This is the authors' version of a paper with reference: Stian Skjong, Eilif Pedersen (2019) On the numerical stability in dynamical distributed simulations. in Mathematics and Computers in Simulation, Volume 163, September 2019, Pages 183-203. The version of record is available at: https://doi.org/10.1016/j.matcom.2019.02.018

On the Numerical Stability in Dynamical Distributed Simulations

Stian Skjong[a,*], Eilif Pedersen[b]

[a] *SINTEF Ocean*
[b] *Norwegian University of Science and Technology, NTNU*
Department of Marine Technology

Abstract

This work takes aim at studying numerical stability in distributed simulations through dynamic stability and stability criteria for explicit solvers. This is done by studying outer stability limits, for example stability conditions when handling unstable subsystems or marginally stable solvers. To conclude global stability of a distributed system simulation both dynamic stability and solver stability must hold, and this work combines these stability criteria into one unified criterion for distributed linear dynamical systems. Some examples are given in order to both highlight numerical stability issues and to prove stability in different case studies. The derived stability criterion is also extended to include distributed systems containing non-linear dynamics.

Keywords: Distributed Simulation, Solver Stability, Solver Dynamics, Numerical Convergence.

1. Introduction

Stability is a widely used term in the field of modelling and is used both in context of dynamical systems and solutions in simulations. These concepts are often separated, even though both are somewhat dependent on the system characteristics, because they usually have separate area of application. In general, stability in systems is a measure of convergence, often expressed through asymptotic- or exponential characteristics, and is related to the dynamics in the system. Dynamic stability is well documented in the field of modelling [1, 2, 3], as well as in the field of control [4, 5]. Stability in numerical solutions is a measure of numerical convergence [6] and is more or less only dependent on the eigenvalues of the system to be solved and the solver characteristics. In other words, the stability of a system is a property characterized by the system dynamics and stability of solutions is a solver property, whether the eigenvalues of the system to be solved are inside the stability region of the chosen solver or not. However, in distributed systems the stability of the system and the solution seems to be more related.

Distributed systems are often characterized as systems that have been divided into smaller subsystems, connected in a distributed setup, communicating at fixed time instances, but often solved separately. This enables distribution of computational power both locally between cores in a computer as well as through a network among different computing members. A more thorough definition of distributed systems can be found in [7]. Distributed systems also give an advantage in industry collaborations. Different industries can keep their "know-how" hidden from competitors through the use of so called "black boxes" where the user only has access to the inputs and outputs from the system and where everything else is hidden. Then, third party vendors can distribute their submodels, or black box models, to cus-

*Corresponding author
Email addresses: stian.skjong@sintef.no (Stian Skjong), eilif.pedersen@ntnu.no (Eilif Pedersen)

tomers, yet still keep their "know-how" in-house. In order for subsystems in a distributed system to interact with each other, a common standard for exporting distributed systems and enabling communication between different distributed subsystems is required. One such standard is the Functional Mockup Interface, FMI [8], which in addition is tool independent and enables "black box" implementations of distributed models. Many examples on the use of distributed simulations, or co-simulations, can be found in different industries such as aerospace industry [9**?**], the automotive industry [10, 11, 12] and the marine industry [13, 14].

Dynamical stability of distributed systems is similar to sampled system stability [15, 16]. However, when using fixed-step size solvers, the dynamical stability of linear distributed systems is closely related to the stability of discretized linear systems [5, 17]. Stability of distributed systems has been studied from many different angles in the literature, for example through zero-stability analysis of coupled integration [18, 19], jacobian-based co-simulation algorithm to overcome stability issues [18], stability and convergence analysis of sequential algorithms [20], modular integration for Runge-Kutta methods [21] and Dahlquist test equations [22] for stability analysis of distributed systems [23, 24]. However, less results containing numerical stability of distributed simulation results can be found in the literature.

In this work the stability of distributed simulations will be studied on a general level, and much focus will be given to linear systems. The contributions in this work is similar to the Dahlquist test equation approach for analysing stability, but is based completely on the solver characteristics and the system dynamics for each connected submodel in a distributed system, with respect to local time steps and global communication time steps. These contributions also enable stability analysis of distributed systems including different explicit local solvers, that may have different local time steps, and is a continuation of the work presented in [25]. Following, the results from studying linear distributed systems with local explicit solvers are extended to also yield for non-linear distributed systems under certain conditions.

Before diving into the core topics, some background and definitions are given in the following.

2. Background Theory and Definitions

A general system of differential equations is here given as

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\tau_c}) \\ \boldsymbol{y} &= \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\tau}_c) \end{aligned} \tag{1}$$

where $\boldsymbol{x} \in \boldsymbol{\mathcal{R}}^n$ is the state vector, $\boldsymbol{u} \in \boldsymbol{\mathcal{R}}^m$ is the input vector given by the surrounding systems, $\boldsymbol{\tau_c} \in \boldsymbol{\mathcal{R}}^p$ is the control vector, $\boldsymbol{f}(\cdot) : \boldsymbol{\mathcal{R}}^n \times \boldsymbol{\mathcal{R}}^m \times \boldsymbol{\mathcal{R}}^p \rightrightarrows \boldsymbol{\mathcal{R}}^n$ is the vector of differential functions, $\boldsymbol{y} \in \boldsymbol{\mathcal{R}}^r$ is the output vector and $\boldsymbol{h}(\cdot) : \boldsymbol{\mathcal{R}}^n \times \boldsymbol{\mathcal{R}}^m \times \boldsymbol{\mathcal{R}}^p \rightrightarrows \boldsymbol{\mathcal{R}}^r$ is the output mapping function vector.

Relevant background theory regarding distributed systems and solver stability are given in the following.

2.1. Distributed Systems

A distributed system is in general a collection of subsystems interacting with each other with a given communication rate. Locally, a subsystem is solved by itself with fixed inputs and a local time step Δt_i, smaller than, or equal to, the communication time step T_d, and when all local time steps converge to the communication time step, the distributed system converges to one total discrete system. In general, dynamical systems given as (1) can be connected in a distributed system as shown in Figure 1. Note that the output mapping in the figure is slightly changed in comparison to (1).

In this work, the main focus will be given to linear dynamical systems, since, as will be shown later on, the results from studying linear systems can be extended to also yield for non-linear systems. This means that (1) may for subsystem i be rewritten as

$$\begin{aligned} \dot{\boldsymbol{x}}_{\boldsymbol{i}} &= \boldsymbol{A}_i \boldsymbol{x}_i + \boldsymbol{B}_i \boldsymbol{u}_i + \boldsymbol{B}_{ci} \boldsymbol{\tau}_{\boldsymbol{c}i} \\ \boldsymbol{y}_i &= \boldsymbol{C}_i \boldsymbol{x}_i \end{aligned} \tag{2}$$

where $\boldsymbol{A}_i \in \boldsymbol{\mathcal{R}}^{n \times n}$ is the state mapping matrix, $\boldsymbol{B}_i \in \boldsymbol{\mathcal{R}}^{n \times m}$ is the input matrix mapping, $\boldsymbol{B}_{ci} \in \boldsymbol{\mathcal{R}}^{n \times p}$ is the control matrix mapping and $\boldsymbol{C}_i \in \boldsymbol{\mathcal{R}}^{r \times n}$ is the output mapping matrix for a set of linear differential equations with n states, m inputs, r outputs and

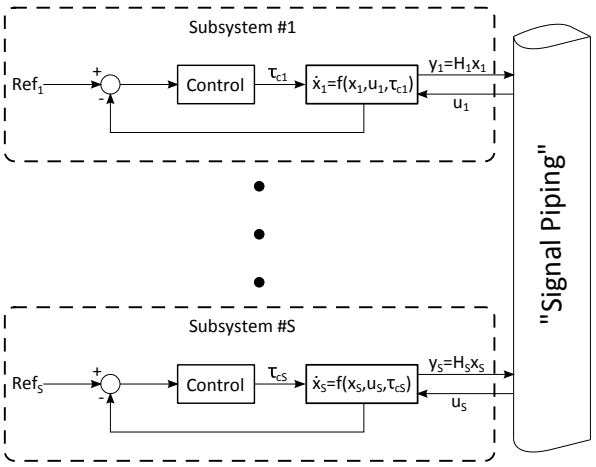


Figure 1: A general distributed system

p control variables. If only two single uncontrolled subsystems are present in the distributed system, the differential equations can be simplified and expressed as

$$\dot{x}_1 = a_1 x_1 + b_1 u_1 \tag{3a}$$
$$y_1 = c_1 x_1$$
$$\dot{x}_2 = a_2 x_2 + b_2 u_2 \tag{3b}$$
$$y_2 = c_2 x_2$$

where y_1 and y_2 are the subsystem outputs, and at each communication point

$$\begin{aligned} u_1 &:= y_2 \\ u_2 &:= y_1 \end{aligned} \tag{4}$$

A simple algorithm for solving such a distributed system is given in Algorithm 1. Note that $T_d \geq \max\{\Delta t_1, \Delta t_2\}$ in the algorithm.

Before solving a distributed simulation each subsystem may be analysed separately and studied as a single system in order to say something about the local subsystem stability. However, this is not enough to guarantee that the distributed system is stable, which will be shown in the following.

