Half-Explicit Exponential Runge-Kutta Methods for Index-1 DAEs in Helicopter Simulation

Elena Kohlwey and Melven Röhrig-Zöllner

Abstract. In this paper we suggest a combination of exponential integrators and half-explicit Runge-Kutta methods for solving index-1 DAE systems with a stiff linear part in their differential equations. We discuss the behavior of the resulting half-explicit exponential Runge-Kutta (HEERK) methods for a simple numerical example and for a coupled rotor simulation. The coupled rotor simulation is based on a modular software design where all subsystems are modeled by ODEs in state-space form. By connecting the subsystems' inputs and outputs we obtain an index-1 DAE system. Large terms in the system can be expressed as a stiff linear part which includes strong damping or oscillation terms as well as coefficients for the discretization of the rotor blades (3d beam equations). We show that the proposed HEERK methods can solve the resulting system efficiently with a reasonable timestep size.

1. Introduction

We aim to simulate complex oscillatory physical systems (e. g. comprehensive helicopter systems) which are composed of a set of subsystems. The behavior of subsystem i is defined by a model in *state-space form*

$$\begin{cases} \dot{\mathbf{x}}_i(t) & = f_i(t, \mathbf{x}_i(t), \mathbf{u}_i(t)) \\ \mathbf{y}_i(t) & = g_i(t, \mathbf{x}_i(t), \mathbf{u}_i(t)) \end{cases}$$

with a state vector $\mathbf{x}_i \in \mathbb{R}^{n_i}$, output vector $\mathbf{y}_i \in \mathbb{R}^{m_i}$, input vector $\mathbf{u}_i \in \mathbb{R}^{\ell_i}$ and time $t \in \mathbb{R}_+$. The model functions $f_i : \mathbb{R}_+ \times \mathbb{R}^{n_i} \times \mathbb{R}^{\ell_i} \to \mathbb{R}^{n_i}$ and $g_i : \mathbb{R}_+ \times \mathbb{R}^{n_i} \times \mathbb{R}^{\ell_i} \to \mathbb{R}^{m_i}$ define the model i.

Combining k models into one *coupled system* leads to a system

$$\begin{cases} \dot{\mathbf{x}}(t) & = f(t, \mathbf{x}(t), \mathbf{y}(t)) \\ \mathbf{y}(t) & = g(t, \mathbf{x}(t), \mathbf{y}(t)) \end{cases} \tag{1}$$

with the global state vector $\mathbf{x} \in \mathbb{R}^n$, $n = \sum_{i=1}^{k} n_i$ and the global output vector $\mathbf{y} \in \mathbb{R}^m$, $m = \sum_{i=1}^{k} m_i$. Accordingly, $f : \mathbb{R}_+ \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and $g : \mathbb{R}_+ \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m$. Here, for each model the entries of the input vector are defined by entries of output vectors of other models. We will call the first equation system the *dynamic part* and the second equation system will be called the *algebraic part*.
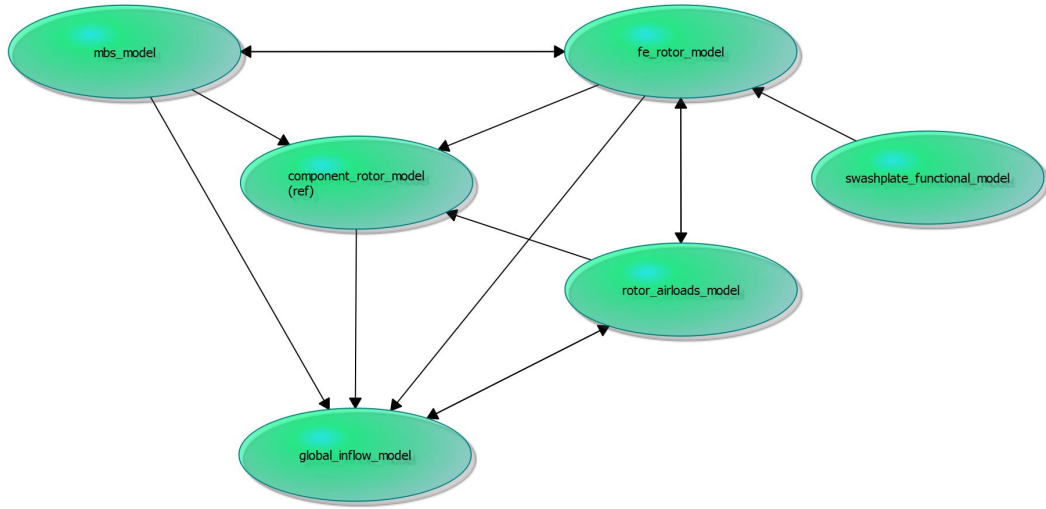
Figure 1. Example for a setup of a rotor simulation with multiple models and their input-output dependencies

The formulation of a complex system as a coupled system has a major computational advantage. Every subsystem can individually be designed, implemented and adapted. Such a modular software approach makes the individual components easily interchangeable, where changes concerning one part of the simulation can be implemented without changing any other part.

The comprehensive simulation of helicopters yields an application of such coupled systems: Helicopters consist of several subsystems that are dominated by different physical phenomena. For example, we have to account for the rotor dynamics and stability, for the arising vortices and for the overall flight dynamics. And as these subsystems are not independent of each other, we have to consider how they interact. One way to implement a holistic approach is to model the helicopter as a coupled system of state-space models as in (1).

Figure 1 illustrates this with an example from our helicopter simulation software for a setup of a simulation with a single rotor in a wind-tunnel. The *mbs_model* (multi-body systems model) allows to model movements and forces of rigid bodies connected through joints. Here it models the (rigid) mounting point of the rotor in the wind-tunnel. The *fe_rotor_model* defines the equations for the flexible rotor blades based on a finite element (FE) discretization of a 3d beam formulation. The *swashplate_functional_model* calculates rotor blade orientations adjusting the angle-of-attack of each blade based on control parameters. These yield boundary conditions for the flexible rotor blades. The aerodynamic part of the simulation is split into two subsystems: a near field approximation and a far field flow (the rotor wake). The *rotor_airloads_model* calculates forces on the rotor blades due to the near field approximation. The *global_inflow_model* is a simplistic model of the far field flow. It remains the *component_rotor_model* which calculates other important derived quantities of the rotor (e.g. its thrust coefficient). An arrow indicates that one model requires outputs from another model as its inputs.

A classical approach to resolve the couplings between those models consists of an iteration through these models in a predefined order, possibly including some weights for damping. This results in a (damped) fixed-point iteration of the algebraic part which might only converge (slowly) under specific assumptions. Nevertheless, this is a suggested approach in the literature (see e.g. section 13.6.2 in [13]). In this paper, we will consider the model couplings as a general system of algebraic equations with some additional assumptions which will be discussed below.

For simulating the rotor dynamics we need to integrate the coupled system over long time periods with large numbers of timesteps: there are relevant effects on the time-scale of a small fraction of a rotor revolution (e. g. 1 degree of rotor rotation), but the flight dynamics happen on a larger scale (between 10 and 100 revolutions). To allow fast calculations we impose some restrictions on the models and the combined coupled systems:

- the functions f and g are sufficiently differentiable,
- $\left(I - \frac{\partial g}{\partial \mathbf{y}}\right)^{-1}$ exists and is bounded in a neighborhood of the solution.

Here, I denotes the identity matrix and $\partial g / \partial \mathbf{y}$ the gradient matrix of the algebraic part. These restrictions make system (1) a differential-algebraic equations (DAE) system of index 1 (cf. [1, 7]). Note that $\mathbf{y}(t)$ is then uniquely determined by g and the choice of $(t, \mathbf{x}(t))$ through the implicit function theorem.

The outline of the paper is as follows: In Section 2, we introduce and motivate half-explicit exponential Runge-Kutta (HEERK) methods. This class of methods is very valuable in our setting, where we deal with a linear stiff part and a nonlinear nonstiff part in our DAE system. To the best of our knowledge, this specific combination of methods has not been used in the literature before. In Section 3, we discuss the advantages of HEERK methods in comparison to classical half-explicit Runge-Kutta (HERK) methods by using a simplified stiff rotor model which exhibits a similar behavior to what we expect from a full helicopter system. Here, we use the mechanical model of [18] as a starting point. In Section 4, we show the results for a small rotor simulation featuring flexible rotor blades coupled to simple aerodynamic models based on the coupled system depicted in Figure 1. Finally, we summarize our findings in Section 5.

2. Numerical solution of index-1 DAE systems

In order to define half-explicit exponential Runge-Kutta methods, let us first start with classical Runge-Kutta methods. Explicit Runge-Kutta (ERK) methods (See [3] and [4]) are popular one-step procedures for solving ordinary differential equations (ODEs) of the kind

$$\begin{cases} \dot{\mathbf{x}}(t) & = f(t, \mathbf{x}(t)), \quad \mathbf{x} \in \mathbb{R}^n \\ \mathbf{x}(t_0) & = \mathbf{x}_0. \end{cases}$$

An explicit s-stage Runge-Kutta method is given by the iteration

$$X_{ni} = x_n + h \sum_{j=1}^{i-1} a_{ij} f(X_{nj}), \quad i = 1, \ldots, s$$

$$x_{n+1} = x_n + h \sum_{i=1}^{s} b_i f(X_{ni})$$

for a step size h and Runge-Kutta coefficients a_{ij} and weights b_i.

As first shown in [7], one can extend the classical Runge-Kutta methods to differential-algebraic equations and define half-explicit Runge-Kutta (HERK) methods by inserting a solution step of the equation

$$0 = \bar{g}(t, \mathbf{x}(t), \mathbf{y}(t)) := g(t, \mathbf{x}(t), \mathbf{y}(t)) - \mathbf{y}(t)$$

in every stage of the ERK method:

$$X_{ni} = x_n + h \sum_{j=1}^{i-1} a_{ij} f(X_{nj}, Y_{nj}), \quad i = 1, \ldots, s \tag{2a}$$

$$0 = \bar{g}(X_{ni}, Y_{ni}), \qquad\qquad i = 1, \ldots, s \tag{2b}$$

$$x_{n+1} = x_n + h \sum_{i=1}^{s} b_i f(X_{ni}, Y_{ni}), \tag{2c}$$

$$0 = \bar{g}(x_{n+1}, y_{n+1}). \tag{2d}$$

In every timestep, we alternately compute X_{ni} as in (2a) and Y_{ni} as in (2b) for all $i = 1, \ldots, s$ and insert the results into (2c) subsequently. Finally, the corresponding output vector y_{n+1} to x_{n+1} is computed in (2d) in order to start the next timestep with coherent state and output vectors. If the function $\bar{g}$ can be solved exactly for Y_{ni} or y_{n+1}, respectively, the convergence results of ERK methods translate to HERK methods. However, an exact solution of (2b) or (2d) is computationally costly in general. Instead, in real-life applications, we approximate the solution of this equation using e. g. the Newton method. Then, convergence results do not exclusively translate from ERK methods but also depend on the accuracy of the Newton solution (see [1]).

Let us come back to our coupled state-space system (1). For its solution, we could use half-explicit RK methods. However, these methods are not suitable if the dynamic part of (1) includes stiff terms. Then, only very small timesteps lead to convergence, similar to the behavior of explicit methods for ODEs. Fully implicit methods are often more suitable for stiff systems. However, they usually require solving large nonlinear systems of equations in every timestep. Both of these variants, classical half-explicit methods and fully-implicit, thus incur high computational costs.

