Co-simulation

Cláudio Gomes
Complexity meets Markets

- Increasing Complexity
  - Interacting heterogeneous components
- Competitive Market
- Concurrent Development
  - Late Integration Problems
  - Contracts
- Independently Developed Sub-systems
  - Specialized Teams
  - External Suppliers
  - IP Protection
- “Holy Grail”:
  - Integration at every stage of development
M&S is not Enough

• Domain Specific Tools/Solvers
  • E.g. discrete event solvers, symplectic numerical solvers;
  • Difficult to build one big model

• Suppliers want to protect IP
  • No detailed models provided
  • Catalogs may lack necessary information (e.g., op. temperature range, wear&tear)
  • Hard to get models

• Black box inductive models
  • Lookup tables,
  • Fitted experimental data.
  • There are no models!

Pedersen, N., Lausdahl, et. al (2017). Distributed Co-Simulation of Embedded Control Software with Exhaust Gas Recirculation Water Handling System using INTO-CPS. In 7th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (pp. 73–82).
Co-simulation

A technique to combine simulators.

Orchestration

Orchestrator

\[ t := t + H \]

\[ \text{getOutput(...) \hspace{1cm} setInput(...)} \]

\[ \text{getOutput}(...) \hspace{1cm} \text{setInput}(...) \]

\[ \text{simulateUntil}(t+H,\ldots) \]

\[ \text{simulateUntil}(t+H,\ldots) \]

\[ t := t + H \]

\[ \ldots \]
Internal Behavior

Orchestrator

\[ S_1 \xrightarrow{F_c} S_2 \]

- `getOutput(...)`
- `setInput(...)`
- `getOutput(...)`
- `setInput(...)`
- `simulateUntil(t+H,...)`
- `simulateUntil(t+H,...)`
- \( t := t + H \)

\[ t \]

\[ t+H \]

\[ t+h_1 \]

\[ t+h_2 \]
The Context

Historical Overview

Casper Thule, David Broman, Peter Gorm Larsen, Hans Vangheluwe


The Context

First discrete event synchronization algorithms

Distributed Simulation Using a Network of Processors

J. Kent Peacock, J.W. Wong, and Eric G. Manning
Department of Computer Science and Engineering, University of Waterloo, Waterloo, Ontario, Canada

Simulation, particularly of networks of queues, is an application with a high degree of inherent parallelism and is of great practical interest. We hope to exploit this parallelism by performing simulations using a network of cheap, but slow, microprocessors.

We define a taxonomy by which we classify a number of parameters of a distributed simulation method, and seek solutions to problems of synchronization, deadlock prevention, and inter-process communication arising with various methods in each class of the taxonomy. We concentrate in particular on the class (loose, event-driven) which seems to possess the greatest potential parallelism, and on the class (synchronized, time-driven) which allows mock-up studies of real systems. We give algorithms for deadlock prevention and detection, and briefly discuss the design of a distributed operating system to support the simulation application.

This research forms part of a joint program with the Dept. of Computer Science, University of Texas at Austin, directed by Prof. K.M. Chandy.

Keywords: Distributed processing, simulation, distributed algorithms, microprocessors, synchronization, deadlock.


The Context

First discrete event synchronization algorithms

Decomposition of sparsely coupled systems.\(^1\)


The Context

First discrete event synchronization algorithms

Decomposition of sparsely coupled systems.

Optimistic Time warp O.S.

The Context

First discrete event synchronization algorithms

- Decomposition of sparsely coupled systems.
- Optimistic Time warp O.S.
- SIMNET (Real-time Military Training)

SIMNET: The Advent of Simulator Networking

DUNCAN C. MILLER, MEMBER, IEEE, AND JACK A. THORPE

Invited Paper

SIMNET was the first successful implementation of large-scale, real-time, man-in-the-loop simulator networking for team training and mission rehearsal in military operations. This paper provides some historical background on how SIMNET was developed within the US Department of Defense, and outlines the key philosophical and architectural principles on which it was based.