2.2. Local Solvers in Distributed Systems

In a distributed system each subsystem has its own local solver, and in this work only explicit solvers

Algorithm 1 Solution procedure for a distributed system containing two single, linear subsystems.

procedure DISTRIBUTEDSIM()
  Initialize()
  while $t \leq t_{stop}$ **do**
    $t = t + T_d$
    $u_1 = c_2 x_2$
    $u_2 = c_1 x_1$
    while $t_1 < t$ **do**
      $\dot{x}_1 = a_1 x_1 + b_1 u_1$
      $[x_1,\ \Delta t_1] = \text{Solve}(\dot{x}_1, x_1, t_1)$
      $t_1 = t_1 + \Delta t_1$
    end while
    while $t_2 < t$ **do**
      $\dot{x}_2 = a_2 x_2 + b_2 u_2$
      $[x_2,\ \Delta t_2] = \text{Solve}(\dot{x}_2, x_2, t_2)$
      $t_2 = t_2 + \Delta t_2$
    end while
    Collect$(t, x_1, x_2 ...)$
  end while
  plot$(t, x_1, x_2 ...)$
end procedure

are considered. A thorough introduction to numerical solvers is given in [26] and will not be given any particular attention here. For simplification reasons, the solvers used here are also assumed to have fixed time-step sizes. This is because the same conditions for assuring stable simulation results will yield for variable time-step sizes, where the maximal (and in some cases the minimal) local time-step sizes are determined. The stability of the local solvers in each subsystem is closely related to the eigenvalues in the subsystem itself, and by knowing the eigenvalues it is possible to choose a solver and a time step that stabilizes the local solution. Moreover, such solvers are also linear which means that when solving non-linear differential equations the solvers approximate the solution of the system by solving piecewise linearized systems of the non-linear system. This fact will be useful later on.

For subsystem i, given as in (2), the eigenvalues λ_i can be found by solving

$$\det(\boldsymbol{I}\lambda - \boldsymbol{A}_i) = 0 \tag{5}$$

where $\boldsymbol{I}$ is the diagonal unit matrix of size $n \times n$ where n is the number of states in the subsystem. By assuming that the forward Euler integration method is used one can find the largest time step for which the solution is stable. A simple differential equation given as

$$\dot{x} = f(x) \tag{6}$$

can be solved by the forward Euler integration method as

$$x_{i+1} = x_i + f(x_i)\Delta t_i \tag{7}$$

where x_{i+1} is the numerical solution of the differential equation at time $t_{i+1} = t_i + \Delta t_i$. The forward Euler integration method is stable if $\forall\ \lambda_i$,

$$|1 + \lambda_i \Delta t_i| \leq 1, \quad \Delta t_i > 0 \tag{8}$$

where λ_i is eigenvalue $i \in \{1, ..., n\}$ of the differential system. If the eigenvalues are all real, the stability criteria can be simplified to

$$\Delta t_i \leq -\frac{2}{\lambda_i} \tag{9}$$

However, for a continuous system including two subsystems the eigenvalues for the entire system are given as

$$\det\left(\boldsymbol{I}\lambda - \begin{bmatrix} \boldsymbol{A}_1 & \boldsymbol{B}_1 \\ \boldsymbol{B}_2 & \boldsymbol{A}_2 \end{bmatrix}\right) = 0 \tag{10}$$

where the state vector for the system is defined as $\boldsymbol{x} = [\boldsymbol{x}_1,\ \boldsymbol{x}_2]^\top$. As long as $\boldsymbol{B}_i$ has at least one non-empty element, the two subsystems are dependent on each other and the eigenvalues would change in comparison to (5). In other words, it would be impossible for a subsystem developer to think of all surrounding subsystems possible in a distributed system in order to always assure stability. However, it is fairly simple to prove than when $T_d \to \infty$, the subsystems become independent on each other since the inputs become constant. This means that the global communication time step has an effect on the numerical simulation stability and must be included in the analysis. This will be elaborated in the following.

3. Numerical Stability in Distributed Systems

When studying a single system it can be determined which solver to use, and what time step to use if the solver is an explicit fixed-step size solver, as assumed here. However the subsystem inputs will affect the total distributed system dynamics and change the eigenvalues respectively. This can cause instability in the local solvers and is the topic of this section.

Two uncontrolled single linear differential systems are considered as in (3). This can be argued for since a system given as (2) including an internal linear stabilizing control law can always be rewritten as a new system given as

$$\begin{aligned} \dot{\boldsymbol{x}}_i &= \boldsymbol{A}_i\boldsymbol{x}_i + \boldsymbol{B}_i\boldsymbol{u}_i \\ \boldsymbol{y}_i &= \boldsymbol{C}_i\boldsymbol{x}_i \end{aligned} \tag{11}$$

when the control reference is assumed zero or obtained through u_i. Since the solver stability is related to the eigenvalues of a system it is of great interest to study the interconnected dynamics between subsystems and how they affect the eigenvalues.

3.1. Continuous System Analysis and Eigenvalues

A single linear differential subsystem given as in (3a) has the eigenvalue $\lambda_1 = \frac{1}{a_1}$. It can be verified that if $a_1 < 0$ and that u_1 is bounded and equal to some constant, the system is stable when assumed continuous. Moreover it is actually input-to-state stable since $\dot{x}_1 = a_1 x_1$ can be proven 0-GAS, meaning that the origin is globally asymptotically stable when $a_1 < 0$ with the storage function $V(x_1) = \frac{1}{2}x_1^2$, using Lyapunov dynamical stability analysis theory. By assuming that the Euler integration method is used to solve this subsystem, the time step requirement for a stable solution given in (9) would require that the solver time step $\Delta t_1 < \frac{-2}{a_1}$, and since the system is stable, $a_1 < 0$, the solver time step $\Delta t_1 > 0$.

Now, consider two single uncontrolled linear differential equations as given in (3) with $u_i = y_k\ \forall i \neq k$ as subsystem connections. These differential equations can be rewritten as a compact set of differential equations in continuous time under the assumption that the two subsystems interchange data continuously,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} a_1 & b_1c_2 \\ b_2c_1 & a_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{12}$$

or in an even more compact form

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} \tag{13}$$

where $\boldsymbol{x} = [x_1 \;\; x_2]^\top$. This system will always be stable when treated as an ordinary system of differential equations if all eigenvalues of $\boldsymbol{A}$ are less than zero. The eigenvalues in (12) can be calculated as in (10), and are found by solving the equation

$$(\lambda - a_1)(\lambda - a_2) - b_1 b_2 c_1 c_2 = 0 \tag{14}$$

which gives the eigenvalues

$$\lambda_{1,2} = \frac{1}{2}\left[a_1 + a_2 \pm \sqrt{(a_1 - a_2)^2 + 4 b_1 b_2 c_1 c_2}\right] \tag{15}$$

This clearly shows that when considering the two subsystem as one continuous system, the eigenvalues for the total system are different from the eigenvalues in each subsystem, as long as $\{b_1, b_2, c_1, c_2\} \neq 0$. Hence, when two or more subsystems are connected, which have own local solvers and continuously exchange data, the stability of the local solvers would depend on the interacting dynamics between subsystems. This is illustrated in the following example.

Example 1 (Two single linear differential subsystems)**.** *Consider two single linear differential subsystems given as*

$$\begin{aligned} \dot{x}_1 &= -2x_1 + u_1 \\ y_1 &= x_1 \end{aligned} \tag{16}$$

and

$$\begin{aligned} \dot{x}_2 &= -1.5x_2 + 0.5u_2 \\ y_2 &= x_2 \end{aligned} \tag{17}$$

These two systems have the eigenvalues $\lambda_1 = -2$ and $\lambda_2 = -1.5$, respectively, and can be solved separately by Euler integration with the time steps $\Delta t_1 \leq 1.0$ and $\Delta t_2 \leq \frac{4}{3}$, respectively, if u_i is considered constant. Now, consider the two subsystems as one continuous system, meaning that $u_1 = y_2 = x_2$ and $u_2 = y_1 = x_1$, such that

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ 0.5 & -1.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{18}$$

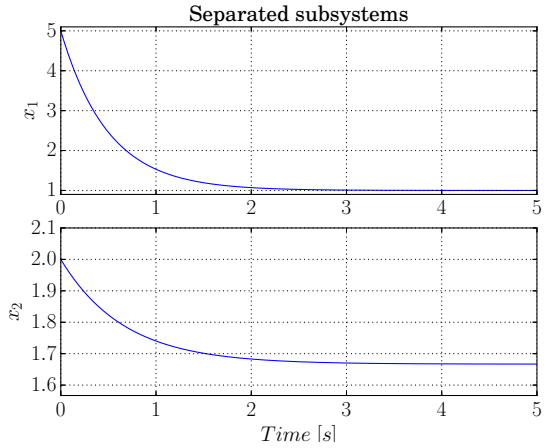
The two eigenvalues for the total system are given as $\lambda_1 = -1.0$ and $\lambda_2 = -2.5$. This means that if the total system is to be solved using Euler integration, $\Delta t \leq \frac{2}{2.5} = 0.8$. This will give a stricter requirement for Δt for the total system compared to the separated subsystems in the distributed system.

The two different system settings are solved using the Euler integration method and the initial conditions are set to $x_1(0) = 5$ and $x_2(0) = 2$. The first simulation shows the results from when the two subsystems are separated and the subsystem inputs are set as $u_1 = x_2(0)$ and $u_2 = x_1(0)$, meaning that u_1 and u_2 are constant and equal to the initial conditions. Figure 2 shows the simulation results with the two different time-step settings.

