Automating Dynamic Decoupling in Object-Oriented Modelling and Simulation Tools

Alessandro Vittorio Papadopoulos and Alberto Leva

Dip. di Elettronica, Informazione e Bioingegneria
Politecnico di Milano

papadopoulos@elet.polimi.it

April 19^{th}, EOOLT 2013 – @Nottingham, UK

Outline

1. **Model approximation in EOOLT**
2. **Dynamic Decoupling**
3. **Application examples**
4. **A unifying manipulation toolchain**

Outline

Introduction

This work is aimed at

- Introducing some "**model approximations**"
- to **improve simulation efficiency**
- in an **automatic way**
- in the context of Equation-based Object-Oriented Languages and Tools (EOOLT).

What and how to approximate?

EOOLT are generally unsuited for the introduction of approximations

- Approximations cannot be specified at component-level
- Approximations come from properties of the **whole model**
- The typical manipulation toolchain does not allow approximation

What and how to approximate?

EOOLT are generally unsuited for the introduction of approximations

- Approximations cannot be specified at component-level
- Approximations come from properties of the **whole model**
- The typical manipulation toolchain does not allow approximation

Two different manipulations can be performed

1. Acting on the continuous-time equations
   - MOR, Transmission Line Modelling, Neglecting Terms, Linearisation, ...
2. Acting on the discrete-time solution
   - **Dynamic Decoupling**, Co-simulation, ...

Outline

Dynamic Decoupling

Dynamic Decoupling is an approximation framework divided into 2 subsequent phases

1. Structural (system-wide) Analysis
   - Eigenvalue Analysis
   - **Cycle Analysis**
   - ...
2. Decoupled Integration
   - **Mixed-Mode integration**
   - Co-Simulation
   - ...

Cycle Analysis

Consider the state space form of a continuous-time ODE system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$$

that discretised with Explicit Euler and an integration step h yields

$$\mathbf{x_{k+1}} = \mathbf{x_k} + h \cdot \mathbf{f}\left(\mathbf{x_k}, \mathbf{u_k}\right)$$

If a small perturbation is applied to a single state variable $x_\mathbf{k}$ at an equilibrium, two things may happen:

1. the perturbation affects the other state variables, however without in turn re-affecting $x_\mathbf{k}$;
2. the perturbation, after some integration steps, re-affects $x_\mathbf{k}$.

Dependency Graph

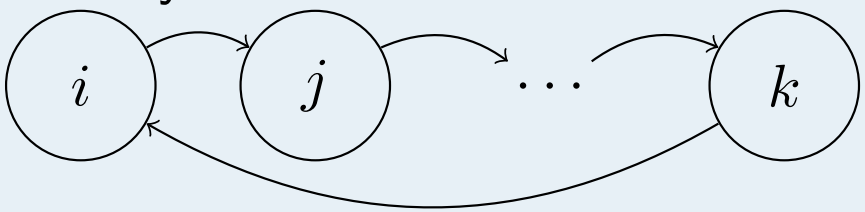
Dependency digraph $G = (N, E)$

- N is the set of the dynamic variables. $|N| = n$
- $E \subseteq N \times N$ formed as

$$e_{i,j} = h \cdot \frac{\partial f_i}{\partial x_j}$$

Definition

A **simple cycle** c is a simple path which starts from a root node $i \in N$ and ends with the same node, without comprising repeated nodes. The length of a simple cycle is $L + 1$, where $L - 1$ is the number nodes in the cycle.

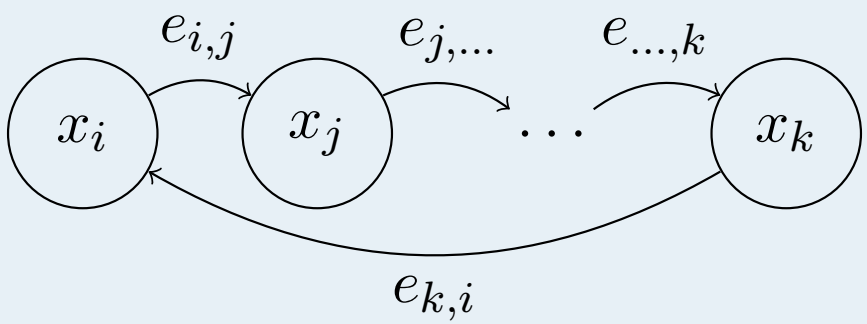


Cycle gain

Definition

The **cycle gain** μ_c of the cycle $c = \langle x_i, x_j, \ldots, x_k, x_i \rangle$

$$\mu_c = \prod_{x_i, x_j \in c} e_{i,j} = h^L \prod_{x_i, x_j \in c} \frac{\partial f_i}{\partial x_j}$$



Cycle gain constraint

For each cycle c in the digraph, find a h such that

$$|\mu_c| \leq \alpha \quad \Rightarrow \quad 0 < h \leq \sqrt[L]{\alpha} \cdot \left| \prod_{x_i, x_j \in c} \frac{\partial f_i}{\partial x_j} \right|^{-\frac{1}{L}}$$

where $\alpha \geq 0$.