In the following we present a possible way out of this dilemma by using exponential integrators. The theoretical background of exponential integrators was studied by several authors, e.g. by [2, 16, 9, 10] and [11]. They study the application of exponential integrators (also called exponential time differencing (ETD) methods) on discretized partial differential equations (PDEs). The resulting ODEs also display stiff behavior in the linear part which suggests the utility of exponential integrators in this setting; cf. [6]. A detailed overview on exponential integrators can be found in [11].
Here, we assume that we can split the state function $f_i(t, \mathbf{x}_i(t), \mathbf{u}_i(t))$ of every subsystem into a linear stiff part with the matrix $S_i \in \mathbb{R}^{n_i \times n_i}$ and a nonlinear less stiff part

$$\tilde{f}_i(t, \mathbf{x}_i(t), \mathbf{u}_i(t)) := f_i(t, \mathbf{x}_i(t), \mathbf{u}_i(t)) - S_i \mathbf{x}_i(t).$$

Then a subsystem i has the form

$$\begin{cases} \dot{\mathbf{x}}_i(t) &= S_i \mathbf{x}_i(t) + \tilde{f}_i(t, \mathbf{x}_i(t), \mathbf{u}_i(t)), \\ \mathbf{y}_i(t) &= g_i(t, \mathbf{x}_i(t), \mathbf{u}_i(t)). \end{cases}$$

Combining all subsystems we obtain

$$\begin{cases} \dot{\mathbf{x}}(t) &= S\mathbf{x}(t) + \tilde{f}(t, \mathbf{x}(t), \mathbf{y}(t)), \quad \mathbf{x} \in \mathbb{R}^n, \ S \in \mathbb{R}^{n \times n} \\ \mathbf{y}(t) &= g(t, \mathbf{x}(t), \mathbf{y}(t)) \end{cases}, \tag{3}$$

where

$$S := \begin{pmatrix} S_1 & 0 & \ldots & 0 \\ 0 & S_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & S_k \end{pmatrix}$$

denotes a blockdiagonal matrix containing the stiff parts S_i of the k models and

$$\tilde{f}(t, \mathbf{x}(t), \mathbf{y}(t)) = \begin{pmatrix} \tilde{f}_1(t, \mathbf{x}_1(t), \mathbf{u}_1(t)) \\ \vdots \\ \tilde{f}_k(t, \mathbf{x}_k(t), \mathbf{u}_k(t)) \end{pmatrix}.$$

With this formulation of the problem we are able to handle the stiff and the nonstiff parts of the dynamic equations individually. In particular, the matrices S_i can be obtained through rough linearizations. Such a splitting is possible for a wide range of application. What the literature does not emphasize enough, is the following: using some kind of linearization, one can often split a problem into a linear stiff part and a nonlinear nonstiff part. So this approach does not depend on very rare properties of the problem. It becomes only problematic when strong nonlinear terms occur.

For a derivation of suitable methods for (3), we first consider the related ODE problem

$$\begin{cases} \dot{\mathbf{x}}(t) &= S\mathbf{x}(t) + \tilde{f}(t, \mathbf{x}(t)), \quad \mathbf{x} \in \mathbb{R}^n, S \in \mathbb{R}^{n \times n} \\ \mathbf{x}(t_0) &= \mathbf{x}_0 \end{cases}. \tag{4}$$

For (4), we can use explicit exponential Runge-Kutta (EERK) methods, where the linear stiff part $S\mathbf{x}(t)$ is integrated exactly and the nonlinear less stiff part $\tilde{f}(t, \mathbf{x}(t))$ is integrated by an explicit Runge-Kutta method. By handling these two parts individually, we circumvent tiny timesteps in explicit methods and the solution of nonlinear systems of equations in implicit methods.

Exponential Runge-Kutta methods are derived from the variation of parameters formula and can be displayed in a Butcher tableau; cf. [9, 10, 11]. For explicit s-stage methods, we have

$$\begin{array}{c|ccccc} c_1 & 0 \\ c_2 & \mathbf{a}_{21}(hS) \\ \vdots & \vdots & \ddots & & \ddots \\ c_s & \mathbf{a}_{s,1}(hS) & \dots & \mathbf{a}_{s,s-1}(hS) \\ \hline & \mathbf{b}_1(hS) & \dots & \mathbf{b}_{s-1}(hS) & \mathbf{b}_s(hS) \end{array},$$

where the coefficients $\mathbf{a}_{ij}(hS)$ and $\mathbf{b}_i(hS)$, $i, j = 1, \dots, s$ are $n \times n$-matrices that are integrals of $\exp(hS)$ and $\exp(c_i hS)$ with $c_1, \dots, c_s \in \mathbb{R}$. An iteration of the EERK method then takes the form

$$X_{ni} = \mathbf{x}_n + h \sum_{j=1}^{i-1} \mathbf{a}_{ij}(hS)\Big(\tilde{f}(X_{nj}) + S\mathbf{x}_n\Big), \quad i = 1, \dots, s \tag{5a}$$

$$x_{n+1} = \mathbf{x}_n + h \sum_{i=1}^{s} \mathbf{b}_i(hS)\Big(\tilde{f}(X_{ni}) + S\mathbf{x}_n\Big). \tag{5b}$$

Since we are interested in the solution of index-1 DAES where the dynamic part has the form (4), we construct and analyze HEERK methods in analogy to HERK methods. Hence similarly, we insert a Newton process in every internal stage of the exponential Runge-Kutta method (5) and obtain

$$X_{ni} = \mathbf{x}_n + h \sum_{j=1}^{i-1} \mathbf{a}_{ij}(hS)\Big(\tilde{f}(X_{nj}, Y_{nj}) + S\mathbf{x}_n\Big), \quad i = 1, \dots, s \tag{6a}$$

$$0 = \bar{g}(X_{ni}, Y_{ni}), \qquad\qquad\qquad i = 1, \dots, s \tag{6b}$$

$$x_{n+1} = \mathbf{x}_n + h \sum_{i=1}^{s} \mathbf{b}_i(hS)\Big(\tilde{f}(X_{ni}, Y_{nj}) + S\mathbf{x}_n\Big), \tag{6c}$$

$$0 = \bar{g}(\mathbf{x}_{n+1}, \mathbf{y}_{n+1}). \tag{6d}$$

The convergence results from EERK methods translate to HEERK methods provided that (6b) and (6d) are solved to a certain accuracy, see [14] for a short discussion on this.

In contrast to a fully-implicit method we obtain smaller nonlinear systems only for the algebraic part and from our observations a few (simplified) Newton steps are required in every internal RK stage as we already have a good start vector assuming that the timesteps are small enough. For half-explicit RK methods, one can prescribe a specific number of simplified Newton steps in every RK stage for different method orders as discussed in [1].

Another interesting class of methods for highly-oscillatory semi-explicit DAEs is presented in [5] with a detailed theoretical analysis. However, in contrast to our approach it is based on splitting the problem into a small nonlinear part and a truncated Fourier series. So the highly-oscillatory part of the problem is formulated in the frequency domain, which might not be easily possible for the models arising in helicopter simulation.

Furthermore, the authors remark that there are interesting numerical questions that are not addressed in this paper: first, we do not analyze suitable conditions for obtaining stability and convergence for the problem at hand. We want to consider this in future work.

Second, parts of the original problem may have a specific structure (e.g. energy-preservation in mechanical systems) which is not exploited here. There are numerical methods that preserve that structure (see e.g. [8] for an overview on geometric integrators) but we cannot use them directly for the general formulation of the coupled system (1) without further assumptions.

The authors point out that the general formulation of the problem also allows to include a model that uses an existing external software package for their calculations. The only requirement is that it defines a set of states and outputs and evaluates f_i and g_i at given points (and that f_i and g_i are somewhat smooth). In fact, we have simulations that use an external software for an aerodynamic model that manages even its states by itself (using a time-integration scheme with interpolated output).

3. Numerical Experiments

In this section we present a simple mechanical problem to illustrate the usefulness of the suggested numerical method. In particular, we want to evaluate the behavior for phenomena with oscillations on different time scales.

3.1. Mechanical Model

For our test problem, we use the work in [18] that models a simplified rotor fixed to a fuselage (see Figure 2). The fuselage is considered to be a rigid body connected to a rotor hub with 4 blades which are represented by a concentrated mass located at a distance b from the hinge offset (point B).

We now present the model of [18] and show our approach for transforming their model into a stiff coupled DAE model. Appendix A contains a more detailed derivation of our model and an explanation of all variables.

The governing equation of the system reads

$$M\ddot{\mathbf{u}}(t) + G\dot{\mathbf{u}}(t) + K\mathbf{u}(t) = F \tag{7}$$

with the unknowns

$$\mathbf{u}(t) = (x_{\text{Fus}}(t), y_{\text{Fus}}(t), \varphi_1(t), \varphi_2(t), \varphi_3(t), \varphi_4(t))^T \in \mathbb{R}^6,$$

and with matrices $M, G, K \in \mathbb{R}^{6\times6}$ and vector $F \in \mathbb{R}^6$. Here, $x_{\text{Fus}}(t)$ and $y_{\text{Fus}}(t)$ denote the longitudinal and transversal displacement of the fuselage and $\varphi_i(t), i = 1, \ldots 4$ denotes the lead-lag angle of the ith blade; cf. Appendix A.1 and [18] equations (1) – (8) and Table 1.

With $\mathbf{x}_1(t) := \big(\mathbf{u}(t)\ \dot{\mathbf{u}}(t)\big)^T$ we transform the system (7) into an explicit first order ODE

$$\dot{\mathbf{x}}_1(t) = \begin{pmatrix} & I \\ -M^{-1}K & -M^{-1}G \end{pmatrix} \mathbf{x}_1(t) + \begin{pmatrix} \\ M^{-1}F \end{pmatrix}, \tag{8}$$
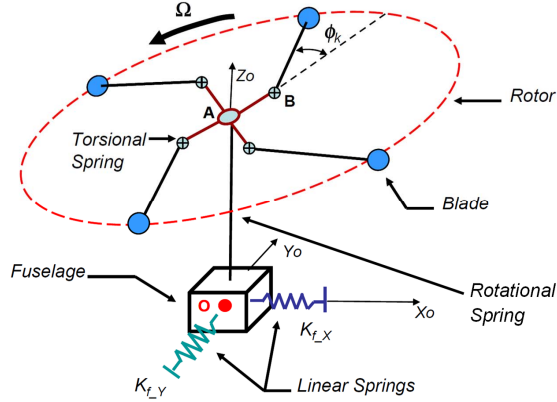
Figure 2. Simplified rotor fixed to a fuselage (adapted sketch from [18] (Figure 1))

where I denotes the identity matrix. The matrices M, K, F and G all depend on t. To make this problem more interesting we allow the mast to have a small play by introducing a linear torsion spring. By adjusting the spring constant we can render the problem more or less stiff. From the modeling point of view the linear torsion spring reflects in a very simplistic way the complex behavior of the engine with the gearbox, the drivetrain and the actual mast. We use two subsystems in state-space form: one subsystem for the mast and one for everything else. As discussed before, we obtain a stiff index-1 DAE system by combining the state-space representations of these two subsystems.

To model the occurring behavior of the linear torsion spring we introduce additional constants and variables. Let K_s denote the stiffness coefficient of the mast, and let $\varphi_{RH}(t)$ denote the rotational angle of the rotor head. Accordingly, $\omega_{RH}(t) := \dot{\varphi}_{RH}(t)$ is its angular velocity. Using Hooke's law we obtain

$$\dot{\omega}_{RH}(t) = \omega_s^2(\Omega t - \varphi_{RH}(t)) - r_{sb}\dot{\omega}_1(t) - r_{sb}\dot{\omega}_2(t) - r_{sb}\dot{\omega}_3(t) - r_{sb}\dot{\omega}_4(t)$$

with the angular frequency

$$\omega_s := \sqrt{\frac{K_s}{4(a+b)^2 m_b + 4I_{zb}}},$$

where a, b, m_b and I_{zb} are constants of the model; see Table 6.