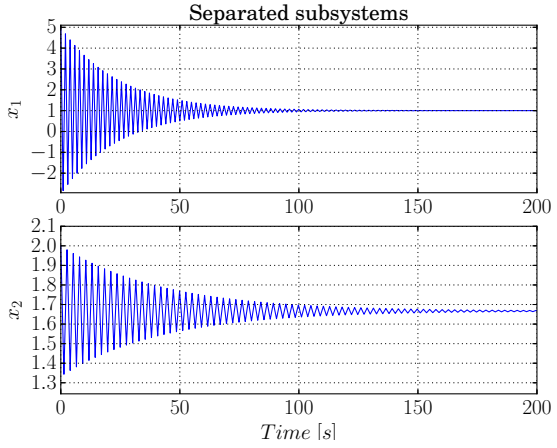
Figure 2a shows the simulation results when the time steps are set to 0.01 s and in Figure 2b the two different time steps are set close to the stability region for the forward Euler integration method, $\Delta t_1 = 0.980\,\mathrm{s}$ and $\Delta t_2 = 1.313\,\mathrm{s}$. Both x_1 and x_2 converge to the same values as in Figure 2a but now the time it takes to converge is increased significantly, about 100 s and 200 s, respectively.

In Figure 3 the simulation results for the system given in (18) are shown when the total system is solved by the Euler integration method with two different time steps. The initial conditions are the same as in the previous case. As can be seen in Figure 3a both states converge to 0 in approximately 5 seconds and the simulation results look smooth when $\Delta t = 0.01\,\mathrm{s}$. Figure 3b shows the simulation results when $\Delta t = 0.78\,\mathrm{s}$, close to the forward Euler integration stability limit. The states still converge to 0 but the time it takes to do that is also in this case significantly increased.

Example 1 shows that the maximal time step allowed for stable solutions for the two subsystems, considered as an ordinary system of differential equations, is lower than the ones allowed when solving the subsystems separately. One could argue that by choosing the local solver time-steps as small as possible the simulation results from the total system would also become stable. However, it is not certain that the combined dynamics are stable, and even if they

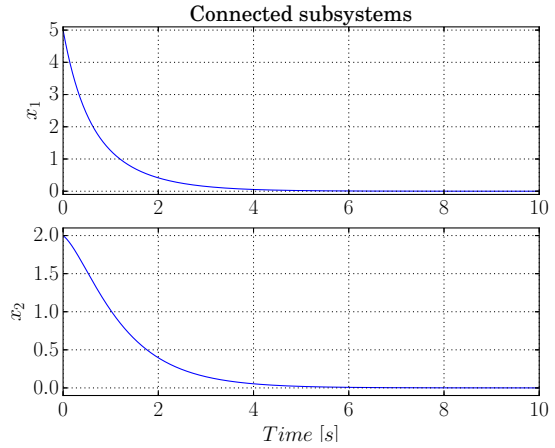


(a) $\Delta t_1 = \Delta t_2 = 0.01$ s

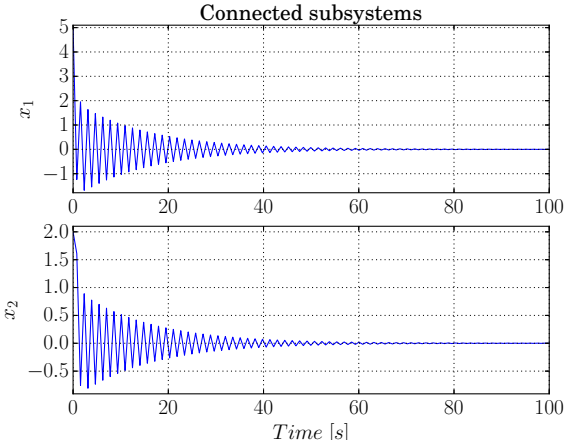


(b) $\Delta t_1 = 0.980$ s, $\Delta t_2 = 1.313$ s

Figure 2: Simulation results of separated subsystems with different time steps.



(a) $\Delta t = 0.01$ s



(b) $\Delta t = 0.78$ s

Figure 3: Simulation results of connected subsystems with different time steps.

are, the time it takes to solve the total system would increase significantly. It is then reasonable to believe that by decreasing the communication rate in a distributed system, the interconnected dynamics get weakened and the total simulation results in the distributed system may remain stable. This is summarized in Lemma 1.

Lemma 1. *A set of subsystems in a distributed system, having their own local fixed step size solvers with large time steps that keep the unconnected subsystems stable by themselves, may not result in stable solutions in a distributed setting due to the interconnected dynamics when the global distributed time step goes to zero, but will become stable when the global time step is set large enough.*

To be able to stabilize the local solvers, one must find a global time step T_d that reduces the effects of the interacting dynamics such that each local solver time step is within the solver's stability limit. However, it is believed that there may be some restric-

tions, when choosing the global time step, related to sampling and signal processing theory. These topics will be studied in the following.

3.2. Stability Analysis of Distributed System with respect to Local Solvers

The example and the discussion in the previous section, roughly summarized in Lemma 1, give a good idea of the goal in this section. However before starting with the stability analysis, a few thoughts and comments regarding expected restrictions need to be mentioned. To be more specific, Lemma 1 points to a solution of the stability problem where the phenomena of *aliasing* is utilized. Aliasing in sampling is when the sampling frequency is chosen too low such that the sampled data fails to represent the sampled system, see Figure 4. This means that we want to

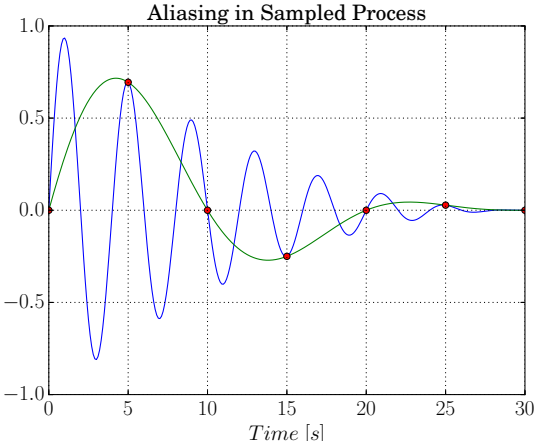


Figure 4: Aliasing in sampled system. The blue graph represents the process to be sampled, the read dots are the sampled values and the green graph is the recreated process from the sampled data.

set T_d large, such that the communication frequency becomes low enough for the local solvers to become stable. However, aliasing often introduces distortions and numerical errors, in this case also difference in global time when subsystems exchange data, which will be given more attention to later on. The question that now comes to mind is why we in the first place want to utilize aliasing and fail to represent the system interactions in a proper manner. The answer is however rather simple. If we do not have the opportunity to choose the local time steps in the submodels, there may exist system configurations where the solution of the total system becomes unstable. We then have two opportunities. We can choose to not use the system in simulations because we are not allowed to change the local time-steps, or, if we are not interested in transient simulation results, we may choose T_d such that we at least may be able to obtain a correct and stable solution after an incorrect transient simulation period. In many cases the latter is the preferred option.

Before moving on with the analysis we need to define the concepts *local* and *global solutions*. A local solution is here defined as the solution obtained locally in one connected subsystem, in comparison to a global solution, which is the collection of local solutions sampled with T_d. In this study the stability of both these solutions would be of interests since they strongly depend on each other, meaning that if all local solutions are stable with a given T_d, so is the global solution. However we cannot guarantee that the local solutions are stable based on the fact that the global solution is stable due to the aliasing side effects.

Given a single linear differential equation as in (3a), representing subsystem i in a distributed simulation, is to be solved locally with the forward Euler integration method, as in (7). Hence, the numerical solution of such a differential equation can be calculated as

$$x_i(k+1) = x_i(k) + \Delta t_i \left(a_i x_i(k) + b_i u_i(k)\right) \tag{19}$$

for a given time step k. By defining that

$$n_i := \frac{T_d}{\Delta t_i}, \tag{20}$$

meaning that the subsystem does not communicate with the rest of the distributed system before n_i local time-steps have been taken, we know that if k was the last communication point,

$$u_i(k) = u_i(k+1) = \ldots = u_i(k+n_i) \tag{21}$$

Since the subsystem i is not communicating with the rest of the distributed system in the time interval

$t \in [k, k+n_i]$, the system only depends on itself and the last input $u_i(k)$. Hence,

$$\begin{aligned} x_i(k+n_i) &= x_i(k+n_i-1) \\ &\quad + \Delta t_i \left(a_i x_i(k+n_i-1) + b_i u_i(k)\right) \\ &= (1+\Delta t_i a_i) x_i(k+n_i-1) + \Delta t_i b_i u_i(k) \\ &= (1+\Delta t_i a_i)^2 x_i(k+n_i-2) \\ &\quad + \left[(1+\Delta t_i a_i)+1\right] \Delta t_i b_i u_i(k) \\ &= (1+\Delta t_i a_i)^3 x_i(k+n_i-3) \\ &\quad + \left[(1+\Delta t_i a_i)^2 + (1+\Delta t_i a_i) + 1\right] \Delta t_i b_i u_i(k) \\ &= ... \\ &= (1+\Delta t_i a_i)^{n_i} x_i(k) \\ &\quad + \sum_{j=1}^{n_i} (1+\Delta t_i a_i)^{j-1} \Delta t_i b_i u_i(k) \end{aligned} \tag{22}$$

