EFFICIENT EXTRAPOLATION FOR PARALLEL CO-SIMULATION OF COUPLED SYSTEMS

(CHOPtrey)

Abir Ben Khaled - El Feki, Laurent Duval, Mongi Ben Gaid
OUTLINE

- Background on co-simulation: context & challenges
- Results from previous work
- Ensuring co-simulation accuracy with CHOPtrey extrapolation approach
- Conclusion and perspectives
BACKGROUND

Co-simulation: Alternative to monolithic simulation → Simulation of a complex system using several coupled subsystems

- A subsystem is modeled using the most appropriate tool instead of using a single modeling software
- Subsystems are modeled and run in a segregated manner → The equations of each model are integrated using a solver separately
- During the execution models exchange data → A synchronization mechanism is used between the models, in such a way that models update their inputs and outputs according to assigned communication steps
- Easy upgrade, reuse, and exchange of models
**BACKGROUND**

- Co-simulation: Alternative to monolithic simulation → Simulation of a complex system using several coupled subsystems
  - A subsystem is modeled using the most appropriate tool instead of using a single modeling software
  - Subsystems are modeled and run in a segregated manner → The equations of each model are integrated using a solver separately
  - During the execution models exchange data → A synchronization mechanism is used between the models, in such a way that models update their inputs and outputs according to assigned communication steps
  - Easy upgrade, reuse, and exchange of models
  - Heterogeneous ODE models → Time consuming simulations
A multi-core co-simulation kernel: Why?

- System-level simulation leads to put together models which are classically disconnected, increasing the CPU demand at simulation time.
- Simulation time becomes an important metric for model complexity.
- Most 0D/1D simulation tools have mono-core kernel and doesn’t exploit available parallelism provided by multi-core computers.

How long will this CPU power remain unused?
BACKGROUND (CONT’D)

- Is integrated with its own solver
- Communicates its data at its own rate

ASAP simulation or HiL
Acceleration thanks to multi-core

FMU
Dymola®
GT-Power®
Simulink®
xMOD™ IFPEN co-simulation software
OUTLINE

- Background on co-simulation: context & challenges
- Results from previous work
- Ensuring co-simulation accuracy with CHOPtrey extrapolation approach
- Conclusion and perspectives
RESULTS FROM PREVIOUS WORK

Case study: Engine simulator

- Spark Ignition engine (Renault)
  - 4 cylinders + Airpath
  - 118 state variables
  - 312 event indicators

- Modeling & simulation tools
  - Dymola (ModEngine library)
  - xMOD (FMUs)

- Solver
  - LSODAR: Root-finding / Stiffness detection
RESULTS FROM PREVIOUS WORK

Case study: Engine simulator

- Spark Ignition engine (Renault)
  - 4 cylinders + Airpath
  - 118 state variables
  - 312 event indicators
- Modeling & simulation tools
  - Dymola (ModEngine library)
  - xMOD (FMUs)
- Solver
  - LSODAR: Root-finding / Stiffness detection
- Multi-core simulation
  - 5 components on 5 cores
RESULTS FROM PREVIOUS WORK

- **Case study: Engine simulator**
  - Spark Ignition engine (Renault)
    - 4 cylinders + Airpath
    - 118 state variables
    - 312 event indicators
  - Modeling & simulation tools
    - Dymola (ModEngine library)
    - xMOD (FMUs)
  - Solver
    - LSODAR: Root-finding / Stiffness detection
  - Multi-core simulation
    - 5 components on 5 cores

- **Splitting is speed-up**
  - Events are related usually to the evolution of a subset of the state vector
  - Discontinuities are independent from a physical point of view
RESULTS FROM PREVIOUS WORK
SPLITTING IS SPEED-UP (CONT’D)

● Number of events is reduced locally

![Diagram showing results from previous work with splitting speed-up]

- Complete engine: Few events
- AirPath: Almost no events
- Cylinder1: Few events
- Cylinder2: Almost no events
RESULTS FROM PREVIOUS WORK
SPLITTING IS SPEED-UP (CONT’D)

