A waveform iteration implementation for black-box multi-rate higher-order coupling

Benjamin Rodenberg[1] **and Benjamin Uekermann**[2]

[1]TUM School of Computation, Information and Technology, Technical University of Munich,
benjamin.rodenberg@cit.tum.de
[2]Institute for Parallel and Distributed Systems (IPVS), University of Stuttgart,
benjamin.uekermann@ipvs.uni-stuttgart.de

November 12, 2025

Many multiphysics simulations involve processes evolving on disparate time scales, posing a challenge for efficient coupling. A naive approach that synchronizes all processes using the smallest time scale wastes computational resources on slower processes and typically achieves only linear convergence in time. Waveform iteration is a promising numerical technique that enables higher-order, multi-rate coupling while treating coupled components as black boxes. However, applying this approach to PDE-based coupled simulations is nontrivial.

In this paper, we integrate waveform iteration into the black-box coupling library preCICE with minimal modifications to its API. We detail how this extension interacts with key preCICE features, including data mapping for non-matching meshes, quasi-Newton acceleration for strongly coupled problems, and parallel peer-to-peer communication. We then showcase that waveform iteration significantly reduces numerical errors—often by orders of magnitude. This advancement greatly enhances preCICE, benefiting its extensive user community.[1]

1 Introduction

Multiphysics simulations present numerous challenges. Among them, the combination of complex components and their efficient coupling in the time domain are two key issues explored in this paper. The first challenge can often be addressed through black-box coupling, which relies on minimal information exchange between components. However,

[1]The manuscript summarizes key parts of the dissertation of Rodenberg [25].

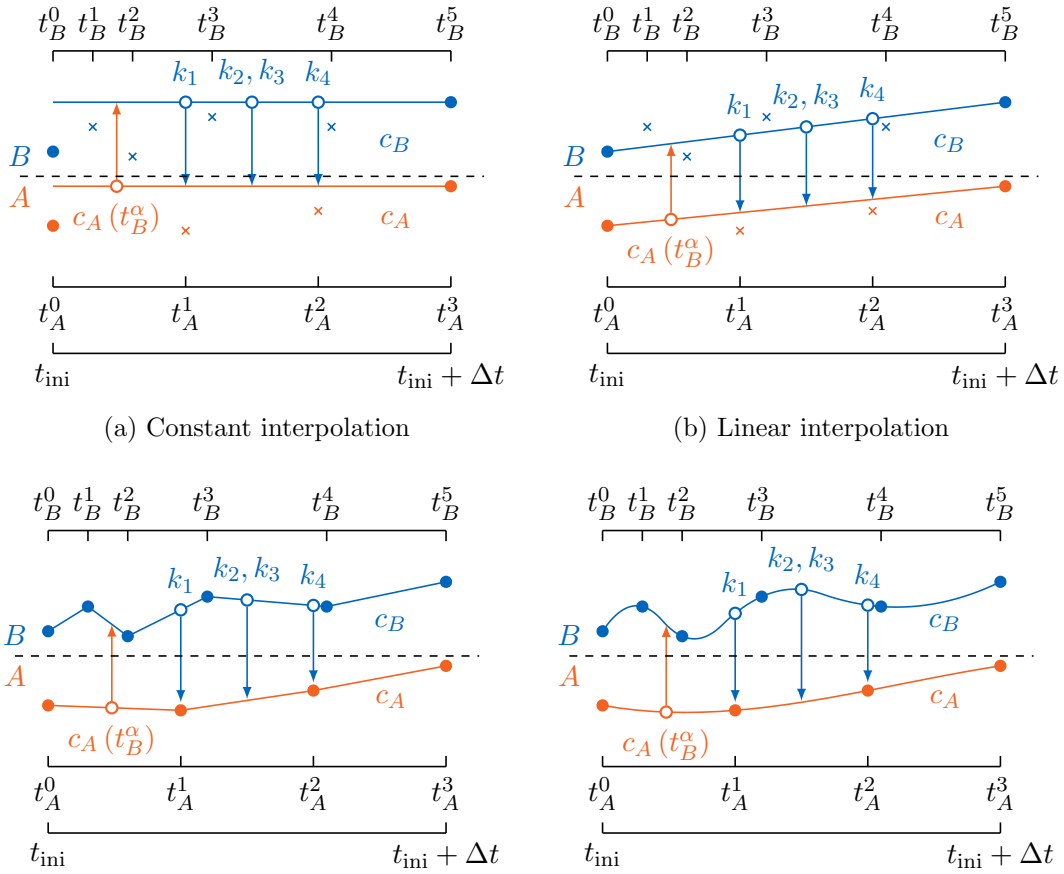
the second challenge requires making the most of this limited information. In the time domain, we must account for components whose underlying processes evolve on disparate time scales. Furthermore, different components may employ distinct time integration methods with varying mathematical properties.

These challenges are common across many application domains. For instance, Earth system modeling necessitates coupling physical processes that evolve over vastly different time scales [14]. The dynamics of ice sheets, for example, are affected by subglacial hydrological systems that evolve slowly in the interior but experience abrupt changes at the margins during peak melt seasons [1]. More broadly, the dynamics of ice sheets, oceans, and the atmosphere all operate on different temporal scales. Similar challenges arise in mechanical engineering. Munafò et al., for example, study inductively coupled plasma wind tunnels that combine fluid, electromagnetic, and solid models, whose time scales differ by two to three orders of magnitude [23]. Even classical coupled problems, such as fluid-structure interaction (FSI)—whether in hemodynamics or aeroelasticity—face challenges in the time domain. While these may not be as extreme, using different time steps and integration methods can still be advantageous to efficiently satisfy stability constraints [17].

For the sake of stability and simplicity, many black-box coupling strategies adopt the smallest time step across all components and synchronize after each step, e.g. [24, 3]. While straightforward, this approach wastes computational effort on slower-evolving processes. An alternative is to synchronize only after a larger, fixed time window, allowing each component to progress at its own rate. Such a multi-rate setup is the focus of this paper. Figure 1 illustrates various multi-rate coupling strategies using two components with three and five time steps per window. Each coupling window can either be executed once (or a fixed number of times) or repeated until a convergence criterion is met—referred to as explicit and implicit coupling, respectively. Additionally, components may advance sequentially or concurrently, denoted as serial and parallel coupling.

The most basic multi-rate approach (Figure 1a) involves exchanging only the most recent coupling values. However, this naïve method often suffers from significant coupling errors and stability issues. It also typically reduces the convergence order of the time integrators, often to only linear convergence. A simple improvement involves linear interpolation without increasing data exchange (Figure 1b), as discussed by De Moerloose et al. in the context of FSI [6]. Other simple alternatives exist. Strang splitting [34], for example, allows coupling two components, whose time step sizes differ by a factor of two. If the individual time steps are executed in the right order, quadratic convergence order can be achieved. Again in the context of FSI, variants of this approach are discussed by Farhat and Lesoinne [9]. Nonetheless, such splitting methods are limited in handling arbitrary multi-rate configurations.

Waveform iteration is a promising general-purpose black-box technique, which uses arbitrary interpolants, so-called waveforms, within each window. For instance, Figures 1c and 1d depict the usage of piecewise linear and third-degree B-spline interpolation, respectively. With appropriate interpolants, it is possible to preserve the original convergence order of the time integrators [20]. Originally developed for parallelizing integrated circuit simulations in electrical engineering [12], waveform iteration is now applied in



(a) Constant interpolation

(b) Linear interpolation

(c) Piecewise linear interpolation

(d) B-spline interpolation

Figure 1: Comparison of different multi-rate coupling strategies. Two participants A and B use three and five time steps in the time window $[t_{\text{ini}}, t_{\text{ini}} + \Delta t]$, respectively. A uses constant time steps and the forth-order Runge-Kutta RK4 method. B use adaptive time steps and the generalized-α method. Interpolants are computed from data values at different points in time and sampled at example locations to retrieve data values for the coupling partner. Notation is formally introduced in Section 2.

co-simulation and PDE contexts (e.g., see [13, 35, 22]). In prior work, we have combined waveform iteration with quasi-Newton acceleration methods for FSI [29].

Nevertheless, applying waveform iteration generically in a black-box fashion to PDE-based coupled simulations is nontrivial. Supporting general multi-rate configurations —including adaptive time stepping and multiple components—requires sophisticated communication logic. Furthermore, the method must integrate smoothly with existing building blocks for coupled PDE problems, such as data mapping for non-matching coupling meshes and fixed-point acceleration for strongly-coupled problems. To address these challenges, we explore the integration of waveform iteration into the black-box coupling library preCICE in this paper, summarizing key parts of the dissertation of Rodenberg [25].

preCICE [5] follows a peer-to-peer library approach for coupling. This means that the coupled components—referred to as participants in preCICE—call the coupler as a library and directly communicate with each other, not requiring any central orchestrator. The library provides various data mapping techniques, ranging from simple projections to advanced partition-of-unity radial-basis-function interpolation [31], as well as multiple fixed-point acceleration methods, from basic underrelaxation to sophisticated quasi-Newton schemes [21]. Communication between participants is asynchronous and parallel, using MPI or TCP/IP. The library includes ready-to-use adapters for several simulation codes (e.g., OpenFOAM [4], FEniCS [27]) and offers APIs in multiple languages, such as C++, Fortran, and Python. Its high-level API hides the underlying communication logic, exposing it through a single call to advance the coupling at each time step. While other coupling libraries exist—such as OpenPALM [8], MUI [36], YAC [16], or DTK [33]—they typically leave the coupling logic, fixed-point acceleration, and time interpolation to the user.

