Co-simulation: State of the art

Cláudio Gomes    Casper Thule    David Broman
Peter Gorm Larsen    Hans Vangheluwe

Modeling, Simulation, and Design Lab (MSDL)

March 2, 2017
Why co-simulation?
And why are we here?
Motivation(s) for Co-simulation

Definition: Simulation of a coupled system, via the composition of sub-system simulations.

Main reasons:
- Performance/Accuracy;
- Heterogeneity of languages and tools;
- Intellectual Property protection;

Main goal: unlock the full potential of simulation.
Why are we here?

Simulation of a coupled system... ... via the composition of subsystem simulations.
Outline

▶ Terminology
▶ Simulation units
▶ Input extrapolation techniques
▶ Orchestration algorithms
▶ Algebraic loops
▶ Convergence
▶ Stability
▶ Wrap-up
▶ Ongoing work
Simulation of a coupled system...
... via the composition of subsystem simulations.
Dynamical Systems

\[ \dot{x} = f(x, u) \]
\[ y = g(x, u) \]
\[ x(0) = p \]

| Experimental Frame | Validity |
|--------------------|----------|
| Valid model        | ✓        |
| Invalid model      | ✓        |
| N/A                | ×        |

Real data

Behavior trace

Deformation
Simulators

Correct SU = Accurate Simulator + Valid Model

\[ \dot{x} = f(x, u) \]
\[ y = g(x, u) \]
\[ x(0) = p \]
Simulation Unit

$$S_i = \langle X_i, U_i, Y_i, \delta_i, \lambda_i, x_i(0), \phi_{U_i} \rangle$$

$$\delta_i : \mathbb{R} \times X_i \times U_i \rightarrow X_i$$

$$\lambda_i : \mathbb{R} \times X_i \times U_i \rightarrow Y_i \text{ or } \mathbb{R} \times X_i \rightarrow Y_i$$

$$x_i(0) \in X_i$$

$$\phi_{U_i} : \mathbb{R} \times U_i \times \ldots \times U_i \rightarrow U_i$$
Simulation Unit and the Functional Mockup Unit

\[ S_i = \langle X_i, U_i, Y_i, \delta_i, \lambda_i, x_i(0), \phi U_i \rangle \]

\[ y_i := \lambda_i(t, x_i, u_i); \]
\[ x_i := \delta_i(H, x_i, u_i); \]

\[ x_i := \delta_i(H, x_i, u_i); \]
\[ y_i := \lambda_i(t + H, x_i, u_i); \]

fmi2SetFMUstate\(_i\)(\(c_i, x_i\));
fmi2SetReal\(_i\)(\(c_i, \ldots, \dim(U_i), u_i\));
fmi2GetReal\(_i\)(\(c_i, \ldots, \dim(Y_i), y_i\));
fmi2DoStep\(_i\)(\(c_i, t, H, \ldots\));
fmi2GetFMUstate\(_i\)(\(c_i, &x_i\));

fmi2SetFMUstate\(_i\)(\(c_i, x_i\));
fmi2SetReal\(_i\)(\(c_i, \ldots, \dim(U_i), u_i\));
fmi2DoStep\(_i\)(\(c_i, t, H, \ldots\));
fmi2GetFMUstate\(_i\)(\(c_i, &x_i\));
fmi2SetReal\(_i\)(\(c_i, \ldots, \dim(U_i), u_i\));
fmi2GetReal\(_i\)(\(c_i, \ldots, \dim(Y_i), y_i\));
Types of Simulation Units

\[ S_i = \langle X_i, U_i, Y_i, \delta_i, \lambda_i, x_i(0), \phi_{U_i} \rangle \]

State transition:

- **reactive** \[ x_i(t + H) = \delta_i(H, x_i(t), u_i(t + H)) \]
- **delayed** \[ x_i(t + H) = \delta_i(H, x_i(t), u_i(t)) \]

Output:

