Robust Simulation of Hybrid Mechanistic and Machine Learning Models

October 15, 2020

Matthew D. Stuber, PhD
Assistant Professor, University of Connecticut
Outline

1. Introduction
2. Robust Simulation of Mechanistic Models
3. Concepts of Hybrid Models
4. Robust Simulation of Hybrid Models
5. Conclusion
Robust Simulation

• A “Robust System” mitigates the effects of uncertainty to ensure performance/safety constraints are satisfied.
A “Robust System” mitigates the effects of uncertainty to ensure performance/safety constraints are satisfied.

“Robust Simulation” refers to the ability to rigorously account for the impacts of uncertainty via a model-based (i.e., simulation) approach

– Conclude whether or not a system can meet the desired performance/safety constraints in the face of uncertainty using mathematical models
Robust Simulation: Another Perspective

• “Robust Simulation” could also be viewed through the modeler’s lens
“Robust Simulation” could also be viewed through the modeler’s lens

- Modeling and simulation of systems often requires changing parameter values and/or model libraries and the solver then fails to converge
• “Robust Simulation” could also be viewed through the modeler’s lens
  – Modeling and simulation of systems often requires changing parameter values and/or model libraries and the solver then fails to converge
• “Robust Simulation” could also be viewed through the modeler’s lens
  – Modeling and simulation of systems often requires changing parameter values and/or model libraries and the solver then fails to converge.

```julia
julia> acos(-1.1)
ERROR: DomainError with -1.1:
acos(x) not defined for |x| > 1
Stacktrace:
[1] acos_domain_error(::Float64) at `<module>`:671
[2] acos(::Float64) at `<module>`:701
[3] top-level scope at REPL[92]:1
[4] include_string(::Function, ::Module, ::String, ::String) at `<module>`:1088
```

Domain violation (numerical issue)
Constraint violation (safety issue)
Robust Simulation: Another Perspective

- Numerical infeasibility encountered in algebraic systems

DOI: 10.1002/aic.14447
Robust Simulation of Mechanistic Models
Accounting for Uncertainty

For a specific design, how would the system respond to uncertainty?

System Model: $h(z, u, p) = 0$

Parametric Uncertainty: $p \in P$

Design: $u \in U$
Accounting for Uncertainty

For a specific design, how would the system respond to uncertainty?

System Model

$$h(z, u, p) = 0$$

Parametric Uncertainty

$$p \in P$$

Design

$$u \in U$$

Operating Envelope

State-Space
Accounting for Uncertainty

For a specific design, how would the system respond to uncertainty?

System Model: \( h(z, u, p) = 0 \)

Parametric Uncertainty: \( p \in P \)

Design: \( u \in U \)

Constraint/Specification:

Operating Envelope:

State-Space: \( z \)
Accounting for Uncertainty

For a specific design, how would the system respond to uncertainty?
Accounting for Uncertainty

For a specific design, how would the system respond to uncertainty?

System Model: \( h(z, u, p) = 0 \)

Parametric Uncertainty: \( p \in P \)

Design: \( u \in U \)

Constraint/Specification

Operating Envelope

State-Space

SYSTEM FAILURE!
Accounting for Uncertainty

For a specific design, how would the system respond to uncertainty?

System Model

\[ h(z, u, p) = 0 \]

Parametric Uncertainty
\[ p \in P \]

Design
\[ u \in U \]

Constraint/Specification

Operating Envelope

ROBUST SYSTEM!
Research Challenge:
Verifying a system is not robust is as simple as finding a single realization of uncertainty that violates the constraint.