In the remainder of this paper, we describe the mathematical framework of waveform iteration in Section 2. Afterwards, Section 3 describes the implementation in preCICE in detail, covering the only minimal necessary changes to the user interface, the data layout, and the interplay with fixed-point acceleration, data mapping, and communication. Section 4 then presents numerical results showcasing that the new coupling approach can handle complex multi-rate setups and can achieve higher-order integration in time, significantly reducing numerical errors and, thus, enhancing preCICE, benefiting its extensive user community.

2 Waveform iteration

We consider two coupled participants, labeled A and B. While the setup is formulated for a pair of coupled participants, the implementation is designed to support an arbitrary number. Each participant operates on its respective spatial domain, denoted Ω_A and Ω_B. These domains may be identical, partially overlapping, or only in contact at a shared interface. The region of interaction between the two participants is referred to as the coupling domain $\Omega_A \cap \Omega_B$. Typically, A and B use different meshes to discretize the coupling domain. Data mappings are required to translate coupling data between these

meshes. For simplicity, we omit these mapping here (cf. [31, 5] for further discussion of data mapping in preCICE).

The temporal domain is divided into a sequence of time windows. In the following, we focus on a single such window, given by $[t_{\text{ini}}, t_{\text{ini}} + \Delta t]$, where t_{ini} is the initial time and Δt the size of the window. Within this time window, each participant performs time integration using its own scheme, which may be explicit or implicit and may feature adaptive time stepping. If needed, we denote any time step size by δ_t. The only requirement is that both participants synchronize at the final time of the window. The time steps taken by participants A and B are denoted

$$t_A^0, t_A^1, \ldots, t_A^{n_A} \quad \text{and} \quad t_B^0, t_B^1, \ldots, t_B^{n_B},$$

where $t_A^0 = t_B^0 = t_{\text{ini}}$ and $t_A^{n_A} = t_B^{n_B} = t_{\text{ini}} + \Delta t$. The number of time steps n_A and n_B might be different in different time windows. At the end of each of their time steps, participants output coupling data:

$$c_A^1, c_A^2, \ldots, c_A^{n_A} \in \mathbb{R}^{d_A}, \quad c_B^1, c_B^2, \ldots, c_B^{n_B} \in \mathbb{R}^{d_B},$$

which are high-dimensional vectors encoding all coupling-relevant quantities.

To obtain continuous representations of the coupling data over the entire time window, we apply interpolation to these time-discrete outputs. This yields the functions:

$$c_A : [t_{\text{ini}}, t_{\text{ini}} + \Delta t] \to \mathbb{R}^{d_A}, \quad c_B : [t_{\text{ini}}, t_{\text{ini}} + \Delta t] \to \mathbb{R}^{d_B}.$$

Both interpolants can be evaluated by the respective participants at arbitrary time points, not just the discrete steps, allowing for use at intermediate stages such as those arising in Runge-Kutta methods.

We define the participant operators

$$\mathcal{A} : c_B \mapsto (c_A^1, \ldots, c_A^{n_A}), \quad \mathcal{B} : c_A \mapsto (c_B^1, \ldots, c_B^{n_B}),$$

which represent the discrete-time output of one participant in response to continuous-time input from the other. Additionally, we define interpolation operators (we omit the subscript when not referring to a specific side):

$$\mathcal{I}_{c^0} : (c^1, \ldots, c^n) \mapsto c,$$

which reconstruct a continuous function c given an initial value c^0 and a set of discrete-time samples. Combining the participant and interpolation operators yields the time-continuous operators

$$\widehat{\mathcal{A}} = \mathcal{I}_{c_A^0} \circ \mathcal{A}, \quad \widehat{\mathcal{B}} = \mathcal{I}_{c_B^0} \circ \mathcal{B},$$

which map continuous inputs to continuous outputs.

In the case of serial coupling and if participant B advances first, the coupling procedure can be expressed as a time-continous fixed-point equation:

$$c_A = \widehat{\mathcal{A}} \circ \widehat{\mathcal{B}}\,(c_A).$$

Alternatively, the same relationship can be written in terms of discrete-time outputs:

$$(c_A^1, \ldots, c_A^{n_A}) = \mathcal{A} \circ \mathcal{I}_{c_B^0} \circ \mathcal{B} \circ \mathcal{I}_{c_A^0}(c_A^1, \ldots, c_A^{n_A}).$$

To ensure stability and accuracy, an implicit coupling strategy is employed by iterating on this fixed-point equation. Denoting the iteration index by k, the basic fixed-point iteration becomes:

$$[(c_A^1, \ldots, c_A^{n_A})]^{k+1} = \mathcal{A} \circ \mathcal{I}_{c_B^0} \circ \mathcal{B} \circ \mathcal{I}_{c_A^0}([(c_A^1, \ldots, c_A^{n_A})]^k).$$

In practice, simple fixed-point iteration is often insufficient for convergence [28]. To enhance performance, we introduce an acceleration operator $\mathcal{Q}$, leading to the iteration:

$$[(c_A^1, \ldots, c_A^{n_A})]^{k+1} = \mathcal{Q} \circ \mathcal{A} \circ \mathcal{I}_{c_B^0} \circ \mathcal{B} \circ \mathcal{I}_{c_A^0}([(c_A^1, \ldots, c_A^{n_A})]^k).$$

In its simplest form, $\mathcal{Q}$ may represent an under-relaxation operator, though more sophisticated variants such as quasi-Newton acceleration can also be employed (cf. [29]).

We summarize the serial-implicit coupling algorithm (with B advancing first) as follows. Tildes (e.g., $\tilde{c}_A^1$) denote values prior to acceleration:

1. Initialize from previous window, obtain c_A^0 and c_B^0 and set constant initial guess $[c_A]^0 \equiv c_A^0$, $k = 0$.
2. Advance B (with n_B time steps): $\left[(c_B^1, \ldots, c_B^{n_B})\right]^{k+1} = \mathcal{B}\left([c_A]^k\right)$.
3. Interpolate: $[c_B]^{k+1} = \mathcal{I}_{c_B^0}\left(\left[(c_B^1, \ldots, c_B^{n_B})\right]^{k+1}\right)$.
4. Advance A (with n_A time steps): $\left[(\tilde{c}_A^1, \ldots, \tilde{c}_A^{n_A})\right]^{k+1} = \mathcal{A}\left([c_B]^{k+1}\right)$.
5. If $\left\| \left[\tilde{c}_A^{n_A}\right]^{k+1} - \left[c_A^{n_A}\right]^k \right\|$ small enough: Go to next window.
6. Accelerate: $\left[(c_A^1, \ldots, c_A^{n_A})\right]^{k+1} = \mathcal{Q}\left(\left[(\tilde{c}_A^1, \ldots, \tilde{c}_A^{n_A})\right]^{k+1}\right)$.
7. Interpolate: $[c_A]^{k+1} = \mathcal{I}_{c_A^0}\left(\left[(c_A^1, \ldots, c_A^{n_A})\right]^{k+1}\right)$.
8. Increment k and go back to step 2.

For the interpolation, several options are possible. To keep the user interface simple, we do not make use of time derivatives. As a general and powerful choice, we employ B-spline interpolation [7] of degree p. For implementation simplicity, we restrict the interpolation to the current time window, which imposes $p \leq n$. If insufficient data points are available, the interpolation degree is automatically reduced. A possible workaround is to let the participant produce dense output, if possible, and sub-sample additional points [25]. Both the restriction on the current time window and the avoidance of time derivatives could be relaxed in future work.

3 Implementation

This section presents the main contribution of our work: the implementation of waveform iteration in the coupling library preCICE. We first outline the minimal changes to the user interface in Section 3.1. Section 3.2 describes the internal data layout, and Section 3.3 focuses on the data flow within preCICE, highlighting the interaction with essential building blocks such as communication, data mapping, and fixed-point acceleration. While our examples again focus on two participants, A and B, the implementation supports arbitrary numbers of participants.

3.1 User interface

A central design philosophy of preCICE is to provide a high-level API, where a single call to advance progresses the coupling for a time step. This single call encapsulates data mapping, fixed-point acceleration, and communication, abstracting away the low-level send and receive operations found in other coupling libraries. This enables users to configure the coupling scheme, including who sends data to whom, at runtime.

preCICE is written in C++, but we use the Python bindings in the following to explain the user-facing interface. A minimal example from the perspective of participant A is shown in Listing 1. The `Participant` object is created in Line 3 using a configuration file, given in Listing 2 and visualized in Figure 3. Coupling meshes are defined next; users can define arbitrary numbers of meshes. The data initialization step is optional and described in more detail in Section 3.3. Calling `initialize` exchanges meshes between participants A and B, prepares data mappings, and establishes communication channels.