The SIMNET battlefield simulation was sponsored by ARPA (then called DARPA), in partnership with the US Army, and was developed and implemented between 1983 and 1990. The emphasis of SIMNET from the outset was on enhancing tactical team performance by providing commanders and troops an opportunity to practice their skills in a dynamic, free-play environment, in which battle outcomes depend on team coordination and individual initiative, rather than on scripted scenarios controlled by an instructor. This program demonstrated the feasibility of linking together hundreds or thousands of simulators (representing tanks, infantry fighting vehicles, helicopters, fixed-wing aircraft, etc.) to create a consistent, virtual world in which all participants experience a coherent, logical sequence of events. In this world, the causal connections among these tactical events, from the individual crew station to the battalion command post, are clear and easily inspectable.

The SIMNET architecture and protocols have evolved into the Distributed Interactive Simulation (DIS) Standard Protocols (IEEE 1278-1993 and its successors), and have provided the foundation for a new generation of battlefield simulations for training, mission rehearsal, tactics development, evaluation of hypothetical new battlefield systems, and command and control.

The Context
The Context

SIMNET (Real-time Military Training) SW/HW co-simulation.

“Software developers were often left to develop code for months with severely limited ability to test [...]. Painful integration efforts came late in the design cycle, and minor miscommunication became major design flaws.”

-- Anne Powell and Shawn Lin

A DSblock may serve as neutral ‘model bus’ to exchange nonlinear models between different modelling and simulation environments.

The Context

SIMNET (Military Training)
SW/HW co-simulation.
DSBlock Standard
MPM&S Research agenda


Simulation for the Future: Progress of the Esprit Basic Research Working Group 8467

Hans L. Vangheluwe, Ghislain C. Vansteenkiste
Applied Mathematics, Biometrics and Process Control
University of Gent
Coupure 653
B-9000 Gent
Belgium

tel: +32 (9) 264 59 37
fax: +32 (9) 223 49 41

Eugène J.H. Kerckhoffs *
Applied Mathematics and Informatics
Delft University of Technology
Julianalaan 132
NL-2628 BL Delft
The Netherlands

Abstract

The Esprit Basic Research Working Group 8467 aims at stimulating the development of simulation for the Future: new concepts, tools and applications”, aiming at a more coherent and unified approach than currently used and exploit this as a basis for transferring simulation technology to real-life applications.

“A communication protocol which defines interaction between simulators in a tool and vendor independent manner.”
The Context

SIMNET (Military Training)
SW/HW co-simulation.
DSBlock Standard
MPM&S Research agenda
- DIS IEEE Standard

The Context

SIMNET (Military Training) SW/HW co-simulation.

DSBlock Standard

MPM&S Research agenda

- DIS IEEE Standard Multi-abstraction

The Context

- SIMNET (Military Training)
- SW/HW co-simulation
- DSBlock Standard
- MPM&S Research agenda
  - DIS IEEE Standard
  - Multi-abstraction
  - Automotive, Railway, HVAC reported applications


The Context
The Context

Automotive, Railway, HVAC reported applications

HLA IEEE Standard

The Context

Automotive, Railway, HVAC reported applications

HLA IEEE Standard

DMU Concept

The Context

- Automotive, Railway, HVAC reported applications
- HLA IEEE Standard
- DMU Concept
  - IP Protection, FMI Standard

The Context

- Automotive, Railway, HVAC reported applications
- HLA IEEE Standard
- DMU Concept
  - IP Protection, FMI Standard
- Digital Twin

The Context

Automotive, Railway, HVAC reported applications

HLA IEEE Standard

DMU Concept

IP Protection, FMI Standard

Digital Twin

Virtual Design, Build, Operation, and Maintenance

2000  2010
The Vision

Virtual Design, Build, Operation, and Maintenance
Virtual Design

[2] Case Study by Kristof Berx, Davy Maes, and Klaas Gadeyne, from Flanders Make
Virtual Design

Frequent full system evaluation

...at multiple levels of refinement
Virtual Build

Virtual Build

Frequent full system evaluation
...at any level of refinement

...between multiple suppliers
Virtual Operation

Virtual Operation

Evaluation between multiple suppliers

...with X-in-the-loop

Frequent full system evaluation...at any level of refinement

---

[2] Case Study by Kristof Berx, Davy Maes, and Klaas Gadeyne, from Flanders Make
Virtual Maintenance