The sum in (22) can be recognized as a known geometric progression,

$$\sum_{j=1}^{n} q^{j-1} = \frac{1-q^n}{1-q}, \tag{23}$$

and if $a_i \neq 0$

$$\sum_{j=1}^{n_i} (1+\Delta t_i a_i)^{j-1} = \frac{(1+\Delta t_i a_i)^{n_i} - 1}{\Delta t_i a_i} \tag{24}$$

By inserting $k := t$, where t is the progressing time, and $k + n_i = t + T_d$, the global solution of the solved subsystem can be expressed as

$$\begin{aligned} x_i(t+T_d) &= (1+\Delta t_i a_i)^{n_i} x_i(t) \\ &\quad + \left[(1+\Delta t_i a_i)^{n_i} - 1\right] \frac{b_i}{a_i} u_i(t) \end{aligned} \tag{25}$$

These results are similar to the solution of discrete-time equations given in [5], and for comparison the solution for a continuous-time state equation with sampled input is given as

$$\begin{aligned} x_i(t+T_d) &= e^{a_i T_d} x_i(t) \\ &\quad + \left(\int_0^{T_d} e^{a_i (T_d - \tau)} d\tau \right) b_i u_i(t) \\ &= e^{a_i T_d} x_i(t) + \left(e^{a_i T_d} - 1\right) \frac{b_i}{a_i} u_i(t) \end{aligned} \tag{26}$$

It can be seen that when $\Delta t_i \to 0$, (25)$\to\approx$(26) where n_i is given as in (20), when neglecting higher order terms in the Tailor expansion of the exponential function. By including the output mapping given as in (3), we might for simplicity rewrite subsystem i in (25) as

$$\begin{aligned} x_i(t+T_d) &= a_{n_i} x_i(t) + b_{n_i} u_i(t) \\ y_i(t) &= c_i x_i(t) \end{aligned} \tag{27}$$

where

$$\begin{aligned} a_{n_i} &:= \begin{cases} (1+\Delta t_i a_i)^{n_i} & \text{for } a_i \neq 0 \\ 1 & \text{for } a_i = 0 \end{cases} \\ b_{n_i} &:= \begin{cases} \frac{b_i}{a_i}(a_{n_i} - 1) & \text{for } a_i \neq 0 \\ T_d b_i & \text{for } a_i = 0 \end{cases} \end{aligned} \tag{28}$$

Equivalently, when a subsystem contains a set of differential equations, we may rewrite (27) as

$$\begin{aligned} \boldsymbol{x}_i(t+T_d) &= \boldsymbol{A}_{n_i} \boldsymbol{x}_i(t) + \boldsymbol{B}_{n_i} \boldsymbol{u}_i(t) \\ \boldsymbol{y}_i(t) &= \boldsymbol{C}_i \boldsymbol{x}_i(t) \end{aligned} \tag{29}$$

where

$$\begin{aligned} \boldsymbol{A}_{n_i} &:= (\boldsymbol{I} + \Delta t_i \boldsymbol{A}_i)^{n_i} \\ \boldsymbol{B}_{n_i} &:= \boldsymbol{A}_i^{-1} (\boldsymbol{A}_{n_i} - \boldsymbol{I}) \boldsymbol{B}_i, \quad \boldsymbol{A}_i \text{ is nonsingular.} \end{aligned} \tag{30}$$

and where $\boldsymbol{x}_i \in \boldsymbol{\mathcal{R}}^m$, $\boldsymbol{A}_i \in \boldsymbol{\mathcal{R}}^{m \times m}$, $\boldsymbol{u}_i \in \boldsymbol{\mathcal{R}}^p$, $\boldsymbol{B}_i \in \boldsymbol{\mathcal{R}}^{m \times p}$, $\boldsymbol{y}_i \in \boldsymbol{\mathcal{R}}^r$ and $\boldsymbol{C}_i \in \boldsymbol{\mathcal{R}}^{r \times m}$, meaning that there are m states, p inputs and r outputs in subsystem i. If $\boldsymbol{A}_{n_i}$ is singular, similar requirement as established in (28) can be applied. Note that some of the subscripts have been neglected because for a matrix $\boldsymbol{C}_i$, size$(\boldsymbol{C}_i)$ = size$(\boldsymbol{C}_{n_i})$.

Now, let us assume that a given distributed system contains s linear subsystems, with solutions given as either (27) or (29). These subsystems are connected together in the total distributed system through a predefined connection configuration, typically

$$\boldsymbol{u}_i = \boldsymbol{M}_i \boldsymbol{y}_i \tag{31}$$

where $\boldsymbol{M}_i$ is a mapping matrix, and we might define the global solutions in the distributed system as

$$\boldsymbol{x}_d(t+T_d) = (\boldsymbol{A}_d + \boldsymbol{B}_d) \boldsymbol{x}_d(t) \tag{32}$$

where x_d denotes all states in the distributed system,

$$\boldsymbol{A}_d := \text{diag}(\boldsymbol{A}_{n_1}, \boldsymbol{A}_{n_2}, ..., \boldsymbol{A}_{n_s}) \tag{33}$$

and $\boldsymbol{B}_d$ is a mapping matrix between outputs and inputs with the diagonal equal to zero. Since $\boldsymbol{A}_d$ and $\boldsymbol{B}_d$ have the same size we might define the solution mapping matrix as

$$\boldsymbol{S}_d = \boldsymbol{A}_d + \boldsymbol{B}_d \tag{34}$$

such that

$$\boldsymbol{x}_d(t + T_d) = \boldsymbol{S}_d \boldsymbol{x}_d(t) \tag{35}$$

To clarify, the solution mapping matrix for a distributed system containing two single linear differential equations such as (3), each implemented as a subsystem, is then given as

$$\boldsymbol{S}_d = \begin{bmatrix} a_{n_1} & b_{n_1} c_2 \\ b_{n_2} c_1 & a_{n_2} \end{bmatrix} \tag{36}$$

where a_{n_i} and b_{n_1} are defined as in 28. Note that a similar expression for $\boldsymbol{S}_d$ can be found when $y_i(t) = cx_i(t) + du_i(t)$, where d is assumed a constant mapping value.

In order for a general distributed system to have a stable global solution, we must first assure that

$$|\boldsymbol{S}_d| \leq 1 \tag{37}$$

by choosing T_d. In other words, we must choose T_d such that all eigenvalues in $\boldsymbol{S}_d$ have an amplitude with absolute value less than, or equal to, 1. However it is actually not enough to guarantee that the solution mapping matrix has decreasing characteristics. This has to do with possible differences between the local subsystem propagating times and the global propagating time and will be discussed more in detail later on.

The example below is a revised version of Example 1 where the derived properties are tested.

Example 2 (Example 1 revised)**.** *By using the same parameters as before, we can write the solution mapping matrix for the system given in Example 1 as*

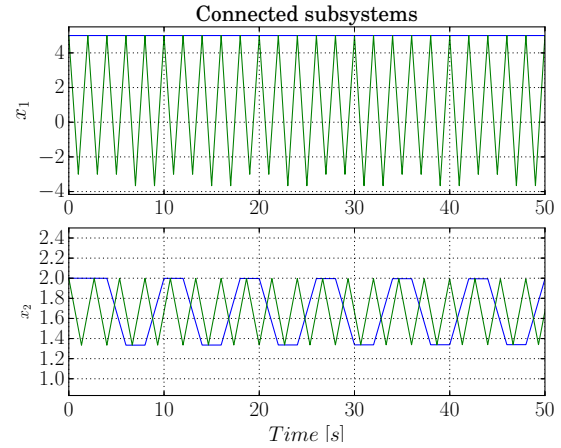
$$\boldsymbol{S}_d = \begin{bmatrix} (1 - 2\Delta t_1)^{n_1} & -\frac{1}{2}[(1 - 2\Delta t_1)^{n_1} - 1] \\ -\frac{1}{3}[(1 - 1.5\Delta t_2)^{n_2} - 1] & (1 - 1.5\Delta t_2)^{n_2} \end{bmatrix} \tag{38}$$

It can be verified that by setting $\Delta t_1 = 1$ and $\Delta t_2 = \frac{4}{3}$, meaning that the local solvers are only marginally stable, the absolute values of the two eigenvalues in $\boldsymbol{S}_d$ are equal to 1 when $T_d = 2$. This gives $n_1 = 2$ and $n_2 = 1.5$ which means that subsystem 1 is allowed 2 local steps and subsystem 2 is allowed 1.5 local steps between each data exchange. This means that subsystem 2 does not always have its local propagating time synchronized with the global propagating time, which may introduce errors causing the distributed system to become unstable. Figure 5 shows the simulation results for the two connected subsystems.