We can eliminate the source term Ωt in the equation above using $\bar{\varphi}_{RH}(t) := \varphi_{RH}(t) - \Omega t$ and $\bar{\omega}_{RH}(t) := \omega_{RH}(t) - \Omega$.
With $\mathbf{x}_2(t) := \begin{pmatrix} \bar{\varphi}_{RH} & \bar{\omega}_{RH} \end{pmatrix}^T$ we obtain the first order ODE

$$\dot{\mathbf{x}}_2(t) = \begin{pmatrix} 0 & 1 \\ -\omega_s^2 & 0 \end{pmatrix} \mathbf{x}_2(t) + \begin{pmatrix} 0 \\ -r_{sb}\dot{\omega}_1(t) - r_{sb}\dot{\omega}_2(t) - r_{sb}\dot{\omega}_3(t) - r_{sb}\dot{\omega}_4(t) \end{pmatrix}. \tag{9}$$

We also need to adapt the first ODE model (8) to accommodate for the variable rotational speed:

$$\dot{\mathbf{x}}_1(t) = \begin{pmatrix} I \\ -M_{DAE}^{-1}K_{DAE} & -M_{DAE}^{-1}G_{DAE} \end{pmatrix} \mathbf{x}_1(t) + \begin{pmatrix} 0 \\ M_{DAE}^{-1}F_{DAE} \end{pmatrix}. \tag{10}$$

See Section A.3 for the adjusted definitions of the matrices M_{DAE}, K_{DAE}, G_{DAE} and F_{DAE} that depend on t and $\mathbf{y}_2(t) := (\varphi_{RH}(t), \omega_{RH}(t), \dot{\omega}_{RH}(t))^T$

For the complete system we obtain the following index-1 DAE formulation of two coupled models in state-space form:

$$\dot{\mathbf{x}}_1(t) = f_1(t, \mathbf{x}_1(t), \mathbf{y}_2(t)),$$
$$\dot{\mathbf{x}}_2(t) = f_2(t, \mathbf{x}_2(t), \mathbf{y}_1(t)),$$
$$\mathbf{y}_1(t) = g_1(t, \mathbf{x}_1(t), \mathbf{y}_2(t)),$$
$$\mathbf{y}_2(t) = g_2(t, \mathbf{x}_2(t), \mathbf{y}_1(t)),$$

with $\mathbf{y}_1 := \begin{pmatrix} \dot{\omega}_1(t) & \dot{\omega}_2(t) & \dot{\omega}_3(t) & \dot{\omega}_4(t) \end{pmatrix}^T$, f_1 defined in (10), f_2 defined in (9) and for g_1 and g_2 consult (16) and (17). To exploit exponential integrators we need to split the problem into a linear and a nonlinear part to obtain the form (3). For the DAE formulation we choose the splitting

$$f_1(t, \mathbf{x}_1(t), \mathbf{y}_2(t)) := S_1 \mathbf{x}_1 + \tilde{f}_1(t, \mathbf{x}_1(t), \mathbf{y}_2(t)),$$
$$f_2(t, \mathbf{x}_2(t), \mathbf{y}_1(t)) := S_2 \mathbf{x}_2 + \tilde{f}_2(t, \mathbf{x}_2(t), \mathbf{y}_1(t))$$

with $S_1 = \boldsymbol{O}$ and

$$S_2 := \begin{pmatrix} 0 & 1 \\ -\omega_s^2 & 0 \end{pmatrix}.$$

The extraction of S_2 is very straight-forward in this case. In Section 3.3, we also discuss a different choice of S.

Please note that all the matrices in the definitions above are time-varying.

3.2. Reference solution

In order to compare the performances of HERK and HEERK methods for our stiff DAE model we need a reference solution to compare the numerical solutions to. Since it is too complex or even impossible to solve our DAE system analytically, we use a numerical reference solution. We transform our DAE system into a *monolithic* ODE formulation:

$$\dot{\mathbf{x}}_{mon} = f_{mon}(t, \mathbf{x}_{mon}(t)) = \begin{pmatrix} I \\ -M_{mon}^{-1}K_{mon} & -M_{mon}^{-1}G_{mon} \end{pmatrix} \cdot \mathbf{x}_{mon} + \begin{pmatrix} \\ M_{mon}^{-1}F_{mon} \end{pmatrix}. \quad (11)$$

Here the state vector $\mathbf{x}_{mon}$ is a combination of $\mathbf{x}_1$ and $\mathbf{x}_2$ and the matrices M_{mon}, G_{mon} and K_{mon} depend on $(t, \mathbf{x}_{mon})$; see Section A.2 for more details. We now obtain a reference solution by solving the monolithic system with an explicit 4th order Runge-Kutta method with small timesteps ($h = 10^{-5}$). For our analysis of the convergence error, we focus on the variables $\varphi_i(t), i = 1, \ldots, 4$ as they exhibit the most interesting behavior.

The function $\varphi_1(t)$ indicates the local angle of the first blade. If we start with a small displacement (0.01 rad) for one blade – say the first – leaving the other blades at rest, then the first blade will vibrate when rotating around the mast. Since the mast has also some play, this vibration will set the other blades into a vibrating motion as well. The frequency of the vibrations and the transmission rate depend on the particular degree of stiffness of the mast. Figure 3 shows the motion of the four blades for $t \in [0, 10]$ and $\omega_s \in \{3.91; 12.35; 39.06\}$ ($K_s \in \{30\,000; 300\,000; 3\,000\,000\}$). We see that the initial displacement of the first blade is 0.01 rad, whereas the other three start with no lag angle (0 rad). The blades then vibrate back and forth, while the overall motion of the first blade decreases and the motion of the other blades builds up due to the play of the mast. Additionally to the low-frequency motions, there is also a high-frequency motion due to the overall vibration of the system. For a less stiff mast ($\omega_s \approx 4$, Figure 3(a)), the vibrations have low frequencies and the movement of the blades due to the initial displacement of the first blade is quickly transmitted to the other blades. The stiffer the mast gets the higher the frequencies of the vibration are and the slower the transmission rate becomes (Figures 3(d) and 3(f)).
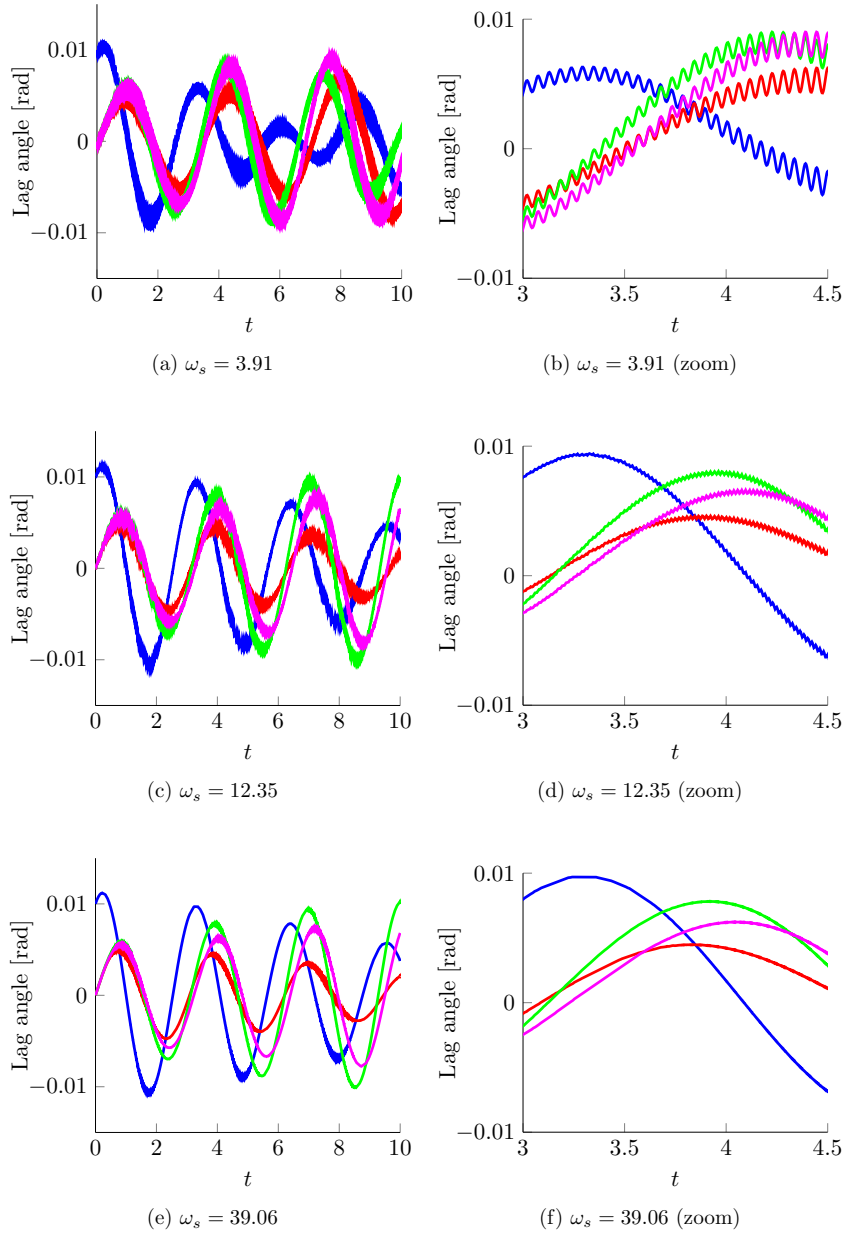
Figure 3. Motion of the four blades (blade 1: blue, blade 2: red, blade 3: green, blade 4: pink) of the model for different values of ω_s

3.3. Comparison of HERK4 and HEERK4

From our testing we expect the HEERK4 method to yield substantially better results for very stiff systems. Particularly, we expect to need fewer timesteps for the HEERK4 methods than for the HERK4 method in order to obtain the same accuracy.
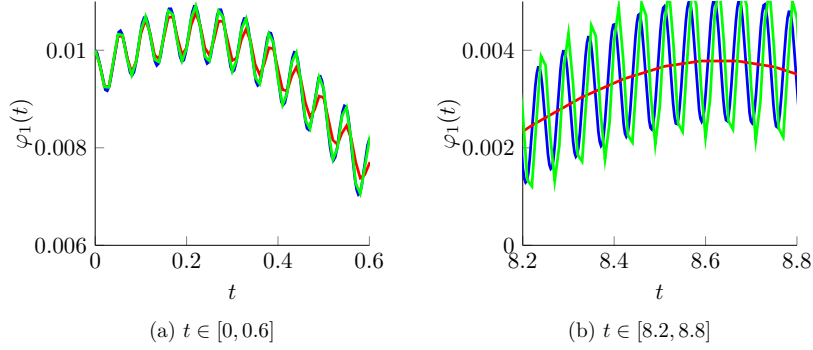
(a) $t \in [0, 0.6]$ (b) $t \in [8.2, 8.8]$

Figure 4. Plot of reference solution (blue), approximation HERK4 (red) and approximation HEERK4 (green) for $\varphi_1(t)$

We start with a stiffness value of $\omega_s = 3.91$ ($K_s = 30\,000$). For $h \in \{10^{-2}; 10^{-3}; 10^{-4}\}$, both methods converge. Table 1 displays the root mean square (RMS) error

$$\varepsilon(\boldsymbol{\varphi}_1) := \sqrt{\frac{\sum\limits_{j=0}^{N+1} [\boldsymbol{\varphi}_1^j - \varphi_1(t_j)]^2}{\sum\limits_{k=0}^{N+1} [\varphi_1(t_j)]^2}},$$

where $\boldsymbol{\varphi}_1 := (\varphi_1^1, \ldots, \varphi_1^N)^T$ denotes the numerical approximation of φ_1, $t_j = t_0 + jh$, $j = 1, \ldots, N$ denote the grid points and φ_1^j denotes the calculated approximation at t_j.

For $h \in \{10^{-3}; 10^{-4}\}$, Table 1 shows that the approximation of HEERK4 is more accurate than the approximation of HERK4. However, for $h = 10^{-2}$, the approximation error is less significant since the approximation of HERK4 starts off with accounting for the high frequency vibrations in the beginning and finally ends up smoothing them out, whereas the approximation of HEERK4 accounts for the vibrations but obtains an offset with advancing time (see Figure 4). Both approximations yield a similar RMS error while approximating the solution in very different ways.

h	HERK4	HEERK4
10^{-2}	$1.5 \cdot 10^{-1}$	$1.2 \cdot 10^{-1}$
10^{-3}	$1.4 \cdot 10^{-4}$	$7.3 \cdot 10^{-6}$
10^{-4}	$1.4 \cdot 10^{-8}$	$7.2 \cdot 10^{-10}$

Table 1. RMS error for $\omega_s = 3.91$

Let us now consider $\omega_s = 12.35$ ($K_s = 300\,000$). For this stiffer system, Figure 3(d) shows that the frequency of the fast mode increases. For a step size of $h = 0.01$, the HERK4 method does not converge anymore, whereas the HEERK4 method yields an approximation in under one second (see Table 2).

Table 2 contains the run times of HERK4 and HEERK4 for the different step widths and values for ω_s. The run times include the calculation time needed for the main loop of the algorithms. The calculation of the coefficients for HEERK4 is constant for any stiffness and any step width and needs around 0.3 seconds. We observe that HEERK4 needs about 30% more time than the respective HERK4 method with the same step width. This overhead is due to the matrix-valued coefficients in the exponential Runge-Kutta methods that cause a higher run time in every iteration than the scalar

coefficients of ordinary Runge-Kutta methods. For different values of ω_s, the run times can differ because the inner Newton iteration may need different numbers of steps in order to reach the desired accuracy.