Verifying a system is robust requires simulating infinitely-many realizations of uncertainty and ensuring the system never violates the constraint.
Accounting for Uncertainty

- Steady-state vs. dynamical systems models

\[ h(z, u, p) = 0 \]
nonlinear algebraic system

\[ \dot{x}(u, p, t) = f(x(u, p, t), u, p, t) \]
nonlinear ODE system
Accounting for Uncertainty

- Steady-state vs. dynamical systems models

\[ h(z, u, p) = 0 \]

\[ \dot{x}(u, p, t) = f(x(u, p, t), u, p, t) \]

Now, we must account for the transient response to uncertainty in our design.
Accounting for Uncertainty

Dynamic System Model

\[ g(x(u, p, t_k), u, p, t_k) \leq 0 \]

Parametric Uncertainty
\[ p \in P \]

Design
\[ u \in U \]

Constraint/Specification

State-Space
\[ x(u, p, t_k) \]

Operating Envelope

Robust Simulation of Mechanistic Models
Accounting for Uncertainty

Dynamic System Model

\[
g(x(u, p, t_{k+1}), u, p, t_{k+1}) \leq 0
\]

Parametric Uncertainty
\[ p \in P \]

Design
\[ u \in U \]

Constraint/Specification
SYSTEM FAILURE!

State-Space
\[ x(u, p, t_{k+1}) \]
Mathematical Preliminaries

• From a design perspective, our objective is to verify performance/safety in the face of (the worst-case) uncertainty over the time horizon.

\[ \gamma(u) = \max_{p \in P, t \in T} g(x(u, p, t), u, p, t) \]

s.t. \[ \dot{x}(u, p, t) = f(x(u, p, t), u, p, t) \]

\[ x(u, p, 0) = x_0(u, p) \]
Mathematical Preliminaries

• From a design perspective, our objective is to verify performance/safety in the face of (the worst-case) uncertainty over the time horizon.

\[
\gamma(u) = \max_{p \in P, t \in T} g(x(u, p, t), u, p, t)
\]

s.t. \[
\dot{x}(u, p, t) = f(x(u, p, t), u, p, t)
\]

\[
x(u, p, 0) = x_0(u, p)
\]

If \(\gamma(u) \leq 0\), we have verified the robustness of our design \(u\).

“For a given design, the system does not violate performance/safety at any point in time, even in the face of the worst-case uncertainty”
Robust Steady-State Simulation

- Previous developments: a set-valued mapping theory that enables the calculation of rigorous bounds on the states over the entire uncertainty space.

Stuber, M.D. et al. (2015) DOI: 10.1080/10556788.2014.924514

Robust Simulation of Mechanistic Models
Robust Steady-State Simulation

- Convex relaxation of nonconvex operating envelope (without actually simulating the operating envelope)

State-Space

Robust Simulation of Mechanistic Models

Stuber, M.D. et al. (2015) DOI: 10.1080/10556788.2014.924514
Robust Dynamic Simulation

• Our dynamic model is reformulated in the discrete form as a nonlinear algebraic system:

\[
\mathbf{h}(\mathbf{y}, \mathbf{u}, \mathbf{p}) = \begin{bmatrix}
\mathbf{y}_0 - \mathbf{x}_0(\mathbf{u}, \mathbf{p}) \\
\mathbf{y}_1 - \mathbf{y}_0 - h\mathbf{f}(\mathbf{y}_1, \mathbf{u}, \mathbf{p}, t_1) \\
\vdots \\
\mathbf{y}_K - \mathbf{y}_{K-1} - h\mathbf{f}(\mathbf{y}_K, \mathbf{u}, \mathbf{p}, t_K)
\end{bmatrix} = \mathbf{0}
\]

where \( h : \mathbb{R}^{n_y(K+1)} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_y(K+1)} \).
Apply our theory for robust dynamic simulation to our system to calculate rigorous bounds on the state variables over the range of uncertainty variables $p$ and design variables $u$, forward in time.
Robust Dynamic Simulation

Robust Simulation of Mechanistic Models
Global optimization of stiff dynamical systems

Matthew E. Wilhelm | Anne V. Le | Matthew D. Stuber

Abstract
We present a deterministic global optimization method for nonlinear programming formulations constrained by stiff systems of ordinary differential equation (ODE) initial value problems (IVPs). The examples arise from dynamic optimization problems exhibiting both fast and slow transient phenomena commonly encountered in model-based systems engineering applications. The proposed approach utilizes unconditionally stable implicit integration methods to reformulate the ODE-constrained problem into a nonconvex nonlinear program (NLP) with implicit functions embedded. This problem is then solved to global optimality in finite time using a spatial branch-and-bound framework utilizing convex/concave relaxations of implicit functions constructed by a method which fully exploits problem sparsity. The algorithms were implemented in the Julia programming language within the EAGO package and demonstrated on five illustrative examples with varying complexity relevant in process systems engineering. The developed methods enable the guaranteed global solution of dynamic optimization problems with stiff ODE-IVPs embedded.