- Number of events is reduced locally
- Integration step can reach maximum allowed value (500µs)
RESULTS FROM PREVIOUS WORK
SPLITTING IS SPEED-UP (CONT’D)

- Number of events is reduced locally
- Integration step can reach maximum allowed value (500µs)
  - Mean value increased from 150µs to 230µs
RESULTS FROM PREVIOUS WORK
SPLITTING IS SPEED-UP (CONT’D)

● Number of events is reduced locally
● Integration step can reach maximum allowed value (500µs)
  ➔ Mean value increased from 150µs to 230µs
● Result on speed-up
  ● Mono-core simulation
    ● 5 threads on 1 core
    ● Speed-up ≈ 2
  ● Thanks to System splitting & Solver coupling
  ● Despite multi threading cost
RESULTS FROM PREVIOUS WORK
SPLITTING IS SPEED-UP (CONT’D)

- Number of events is reduced locally
- Integration step can reach maximum allowed value (500µs)
  ➔ Mean value increased from 150µs to 230µs
- Result on speed-up
  - Mono-core simulation
    - 5 threads on 1 core
    - Speed-up ≈ 2
    - Thanks to System splitting & Solver coupling
  - Multi-core simulation
    - 5 threads on 5 cores

![Diagram showing complete engine and individual cylinders with time-step and speed-up metrics.](image)

\[ \text{time-step} = \text{max} \]

\[ \text{Speed-up} \approx 2 \]

Thanks to System splitting & Solver coupling

Despite multi threading cost

Cyl1, Cyl2, Cyl3, Cyl4
RESULTS FROM PREVIOUS WORK
SPLITTING IS SPEED-UP (CONT’D)

- Number of events is reduced locally
- Integration step can reach maximum allowed value (500µs)
  - Mean value increased from 150µs to 230µs
- Result on speed-up
  - Mono-core simulation
    - 5 threads on 1 core
    - Speed-up ≈ 2
    - Thanks to System splitting & Solver coupling
  - Multi-core simulation
    - 5 threads on 5 cores
    - Speed-up ≈ 8 (AP then 4Cyls in //)

```
RESULTS FROM PREVIOUS WORK
SPLITTING IS SPEED-UP (CONT’D)

- Number of events is reduced locally
- Integration step can reach maximum allowed value (500µs)
  - Mean value increased from 150µs to 230µs
- Result on speed-up
  - Mono-core simulation
    - 5 threads on 1 core
    - Speed-up ≈ 2
    - Thanks to System splitting & Solver coupling
  - Multi-core simulation
    - 5 threads on 5 cores
    - Speed-up ≈ 8 (AP then 4Cyls in //)
```

![Diagram](image-url)
RESULTS FROM PREVIOUS WORK
SPLITTING IS SPEED-UP (CONT’D)

- Number of events is reduced locally
- Integration step can reach maximum allowed value (500μs)
  - Mean value increased from 150μs to 230μs
- Result on speed-up
  - Mono-core simulation
    - 5 threads on 1 core
    - Speed-up ≈ 2
    - Thanks to System splitting & Solver coupling
    - Despite multi threading cost
  - Multi-core simulation
    - 5 threads on 5 cores
    - Speed-up ≈ 8 (AP then 4Cyls in //)
    - Speed-up ≈ 9 (both AP and 4Cyls in //)
RESULTS FROM PREVIOUS WORK
IMPROVING PARALLELISM WITH THE RCOSIM APPROACH

System splitting brings virtual algebraic loops

\[
\begin{align*}
\dot{X} &= f(t, X, U_{ext}) \\
Y_{ext} &= g(t, X, U_{ext})
\end{align*}
\]
RESULTS FROM PREVIOUS WORK
IMPROVING PARALLELISM WITH THE RCOSIM APPROACH

