Automating efficiency-targeted approximations in modelling and simulation tools: dynamic decoupling and mixed-mode integration

Alessandro Vittorio Papadopoulos[1]*and Alberto Leva[2]

[1]Lund University, Department of Automatic Control, Lund, Sweden

[2]Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Milano, Italy

July 15, 2014

Abstract

Modelling and simulation nowadays permeate virtually any engineering activity, requiring tools capable of managing complex models efficiently. Nonetheless, whereas modern modelling languages and tools allow to *construct* such models even on lightweight platforms (e.g., a laptop), the same is not true when it comes to *numerically integrate* those models. For the latter purpose, modellers usually pursue efficiency by resorting to approximation and reduction techniques. However, such techniques are unnatural to include in modelling and simulation tools. This is particularly true with *object-oriented* ones, which on the other hand are the most interesting for dealing with complexity from the model construction viewpoint. This paper presents a novel approximation technique that can be easily included in modelling and simulation tools, and relates the proposal to literature alternatives so as to evidence its peculiarities. An extended manipulation toolchain is also proposed, allowing for the introduction of other (classical) efficiency-targeted approximation techniques, within a unified framework. Some application examples illustrate the achieved advantages and motivate the major design choices from an operational viewpoint.

Keywords: dynamic decoupling, mixed-mode integration, modelling and simulation tools

Received: 25-Sep-2013, **Revised:** 06-Feb-2014 (Major).
Received: 20-Mar-2014, **Revised:** 18-Jun-2014 (Minor).
Received: 04-Jul-2014, **Accepted:** 14-Jul-2014.

1 Introduction and motivation

Nowadays, the availability of powerful and flexible model creation, maintenance, and simulation tools, is very important at virtually any stage of engineering projects. Even more important is however the integration of all the mentioned activities in a unified *toolchain*, allowing the engineer to take full profit of them without undue overheads, as witnessed by the interest in the emerging field of Modelling and Simulation (M&S) [1].

*Corresponding author: Alessandro Vittorio Papadopoulos, Lund University, Department of Automatic Control, Ole Römers väg 1, 22363, Lund, Sweden. Email: alessandro.papadopoulos@control.lth.se

Modern M&S toolchains already take care of the transition between a model expressed, e.g., in terms of equations, and the code needed for its simulation. However, the evolution itself of *modelling* tools is at present resulting in challenges for *simulation* ones. In fact, it is now possible to construct very complex models on lightweight platforms, like a laptop, but the simulation of those models on the same platform may prove too demanding, unless some "optimisation" (the reason for the quotes will emerge soon) is introduced in the equations-to-code path. In other words, computational resources often become the bottleneck of engineering M&S tools, so that making toolchains capable of *transparently* obtaining an efficient numerical integration of complex models, is a key issue.

Quite often, simulation efficiency is obtained by introducing suitable approximations (in the broadest sense of the term), which is precisely what available M&S toolchains do *not* allow [2]. This paper is part of a long-term research aimed at filling this gap.

The presented work refers to the context of Equation-based Object-Oriented (EOO) M&S tools, where the gap appears with particular evidence, for (at least) two main reasons. A first intuitive one is that in EOO languages managing model complexity at the component level has practically no cost, thereby exacerbating the possible computational bottleneck. A second and more subtle reason is that many approximation techniques aimed at numerical efficiency are based on structural properties of the *complete* model, not of its components, while the EOO paradigm explicitly relies on the idea of writing the component models independently of how they will be connected. More specifically, this paper addresses a particularly effective approximation technique aimed at simulation efficiency, namely the Dynamic Decoupling (DD) one, and its integration in a typical EOO M&S toolchain.

The rest of the paper is organised as follows. Section 2 reports a brief review of related work, to contextualise the presented research and evidence the proposed advances. Section 3 describes a technique to analyse a model in Differential and Algebraic Equation (DAE) form, so as to evidence the time scales of its dynamics, in both the linear and the nonlinear case, providing the engineer with easily interpreted information for clustering said time scales in a view to possibly partitioning the model for the numerical integration phase. That information takes the form of some suitably defined *separability indices*, to which Section 4 is devoted. Section 5 then deals with how such partitions can be exploited with different simulation technologies. Section 6 illustrates how the presented functionalities can be integrated in a typical EOO M&S toolchain, referring to a Modelica translator as a representative case. Some application examples are reported in Section 7, while Section 8 draws some conclusions and sketches out future research developments.

2 Related work and contribution

To contextualise DD in the field efficiency-targeted approximation techniques, with specific emphasis on their applicability and convenience in EOO M&S tools, we start with a few general remarks on the typical M&S toolchain, thereby also motivating some statements of Section 1.

Referring to Figure 1, the EOO chain of operations, from component equations to simulation code, can be broadly divided into two parts. The first one, which we call *acting on the continuous-time equations*, transforms the DAE system coming from the flattening phase – in which all the equations of the single components are collected in a single monolithic system according to the connections specified by the modeller – into a causal ODE one. This can be done without altering the equations' semantic, with techniques such as the Tarjan algorithm, alias elimination, index reduction, and so forth [3]. It can also be done by accepting some semantic alteration in exchange

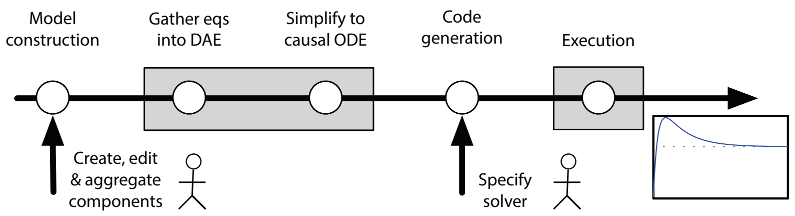

Figure 1: The typical EOO M&S toolchain.

for an efficiency improvement, the major techniques being Model Order Reduction (MOR) [4] or scenario-based [5] approximations.

The second part of the EOO toolchain, which we call *acting on the discrete-time solution*, consists of taking the ODE model as the basis to generate routines that linked to the numeric solver of choice, result in the simulation code. Assuming that this is done "correctly", i.e., preserving numerical stability, here too two ways of operating can be distinguished. The first does not alter the solution semantic, and the chosen discretisation method is applied as is: thus, errors in the solution only come from the inherent "imperfection" of that method. The second way conversely alters the semantic of the discrete-time system, by deliberately deviating from the natural application of the discretisation method. Notice that most of the co-simulation techniques fall in this class naturally [6, 7].

In this work we concentrate on this last type of operation, where a technique of election is DD [8, 9]. For the purpose of this section, suffice to say that DD aims at partitioning the monolithic system into submodels, based on a time-scale separation. The method is a computational approximation of the original system that makes the submodels mutually independent within an integration step, by exploiting structural properties of the system. This allows good performance improvements, especially in some specific context, e.g., process control [8] or multibody system dynamics [10]. The method is particularly of interest – as will be detailed better in Section 3 – because it can be divided into two well separated phases: an analysis part performed on the overall model, and a simulation part that can either be monolithic, or make use of co-simulation.

2.1 Alternative Approaches

To motivate our focus on DD, we now briefly consider the major possible alternatives. As already stated, among the techniques that act on the continuous-time equations, MOR ones are the most adopted, and there exists a vast literature on the matter. MOR is based on the idea of approximating a certain part of a high-dimensional state space of the original system with a lower-dimensional space, performing a projection. Roughly speaking, the main differences among MOR techniques come from the way this projection is performed. Most techniques have been developed for linear systems [4], and this hampers their application to the typical *engineering* EOO models, that are often both high-dimensional and nonlinear.

In the literature, some extension to the nonlinear case are present, e.g., based on linearisation or Taylor expansion [11], or bilinearisation [12], as well as functional Volterra series expansion [13], followed by a suitable projection. Other interesting extensions are those based on Proper Orthogonal Decomposition (POD), that produce approximate truncated balanced realisations [14], often exploiting POD to find empirical [15] and (for switched systems) generalized [16] gramians. However, for the former extension, practical implementations typically stick to quadratic expansions, limiting simplification capabilities. As for the latter, the cost of evaluating the projected nonlinear operator is often very high, reducing computational performance.

Recently, works specifically dealing with the reduction of EOO model appeared [5, 17]. The main idea is that one can define some operation to be performed on the nonlinear system, e.g., neglect a "term", linearise a part of the model, and so on, and use some ranking metrics to identify *a priori* the "best" (single) such manipulation that can be done. Apparently, the limit of this approach is that ranking all the possible manipulation combinations is not feasible. Moreover, there is no guarantee that performing the manipulations in the ranked order will bring to any optimum. Another problem is the high cost of generating the reduced order models, due to necessity of computing "snapshots" in the time domain, which in turn requires many simulations of the original nonlinear system. Furthermore, this approach is scenario-based, i.e., the simplified model is guaranteed to be good only for a set of initial conditions, a set of inputs, and a time span. If the scenario is changed, the overall manipulation must be performed again.

A last interesting approach – related to the basic idea of DD – is the one of Transmission Line Modelling (TLM) [18]. The basic idea of TLM is to model a system so that components can be somehow numerically isolated, and each can solve its own equations independently. This is achieved by replacing capacitive components (e.g., volumes in hydraulic systems) with transmission line elements with a propagation time corresponding to one simulation time step. At the cost of some internal delays, the result is a physically accurate description of wave propagation in the system. However, the analyst has to deliberately act on the model, based on his/her intuition. This work conversely aims at having decoupling emerge from an automated analysis of the model.

2.2 A Brief Comparison

Based on the previous discussion, we now point out the advantages of the proposed technique with respect to the analysed alternatives. In comparison with MOR, our proposal does not alter the state vector, nor does it involve base changes in the state space. Also, instead of attempting to simplify the model in a view to monolithic solution, we go exactly in the opposite direction, as the model is not reduced but *partitioned*, allowing for parallel simulation. This can in turn be exploited in two ways. One is to ease a monolithic solution, in some sense adapting the model to the used (single solver) architecture. The other is to conversely tailor the solution architecture to the model *as analysed and partitioned by the method*; this can be used to fruitfully employ parallel simulation, or even co-simulation. Finally, the proposed method is naturally keen to be applied in a nonlinear context.

With respect to scenario-based approximations, the most computing-intensive part of the proposal (as will be explained later on) is simply not scenario-based: information related to the considered *scenarii* come into play only at a later stage, and this separation results in lightening the computing effort. Furthermore, the proposal does not alter the model equations, thus being less exposed to the possible unpredictable effects of local modifications at the overall system level.