Cycle gain constraint

For each cycle c in the digraph, find a h such that

$$|\mu_c| \leq \alpha \quad \Rightarrow \quad 0 < h \leq \sqrt[L]{\alpha} \cdot \left| \prod_{x_i, x_j \in c} \frac{\partial f_i}{\partial x_j} \right|^{-\frac{1}{L}}$$

where $\alpha \geq 0$.

Defining the set of cycles associated with the dynamic variable x_i as

$$\mathfrak{C}_{x_i} = \{c \in \mathcal{C} | x_i \in c\} \subseteq \mathcal{C}$$

An upper bound $\overline{h}_{x_i}$ is associated with each x_i as

$$\begin{aligned} \overline{h}_{x_i} = \max \quad & h \\ \text{s.t.} \quad & h > 0, \\ & 0 < \overline{h}_i \leq \sqrt[L]{\alpha} \cdot \left| \prod_{x_j, x_k \in c} \frac{\partial f_j}{\partial x_k} \right|^{-\frac{1}{L}}, \forall c \in \mathfrak{C}_{x_i}. \end{aligned}$$

Cutting the model

The result of the cycle analysis is twofold

1. We know which are the variables that are mutually coupled
   - Exploit this information to make the simulation parallel
2. Each variable is associated to a time scale

Example

We can cut the model accordingly to the time scales

$$
\begin{aligned}
x_1&: \quad h \leq 0.01 \\
x_2&: \quad h \leq 0.05 \\
x_3&: \quad h \leq 1.00 \\
x_4&: \quad h \leq 10.0 \\
x_5&: \quad h \leq 15.0
\end{aligned}
$$

Cutting the model

The result of the cycle analysis is twofold

1. We know which are the variables that are mutually coupled
   - Exploit this information to make the simulation parallel
2. Each variable is associated to a time scale

Example

We can cut the model accordingly to the time scales

Fast dynamics	x_1:	$h \leq 0.01$
	x_2:	$h \leq 0.05$
Slow dynamics	x_3:	$h \leq 1.00$
	x_4:	$h \leq 10.0$
	x_4:	$h \leq 15.0$

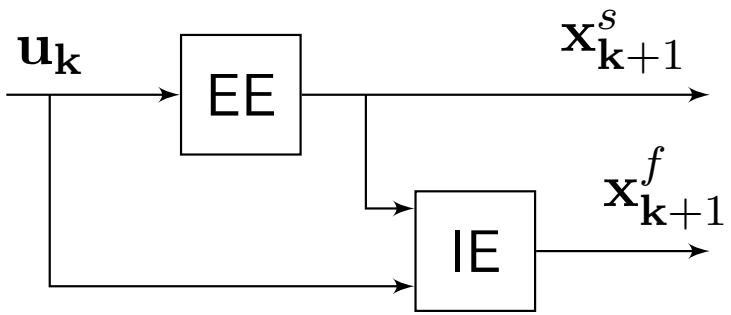
Mixed-mode integration

If the model is cut in two, a mixed-mode integration can be used

$$\begin{cases} \mathbf{x}^s_{\mathbf{k}+1} = \mathbf{x}^s_{\mathbf{k}} + h \cdot \mathbf{f}\left(\mathbf{x}^s_{\mathbf{k}}, \mathbf{x}^f_{\mathbf{k}}, \mathbf{u}_{\mathbf{k}}\right) \\ \mathbf{x}^f_{\mathbf{k}+1} = \mathbf{x}^f_{\mathbf{k}} + h \cdot \mathbf{f}\left(\mathbf{x}^s_{\mathbf{k}+1}, \mathbf{x}^f_{\mathbf{k}+1}, \mathbf{u}_{\mathbf{k}+1}\right) \end{cases}$$

with the result that

- The integration step can be larger
- The implicit method integrates a smaller system (complexity $\mathcal{O}\left(n^3\right)$)
- The fast subsystem takes the slow subsystem solution as input