- System splitting brings virtual algebraic loops
- Involve delayed outputs, even with an efficient execution order
- Problem with accuracy

\[
\begin{align*}
\dot{X} &= f(t, X, U_{ext}) \\
Y_{ext} &= g(t, X, U_{ext})
\end{align*}
\]
RESULTS FROM PREVIOUS WORK
RCOSIM: REFINED CO-SIMULATION

Before FMI
- Only dependencies between models are specified by the user
- Models are black boxes → can’t identify locally if Y is dependent on U

With FMI
- Relationships between each Y and U is known
- Each Y and U is computed with a different FMU function

→ Build refined dependency graph
- Vertices: IN, OUT and STATE operations
- Directed edges: precedencies between operations
- Target: Ordinary Differential Equations (ODEs)
  - No algebraic loops → Directed Acyclic Graph

Apply a multi-core scheduling heuristic on the dataflow graph
RESULTS FROM PREVIOUS WORK
IMPROVING PARALLELISM WITH THE RCOSIM APPROACH

- Torque is a direct feedthrough output: e.g. $Y_{A3}$
- Expected delays with Standard Co-simulation (Std-Cosim) due to arbitrary order execution decision between models
RESULTS FROM PREVIOUS WORK
IMPROVING PARALLELISM WITH THE RCOSIM APPROACH

- Torque is a direct feedthrough output: e.g. $Y_{A3}$
- Expected delays with Standard Co-simulation (Std-Cosim) due to arbitrary order execution decision between models
- Elimination of delays with RCosim
  - The execution order is compliant with initial model
- Speed-up $\approx 10$
  - No more delays $\rightarrow$ Correct data $\rightarrow$ Less iteration of the solver

\[
\begin{array}{|c|c|c|}
\hline
\text{Simulation method} & \text{Std-Cosim} & \text{RCosim} \\
\hline
\text{Er(\%)} \text{ with } H=100\mu s & 2.95 & 0.68 \\
\text{Er(\%)} \text{ with } H=250\mu s & 9.12 & 1.1 \\
\text{Er(\%)} \text{ with } H=500\mu s & 19.83 & 1.37 \\
\hline
\end{array}
\]
OUTLINE

- Background on co-simulation: context & challenges
- Results from previous work
- Ensuring co-simulation accuracy with CHOPtrey extrapolation approach
- Conclusion and perspectives
**Limitation**: with RCosim, errors are reduced but still exist

**Reason**: Input data is held constant during the communication step

**Dilemma**: 
- Speed-up
- Integration error

**CHOPtrey EXTRAPOLATION APPROACH**
**IMPROVE AGAIN THE SIMULATION ACCURACY**
**CHOPtrey EXTRAPOLATION APPROACH**

**IMPROVE AGAIN THE SIMULATION ACCURACY**

- **Limitation:** with RCoSim, errors are reduced but still exist
- **Reason:** Input data is held constant during the communication step
- **Dilemma:**
  - Speed-up
  - Integration error
- **Idea:** Extrapolate input signals to
  - Enlarge intervals
  - Reduce simulation errors
RELATED WORK ON PREDICTION

**Difficulties**
- Related work on extrapolations treated mostly the continuous case
  - Successful for non-stiff systems
  - Encountered problems with stiff systems → polynomial prediction may fail
- Complex systems with hybrid behavior is even more difficult to predict
  - Nonlinearities, discontinuities,...
  - No universal prediction scheme, efficient with every signal

**Challenges**: fast, causal and reliable prediction
- Predictor computing cost << extra model computations with small communication steps
- Accurate predictions for any signal (blocky/smooth; slow/steep onsets)

**Idea**: Borrow the concept of context-based approach from lossless image encoders
- Predict a pixel value based on a pattern of causal neighboring pixels
- Different contexts: flat, smooth, +45° or -45° edges, etc.
We propose a Computationally Hasty Online Prediction framework (CHOPred)

It is based on Causal Hopping Oblivious Polynomials (CHOPoly)

$P_{\delta,\lambda,\omega}$: least squares polynomial predictor
- $\delta$: prediction degree;
- $\lambda$: prediction frame length;
- $\omega$: weighting factor

$u$: input signal; $\tau$: relative time for prediction

Weighted moment: $\bar{m}_{d,\lambda,\omega} = \sum_{l=0}^{\lambda-1} (\lambda - l)^{\omega} l^d \bar{u}_l$.