[2] https://www.youtube.com/watch?v=2dCz3oL2rTw
The Baby Steps: Semantic adaptation

Our contributions

Bart Meyers, Joachim Denil, Casper Thule, Kenneth Lausdahl
Peter Gorm Larsen, Hans Vangheluwe, Paul De Meulenaere

Functional Mock-up Interface Standard

- Standard interaction with any simulator
- A Functional Mockup Unit is a zip-file (.fmu) consisting of
  - C Library (.dll or .so)
  - XML file (metadata)
- Every simulator is a black box.
  - Executed locally but can communicate with a remote server
- The coupling (master algorithm) must be provided
FMU (Conceptual) Internals

\( S_1 \)

Model
\[ \dot{x}_1 = F_1(x_1, u_1) \]
\[ y_1 = G_1(x_1, u_1) \]

Solver
\[ x(t + h) = x(t) + F(x(t), u(t)) \times h \]

Input Approximation
\[ u(t) = \phi_u(t, u(n \cdot H), u((n - 1) \cdot H), \ldots) \]

\( S_2 \)

Orchestrator

[diagram showing the flow of information between models]
FMU Example

```c
fmi2Status fmi2GetReal(fmi2Component fc, 
{ 
    const fmi2ValueReference vr[], size_t nvr, fmi2Real value[]
} 
FMUInstance* comp = (FMUInstance *)tc;
int i;
for (i = 0; i < nvr; i++)
{ 
    value[i] = comp->r([vr[i]]);
}
return fmi2OK;
}

fmi2Status fmi2DoStep(fmi2Component fc, fmi2Real currentCommPoint, fmi2Real commStepSize, fmi2Boolean noPrevFMUState)
{ 
    FMUInstance* fi = (FMUInstance *)fc;
    fmi2Status simStatus = fmi2OK;
    printf("%s in fmiDoStep()\n", fi->instanceName);
    fi->currentTime = currentCommPoint + commStepSize;
    printf("Motor_in: %f\n", fi->r[_motor_in]);
    printf("slave CBD_PART2 now at time: %f\n", fi->currentTime);

    fi->r[_position] = fi->r[_position] + fi->r[_velocity] * commStepSize;
    fi->r[_velocity] = fi->r[_velocity] + fi->r[_acceleration_after_friction] * commStepSize;
    fi->r[_friction] = fi->r[_velocity] * 5.81;
    fi->r[_motor_acceleration] = fi->r[_motor_in] * 40;
    fi->r[_acceleration_after_friction] = fi->r[_motor_acceleration] - fi->r[_friction];

    return simStatus;
}
```
Motivation for Semantic Adaptation

• Quick and sound way of adapting the behaviour of an interconnected set of FMUs
  • Unit conversion
  • Interaction protocol modification
    • Time triggered vs Event triggered execution
  • Affect accuracy/performance
Semantic Adaptation

• Actions by which the behavior of an original set of interconnected FMUs is altered, following the transparency and modularity principles.

How?
Semantic Adaptation

• Actions by which the behavior of an original set of interconnected FMUs is altered, following the transparency and modularity principles.

How?

![Diagram showing the process of semantic adaptation](image-url)
A DSL for Semantic Adaptation

```plaintext
semantic adaptation reactive moore ControllerSA controller_sa
at "./path/to/ControllerSA.fmu"

for inner fsm Controller ctrl
at "./path/to/LazySA.fmu"
with input ports obj_detected, passenger_up, passenger_down
with output ports up, down, stop

input ports
  armature_current -> ctrl.obj_detected,
  passenger_up -> ctrl.passenger_up,
  passenger_down -> ctrl.passenger_down,
  passenger_stop -> ctrl.passenger_stop,
  driver_up -> ctrl.driver_up,
  driver_down -> ctrl.driver_down,
  driver_stop -> ctrl.driver_stop

output ports u, d
```
A DSL for Semantic Adaptation

```plaintext
control rules {
  var step_size := H;
  var aux_obj_detected := false;
  var crossedTooFar := false;
  if ((not is_close(p_v, T, RTOL, ATOL) and p_v < T)
      and (not is_close(f_v, T, RTOL, ATOL) and f_v > T)) {
    crossedTooFar := true;
    var negative_value := p_v - T;
    var positive_value := f_v - T;
    step_size := (H * (-negative_value)) / (positive_value - negative_value);
  } else {
    if ((not is_close(p_v, T, RTOL, ATOL) and p_v < T)
        and is_close(f_v, T, RTOL, ATOL)) {
      c := true;
    }
  }

  if (not crossedTooFar){
    step_size := do_step(ctrl, t, H);
  }

  if (is_close(step_size, H, RTOL, ATOL)) {
    p_v := f_v;
  }

  return step_size;
}
A DSL for Semantic Adaptation