3 Cycle analysis

Modern EOO M&S tools typically start from a DAE system, and transform it into an ODE one [3]. Notice that it is not necessary to have the ODE system in a symbolic form, yet the numerical Jacobian is sufficient for performing the overall analysis. This allows to handle both the cases in which semi-explicit DAE are the result of the symbolic manipulations, and the cases in which only a numerical description of the model is available, e.g., thermo-hydraulic systems using steam tables. Therefore, we can consider as our starting point, without loss of generality, the generic ODE system

$$\dot{\mathbf{x}}_{CT}(t) = \mathbf{f}(\mathbf{x}_{CT}(t), \mathbf{u}_{CT}(t)) \tag{1}$$

where $\mathbf{x}_{CT} \in \mathbb{R}^{n_{CT}}$ is the state vector, and $\mathbf{u}_{CT} \in \mathbb{R}^{m_{CT}}$ the input vector. The core idea of Cycle Analysis (CA) is to obtain from the discretisation of (1) a directed graph representing the mutual influence among the state variables along the integration steps, and then to compute quantities that generalise – details follow – the idea of "time constants" for the linear case.

The mentioned discretisation of (1) can be done with any explicit, fixed-step method, which we call here the "probe method", as it is just functional to the analysis, and in no sense constrains the subsequent simulation phase to use it. The corresponding discrete-time system, h denoting the time step, can be written as

$$\mathbf{x}_{k+1} = \mathbf{F}_{\mathcal{N}}(\mathbf{x}_k, \mathbf{u}_k, h) \tag{2}$$

where $\mathbf{x}_k^T = \begin{bmatrix} x_1(k) & x_2(k) & \cdots & x_n(k) \end{bmatrix}^T = \begin{bmatrix} \mathbf{x}_{CT}^T(kh) & \xi^T(kh) \end{bmatrix}^T \in \mathbb{R}^n$, with $n \geq n_{CT}$ is the discrete-time state, vector ξ possibly containing additional state variables to accommodate for multi-step methods (see [3, Chapter 4]), while the form of function $\mathbf{F}_{\mathcal{N}}(\cdot,\cdot,\cdot)$ depends on the particular numerical integration method $\mathcal{N}$.

The dependency directed graph (or digraph) G, straightforwardly built from the structure of (2), is formally defined as

$$G = (N, E), \qquad N = \{1, \ldots, n\}, \quad E = \{e^{i,j}\} \subseteq N \times N. \tag{3}$$

The nodes of G are associated with the discrete-time model state variables, while its edges are characterised by a source node, a destination node, and a weight, defined by the operators

$$\varsigma\left[e^{i,j}\right] := i, \quad \delta\left[e^{i,j}\right] := j, \quad \rho\left[e^{i,j}\right] := \frac{\partial F_i}{\partial x_j}. \tag{4}$$

We can now give some definitions.

Definition 1. *A* path *p of length L in a digraph $G = (N, E)$ is an ordered sequence of L edges, where the destination node of each edge is the source node of the following one in the sequence. Formally,*

$$\begin{aligned} &p := \langle e_1, e_2, \ldots, e_L \rangle, \quad \text{with } e_i \in E, \quad \forall i \in \{1, \ldots, L\}, \\ &\text{with } \delta\left[e_i\right] = \varsigma\left[e_{i+1}\right], \quad \forall i \in \{1, \ldots, L-1\}. \end{aligned}$$

A path can be also denoted by means of the ordered sequence of touched nodes, i.e.

$$p = \langle \varsigma\left[e_1\right], \varsigma\left[e_2\right], \ldots, \varsigma\left[e_L\right], \delta\left[e_L\right] \rangle.$$

Definition 2. *A path with no repeated nodes is called a* simple path *(or* walk*).*

Definition 3. *A* simple cycle *c of length L exists in a digraph $G = (N, E)$ iff*

1. *there exists a simple path $\langle e_1, e_2, \ldots, e_{L-1} \rangle$,*
2. *there exists one edge e_L from $\delta\left[e_{L-1}\right]$ to $\varsigma\left[e_1\right]$.*

Adopting the same notation used for paths, a simple cycle can be denoted as

$$c = \langle \varsigma\left[e_1\right], \varsigma\left[e_2\right], \ldots, \varsigma\left[e_{L-1}\right], \delta\left[e_{L-1}\right], \varsigma\left[e_1\right] \rangle,$$

i.e., by listing the ordered sequence of the touched nodes. Since in the following we shall only refer to simple cycles, the terms "cycle" and "simple cycle" will be used interchangeably.

Definition 4. *The* cycle gain $\mu_c(h)$ *of a cycle* c *is defined as*

$$\mu_c(h) = \prod_{e_i \in c} \rho\left[e_i\right]. \tag{5}$$

3.1 An explanatory example

Let us consider the continuous-time linear time-invariant dynamic system

$$\dot{\mathbf{x}}_{CT} = A\mathbf{x}_{CT} = \begin{bmatrix} -1 & 0.5 & 0 \\ 0.5 & -1.5 & 0.5 \\ 0 & 0.5 & -1 \end{bmatrix} \mathbf{x}_{CT},$$

Using as probe method the Heun's one [3], the corresponding discrete-time system (2) is

$$\mathbf{x}_{k+1} = \mathbf{F}_{\text{Heun}}(\mathbf{x}_k, h), \quad \mathbf{F}_{\text{Heun}}(\mathbf{x}_k, h) = \left(I_{3\times3} + Ah + \frac{(Ah)^2}{2} \right) \mathbf{x}_k, \tag{6}$$

where $I_{3\times3}$ is a 3×3 identity matrix, and $\mathbf{x}_k = \mathbf{x}_{CT}(kh)$. Therefore, the dependency graph G has the weight matrix

$$W = \frac{\partial \mathbf{F}_{\text{Heun}}}{\partial \mathbf{x}} = I_{3\times3} + Ah + \frac{(Ah)^2}{2} = I_{3\times3} + \frac{h}{8} \begin{bmatrix} 5h-8 & 4-5h & h \\ 4-5h & 11h-12 & 4-5h \\ h & 4-5h & 5h-8 \end{bmatrix}, \tag{7}$$

and is completely connected, see Figure 2.

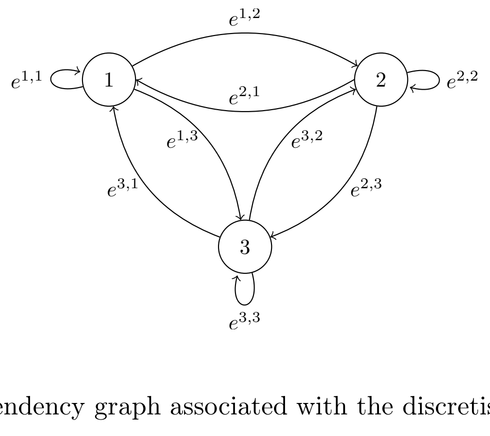

Figure 2: Dependency graph associated with the discretised system (6).

In this case, the set of simple cycles $\mathcal{C} = \{c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8\}$ in G, and the corresponding cycle gains, are

$$\begin{array}{llll}
c_1 = \langle e^{1,1} \rangle, & \mu_{c_1}(h) = \dfrac{5h^2}{8} - h + 1, & c_2 = \langle e^{2,2} \rangle, & \mu_{c_2}(h) = 1 + \dfrac{h}{8}(11h - 12), \\
c_3 = \langle e^{3,3} \rangle, & \mu_{c_3}(h) = \dfrac{5h^2}{8} - h + 1, & c_4 = \langle e^{1,2}, e^{2,1} \rangle, & \mu_{c_4}(h) = \dfrac{h^2}{64}(4 - 5h)^2, \\
c_5 = \langle e^{1,3}, e^{3,1} \rangle, & \mu_{c_5}(h) = \dfrac{h^4}{64}, & c_6 = \langle e^{2,3}, e^{3,2} \rangle, & \mu_{c_6}(h) = \dfrac{h^2}{64}(4 - 5h)^2, \\
c_7 = \langle e^{1,2}, e^{2,3}, e^{3,1} \rangle, & \mu_{c_7}(h) = \dfrac{h^4}{512}(4 - 5h)^2, & c_8 = \langle e^{1,3}, e^{3,2}, e^{2,1} \rangle, & \mu_{c_8}(h) = \dfrac{h^4}{512}(4 - 5h)^2.
\end{array}$$

3.2 The analysis technique

The analysis goal is to (automatically) recognise the presence in the model of different time scales, and cluster the state variables accordingly, based on a convenient interpretation of the cycle gains as per Definition 4. For this interpretation, consider system (2) at an asymptotic stable equilibrium, and suppose to apply a small impulsive perturbation to one state variable x_i. In the so provoked transient two situations can occur:

- if in G, there is no cycle involving node i, the perturbation affects the other state variables, but without re-affecting x_i;
- if in G there exists at least one cycle involving node i, the perturbation re-affects x_i after some integration steps.

In the first case, the (probe) integration method cannot introduce numerical instability, while this is possible in the second case, if the perturbation undergoes a sufficient amplification along at least one of the involved cycles. Said amplification is quantified by the corresponding cycle gain, allowing to conjecture that the perturbation vanishes if all the gains of the involved cycles are in magnitude less than a certain $\underline{\mu}$, while instability arises if at least one of them exceeds in magnitude a certain $\overline{\mu} > \underline{\mu}$.

Consider now an ODE system at a certain stable equilibrium. Apparently, if the state remains near enough to that equilibrium for the linearisation of the original system to be reliable enough, there exists one boundary value of h between a stable and an unstable behaviour of the discrete-time solution. With explicit methods, also, instability notoriously originates from dynamics that are too fast with respect to the integration step. And since the cycle gains depend on h, if an unstable behaviour is observed, one can legitimately say that the state variables involved in the cycles with the "excessive" amplification, are evolving with a time scale that is "fast" with respect to h.

The qualitative considerations above form the basis of the CA procedure, that can be summarised as follows.

1. Select an explicit fixed-step integration method as the probe one (the choice is discussed later on).
2. Discretise the system with that method.
3. Construct the corresponding digraph.
4. Find the set $\mathcal{C}$ of all the (simple) cycles.

5. Express the cycle gains with (5).
6. Construct the set of inequalities
$$|\mu_c(h)| \leq \alpha, \qquad \forall c \in \mathcal{C},$$
where $\alpha \in (0,1)$ is the (single) CA parameter, discussed in the following.
7. Solve each single inequality for h, which associates each cycle with a value for h producing "low enough" a gain magnitude.
8. Associate each x_i with the largest h value, $\overline{h}_{x_i}$ to name it, among those found for the cycles involving x_i, which is formally expressed as
$$\begin{aligned}\overline{h}_{x_i} = \max \quad & h \\ \text{s.t.} \quad & h > 0, \\ & |\mu_c(h)| \leq \alpha, \quad \forall c \in \mathfrak{C}_{x_i}.\end{aligned}$$

The result of the analysis is that each state variable is associated with a time scale, since if α was correctly chosen (in a sense to be discussed) and h is set below the so obtained $\overline{h}_i$, then the discretised ODE equation that computes $x_{i,k+1}$ cannot be responsible for instabilities.