Weighted sum of integer powers: $\bar{z}_{d,\lambda,\omega} = \sum_{l=0}^{\lambda-1} (\lambda - l)^{\omega} l^d$

General formula for extrapolation:

$$u(\tau) = \begin{bmatrix} 1 & \tau & \cdots & \tau^\delta \end{bmatrix} \begin{bmatrix} \bar{z}_0,\lambda,\omega & -\bar{z}_1,\lambda,\omega & \cdots & (-1)^\delta \bar{z}_{\delta,\lambda,\omega} \\ -\bar{z}_1,\lambda,\omega & \cdots & \cdots & \cdots \\ \vdots & \ddots & \ddots & \ddots \\ \vdots & \cdots & \cdots & \cdots \\ \bar{m}_{0,\lambda,\omega} & \bar{m}_{1,\lambda,\omega} & \cdots & \bar{m}_{\delta,\lambda,\omega} \end{bmatrix}^{-1} \begin{bmatrix} \bar{m}_{0,\lambda,\omega} \\ -\bar{m}_{1,\lambda,\omega} \\ \vdots \\ \vdots \end{bmatrix}$$
CHOPtrey EXTRAPOLATION APPROACH
A FAST AND CAUSAL PREDICTION

- We propose a Computationally Hasty Online Prediction framework (CHOPred)
- It is based on Causal Hopping Oblivious Polynomials (CHOPoly)
- \( P_{\delta,\lambda,\omega} \): least squares polynomial predictor
  - \( \delta \): prediction degree;
  - \( \lambda \): prediction frame length;
  - \( \omega \): weighting factor
- \( u \): input signal; \( \tau \): relative time for prediction
- Weighted moment: \( \bar{m}_{d,\lambda,\omega} = \sum_{l=0}^{\lambda-1} (\lambda - l)^\omega \bar{p}^d u_{l-1} \)
- Weighted sum of integer powers: \( \bar{z}_{d,\lambda,\omega} = \sum_{l=0}^{\lambda-1} (\lambda - l)^\omega \bar{p}^d \)
- General formula for extrapolation:

\[
\hat{u}(\tau) = \begin{bmatrix} 1 & \tau & \cdots & \tau^\delta \end{bmatrix} \begin{bmatrix} \bar{z}_{0,\lambda,\omega} & -\bar{z}_{1,\lambda,\omega} & \cdots & (-1)^\delta \bar{z}_{\delta,\lambda,\omega} \\ \vdots & \ddots & \ddots & \vdots \\ -\bar{z}_{1,\lambda,\omega} & \cdots & \cdots & \bar{z}_{2\delta,\lambda,\omega} \\ \vdots & \ddots & \ddots & \vdots \\ (-1)^\delta \bar{m}_{0,\lambda,\omega} & \cdots & \cdots & (-1)^\delta \bar{m}_{\delta,\lambda,\omega} \end{bmatrix}^{-1} \begin{bmatrix} \bar{m}_{0,\lambda,\omega} \\ -\bar{m}_{1,\lambda,\omega} \\ \vdots \\ -\bar{m}_{\delta,\lambda,\omega} \end{bmatrix}
\]

Use of LUT ➔ Fast computation
We propose a Computationally Hasty Online Prediction framework (CHOPred)

It is based on Causal Hopping Oblivious Polynomials (CHOPoly)

$P_{\delta,\lambda,\omega}$: least squares polynomial predictor
- $\delta$: prediction degree;
- $\lambda$: prediction frame length;
- $\omega$: weighting factor

$u$: input signal; $\tau$: relative time for prediction

Weighted moment: $\overline{m}_{d,\lambda,\omega} = \sum_{l=0}^{\lambda-1} (\lambda - l)^{\omega} l^d u_{l-1}$