- **mealy** \[ y_i(t) = \lambda_i(t, x_i(t), u_i(t)) \]
- **moore** \[ y_i(t) = \lambda_i(t, x_i(t)) \]
## Types of Input Extrapolations

| Type                        | Equation |
|-----------------------------|----------|
| **Constant**                | \( u(t) \) |
| **Linear**                  | \( u(t) \) |
| **Polynomial**              | \( u(t) \) |
| **Extrapolated/Interpolation** | \( \phi_U_i(H, u_i(t - H), \ldots) \) |
| **Context-aware**           | \( \phi_U_i = \{ \ldots \} \) |
| **Model ID’ed**             | \( \phi_U_i = \tilde{g}(w, \ldots) \) |
Checkpoint

- Dynamical Systems
- Simulators
- Simulation units (externals and internals)

- Interactions between Simulation Units
Co-simulation Scenario

\[ \langle \{ S_i : i \in D \} , L \rangle \]

\[ S_i = \langle X_i, U_i, Y_i, \delta_i, \lambda_i, x_i(0), \phi U_i \rangle \]

\[ L : (\prod_{i \in D} Y_i) \times Y_{CS} \times (\prod_{i \in D} U_i) \times U_{CS} \rightarrow \mathbb{R}^m \]

Coupling: \( L = 0 \)

\[ \langle \{ 1, 2 \} , \{ S_1, S_2 \} , L \rangle \]

\[ L = \begin{bmatrix} x_c - v_1 \\ \dot{x}_c - x_1 \\ F_e - F_c \end{bmatrix} \]
Jacobi Type Orchestrator

\[ \langle \{ S_i : i \in D \} , L \rangle \]

\[ S_i = \langle X_i, U_i, Y_i, \delta_i, \lambda_i, x_i(0), \phi U_i \rangle \]

\[ L : (\prod_{i \in D} Y_i) \times Y_{CS} \times (\prod_{i \in D} U_i) \times U_{CS} \to \mathbb{R}^m \]

\[
\begin{align*}
t &:= 0 ; \\
x_i &:= x_i(0) \text{ for } i = 1, \ldots, n ; \\
\textbf{while true do} & \\
\quad &\text{Solve the following system for the unknowns:} \\
\quad &\begin{cases}
    y_1 = \lambda_1(t, x_1, u_1) \\
    \vdots \\
    y_n = \lambda_n(t, x_n, u_n) \\
    L(y_1, \ldots, y_n, y_{CS}, u_1, \ldots, u_n) = \bar{0}
\end{cases} \\
\quad &x_i := \delta_i(H, x_i, u_i), \text{ for } i = 1, \ldots, n ; \\
\quad &t := t + H \\
\textbf{end}
\]

* Delayed units only.
Compositional Co-simulation

The *raison d’etre* of the orchestrator is to produce a correct co-simulation trace, assuming that *each simulation unit is correct*. 
Algebraic Loops

I/O

\( x_i, x_j \) known.

\[
\begin{align*}
  y_i &= \lambda_i(t, x_i, u_i) \\
  u_j &= y_i \\
  y_j &= \lambda_j(t, x_j, u_j) \\
  u_i &= y_j
\end{align*}
\]

State and I/O

\( x_i, x_j, y_j, u_i \) known.

\[
\begin{align*}
  \tilde{x}_i &= \delta_i(H, x_i, u_i) \\
  \tilde{y}_i &= \lambda_i(t + H, \tilde{x}_i, \tilde{u}_i) \\
  \tilde{u}_j &= \tilde{y}_i \\
  \tilde{x}_j &= \delta_j(H, x_j, \tilde{u}_j) \\
  \tilde{y}_j &= \lambda_j(t + H, \tilde{x}_j) \\
  \tilde{u}_i &= \tilde{y}_j
\end{align*}
\]

Fixed point iterations

Strong coupling, Waveform iteration, Semi-implicit
Algebraic Couplings

\[ g(F_e) = \hat{x}_1(F_e) - \hat{x}_3(-F_e) = 0 \]