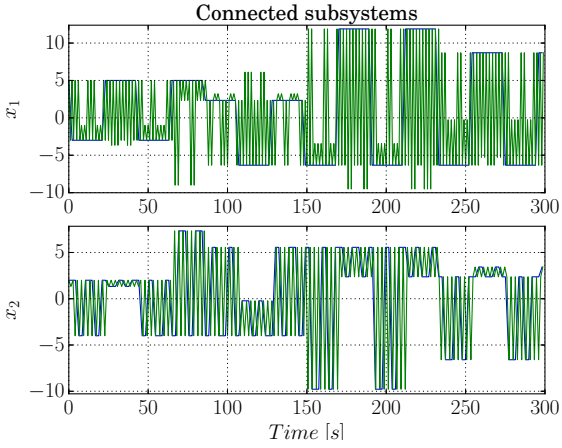
The first plot in Figure 5a shows the global solution, blue graph, compared to the local solution, green graph, for subsystem 1. Due to the sampling rate, the communication frequency between the subsystem, the global solution is constant in comparison to the local solution which oscillates and is barely affected by subsystem 2. This is as expected when the global time step is chosen twice as large as the local time step and the local solver is marginally stable. However the local propagating time is always synchronized with the global propagating time. The second plot in Figure 5a shows a rather different picture. Here, the global solution oscillates and with a different frequency than the local solution. However, the total system is stable because the global solution of subsystem 1 is constant and does not excite subsystem 2 in any particular way other than giving an offset.

To illustrate more in detail the effect of unsynchronized propagating times the global time step is changed to $T_d = 2.1\,\text{s}$ which gives $|eig(\boldsymbol{S}_d)| = \{0.9877,\ 0.9560\}$. Figure 5b shows the corresponding simulation results. Clearly, the total distributed system is unstable, which indicates that synchronized propagating time is closely related to stability in this setting.

Example 2 shows that synchronization of the propagating times is crucial for global solution stability. It can also be shown in the derivation of the solution mapping matrix where it is actually assumed that the times are synchronized without giving it any thoughts. Based on the derived stability criterion and the experiences gained in Example 2 the following



(a) $\Delta t_1 = 1.0$ s, $\Delta t_2 = 1.333$ s, $T_d = 2.0$ s



(b) $\Delta t_1 = 1.0$ s, $\Delta t_2 = 1.333$ s, $T_d = 2.1$ s

Figure 5: Simulation results of two subsystems, with marginally stable local solvers, connected in a distributed system. Green graph denotes local solution and blue graph denotes global solution.

theorem can be established.

Theorem 1 (Convergence of local and global solutions in distributed systems.)**.** *Given a set of linear subsystems in a distributed system, each given as*

$$\begin{aligned} \dot{\boldsymbol{x}}_i &= \boldsymbol{A}_i \boldsymbol{x}_i + \boldsymbol{B}_i \boldsymbol{u}_i \\ \boldsymbol{y}_i &= \boldsymbol{C}_i \boldsymbol{x}_i \end{aligned} \tag{39}$$

that are solved locally by fixed step size solvers, such as the forward Euler integration method with a time step Δt_i, and by assuming that each local propagating time is synchronized with the global propagating time, the global solution of subsystem i can be expressed as

$$\begin{aligned} \boldsymbol{x}_i(t + T_d) &= \boldsymbol{A}_{n_i} \boldsymbol{x}_i(t) + \boldsymbol{B}_{n_i} \boldsymbol{u}_i(t) \\ \boldsymbol{y}_i(t) &= \boldsymbol{C}_i \boldsymbol{x}_i(t) \end{aligned} \tag{40}$$

where $\boldsymbol{A}_{n_i}$ and $\boldsymbol{B}_{n_i}$ are solver dependent matrices (for the forward Euler integration method see (30)*) where*

$$n_i := \frac{T_d}{\Delta t_i} \tag{41}$$

and T_d is the global step size in the distributed system. By collecting the subsystems and applying the given connection setup, the total global solution of the distributed system may be expressed as

$$\boldsymbol{x}_d(t + T_d) = \boldsymbol{S}_d(\boldsymbol{A}_{n_i}, \boldsymbol{B}_{n_i}, \boldsymbol{C}_i) \boldsymbol{x}_d(t) \tag{42}$$

where $\boldsymbol{S}_d(\cdot)$ is denoted the solution mapping matrix for the total distributed system. Then, if T_d can be chosen such that

$$|eig(\boldsymbol{S}_d(\cdot))| \leq 1 \tag{43}$$

then both the local and the global solution will be stable. Moreover, if

$$|eig(\boldsymbol{S}_d(\cdot))| < 1 \tag{44}$$

the global steady state solution of the distributed system will converge to the local steady state solutions.

Proof. The proof is equal to the derivation of $\boldsymbol{S}_d(\cdot)$, where the forward Euler integration method is used, when assuming synchronized propagating times. The proof is similar when other explicit integration methods are used. □

Definition 1. *A distributed system is said to have synchronized local and global propagating times when $n_i \in \mathcal{N}_{\geq 1}$ for each subsystem i, where $\mathcal{N}_{\geq 1}$ denotes all integers larger than, or equal to 1.*

As can be seen in Theorem 1 we do not need to specify that each subsystem is locally stable, both in the dynamics and in the solution. This has to do with

the fact that both the solver characteristics and the system dynamics are already included in the stability criteria.

Example 3 shows the use of Theorem 1.

Example 3 (Example 2 continued)**.** *Assuming a distributed system is given as in Example 2, but now* $\Delta t_1 = 0.9$, $\Delta t_2 = 0.9$ *and* $T_d = 1.8$, *giving* $n_1 = n_2 = 2$. *Note that these local time steps guarantee that the local solutions will be stable by themselves when using the Euler integration method. Then the amplitudes of the eigenvalues in the solution mapping matrix are given as* $|eig(\boldsymbol{S}_d)| = \{0.7271,\ 0.0354\}$. *Since* $|eig(\boldsymbol{S}_d)| < 1$ *and* $n_i \in \mathcal{N}_{\geq 1}\ \forall i$, *the steady state global solution will converge to the local solutions according to Theorem 1. The simulation results are shown in Figure 6. As can be seen in the figure the steady state global solution converges to the two steady state local solutions. Hence, the total system solution is stable.*

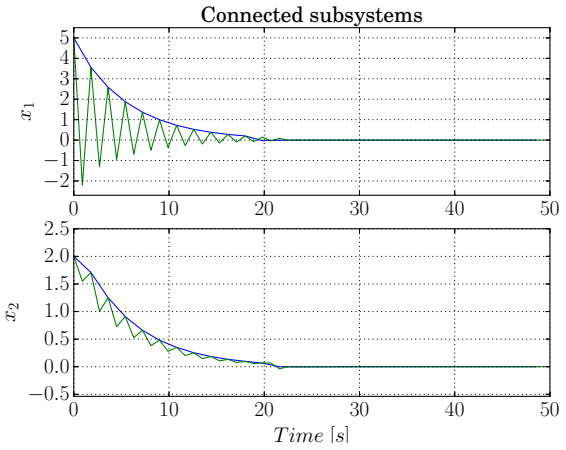


Figure 6: Simulation result of two subsystems with stable local solvers connected as in a distributed system. Green graph denotes local solution and blue graph denotes global solution.

As can be seen in Example 3 the global solution converges to the local solutions when the propagating local times are synchronized with the global propagating time. However Theorem 1 may be relaxed with respect to time synchronization when the local solvers are robust. When a local solver is only marginally stable we must assure that the propagating times are synchronized in order to guarantee stability. This is a measure of robustness of solutions and when $\Delta t_i \to 0$ we might choose T_d arbitrary, as long as the subsystems have certain passivity properties, due to the overhead in the local solvers.

3.3. Global Distributed System Stability

Until now only distributed systems containing subsystems with stable dynamics and stable local solvers have been studied. However Theorem 1 guarantees stability in a distributed system containing unstable subsystems or subsystems without certain passivity properties as long as the propagating times are synchronized. This means that we also can use Theorem 1 to guarantee stability if subsystems in a distributed system do not have certain passivity properties. It is also possible to determine stability limits for a distributed system containing one or more subsystems that have unstable dynamics, that are dependent on surrounding stabilizing systems in order for the total distributed simulation to become stable. This is shown in Example 4.

Example 4 (Unstable Subsystem)**.** *Two linear single differential systems are given as*

$$\begin{aligned} \dot{x}_1 &= -x_1 + u_1 \\ \dot{x}_2 &= 0.1x_2 - u_2 \end{aligned} \tag{45}$$

It can be verified that the eigenvalues for the continuously connected systems are given as $\lambda_{1,2} = \frac{-9 \pm 3i\sqrt{31}}{20}$, *and are therefore stable. However, when solving the two systems separately in a distributed manner, system 2 would become unstable when* T_d *becomes arbitrarily large. The Euler integration method is used to solve both systems locally, and the time steps are set to* $\Delta t_1 = \Delta t_2 = 0.01$, *respectively. It can be verified that when choosing* $T_d = 0.89$, *giving* $n_1 = n_2 = 89$, $|eig(\boldsymbol{S}_d)| = \{0.9984,\ 0.9984\}$. *Hence, from Theorem 1 the total distributed system is stable. The simulation results from the system when* $x_1(0) = 5$ *and* $x_2(0) = 2$ *are shown in Figure 7.*

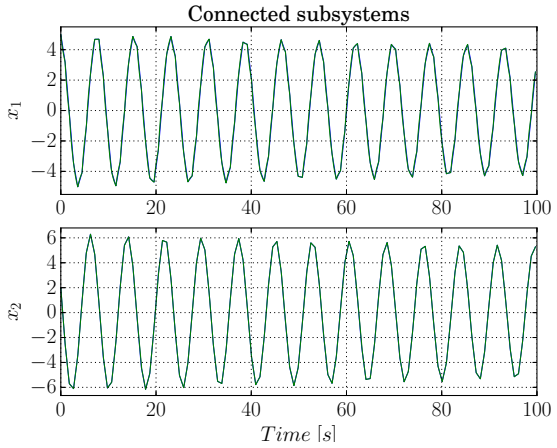


Figure 7: Simulation result of speed regulated mass-damper-spring system implemented in a distributed system. Green graph denotes local solution and blue graph denotes global solution.