```plaintext
in var f_v := INIT_V;
in rules {
  true -> {
    f_v := controller_sa.armature_current;
  } --> {
    ctrl.obj_detected := c;
  };
}

out rules {
  ctrl.up -> { } --> {controller_sa.u := 1.0; };
  not ctrl.up -> { } --> {controller_sa.u := 0.0; };
  ctrl.down -> { } --> {controller_sa.d := 1.0; };
  not ctrl.down -> { } --> {controller_sa.d := 0.0; };
  ctrl.stop -> { } --> {controller_sa.u := 0.0 ; controller_sa.d := 0.0; };
}
The Baby Steps: Stability of Adaptive Co-simulation

Our contributions

Benoît Legat, Raphael Jungers, Hans Vangheluwe

Running Example – Original System

Running Example – Co-simulation
Orchestration

\[
\begin{align*}
&\text{Orchestrator} \\
&S_1 \xrightarrow{F_c} S_2 \\
&t := t + H
\end{align*}
\]
Internal Behavior

Orchestrator

\[
t := t + H
\]

\[
\begin{align*}
&\text{getOutput(...)} \\
&\text{setInput(...)} \\
&\text{getOutput(...)} \\
&\text{setInput(...)} \\
&\text{simulateUntil}(t+H,...) \\
&\text{simulateUntil}(t+H,...) \\
&\text{...}
\end{align*}
\]
Simulator Internals

\[ S_1 \]

Model
\[ \dot{x}_1 = F_1(x_1, u_1) \]
\[ y_1 = G_1(x_1, u_1) \]

Solver
\[ x(t + h) = x(t) + F(x(t), u(t)) \times h \]

Input Approximation
\[ u(t) = \phi_u(t, u(n \cdot H), u((n - 1) \cdot H), \ldots) \]

\[ S_2 \]

Orchestrator

\[ S_1 \overset{[x_1, v_1]}{\longrightarrow} S_2 \]

\[ F_c \]

\[ x_1, v_1 \]

\[ x_1, v_1 \]
Orchestration Space – Inputs


Input Approximations

Extrapolation

Interpolation

Polynomial °0

Polynomial °1

Context-aware

Model ID’ed

C^0 Continuous

C^1 Continuous

...
Orchestration Space – Solvers

Numerical Solvers

<table>
<thead>
<tr>
<th>Parallel</th>
<th>Order 0</th>
<th>Implicit</th>
<th>Semi-Explicit</th>
<th>Explicit</th>
<th>Step size</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential</td>
<td>Order 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Solver
\[
x(t + h) = x(t) + F(x(t), u(t)) \times h
\]

Input Approximation
\[
u(t) = \phi_u(t, u(n \cdot H), u((n - 1) \cdot H), \ldots)
\]

Orchestration Space – Synchronization


Orchestration Algorithms

Parallel
Sequential

Jacobi
Gauss-Seidel
...

Implicit
Semi-Explicit
Explicit

Step size
...

Capability Interaction

Orchestration Algorithms

Parallel
Sequential

Jacobi
Gauss-Seidel
...

Implicit
Semi-Explicit
Explicit

Step size
...

Input Approximations

Extrapolation
Interpolation

Polynomial °0
Polynomial °1
...

C° Continuous
...