Ranking the state variables by $\overline{h}_i$ can thus provide the basis for their clustering, and for decoupled integration. Before entering that subject, it is however convenient to exemplify how the procedure above can be implemented.

3.3 A possible implementation

Taking the Explicit Euler (EE) method as the probe one, the discretised system is
$$\mathbf{x}_{k+1} = \mathbf{x}_k + h \cdot \mathbf{f}\left(\mathbf{x}_k, \mathbf{u}_k\right). \tag{8}$$

Thus, the edge weights of G take the form
$$\rho\left[e^{i,j}\right](h) = \begin{cases} 1 + h \cdot \dfrac{\partial f_i}{\partial x_i} & \text{if } i = j, \\ h \cdot \dfrac{\partial f_i}{\partial x_j} & \text{if } i \neq j. \end{cases}$$

The cycle gains (5) can be computed as
$$\mu_c(h) = \begin{cases} 1 + h \cdot \dfrac{\partial f_i}{\partial x_i} & \text{if } L = 1 \text{ and } \dfrac{\partial f_i}{\partial x_i} \leq 0, \\ h^L \displaystyle\prod_{e^{i,j} \in c} \dfrac{\partial f_i}{\partial x_j} & \text{otherwise,} \end{cases} \tag{9}$$

resulting in the constraint set
$$|\mu_c(h)| \leq \alpha \quad \Rightarrow \quad \begin{cases} 0 < h \leq (1+\alpha) \left|\dfrac{\partial f_i}{\partial x_i}\right|^{-1} & \text{if } L = 1 \text{ and } \dfrac{\partial f_i}{\partial x_i} \leq 0, \\ 0 < h \leq \sqrt[L]{\alpha} \cdot \left|\displaystyle\prod_{e^{i,j} \in c} \dfrac{\partial f_i}{\partial x_j}\right|^{-\frac{1}{L}} & \text{otherwise.} \end{cases} \tag{10}$$

Notice that in the case that $\partial f_i/\partial x_i = 0$, (10) leads to the constraint that $h < \infty$, which is correct since the variable has no dynamic. On the other hand, if $\partial f_i/\partial x_i > 0$ the constraint $|1 + h \cdot \partial f_i/\partial x_i| < \alpha$ cannot be satisfied for any value of $\alpha \in (0, 1]$. It is thus necessary to compute the cycle gain as specified in (9) in order to obtain a sensible upper bound for the integration step.

Moreover, Equation (10) shows that there is a direct relationship between the maximum magnitude of a cycle gain α and the integration step h. The same result can be obtained (in a more complex form) for virtually any explicit method. Recalling that for the generic explicit method, it is possible to make all the eigenvalues lie in the region of numerical stability, by choosing a sufficiently small value of h, the above remark implies that stability can always be achieved with sufficiently small value of α.

The presented CA implementation is summarised by the pseudo-code in Algorithm 1. The functions `pop` and `append` applied to a queue mean that the first element of the queue is extracted, and the the input elements are added to the rear of the queue respectively. Notice that the problem of finding all the cycles in a directed graph is a widely studied problem, and different algorithms have been proposed in the literature [19, 20].

Algorithm 1 Algorithm to detect all the cycles in the dependency digraph.

```
function GET_PATH_FROM_A_TO_B(graph, a, b)
    paths = $\emptyset$;                                        // Initialise two empty lists
    q = $\emptyset$;
    q.append(a);
    while q $\neq \emptyset$ do
        path = q.pop();                                  // Get the first element of q
        final = path(end);
        if final == b and length(path)>1 then
            paths.append(path);
        end if
        for e $\in$ graph.successors(final) do
            if e $\notin$ path(2:end) then
                next = path;
                next.append(e);
                q.append(next);
            end if
        end for
    end while
    return paths;
end function

function GET_CYCLES(graph)
    cycles = $\emptyset$;                    // Initialise an empty list of dependency cycles
    nodes = graph.nodes();                 // Get the list of all the nodes in the graph
    while length(nodes)$\neq \emptyset$ do
        n = nodes.pop();                               // Get the first node in the list
        paths = get_path_from_a_to_b(graph,n,n);
        if paths is not empty then
            cycles.append(paths);
        end if
        graph.remove_node(n);                    // All the cycles involving n are detected
    end while
    return cycles
end function
```

Of course the analysis could use any explicit fixed-step probe method other than EE, which however has two main advantages. First, it yields *explicit* constraints with α as the sole parameter. This makes it computationally affordable to analyse the model at hand for different values of α, i.e., to conduct a *parametric* separability analysis (this is shown in Section 7). Then, it results in the maximum sparsity degree of the discrete-time model dynamic matrix, in apparent favour of an efficient cycle detection. In the example of Section 3.1, should one use EE instead of the Heun's method, the resulting dynamic matrix would in fact be

$$W = \frac{\partial \mathbf{F}_{\mathrm{EE}}}{\partial \mathbf{x}} = I_{3\times 3} + Ah = I_{3\times 3} + \frac{h}{2}\begin{bmatrix} -2 & 1 & 0 \\ 1 & -3 & 1 \\ 0 & 1 & -2 \end{bmatrix},$$

which is remarkably more sparse than (7). The EE method thus provides reasonably conservative stability regions with a low computational effort, and is thus chosen here as the probe one.

3.3.1 Discussion

The possible implementation of CA presented in the last section allows for a parametric analysis on any set of scenarii in an analytical manner. In particular, one could explore the set of interest offline, on the basis of the considered operating point, and then obtain more than one partition, each one to be used for the scenario to which it pertains.

When considering non-linear systems, it may happen that some simulation traverse operating points that are so different to not belong to any single scenario. In such a case, one could take the examined set of scenarii, and perform a partition that only preserves the surviving cuts, representing the worst case scenario. It may even turn out that for some simulation the only way to go is to keep the model not partitioned, but the presented method may be a mean to detect and motivate such a necessity.

A possible issue that may raise using CA is that the index reduction techniques [21], that are present for example in the Modelica compilers, acting on the DAE system may adjust the state variable selection at runtime. As mentioned before, the proposed technique considers only an offline analysis of the structural properties of the system aimed at finding a partition, and does not take into account possible adjustments of the state variables selection at runtime. Although it is in accordance to many of the co-simulation techniques present in the literature [6, 22], this is apparently an issue to be addressed in future works.

To end this section, some words are also in order on the two major techniques that can provide an alternative about time scale analysis, namely eigenvalue [23] and Lyapunov exponent analysis [24, 25].

Concentrating for brevity on the former, as the latter is its natural extension to the nonlinear case, we can intuitively observe that in general a value of α can be found so that the CA constraints also guarantee stability, as the eigenvalue ones conversely do by construction [9]. Despite this apparent inferiority, nonetheless, CA has also two relevant strong points. First, the CA bounds are inherently related to the *interactions* among state variables, while eigenvalue-based ones are not. Second, the number of the CA bounds is that of the cycles, not the state variables, thus yielding a more fine-grained structural information. In other words, with eigenvalue analysis one just observes the system mode by mode, while CA explicitly evidences the mode couplings.

Incidentally, in the linear case, CA provides exactly as many constraints as the system order in the particular case of a triangular system with real eigenvalues, and in this case, not surprisingly, the value of α discriminating stability from instability is the unity.

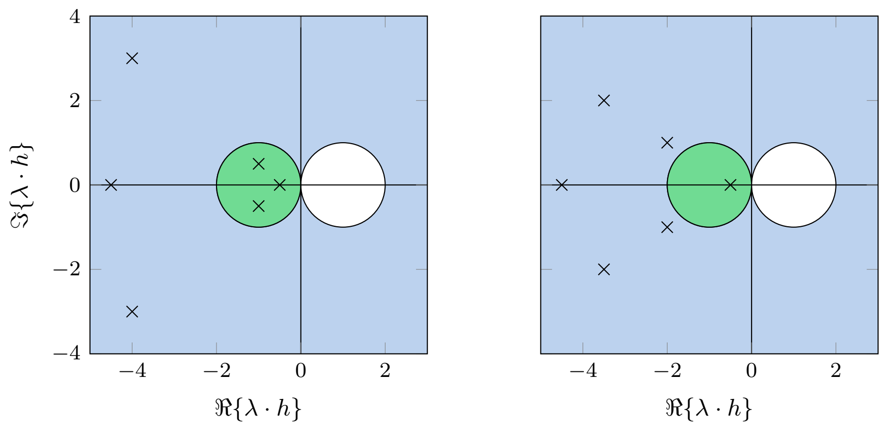

Figure 3: Two cases of linear systems with the same stiff ratio.

4 Separability indices

The result of CA is that we associated a time scale, expressed in terms of $\overline{h}_{x_i}$, with each dynamic variable x_i. Based on those time scales some synthetic indices can thus be defined for deciding how to partition the original model into weakly coupled submodels. Such indices can also be interpreted so as to extend the idea of "stiffness", in the same way as CA was shown to yield more decoupling-related information than eigenvalue analysis.

We start the discussion by considering the classical eigenvalue-based stiffness indicator, i.e., the *stiffness ratio* [3].

Definition 5 (Stiffness ratio)**.** *The* stiffness ratio σ_R *is defined as the ratio between the absolute largest real part and the absolute smallest real part of any eigenvalue:*

$$\sigma_R = \frac{\max_i |\Re\{\lambda_i\}|}{\min_i |\Re\{\lambda_i\}|}.$$

Apparently, σ_R is defined for a linear system, and quantifies the span from the fastest to the slowest time scale. This is useful to decide on the usefulness of an integration method for stiff systems on the *entire* model, but gives no information on how many "time scale clusters" exist, nor on which state variables they contain.

To exemplify, consider Figure 3. In the left graph, the continuous-time eigenvalues of the system multiplied by h (indicated with the cross) are not equally spaced in the left-half-plane, and can be divided into two clusters: those that are close to the origin are associated with "slow dynamics", while the others are associated with "fast dynamics". The presence of the two different time scales is also evidenced by computing the stiffness ratio of Definition 5, i.e., $\sigma_R = 4.5/0.5 = 9$. Consider now the right graph of the same figure. The stiffness ratio is the same, since the closest and the farthest eigenvalues from the origin are the same, while the eigenvalues of the system are almost equally distributed in the left-half-plane. This feature of the system is strictly related to how much the system is "separable", and is not evidenced by the stiffness ratio.