Each iteration of the while-loop (lines 14 to 28) corresponds to one time step. If implicit coupling is used, the simulation may jump back in time to the start of the time window via checkpointing (lines 15-16 and 27-29). The variable `solver_dt` is the time step size desired by the solver, while `precice_dt` represents the maximum permissible time step within the current window. The actual time step `dt` is chosen as the minimum of these two. See Figure 2 for a sketch.

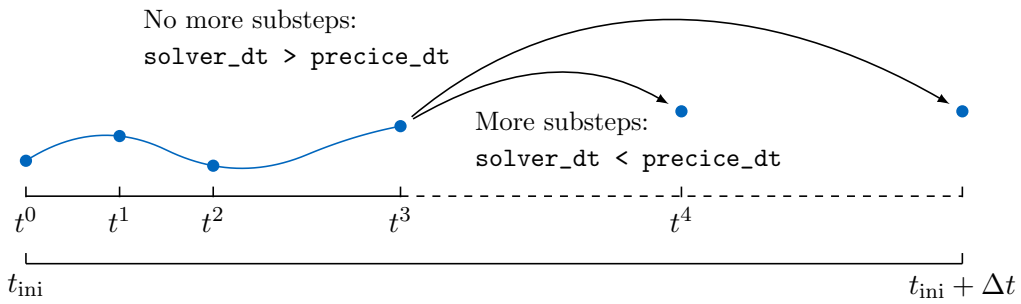


Figure 2: Time stepping within a time window $[t_{\text{ini}}, t_{\text{ini}} + \Delta t]$. `dt = min(solver_dt, precice_dt)` reaches the next time step or the end of the time window.

```python
import precice

participant = precice.Participant("A", "../precice-config.xml", ...)

vertices = extract_coupling_mesh() # shape "number vertices × mesh dimension"
vertex_ids = participant.set_mesh_vertices("Mesh_A", vertices)

if participant.requires_initial_data:
    data_A = initialize_data() # shape "number vertices × data dimension"
    participant.write_data("Mesh_A", "Data_A", vertex_ids, data_A)

participant.initialize()

while participant.is_coupling_ongoing():
    if participant.requires_writing_checkpoint():
        write_checkpoint()

    solver_dt = compute_adaptive_dt()
    precice_dt = participant.get_max_time_step_size()
    dt = min(solver_dt, precice_dt)

    data_B = participant.read_data("Mesh_A", "Data_B", vertex_ids, dt)
    data_A = solve_A(data_B, dt)
    participant.write_data("Mesh_A", "Data_A", vertex_ids, data_A)
    participant.advance(dt)

    if participant.requires_reading_checkpoint():
        read_checkpoint()
```

Listing 1: Minimal preCICE example in Python

Data is read from and written to internal buffers in lines 22 and 24. The function `solve_A` advances the participant A by one time step `dt`. The function `advance` then progresses the coupling in preCICE, including mapping, acceleration, and communication. For more details, we refer to the official documentation[2].

Prior to waveform iteration, up to version 2 of preCICE, constant interpolation was used as multi-rate strategy (depicted in Figure 1a). All data written in a window was overwritten and only the last value retained. `read_data` always returned this final value. The time argument used in Listing 1 line 22 did not yet exist. We gradually ported all major components to support waveform iteration (depicted in Figures 1c and 1d), an effort we concluded in version 3.2. Now, the time argument in `read_data` can be used to specify a relative time within each time step, ranging from `0` (corresponding to the time step start) to `dt` (the time step end). This allows evaluation at intermediate times, for example at `0`, `dt/2`, and `dt` for the stages of the classical Runge-Kutta scheme. Data is no

[2] `https://precice.org/docs.html`

```xml
<precice-configuration>

  <data:scalar name="Data_A" waveform-degree="3" />
  <data:scalar name="Data_B" waveform-degree="3" />

  <mesh name="Mesh_A"  dimensions="3">
    <use-data name="Data_A" />
    <use-data name="Data_B" />
  </mesh>

  <mesh name="Mesh_B"  dimensions="3">
    <use-data name="Data_A" />
    <use-data name="Data_B" />
  </mesh>

  <participant name="A">
    <provide-mesh name="Mesh_A" />
    <write-data name="Data_A" mesh="Mesh_A" />
    <read-data name="Data_B" mesh="Mesh_A" />
  </participant>

  <participant name="B">
    <receive-mesh name="Mesh_A" from="A" />
    <provide-mesh name="Mesh_B" />
    <mapping:rbf direction="write" from="Mesh_B" to="Mesh_A"
    ↪  constraint="conservative" />
    <mapping:rbf direction="read" from="Mesh_A" to="Mesh_B"
    ↪  constraint="consistent" />
    <write-data name="Data_B" mesh="Mesh_B" />
    <read-data name="Data_A" mesh="Mesh_B" />
  </participant>

  <m2n:sockets acceptor="A" connector="B" />

  <coupling-scheme:serial-implicit>
    <participants first="B" second="A" />
    <max-time-windows value="2" />
    <time-window-size value="1.0" />
    <max-iterations value="2" />
    <exchange data="Data_A" mesh="Mesh_A" from="A" to="B" substeps="true"
    ↪  initialize="true" />
    <exchange data="Data_B" mesh="Mesh_A" from="B" to="A" substeps="true" />
    <relative-convergence-measure limit="1e-4" data="Data_A" mesh="Mesh_A" />
    <acceleration:IQN-ILS>
      <data name="Data_A" mesh="Mesh_A" />
    </acceleration:IQN-ILS>
  </coupling-scheme:serial-implicit>
</precice-configuration>
```

Listing 2: preCICE configuration example. The mesh of A, `"Mesh_A"`, is sent from participant A to participant B and data mappings in both directions are computed on participant B. The B-spline (waveform) degree p is set to 3 for both data fields. The exchange of substep data, necessary for waveform iteration, is switched on (default for implicit coupling).

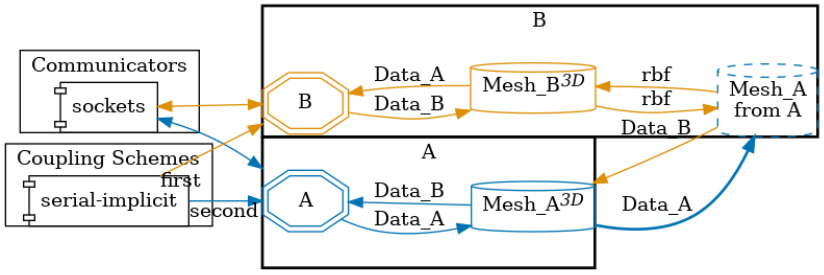


Figure 3: Visualization of preCICE configuration example of Listing 2 using the preCICE CLI

longer overwritten but stored and used to construct interpolants. The `waveform-degree` attribute specifies the degree of interpolation—in Listing 2, cubic B-splines are used—subject to the limitations discussed in Section 2. If the attribute `substeps="off"` is added to an `exchange` tag, then substep handling and associated communication are turned off, corresponding to Figure 1b. While this mode might also have an efficiency sweet spot, it is mainly retained for comparison and may be removed in the future.

3.2 Data layout

The core challenge in implementing waveform iteration is that adaptive time stepping introduces variable numbers of data entries within each time window. In an earlier prototype [29], we used fixed time step sizes.

Figure 4 illustrates the relationship between the core data structures. The class `Mesh` (such as `"Mesh_A"`) can hold multiple `Data` fields (such as `"Data_A"` or `"Data_B"`), as seen in the configuration. Each data field uses a `Storage` backend that maintains a time-ordered list of `Stample` objects—short for time-stamped samples.

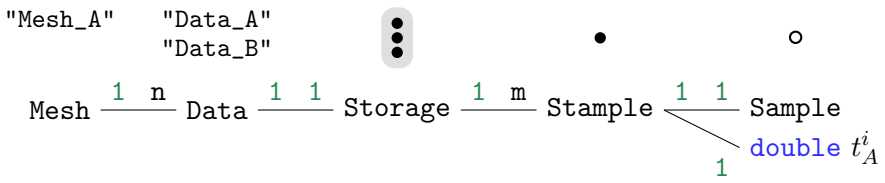


Figure 4: Core data structures in preCICE and their relationships. The symbols for `Storage`, `Stample`, and `Sample` are used throughout the paper.

For clarity, we only consider scalar data in the following. In the Python API, data is passed as NumPy arrays; in the C++ backend, this becomes a generic span and is ultimately wrapped into an `Eigen::VectorXd` [15], forming the core of a `Sample`. Each Stample then holds a `Sample` and an associated time stamp. The time stamp is not known when `write_data` is called, but only during `advance`. Therefore, the implementation splits writing into two steps: first, we collect multiple `Sample` objects, then

during `advance`, we turn each into a `Stample`. This design simplifies the user interface by avoiding the need to manually specify time stamps. The design specifically assumes that written data corresponds to the end of each time step, which matches most time integration schemes. For others, users may need to perform interpolation themselves. Absolute time stamps are used internally, which simplifies synchronization across participants and potentially multiple coupling schemes with different time windows each.