KEYWORDS:
dynamic simulation, global optimization, implicit functions, stiff systems

1 | INTRODUCTION
Dynamic optimization problems of the form
\[ \begin{align*}
\min_{x(n)} & \quad f(x(n),p) \\
\text{subject to} & \quad g(x(n),p) = 0, \quad h(x(n),p) \leq 0
\end{align*} \] (1)
are of extreme importance to process systems engineers and the broader model-based systems engineering community as they can be formulated for a variety of systems whose transient behavior is of particular interest, from optimal control to mechanistic model validation.

The first major complicating detail of the optimization formulation (1) is that it is constrained by a system of ordinary differential equation-initial value problems (ODE-IVPs). Therefore, simply verifying a feasible point requires the solution of a system of ODE-IVPs. The second major complicating detail that (1) is a nonconvex program, in general, and therefore verifying optimality requires deterministic global optimization. The focus of this paper is on solving (1) to guaranteed global optimality (or declaration of infeasibility).

The methods developed in this work are of specific importance when the ODE-IVP system is stiff. Methods for solving (1) rigorously to global optimality rely on the spatial branch-and-bound (SBB) framework\(^{[2,3]}\) or some variant. The SBB algorithm requires the ability to calculate rigorous upper and lower bounds on the global optimal solution value. An open bound can be calculated by simply evaluating \(f(x^k,\cdot)\), at any feasible point.

However, calculating rigorous lower bounds poses significant challenges as this step requires that rigorous and accurate global bounds are known or are readily calculated for all variables and functions of (1). For general nonlinear systems programs (NLPs, i.e., without dynamical systems constraints), rigorous lower bounds on the optimal solution value are obtained by calculating convex and concave relaxations of the functions and solving a corresponding convex lower bounding problem. Applying this approach to a dynamic optimization problem...
Control of a 9-species biological reaction for wastewater treatment.
Robust Dynamic Simulation

Next steps: verification and validation of robustness

\[
\begin{align*}
\min_{\gamma} \gamma \\
\text{s.t. } \gamma \geq \max_{p \in P, t \in I} g(x(u, p, t), u, p, t) \\
\text{s.t. } \dot{x}(u, p, t) &= f(x(u, p, t), u, p, t), t \in I \\
x(u, p, 0) &= x_0(u, p)
\end{align*}
\]

Robust Design
Robust Dynamic Simulation

Next steps: verification and validation of robustness

\[
\begin{align*}
\min_{u \in U} \gamma \\
\text{s.t. } \gamma \geq \max_{p \in P, t \in I} g(x(u, p, t), u, p, t) \\
\text{s.t. } \dot{x}(u, p, t) = f(x(u, p, t), u, p, t), t \in I \\
\text{s.t. } x(u, p, 0) = x_0(u, p)
\end{align*}
\]

\[
\begin{align*}
\max_{p \in P} \gamma \\
\text{s.t. } \gamma \geq \min_{u \in U} g(x(u, p, t_f), u, p, t_f) \\
\text{s.t. } \dot{x}(u, p, t) = f(x(u, p, t), u, p, t), t \in I \\
\text{s.t. } x(u, p, 0) = x_0(u, p)
\end{align*}
\]

Robust Design

Robust Operation
Next steps: verification and validation of robustness

Find the best design $u$ and seek the worst-case realization of uncertainty $p$ to see if the system violates the performance/safety specifications.

Robust Design

Robust Operation

$$\max_{p \in P} \gamma$$

s.t. $\gamma \geq \min_{u \in U} g(x(u, p, t_f), u, p, t_f)$

s.t. $\dot{x}(u, p, t) = f(x(u, p, t), u, p, t), t \in I$

$x(u, p, 0) = x_0(u, p)$
Robust Dynamic Simulation

Next steps: verification and validation of robustness

Find the best design $u$ and seek the worst-case realization of uncertainty $p$ to see if the system violates the performance/safety specifications.