Coming back to the CA approach, two different indices based on it can be defined. One, the *stiffness index*, quantifies the span of the time scales in the model, analogously to the one of Definition 5. The other, the *separability index*, indicates to what extent the clusters of state variables corresponding to those time scales can be computed in a decoupled manner. Both indices are function of α, and being based on CA, they can be computed also for nonlinear systems.

Denote by $\mathcal{H}$ the set of integration steps h_{x_i} associated with each state variable, and assume $\mathcal{H}$ ordered by ascending values of h, i.e., $\mathcal{H} = \{h_1 \leq h_2 \leq \ldots \leq h_N\}$. Based on that, the following definitions can be given.

Definition 6 (Stiffness index)**.** *The* stiffness index *for a given α is the ratio between the minimum and the maximum integration step found with the cycle analysis, i.e.,*

$$\sigma(\alpha) = \frac{h_{\max}(\alpha)}{h_{\min}(\alpha)}. \tag{11}$$

Analogously to the stiffness ratio σ_R, also for the stiffness index highly stiff systems are associated with high values of σ.

Definition 7 (Separability term)**.** *The* separability term *for a given α, and for a given couple of variables x_i and x_j is*

$$s_\alpha(i,j) = \frac{|h_i(\alpha) - h_j(\alpha)|}{\max_m \left(h_{m+1}(\alpha) - h_m(\alpha)\right)}, \qquad h_i, h_j \in \mathcal{H}.$$

Definition 8 (Separability index)**.** *The* separability index *for a given α is the unity minus the ratio between the maximum and the average difference among two subsequent values of the time scales, i.e.,*

$$s(\alpha) = 1 - \frac{\frac{1}{N-1}\sum_{i=1}^{N-1} h_{i+1}(\alpha) - h_i(\alpha)}{\max_i \left(h_{i+1}(\alpha) - h_i(\alpha)\right)} = 1 - \frac{1}{N-1}\sum_{i=1}^{N-1} s_\alpha(i+1,i).$$

5 Mixed-mode integration

This section deals with how the information coming from CA can be exploited to improve simulation efficiency. Broadly speaking, since a complete treatise of this matter will require more than one future work, we can say that such an exploitation takes place along two fundamental axes. One refers to the used integration methods, the other to the adopted simulation architecture.

5.1 Exploiting by integration methods

Having clustered the dynamic variables by time scale, one can use explicit integration methods for the slow ones, and implicit methods for the fast ones. This implies loosing a precise representation of fast phenomena, but apparently improves efficiency. Moreover, the type of information provided by CA facilitates the modeller, as he/she has just to decide which is the smallest time scale of interest for the simulation study at hand.

We now illustrate some applications of this idea, limiting the scope to a system partitioned in two subsystems. This is done for simplicity and without loss of generality, since after a first partition one could simple re-apply the proposed technique to one or more of the obtained subsystems. We also use different couples of integration methods, to show that the proposed exploitation is feasible whatever couple (in general, set) of such methods is selected [26]. Incidentally, this further highlights the neat separation between the analysis and the simulation part, thus between the probe method used for CA, and the integration methods. Finally, we show that once the order of accuracy of the used methods is chosen, the technique leads to the same mixed-mode integration algorithm whatever the particular methods are. This means that the couples of considered methods are equivalent from an accuracy viewpoint.

Coming to the exploitation technique, consider the generic nonlinear ODE system

$$\dot{\mathbf{x}} = \mathbf{f}\left(\mathbf{x}\right) \tag{12}$$

and assume it to be partitioned into two subsystem: one with slow dynamics, the other with fast dynamics. Following an approach similar to the one presented in [23], we can left-multiply the state vector by a projection matrix $P = \text{diag}\{p_1, p_2, \ldots, p_n\}$, with $p_i \in \{0, 1\}$ to select the slow part, and by $\overline{P} = I - P$ to select the fast part. Therefore (12) can be written as

$$\begin{cases} \dot{\mathbf{x}}^S = P\dot{\mathbf{x}} = P\mathbf{f}\left(\mathbf{x}^S, \mathbf{x}^F\right) \\ \dot{\mathbf{x}}^F = \overline{P}\dot{\mathbf{x}} = \overline{P}\mathbf{f}\left(\mathbf{x}^S, \mathbf{x}^F\right) \end{cases} \tag{13}$$

where $\mathbf{x}^S$ represents the slow variables, and $\mathbf{x}^F$ the fast ones.

To qualify the used methods, we adopt the classical notation used for Runge-Kutta ones, i.e., a method of order s is expressed in the general form

$$x_{k+1} = x_k + h\sum_{i=1}^{s} b_i\kappa_i, \quad \text{with} \quad \kappa_i = \mathbf{f}\left(x_k + h\sum_{j=1}^{s} a_{ij}\kappa_j, t_k + c_i h\right), \tag{14}$$

where the coefficients a_{ij}, b_i and c_i are usually expressed by means of the so-called Butcher tableau

$$\begin{array}{c|c} c & A \\ \hline & b \end{array} = \begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & \cdots & a_{ss} \\ \hline & b_1 & \cdots & b_s \end{array}.$$

Recall for convenience that an explicit Runge-Kutta method is characterised by $a_{ij} = 0$ for all $j \geq i$, which is not true for implicit ones.

5.1.1 Explicit-Implicit Euler

First of all we consider the simplest exploitation case, i.e., using Explicit Euler (EE) for the slow part, and Implicit Euler (IE) for the fast part. This approach has already been presented in [23] for the linear case.

The Butcher tableaux of the two methods are respectively

$$\text{EE:}\quad \begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array} \qquad\qquad \text{IE:}\quad \begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

Correspondingly, equation (13) can be expressed as

$$\begin{aligned} \mathbf{x}_{k+1}^S =& P\mathbf{x}_{k+1} = P\mathbf{x}_k + hP\mathbf{f}\left(\mathbf{x}_k^S, \mathbf{x}_k^F, t_k\right) \\ \mathbf{x}_{k+1}^F =& \overline{P}\mathbf{x}_{k+1} = \overline{P}\mathbf{x}_k + h\overline{P}\mathbf{f}\left(\mathbf{x}_{k+1}^S, \mathbf{x}_{k+1}^F, t_k\right), \end{aligned}$$

which in the linear case becomes

$$\begin{aligned} \mathbf{x}_{k+1}^S =& P\mathbf{x}_k + hPA\mathbf{x}_k \\ \mathbf{x}_{k+1}^F =& \overline{P}\mathbf{x}_k + h\overline{P}A\mathbf{x}_{k+1}. \end{aligned}$$

Composing those two equations, and solving for $\mathbf{x}_{k+1}$, we can obtain

$$\mathbf{x}_{k+1} = \left(I - h\left(I - P\right)A\right)^2 \left(I + hPA\right)\mathbf{x}_k.$$

5.1.2 Explicit-Implicit midpoint

In this section we consider the Explicit Midpoint (EM) and Implicit Midpoint (IM), two second-order accurate integration methods. The corresponding Butcher tableaux are

$$\text{EM:}\quad \begin{array}{c|cc} 0 & 0 & 0 \\ 1/2 & 1/2 & 0 \\ \hline & 0 & 1 \end{array} \qquad\qquad \text{IM:}\quad \begin{array}{c|c} 1/2 & 1/2 \\ \hline & 1 \end{array}$$

According to (14), the slow part of equation (13) becomes

$$\mathbf{x}_{k+1}^S = P\mathbf{x}_{k+1} = P\mathbf{x}_k + h\kappa_2 \qquad \text{with} \qquad \begin{aligned} \kappa_1 =& P\mathbf{f}\left(\mathbf{x}_k^S, \mathbf{x}_k^F, t_k\right) \\ \kappa_2 =& P\mathbf{f}\left(\mathbf{x}_k^S + \frac{h}{2}\kappa_1, \mathbf{x}_k^F, t_k + \frac{h}{2}\right), \end{aligned}$$

leading to the more compact equation

$$\mathbf{x}_{k+1}^S = P\mathbf{x}_k + hP\mathbf{f}\left(\mathbf{x}_k^S + P\frac{h}{2}\mathbf{f}_k, \mathbf{x}_k^F, t_k + \frac{h}{2}\right)$$

where

$$\mathbf{f}_k \triangleq \mathbf{f}\left(\mathbf{x}_k^S, \mathbf{x}_k^F, t_k\right).$$

As for the fast part, the IM method can be used. Thus, using the numerical solution of the slow part as an input for the fast part, leads to

$$\mathbf{x}_{k+1}^F = \overline{P}\mathbf{x}_{k+1} = \overline{P}\mathbf{x}_k + h\overline{P}\kappa_1 \qquad \text{with} \qquad \kappa_1 = \mathbf{f}\left(\mathbf{x}_k + \frac{h}{2}\kappa_1, t_k + \frac{h}{2}\right). \tag{15}$$

Observing that $\kappa_1 = (\mathbf{x}_{k+1} - \mathbf{x}_k)/h$, equation (15) can be written in a more compact form

$$\mathbf{x}_{k+1}^F = \overline{P}\mathbf{x}_k + h\overline{P}\mathbf{f}\left(\frac{\mathbf{x}_{k+1}^S + \mathbf{x}_k^S}{2}, \frac{\mathbf{x}_{k+1}^F + \mathbf{x}_k^F}{2}, t_k + \frac{h}{2}\right).$$

In the linear case, the two expressions become

$$\begin{aligned}\mathbf{x}_{k+1}^S =& P\mathbf{x}_k + hPA\mathbf{x}_k + \frac{h^2}{2}\left(PA\right)^2\mathbf{x}_k \\ \mathbf{x}_{k+1}^F =& \overline{P}\mathbf{x}_k + \frac{h}{2}\overline{P}A\mathbf{x}_k + \frac{h}{2}\overline{P}A\mathbf{x}_{k+1}\end{aligned}.$$

Composing the two expressions, the overall dynamics can be computed as

$$\mathbf{x}_{k+1} = \left(I - \frac{h}{2}\left(I-P\right)A\right)^{-1}\left(I + \frac{h}{2}\left(I+P\right)A + \frac{h}{2}\left(PA\right)^2\right)\mathbf{x}_k.$$

5.1.3 Heun-Lobatto mixed-mode

Analogous computations can be done with different integration methods. For example the Heun's method can be taken as the explicit method, and the second-order Lobatto IIIA one for the implicit part. The Butcher tableaux of the two methods are respectively

$$\text{Heun:}\quad \begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & 1/2 & 1/2 \end{array} \qquad\qquad \text{Lobatto IIIA:}\quad \begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \end{array}$$

With similar computations to the ones performed for the midpoints methods, it is easy to show that the slow part dynamics are ruled by

$$\begin{aligned}\mathbf{x}_{k+1}^S =& P\mathbf{x}_k + h\mathbf{f}_k\frac{h^2}{2}\mathbf{f}\left(\mathbf{x}_k^S + hP\mathbf{f}_k, \mathbf{x}_k^F, t_k + h\right) \\ \mathbf{x}_{k+1}^F =& \overline{P}\mathbf{x}_k + \frac{h}{2}\overline{P}\left(\mathbf{f}_k + \mathbf{f}_{k+1}\right)\end{aligned}$$

Which in the linear case become

$$\begin{aligned}\mathbf{x}_{k+1}^S =& P\mathbf{x}_k + hPA\mathbf{x}_k + \frac{h^2}{2}\left(PA\right)^2\mathbf{x}_k \\ \mathbf{x}_{k+1}^F =& \overline{P}\mathbf{x}_k + \frac{h}{2}\overline{P}A\mathbf{x}_k + \frac{h}{2}\overline{P}A\mathbf{x}_{k+1}\end{aligned}$$

Leading to the overall dynamics

$$\mathbf{x}_{k+1} = \left(I - \frac{h}{2}\left(I-P\right)A\right)^{-1}\left(I + \frac{h}{2}\left(I+P\right)A + \frac{h^2}{2}\left(PA\right)^2\right)\mathbf{x}_k$$

which is exactly the same dynamic obtained by the combination of any second-order accurate couple of Runge-Kutta methods.