Outline

DC Motor

Consider a simple DC Motor

$$\begin{cases} L \cdot \dot{I} &= -R \cdot I - k_m \cdot \omega + u(t) \\ J \cdot \dot{\omega} &= k_m \cdot I - b \cdot \omega - \tau(t) \\ \dot{\varphi} &= \omega \end{cases}$$

For a given set of parameters, the cycle analysis leads to

$$\begin{aligned} I: &\quad h \leq 0.060 \\ \omega: &\quad h \leq 0.313 \\ \varphi: &\quad h \leq +\infty \end{aligned}$$

DC Motor

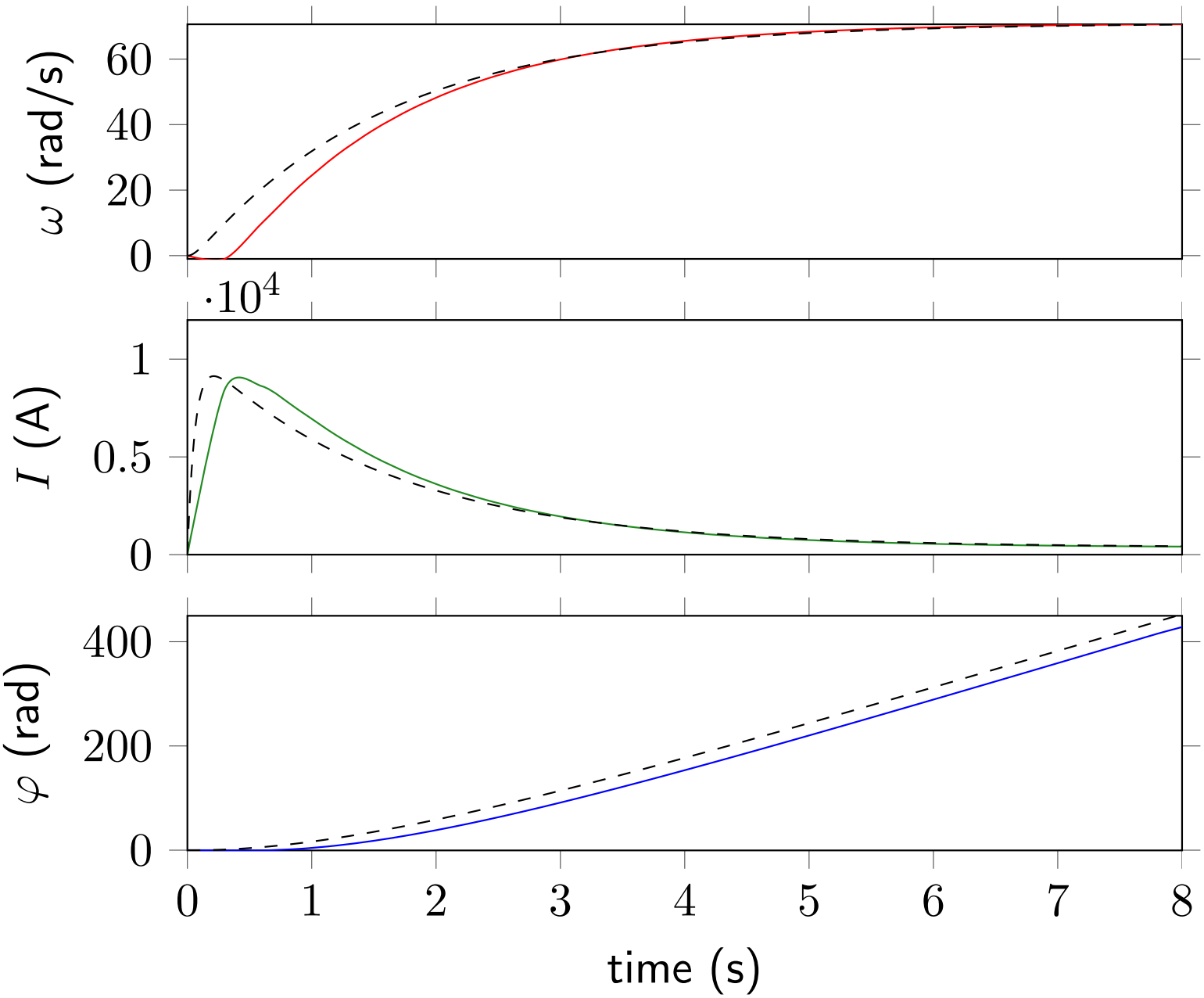
Consider a simple DC Motor

$$\begin{cases} L \cdot \dot{I} &= -R \cdot I - k_m \cdot \omega + u(t) \\ J \cdot \dot{\omega} &= k_m \cdot I - b \cdot \omega - \tau(t) \\ \dot{\varphi} &= \omega \end{cases}$$