...
Adaptive Orchestration

- Forward Euler: $h=0.04$
- Midpoint: $h=0.01$
- Constant Extrapolation

Model

$S_1$
- Forward Euler: $h=0.04$
- Midpoint: $h=0.01$
- Constant Extrapolation

$S_2$
- Forward Euler: $h=0.2$
- Midpoint: $h=0.1$
- Constant Extrapolation

Jacobi: $H=0.2$
$H=0.1$
Adaptive Orchestration

Research problem: Is this policy stable?
Non-adaptive Stability Analysis

Model Forward Euler: $h=0.04$

$S_1$

Constant Extrapolation

Model Forward Euler: $h=0.2$

$S_2$

Constant Extrapolation

Jacobi: $H=0.2$

Stability definition

\[ x^{(n+1)} = Ax^{(n)} \]

Non-adaptive:

\[ \forall x^{(0)} \lim_{k \to \infty} \left\| A^k x^{(0)} \right\| = 0 \]

Adaptive:

\[ x^{(n+1)} = \begin{cases} 
A_1 x^{(n)} \\
\text{or } A_2 x^{(n)} \\
\vdots \\
\text{or } A_N x^{(n)} 
\end{cases} \]

\[ \forall x^{(0)} \lim_{k \to \infty} \left\| A_1 A_2 \ldots A_k x^{(0)} \right\| = 0 \]

for any permutation \((A_1, A_2, \ldots, A_k) \in \text{Perm}(\Sigma)\)
Non-adaptive Stability Analysis

\[ \forall x^{(0)} \]

\[
\lim_{k \to \infty} \left\| \underbrace{AA \ldots A}_{k \text{ times}} x^{(0)} \right\| = 0 \iff \rho(A) < 1
\]

\[
\rho(A) = \lim_{k \to \infty} \max_{\|x\|=1} \left\| A^k x \right\|^{1/k}
\]

Upper bound:

\[
\rho(A) < \max_{\|x\|=1} \left\| A^k x \right\|^{1/k}
\]

<table>
<thead>
<tr>
<th>k</th>
<th>upper bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.000050</td>
</tr>
<tr>
<td>10</td>
<td>1.000049</td>
</tr>
<tr>
<td>100</td>
<td>0.999971</td>
</tr>
<tr>
<td>1000</td>
<td>0.999577</td>
</tr>
<tr>
<td>10000</td>
<td>0.999553</td>
</tr>
</tbody>
</table>
Adaptive Cosim Stability Analysis

Stability: \( \forall x^{(0)} \)

\[
\lim_{k \to \infty} \left\| \underbrace{A_1 A_2 \ldots A_k x^{(0)}}_{k \text{ times}} \right\| = 0 \quad \text{for any permutation} \ (A_1, A_2, \ldots, A_k) \in \text{Perm}(\Sigma)
\]

\[
\hat{\rho}_k(\Sigma) = \sup_{(A_1, A_2, \ldots, A_k) \in \text{Perm}(\Sigma)} \max_{\|x\|=1} \| A_1 A_2 \ldots A_k x \|
\]

\[
\rho(\Sigma) = \lim_{k \to \infty} \hat{\rho}_k(\Sigma)^{1/k}
\]

\[
\text{Stable? } \iff \rho(\Sigma) < 1
\]

Upper bound can be computed: \( \hat{\rho}(\Sigma) < \hat{\rho}_k(\Sigma)^{1/k} \)

http://doi.org/10.1137/110855272

http://doi.org/10.1016/j.laa.2007.12.027

Example

\[ \Sigma = \left\{ \begin{array}{ll}
A_{H=0.2, S_1=\langle FE, 0.04 \rangle, S_2=\langle FE, 0.2 \rangle}, & A_{H=0.1, S_1=\langle Mid, 0.01 \rangle, S_2=\langle FE, 0.1 \rangle} \end{array} \right\} \]