5.1.4 General application

Alternatively to the proposed techniques, other couples of Explicit-Implicit Runge-Kutta methods can be used, for example all the ones proposed in [26]. In principle one could also use completely different methods, which however we do not discuss here owing to the predominant diffusion of

Runge-Kutta ones, and also because abandoning that class generally means giving up also the Butcher tableau formalism, to the (sole, however) detriment of the treatise simplicity.

Summarising, exploiting CA by integration method leads to join the best of implicit and explicit integration in a knowledgeable manner for the case under question. The resulting integration scheme is represented in Figure 4.

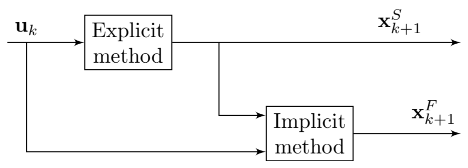

Figure 4: Mixed-mode integration scheme.

This kind of scheme will be used in Section 7 to evaluate the performance of the proposed methods.

5.2 Exploiting by simulation architecture

Independently of the time scale associated with each subsystem, CA provides structural information on the mutual dependencies among the dynamic variables, as it inherently analyses the graph of said dependencies. As such, one can further exploit CA by introducing any kind of parallelism for the parts of the system that CA has shown to be weakly coupled—it is worth stressing, no matter what integration method is selected for them.

This idea can have several applications. For example, if two subsystems are not coupled to one another but only via a third one, then apparently the first two can be solved in parallel (such a situation may appear in the presence of several hydraulic circuits connected to a large central capacity, and a number of other similar cases can be thought of). Most important, detecting such parallelisation possibilities is done easily by applying e.g. BLT-like techniques to the subsystems obtained from CA, i.e., also this kind of exploitation can be automated: an example of this *modus operandi* can be found in [27].

From a more technological standpoint, one can then just employ parallel computing architectures, or even use the so obtained information to structure a co-simulation setup. In the latter case, the proposed technique provides more formally grounded an alternative to heuristics based e.g. on the minimisation of the number of signals exchanged among the co-simulation units [28, 29]. Of course such optimisations are not possible when the structure of the simulation setup is dictated by the used software tools, but in the last years formalisms and standards have been emerging to provide designers with more freedom in this respect, see e.g. [30, 31].

As a final but important remark, the proposed approach allows to obtain a co-simulation setup starting from a monolithic model. This can be extremely useful to solve the initialisation problem, as doing so in a centralised manner generally yields improved convergence guarantees with respect to a distributed approach [32].

6 Toolchain extension

This section presents a possible way of complementing the typical EOO M&S manipulation toolchain (see Figure 5) so as to include simplification techniques like that proposed herein. The goal is to allow the analyst to provide the tool with approximation-related specifications at a high level, like it is already possible with information pertaining e.g. to tolerances, integration algorithm, and the like.

Figure 5 depicts the extended toolchain. The decision nodes (the diamond ones in the diagram) show where additional manipulation for simplification can be performed. If no simplification is required, clearly, the classical manipulation toolchain comes out, while in the opposite case, the desired approximation techniques are applied at the suitable points. The diagram also reports some coloured dashed boxes on the right side. Red boxes stand for already available methodologies, automatically applicable at the corresponding level of the manipulation. Green ones stand for potential methodologies which may be introduced as automatic procedures, but to date are not exploited in the context of EOO modelling.

6.1 An example toolchain implementation

To prove the feasibility of our extension proposal, the task of realising it was actually carried out by using JModelica[1] as the Modelica translator, exporting the model as a Functional Mockup Unit (FMU), and employing Assimulo[2] for the numerical integration, having developed the mixed-mode integrator *ad hoc*.

More in detail, the toolchain of Figure 5 was modified – for the case when "simplify" is desired – as shown in Figure 6: the output of the continuous-time part (the manipulated `model.mo`) is exported by means of the Functional Mockup Interface (FMI) to `model.fmu`, elaborated by the external python module `jd2.py` that performs CA (i.e., takes care of the "discretisation" and the "solution manipulation" blocks); the partitioned model is then simulated with Assimulo, with the developed mixed-mode method. It is worth noticing that the integration of a new functionality (like DD) into an EOO modelling toolchain was greatly eased by adopting tools that allow for some common interchange format—a feature of great importance indeed.

The developed code, including the reported examples, is available as free software[3], within the terms of the Modelica License v2.

7 Application examples

In this section two different application examples are described and analysed. In particular a parametric separability analysis is presented (so as to analyse the system behaviour independently of the choice of α), and a mixed-mode integration method is used to evaluate the simulation performance.

The obtained results are compared with other integration methods, i.e., two first-order fixed step integration methods – Explicit Euler (EE) and Implicit Euler (IE) – and another sophisticated multi-step methods – Backward Differentiation Formulas (BDF) – is used as baseline to compute

[1]`http://www.jmodelica.org`
[2]`http://www.jmodelica.org/assimulo`
[3]The code is available at `http://home.dei.polimi.it/leva/jd2.html`.

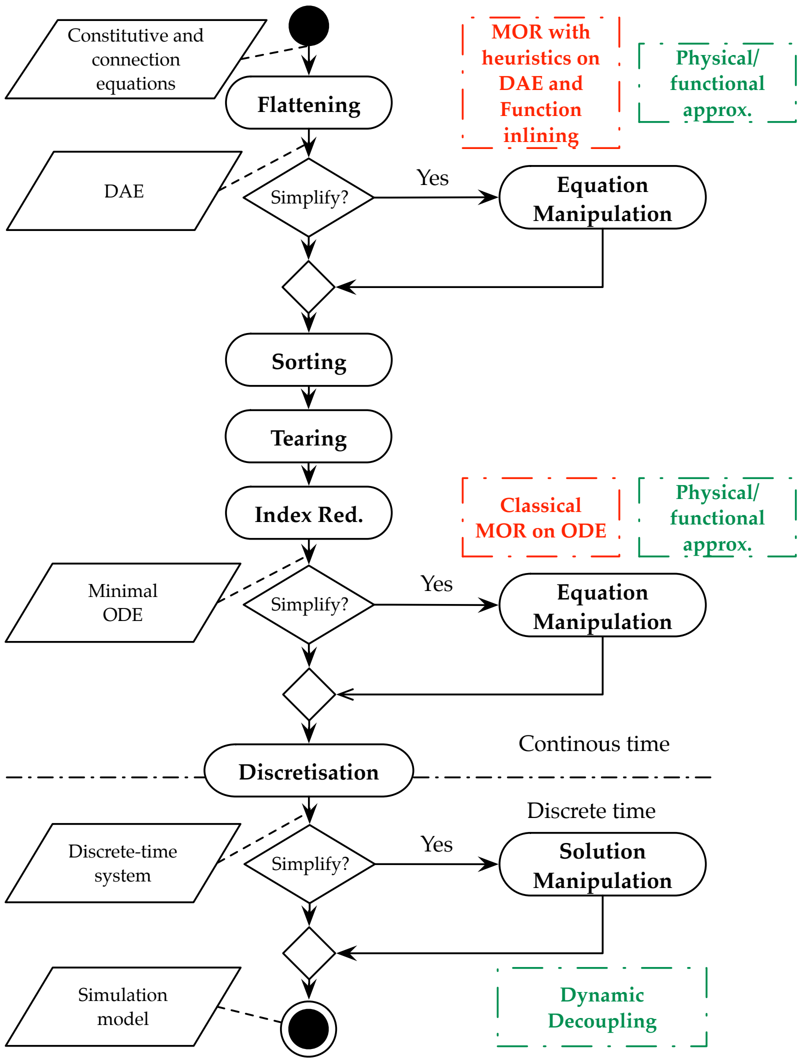

Figure 5: Activity diagram of the modified manipulation toolchain.

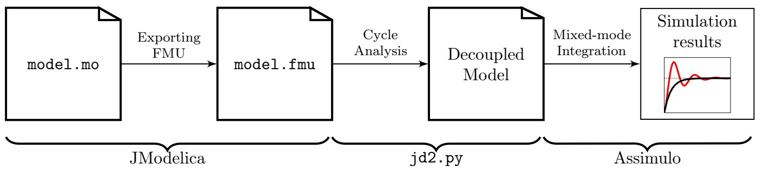

Figure 6: Integration of DD in the toolchain of Figure 5.

the quality of the computed solution in terms of Root Mean Square Error (RMSE)

$$\mathrm{RMSE} = \max_{i=1,\ldots,n} \sqrt{\frac{\sum_{k=1}^{\lfloor \frac{T}{h} \rfloor} \left(\hat{x}_{i,k} - x_{i,k}\right)^2}{T}},$$

where T is the final time, h is the integration step, $x_{i,k}$ and $\hat{x}_{i,k}$ represent the reference and the approximated solution of the i-th variable, respectively. All the simulation results were obtained using JModelica and Assimulo.