We observe that the lower limit of step widths for which the methods deliver approximations, is about 10 times higher for the HEERK4 method (i. e. for $\omega_s = 12.35$, HERK4 does not converge anymore for $h = 10^{-2}$ while HEERK4 still does). Furthermore, it is interesting to observe how both methods deliver smoothed approximations (as shown in Figure 4) for the smallest amount of timesteps, for which they deliver an approximation at all. This applies to the HERK4 with a step width of $h = 10^{-2}$ and $\omega_s = 3.91$, HEERK4 with a step width of $h = 10^{-2}$ and $\omega_s = 12.35$ and HERK4 with a step width of $h = 10^{-3}$ and $\omega_s = 39.06$.

h	$\omega_s = 3.91$		$\omega_s = 12.35$		$\omega_s = 39.06$	
	HERK4	HEERK4	HERK4	HEERK4	HERK4	HEERK4
10^{-1}	n.c.	n.c.	n.c.	n.c.	n.c.	n.c.
10^{-2}	0.7	1.0	n.c.	0.9	n.c.	n.c.
10^{-3}	7.4	9.8	7.5	9.7	7.4	10.8
10^{-4}	70.0	93.7	67.1	93.0	74.0	104.0

Table 2. Run time comparison in seconds

The observed advantage of HEERK4 over HERK4 is not as clear as we expected. If we consider the eigenvalues of the linear part of the monolithic system for $t = 0$ and respective initial conditions, we find that the eigenvalues with the highest modulus are much higher than the eigenvalues of S (see Table 3).

max. eigenvalues of S	max. eigenvalues of mon. system
± 3.91i	± 114i
± 12.35i	± 326i
± 39.06i	$\pm 1\,021$i

Table 3. Highest modulus eigenvalues of S vs. highest modulus eigenvalues of the monolithic system for $t = 0$

Apparently, this means that the chosen linear part does not capture the system's stiffness well enough. Most of the stiffness stays in the nonlinear part $\tilde{f}$.

Using the monolithic case and a fourth-order explicit exponential Runge-Kutta (EERK4) method, we want to find out if the exponential methods could perform even better on stiff systems and hence achieve an even more significant predominance over ordinary methods. If we consider the monolithic model and its governing matrix

$$\begin{pmatrix} 0_{7\times 7} & I_{7\times 7} \\ -M_{mon}^{-1}K_{mon} & -M_{mon}^{-1}G_{mon} \end{pmatrix}$$

from equation (11) at $t = t_0, \mathbf{x} = \mathbf{x}_0$, we observe that $-M_{mon}^{-1}K_{mon}$ is a 7×7-matrix, where all entries of the seventh column linearly depend on ω_s^2. These are also the highest entries in modulus of this matrix. If we divide the whole column by ω_s^2, we roughly obtain the vector $(0, 0, 705, 705, 705, 705, -680)^T$.

If we now set

$$S_{mon} := \begin{pmatrix} & 0_{6\times 7} & \\ 0_{7\times 7} & 0 \quad 0 \quad 705\omega_s^2 \quad 705\omega_s^2 \quad 705\omega_s^2 \quad 705\omega_s^2 \quad -680\omega_s^2 \\ I_{7\times 7} & 0_{7\times 7} \end{pmatrix}^T \in \mathbb{R}^{14\times 14},$$

$$\tilde{f}_{mon}(t, \mathbf{x}(t)) := f_{mon}(t, \mathbf{x}(t)) - S_{mon} \cdot \mathbf{x}(t)$$

and apply the EERK4 to $\dot{x}(t) = S_{mon}x(t) + \tilde{f}_{mon}(t, x(t))$, then $\tilde{f}_{mon}$ is less stiff than f_{mon} and we can increase ω_s further without loosing the stability of the exponential Runge-Kutta method (see Table 4). Only for a value of 390.61 for ω_s ($K_s = 300\,000\,000$), the EERK4 method becomes unstable for a step width of $h = 10^{-1}$. This is quite impressive considering that the system is very stiff since the highest modulus eigenvalues of the monolithic system at time $t = 0$ are $\pm 10\,199\mathrm{i}$ for $K_s = 300\,000\,000$.

h	$\omega_s = 3.91$	$\omega_s = 12.35$	$\omega_s = 39.06$	$\omega_s = 123.52$	$\omega_s = 390.61$
10^{-1}	✓	✓	✓	✓	✗
10^{-2}	✓	✓	✓	✓	✓
10^{-3}	✓	✓	✓	✓	✓
10^{-4}	✓	✓	✓	✓	✓

Table 4. Converging step widths for EERK4

The seventh column of the system matrix refers to the seventh entry of x which is $\bar{\varphi}_{RH}(t)$. This explains the above phenomenon. We chose $\bar{\varphi}_{RH}(t)$ – a variable that itself exhibits stiff behavior – as an in-/output variable. Hence, the respective parts in the DAE system are stored in the function g_{DAE} and not in f_{DAE} and thus cannot be captured in S_{DAE}. So, when defining the submodels of a model, we probably should avoid using oscillatory variables as in-/output variables in order to be able to put most of the stiffness of the system into the constant linear part S.

From Table 4, we also observe that a step width of $h = 10^{-1}$ is still enough for $\omega_s = 123.52$ ($K_s = 30\,000\,000$). Using the HERK4 method for such a setting, we require a step width of $h = 10^{-4}$ in order to obtain a converging approximation. That means the required timesteps are a thousand times smaller for a converging solution. This is a significant factor.

Another possibility of obtaining a better S in our DAE model would be a linearization. However that way, we loose the block diagonal structure of S which makes the computation of $\exp(hS)$ in the calculation of the coefficients of exponential integrators more costly.

We conclude that the HEERK4 method is better suited for stiff problems as soon as we can put a significant share of the system's stiffness in the linear constant part. Then, HEERK4 delivers better results than HERK4 for equal step widths and HEERK4 even yields converging results where HERK4 does not anymore. However, we should not choose variables that exhibit stiff behavior as in-/output variables when defining the submodels of the system.

4. Coupled simulation with a finite element rotor blade model and simple nonlinear aerodynamics

In this section we show the results for a complete rotor simulation with multiple models as depicted in Figure 1. Only the FE rotor blade model uses the exponential integrator. All other models define a zero linear part.

The rotor blades are represented by beam equations that feature torsion and bending in flap and lag direction (based on [12]). These equations are discretized using an FE approach with mixed linear and cubic Hermite polynomials. The discrete system has the form:

$$\begin{pmatrix} I & 0 \\ 0 & M(\mathbf{u_i}(t)) \end{pmatrix} \dot{\mathbf{x}_i}(t) = \begin{pmatrix} 0 & I \\ -K(\mathbf{u_i}(t)) & -C(\mathbf{u_i}(t)) \end{pmatrix} \mathbf{x_i}(t) + \bar{f}_i(t, \mathbf{x_i}(t), \mathbf{u_i}(t)).$$

The upper part of the equation results from the reduction of a second order ODE in time to a first order ODE. The mass matrix M and the stiffness matrix K are symmetric positive definite and

(a) linear aerodynamics
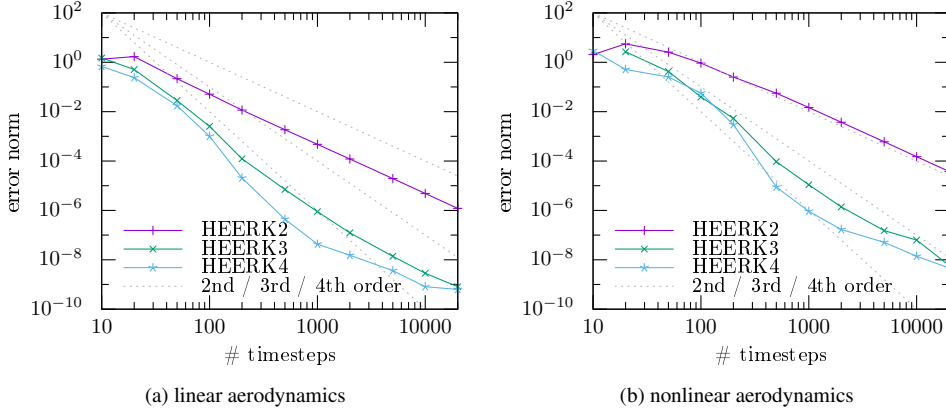
(b) nonlinear aerodynamics

Figure 5. Weighted error norm obtained with different HEERK schemes and timestep sizes for a simulation of one rotor revolution.
The error includes the norm $\| \cdot \|_D$ of the state and output vectors with a diagonal weighting matrix D to accommodate for different expected orders of magnitude of the vector entries. The reference solution is calculated using 10^5 timesteps.

depend on the rotor turn rate (which is part of the input vector $\mathbf{u_i}$). The damping matrix is defined as $C := \alpha M + \beta K$ with some small constants α and β. We have also tested our approach without any damping but usually we are interested in a solution of a slightly damped system. The function $\bar{f}_i$ contains further terms such as the boundary conditions and the aerodynamic forces.
To obtain a linear stiff part we rewrite the equation as

$$\dot{\mathbf{x}}_i(t) = S_i\mathbf{x}_i(t) + \tilde{f}_i(t, \mathbf{x}_i(t), \mathbf{u}_i(t)), \quad \text{with}$$

$$S_i := \begin{pmatrix} I & 0 \\ 0 & M(\mathbf{u}_i^\star) \end{pmatrix}^{-1} \begin{pmatrix} 0 & I \\ -K(\mathbf{u}_i^\star) & -C(\mathbf{u}_i^\star) \end{pmatrix} \quad \text{and}$$

$$\tilde{f}_i := \begin{pmatrix} 0 & I \\ -M^{-1}K & -M^{-1}C \end{pmatrix} \mathbf{x}_i + \bar{f}_i - S\mathbf{x}_i.$$

Here, $\mathbf{u}_i^\star$ contains some fixed nominal turn rate. Please note that the numerical experiments shown in this paper use a constant turn rate. Then S_i represents the original linear part of the system almost exactly (except for round-off errors due to explicitly calculating the operator $M^{-1}K$). However, the approach still works fine for variable turn rates from our experience.
The presented results are calculated with 10 structural blade elements (and $n_{DOF} = n_{Elem} \cdot n_{Var} \cdot n_{Blades} = 400$ degrees of freedom) and feature strongly varying coefficients along the span as the root section of the blade has significantly higher mass and stiffness coefficients (about a factor of 3 to 10).

The nonlinear aerodynamic model calculates forces and moments on a blade segment using a parameterized approximation (a semi-empirical model based on [15]). The parameters depend on the incoming airflow direction and speed (up to Mach 1) and can be identified from wind-tunnel measurements with rotor blade profiles. Here we use a piecewise linear interpolation between different *measured* points in a multidimensional lookup-table. This means that the output function g_i of the aerodynamic model might not fullfil the smoothness conditions required for higher order convergence. Therefore we also show results with a linear model for the blade aerodynamics.

Scheme	# steps / revolution	# g_i-evaluations	average g_i evaluations per stage
HEERK2	10^1	71	3.55
	10^3	7001	~ 3.5
	10^5	700027	~ 3.5
HEERK3	10^1	121	~ 4.0
	10^3	12001	~ 4.0
	10^5	1200029	~ 4.0
HEERK4	10^1	191	~ 3.8
	10^3	19056	~ 3.8
	10^5	1900075	~ 3.8

Table 5. Number of evaluations of the algebraic part of the FE rotor blades model for different schemes and timestep sizes. Interestingly, the average numbers are (almost) independent of the timestep size even though we only prescribe the required residual tolerance of the inner Newton iteration.

4.1. Asymptotic convergence

Figure 5 illustrates the convergence rate for the schemes of 2nd, 3rd and 4th order. With the smooth, linear aerodynamic model (Figure 5a), the error is significantly smaller for the same timestep size than with nonlinear aerodynamics (Figure 5b). The linear aerodynamics neglect important physical effects though, which probably results in an overall smoother solution.

The HEERK2 scheme achieves the expected convergence rate of order 2 quite accurately for both variants. The convergence rate of the HEERK3 and the HEERK4 scheme is less regular. For up to 1000 timesteps per revolution they converge asymptotically with the expected rate of order 3 respectively order 4. For smaller timesteps the convergence rate seems to flatten out.