For a given set of parameters, the cycle analysis leads to

$$\begin{array}{rl} I: & h \leq 0.060 \\ \hline \omega: & h \leq 0.313 \\ \varphi: & h \leq +\infty \end{array} \quad h = 0.3$$

DC Motor: Simulation results

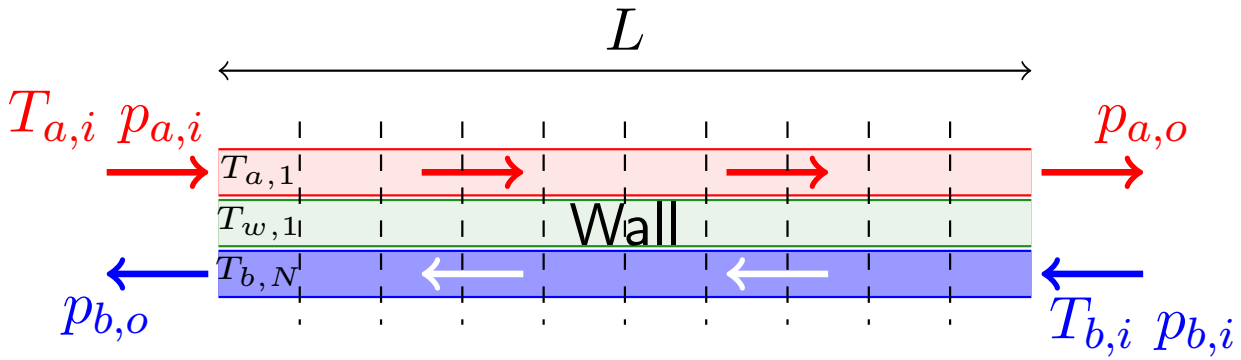


DC Motor: Simulation statistics

	Mixed-mode	BDF	IE	EE[1]
# Steps	**28**	136	**28**	162
# Function ev.	**86**	157	**86**	–
# Jacobian ev.	**2**	3	**2**	–
# Fun. ev. in Jac. ev.	**4**	9	8	–
# Newton iterations	**58**	153	**58**	–
Accuracy	**1.118**	–	1.213	10.043
Sim time	**0.04s**	0.05s	0.06s	**0.04s**

[1] For EE $h = 0.05$ for numerical stability reasons.

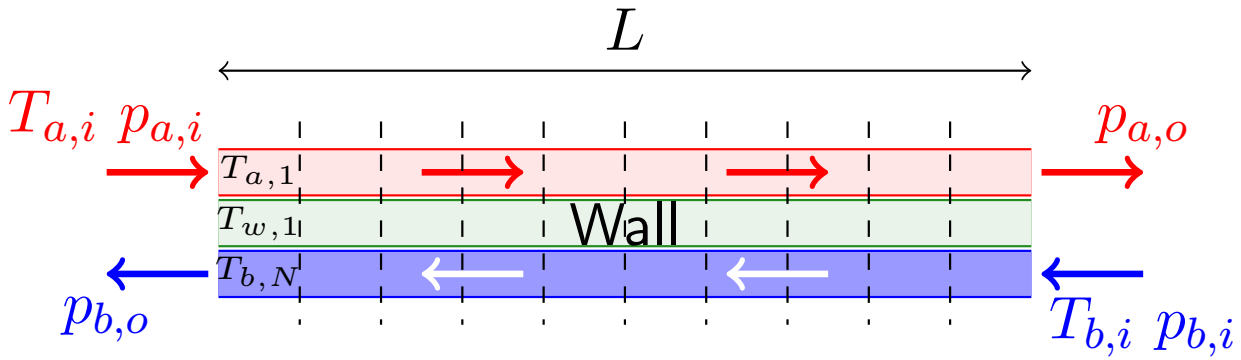
Heat exchanger



We can choose $N = 10$, obtaining a dynamic system of order 30
For a given set of parameters, the cycle analysis leads to

$$\begin{aligned} T_{a,j}: &\quad h \leq 10.383 \\ T_{b,j}: &\quad h \leq 13.327 \\ T_{w,j}: &\quad h \leq 13.658 \end{aligned}$$