7.1 Automotive suspension

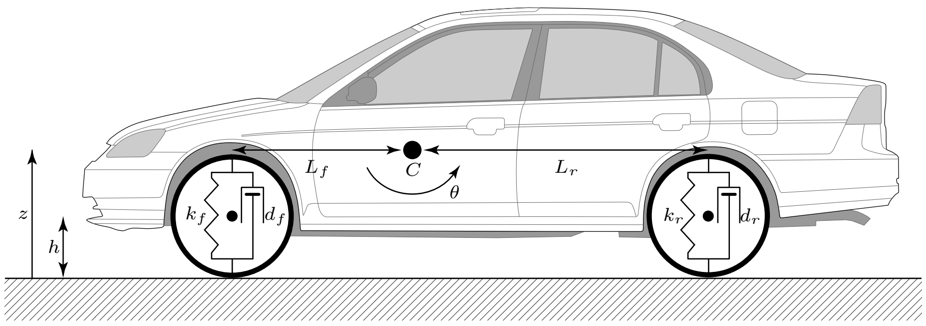

Figure 7: Half-car suspension model.

Figure 7 illustrates the modelled characteristics of a half-car. The front and rear suspension are modelled as spring/damper systems. A more detailed model would include a tire model, and damper nonlinearities such as velocity-dependent damping (with greater damping during rebound than compression). The vehicle body has pitch and bounce degrees of freedom. They are represented in the model by four dynamic states: vertical displacement z, vertical velocity $\dot{z}$, pitch angular displacement θ, and pitch angular velocity $\dot{\theta}$. The front and rear suspensions influence the bounce (i.e., vertical degree of freedom) according to the equations

$$F_f = 2k_f \left(L_f\theta - z\right) + 2d_f \left(L_f\dot{\theta} - \dot{z}\right), \quad F_r = 2k_r \left(L_r\theta + z\right) - 2d_r \left(L_r\dot{\theta} + \dot{z}\right),$$

where F_f and F_r are the upward force on body from front and rear suspension, k_f and k_r are the front and rear suspension spring constant, d_f and d_r are the front and rear damping factors, L_f and L_r are the horizontal distance from the center of gravity (C) to front and rear suspensions. The pitch contribution to the front and rear suspension is given by

$$\tau_f = -L_f F_f, \quad \tau_r = L_r F_r,$$

where τ_f and τ_r are the pitch torque due to the front and rear suspension. Hence, using the Newton's law, the forces and moments are balanced as

$$M\ddot{z} = F_f + F_r - mg, \quad J\ddot{\theta} = \tau_f + \tau_r + \tau_a,$$

where M is the body mass, J is the body momentum of inertia about the center of gravity and τ_a is the pitch torque induced by vehicle acceleration. In this example we initially consider the vehicle accelerating, so $\tau_a = 500\,\mathrm{N\,m}$ for $t \geq 0$ (0 otherwise). Hence, at time $t = 2$, we consider the road height h increased by $0.15\,\mathrm{m}$. The parameters used for this example are reported in Table 1.

Parameters							
M	1000 kg	k_f	28 000 N/m	d_f	3500 N s/m	L_f	0.9 m
J	2100 kg m^2	k_r	21 000 N/m	d_r	3500 N s/m	L_r	1.2 m

Table 1: Half-car suspension parameters.

The result of a parametric cycle analysis is presented in Figure 8, showing that it is possible to separate the model between the first and the second variable, if a medium value of α is chosen. However, in this very simple case it is quite hard to understand if a neat separation exists.

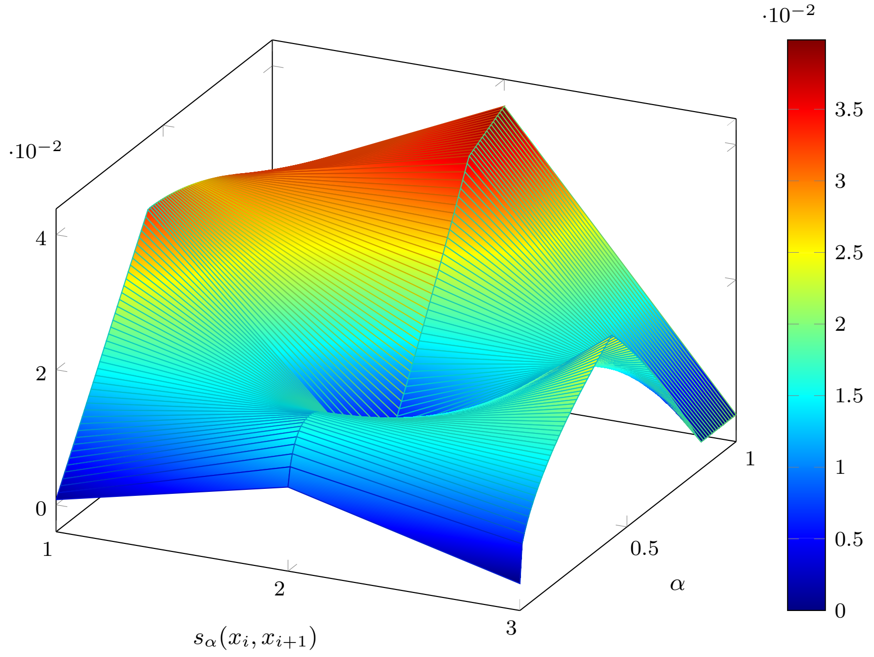

Figure 8: Separability parametric analysis of half-car suspension model.

According to CA there are 7 cycles in the model digraph, and choosing $\alpha = 0.5$, the following constraints on the integration step are obtained.

$$\begin{aligned} \dot{z}: \quad & h \leq 0.0384615 & z: \quad & h \leq 0.0714286 \\ & & \dot{\theta}: \quad & h \leq 0.0769231 \\ & & \theta: \quad & h \leq 0.0996024 \end{aligned} \tag{16}$$

hence, choosing an integration step $h = 0.05$, we can separate the system accordingly to what suggested by Figure 8. Figure 9 shows the simulation results with respect to a reference solution, obtained with a BDF (Backward Differentiation Formulas) method, with both the absolute and relative tolerances set to 10^{-6}.

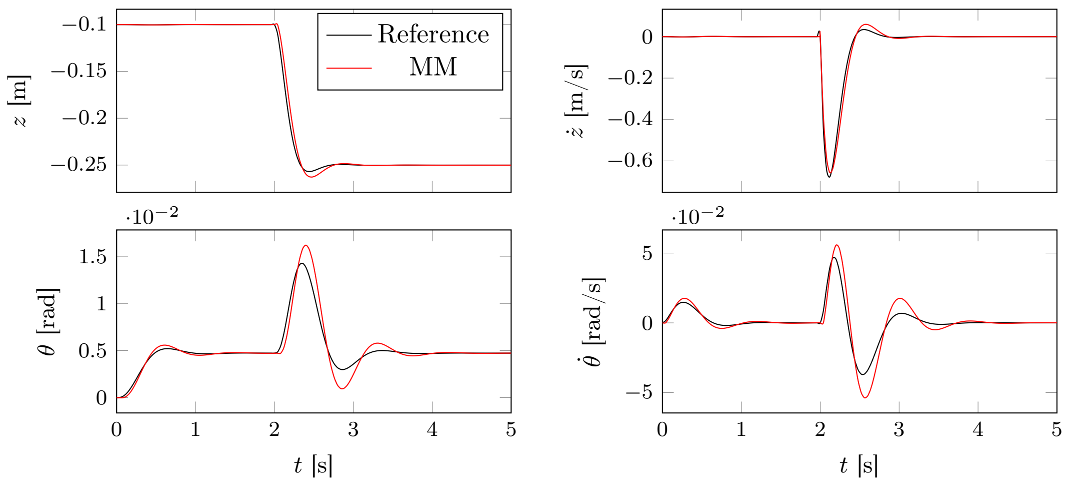

Figure 9: Simulation results of the half-car suspension model.

Table 2 shows the simulation statistics for different integration methods. It is worth noticing that the dimension of the system the Newton iteration has to solve is reduced from 4 to 1 in the mixed-mode method. Notice, also that the EE method needs a smaller step size ($h = 0.02$) for numerical stability reasons. The mixed-mode method in this very simple example represents a tradeoff between the IE and EE, as achieve the same precision in terms of RMSE as IE, but with a lower simulation time. Notice also that the time required for the cycle analysis is negligible with respect to the overall simulation time.

	Mixed-mode	**BDF**	**IE**	**EE**
# Steps	**100**	198	**100**	250
# Function ev.	376	236	**375**	–
# Jacobian ev.	**6**	5	**6**	–
# Fun. ev. in Jac. ev.	**12**	20	30	–
# Newton iterations	250	228	**249**	–
RMSE	0.056	–	**0.053**	0.059
Sim time	0.1s	0.16s	0.13s	**0.08s**
CA time	0.654×10^{-3}s			

Table 2: Simulation statistics for half-car model.

To complete the example, the indices proposed in Section 4 are here computed, yielding the following indices

$$\sigma_R = 1.869, \quad \sigma(0.5) = 2.590, \quad s(0.5) = 0.382.$$

The stiffness $\sigma(\alpha)$ index shows that the system is stiff, while the separability one shows that this kind of system is sufficiently suited for separation.

7.2 Counterflow heat exchanger

This example refers to a counterflow heat exchanger with two incompressible streams (Figure 10). Both streams and the interposed wall are spatially discretised with the finite volume approach,

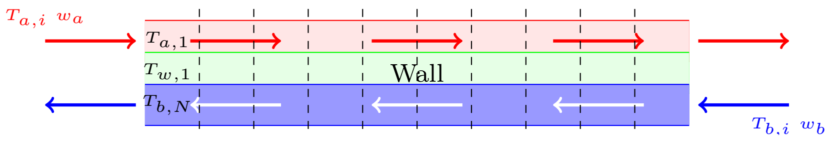

Figure 10: Counterflow heat exchanger scheme.

neglecting axial diffusion in the wall – as is common practice – and also in the streams, as zero-flow operation is not considered for simplicity. Taking ten volumes for both streams and the wall, with the same spatial division (again, for simplicity) leads to a nonlinear dynamic system of order 30, having as boundary conditions the four pressures at the stream inlets and outlets, and the two temperatures at the inlets. Notice that the indices counting the sections go from the inlet flow to the outlet flow. Formally, the system is given by

$$\begin{cases} c_a \frac{M_a}{N} \dot{T}_{a,i} = w_a c_a \cdot (T_{a,i-1} - T_{a,i}) + \frac{G_a}{N} \cdot (T_{w,i} - T_{a,i}) \\ c_w \frac{M_w}{N} \dot{T}_{w,i} = -\frac{G_a}{N} \cdot (T_{w,i} - T_{a,i}) - \frac{G_b}{N} \cdot (T_{w,i} - T_{b,N-i+1}) \\ c_b \frac{M_b}{N} \dot{T}_{b,i} = w_b c_b \cdot (T_{b,i-1} - T_{b,i}) + \frac{G_b}{N} \cdot (T_{w,N-i+1} - T_{b,i}) \end{cases} \tag{17}$$

where T stands for temperature, w for mass flowrate, c for (constant) specific heat, M for mass, and G for thermal conductance; the a, b and w subscripts denote respectively the two streams and the wall, while $i \in [1, N]$ ($i = 0$ for boundary conditions) is the volume index, counted for both streams from inlet to outlet, the wall being enumerated like stream a.