The authors assume that this effect is caused by inaccuracies in the evaluation of the required matrix exponentials in the exponential Runge-Kutta scheme. For a simpler test case we obtain the expected asymptotic convergence rate even for smaller timesteps when switching to quadruple precision in the algorithm for calculating matrix exponentials. The current implementation is based on a simple scaling and squaring approach (see [17]) and does not exploit the structure of the matrix S_i of the FE blade model. However, this only affects timestep sizes that are not interesting for practical simulations in our case.

In addition, there seems to be a bump in the convergence of the HEERK4 method in the range of about 100 timesteps per revolution before it becomes more accurate then the HEERK3 scheme.

The linear part of the system has eigenvalues of the order $\lambda_{max} \approx \pm 6 \cdot 10^5 \mathrm{i}$. And as expected a *normal* half-explicit Runge-Kutta method (HERK4) is unstable for less than $\sim 10^5$ timesteps per revolution ($T_{revolution} \approx 0.14$). This underlines the well-known fact that (non-exponential) explicit methods are not suited for systems involving parabolic PDEs.

All in all, the HEERK3 and HEERK4 schemes work well in our experiments for the interesting range of timestep sizes of about 200 to 1000 timesteps per revolution. Usually, the engineers in our group are interested in output with a resolution of ~ 360 steps per revolution which resolves the important phyiscal effects in the system.

For this range of timestep sizes the HEERK2 scheme still has a large error (bigger than 10^{-2}), so it might not be very useful in practice.

4.2. Approximate solution of the algebraic part

We will further analyze the computational effort required for solving the algebraic part of the system. Table 5 shows the number of calls to the output equation of the FE rotor model. Please note that the number of calls depends on the model as we use some heuristic to avoid recalculating g_i for a model

(a) original gradient with $1.8 \cdot 10^4$ non-zero entries

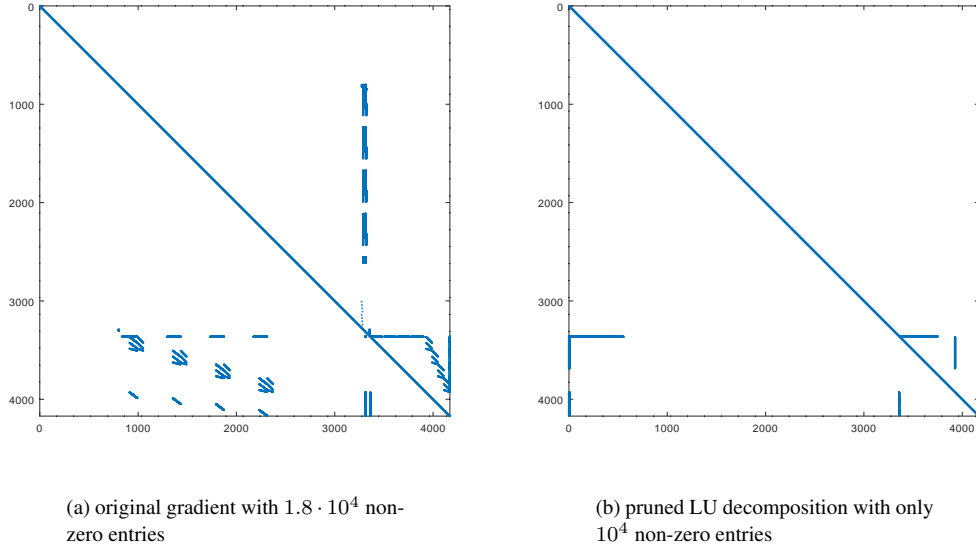(b) pruned LU decomposition with only 10^4 non-zero entries

Figure 6. Sparsity pattern of the gradient of the algebraic part and of its pruned LU decomposition (see text). Please note that the LU decomposition uses a different ordering of the matrix rows and columns. This specific example features a very sparse gradient, the number of non-zeros in the gradient can also be one to two magnitudes higher depending on the choice of models.

when its state or input vector do not change. This happens during the simplified Newton iteration of the outputs when there are only changes in some part of the output vector. These changes then *travel* along the edges of the model graph (Figure 1). Here, the abort tolerance for the simplified Newton iteration is set to

$$\epsilon_{tol} := 0.001 \cdot \min\left(1, \left(\frac{h}{h_{\text{ref}}}\right)^p\right),$$

where h_{ref} denotes the timestep size related to 100 timesteps per revolution and p the order of the HEERK scheme. This formula ensures that both for large timesteps the algebraic part is solved with a desired accuracy as well as that the convergence order of the HEERK scheme is maintained. When choosing a slightly larger leading constant in the formula (e.g. 0.01 instead of 0.001), the number of required Newton iterations and of g_i evaluations decreases.

From Table 5 we see that very few Newton iterations suffice to approximate the solution of the algebraic part of the system accurately enough.

4.3. Heuristics for the gradient of the algebraic part

For our application the gradient of the algebraic part is usually very sparse because the output vector of a model often depends mostly on its state vector. As an example, the output vector of the FE rotor model contains the position, velocity and acceleration of collocation points on the rotor blades. Only the acceleration depends on the input vector which contains the aerodynamic forces at those points. We can exploit this to reduce the computational costs of the simplified Newton scheme for the algebraic part of the system.

Figure 6a shows the sparsity pattern of the gradient of the operation $\mathbf{y} - g(t, \mathbf{x}, \mathbf{y})$ wrt. $\mathbf{y}$. However, only a part of the non-zero matrix entries are actually relevant for us: output vector segments $\mathbf{y}_i$

that are not part of a loop in the model graph (Figure 1) can be calculated simply by evaluating the models output equations g_i in a specific order (when the entries in loops are known). We can also apply this idea on the level of individual output vector entries using the adjacency graph of the sparse gradient matrix. This is a heuristic because we cannot ensure that zero entries in the gradient evaluated at some points will remain zero for all points. Still, this criterion is useful in practice.

So, to reduce the computational costs we remove all lines and rows in the gradient (except for the unit diagonal) that do not belong to an output vector entry in a loop in the adjacency graph. Afterwards, we compute a sparse LU decomposition. Even with fill-in we obtain a matrix that is usually even more sparse (see Figure 6b). The computed LU decomposition can be reused for at least one revolution in most of our simulations (simplified Newton iteration). When we need to recalculate the gradient, we also exploit the known desired sparsity pattern to compute only the required matrix entries.

Note that most models use numerical differentiation (forward differences) to calculate their block of the gradient which seems to work well. Since the gradient is only updated rarely (and only the required parts of it), the computational time for recalculating the gradient and its LU decomposition is negligible.

5. Conclusion

We successfully combine half-explicit Runge-Kutta methods with exponential integrators to tackle problems from helicopter simulation. These half-explicit exponential Runge-Kutta (HEERK) methods focus on index-1 DAE systems with a stiff linear part. In a small numerical example we extracted a stiff linear part by using an approximate linearization of the problem. This illustrates the robustness of the approach and that it can be used in a heuristic way: the original problem does not need to have a special form but we can construct a suitable linear factor by using some rough approximation of a gradient evaluated at some point.

Furthermore, we show results for a simulation of a helicopter rotor including finite element beams for the rotor blades and a semi-empirical model for the aerodynamics. We express the problem as a set of interconnected ODE models in state-space form. The combined system of all models forms an index-1 DAE with a block-diagonal stiff linear part. This allows a modular software design where we can easily replace one ODE model by another with similar outputs. The latter is also interesting from an engineering point of view as it facilitates to compare results obtained with different models for the same physical system.

In contrast to fully-implicit methods the half-explicit methods require only very few simplified Newton iterations for the algebraic part. Thus, they avoid calculating the gradient of the dynamic part of the DAE. The required matrix exponentials (or their application on a vector) can be computed efficiently if a special form of the extracted stiff linear part can be exploited, such as the block-diagonal structure in our case.

As a result, HEERK methods are less expensive than fully-implicit methods but they still provide a stable and accurate solution of the presented problem for appropriate timestep sizes. This cannot be achieved with (normal) half-explicit methods because they are unstable unless very tiny timesteps are used.

For future work we want to analyze the behavior of HEERK methods for larger helicopter simulations with more accurate models for structure- and aerodynamics.

In addition, we will focus on the automatic extraction of a suitable linear part S for a given problem since a smart choice of S is crucial for both performance and stability of HEERK methods. This can be done in a preprocessing step. An automatic approach could for example use a partial eigenvalue decomposition of the gradient of the dynamic part of the DAE.

Further, we want to investigate if and how the current approach can be combined with geometric

integrators to preserve the structure of some models in the system and thus obtain e.g. discrete energy conservation.

Acknowledgments

References

[1] M. Arnold, K. Strehmel, and R. Weiner. Half-explicit Runge-Kutta methods for semi-explicit differential-algebraic equations of index 1. *Numerische Mathematik*, 64(1):409–431, 1993.

[2] G. Beylkin, J.M. Keiser, and L. Vozovoi. A new class of time discretization schemes for the solution of nonlinear PDEs. *Journal of Computational Physics*, 147(2):362–387, 1998.

[3] J.C. Butcher. A history of Runge-Kutta methods. *Applied Numerical Mathematics*, 20(3):247–260, 1996.

[4] J.C. Butcher and G. Wanner. Runge-Kutta methods: some historical notes. *Applied Numerical Mathematics*, 22(1–3):113–151, 1996.

[5] M. Condon, J. Gao, and A. Iserles. On asymptotic expansion solvers for highly oscillatory semi-explicit DAEs. *Discrete & Continuous Dynamical Systems - A*, 36(9):4813–4837, 2016.

[6] SM Cox and PC Matthews. Exponential time differencing for stiff systems. *Journal of Computational Physics*, 176(2):430–455, 2002.

[7] E. Hairer, C. Lubich, and M. Roche. *The numerical solution of differential-algebraic systems by Runge-Kutta methods, vol. 1409 of Lect.* Notes in Math., Springer, Berlin, 1989.

[8] E. Hairer, C. Lubich, and G. Wanner. *Geometric Numerical Integration*, volume 31 of *Spriger Series in Computational Mathematics*. Springe, 2006.

[9] M. Hochbruck and A. Ostermann. Explicit exponential Runge-Kutta methods for semilinear parabolic problems. *SIAM Journal on Numerical Analysis*, 43(3):1069–1090, 2005.

[10] M. Hochbruck and A. Ostermann. Exponential Runge-Kutta methods for parabolic problems. *Applied Numerical Mathematics*, 53(2–4):323–339, 2005.

[11] M. Hochbruck and A. Ostermann. Exponential integrators. *Acta Numerica*, 19:209–286, 2010.

[12] J.C. Houbolt, G.W. Brooks, and United States. National Advisory Committee for Aeronautics. *Differential Equations of Motion for Combined Flapwise Bending, Chordwise Bending, and Torsion of Twisted Nonuniform Rotor Blades*. NACA technical report. National Advisory Committee for Aeronautics, 1957.

[13] W. Johnson. *Rotorcraft Aeromechanics*. Cambridge Aerospace Series. Cambridge University Press, 2013.

[14] E. Kohlwey. Towards helicopter simulation with an index-1 differential-algebraic equations system - efficient time integration methods. Master's thesis, Universität zu Köln, 2017.

[15] U. Leiss and S. Wagner. Toward a unified representation of rotor blade airloads with emphasis on unsteady and viscous effects. 1987.

[16] B.V. Minchev. *Exponential integrators for semilinear problems*. University of Bergen, 2004.

[17] C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49, 2003.

[18] L. Sanches, G. Michon, A. Berlioz, and D. Alazard. Instability zones for isotropic and anisotropic multi-bladed rotor configurations. *Mechanism and Machine Theory*, 46(8):1054–1065, 2011.