3.4. Explicit Fixed Step Size Solvers

So far, only the forward Euler integration method has been studied. However, at least two more fixed step size solvers, that are quite popular, should be studied in the context of distributed simulation stability, namely the Runge-Kutta 2 integration method, RK2, and the Runge-Kutta 4 integration method, RK4.

3.4.1. Runge-Kutta 2

The classical RK2 integration method is known by many names like Heun's method and Improved Euler integration method. For a general single differential equation i given as

$$\dot{x}_i = f(t, x_i) \tag{46}$$

the classical RK2 algorithm is given as

$$\begin{aligned} k_1 &= f(t(j), x_i(j)) \\ k_2 &= f(t(j) + \Delta t_i, x_i(j) + \Delta t_i k_1) \\ x_i(j+1) &= x_i(j) + \frac{\Delta t_i}{2}(k_1 + k_2) \end{aligned} \tag{47}$$

Assuming a single linear differential equation given as

$$\dot{x}_i = a_i x_i + b_i u_i \tag{48}$$

it can be shown that the solution for time step $j + n_i$ is given as

$$x_i(j + n_i) = a_{n_i} x_i(j) + b_{n_i} u_i(j) \tag{49}$$

where

$$\begin{aligned} a_{n_i} &:= \begin{cases} (1 + a_i \Delta t_i + \frac{a_i^2}{2} \Delta t_i^2)^{n_i} & \text{for } a_i \neq 0 \\ 1 & \text{for } a_i = 0 \end{cases} \\ b_{n_i} &:= \begin{cases} (a_{n_i} - 1)\frac{b_i}{a_i} & \text{for } a_i \neq 0 \\ T_d b_i & \text{for } a_i = 0 \end{cases} \end{aligned} \tag{50}$$

If subsystem i contains a set of linear differential equations, then

$$\boldsymbol{x}_i(j + n_i) = \boldsymbol{A}_{n_i} \boldsymbol{x}_i(j) + \boldsymbol{B}_{n_i} \boldsymbol{u}_i(j) \tag{51}$$

where

$$\begin{aligned} \boldsymbol{A}_{n_i} &= (\boldsymbol{I} + \boldsymbol{A}_i \Delta t_i + \frac{1}{2} \boldsymbol{A}_i^2 \Delta t_i^2)^{n_i} \\ \boldsymbol{B}_{n_i} &= \boldsymbol{A}_i^{-1} (\boldsymbol{A}_{n_i} - \boldsymbol{I}) \boldsymbol{B}_i, \quad \boldsymbol{A}_{n_i} \text{ nonsingular} \end{aligned} \tag{52}$$

When the RK2 integration method is used, $\boldsymbol{A}_{n_i}$ and $\boldsymbol{B}_{n_i}$ given in (52) are used in Theorem 1 to construct the solution mapping matrix. Note that when $\boldsymbol{A}_{n_i}$ is singular, similar results as found in (50) can be applied.

3.4.2. Runge-Kutta 4

For a linear differential equation as given in (46) the RK4 algorithm may be given as

$$\begin{aligned} k_1 &= \Delta t_i f(t(j), x_i(j)) \\ k_2 &= \Delta t_i f(t(j) + \frac{\Delta t_i}{2}, x_i(j) + \frac{k_1}{2}) \\ k_3 &= \Delta t_i f(t(j) + \frac{\Delta t_i}{2}, x_i(j) + \frac{k_2}{2}) \\ k_4 &= \Delta t_i f(t(j) + \Delta t_i, x_i(j) + k_3) \\ x_i(j+1) &= x_i(j) + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} \end{aligned} \tag{53}$$

By assuming a single linear differential equation as given in (48), it can be shown that the solution for time step $j + n_i$ is given as

$$x_i(j + n_i) = a_{n_i} x_i(j) + b_{n_i} u_i(j) \tag{54}$$

Table 1: Summary of solver dependent matrices. Note that $\boldsymbol{S}_i$ is given as in (59). Note that *FE* and *RK* in the table are an abbreviations for the forward Euler integration method and the Runge-Kutta integration method, respectively

Solver	$\boldsymbol{A}_{n_i}$	$\boldsymbol{B}_{n_i}$
FE	$(\boldsymbol{I}+\Delta t_i\boldsymbol{A}_i)^{n_i}$	$\boldsymbol{A}_i^{-1}(\boldsymbol{A}_{n_i}-\boldsymbol{I})\boldsymbol{B}_i$
RK2	$(\boldsymbol{I}+\Delta t_i\boldsymbol{A}_i+\frac{1}{2}\Delta t_i^2\boldsymbol{A}_i^2)^{n_i}$	$\boldsymbol{A}_i^{-1}(\boldsymbol{A}_{n_i}-\boldsymbol{I})\boldsymbol{B}_i$
RK4	$(\boldsymbol{I}+\boldsymbol{S}_i)^{n_i}$	$\boldsymbol{A}_i^{-1}(\boldsymbol{A}_{n_i}-\boldsymbol{I})\boldsymbol{B}_i$

where

$$\begin{aligned} a_{n_i} &:= \begin{cases} (1+s_i)^{n_i} & \text{for} a_i \neq 0 \\ 1 & \text{for } a_i = 0 \end{cases} \\ b_{n_i} &:= \begin{cases} (a_{n_i}-1)\frac{b_i}{a_i} & \text{for } a_i \neq 0 \\ T_d b_i & \text{for } a_i = 0 \end{cases} \end{aligned} \tag{55}$$

and

$$s_i := \frac{a_i^4}{24}\Delta t^4 + \frac{a_i^3}{6}\Delta t^3 + \frac{a_i^2}{2}\Delta t^2 + a_i\Delta t_i \tag{56}$$

If subsystem i contains a set of linear differential equations, then

$$\boldsymbol{x}_i(j+n) = \boldsymbol{A}_{n_i}\boldsymbol{x}_i(j) + \boldsymbol{B}_{n_i}\boldsymbol{u}_i(j) \tag{57}$$

where

$$\begin{aligned} &\boldsymbol{A}_{n_i} = (\boldsymbol{I}+\boldsymbol{S}_i)^{n_i} \\ &\boldsymbol{B}_{n_i} = \boldsymbol{A}_i^{-1}(\boldsymbol{A}_{n_i}-\boldsymbol{I})\boldsymbol{B}_i, \quad \boldsymbol{A}_{n_i} \text{ nonsingular} \end{aligned} \tag{58}$$

and

$$\boldsymbol{S}_i := \frac{1}{24}\boldsymbol{A}_i^4\Delta t^4 + \frac{1}{6}\boldsymbol{A}_i^3\Delta t^3 + \frac{1}{2}\boldsymbol{A}_i^2\Delta t^2 + \boldsymbol{A}_i\Delta t_i \tag{59}$$

When the RK4 integration method is used, $\boldsymbol{A}_{n_i}$ and $\boldsymbol{B}_{n_i}$ given in (58) are used in Theorem 1 to construct the solution mapping matrix. If $\boldsymbol{A}_{n_i}$ is singular, similar results as found in (55) can be used. A summary of solver dependent system matrices are given in Table 1.

The following example illustrates the use of two different solvers applied on a mass-damper-spring system that is affected by a speed regulator.

Example 5 (Distributed mass–damper-spring system with speed regulator)**.** *A mass-damper-spring system as given in Figure 8 is to be implemented as a subsystem in a distributed system, and the differential equations describing its dynamics are given as*

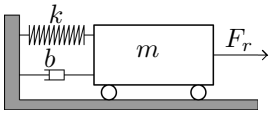


Figure 8: Mass-damper-spring system.

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{1}{m}(F_r - bx_2 - kx_1) \end{aligned} \tag{60}$$

where x_1 is the position, x_2 is the speed and F_r is the speed regulator force. This regulator force is assumed to be another subsystem in the distributed system given as

$$\dot{F}_r = K_I F_r + K_P x_2 \tag{61}$$

and is comparable to a PI-controller where the speed reference is set to zero. We might assume that the mass-damper-spring dynamics are solved with the Euler integration method and the regulator with the RK2 integration method. Hence, we might write

$$\boldsymbol{S}_d = \begin{bmatrix} \boldsymbol{A}_{n_1} & \boldsymbol{B}_{n_1} \\ 0, b_{n_2} & a_{n_2} \end{bmatrix} \tag{62}$$

where

$$\boldsymbol{A}_{n_1} = \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \Delta t_1 \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} \right)^{n_1} \tag{63}$$

$$\boldsymbol{B}_{n_1} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix}^{-1} \left(\boldsymbol{A}_{n_1} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \tag{64}$$

$$a_{n_2} = \left(1 + K_I\Delta t_2 + \frac{K_I^2}{2}\Delta t_2^2 \right)^{n_2} \tag{65}$$

and

$$b_{n_2} = (a_{n_2} - 1)\frac{K_P}{K_I} \tag{66}$$

where n_i is as defined in (20). By setting $m = 2$, $k = 10$, $b = 1$, $K_I = -0.1$, $K_P = -0.5$, $\Delta t_1 = 0.01$ and $\Delta t_2 = 0.1$, it can be verified that $T_d = 1.0$ gives $n_1 = 100$, $n_2 = 10$ and $|eig(\boldsymbol{S_d})| = \{0.8424,\ 0.8424,\ 0.9075\}$, which means that the distributed system is stable when solved. The initial values are set as $x_1(0) = 5$, $x_2(0) = 0$ and $F_r(0) = 2$, and the simulation results are shown in Figure 9.