The parameter values used in the example are reported in Table 3.

Parameters					
N	10	M_b	1 kg	c_w	3500 J/(kg K)
$T_{a,\text{in}}$	323.15 K	M_w	10 kg	G_a	8000 W/K
$T_{b,\text{in}}$	288.15 K	c_a	4200 J/(kg K)	G_b	8000 W/K
M_a	0.1 kg	c_b	3500 J/(kg K)		

Table 3: Parameter values of Model (17).

A parametric CA is performed so as to analyse the structure of the system, and Figure 13 shows its result. In particular, 95 cycles are present in the system.

The separability analysis evidences that there are at least a couple of points where the system can be separated. However, for $\alpha = 1.0$ there is only one point in which the system can be split, and it is between the 10-*th* and the 11-*th* variable. It is worth noticing that in this example, there is no neat physical separation between the dynamics, since they all belong to the same physical domain, and also to the same physical object. Separability analysis, however, can detect those structural properties of the system independently of its nature.

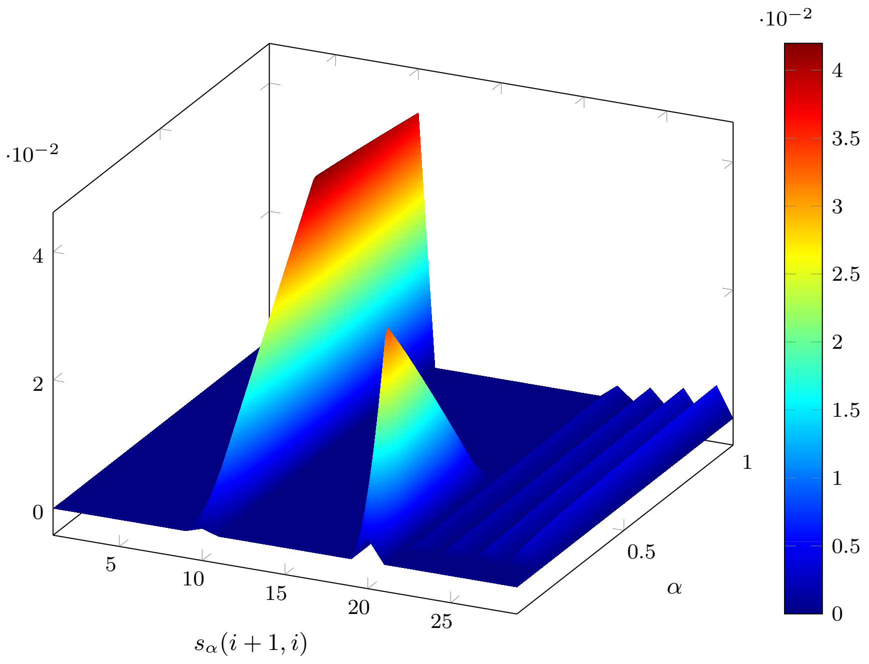

Figure 11: Separability analysis of the heat exchanger (17) with $N = 10$.

Hence, choosing $\alpha = 1.0$ CA leads to the following constraints:

$$T_{a,i}: \quad h \leq 0.0084 \qquad \begin{aligned} T_{b,i}: &\quad h \leq 0.0478 \\ T_{w,1,10}: &\quad h \leq 0.0478 \\ T_{w,2,9}: &\quad h \leq 0.0498 \\ T_{w,3,8}: &\quad h \leq 0.0524 \\ T_{w,4,7}: &\quad h \leq 0.0559 \\ T_{w,5,6}: &\quad h \leq 0.0606 \end{aligned}$$

Choosing an integration step $h = 0.04$ yields a partition of the system that considers the $T_{a,i}$ as the fast while the $T_{b,i}$ and $T_{w,i}$ as the slow states. Fig. 12 shows the simulation results compared to a reference solution, obtained again with a BDF method, with both the absolute and relative tolerances set to 10^{-6}. Notice that the temperatures are reported with different scales.

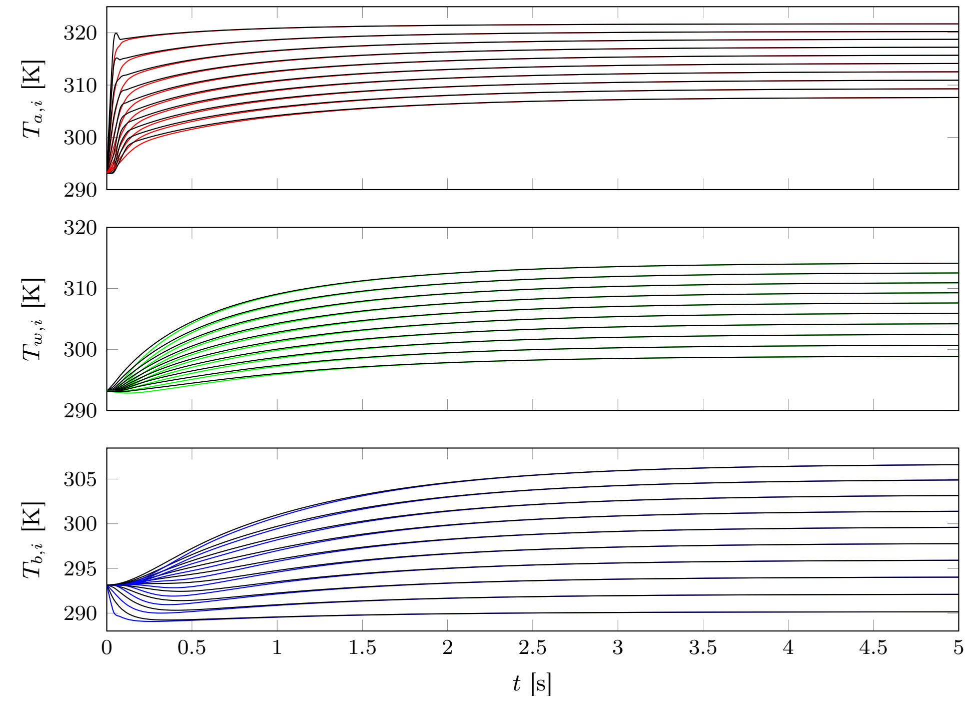

Figure 12: Simulation results of (17). Black lines represent the reference trajectories, while the coloured lines the mixed-mode ones.

Table 4 shows the simulation statistics for different integration methods. It is worth noticing that, since the complexity of IE is $\mathcal{O}(n^3)$, where n is the dimension of the model, integrating implicitly only $T_{a,i}$ instead of the whole model, reduces the computations from $30^3 = 27000$ to $10^3 = 1000$, leading to a significant improvement in terms of simulation efficiency. Notice also that the EE method needs a smaller step size ($h = 0.01$) for numerical stability reasons. In this example, the simulation time is half of the one needed by IE, while achieving slightly worse performance in terms of RMSE.

	Mixed-mode	**BDF**	**IE**	**EE**
# Steps	**125**	260	**125**	500
# Function ev.	375	**296**	375	–
# Jacobian ev.	6	**5**	6	–
# Fun. ev. in Jac. ev.	**66**	150	186	–
# Newton iterations	**250**	292	**250**	–
RMSE	0.091	–	**0.080**	0.836
Sim time	**0.06s**	0.21s	0.12s	0.15s
CA time	6.708×10^{-3}s			

Table 4: Simulation statistics for Model (17) ($h = 0.04$).

Now, the indices proposed in Section 4 are here computed, yielding the following results—notice that also in this case, due to the nonlinearity of the system, σ_R cannot be computed.

$$\sigma(0.5) = 13.612, \quad s(0.5) = 0.954.$$

The stiffness $\sigma(\alpha)$ index shows that the considered system is sufficiently stiff, but the more interesting aspect is that the separability one shows that it is very suited for the partition, since its time scales are very well separated.

The presented examples have been kept as small as possible in order to improve results readability, but the method can be applied to larger models as well. For example, by changing in (17) the parameter N to 30, the model becomes of order 90. Thus, CA detects 585 cycles and the same parametric separability analysis can be performed.

In this case, the chosen integration step here is $h = 0.01$. The obtained simulation results are summarised in Table 5.

	Mixed-mode	**BDF**	**IE**	**EE**
# Steps	500	**290**	500	2000
# Function ev.	1321	**323**	1348	–
# Jacobian ev.	24	**6**	24	–
# Fun. ev. in Jac. ev.	744	**540**	2184	–
# Newton iterations	820	**319**	847	–
RMSE	**0.126**	–	0.131	0.023
Sim time	**0.76s**	0.96s	1.43s	0.80s
CA time	0.022s			

Table 5: Simulation statistics for Model (17) with $N = 30$.

In this case the indices become

$$\sigma(0.5) = 9.771, \quad s(0.5) = 0.986,$$

thus, even if the order of the system is changed, but the system is actually the same, those structural indices have the same order of magnitude, and, what is more important are not changed too much, providing the same information.

7.3 Discussion

After showing the examples, their collective outcome could be summarised as follows. First, when there is an evident dynamic separation in the system, the proposed technique finds it without

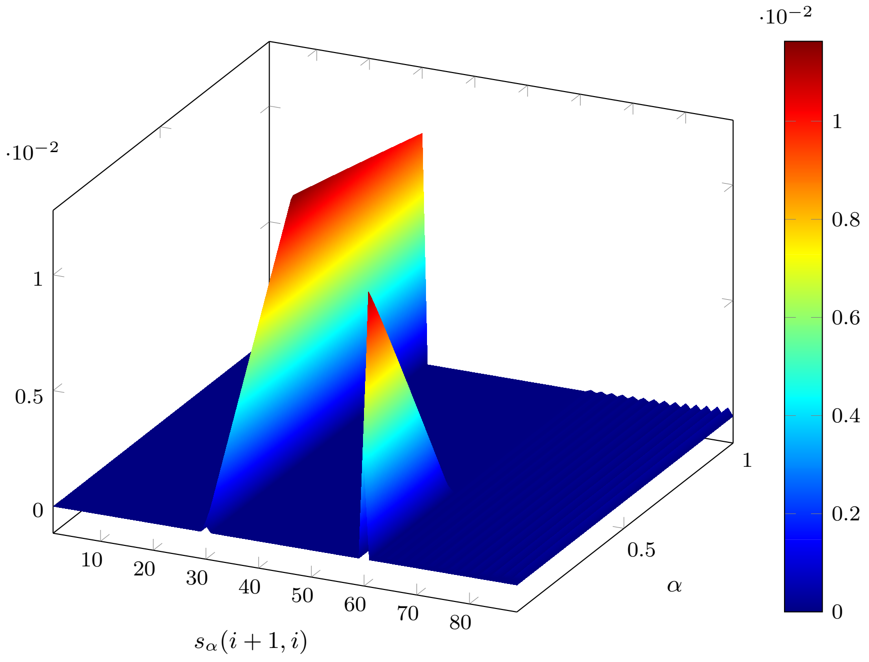

Figure 13: Separability analysis of the heat exchanger (17) with $N = 30$.

requiring *a priori* information on the part of the user. In other words, the technique is backed up by observing that the produced results are in accordance with intuition, when intuition can figure them out.

