢ Support for distributed computing via ClusterManager abstraction
- ➢ User-specified entry point for parallelism
    - Parallel evaluation of relaxations
    - Parameter setting in optimizers used by subproblem
    - Lower bounding, upper bounding subproblems



Conclusions

- EAGO is an extensible deterministic global optimization solver
  - Architected specifically for user-defined functions and routines
  - Performance comparable with state-of-the-art solvers
  - Open-source and free for non-commercial use
- Now and Near Future:
  - Exhaustive library of relaxation envelopes
  - Additional relaxations (αBB and AVM)
  - Release of dynamic optimization (optimal control) package
  - Implicit SIP algorithm (for simulation-based problems)
  - Integer variables
- Feature requests welcome on our GitHub!

Thank You – Any Questions?

- PSORLab@UCONN
- EURO 2021 Organizers
- Funding: National Science Foundation

This material is based upon work supported by the National Science Foundation under Grant No. 1932723. 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.

https://www.psor.uconn.edu

https://www.github.com/PSORLab/EAGO.jl

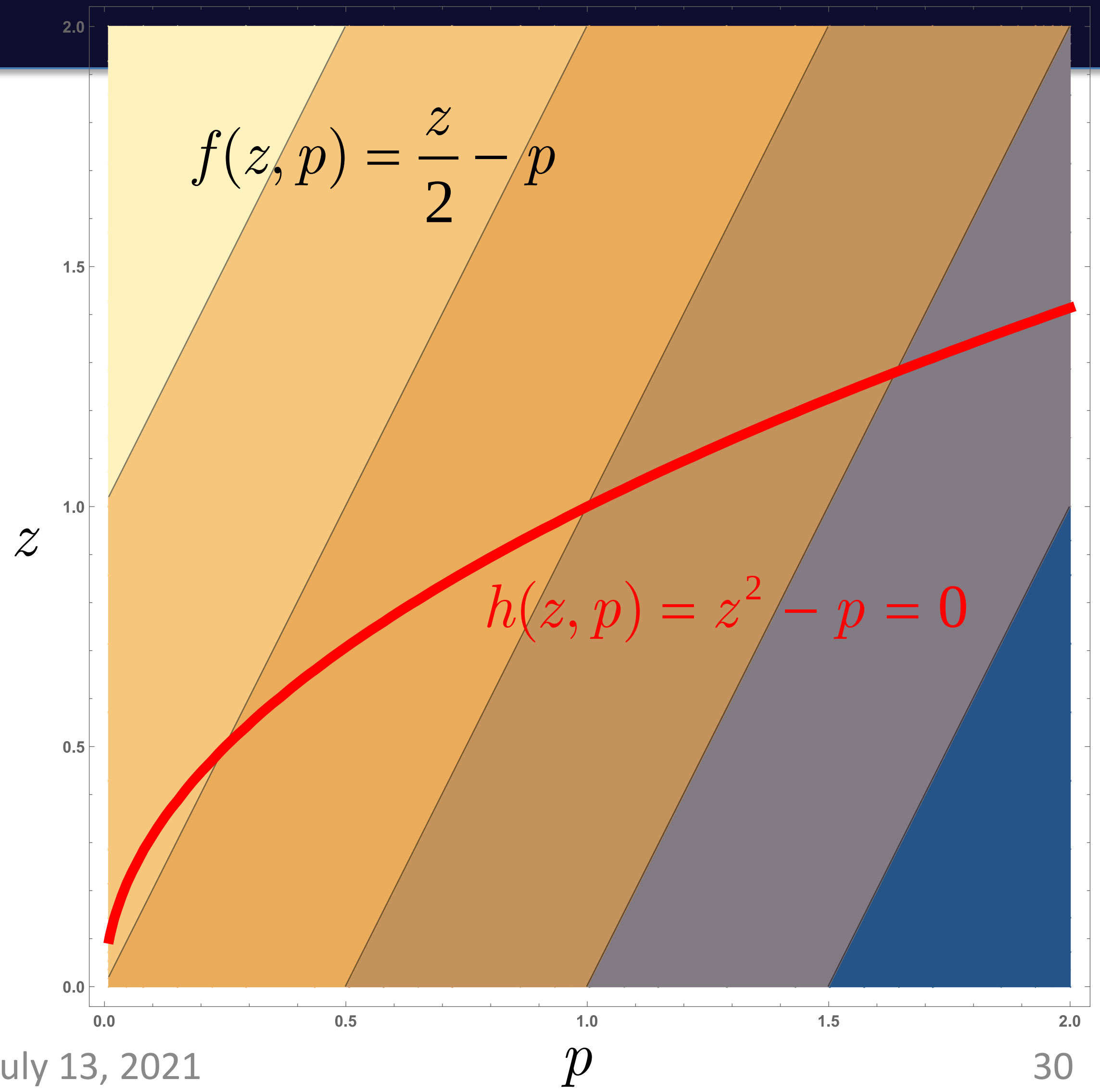
UCONN
UNIVERSITY OF CONNECTICUT

EURO ATHENS 2021 GREECE HYBRID

Process Systems and Operations Research Laboratory

Motivation: Reduced-Space Optimization

$$\min_{z,p} \frac{z}{2} - p$$
$$\text{s.t.} \quad z^2 - p = 0$$
$$z \in [0, 2]$$
$$p \in [0.01, 2]$$