Weighted sum of integer powers: $\overline{z}_{d,\lambda,\omega} = \sum_{l=0}^{\lambda-1} (\lambda - l)^{\omega} l^d$

General formula for extrapolation:

$u(\tau) = \begin{bmatrix} 1 & \tau & \cdots & \tau^\delta \end{bmatrix}$
CHOPtrex EXTRAPOLATION APPROACH
A RELIABLE PREDICTION

- It uses a Contextual & Hierarchical Ontology of Patterns (CHOPatt)
  - To handle the discontinuities by selecting the appropriate $P_{\delta,\lambda,\omega}$
- **STEP1**: Decisional context selection
  - Worst case scenario without extrapolation: $\Delta_{\text{worst}} = |u_0 - u_{-1}|$
  - Best prediction pattern: $\Delta_{\text{best}} = \min_{\omega \in \Omega} |u_0 - \hat{u}_{-1}^{\omega}|$; $\Omega = \{0, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1, 2\}$
  - Ratio: $\rho = \frac{\Delta_{\text{best}}}{\Delta_{\text{worst}}}$
  - Threshold: $0.7 \leq \Gamma < 1$ e.g. $\Gamma = 90\%$
  - If $\rho > \Gamma$ then sharp and fast variation $\Rightarrow$ Select “cliff” context
CHOPtrey EXTRAPOLATION APPROACH
A RELIABLE PREDICTION

**STEP2:** Functional context selection

- **Differences (variations):**
  \[ d_0 = u_0 - u_{-1} \quad \text{and} \quad d_{-1} = u_{-1} - u_{-2} \]

- **Thresholds:**
  \[ \gamma_0 = \gamma_{-1} = \frac{1}{2} \max_{i \in [-1,-2,-3]} (|u_i - u_{i+1}|) \]

- **Conditions:**
  - \( O \) if \( |d_i| = 0 \);
  - \( C_i \) if \( 0 < |d_i| \leq \gamma_i \);
  - \( \overline{C_i} \) if \( |d_i| > \gamma_i \).
**STEP2: Functional context selection**

- Differences (variations): \( d_0 = u_0 - u_{-1} \) and \( d_{-1} = u_{-1} - u_{-2} \)
- Thresholds: \( \gamma_0 = \gamma_{-1} = \frac{1}{\max_{i \in \{1-\lambda, ..., -3\}} \left| u_i - u_{i+1} \right|} \)
- Conditions:
  - \( O \) if \( |d_i| = 0 \);
  - \( C_i \) if \( 0 < |d_i| < \gamma_i \);
  - \( \overline{C_i} \) if \( |d_i| > \gamma_i \).

| n(ame) | # | \( |d_{-1}| \) | \( |d_0| \) | \( d_{-1}.d_0 \) | \( (\delta, \lambda, \omega) \) |
|--------|---|----------------|----------------|----------------|----------------|
| f(lat) | 0 | \( O \) | \( O \) | \( O \) | \( (0, 1,. ,) \) |
| c(alm) | 1 | \( C_1 \) | \( C_2 \) | any | \( (2, 5,. ,) \) |
| m(ove) | 2 | \( C_1 \) | \( \tilde{C}_2 \) | any | \( (0, 1,. ,) \) |
| r(est) | 3 | \( \tilde{C}_1 \) | \( C_2 \) | any | \( (0, 2,. ,) \) |
| t(ake) | 4 | \( \tilde{C}_1 \) | \( \tilde{C}_2 \) > 0 | \( (1, 3,. ,) \) |
| j(ump) | 5 | \( \tilde{C}_1 \) | \( \tilde{C}_2 \) < 0 | \( (0, 1,. ,) \) |
SIMULATION RESULTS WITH CHOPtrey

AUTOMATIC DETECTION OF SHARP VARIATION

- Conventional 1\textsuperscript{st} & 2\textsuperscript{nd} order extrapolation
  - Fails on the engine model
  - Major causes:
    - Discontinuities
    - Sharp variations