The `Storage` collects multiple `Stample` objects and provides methods to add and retrieve data, see Figure 5. When `advance` is called, the new `Sample` is converted and stored (using `setSampleAtTime`). If a time window converges, all data before the window end is trimmed (using `trimBefore`), with the final `Stample` retained as new starting data. If the window does not converge, all data after the window start is discarded (using `trimAfter`), preserving only the initial `Stample`. When calling `read_data`, the relative time is translated to an absolute one and used to sample from the `Storage`. To this end, the `Storage` holds a B-spline interpolation class, adapted from Eigen's implementation for cache efficiency[3].

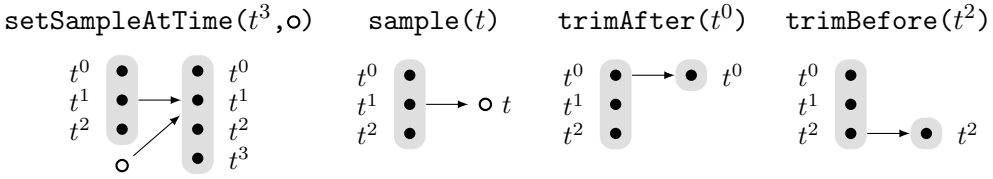


Figure 5: Methods to add and retrieve data from the `Storage` class

3.3 Data flow

While waveform iteration is well known in co-simulation, it is not widely used in black-box coupling of PDE-based solvers. The additional complexity arises because PDE simulations require the integration of multiple building blocks: data mapping, parallel communication, and fixed-point acceleration. We now describe how preCICE manages their interaction under waveform iteration.

A fundamental design decision is to determine which participant creates the waveform—the writing or the reading participant. This choice affects all components. For instance, if participant B writes `"Data_B"` and participant A reads it, then in the first option we communicate the entire Storage to A, who then constructs the waveform locally. In the second option, B constructs the waveform and sends only the evaluated samples to A. The latter would require A to communicate sampling times to B, either in `read_data` or in the previous `advance`. This either contradicts the preCICE design of only communicating in `advance` or impairs usability with adaptive time stepping. Therefore, we choose the first option: waveform construction occurs on the reader side as depicted in Figure 6. This does increase communication volume when B uses more time steps

[3] `https://github.com/precice/precice/pull/1765#pullrequestreview-1632933653`

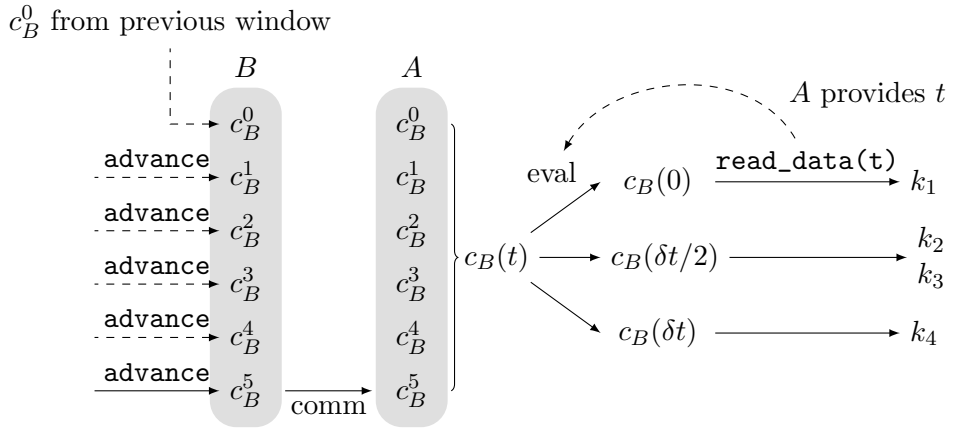


Figure 6: Fundamental design decision of the implementation. Communicatin of storage from B to A at the end of a time window (here, after 5 time steps) and construction of waveform c_B on reading participant A.

than A requires, but the opposite direction might cause more performance issues. For mitigation, we want to study data compression in future work.

For data mapping, preCICE separates the typically costly initialization based on the meshes (in initialize) and the actual data-dependend mapping in advance. The latter is performed at the window end. Similarly as in previous preCICE versions [5], the write mapping (e.g., `"Data_B"` from `"Mesh_B"` to `"Mesh_A"` on B) is applied before communication; the read mapping (e.g., `"Data_A"` from `"Mesh_A"` to `"Mesh_B"` on B) is applied afterward. With waveform iteration now, each `Sample` is individually mapped, which does, however, not increase the initialization cost. The sample at the window start is excluded, as it was already mapped before—either in the previous time window or the previous iteration.

Communication in preCICE builds on repartitioning the coupling meshes to establish parallel communication channels [38]. This step as well as the actual communication is independent of waveform iteration, only more data is now communicated. Storage objects are serialized; due to dynamic size, the message size must be communicated beforehand. For simplicity, we also transmit the initial sample in each window, though this could be optimized.

Coupling schemes define the order in which coupled participants advance. Both serial (one after the other) and parallel (both simultaneously) coupling are supported. The extension to waveform iteration is conceptually straightforward, with full storages now communicated. A slight complication is initialization. If `initialize="true"` (as in Listing 2 for `"Data_A"`), initial values are exchanged. In parallel coupling, both sides can initialize and exchange data immediately. In serial coupling, only the second participant (A) previously needed the final value from the first (B) to start. Now, the initial value is also needed to construct a first waveform, so B must write it, and A receives it normally as part of the first `Storage` communication in the first advance. Only if `substeps="off"`,

then an extra communication step is needed. We omit technical details here, but refer to Rodenberg [25].

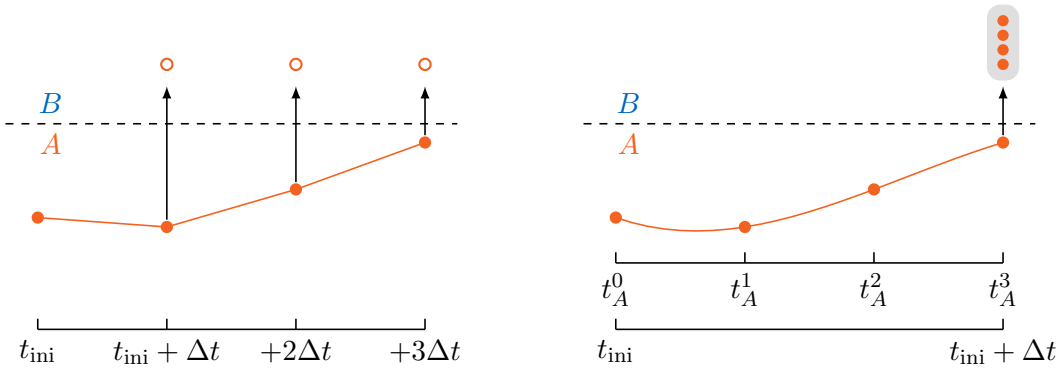
Acceleration is the most complex component to adapt. preCICE supports several variants, from simple underrelaxation to advanced quasi-Newton methods. They all rely on tracking histories of iterates and residuals, which becomes problematic under adaptive time stepping, as time grids differ. In our earlier prototype [29], we therefore considered fixed time grids only. Two main solutions are possible: always interpolating all history data to the current time grid or using a fixed auxiliary grid. The latter is far less computationally expensive. Still, the additional interpolation leads to additional computational cost and introduces an additional numerical error, which can be controlled by refining the auxiliary grid. This approach was implemented and analyzed independently by Kotarsky and Birken [18], also showing the generality and modularity of our implementation. Their evaluation compared different time grids and concluded that reusing the grid from the first iterate suffices. This also supports the fixed grid case automatically. They further demonstrate the efficiency benefits of adaptive stepping, which is why we do not present such results here. We refer the reader to their work for further details.

To conclude the section, we compare the computational cost of two scenarios with identical time step sizes, which therefore require no interpolation: (i) one where the time window equals the time step size ($\Delta t = \delta t$, Figure 7a), and (ii) one with subcycling ($\Delta t = n \cdot \delta t$, Figure 7b). Assuming both variants converge in k iterations, each requires $k \cdot n$ mapping operations. Communication costs are $k \cdot n$ in the first case and $k \cdot (n+1)$ in the second, although this overhead could be reduced. Subcycling, however, decreases the number of synchronization points, which is particularly beneficial when individual time steps have differing runtimes. How this reduced synchronization influences the number of iterations is investigated in Section 4. For acceleration, the cost depends on the chosen method. All acceleration techniques scale linearly with the number of degrees of freedom (e.g., [30]). In the case of quasi-Newton acceleration combined with waveform iteration, we distinguish between a full variant, which uses all substep data to compute the update, and a reduced variant, which relies only on data at the end of each time window [18, 29]. While the full variant is expected to incur comparable costs, the reduced variant is significantly cheaper.