Find the worst-case realization of uncertainty $p$ and seek a recourse (control) $u$ and to see if the system violates the performance/safety specifications.

Robust Design

Robust Operation
Robust Dynamic Simulation

Next steps: verification and validation of robustness

\[
\begin{align*}
\min_{u \in U} & \quad \gamma \\
\text{s.t.} & \quad \gamma \geq \max_{p \in P, t \in I} g(x(u, p, t), u, p, t) \\
\text{s.t.} & \quad \dot{x}(u, p, t) = f(x(u, p, t), u, p, t), t \in I \\
& \quad x(u, p, 0) = x_0(u, p)
\end{align*}
\]

\[
\begin{align*}
\max_{p \in P} & \quad \gamma \\
\text{s.t.} & \quad \gamma \geq \min_{u \in U} g(x(u, p, t_f), u, p, t_f) \\
\text{s.t.} & \quad \dot{x}(u, p, t) = f(x(u, p, t), u, p, t), t \in I \\
& \quad x(u, p, 0) = x_0(u, p)
\end{align*}
\]

Robust Design

Robust Operation
Software Tools

- **EAGO.jl**: Easy Advanced Global Optimization in Julia.
  - Open-source, competitive with state-of-the-art commercial solvers but much more flexible to account for complicated user-defined functions (UDFs)

![Graph showing comparison of different solvers](image)

DOI: [10.1080/10556788.2020.1786566](https://doi.org/10.1080/10556788.2020.1786566)
Software Tools

- EAGO.jl: Easy Advanced Global Optimization in Julia.
  - Open-source, competitive with state-of-the-art commercial solvers but much more flexible to account for complicated user-defined functions.
Software Tools

- **EAGO.jl**: Easy Advanced Global Optimization in Julia.
  Software access: registered Julia package

  Github: [https://github.com/PSORLab/EAGO.jl](https://github.com/PSORLab/EAGO.jl)
Software Tools

• EAGO.jl: Easy Advanced Global Optimization in Julia.

DOI: 10.1080/10556788.2020.1786566
• Consider the anharmonic oscillator (e.g., pendulum)

\[
\ddot{x}(t) = -kx(t) - \alpha x(t)^3 - \beta \dot{x}(t) - \gamma \dot{x}(t)^3
\]

\[
x(0) = 1, \dot{x}(0) = 0
\]
Dynamic NN Prediction Errors

Tensor product layer with 10th-Order Legendre Basis

(Example from DiffEqFlux.jl)
Dynamic NN Prediction Errors

Anharmonic Oscillator Training Region

Anharmonic Oscillator Prediction Region

Concepts of Hybrid Models
Dynamic NN Prediction Errors

Anharmonic Oscillator Prediction Region

-0.5

0.0

0.5

1.0

0 10 20 30 40

time

X (ODE)
V (ODE)
X (NN)
V (NN)
Dynamic NN Prediction Errors

Anharmonic Oscillator Prediction Region

Actual trajectories exhibit rapid decay
Dynamic NN Prediction Errors

Anharmonic Oscillator Prediction Region

- Actual trajectories exhibit rapid decay
- ML model fails to capture this behavior
Dynamic NN Prediction Errors

Anharmonic Oscillator Prediction Region

Actual trajectories exhibit rapid decay

ML model fails to capture this behavior
Dynamic NN Prediction Errors

Anharmonic Oscillator Prediction Region

- X (ODE)
- V (ODE)
- X (NN)
- V (NN)

Concepts of Hybrid Models
Outstanding ML Challenges

• Absence of theory
• Absence of causal models (correlation not causation)
• Sensitivity to imperfect data
• Computational expense (training)

Outstanding ML Challenges (SE Perspective)

- Lack requirements specification
- Lack design specification
- Lack interpretability (causal relationships)
- Lack robustness

Kuwajima, H., Yasuoka, H., Nakae, T., (2020) DOI: 10.1007/s10994-020-05872-w
Outstanding ML Challenges (SE Perspective)

- Lack requirements specification
- Lack design specification
- Lack interpretability
- Lack robustness

“Greatest impact on conventional system quality models”

Kuwajima, H., Yasuoka, H., Nakae, T., (2020) DOI: 10.1007/s10994-020-05872-w
Outstanding ML Challenges (SE Perspective)

- Lack requirements specification
- Lack design specification
- Lack interpretability
- Lack robustness

Research Challenge:
Can we exploit machine learning approaches for safety-critical systems?