Appendix A. The Helicopter model

We adapt the mechanical model by [18]. In contrary to their model, we do not consider anisotropic blades, hence the masses and inertias of every blade are equal in our model. It contains the following variables:

Var.	Explanation	Value [unit]
a	Rotor eccentricity	0.2 [m]
b	Blade length	2.5 [m]
F	External force vector of the original model	
F_{DAE}	External force vector of the advanced DAE model	
F_{mon}	External force vector of the advanced monolithic model	
G	Damping matrix of the original model	
G_{DAE}	Damping matrix of the advanced DAE model	
G_{mon}	Damping matrix of the advanced monolithic model	
I_{zb}	Lag rotational inertia of a blade around its center of gravity	259 [kg m^2]
K	Stiffness matrix of the original model	
K_{DAE}	Stiffness matrix of the advanced DAE model	
K_{mon}	Stiffness matrix of the advanced monolithic model	
K_s	Stiffness coefficient of the mast	
M	Mass matrix of the original model	
M_{DAE}	Mass matrix of the advanced DAE model	
M_{mon}	Mass matrix of the advanced monolithic model	
m_b	Mass of a blade	31.9 [kg]
m_f	Fuselage mass	2902.9 [kg]
r_a	$\sqrt{ar_b}$	
r_m	Ratio between the static moment of a blade over the total mass of the helicopter	[m]
r_b	Ratio between the static moment over the total lead-lag rotational inertia of a blade	[m^{-1}]
$\mathbf{u}(t)$	Vector of general variables of the original model	
$\mathbf{u}_{mon}(t)$	Vector of general variables of the advanced monolithic model	
$x_{\mathrm{Fus}}(t)$	Longitudinal displacement of the fuselage	[m]
$y_{\mathrm{Fus}}(t)$	Transversal displacement of the fuselage	[m]
$\varphi_i(t)$	Lead-lag angle of ith blade	[rad]
$\varphi_{RH}(t)$	Rotational angle of the rotor head	[rad]
ω_b	Lag resonance frequency of the a blade at $\Omega = 0$	1.5 [Hz]
$\omega_{RH}(t)$	Rotor head's resonance frequency	[Hz]
ω_x	Fuselage's resonance frequency in x direction	3 [Hz]
ω_y	Fuselage's resonance frequency in y direction	3 [Hz]
Ω	Rotor speed	[Hz]

Table 6. Variables for the rotor model given, cf. [18], Section 1.1 and Table 1

This section contains three parts. We first state the original model from Sanches et al. [18] in Subsection A.1. In Subsection A.2, we present the advanced model in a monolithic form and subsequently we deal with the advanced model in a DAE formulation in Subsection A.3.

A.1. Original model

The matrices of the original model are

$$
M = \begin{pmatrix}
1 & 0 & -r_m \sin(\Omega t) & -r_m \cos(\Omega t) & r_m \sin(\Omega t) & r_m \cos(\Omega t) \\
0 & 1 & r_m \cos(\Omega t) & -r_m \sin(\Omega t) & -r_m \cos(\Omega t) & r_m \sin(\Omega t) \\
-r_b \sin(\Omega t) & r_b \cos(\Omega t) & 1 & 0 & 0 & 0 \\
-r_b \cos(\Omega t) & -r_b \sin(\Omega t) & 0 & 1 & 0 & 0 \\
r_b \sin(\Omega t) & -r_b \cos(\Omega t) & 0 & 0 & 1 & 0 \\
r_b \cos(\Omega t) & r_b \sin(\Omega t) & 0 & 0 & 0 & 1
\end{pmatrix},
$$

$$
G = \begin{pmatrix}
0 & 0 & -2r_m \Omega \cos(\Omega t) & 2r_m \Omega \sin(\Omega t) & 2r_m \Omega \cos(\Omega t) & -2r_m \Omega \sin(\Omega t) \\
0 & 0 & -2r_m \Omega \sin(\Omega t) & -2r_m \Omega \cos(\Omega t) & 2r_m \Omega \sin(\Omega t) & 2r_m \Omega \cos(\Omega t) \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix},
$$

$$
K = \begin{pmatrix}
\omega_x^2 & 0 & \Omega^2 r_m \sin(\Omega t) & \Omega^2 r_m \cos(\Omega t) & -\Omega^2 r_m \sin(\Omega t) & -\Omega^2 r_m \cos(\Omega t) \\
0 & \omega_y^2 & -\Omega^2 r_m \cos(\Omega t) & \Omega^2 r_m \sin(\Omega t) & \Omega^2 r_m \cos(\Omega t) & -\Omega^2 r_m \sin(\Omega t) \\
0 & 0 & \Omega^2 a r_b + \omega_b^2 & 0 & 0 & 0 \\
0 & 0 & 0 & \Omega^2 a r_b + \omega_b^2 & 0 & 0 \\
0 & 0 & 0 & 0 & \Omega^2 a r_b + \omega_b^2 & 0 \\
0 & 0 & 0 & 0 & 0 & \Omega^2 a r_b + \omega_b^2
\end{pmatrix}
$$

and

$$
F = \mathbf{0}_7 \in \mathbb{R}^7.
$$

The model with

$$
\mathbf{u}(t) = (x_{\text{Fus}}(t), y_{\text{Fus}}(t), \varphi_1(t), \varphi_2(t), \varphi_3(t), \varphi_4(t))^T
$$

reads

$$
M\ddot{\mathbf{u}}(t) + G\dot{\mathbf{u}}(t) + K\mathbf{u}(t) = F. \tag{13}
$$

With

$$
\mathbf{v}(t) := (\dot{x}_{\text{Fus}}(t), \dot{y}_{\text{Fus}}(t), \dot{\varphi}_1(t), \dot{\varphi}_2(t), \dot{\varphi}_3(t), \dot{\varphi}_4(t))^T,
$$

we transform system (13) into a first order system in the variables $\mathbf{u}(t)$ and $\mathbf{v}(t)$

$$
\begin{cases}
\dot{\mathbf{u}}(t) = & \mathbf{v}(t) \\
\dot{\mathbf{v}}(t) = & -M^{-1}G\mathbf{v}(t) - M^{-1}K\mathbf{u}(t) + M^{-1}F.
\end{cases}
$$

A.2. Advanced model in monolithic form

In the advanced model, we have additional springs at both ends of the mast that connects the fuselage with the rotor. We need additional constants and variables to model the occurring behavior. Let

- K_s denote the stiffness coefficient of the mast
- $\varphi_{RH}(t)$ denote the rotational angle of the rotor head.

Then we define

$$\omega_i(t) := \dot{\varphi}_i(t), \; i = 1, \ldots, 4,$$

$$\omega_{RH}(t) := \dot{\varphi}_{RH}(t),$$

$$\alpha_{RH}(t) := \dot{\omega}_{RH}(t),$$

$$\omega_s := \sqrt{\frac{K_s}{4(a+b)^2 m_b + 4I_{zb}}},$$

$$r_{sb} := \frac{b(a+b)m_b + I_{zb}}{4(a+b)^2 m_b + 4I_{zb}}.$$

The model matrices become

$$
M_{mon} = \left(
\begin{array}{ccccc}
1 & 0 & -r_m \sin(\varphi_{RH}(t)) & -r_m \cos(\varphi_{RH}(t)) & r_m \sin(\varphi_{RH}(t)) \\
0 & 1 & r_m \cos(\varphi_{RH}(t)) & -r_m \sin(\varphi_{RH}(t)) & -r_m \cos(\varphi_{RH}(t)) \\
-r_b \sin(\varphi_{RH}(t)) & r_b \cos(\varphi_{RH}(t)) & 1 & 0 & 0 \\
-r_b \cos(\varphi_{RH}(t)) & -r_b \sin(\varphi_{RH}(t)) & 0 & 1 & 0 \\
r_b \sin(\varphi_{RH}(t)) & -r_b \cos(\varphi_{RH}(t)) & 0 & 0 & 1 \\
r_b \cos(\varphi_{RH}(t)) & r_b \sin(\varphi_{RH}(t)) & 0 & 0 & 0 \\
0 & 0 & r_{sb} & r_{sb} & r_{sb}
\end{array}
\right.
$$

$$
\left.
\begin{array}{cc}
r_m \cos(\varphi_{RH}(t)) & -r_m \left((\varphi_1(t) - \varphi_3(t)) \cos(\varphi_{RH}(t)) - \sin(\varphi_{RH}(t))(\varphi_2(t) - \varphi_4(t)) \right) \\
r_m \sin(\varphi_{RH}(t)) & -r_m \left((\varphi_2(t) - \varphi_4(t)) \cos(\varphi_{RH}(t)) + \sin(\varphi_{RH}(t))(\varphi_1(t) - \varphi_3(t)) \right) \\
0 & r_a^2 + 1 \\
0 & r_a^2 + 1 \\
0 & r_a^2 + 1 \\
1 & r_a^2 + 1 \\
r_{sb} & 1
\end{array}
\right),
$$

$$G_{mon} = 0_{7 \times 7},$$

and

$$
K_{mon} = \left(
\begin{array}{ccccccc}
\omega_x^2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \omega_y^2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & r_a^2 \omega_{RH}^2(t) + \omega_b^2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & r_a^2 \omega_{RH}^2(t) + \omega_b^2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & r_a^2 \omega_{RH}^2(t) + \omega_b^2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & r_a^2 \omega_{RH}^2(t) + \omega_b^2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \omega_s^2
\end{array}
\right).
$$

With $\varphi_{2,4} = \varphi_2(t) - \varphi_4(t)$, $\varphi_{3,1} = \varphi_3(t) - \varphi_1(t)$, $\varphi_{RH} := \varphi_{RH}(t) \; \omega_{RH} := \omega_{RH}(t)$ and $\omega_i := \omega_i(t)$, the right side of the monolithic model F_{mon} takes the form

$$
F_{mon} = \left(
\begin{array}{c}
-r_m \left((\varphi_{2,4}\omega_{RH} - 2\omega_1 + 2\omega_3) \cos(\varphi_{RH}) - (\varphi_{3,1}\omega_{RH} - 2\omega_2 + 2\omega_4) \sin(\varphi_{RH}) \right) \omega_{RH} \\
-r_m \left((\varphi_{3,1}\omega_{RH} - 2\omega_2 + 2\omega_4) \cos(\varphi_{RH}) + (\varphi_{2,4}\omega_{RH} - 2\omega_1 + 2\omega_3) \sin(\varphi_{RH}) \right) \omega_{RH} \\
0 \\
0 \\
0 \\
0 \\
\omega_s^2 \Omega t
\end{array}
\right).
$$

The model with

$$\mathbf{u}_{mon}(t) = (x_{\mathrm{Fus}}(t), y_{\mathrm{Fus}}(t), \varphi_1(t), \varphi_2(t), \varphi_3(t), \varphi_4(t), \varphi_{RH}(t))^T$$

reads

$$M_{mon}\ddot{\mathbf{u}}_{mon}(t) + G_{mon}\dot{\mathbf{u}}_{mon}(t) + K_{mon}\mathbf{u}_{mon}(t) = F_{mon}. \tag{14}$$

With

$$\mathbf{v}_{mon}(t) = (\dot{x}_{\text{Fus}}(t), \dot{y}_{\text{Fus}}(t), \dot{\varphi}_1(t), \dot{\varphi}_2(t), \dot{\varphi}_3(t), \dot{\varphi}_4(t), \dot{\varphi}_{RH}(t))^T,$$

we transform system (14) into a first order system in the variables $\mathbf{u}_{mon}(t)$ and $\mathbf{v}_{mon}(t)$

$$\begin{cases} \dot{\mathbf{u}}_{mon}(t) = & \mathbf{v}_{mon}(t) \\ \dot{\mathbf{v}}_{mon}(t) = & -M_{mon}^{-1}G_{mon}\mathbf{v}_{mon}(t) - M_{mon}^{-1}K_{mon}\mathbf{u}_{mon}(t) + M_{mon}^{-1}F_{mon}. \end{cases}$$

Defining

$$\mathbf{x}_{mon} := \begin{pmatrix} \mathbf{u}_{mon}(t) \\ \mathbf{v}_{mon}(t) \end{pmatrix},$$

we obtain a 14-dimensional linear monolithic system

$$\dot{\mathbf{x}}_{mon} = \begin{pmatrix} 0_{7\times7} & I_{7\times7} \\ -M_{mon}^{-1}K_{mon} & -M_{mon}^{-1}G_{mon} \end{pmatrix} \cdot \mathbf{x}_{mon} + \begin{pmatrix} \mathbf{0}_7 \\ M_{mon}^{-1}F_{mon} \end{pmatrix} =: f_{mon}(t, \mathbf{x}_{mon}(t)),$$

where $0_{7\times7}$ denotes the 7×7-zero matrix, $I_{7\times7}$ denotes the 7×7-identity matrix and $\mathbf{0}_7$ denotes the 7-dimensional zero vector.