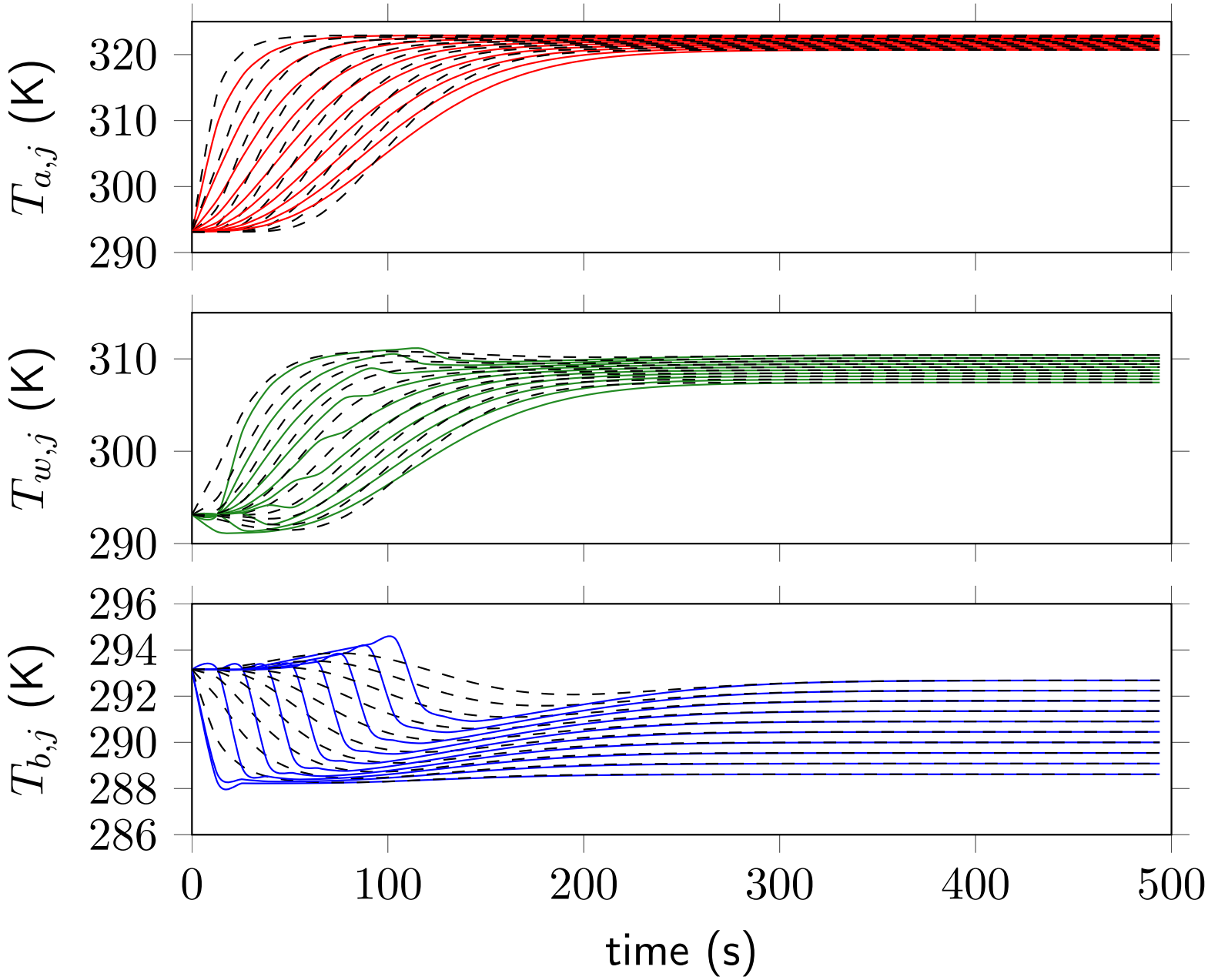
Heat exchanger



We can choose $N = 10$, obtaining a dynamic system of order 30
For a given set of parameters, the cycle analysis leads to

$$\left.\begin{array}{rl} T_{a,j}: & h \leq 10.383 \\ \hline T_{b,j}: & h \leq 13.327 \\ T_{w,j}: & h \leq 13.658 \end{array}\right\} \; h = 13.0$$

Heat exchanger: Simulation results



Heat exchanger: Simulation statistics

	Mixed-mode	**BDF**	**IE**	**EE**[2]
# Steps	**38**	212	**38**	50
# Function ev.	**114**	241	**114**	–
# Jacobian ev.	**2**	4	**2**	–
# Fun. ev. in Jac. ev.	**22**	120	62	–
# Newton iterations	**76**	237	**76**	–
Accuracy	0.017	–	**0.014**	0.059
Sim time	**0.04s**	0.15s	0.06s	0.08s

[2]For EE $h = 10.0$ for numerical stability reasons.