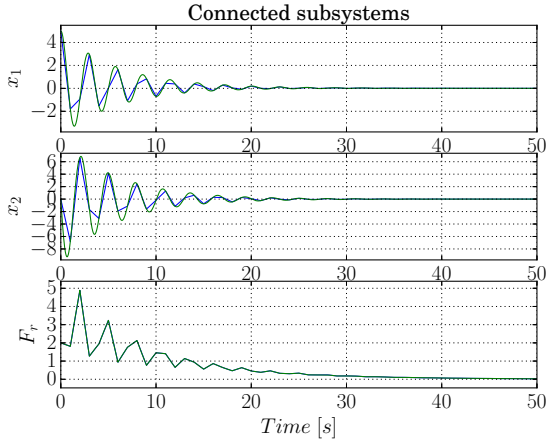


Figure 9: Simulation results showing the speed controlled mass-damper-spring system. Green graph denotes local solution and blue graph denotes global solution.

It should be mentioned that the control law would have preformed much better if $K_P > 0$ which would have put F_r and x_2 in opposite phases, not equal as shown in the figure.

As seen in the example, it can be more time consuming to calculate the solution matrix when using higher order explicit solvers in comparison to first order solvers such as the forward Euler integration method, at least when the total system becomes large. However, if only a conservative stability result is required it is possible to assure stability by using a lower order solver in the analysis as long as the stability regions of the solvers overlap. This is summarized in Corollary 1.

Corollary 1. *A conservative stability analysis can be performed for a distributed system containing higher order solvers by reducing the order of the solvers. In particular, a conservative stability result can be found for distributed systems containing higher order explicit Runge-Kutta integration methods by assuming that the forward Euler integration method is used in the simulation.*

Proof. Since the higher order explicit Runge-Kutta integration methods contain the stability region for the forward Euler integration method, simulation results from a distributed system containing higher order Runge-Kutta integration methods would be stable if stability is assured when using the forward Euler integration method. □

3.5. Towards Nonlinear Systems

So far, much attention has been given to distributed linear dynamical systems. However, as it turns out, the numerical stability criteria for linear dynamical systems can also be applied to nonlinear systems, although generating more conservative results. This is because linearization of the nonlinear system, and an analysis of the eigenvalues, are required. To illustrate this, assume that a nonlinear dynamical system is given as

$$\begin{aligned} \dot{x} &= -x^3 + u \\ y &= x + u \end{aligned} \tag{67}$$

It can be shown that the linearized system is given as

$$\begin{aligned} \dot{\Delta x} &= -3x_0^2 \Delta x + u \\ x &= x_0 + \Delta x \\ y &= x + u \end{aligned} \tag{68}$$

where x_0 is the operating point for the linearized system. The linearized system has a range of eigenvalues, given as a function of x_0,

$$\lambda(x_0) = -3x_0^2 \tag{69}$$

which in this case only contains negative eigenvalues. This is of no surprise since the nonlinear dynamics are stable when the input is set to zero. It is then possible to specify a validity range of the nonlinear system, such that the system is only valid for a finite

range of values for x, typically $x \in [x_l, x_u]$ where the subscript $_l$ stands for the lower limit and subscript $_u$ stands for the upper limit. If $x_l = -x_u$ for the system given in (67), the eigenvalues for the system are in the range $\lambda \in [-3x_u^2, 0]$. This means that when using for example the Euler integration method, a simulation of the nonlinear system will be stable as long as $\Delta t \leq \frac{2}{3x_u^2}$.

In general, for a nonlinear system given as

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \\ \boldsymbol{y} &= \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}) \end{aligned} \tag{70}$$

the linearized system can for the operation points $\boldsymbol{x}_0$, $\boldsymbol{u}_0$ and $\boldsymbol{y}_0$ be expressed as

$$\begin{aligned} \dot{\Delta \boldsymbol{x}} &= \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_0 \\ \boldsymbol{u}=\boldsymbol{u}_0}} \Delta \boldsymbol{x} + \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_0 \\ \boldsymbol{u}=\boldsymbol{u}_0}} \Delta \boldsymbol{u} \\ \Delta \boldsymbol{y} &= \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_0 \\ \boldsymbol{u}=\boldsymbol{u}_0}} \Delta \boldsymbol{x} + \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{u}}\right|_{\substack{\boldsymbol{x}=\boldsymbol{x}_0 \\ \boldsymbol{u}=\boldsymbol{u}_0}} \Delta \boldsymbol{u} \\ \boldsymbol{x} &= \boldsymbol{x}_0 + \Delta \boldsymbol{x} \\ \boldsymbol{u} &= \boldsymbol{u}_0 + \Delta \boldsymbol{u} \\ \boldsymbol{y} &= \boldsymbol{y}_0 + \Delta \boldsymbol{y} \end{aligned} \tag{71}$$

Note that both the inputs, the outputs and the states need to be linearized in order to apply the stability criterion. Also note that the control vector $\boldsymbol{\tau}_c$ is omitted in (71) since the control law can be treated as internal system dynamics or a separate distributed subsystem. Corollary 2 summarizes the procedure for applying Theorem 1 for analysing stability in nonlinear distributed systems.

Corollary 2 (Simulation Stability Criterion for Nonlinear Systems)**.** *A nonlinear dynamical system as given in* (70) *can be linearized according to* (71) *and if finite ranges for the states, the inputs and the outputs can be determined and given as* $\boldsymbol{x} \in [\boldsymbol{x}_l, \boldsymbol{x}_u]$, $\boldsymbol{u} \in [\boldsymbol{u}_l, \boldsymbol{u}_u]$ *and* $\boldsymbol{y} \in [\boldsymbol{y}_l, \boldsymbol{y}_u]$*, respectively, Theorem 1 can be applied where* $\boldsymbol{A}_i(\boldsymbol{x}_{i,0})$*,* $\boldsymbol{B}_i(\boldsymbol{u}_{i,0})$ *and* $\boldsymbol{C}_i(\boldsymbol{y}_{i,0})$ *are used to determine conservative values for* Δt_i *and* T_d *that make the total distributed simulation stable.*

Proof. A linearized version of a nonlinear system i, given in (71), can be expressed as

$$\begin{aligned} \dot{\Delta \boldsymbol{x}}_i &= \boldsymbol{A}_i(\boldsymbol{x}_{0i}, \boldsymbol{u}_{0i})\Delta \boldsymbol{x}_i + \boldsymbol{B}_i(\boldsymbol{x}_{0i}, \boldsymbol{u}_{0i})\Delta \boldsymbol{u}_i \\ \Delta \boldsymbol{y}_i &= \boldsymbol{C}_i(\boldsymbol{x}_{0i}, \boldsymbol{u}_{0i})\Delta \boldsymbol{x}_i + \boldsymbol{D}_i(\boldsymbol{x}_{0i}, \boldsymbol{u}_{0i})\Delta \boldsymbol{u}_i \end{aligned} \tag{72}$$

Given $\boldsymbol{x}_{0i} \in [\boldsymbol{x}_{li}, \boldsymbol{x}_{ui}]$, $\boldsymbol{u}_{0i} \in [\boldsymbol{u}_{li}, \boldsymbol{u}_{ui}]$ and $\boldsymbol{y}_{0i} \in [\boldsymbol{y}_{li}, \boldsymbol{y}_{ui}]$ then $\exists$ eigenvalues for system i so that $\boldsymbol{\lambda}_i(\boldsymbol{x}_{0i}, \boldsymbol{u}_{0i}) \in [\boldsymbol{\lambda}_{li}, \boldsymbol{\lambda}_{ui}]$, which means that a finite range of eigenvalues can be determined. Moreover, $\exists$ $\boldsymbol{A}_{n_i}(\boldsymbol{x}_{0i}, \boldsymbol{u}_{0i})$, $\boldsymbol{B}_{n_i}(\boldsymbol{x}_{0i}, \boldsymbol{u}_{0i})$, $\boldsymbol{C}_{n_i}(\boldsymbol{x}_{0i}, \boldsymbol{u}_{0i})$ and $\boldsymbol{D}_{n_i}(\boldsymbol{x}_{0i}, \boldsymbol{u}_{0i})$ such that

$$\mathrm{eig}(\boldsymbol{S}_d(\boldsymbol{A}_{n_1}, \boldsymbol{B}_{n_1}, \cdots, \boldsymbol{D}_{n_N})) \in [\mathrm{eig}(\boldsymbol{S}_d)_l, \mathrm{eig}(\boldsymbol{S}_d)_u] \tag{73}$$

Then, if both $|\mathrm{eig}(S_d)_l| \leq 1$ and $|\mathrm{eig}(S_d)_u| \leq 1$, the simulation results from the nonlinear system are stable. □

Typically the finite ranges for the states, the inputs and the outputs are chosen based on validity regions of the subsystems, initial values, maximal expected values or saturation limits.

To illustrate the use of Corollary 2, Example 5 is revised.