Remark 15.
For the implementation, we use the variables $\bar{\varphi}_{RH}(t) := \varphi_{RH}(t) - \Omega t$ and $\bar{\omega}_{RH}(t) := \dot{\bar{\varphi}}_{RH}(t) = \omega_{RH}(t) - \Omega$. Accordingly, all entries of M_{mon}, K_{mon} and F_{mon} containing $\varphi_{RH}(t)$ or $\omega_{RH}(t)$ need to be adapted. Additionally, the last entry of F_{mon} becomes zero.

A.3. Advanced model in DAE form

The model matrices for the advanced model in DAE formulation read

$$M_{DAE} = \begin{pmatrix} 1 & 0 & -r_m\sin(\varphi_{RH}(t)) & -r_m\cos(\varphi_{RH}(t)) & r_m\sin(\varphi_{RH}(t)) & r_m\cos(\varphi_{RH}(t)) \\ 0 & 1 & r_m\cos(\varphi_{RH}(t)) & -r_m\sin(\varphi_{RH}(t)) & -r_m\cos(\varphi_{RH}(t)) & r_m\sin(\varphi_{RH}(t)) \\ -r_b\sin(\varphi_{RH}(t)) & r_b\cos(\varphi_{RH}(t)) & 1 & 0 & 0 & 0 \\ -r_b\cos(\varphi_{RH}(t)) & -r_b\sin(\varphi_{RH}(t)) & 0 & 1 & 0 & 0 \\ r_b\sin(\varphi_{RH}(t)) & -r_b\cos(\varphi_{RH}(t)) & 0 & 0 & 1 & 0 \\ r_b\cos(\varphi_{RH}(t)) & r_b\sin(\varphi_{RH}(t)) & 0 & 0 & 0 & 1 \end{pmatrix},$$

$$G_{DAE} = \begin{pmatrix} 0 & 0 & -2r_m\omega_{RH}(t)\cos(\varphi_{RH}(t)) & 2r_m\omega_{RH}(t)\sin(\varphi_{RH}(t)) & 2r_m\omega_{RH}(t)\cos(\varphi_{RH}(t)) & -2r_m\omega_{RH}(t)\sin(\varphi_{RH}(t)) \\ 0 & 0 & -2r_m\omega_{RH}(t)\sin(\varphi_{RH}(t)) & -2r_m\omega_{RH}(t)\cos(\varphi_{RH}(t)) & 2r_m\omega_{RH}(t)\sin(\varphi_{RH}(t)) & 2r_m\omega_{RH}(t)\cos(\varphi_{RH}(t)) \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

and

$$
K_{DAE} =
\begin{pmatrix}
\omega_x^2 & 0 & -r_m(-\omega_{RH}^2(t)\sin(\varphi_{RH}(t))+\alpha_{RH}(t)\cos(\varphi_{RH}(t))) & r_m(\omega_{RH}^2(t)\cos(\varphi_{RH}(t))+\alpha_{RH}(t)\sin(\varphi_{RH}(t))) \\
0 & \omega_y^2 & -r_m(\omega_{RH}^2(t)\cos(\varphi_{RH}(t))+\alpha_{RH}(t)\sin(\varphi_{RH}(t))) & -r_m(-\omega_{RH}^2(t)\sin(\varphi_{RH}(t))+\alpha_{RH}(t)\cos(\varphi_{RH}(t))) \\
0 & 0 & r_a^2\omega_{RH}^2(t)+\omega_b^2 & 0 \\
0 & 0 & 0 & r_a^2\omega_{RH}^2(t)+\omega_b^2 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}
$$

$$
\begin{matrix}
r_m(-\omega_{RH}^2(t)\sin(\varphi_{RH}(t))+\alpha_{RH}(t)\cos(\varphi_{RH}(t))) & -r_m(\omega_{RH}^2(t)\cos(\varphi_{RH}(t))+\alpha_{RH}(t)\sin(\varphi_{RH}(t))) \\
r_m(\omega_{RH}^2(t)\cos(\varphi_{RH}(t))+\alpha_{RH}(t)\sin(\varphi_{RH}(t))) & r_m(-\omega_{RH}^2(t)\sin(\varphi_{RH}(t))+\alpha_{RH}(t)\cos(\varphi_{RH}(t))) \\
0 & 0 \\
0 & 0 \\
r_a^2\omega_{RH}^2(t)+\omega_b^2 & 0 \\
0 & r_a^2\omega_{RH}^2(t)+\omega_b^2
\end{matrix} \Bigg) .
$$

The right side becomes

$$
F_{DAE} =
\begin{pmatrix}
0 \\
0 \\
(-r_a^2 - 1)\alpha_{RH}(t) \\
(-r_a^2 - 1)\alpha_{RH}(t) \\
(-r_a^2 - 1)\alpha_{RH}(t) \\
(-r_a^2 - 1)\alpha_{RH}(t)
\end{pmatrix} .
$$

For the two models we then obtain:

Model 1:
Inputs: $\varphi_{RH}(t), \omega_{RH}(t)$ and $\alpha_{RH}(t)$
Outputs: $\dot{\omega}_1(t), \dot{\omega}_2(t), \dot{\omega}_3(t), \dot{\omega}_4(t)$
The local state vector $\mathbf{x}_1(t)$ is given by

$$
\mathbf{x}_1(t) = \begin{pmatrix} \mathbf{u}(t) \\ \mathbf{v}(t) \end{pmatrix} = (x_{\text{Fus}}(t), y_{\text{Fus}}(t), \varphi_1(t), \varphi_2(t), \varphi_3(t), \varphi_4(t), \dot{x}_{\text{Fus}}(t), \dot{y}_{\text{Fus}}(t), \omega_1(t), \omega_2(t), \omega_3(t), \omega_4(t))^T \in \mathbb{R}^{12}.
$$

This yields

$$
\dot{\mathbf{x}}_1(t) = f_1(t, \mathbf{x}_1(t), \mathbf{y}_2(t)) = \begin{pmatrix} 0_{6\times 6} & I_{6\times 6} \\ -M_{DAE}^{-1}K_{DAE} & -M_{DAE}^{-1}G_{DAE} \end{pmatrix} \mathbf{x}_1(t) + \begin{pmatrix} \mathbf{0}_6 \\ M_{DAE}^{-1}F_{DAE} \end{pmatrix},
$$

where $0_{6\times 6}$ denotes the 6×6-zero matrix, $I_{6\times 6}$ denotes the 6×6-identity matrix and $\mathbf{0}_6$ denotes the 6-dimensional zero vector. The dependence on $\mathbf{y}_2(t) = (\varphi_{RH}(t), \omega_{RH}(t), \alpha_{RH}(t))^T$ is manifested in the definitions of $M_{DAE}, K_{DAE}, G_{DAE}$ and F_{DAE}.

We see that the outputs of the first model $(\dot{\omega}_1(t), \dot{\omega}_2(t), \dot{\omega}_3(t), \dot{\omega}_4(t))$ are the time derivatives of the 9th to 12th entries of $\mathbf{x}_1(t)$. Hence, in order to state the algebraic function $g_1(t, \mathbf{x}_1(t), \mathbf{y}_2(t))$

explicitly, we need to calculate $-M_{DAE}^{-1}K_{DAE}$ and $-M_{DAE}^{-1}G_{DAE}$:

$$\mathbf{y}_1(t) = \begin{pmatrix} \dot{\omega}_1(t) \\ \dot{\omega}_2(t) \\ \dot{\omega}_3(t) \\ \dot{\omega}_4(t) \end{pmatrix} = \begin{pmatrix} \dot{\mathbf{x}}_1^{(9)} \\ \dot{\mathbf{x}}_1^{(10)} \\ \dot{\mathbf{x}}_1^{(11)} \\ \dot{\mathbf{x}}_1^{(12)} \end{pmatrix} = \frac{1}{2r_b r_m - 1} \begin{pmatrix} \sin(\varphi_{RH}(t))r_b\omega_x^2 & -\cos(\varphi_{RH}(t))r_b\omega_y^2 \\ \cos(\varphi_{RH}(t))r_b\omega_x^2 & \sin(\varphi_{RH}(t))r_b\omega_y^2 \\ -\sin(\varphi_{RH}(t))r_b\omega_x^2 & \cos(\varphi_{RH}(t))r_b\omega_y^2 \\ -\cos(\varphi_{RH}(t))r_b\omega_x^2 & -\sin(\varphi_{RH}(t))r_b\omega_y^2 \end{pmatrix}$$

$$\begin{pmatrix}
\begin{matrix}((-r_a^2+1)r_m r_b + r_a^2)\omega_{RH}(t)^2 - \omega_b^2(r_b r_m - 1) \\ -r_b r_m \alpha_{RH}(t) \\ -r_b r_m(\omega_{RH}(t)^2 + r_a^2\omega_{RH}(t)^2 + \omega_b^2) \\ r_b r_m \alpha_{RH}(t)\end{matrix} & \begin{matrix}r_b r_m \alpha_{RH}(t) \\ ((-r_a^2+1)r_m r_b + r_a^2)\omega_{RH}(t)^2 - \omega_b^2(r_b r_m - 1) \\ -r_b r_m \alpha_{RH}(t) \\ -r_b r_m(\omega_{RH}(t)^2 + r_a^2\omega_{RH}(t)^2 + \omega_b^2)\end{matrix} \\[4em]
\begin{matrix}-r_b r_m(\omega_{RH}(t)^2 + r_a^2\omega_{RH}(t)^2 + \omega_b^2) \\ r_b r_m \alpha_{RH}(t) \\ ((-r_a^2+1)r_m r_b + r_a^2)\omega_{RH}(t)^2 - \omega_b^2(r_b r_m - 1) \\ -r_b r_m \alpha_{RH}(t)\end{matrix} & \begin{matrix}-r_b r_m \alpha_{RH}(t) \\ -r_b r_m(\omega_{RH}(t)^2 + r_a^2\omega_{RH}(t)^2 + \omega_b^2) \\ r_b r_m \alpha_{RH}(t) \\ ((-r_a^2+1)r_m r_b + r_a^2)\omega_{RH}(t)^2 - \omega_b^2(r_b r_m - 1)\end{matrix}
\end{pmatrix}$$

$$\cdot \begin{pmatrix} x_{\text{Fus}}(t) \\ y_{\text{Fus}}(t) \\ \varphi_1(t) \\ \varphi_2(t) \\ \varphi_3(t) \\ \varphi_4(t) \end{pmatrix} + \frac{2\omega_{RH}(t)r_b r_m}{2r_b r_m - 1} \begin{pmatrix} 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 1 & 0 & -1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \omega_1(t) \\ \omega_2(t) \\ \omega_3(t) \\ \omega_4(t) \end{pmatrix} - (r_a^2+1)\alpha_{RH}(t)\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

$$=: g_1(t, \mathbf{x}_1(t), \mathbf{y}_2(t)). \tag{16}$$

From the first model we do not extract a stiff linear part. Hence,

$$S_1 := 0_{12 \times 12}$$

and

$$\tilde{f}_1(t, \mathbf{x}_1(t), \mathbf{y}_2(t)) := f_1(t, \mathbf{x}_1(t), \mathbf{y}_2(t)).$$

Model 2:
Inputs: $\dot{\omega}_1(t), \dot{\omega}_2(t), \dot{\omega}_3(t), \dot{\omega}_4(t)$
Outputs: $\varphi_{RH}(t), \omega_{RH}(t)$ and $\alpha_{RH}(t)$
The governing equation of the second model reads

$$\dot{\omega}_{RH}(t) = \omega_s^2(\Omega t - \varphi_{RH}(t)) - r_{sb}\dot{\omega}_1(t) - r_{sb}\dot{\omega}_2(t) - r_{sb}\dot{\omega}_3(t) - r_{sb}\dot{\omega}_4(t).$$

With the local state vector

$$\mathbf{x}_2(t) := \begin{pmatrix} \varphi_{RH}(t) \\ \omega_{RH}(t) \end{pmatrix} \in \mathbb{R}^2,$$

this yields

$$\dot{\mathbf{x}}_2(t) = \begin{pmatrix} 0 & 1 \\ -\omega_s^2 & 0 \end{pmatrix}\mathbf{x}_2(t) + \begin{pmatrix} 0 \\ \omega_s^2\Omega t - r_{sb}\dot{\omega}_1(t) - r_{sb}\dot{\omega}_2(t) - r_{sb}\dot{\omega}_3(t) - r_{sb}\dot{\omega}_4(t) \end{pmatrix}.$$