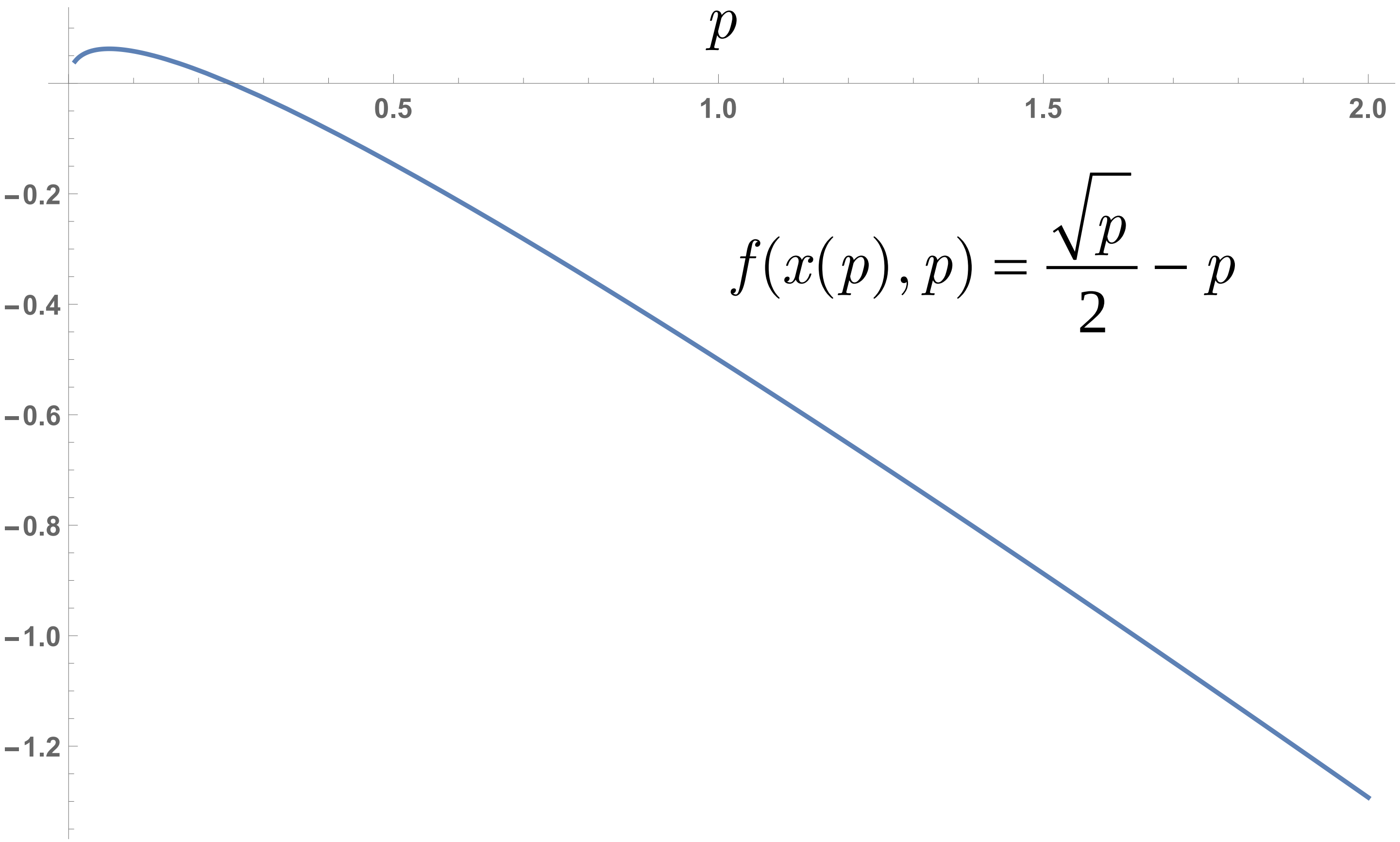


Motivation: Reduced-Space Optimization

$$z^2 - p = 0$$

$$\Rightarrow x(p) = \sqrt{p}$$

$$\min_p \frac{x(p)}{2} - p$$
$$\text{s.t.} \;\; p \in [0.01, 2]$$



Robust Dynamic Optimization

Robust Operation SIP

$$\gamma^* = \max_{p \in P, \gamma \in \Gamma} \gamma$$
$$\text{s.t. } \gamma \le T(t_f, \mathbf{u}) - p, \ \forall \mathbf{u} \in U$$

Dynamical System (Mass & Energy Balance)

$$\frac{dC_m}{dt} = (1 + \epsilon C_m / C_{m_0}) R_m,$$
$$\frac{dC_i}{dt} = R_i + \epsilon C_i / C_{m_0} R_m,$$
$$\frac{dT}{dt} = \frac{\alpha_0 k_P \xi_0 C_m}{1 + \epsilon C_m / C_{m_0}} + \alpha_1 (T_j - T)$$

Relaxations of Dynamic System

Rate constants (R_m, R_i) from pseudo-empirical models

Rate Expression *(Greatly Simplified….)*

$$R_m = -C_m \xi_0 (k_P + k_{fm}),$$
$$R_i = -k_i C_i,$$
$$\boxed{\xi_0^2 k_t(\xi_0, C_m, T) - 2\zeta k_i C_i = 0.}$$

Use 3-layer GeLU ANN as in place of solving nonlinear equation from quasi-steady state assumption

Able to solve SIP in 57.8 s using a modified SIPres algorithm.

EAGO.jl: Core Additions

New multi-graph representation:

- Introduce support for multiple-output subexpressions.
- Separate caches of information from graph structure.
- Preliminary implementations for graph-walking convexity detection.

Currently, under development. Expected by end of August 2021.

Roadmap to Improved Composability of Nonlinear Functions (via JuMP extension):

- Introduce support for multiple-output subexpressions.
- Introduce auxiliary variables.
- Facilitates chaining blocks of general nonlinear expressions (such as implicitly defined functions).

$$\mathbf{x} \xrightarrow{\mathbf{h}_1(\mathbf{y},\mathbf{x}) = \mathbf{0}} \mathbf{y} \xrightarrow{\mathbf{h}_2(\mathbf{z},\mathbf{y}) = \mathbf{0}} \mathbf{z} \xrightarrow{\mathbf{f}(\mathbf{z})} \mathbf{f(x)}$$