➔ CHOPtrey?
SIMULATION RESULTS WITH CHOPtrey
AUTOMATIC DETECTION OF SHARP VARIATION

- Conventional 1\textsuperscript{st} & 2\textsuperscript{nd} order extrapolation
  - Fails on the engine model
  - Major causes:
    - Discontinuities
    - Sharp variations

\rightarrow CHOPtrey?
SIMULATION RESULTS WITH CHOPtrey
AUTOMATIC DETECTION OF SHARP VARIATION

- Conventional 1\textsuperscript{st} & 2\textsuperscript{nd} order extrapolation
  - Fails on the engine model
  - Major causes:
    - Discontinuities
    - Sharp variations

\(\Rightarrow\) CHOPtrey?
SIMULATION RESULTS WITH CHOPtrey
AUTOMATIC SELECTION OF THE WEIGHTING FACTOR

- Simple model with no coupling
  ➔ The higher the weighting factor, the smaller the error
SIMULATION RESULTS WITH CHOPtrey
AUTOMATIC SELECTION OF THE WEIGHTING FACTOR

- Simple model with no coupling
  ➔ The higher the weighting factor, the smaller the error

- Complex coupled models, i.e. engine model
  ➔ No unique best weighting factor $\omega$
SIMULATION RESULTS WITH CHOPtrey
AUTOMATIC SELECTION OF THE WEIGHTING FACTOR

- Simple model with no coupling
  ➔ The higher the weighting factor, the smaller the error

- Complex coupled models, i.e. engine model
  ➔ No unique best weighting factor $\omega$

- Dynamic selection of $\omega$
  - At each communication step, $\omega_{\text{best}}$ is selected and used for the current step
  ➔ Cumulative integration error is the lowest one
CHOPtrey PERFORMANCE
SPEED-UP VERSUS ACCURACY

- The speed-up factor is still compared with single-threaded reference
- The model is split into 5 threads integrated in parallel on 5 cores
  - Containment of events detection handling $\rightarrow$ solvers accelerations $\rightarrow$ overcompensate multi-threading costs
- The relative error variation is compared with ZOH at 100 $\mu$s

<table>
<thead>
<tr>
<th>Communication step</th>
<th>Prediction</th>
<th>Speed-up factor</th>
<th>Relative error variation (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Burned gas density</td>
</tr>
<tr>
<td>100 $\mu$s</td>
<td>ZOH</td>
<td>8.9</td>
<td>$+12.5%$</td>
</tr>
<tr>
<td>250 $\mu$s</td>
<td>ZOH</td>
<td>10.01</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>CHOPtrey</td>
<td>10.07</td>
<td>-26</td>
</tr>
</tbody>
</table>
OUTLINE

- Background on co-simulation: context & challenges
- Results from previous work
- Ensuring co-simulation accuracy with CHOPtrey extrapolation approach
- Conclusion and perspectives
CONCLUSION

- The use of large communication steps allows to accelerate the simulation at the cost of precision.
- Conventional extrapolation methods fails with hybrid dynamical systems.
- CHOPtrey extrapolation technique provides a solution for the trade-off between speed-up and accuracy, thanks to:
  - The combination of a prediction and a multi-level context selection.
  - Negligible computational overheads.
- CHOPtrey combination with model splitting and parallel simulation on a hybrid dynamical engine model allows supra-linear speed-up (10 time faster with 5 cores) with acceptable result accuracy.
Decompose signals into morphological components such as polynomial trends, singularities and oscillations

- Allow to adapt detection thresholds
- Improve context assignment

Use of the knowledge of the plant model

- Discard out-of-bound values as nonnegative variables
- Improve the discrimination of cliff behaviors

Use of adaptive communication steps

- Context-based and error-based closed-loop control

Access on the input derivatives of the models

- Provided by FMI for co-simulation
- Improve the extrapolation
Innovating for energy

Find us on:

🌐 www.ifpenergiesnouvelles.com
🐦 @IFPENinnovation