Since ω_s^2 is the variable which is responsible for the stiffness of the system, we want it to be solely part of the first linear summand of the differential equations system. Hence we define

$$\mathbf{x}_2^{(1)} := \bar{\varphi}_{RH}(t) := \varphi_{RH}(t) - \Omega t,$$
$$\mathbf{x}_2^{(2)} := \bar{\omega}_{RH}(t) := \dot{\bar{\varphi}}_{RH}(t) = \bar{\varphi}_{RH}(t) - \Omega,$$

which yields

$$\dot{\mathbf{x}}_2(t) = \begin{pmatrix} 0 & 1 \\ -\omega_s^2 & 0 \end{pmatrix} \mathbf{x}_2(t) + \begin{pmatrix} 0 \\ -r_{sb}\dot{\omega}_1(t) - r_{sb}\dot{\omega}_2(t) - r_{sb}\dot{\omega}_3(t) - r_{sb}\dot{\omega}_4(t) \end{pmatrix}.$$

Accordingly, we define

$$S_2 := \begin{pmatrix} 0 & 1 \\ -\omega_s^2 & 0 \end{pmatrix}$$

and

$$\tilde{f}_2(t, \mathbf{x}_2(t), \mathbf{y}_1(t)) := \begin{pmatrix} 0 \\ -r_{sb}\dot{\omega}_1(t) - r_{sb}\dot{\omega}_2(t) - r_{sb}\dot{\omega}_3(t) - r_{sb}\dot{\omega}_4(t) \end{pmatrix}.$$

The local output vector and hence the local output function $g_2(t, \mathbf{x}_2(t), \mathbf{y}_1(t))$ is given by

$$
\begin{aligned}
\mathbf{y}_2(t) &= \begin{pmatrix} \varphi_{RH}(t) \\ \omega_{RH}(t) \\ \alpha_{RH}(t) \end{pmatrix} = \begin{pmatrix} \mathbf{x}_2^{(1)}(t) + \Omega t \\ \mathbf{x}_2^{(2)}(t) + \Omega \\ \dot{\mathbf{x}}_2^{(2)}(t) \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{x}_2^{(1)}(t) + \Omega t \\ \mathbf{x}_2^{(2)}(t) + \Omega \\ -\omega_s^2 \mathbf{x}_2^{(1)}(t) - r_{sb}\dot{\omega}_1(t) - r_{sb}\dot{\omega}_2(t) - r_{sb}\dot{\omega}_3(t) - r_{sb}\dot{\omega}_4(t) \end{pmatrix} =: g_2(t, \mathbf{x}_2(t), \mathbf{y}_1(t)).
\end{aligned}
\tag{17}
$$

Global model:

For the global model we then obtain the global state vector

$$\mathbf{x}(t) = (x_{\text{Fus}}(t), y_{\text{Fus}}(t), \varphi_1(t), \varphi_2(t), \varphi_3(t), \varphi_4(t), \dot{x}_{\text{Fus}}(t), \dot{y}_{\text{Fus}}(t), \omega_1(t), \omega_2(t), \omega_3(t), \omega_4(t), \bar{\varphi}_{RH}(t), \bar{\omega}_{RH}(t))^T$$

and our global output vector reads

$$\mathbf{y}(t) = (\dot{\omega}_1(t), \dot{\omega}_2(t), \dot{\omega}_3(t), \dot{\omega}_4(t), \varphi_{RH}(t), \omega_{RH}(t), \alpha_{RH}(t))^T$$

The global state and output functions read

$$f_{DAE}(t, \mathbf{x}(t), \mathbf{y}(t)) = S\mathbf{x} + \tilde{f}_{DAE}(t, \mathbf{x}(t), \mathbf{y}(t))$$

with

$$S := \begin{pmatrix} S_1 & 0_{12\times2} \\ 0_{2\times12} & S_2 \end{pmatrix},$$

$$\tilde{f}_{DAE}(t, \mathbf{x}(t), \mathbf{y}(t)) := \begin{pmatrix} \tilde{f}_1(t, \mathbf{x}_1(t), \mathbf{y}_2(t)) \\ \tilde{f}_2(t, \mathbf{x}_2(t), \mathbf{y}_1(t)) \end{pmatrix}$$

and

$$g_{DAE}(t, \mathbf{x}(t), \mathbf{y}(t)) := \begin{pmatrix} g_1(t, \mathbf{x}_1(t), \mathbf{y}_2(t)) \\ g_2(t, \mathbf{x}_2(t), \mathbf{y}_1(t)) \end{pmatrix}.$$

Since we use the Newton method on the function $\bar{g}_{DAE}(t, \mathbf{x}(t), \mathbf{y}(t)) := \mathbf{y}(t) - g_{DAE}(t, \mathbf{x}(t), \mathbf{y}(t))$ when applying half-explicit methods, we need to calculate the gradient $\nabla_{\mathbf{y}}\bar{g}_{DAE}$:

$$\nabla_{\mathbf{y}}\bar{g}_{DAE} = \begin{pmatrix}
1 & 0 & 0 & 0 & -\frac{\partial g_{DAE}^{(1)}}{\partial \varphi_{RH}(t)} & -\frac{\partial g_{DAE}^{(1)}}{\partial \omega_{RH}(t)} & -\frac{\partial g_{DAE}^{(1)}}{\partial \alpha_{RH}(t)} \\
0 & 1 & 0 & 0 & -\frac{\partial g_{DAE}^{(2)}}{\partial \varphi_{RH}(t)} & -\frac{\partial g_{DAE}^{(2)}}{\partial \omega_{RH}(t)} & -\frac{\partial g_{DAE}^{(2)}}{\partial \alpha_{RH}(t)} \\
0 & 0 & 1 & 0 & -\frac{\partial g_{DAE}^{(3)}}{\partial \varphi_{RH}(t)} & -\frac{\partial g_{DAE}^{(3)}}{\partial \omega_{RH}(t)} & -\frac{\partial g_{DAE}^{(3)}}{\partial \alpha_{RH}(t)} \\
0 & 0 & 0 & 1 & -\frac{\partial g_{DAE}^{(4)}}{\partial \varphi_{RH}(t)} & -\frac{\partial g_{DAE}^{(4)}}{\partial \omega_{RH}(t)} & -\frac{\partial g_{DAE}^{(4)}}{\partial \alpha_{RH}(t)} \\
0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
r_{sb} & r_{sb} & r_{sb} & r_{sb} & 0 & 0 & 1
\end{pmatrix}$$

with

$$\frac{\partial g_{DAE}^{(1)}}{\partial \varphi_{RH}(t)} = \frac{r_b \omega_x^2}{2r_b r_m - 1} \cos(\varphi_{RH}(t)) x_{\text{Fus}}(t) + \frac{r_b \omega_y^2}{2r_b r_m - 1} \sin(\varphi_{RH}(t)) y_{\text{Fus}}(t),$$

$$\frac{\partial g_{DAE}^{(2)}}{\partial \varphi_{RH}(t)} = -\frac{r_b \omega_x^2}{2r_b r_m - 1} \sin(\varphi_{RH}(t)) x_{\text{Fus}}(t) + \frac{r_b \omega_y^2}{2r_b r_m - 1} \cos(\varphi_{RH}(t)) y_{\text{Fus}}(t),$$

$$\frac{\partial g_{DAE}^{(3)}}{\partial \varphi_{RH}(t)} = -\frac{r_b \omega_x^2}{2r_b r_m - 1} \cos(\varphi_{RH}(t)) x_{\text{Fus}}(t) - \frac{r_b \omega_y^2}{2r_b r_m - 1} \sin(\varphi_{RH}(t)) y_{\text{Fus}}(t),$$

$$\frac{\partial g_{DAE}^{(4)}}{\partial \varphi_{RH}(t)} = \frac{r_b \omega_x^2}{2r_b r_m - 1} \sin(\varphi_{RH}(t)) x_{\text{Fus}}(t) - \frac{r_b \omega_y^2}{2r_b r_m - 1} \cos(\varphi_{RH}(t)) y_{\text{Fus}}(t),$$

$$\frac{\partial g_{DAE}^{(1)}}{\partial \omega_{RH}(t)} = \frac{2((-r_a^2 + 1)r_m r_b + r_a^2)\varphi_1(t) - 2r_b r_m(1 + r_a^2)\varphi_3(t)}{2r_b r_m - 1} \omega_{RH}(t) + \frac{2r_b r_m}{2r_b r_m - 1}(\omega_2(t) - \omega_4(t)),$$

$$\frac{\partial g_{DAE}^{(2)}}{\partial \omega_{RH}(t)} = \frac{2((-r_a^2 + 1)r_m r_b + r_a^2)\varphi_2(t) - 2r_b r_m(1 + r_a^2)\varphi_4(t)}{2r_b r_m - 1} \omega_{RH}(t) + \frac{2r_b r_m}{2r_b r_m - 1}(\omega_3(t) - \omega_1(t)),$$

$$\frac{\partial g_{DAE}^{(3)}}{\partial \omega_{RH}(t)} = \frac{2((-r_a^2 + 1)r_m r_b + r_a^2)\varphi_3(t) - 2r_b r_m(1 + r_a^2)\varphi_1(t)}{2r_b r_m - 1} \omega_{RH}(t) + \frac{2r_b r_m}{2r_b r_m - 1}(\omega_4(t) - \omega_2(t)),$$

$$\frac{\partial g_{DAE}^{(4)}}{\partial \omega_{RH}(t)} = \frac{2((-r_a^2 + 1)r_m r_b + r_a^2)\varphi_4(t) - 2r_b r_m(1 + r_a^2)\varphi_2(t)}{2r_b r_m - 1} \omega_{RH}(t) + \frac{2r_b r_m}{2r_b r_m - 1}(\omega_1(t) - \omega_3(t)),$$

$$\frac{\partial g_{DAE}^{(1)}}{\partial \alpha_{RH}(t)} = \frac{r_b r_m}{2r_b r_m - 1}(\varphi_2(t) - \varphi_4(t)) - (r_a^2 + 1),$$

$$\frac{\partial g_{DAE}^{(2)}}{\partial \alpha_{RH}(t)} = \frac{r_b r_m}{2r_b r_m - 1}(\varphi_3(t) - \varphi_1(t)) - (r_a^2 + 1),$$

$$\frac{\partial g_{DAE}^{(3)}}{\partial \alpha_{RH}(t)} = \frac{r_b r_m}{2r_b r_m - 1}(\varphi_4(t) - \varphi_2(t)) - (r_a^2 + 1),$$

$$\frac{\partial g_{DAE}^{(4)}}{\partial \alpha_{RH}(t)} = \frac{r_b r_m}{2r_b r_m - 1}(\varphi_1(t) - \varphi_3(t)) - (r_a^2 + 1).$$

Appendix B. Exponential Runge-Kutta schemes

The HEERK schemes used in this paper are based on the following exponential Runge-Kutta schemes (see [11] for a discussion of different families of schemes for a specific order).

Exponential Butcher table for the 2$^{\text{nd}}$ order 2-stage scheme HEERK2:

$$\begin{array}{c|cc} 0 & & \\ 1 & \varphi_{1,1} & \\ \hline & \frac{1}{2}\varphi_1 & \frac{1}{2}\varphi_1 \end{array}$$

Exponential Butcher table for the 3$^{\text{rd}}$ order 3-stage scheme HEERK3 (ETD2CF3):

$$
\begin{array}{c|cccc}
0 & & & \\
\frac{1}{3} & \frac{1}{3}\varphi_{1,1} & & \\
\frac{2}{3} & \frac{2}{3}\varphi_{1,2} - \frac{4}{3}\varphi_{2,2} & \frac{4}{3}\varphi_{2,2} & \\
\hline
 & \varphi_1 - \mathbf{b}_2 - \mathbf{b}_3 & 6\varphi_2 - 18\varphi_3 & -\frac{3}{2}\varphi_2 + 9\varphi_3
\end{array}
$$

The 4th order 5-stage method HEERK4 is based on the corresponding scheme proposed in [9].

Elena Kohlwey
TH Köln
Gustav-Heinemann-Ufer 54
50968 Cologne
Germany
e-mail: `Elena.Kohlwey@TH-Koeln.de`

Melven Röhrig-Zöllner
German Aerospace Center (DLR)
Simulation and Softwre Technology
Linder Höhe
51147 Cologne
Germany
Telephone: +49 2203 / 601 - 2574
ORCID-ID: 0000-0001-9851-5886
e-mail: `Melven.Roehrig-Zoellner@DLR.de`