4 Results

We analyze three academic test cases, all taken from the preCICE tutorials[4]. Although simple, these cases demonstrate the significant advantages of waveform iteration over earlier multi-rate approaches implemented in preCICE. Detailed setups and results are provided in a separate data publication [26].

[4] `https://precice.org/tutorials.html`



(a) No subcycling (b) Subcycling

Figure 7: Illustration of data used in communication and acceleration with A using identical time step and window size Δt (Figure 7a) and A using time steps samller than window size Δt (Figure 7b). In Figure 7a, A sends, maps, and accelerates a single `Sample` at the end of each time window. In Figure 7b, A performs 3 time steps and sends, maps, and accelerates the resulting `Storage`.

4.1 Partitioned oscillator

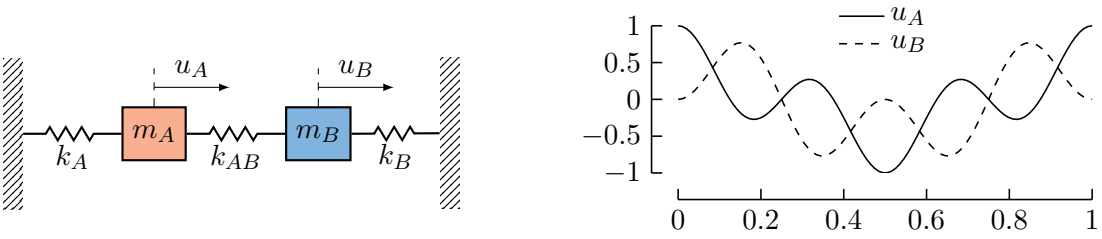
Schüller et al. [32] introduce a simple partitioned oscillator model, which we use to illustrate the core ideas of waveform iteration, validate our implementation, and investigate fundamental behavior. The system is modeled by two coupled ODEs describing the dynamics of a dimensionless two-mass-spring setup, as shown in Figure 8a:

$$m_A \ddot{u}_A + (k_A + k_{AB})\, u_A - k_{AB} u_B = 0 \,, \tag{1}$$
$$m_B \ddot{u}_B - k_{AB} u_A + (k_B + k_{AB})\, u_B = 0 \,. \tag{2}$$

We use $k_A = k_B = 4\pi^2$, $k_{AB} = 16\pi^2$, $m_A = m_B = 1$, and the initial conditions

$$u_A(t=0) = 1 \,, \qquad \dot{u}_A(t=0) = 0 \,, \qquad u_B(t=0) = 0 \,, \qquad \dot{u}_B(t=0) = 0 \,.$$

Figure 8b shows the analytical solution for this specific case (cf. [32]).



(a) Schematic drawing (b) Displacement u_A and u_B over time t

Figure 8: Oscillating two-mass-spring-system (left) and its analytical solution (right)

To study coupling strategies, we artificially partition the system following an overlapping Schwarz approach, illustrated in Figure 9. This means we solve Equations (1)

and (2) independently using separate ODE solvers, each relying on approximations for u_B and u_A, respectively. We examine the interval $t \in [0, 1]$, covering one full oscillation period, for various time window and time step sizes. For both displacements u_A and u_B, we compute the maximum error across all time steps, denoted by e_A and e_B.

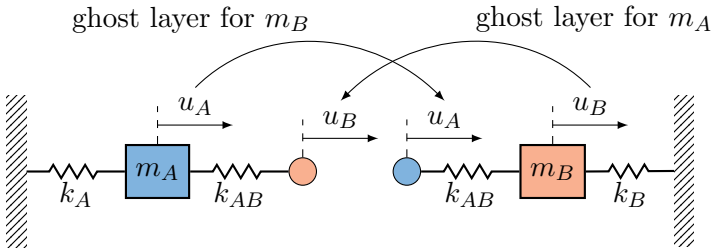


Figure 9: Partitioning of the two-mass-spring system

In a first experiment, we compare the four multi-rate coupling approaches introduced in Figure 1. As in that figure, participant A employs the RK4 method, while participant B uses the generalized Alpha (GA) method. A and B use 5 and 100 constant time steps per time window, respectively; this extreme ratio is chosen to clearly highlight the impact of the method. Figure 10 shows the resulting errors for decreasing time window sizes. In all oscillator experiments, we use serial-implicit coupling and iterate until the relative coupling error (defined as the maximum change in u_A and u_B between successive iterations) drops below 10^{-10}. No acceleration method is needed to stabilize the coupling.

As expected, subcycling with constant interpolation yields first-order convergence, and with linear interpolation, second-order convergence. Waveform iteration with piecewise-linear interpolation also achieves second-order convergence, but with considerably smaller errors. When using third-degree B-spline interpolation, waveform iteration exhibits fourth-order convergence for large time windows, where RK4 in A dominates, and second-order convergence for smaller windows, where GA in B dominates.

This first experiment already highlights that waveform iteration is a powerful framework for coupling different time integration methods. However, it also shows that balancing the associated errors becomes a non-trivial task. In a second experiment, we use waveform iteration with third-degree B-spline interpolation over a wide range of constant time step and time window sizes to explore the optimal ratio of time step sizes. Specifically, we vary Δt from 1/5 to 1/800, keeping A's time step fixed as $\delta t_A = \Delta t/4$, while B's time step δt_B is varied from δt_A to $\delta t_A/512$, in order to account for B's lower order (GA) compared to A (RK4). Figure 11 shows a contour plot of the convergence behavior and empirically identifies a corridor of optimal time step ratios (gray squares). As expected, halving δt_A requires dividing δt_B by four to maintain optimal accuracy.

In a final experiment on the partitioned oscillator, we fix the time step sizes close to the optimal ratio at $\delta_A = 0.005$ and $\delta_B = 0.0002$ (approximately corresponding to the middle gray square in Figure 11), and increase the time window size from $\Delta t = 0.005$ to $\Delta t = 0.2$, similarly to the thought experiment at the end of Section 3. Table 1

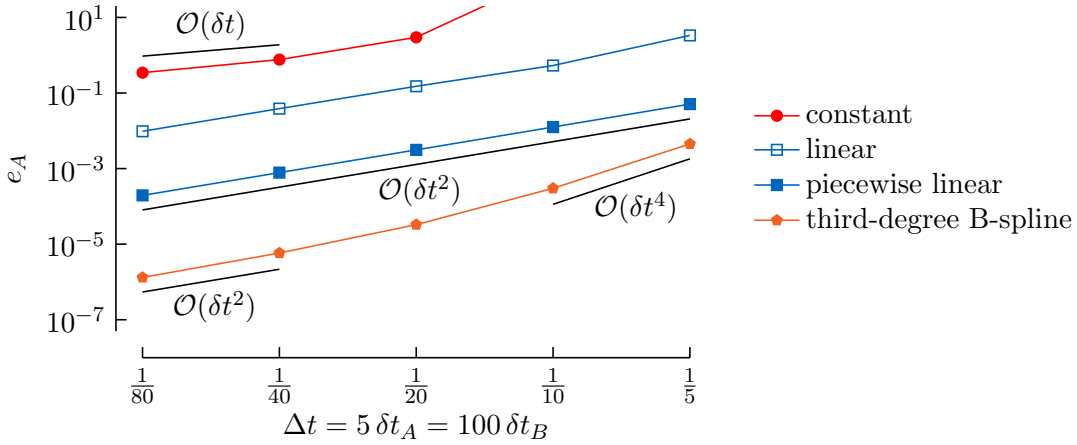


Figure 10: Illustrative comparison of the four multi-rate approaches introduced in Figure 1 for the partitioned oscillator. A uses RK4 with $\delta t_A = \Delta t/5$, B uses GA with $\delta t_B = \Delta t/100$.

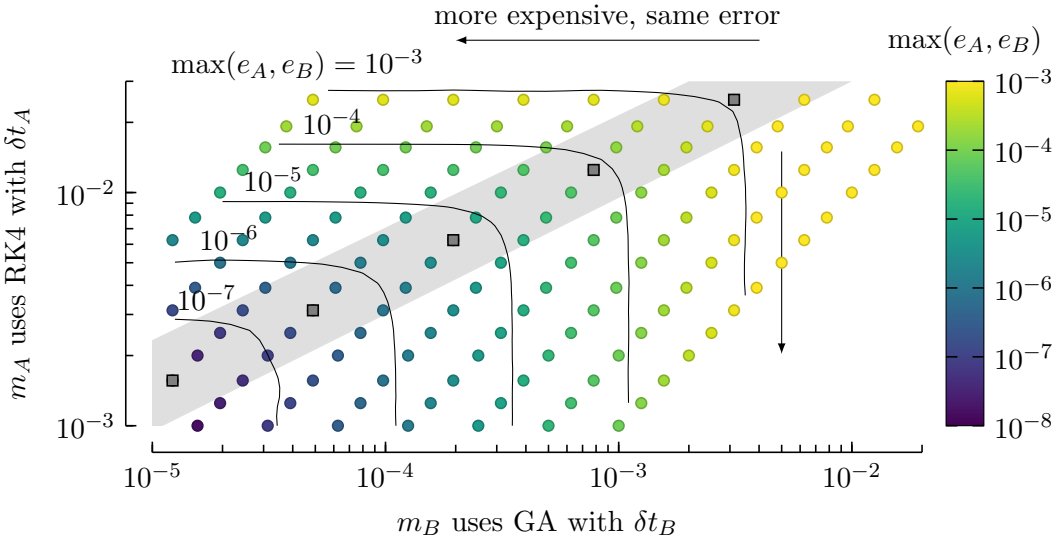


Figure 11: Contour plot for varying both time step sizes δt_A and δt_B of the partitioned oscillator. Contour lines help identify regions of identical error. If one of the ODE integrators dominates the error, the theoretical convergence orders–two for GA and four for RK–can be inferred from the spacing between contour lines along the respective axis. The optimal ratio of time step sizes, where reducing either time step size significantly reduces the error, is highlighted with a gray corridor and gray squares.