- \(A_{fe, H=0.01, S_1=\langle FE, 0.04 \rangle, S_2=\langle FE, 0.2 \rangle}\)
- \(A_{mid, H=0.1, S_1=\langle Mid, 0.01 \rangle, S_2=\langle FE, 0.1 \rangle}\)
Example

\[ \Sigma = \begin{cases} \mathbf{A}_{H=0.2, S_1=\langle FE, 0.04 \rangle, S_2=\langle FE, 0.2 \rangle}, & \mathbf{A}_{H=0.1, S_1=\langle Mid, 0.01 \rangle, S_2=\langle FE, 0.1 \rangle} \end{cases} \]

Forward Euler: \( h=0.04 \)

\( \lim_{k \to \infty} \max_{\|\mathbf{x}\|=1} \left\| \mathbf{A}_{fe}^k \mathbf{x} \right\|^{1/k} = \rho(\mathbf{A}_{fe}) > 1 \)

Midpoint: \( h=0.01 \)

Performance

Accuracy
Example

\[ \Sigma = \{ A_{mid}, A_{fe} A_{mid} \} \]

\[ \rho(\Sigma) < 0.99986 \]

https://github.com/blegat/SwitchOnSafety.jl
Summary of the Approach

1. Capture possible orchestration decisions
   - IP can be protected if solver is embedded

\[
\begin{bmatrix}
    x_1^{(n+1)} \\
    v_1^{(n+1)}
\end{bmatrix} = A_1^{k_1} \begin{bmatrix}
    x_1^{(n)} \\
    v_1^{(n)}
\end{bmatrix} + \left( \sum_{m=0}^{k_1-1} A_1^m B_1 \right) u_1^{(n)}
\]

\[
y_1^{(n)} = \begin{bmatrix}
    x_1^{(n)} \\
    v_1^{(n)}
\end{bmatrix}
\]

\[
\begin{bmatrix}
    x_2^{(n+1)} \\
    v_2^{(n+1)}
\end{bmatrix} = A_2^{k_2} \begin{bmatrix}
    x_2^{(n)} \\
    v_2^{(n)}
\end{bmatrix} + \left( \sum_{m=0}^{k_2-1} A_2^m B_2 \right) u_2^{(n)}
\]

\[
y_2^{(n)} = \begin{bmatrix}
    c_k & d_k \\
    -c_k & -d_k
\end{bmatrix} \begin{bmatrix}
    x_2^{(n)} \\
    v_2^{(n)}
\end{bmatrix} + \begin{bmatrix}
    u_2^{(n)}
\end{bmatrix}
\]

\[
\begin{bmatrix}
    x_1^{(n+1)} \\
    v_1^{(n+1)} \\
    x_2^{(n+1)} \\
    v_2^{(n+1)}
\end{bmatrix} = A_{H=0.2,S_2=\langle FE,0.04\rangle,S_2=\langle FE,0.2\rangle}
\begin{bmatrix}
    x_1^{(n)} \\
    v_1^{(n)} \\
    x_2^{(n)} \\
    v_2^{(n)}
\end{bmatrix}
\]

\[
\Sigma = \{ A_{mid}, A_{fe} \} 
\]
Summary of the Approach

1. Capture possible orchestration decisions
   • IP can be protected if solver is embedded

2. Check stability of unrestricted adaptive orchestration

\[ \Sigma = \{ A_{mid}, A_{fe} \} \]
\[ \rho(\Sigma) < 1 \]
Summary of the Approach

1. Capture possible orchestration decisions
   • IP can be protected if solver is embedded
2. Check stability of unrestricted adaptive orchestration
3. Restrict if needed

\[ \Sigma = \{ A_{mid}, A_{fe} \} \]

\[ \Sigma = \{ A_{mid}, A_{fe} A_{mid} \} \]
Limitations

• Scalability

• How to efficiently restrict the orchestration policy?
The Baby Steps:  
Stability of Hybrid Co-simulation 

Our contributions 

Paschalis Karalis, Eva Navarro-López, Hans Vangheluwe  

Running Example: Bouncing Ball

\[ \sigma(t) \in \{1, 2\} \]

\[ \begin{align*}
    &\dot{x}_1 = x_2 \\
    &\dot{x}_2 = \frac{1}{m} \left( -(c_s + c_c)x_1 - (d_s + d_c)x_2 \right) \quad x_1 \leq l \\
    &\dot{x}_1 = x_2 \\
    &\dot{x}_2 = \frac{1}{m} \left( -c_s x_1 - d_s x_2 \right) \quad x_1 > l
\end{align*} \]