Heat exchanger: changing α

By choosing a smaller value of α, e.g., $\alpha = 0.5$, the simulation is more accurate

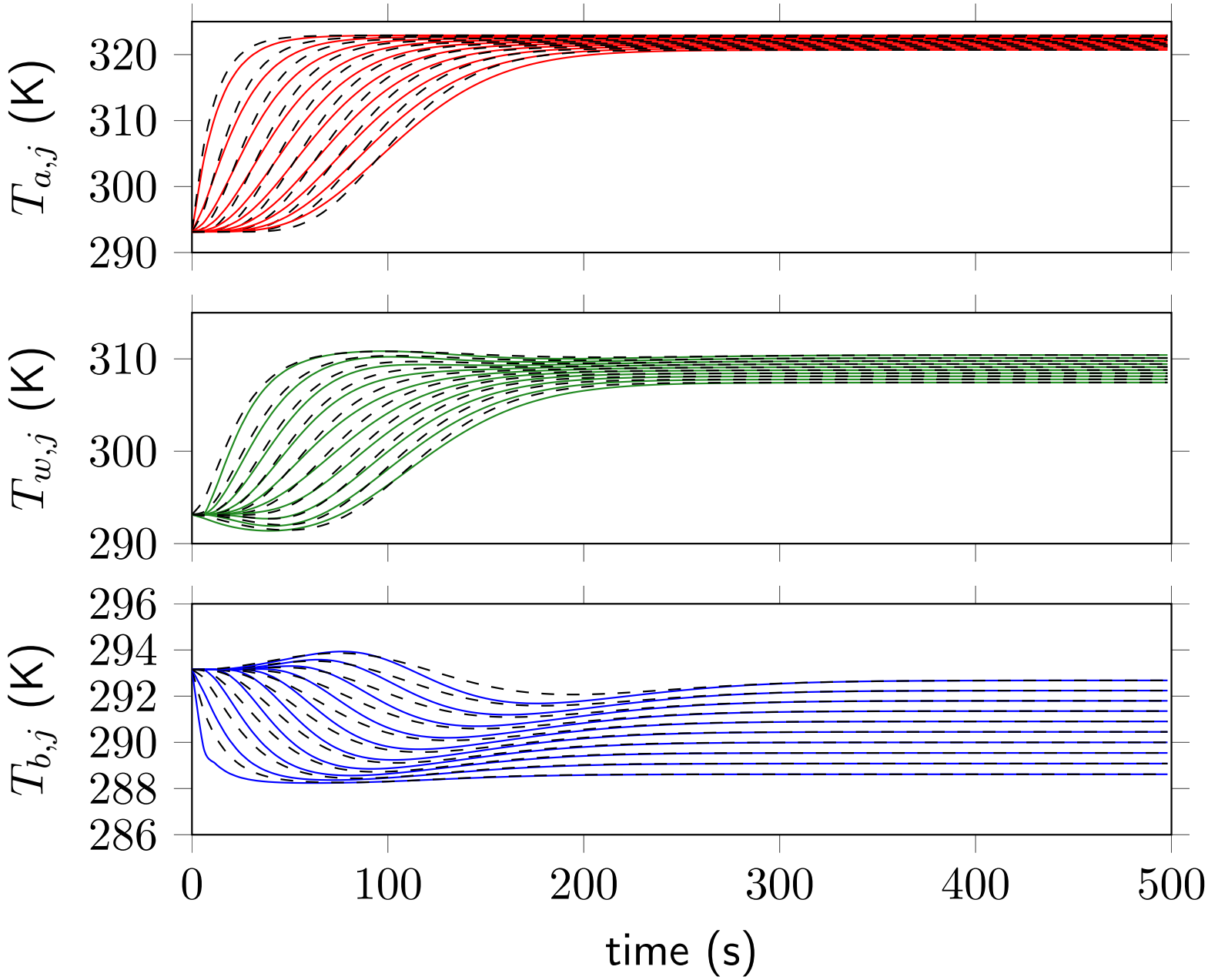
$$\begin{aligned} T_{a,j}: &\quad h \leq 5.191 \\ T_{b,j}: &\quad h \leq 6.664 \\ T_{w,j}: &\quad h \leq 6.829 \end{aligned}$$

Heat exchanger: changing α

By choosing a smaller value of α, e.g., $\alpha = 0.5$, the simulation is more accurate

$$\left.\begin{array}{rl} T_{a,j}: & h \leq 5.191 \\ \hline T_{b,j}: & h \leq 6.664 \\ T_{w,j}: & h \leq 6.829 \end{array}\right\} h = 6.0$$

Heat exchanger: Simulation results



Heat exchanger: Simulation statistics

	Mixed-mode	**BDF**	**IE**	**EE**[3]
# Steps	**83**	213	**83**	100
# Function ev.	**234**	243	243	–
# Jacobian ev.	**4**	**4**	**4**	–
# Fun. ev. in Jac. ev.	**44**	120	124	–
# Newton iterations	**151**	239	160	–
Accuracy	0.011	–	**0.008**	0.018
Sim time	**0.08s**	0.15s	0.16s	0.10s

[3] For EE $h = 5.0$ for numerical stability reasons.