presents the resulting errors and the average number of coupling iterations per time window. While the error remains stable, the number of iterations increases gradually. For instance, reducing the synchronization frequency by a factor of ten (from $\Delta t = 0.01$

Δt	e_A	e_B	Avg. its.
0.005	7.80E-04	7.46E-04	3.06
0.01	5.78E-06	4.93E-06	3.08
0.02	3.94E-06	3.55E-06	4.00
0.05	4.27E-06	3.80E-06	4.80
0.1	4.39E-06	3.89E-06	5.20
0.2	4.45E-06	3.93E-06	7.00

Table 1: Error and average number of iterations for fixed time step sizes $\delta_A = 0.005$, $\delta_S = 0.0002$ and increasing time window sizes. $\Delta t = 0.005$ marks an outlier as A does not have enough samples per time window to build a third-degree B-spline.

to $\Delta t = 0.1$) results in a 69% increase in the number of iterations. Repeating the same experiment with quasi-Newton acceleration leads to slightly lower iteration counts, but similar scaling behavior (not shown here, but included in the data publication [26]).

4.2 Partitioned heat equation

Next, we evaluate the waveform iteration capabilities of preCICE on a first PDE example, the heat equation,

$$\frac{\partial u}{\partial t} = \Delta u + f \ ,$$

on the rectangular domain $\Omega = [0, 2] \times [0, 1]$ and over the time intervall $[0, 1]$. We use the manufactured solution [19, Section 3.1]

$$u_{\text{exact}}(x, y, t) = 1 + (\sin(t) + \cos(t))x^2 + 3y^2 + 1.2t \ ,$$

to define initial conditions, Dirichlet boundary conditions on $\partial\Omega$, and the right-hand side f. For the spatial discretization, we use second-order finite elements, which allow exact representation of the polynomial terms x^2 and y^2 in u_{exact}, including their fluxes. This lets the spatial error vanish and allows us to focus on the time discretization error. We investigate several Runge-Kutta time-stepping schemes following the approach used in the Firedrake module Irksome [10].

To define a coupled problem, we artificially split the domain into two non-overlapping subdomains, as illustrated in Figure 12. We apply a Dirichlet-Neumann coupling: Participant A uses the interface temperature u_C as a Dirichlet boundary condition and computes the heat flux q on the interface, which is then used by participant B as a Neumann boundary condition. B, in turn, returns the interface temperature. The coupling is performed using serial-implicit coupling with reduced quasi-Newton acceleration [18, 29], iterating until the relative residuals of both temperature and flux at the interface fall below 10^{-12}. Since the meshes match at the coupling interface, no mapping error occurs. To assess the accuracy of different time-stepping methods, we compute the relative L^2-error over the subdomain of A and report the maximum value across all time steps,

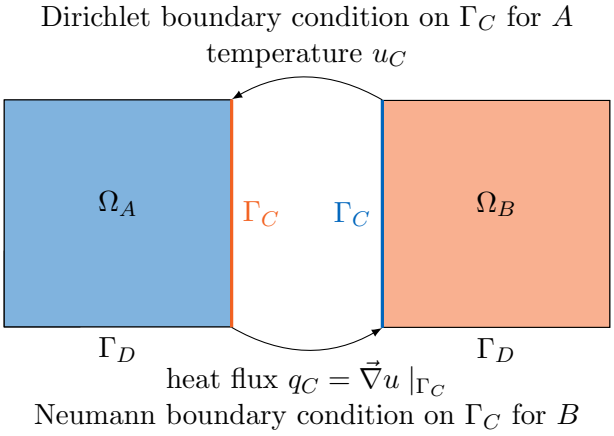


Figure 12: Setup of the partitioned heat equation problem. A acts as Dirichlet participant and B as Neumann participant in a non-overlapping Dirichlet-Neumann coupling

denoted by e_A. Unlike the oscillator example, this experiment uses a more realistic software setup: We solve both subdomains using FEniCS [2] and use the FEniCS-preCICE adapter [27] to call preCICE on both sides. The implementation of the time-stepping has been carried out in Vinnitchenko [39].[5]

Figure 13 presents a convergence study in which we reduce both the time window and time step sizes while keeping the ratio $\Delta t/\delta t = 5$ constant, and observe the resulting error e_A. All configurations are symmetric: Both participants use the same time stepping methods and identical step sizes. Waveform iteration is performed with B-spline interpolation of degree $p = 2, 3$, and 5.

The implicit Euler (IE) method shows first-order convergence, with nearly identical error levels for all p. The two-stage Gauss-Legendre method (GL(2)) achieves second-order convergence for $p = 2$. For $p = 3, 5$, the results are almost identical and we observe a convergence order between third order and the expected fourth-order of the monolithic experiment. The three-stage Gauss-Legendre method (GL(3)) achieves only fourth-order convergence for $p = 3$. We observe a clear improvement when increasing the degree of the interpolation to $p = 5$ and almost reach the expected fifth-order convergence.

Finally, we verify the correct implementation of the quasi-Newton acceleration in the waveform iteration framework by comparing iteration counts to those of the original prototype implementation [29]. The results are consistent and successfully reproduced (not shown here; see Rodenberg [25] for details).

[5] Internally, this approach does not use the interface temperature u_C but its time derivative $\dot{u}_C$ as a Dirichlet boundary condition for the stages of the time stepping schemes [10]. Since preCICE does not yet offer an API to obtain the time derivatives of waveforms (see `https://github.com/precice/precice/issues/1908`), the Dirichlet participant has to perform an additional reconstruction step to compute the time derivatives.

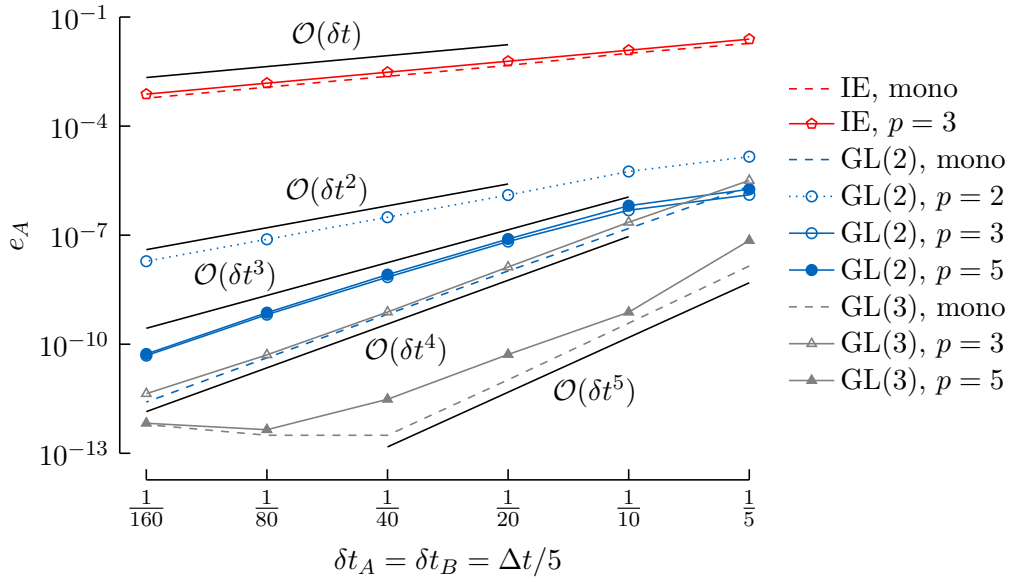


Figure 13: Time stepping error for the partitioned heat conduction problem using different time stepping methods. Waveform iteration with B-spline interpolation of degree $p = 2, 3$, and 5 is used. The subcycling ratio $\Delta t/\delta t = 5$ is fixed. Monolithic experiments are shown as reference. Similar results can be obtained for the LobattoIIIC scheme with three stages (not shown here, included in the data publication [26]).