Bouncing ball Co-simulation

\[ t := t + H \]

Orchestrator

Controller Simulator \rightarrow Plant Simulator

Controller Simulator \leftarrow Plant Simulator

\text{Ctrl. Simulator}

\text{Plant Simulator}

getOutput(…)

setInput(…)

getOutput(…)

setInput(…)

simulateUntil(t+H,…)

simulateUntil(t+H,…)

\[ t := t + H \]

...
The Challenge: Energy Conservation

![Graph showing mode and energy over time]
The Challenge: Energy Conservation

![Graphs showing energy and mode position over time](image)

- Mode position
- Energy

H = 0.001
The Challenge: Energy Conservation

H = 0.01
The Challenge: Energy Conservation

The Challenge: Energy Conservation
The Challenge: Energy Conservation

Controller

Plant

\[ \sigma(t) \]

\[ g(x) = 0 \]

\[ x(t) \]

\[ t_c \quad t_i \quad t_i + H \]

H=0.01

H=0.1

mode position

mode position
Outline of the Approach

Original System:

Controller Simulator \quad \text{Plant Simulator}

Hybrid Automaton:

Original System:

Hybrid Automaton:

\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \frac{1}{m} \left( -(c_s + c_c)x_1 - (d_s + d_c)x_2 \right)
\end{align*}

\text{Non-deterministic Hybrid Automaton:}

\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \frac{1}{m} \left( -(c_s x_1 - d_s x_2) \right)
\end{align*}

\text{Stability range computation:}

\text{Non-deterministic Hybrid Automaton:}

\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \frac{1}{m} \left( -(c_s + c_c)x_1 - (d_s + d_c)x_2 \right)
\end{align*}

\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \frac{1}{m} \left( -(c_s x_1 - d_s x_2) \right)
\end{align*}
Hybrid Automata Creation

- Limited to bi-modal scenarios
- Requires access to each mode’s dynamics

\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \frac{1}{m} \left( -(c_s + c_c)x_1 - (d_s + d_c)x_2 \right) \\
&\quad \text{if } x_1 \leq l
\end{align*}
\]

\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \frac{1}{m} \left( -c_s x_1 - d_s x_2 \right) \\
&\quad \text{if } x_1 > l
\end{align*}
\]
Invariant Relaxation

Hybrid Automaton:
\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \frac{1}{m} \left(- (c_s + c_c) x_1 - (d_s + d_c) x_2 \right)
\end{align*}
\]

\[ x_1 \leq l \]

\[ x_1 > l \]

Non-deterministic Hybrid Automaton:
\[
\begin{align*}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \frac{1}{m} \left(- (c_s + c_c) x_1 - (d_s + d_c) x_2 \right)
\end{align*}
\]

\[ x_1 \leq l + \xi_1(H) \]

\[ x_1 > l - \xi_2(H) \]
Approximate Stability Analysis

Non-deterministic Hybrid Automaton:

1. \[ \dot{x}_1 = x_2 \]
   \[ \dot{x}_2 = \frac{1}{m} \left( -(c_s + c_c)x_1 - (d_s + d_c)x_2 \right) \]
   \[ x_1 \leq l + \xi_1(H) \]

2. \[ \dot{x}_1 = x_2 \]
   \[ \dot{x}_2 = \frac{1}{m} \left( -c_s x_1 - d_s x_2 \right) \]
   \[ x_1 > l - \xi_2(H) \]
Algorithm

Range of initial conditions given

Initial maximum step size:

H=0.1
Algorithm

\[ H = 0.01 \]
Algorithm

$H=0.001$
Results – Cart Example

• Mode 1: $H < 0.002232$

• Mode 2: $H < 0.002204$

• Safest $H = 0.002204$
Conclusion & Future Work

• Conservative algorithm.

• Restricted to bi-modal planar hybrid systems.

• Restricted to bounded initial conditions.

• Requires access to state and derivatives.

• Jacobi orchestration only.

• Use stronger stability theorems