Heat exchanger: Simulation statistics for $N = 30$

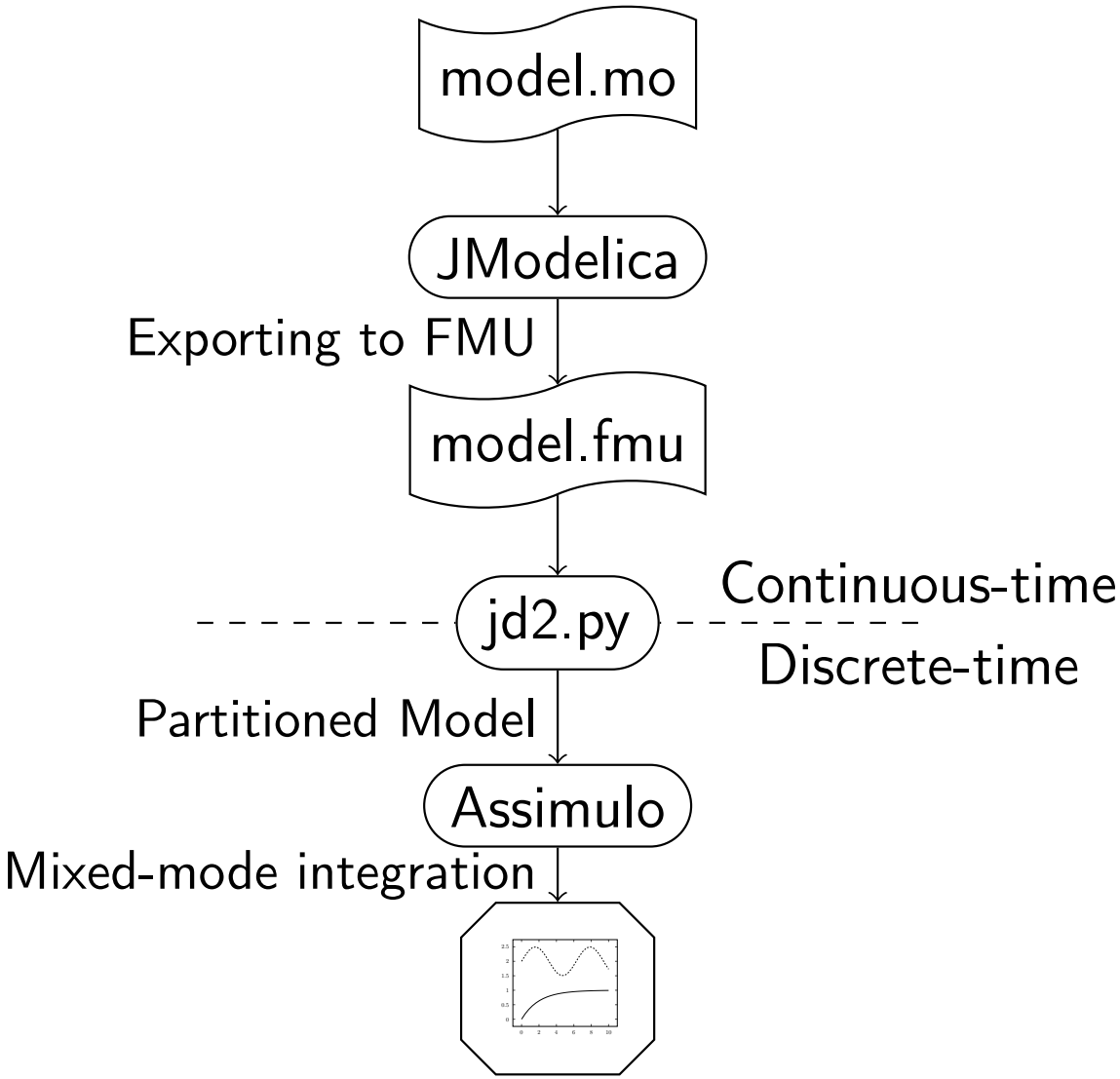
Increasing N to 30, yields a system of order 90.