Also, and in some sense as a complement, the proposed indices allow to synthetically appreciate the possible internal model couplings that can be exploited via DD, even when these are not apparent at all.

Moreover, and specific to the use of the technique for mixed-mode integration, its characteristics are very suited to the typical studies that are required to really have simulation follow the life-cycle of the project, as envisaged in the introduction. On this final point, however, some more words are in order.

When a simulation study is required, the simulation time becomes critical, since a large number of simulations must be run, e.g., for optimization purposes. The presented cycle analysis takes less than 1 second in all the presented examples, and must be performed only once, during an offline phase. However, the structural analysis, and the consequent improved simulation efficiency speed up the simulation time, allowing for a faster simulation study.

In addition, when a simulation study is required to answer a specific question, most frequently the focus is on part of the system, or – somehow equivalently – on part of the phenomenon occurring in it. In such a very frequent case, the rest of the system does not need to be simulated accurately, provided that the boundary conditions presented to the part that is relevant for the study, allow for a precise evaluation of the investigated quantities. In many situations of the type just mentioned, the interest of the analyst is on certain time scales on the system phenomena, and provided these are well reproduced, loosing faster behaviours is not only acceptable, but in fact necessary to achieve the desired performance. In fact, the same remark holds also for almost the totality of MOR techniques, where low-frequency approximations of the original model are the typical result, and the quality of a reduction is not evaluated in terms of time error – which is typically large due to the transients – but rather in terms H_∞-norm of the difference between the original and the reduced model, i.e., in the frequency domain. This is totally analogous to the proposed approach, where a good approximation is not strictly related to a small simulation error, but to a good representation of the time scales of interest.

8 Conclusions and future work

A technique was presented to allow EOO M&S tools to take profit of DD, partitioning a complex model into "weakly coupled" submodels in a view to enhancing the obtained simulation efficiency. With respect to the major available alternatives, the presented technique has more than one advantage. It preserves the state space of the model, can be considered scenario-free, and is applicable both in the case of a monolithic solution and of co-simulation.

The technique is based on a structural analysis of the system – called here cycle analysis, and novel – coupled to a convenient use of mixed-mode integration. Both aspects were discussed, leading also to the definition of convenient separability indices, that allow the engineer to tailor the model partitioning transparently, based on easily interpreted information. The emphasis was also set on how the presented technique can be integrated in the typical EOO M&S toolchain, considering a Modelica translator as a representative case. Simulation tests were reported to illustrate the achieved benefits, and the realised implementation was made available as free software to the community.

Future work will concern the exploitation of the mentioned keenness to co-simulation, and a comprehensive user interface to further ease the toolchain integration, having as ultimate goal a unifying approximation framework comprising the proposed technique and alternative ones like MOR, TLM, and so forth. Another subject will be a further investigation of the physical interpretation of α, and on how to formally relate it to the simulation accuracy.

Finally, models of higher complexity will be addressed, so as to possibly improve the time performance of the software implementation.

Acknowledgement

This work was partially supported by the Swedish Research Council (VR) under contract number C0590801 (2012-5908) for the project Cloud Control and through the LCCC Linnaeus Center.

References

[1] Zimmer, D. (2013, Aug). A new framework for the simulation of equation-based models with variable structure. *SIMULATION: Transactions of the Society for Modeling and Simulation International 89*(8), 935–963.

[2] Papadopoulos, A. V. and A. Leva (2013). Automating dynamic decoupling in object-oriented modelling and simulation tools. In *5th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools.*, pp. 37–44.

[3] Cellier, F. E. and E. Kofman (2006). *Continuous system simulation.* London, UK: Springer.

[4] Antoulas, A. (2005). *Approximation of large-scale dynamical systems*, Volume 6. Society for Industrial Mathematics.

[5] Mikelsons, L. and T. Brandt (2011). Generation of continuously adjustable vehicle models using symbolic reduction methods. *Multibody System Dynamics 26*, 153–173.

[6] Arnold, M. and W. O. Schiehlen (2009). *Simulation Techniques for Applied Dynamics*, Volume 507. Springer.

[7] Bastian, J., C. Clauß, S. Wolf, and P. Schneider (2011). Master for co-simulation using FMI. In *Proc. of the 8th International Modelica Conference*, pp. 115–120.

[8] Bartolini, A., A. Leva, and C. Maffezzoni (1998). A process simulation environment based on visual programming and dynamic decoupling. *SIMULATION: Transactions of the Society for Modeling and Simulation International 71*(3), 183–193.

[9] Papadopoulos, A. V., J. Åkesson, F. Casella, and A. Leva (2013). Automatic partitioning and simulation of weakly coupled systems. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pp. 3172–3177.

[10] Papadopoulos, A. V. and A. Leva (2014). A model partitioning method based on dynamic decoupling for the efficient simulation of multibody systems. *Multibody System Dynamics*, 1–28.

[11] Chen, J., S.-M. Kang, J. Zou, C. Liu, and J. Schutt-Aine (2004). Reduced-order modeling of weakly nonlinear MEMS devices with Taylor-series expansion and Arnoldi approach. *Microelectromechanical Systems, Journal of 13*(3), 441– 451.

[12] Phillips, J. R. (2000). Projection frameworks for model reduction of weakly nonlinear systems. In *Proc. of the 37th Annual Design Automation Conf.*, DAC '00, New York, NY, USA, pp. 184–189. ACM.

[13] Innocent, M., P. Wambacq, S. Donnay, H. Tilmans, W. Sansen, and H. De Man (2003). An analytic volterra-series-based model for a mems variable capacitor. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Trans. on 22*(2), 124–131.

[14] Scherpen, J. (1993). Balancing for nonlinear systems. *Systems & Control Letters 21*(2), 143–153.

[15] Lall, S., J. Marsden, and S. Glavaski (2002). A subspace approach to balanced truncation for model reduction of nonlinear control systems. *International Journal of Robust and Nonlinear Control 12*, 519–535.

[16] Shaker, H. R. and R. Wisniewski (2012). Model reduction of switched systems based on switching generalized gramians. *International Journal of Innovative Computing, Information and Control 8*(7(B)), 5025–5044.

[17] Mikelsons, L. and T. Brandt (2009). Symbolic model reduction for interval-valued scenarios. In *ASME Conf. Proc.*, Volume 49002, pp. 263–272. ASME.

[18] Sjölund, M., R. Braun, P. Fritzson, and P. Krus (2010). Towards efficient distributed simulation in modelica using transmission line modeling. In *3rd Int. workshop on Equation-Based Object-Oriented Modeling Languages and Tools*, pp. 71–80.

[19] Tarjan, R. (1972). Enumeration of the elementary circuits of a directed graph. *SIAM Journal on Computing 2*(3), 211–216.

[20] Johnson, D. (1975). Finding all the elementary circuits of a directed graph. *SIAM Journal on Computing 4*(1), 77–84.

[21] Mattsson, S. E. and G. Söderlind (1993). Index reduction in differential-algebraic equations using dummy derivatives. *SIAM Journal on Scientific Computing 14*(3), 677–692.

[22] Schierz, T., M. Arnold, and C. Clauß (2012). Co-simulation with communication step size control in an FMI compatible master algorithm. In *Proc. of the 9th International Modelica Conference*, pp. 205–214.

[23] Schiela, A. and H. Olsson (2000). Mixed-mode integration for real-time simulation. In *Modelica Workshop 2000 Proceedings*, pp. 69–75.

[24] Wolf, A., J. B. Swift, H. L. Swinney, and J. A. Vastano (1985). Determining Lyapunov exponents from a time series. *Physica D: Nonlinear Phenomena 16*(3), 285–317.

[25] Kuznetsov, Y. (2004). *Elements of Applied Bifurcation Theory*, Volume 112 of *Applied Mathematical Sciences*. Springer.

[26] Ascher, U. M., S. J. Ruuth, and R. J. Spiteri (1997). Implicit-explicit runge-kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics 25*(2–3), 151–167.

[27] Casella, F. (2013). A strategy for parallel simulation of declarative object-oriented models of generalized physical networks. In *5th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools.*, pp. 45–51.

[28] Kernighan, B. and S. Lin (1970). An efficient heuristic procedure for partitioning graphs. *Bell system technical journal 29*, 291–307.

[29] Hendrickson, B. and K. Devine (2000). Dynamic load balancing in computational mechanics. *Computer Methods in Applied Mechanics and Engineering 184*(2âĂŞ4), 485 – 500.

[30] Andersson, C., J. Åkesson, C. Führer, and M. Gäfvert (2011). Import and export of functional mock-up units in jmodelica.org. In *Proc. of the 8th International Modelica Conference*, pp. 329–338.

[31] Blochwitz, T., M. Otter, J. Åkesson, M. Arnold, C. Clauß, H. Elmqvist, M. Friedrich, A. Junghanns, J. Mauss, D. Neumerkel, H. Olsson, and A. Viel (2012). Functional Mockup Interface 2.0: The standard for tool independent exchange of simulation models. In *9th International Modelica Conference*, Münich, Germany, pp. 173–184.

[32] Burrage, K. (1993). Parallel methods for initial value problems. *Applied Numerical Mathematics 11*(1–3), 5–25.

Authors biographies

Alessandro Vittorio Papadopoulos received his PhD in Information Technology from the Politecnico di Milano, Italy. Currently, he is a Postdoctoral researcher in the Department of Automatic Control at Lund University (Sweden). He is also a member of the Lund Center for Control of Complex Engineering Systems (LCCC) Linnaeus Center. His research interests include modelling and simulation of dynamical systems, model order reduction for hybrid systems, and the application of control theory to the design and implementation of computing systems, especially in cloud computing.
Alberto Leva is Associate Professor of Automatic Control in the Dipartimento di Elettronica, Informazione e Bioingegneria of the Faculty of Engineering at the Politecnico di Milano, Italy. His main research interests are process modelling, simulation and control, automatic tuning of industrial regulators, and more recently, control and control-based design of computing systems. He is author or co-author of about 120 peer-reviewed publications in international journals or conferences.