Kuwajima, H., Yasuoka, H., Nakae, T., (2020) DOI: 10.1007/s10994-020-05872-w
Hybrid Mechanistic ML Models

• Not a “new” idea (emergence in 1992)
• Combine aspects of machine learning and mechanistic modeling
• Black-Box → Gray-Box
Hybrid Mechanistic ML Models

Parallel

First-Principles Model

ML Model

Series

ML Model

First-Principles Model

ML Model

First-Principles Model
Hybrid Mechanistic ML Models

Concepts of Hybrid Models
Hybrid Mechanistic ML Models

Machine Learning Model
- Empirical model
- Artificial neural net
- Dyn. neural net
- Support vect. machine
Hybrid Mechanistic ML Models

\[ y_{k+1} \]

\[ z_k \]

\[ y_k, u, p \]

ANN

Mechanistic Model
Hybrid Mechanistic ML Models

Concepts of Hybrid Models

“Intermediate” states or calculations. E.g., Complex nonlinear dynamics.
Hybrid Mechanistic ML Models

Benefits over pure data-driven models:
• Requires less data
• Have system insight
• Better controller performance
• Better performance with nonlinear dynamics
• Better performance in extrapolation
  – More useful for optimization applications
Robust Simulation of Hybrid Models
Physics-Informed Data-Driven Models

• Want to control wastewater treatment processes to optimize energy consumption and meet discharge requirements.
Physics-Informed Data-Driven Models

- Developed a compartment model with unknown parameters for mass transfer between compartments
- Applied deterministic global optimization for training to obtain guaranteed best-possible fit
Physics-Informed Data-Driven Models

Percent error relative to physics-informed data-driven model

<table>
<thead>
<tr>
<th>Experiment</th>
<th>CFD</th>
<th>Pure DD (ML) Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment 1</td>
<td>108%</td>
<td>329%</td>
</tr>
<tr>
<td>Experiment 2</td>
<td>1287%</td>
<td>588%</td>
</tr>
<tr>
<td>Experiment 3</td>
<td>511%</td>
<td>319%</td>
</tr>
</tbody>
</table>
Hybrid Mechanistic ML Models

System Model

\[ h(y, u, p) = 0 \]

Parametric Uncertainty
\[ p \in P \]

Design
\[ u \in U \]

Robust Simulation of Hybrid Models

Operating Envelope

State-Space

\[ y_{k+1} = (, , ) \]

\[ h_{k} = (, , ) \]

\[ z_{k} = (, , ) \]
Robust Simulation of Hybrid Models
Robust Simulation of Hybrid Models
Robust Simulation of Hybrid Models

• Use our set-valued bounding theory to rigorously bound the states
Relaxations of Activation Functions

\[
\log(1 + \exp(x)) \quad \max(x, \frac{1}{1 + \exp(-x)}) \quad \max(x, \tanh(x))
\]

Robust Simulation of Hybrid Models
Relaxations of Activation Functions

\[
\frac{x}{1 + |x|}
\]

\[
\frac{1}{1 + \exp(-x)}
\]

\[
1 - \frac{2}{1 + \exp(-x)}
\]
Conclusion

• Developed rigorous bounding theory for steady-state and dynamical systems for mechanistic models
  – Formal uncertainty quantification
  – Extremely powerful open-source deterministic global optimizer for advanced user-defined models

• Want to exploit hybrid modeling approaches to overcome challenges with pure mechanistic and pure data-driven approaches
Conclusion

• Applied global optimization for training a physics-informed data-driven model to demonstrate the tradeoff

• Preliminary work on bounding a library of common basis functions for NN
  – Enable rigorously bounding hybrid models
  – Formal uncertainty quantification of hybrid models
This material is based upon work supported by the National Science Foundation under Grant No.: 1706343, 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.