Example 6 (Regulated and distributed mass–damper-spring system with nonlinear spring stiffness)**.** *The mass-damper-spring system given in Figure 8 has now a spring with nonlinear spring stiffness, and the system of differential equations are now given as*

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{1}{m}(F_r - bx_2 - kx_1^3) \end{aligned} \tag{74}$$

The linearized mass-damper-spring system can be written as

$$\begin{aligned} \dot{\Delta x}_1 &= x_2 \\ \dot{x}_2 &= \frac{1}{m}(F_r - bx_2 - 3kx_{01}^2 \Delta x) \end{aligned} \tag{75}$$

which gives

$$\boldsymbol{A}_{n_1}(x_{01}) = \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \Delta t_1 \begin{bmatrix} 0 & 1 \\ -\frac{3kx_{01}^2}{m} & -\frac{b}{m} \end{bmatrix} \right)^{n_1} \tag{76}$$

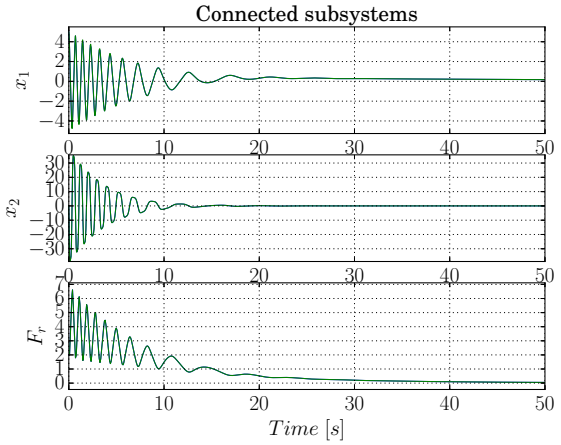


Figure 10: Simulation results from regulated and distributed mass-damper-spring system with nonlinear spring stiffness. Green graph denotes local solution and blue graph denotes global solution

$$\boldsymbol{B}_{n_1}(x_{01}) = \left[\begin{array}{cc} 0 & 1 \\ -\frac{3kx_{01}^2}{m} & -\frac{b}{m} \end{array}\right]^{-1} \left(\boldsymbol{A}_{n_1} - \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array}\right]\right) \left[\begin{array}{c} 0 \\ \frac{1}{m} \end{array}\right] \tag{77}$$

Since the uncontrolled system is strictly passive and since the regulator was able to stabilize the linear mass-damper-spring system in Example 5, it is reasonable to assume that the initial value for x_1 *in* (75) *would be the highest value for* x_1 *such that* $x_1 \in [-x_1(0), x_1(0)]$. *By assuming that all values are as in Example 5, it can be verified that* $|eig(\boldsymbol{S}_d)| = \{1.0000, 1.0000, 0.9922\}$ *when* $\Delta t_1 = 0.0013\,s$, $\Delta t_2 = 0.026\,s$ *and* $T_d = 0.078\,s$, *giving* $n_1 = 60$ *and* $n_2 = 3$. *The simulation results are shown in Figure 10. As can be seen in the figure the total distributed system is stable and converges relatively quickly to zero. This means that the stability criterion for nonlinear systems, as given in Corollary 2 is a bit conservative. However, this is not surprising since maximal values for the states, the inputs and the outputs were used in the stability criterion.*

4. Conclusion

In this work numerical stability of distributed simulations has been studied by exploring outer solver stability limits. Numerical stability of distributed simulations has not been treated in any detail in the literature. In this work stability requirements for explicit solvers for differential equations have been combined with system dynamics and the global communication time-step in distributed system, and a criterion for guaranteeing stable simulations has been derived. This criterion is also extended to include nonlinear system dynamics in the stability analysis. Several examples are given to illustrate the use of the criterion as well as illustrating stable numerical results from distributed simulations.

Acknowledgment

This work was funded by the Research Council of Norway (project no. 225322 MAROFF) and the industrial partners in the ViProMa project, Virtual Prototyping of Marine Systems and Operations (VARD, Rolls-Royce Marine, and DNV GL). We are grateful for their financial support.

References

References

[1] D. C. Karnopp, D. L. Margolis, R. C. Rosenberg, System Dynamics: Modeling and Simulation of Mechatronic Systems, John Wiley & Sons, Inc., New York, NY, USA, 2006.

[2] F. E. Cellier, E. Kofman, Continuous System Simulation, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[3] W. Borutzky (Ed.), Bond Graph Modelling of Engineering Systems – Theory, Applications and Software Support, 1st Edition, Springer-Verlag, NY, NY, U.S.A, NY, NY, U.S.A., 2011.

[4] H. K. Khalil, Nonlinear Systems, Prentice Hall PTR, 2002.

[5] C.-T. Chen, Linear System Theory and Design, 3rd Edition, Oxford University Press, Inc., New York, NY, USA, 1998.

[6] M. Arnold, Numerical methods for simulation in applied dynamics, Vol. 507 of CISM International Centre for Mechanical Sciences, Springer Vienna, 2009, book section 5, pp. 191–246.

[7] G. F. Coulouris, J. Dollimore, T. Kindberg, Distributed systems: concepts and design, pearson education, 2005.

[8] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauss, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold, D. Neumerkel, H. Olsson, J. v. Peetz, S. Wolf, A. S. Gmbh, Q. Berlin, F. Scai, S. Augustin, The Functional Mockup Interface for Tool independent Exchange of Simulation Models, in: In Proceedings of the 8th International Modelica Conference, 2011.

[9] J. S. Dahmann, R. Fujimoto, R. M. Weatherly, The Department of Defense High Level Architecture, in: Proceedings of the 29th conference on Winter simulation, {WSC} 1997, Atlanta, GA, USA, December 7-10, 1997, 1997, pp. 142–149.

[10] R. L. Bucs, L. G. Murillo, E. Korotcenko, G. Dugge, R. Leupers, G. Ascheid, A. Ropers, M. Wedler, A. Hoffmann, Virtual hardware-in-the-loop co-simulation for multi-domain automotive systems via the functional mock-up interface, in: Specification and Design Languages (FDL), 2015 Forum on, 2015, pp. 1–8.

[11] A. Abel, T. Blochwitz, A. Eichberger, P. Hamann, U. Rein, Functional Mock-up Interface in Mechatronic Gearshift Simulation for Commercial Vehicles.

[12] Y. G. Liao, H. I. Du, Cosimulation of multibody-based vehicle dynamics and an electric power steering control system, Proceedings of the Institution of Mechanical Engineers, Part K: Journal of Multi-body Dynamics 215 (3) (2001) 141–151.

[13] Y. Chu, L. I. Hatledal, F. Sanfilippo, H. G. Schaathun, V. Æsøy, H. Zhang, Virtual Prototyping System for Maritime Crane Design and Operation Based on Functional Mock-up Interface, in: Proceeding of the MTS/IEEE Oceans '15 Conference, Genova, Italy, MTS/IEEE, 2015, pp. 1–4.

[14] J.-N. Paquin, W. Li, J. Belanger, L. Schoen, I. Peres, C. Olariu, H. Kohmann, A modern and open real-time digital simulator of All-Electric Ships with a multi-platform co-simulation approach, in: Electric Ship Technologies Symposium, 2009. ESTS 2009. IEEE, 2009, pp. 28–35.

[15] T. Chen, B. A. Francis, Input-Output Stability Of Sampled-Data Systems, IEEE Transactions on Automatic Control 36 (1) (1991) 50–58.

[16] T. Chen, B. A. Francis, H2-Optimal Sampled-Data Control, IEEE Transactions on Automatic Control 36 (4) (1991) 387–397.

[17] S. Gottlieb, C.-W. Shu, E. Tadmor, Strong stability-preserving high-order time discretization methods, SIAM Review 43 (1) (2001) 89–112.

[18] R. Kübler, W. Schiehlen, Two Methods of Simulator Coupling, Mathematical and Computer Modelling of Dynamical Systems 6 (2) (2000) 93–113.

[19] M. Arnold, C. Clauss, T. Schierz, Error Analysis and Error Estimates for Co-Simulation in FMI for Model Exchange and Co-Simulation V2.0, Archive of Mechanical Engineering 60 (1) (2013) 75–94.

[20] M. Arnold, Stability of sequential modular time integration methods for coupled multibody system models, Journal of Computational and Nonlinear Dynamics 5 (3) (2010) 1–9.

[21] B. Owren, Stability of Runge-Kutta methods used in modular integration, Journal of Computational and Applied Mathematics 62 (1) (1995) 89–101.

[22] M. Busch, Zur effizienten Kopplung von Simulationsprogrammen, Thesis (2012).

[23] G. Dahlquist, Convergence and stability in the numerical integration of ordinary differential equations, Mathematica Scandinavica 4 (1956) 33–53.

[24] B. Schweizer, P. Li, D. Lu, Explicit and Implicit Cosimulation Methods: Stability and Convergence Analysis for Different Solver Coupling Approaches, Journal of Computational and Nonlinear Dynamics 10 (5) (2015) 51007.

[25] S. Skjong, E. Pedersen, The Theory of Bond Graphs in Distributed Systems and Simulations, in: 2016 International Conference on Bond Graph Modeling and Simulation (ICBGM 2016), Society for Modeling & Simulation International, Montreal, Canada, 2016.

[26] U. M.Ascher, L. R.Petzold, Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations, SIAM Journal on Scientific Computing (1998) 332.