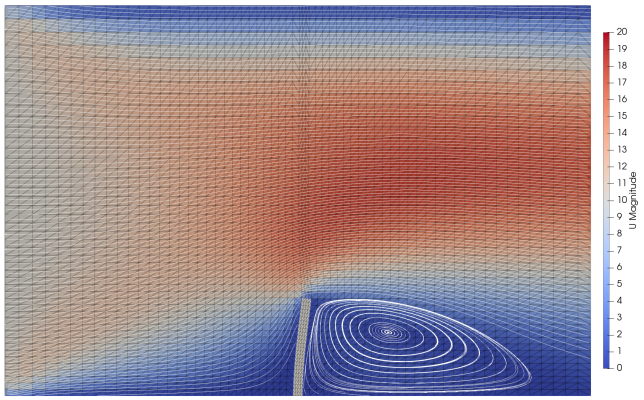


Figure 14: Perpendicular flap with streamlines and discretization mesh of the fluid field and the elastic beam.

4.3 Perpendicular flap

As a final test case, we consider a classical fluid-structure interaction (FSI) scenario: an elastic flap mounted perpendicularly in a cross-flow, see Figure 14. The fluid domain is governed by the incompressible Navier-Stokes equations, while the solid domain follows a linear elasticity model. The geometry, boundary conditions, and material parameters follow the setup from the preCICE tutorials[6].

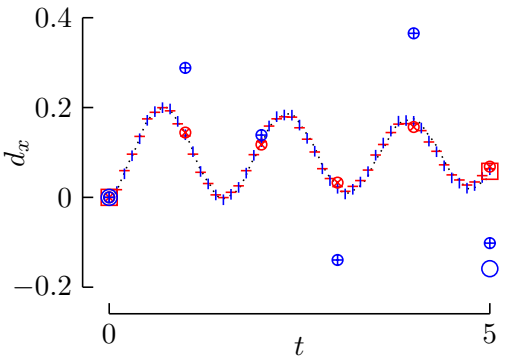
The solid participant uses finite element discretization via FEniCS [2] and is coupled through the FEniCS-preCICE adapter [27]. The fluid participant uses finite volume discretization through OpenFOAM [40], coupled using the OpenFOAM-preCICE adapter [4]. We again use a classical Dirichlet-Neumann coupling, where the fluid participants uses a Dirichlet boundary condition at the coupling interface and the solid participant a Neumann boundary condition. We also use again a serial-implicit coupling with reduced quasi-Newton acceleration [18, 29], iterating until the relative residuals of both displacements and forces at the interface fall below $5.0 \cdot 10^{-3}$. We use non-matching meshes and a partition-of-unity radial-basis-function interpolation [31] for data mapping.

For time integration, FEniCS employs the generalized-alpha method. OpenFOAM also applies a second-order time integration scheme, which, however, effectively degrades to first-order accuracy in the presence of FSI coupling [37]. Achieving true second-order accuracy in a CFD solver under mesh motion and coupling constraints is a non-trivial task and cannot be expected from standard off-the-shelf solvers [11, 29]. The coupled system exhibits instability for fluid time step sizes $\delta t_F > 0.01$. This relatively strict time-step limitation on the CFD side, combined with the second-order time integration available on the solid side, makes this setup an interesting candidate for evaluating non-matching time step sizes.

Still, in a first experiment, identical time step sizes are used for both participants,

[6] `https://precice.org/tutorials-perpendicular-flap.html`

with increasing time window sizes. We compare waveform iteration with third-degree B-splines (cf. Figure 1d) against the much cheaper single-value linear interpolation (Figure 1b), which is often used for FSI problems (e.g., [6]). Figure 15 shows the oscillating tip displacement of the flap over the time interval $[0, 5]$ and the average number of coupling iterations for various configurations.

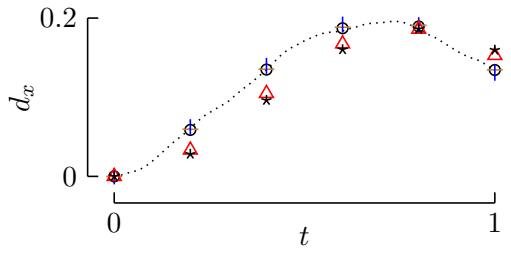


Δt	linear (Fig. 1b)	WI (Fig. 1d)
0.01	2.01	2.01
0.03	2.04	2.40
0.05	2.07	2.54
0.1	2.62	2.78
0.25	3.75	3.90
0.5	4.80	5.20
1.0	9.60	6.80
2.5	24.00	12.50
5.0	8.00	19.00

Figure 15: Tip displacement (left) and average coupling iterations (right) for the perpendicular flap problem for identical time step sizes of $\delta t = 0.01$ and increasing window sizes. Single-value linear interpolation (linear) is compared to waveform iteration with third-degree B-splines (WI).

Waveform iteration with third-degree B-splines resolve the tip displacement very well even if we only use a single time window for the complete time interval. Single-value linear interpolation shows good results up to moderate subcycling ($\Delta t = 10\,\delta t$), but not beyond. The average number of iterations grows slowly, but not drastically with an increasing window size. For instance, synchronizing only every 10 time steps costs an increase of 38% in terms of iterations showing the potential. For larger time domains, we even observed a slower increase (not shown).

In a second experiment, we fix the fluid time step size at the critical value $\delta t_F = 0.01$ and vary the solid time step size δt_S. The time window size is kept constant at $\Delta t = 0.2$. As a reference solution, we use a significantly finer simulation with $\delta t_F = \delta t_S = 0.001$ and $\Delta t = 0.02$. Figure 16 shows the tip displacement of the flap over a shorter time interval $[0, 1]$, together with the average number of coupling iterations. We observe that the solid time step size can be increased up to $\delta t_S = 0.02$ without a visible loss in accuracy. For larger step sizes, however, the displacement begins to deviate from the reference solution. Notably, the average number of coupling iterations remains nearly constant across all tested configurations. Overall, this experiment demonstrates the potential of waveform iteration to independently choose and optimize the time step sizes of the coupled participants and optimize their ratio, improving both flexibility and efficiency in multiphysics simulations.



	Δt	δt_F	δt_S	Avg. its.
·····	0.02	0.001	0.001	5.34
○	0.2	0.01	0.005	5.00
—	0.2	0.01	0.01	5.20
|	0.2	0.01	0.02	5.60
△	0.2	0.01	0.04	5.80
⋆	0.2	0.01	0.1	5.60

Figure 16: Tip displacement (left) and average coupling iterations (right) for the perpendicular flap problem for fixed fluid time step size $\delta t_F = 0.01$, fixed window size $\Delta t = 0.2$, and varying solid time step size δt_S.

5 Conclusions

We have integrated waveform iteration into the coupling library preCICE and analyzed its efficiency. To the best of our knowledge, this is the first implementation of waveform iteration in a generic PDE coupling software. Achieving this required carefully resolving the interplay with key PDE building blocks, including data mapping, parallel communication, and quasi-Newton acceleration. Despite the technical complexity, only minimal changes are required in existing preCICE adapters, while users gain access to a far more powerful numerical framework.

The new implementation enables coupled participants to use different, even adaptive, time step sizes. This flexibility allows users to balance errors between solvers, though determining optimal step size ratios remains non-trivial. With a sufficiently high interpolation degree, waveform iteration can deliver overall higher-order convergence in time. Importantly, benefits arise even when the individual solvers are only first-order accurate, since users retain the freedom to optimize step sizes independently.

Our experiments also show the potential for drastically fewer synchronization points, an advantage of particular relevance in high-performance computing environments. Quasi-Newton acceleration remains effective, with only a slight increase in iteration counts, even at extreme multirate ratios of up to 100:1 or 1000:1.

Future work will focus on further extending these ratios, for example through data compression strategies, as well as broadening the implementation to support extrapolation, explicit coupling, and interpolation across multiple time windows. We will support the preCICE community in validating the approach on real-world applications.

Acknowledgments

The authors gratefully acknowledge the support by the Stuttgart Center for Simulation Science (SimTech). The authors moreover acknowledge the assistance of AI-tools (ChatGPT-4) for editorial suggestions conformal with the editorial policy of SIAM.

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Re-

search Foundation) under Germany's Excellence Strategy, EXC 2075—390740016.

References

[1] D. Abele et al. "Coupling of the Ice-sheet and Sea-level System Model (version 4.24) with hydrology model CUAS-MPI (version 0.1) using the preCICE coupling library". In: *EGUsphere* (2025), pp. 1–25. DOI: 10.5194/egusphere-2025-3345.

[2] Martin S Alnæs et al. "The FEniCS Project Version 1.5". In: (2015).

[3] M. Breuer et al. "Fluid–structure interaction using a partitioned semi-implicit predictor–corrector coupling scheme for the application of large-eddy simulation". In: *Journal of Fluids and Structures* 29 (2012), pp. 107–130. ISSN: 0889-9746. DOI: https://doi.org/10.1016/j.jfluidstructs.2011.09.003.

[4] Gerasimos Chourdakis, David Schneider, and Benjamin Uekermann. "OpenFOAM-preCICE: Coupling OpenFOAM with External Solvers for Multi-Physics Simulations". In: *OpenFOAM® Journal* 3 (Feb. 2023), pp. 1–25. DOI: 10.51560/ofj.v3.88.