1. Guess \( F_e \);
2. \( \hat{x}_1 := \delta_1(H, x_1, F_e) \);
3. \( \hat{x}_3 := \delta_i(H, x_3, -F_e) \);
4. if \( \hat{x}_1 \approx \hat{x}_3 \) then
   5. end
   6. if \( \hat{x}_1 \approx \hat{x}_3 \) then
      7. \( \bar{x}_1 := \delta_1(H, x_1, F_e + \epsilon) \);
      8. \( \bar{x}_3 := \delta_i(H, x_3, -F_e + \epsilon) \);
      9. \( \frac{\partial x_1}{\partial F_e} \approx \frac{\bar{x}_1 - \hat{x}_1}{\epsilon} \);
      10. \( \frac{\partial x_2}{\partial F_e} \approx \frac{\bar{x}_3 - \hat{x}_3}{\epsilon} \);
      11. \( \frac{\partial g}{\partial F_e} = \frac{\partial x_1}{\partial F_e} + \frac{\partial x_2}{\partial F_c} \);
      12. \( F_e := F_e(n \cdot H) - \left[ \frac{\partial g(F_e(n \cdot H))}{\partial F_e} \right]^{-1} \cdot g(F_e(n \cdot H)) \);
      13. Go to Line 1;
Error Control

- Convergence – Deviation of co-simulation trace from true solution $e(t)$ ultimately $(t \to 0)$ tends to zero, as $H \to 0$.
  - Sufficient condition: coupled model is an ODE.
  - Danger: algebraic loops in the coupled model.
  - Order bottle neck is $\phi U_i$.

- Error Estimation
  - Richardson extrapolation: compare steps with half-steps;
  - Multi-Order Input Extrapolation: compare different order input approximations;
  - Milne’s Device: compare guessed input with given input;
  - Parallel Embedded Method: take derivative of some unit and run an ODE solver in parallel;
  - Conservation Laws: track energy excesses/defects;
  - Embedded Solver Method: let units decide;

- Step size selection: all traditional simulation techniques apply.
Stability

Relevant question: does the orchestrator cause \( \lim_{t \to \infty} e(t) \neq 0 \), for \( H > 0 \)?

- Assume coupled ODE system (which must be LTI) is stable: \( \lim_{t \to \infty} \dot{\hat{x}}(t) = 0 \)
- Write each simulation unit as a discrete time system:
  \[
  x_i^{(n+1)} = e^{A_i H} x_i^{(n)} + K_i B_i u_i^{(n)}
  \]
- And its output:
  \[
  y_i^{(n+1)} = C_i e^{A_i H} x_i^{(n)} + (C_i K_i B_i [+D_i]) u_i^{(n)}
  \]
- Replacing all inputs \( u_i \) by the coupling conditions, we get a big discrete system:

\[
\begin{bmatrix}
  x_1^{(n+1)} \\
  v_1^{(n+1)} \\
  y_1^{(n+1)} \\
  x_2^{(n+1)} \\
  v_2^{(n+1)} \\
  y_2^{(n+1)}
\end{bmatrix}
= \begin{bmatrix}
  e^{A_1 H} & 0 & 0 & 0 & K_1 B_1 \\
  C_1 e^{A_1 H} & 0 & 0 & 0 & C_1 K_1 B_1 \\
  0 & K_2 B_2 & e^{A_2 H} & 0 & 0 \\
  C_2 K_2 B_2 + D_2 & 0 & C_2 e^{A_2 H} & 0 & 0
\end{bmatrix}
\begin{bmatrix}
  x_1^{(n)} \\
  v_1^{(n)} \\
  y_1^{(n)} \\
  x_2^{(n)} \\
  v_2^{(n)} \\
  y_2^{(n)}
\end{bmatrix}
\]

- Check if \( \rho(A) < 1 \)
Summary

- Simulation units
- Orchestration algorithms
- Compositionality for correct co-simulation
- Threats to compositionality
Thank you!
References

[1] Cláudio Gomes. Foundations for Co-simulation – IWT Proposal. Technical report, University of Antwerp, Antwerp, 2015.

[2] Cláudio Gomes. Foundations for Continuous Time Hierarchical Co-simulation. In ACM Student Research Competition (ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems), page to appear, Saint Malo, Brittany, France, 2016.

[3] Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, and Hans Vangheluwe. Co-simulation: State of the art. Technical report, feb 2017.