	Mixed-mode	**BDF**	**IE**	**EE**[4]
# Steps	**125**	304	**125**	250
# Function ev.	**336**	337	345	–
# Jacobian ev.	**6**	**6**	**6**	–
# Fun. ev. in Jac. ev.	**186**	540	546	–
# Newton iterations	**211**	333	220	–
Accuracy	**0.014**	–	**0.014**	0.084
Sim time	0.21s	0.43s	0.41s	**0.20s**

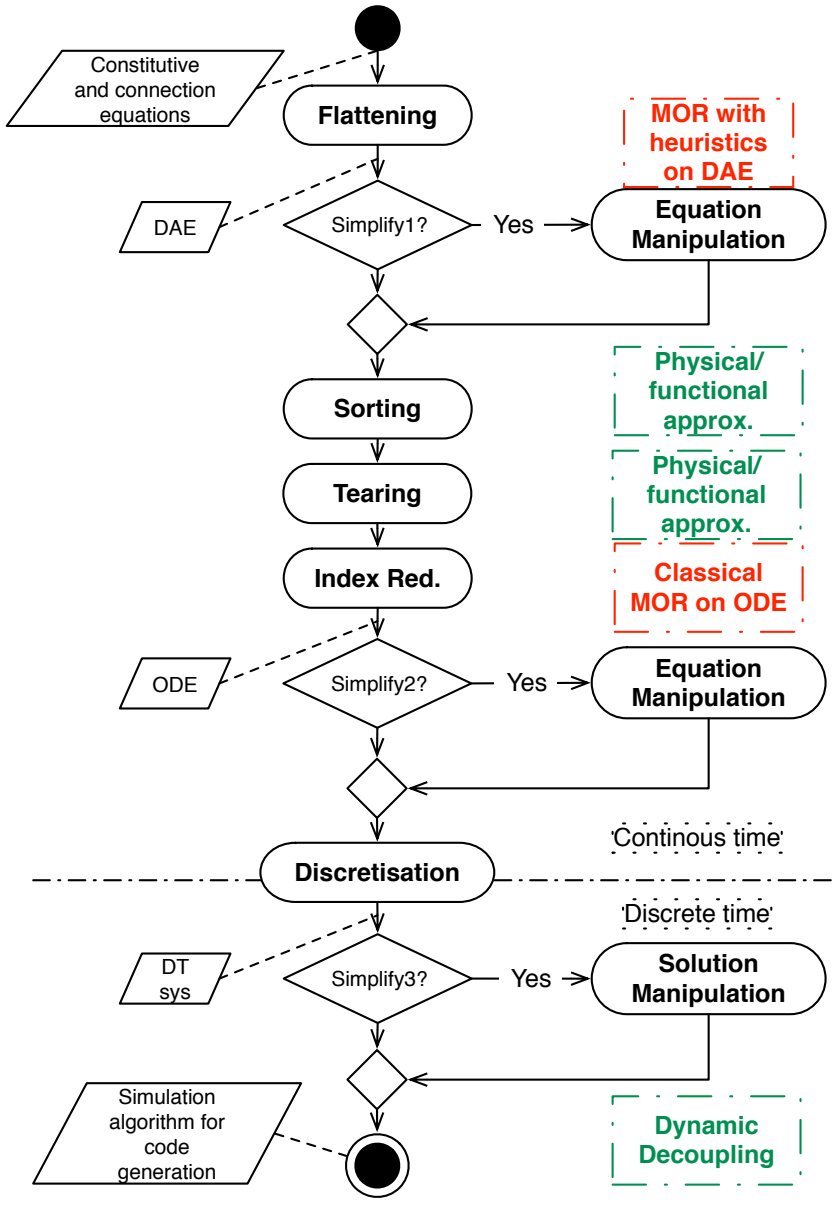
[4] For EE $h = 2.0$ for numerical stability reasons.

Outline

The Dynamic Decoupling toolchain



A unifying manipulation toolchain



Conclusion and future work

In this work we proposed

- A novel framework for the introduction of model approximation in EOOLT
- An automatic technique for improving simulation efficiency, i.e., **Dynamic Decoupling**
- The integration of the proposed technique in a classical Modelica translator

Future work

- Cycle analysis for parallel simulation
- "Separability indices" to better automate the decoupling process
- Co-simulation frameworks
- More complex models, e.g., smart grids

Acknowledgements

Prof. F. Casella and **Prof. J. Åkesson** for the useful discussions and constructive criticisms.

Thank you for your attention

Questions, suggestions, comments are welcome!

Alessandro Vittorio Papadopoulos
Politecnico di Milano, Italy
(papadopoulos@elet.polimi.it)