[5] Gerasimos Chourdakis et al. "preCICE v2: A sustainable and user-friendly coupling library". In: *Open Research Europe* 2 (Apr. 2022), p. 51. DOI: 10.12688/openreseurope.14445.1.

[6] Laurent De Moerloose et al. "Analysis of several subcycling schemes in partitioned simulations of a strongly coupled fluid-structure interaction". In: *International Journal for Numerical Methods in Fluids* 89.6 (2019), pp. 181–195. DOI: 10.1002/fld.4688.

[7] Paul Dierckx. *Curve and Surface Fitting with Splines*. Oxford University Press, 1993. DOI: 10.1093/oso/9780198534419.001.0001.

[8] Florent Duchaine et al. "Analysis of high performance conjugate heat transfer with the OpenPALM coupler". In: *Computational Science & Discovery* 8.1 (2015), p. 015003. DOI: 10.1088/1749-4699/8/1/015003.

[9] Charbel Farhat and Michel Lesoinne. "Higher-order staggered and subiteration free algorithms for coupled dynamic aeroelasticity problems". en. In: *36th AIAA Aerospace Sciences Meeting and Exhibit*. Reno,NV,U.S.A.: American Institute of Aeronautics and Astronautics, 1998. DOI: 10.2514/6.1998-516.

[10] Patrick E. Farrell, Robert C. Kirby, and Jorge Marchena-Menéndez. "Irksome: Automating Runge-Kutta Time-Stepping for Finite Element Methods". In: *ACM Transactions on Mathematical Software* 47.4 (Sept. 2021). ISSN: 0098-3500. DOI: 10.1145/3466168.

[11] Niklas Fehn et al. "High-order arbitrary Lagrangian–Eulerian discontinuous Galerkin methods for the incompressible Navier–Stokes equations". In: *Journal of Computational Physics* 430 (2021), p. 110040. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2020.110040.

[12] Martin J. Gander. "Waveform Relaxation". In: *Encyclopedia of Applied and Computational Mathematics*. Ed. by Björn Engquist. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 1549–1552. ISBN: 978-3-540-70529-1. DOI: 10.1007/978-3-540-70529-1.

[13] Idoia Cortes Garcia et al. "Waveform Relaxation for Low Frequency Coupled Field/Circuit Differential-Algebraic Models of Index 2". In: *Scientific Computing in Electrical Engineering*. Ed. by Martijn van Beurden, Neil Budko, and Wil Schilders. Cham: Springer International Publishing, 2021, pp. 201–209. ISBN: 978-3-030-84238-3.

[14] Markus Gross et al. "Physics–Dynamics Coupling in Weather, Climate, and Earth System Models: Challenges and Recent Progress". In: *Monthly Weather Review* 146.11 (2018), pp. 3505–3544. DOI: 10.1175/MWR-D-17-0345.1.

[15] Gaël Guennebaud, Benoît Jacob, et al. *Eigen v3*. 2010. URL: https://libeigen.gitlab.io.

[16] M. Hanke et al. "YAC 1.2.0: new aspects for coupling software in Earth system modelling". In: *Geoscientific Model Development* 9.8 (2016), pp. 2755–2769. DOI: 10.5194/gmd-9-2755-2016.

[17] Jurgen Kersschot et al. "Simulation of the vibro-acoustic interaction in a flexible flow duct using a partitioned approach in the time domain". In: *AIAA AVIATION 2021 Forum*. DOI: 10.2514/6.2021-2148.

[18] Niklas Kotarsky and Philipp Birken. "A Time-Adaptive Multirate Quasi-Newton Waveform Iteration for Coupled Problems". In: *International Journal for Numerical Methods in Engineering* 126.12 (2025), e70063. DOI: 10.1002/nme.70063.

[19] Hans Petter Langtangen and Anders Logg. *Solving PDEs in Python - The FEniCS Tutorial I*. Springer Cham, 2017, pp. XI, 146. ISBN: 978-3-319-52462-7. DOI: 10.1007/978-3-319-52462-7.

[20] Hermann G. Matthies and Jan Steindorf. "Partitioned strong coupling algorithms for fluid-structure interaction". In: *Computers & Structures* 81.8 (2003), pp. 805–812. ISSN: 0045-7949. DOI: 10.1016/S0045-7949(02)00409-1.

[21] Miriam Mehl et al. "Parallel coupling numerics for partitioned fluid–structure interaction simulations". In: *Computers & Mathematics with Applications* 71.4 (2016), pp. 869–891. ISSN: 0898-1221. DOI: 10.1016/j.camwa.2015.12.025.

[22] Azahar Monge and Philipp Birken. "A time-adaptive Dirichlet-Neumann waveform relaxation method for coupled heterogeneous heat equations". In: *PAMM* 19.1 (2019), e201900206. DOI: 10.1002/pamm.201900206.

[23] Alessandro Munafò et al. "A multi-physics modeling framework for inductively coupled plasma wind tunnels". In: *AIAA SCITECH 2022 Forum*. DOI: 10.2514/6.2022-1011.

[24] Monica Nonino et al. “Projection Based Semi-Implicit Partitioned Reduced Basis Method for Fluid-Structure Interaction Problems”. In: *Journal of Scientific Computing* 94.1 (2022), p. 4. DOI: 10.1007/s10915-022-02049-6.

[25] Benjamin Rodenberg. “Flexible and robust time stepping for partitioned multiphysics”. Dissertation. Technical University of Munich, 2025. URL: https://nbn-resolving.org/urn:nbn:de:bvb:91-diss-20250424-1763172-0-4.

[26] Benjamin Rodenberg and Benjamin Uekermann. *Replication Data for: A Waveform Iteration Implementation for Black-box Multi-rate Higher- order Coupling*. Version v202510.0. Oct. 2025. DOI: 10.5281/zenodo.17352084.

[27] Benjamin Rodenberg et al. “FEniCS–preCICE: Coupling FEniCS to other simulation software”. In: *SoftwareX* 16 (2021), p. 100807. ISSN: 2352-7110. DOI: 10.1016/j.softx.2021.100807.

[28] Benjamin Rüth et al. “Time stepping algorithms for partitioned multi-scale multiphysics in preCICE”. en. In: *ECCM 6 / ECFD 7*. June 2018.

[29] Benjamin Rüth et al. “Quasi-Newton waveform iteration for partitioned surface-coupled multiphysics applications”. In: *International Journal for Numerical Methods in Engineering* 122.19 (2021), pp. 5236–5257. DOI: 10.1002/nme.6443.

[30] Klaudius Scheufele and Miriam Mehl. “Robust Multisecant Quasi-Newton Variants for Parallel Fluid-Structure Simulations—and Other Multiphysics Applications”. In: *SIAM Journal on Scientific Computing* 39.5 (2017), S404–S433. DOI: 10.1137/16M1082020.

[31] David Schneider and Benjamin Uekermann. “Efficient Partition-of-Unity Radial-Basis-Function Interpolation for Coupled Problems”. In: *SIAM Journal on Scientific Computing* 47.2 (2025), B558–B582. DOI: 10.1137/24M1663843.

[32] Valentina Schüller et al. “A Simple Test Case for Convergence Order in Time and Energy Conservation of Black-Box Coupling Schemes”. In: *WCCM-APCOM*. 2022. DOI: 10.23967/wccm-apcom.2022.038.

[33] Stuart R. Slattery. “Mesh-free data transfer algorithms for partitioned multiphysics problems: Conservation, accuracy, and parallelism”. In: *Journal of Computational Physics* 307 (2016), pp. 164–188. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2015.11.055.

[34] Gilbert Strang. *Computational Science and Engineering*. Philadelphia, PA: Wellesley-Cambridge Press, 2007. DOI: 10.1137/1.9780961408817.

[35] H. S. Tang, R. D. Haynes, and G. Houzeaux. “A Review of Domain Decomposition Methods for Simulation of Fluid Flows: Concepts, Algorithms, and Applications”. In: *Archives of Computational Methods in Engineering* 28.3 (May 2021), pp. 841–873. ISSN: 1886-1784. DOI: 10.1007/s11831-019-09394-0.

[36] Yu-Hang Tang et al. “Multiscale Universal Interface: A concurrent framework for coupling heterogeneous solvers”. In: *Journal of Computational Physics* 297 (2015), pp. 13–31. DOI: 10.1016/j.jcp.2015.05.004.

[37] Marc Amorós Trepat. “Time stepping review of open-source solvers”. en. MA thesis. Technical University of Munich, 2024. URL: https://mediatum.ub.tum.de/1740774.

[38] Benjamin Uekermann. “Partitioned Fluid-Structure Interaction on Massively Parallel Systems”. Dissertation. Technical University of Munich, 2016. DOI: 10.14459/2016md1320661.

[39] Niklas Vinnitchenko. “Evaluation of higher-order coupling schemes with FEniCS-preCICE”. en. Bachelor’s thesis. Technical University of Munich, 2024.

[40] H. G. Weller et al. “A Tensorial Approach to Computational Continuum Mechanics Using Object-oriented Techniques”. In: *Computational Physics* 12.6 (Nov. 1998), pp. 620–631. ISSN: 0894-1866. DOI: 10.1063/1.168744.