# In-Situ Visualization of Solver Residual Fields

Kai Sdeo, Boyan Zheng, Marian Piatkowski, and Filip Sadlo

Heidelberg University, Germany
{kai.sdeo, boyan.zheng, marian.piatkowski}@iwr.uni-heidelberg.de
sadlo@uni-heidelberg.de

**Abstract.** Whereas the design and development of numerical solvers for field-based simulations is a highly evolved discipline, and whereas there exists a wide range of visualization techniques for the (in-situ) analysis of their numerical results, the techniques for analyzing the operation of such solvers are rather elementary. In this paper, we present a visualization approach for in-situ analysis of the processes within numerical solvers. That is, instead of visualizing the data that result from such solvers, we address the visualization of the processes that generate the data. We exemplify our approach using different simulation runs, and discuss its in-situ application in high-performance computing environments.

**Keywords:** Residual analysis · Solver analysis · In-situ visualization.

## 1 Introduction

During the last decades, numerical simulation has been more and more replacing physical experiments in science and engineering—providing various advantages, including reproducibility, simplified setup, and reduced cost. The wide application of simulation techniques has, on the other hand, led to an intense increase in compute demands, necessitating ever-growing supercomputing facilities. This development is currently at the threshold to exascale computing, where the limited communication bandwidths and storage resources inhibit storage and subsequent analysis of finely space-time resolved results. By employing preprocessing and data reduction at the compute nodes during simulation, and thus enabling transfer of only the essential information to the user, in-situ visualization is considered a main solution to this dilemma in today's and tomorrow's computing.

In this paper, we do not follow the typical track of in-situ visualization techniques, i.e., we do not process the data that a compute node produces. Instead, it is our aim to complement these techniques by providing a tool to support the effective operation of numerical solvers in high-performance computing. Traditionally, numerical solvers are monitored in terms of a defect, or *total residual*, which is typically a single value for each iteration of a solver. This residual is traditionally plotted (Figure 1) during operation of the solver, to monitor its convergence behavior and accuracy. However, whereas such simple plotting is good at indicating the trend of the residual and thus at indicating convergence problems, it is typically not sufficient to help in understanding the structure and

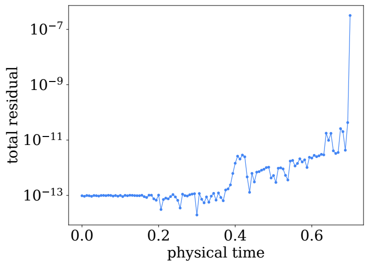


**Fig. 1.** Diverging simulation (Kármán Run I dataset), monitored with traditional residual plot. Solver fails to converge after last time step (physical time step 112). While such plots can indicate problems, they do not support the reasoning of their causes.

the causes of such problems. In this paper, we examine approaches to provide more information to the user, to give a better picture of solver behavior, and to support in-situ operation, and development of numerical solvers in general.

Our approach addresses field-based simulations, with the Navier–Stokes equations being a prominent example, and is based on residual *fields*. Residuals can be seen as the discrepancy of an (intermediate) solution with respect to the discretized formulation of the underlying problem and the employed numerical scheme. That is, the solver minimizes some norm of these residuals to obtain a solution. Instead of plotting the total residual, i.e., the norm of such intermediate residuals, we map the residuals back to the domain, and thus obtain spatial fields. Consequently, such *residual fields* are available for all quantities a solver computes in such a manner (Figure 2).

In this work, we investigate the analysis of residual fields to obtain additional insights into solver dynamics, also w.r.t. slow (or failing) convergence. Our overall approach is designed for in-situ operation in field-based solvers. Nevertheless, our current prototype is implemented and evaluated only on a single compute node—integration and evaluation in a high-performance compute environment is to be carried out as future work. That is, this work researches the basics for in-situ residual field visualization.

## 2 Related Work

Surprisingly, we have not been able to find any previous work on the visualization of residual fields. Whereas there is virtually no numerical solver that does not perform a plotting of a residual, the spatiotemporal structure of residual fields has been ignored so far, at least from a visualization research point of view.

Convergence analysis [3] is a very broad and mature field, which, however, does typically not make use of (advanced) visualization techniques, and does not consider residual fields. Less closely related, but more graphical, are techniques that visualize the "state space" of numerical algorithms, such as Newton fractals [6], which depict the convergence behavior of Newton's method. Convergence analysis of integration schemes, e.g., with respect to A-stability [4, 3] is also an example of weakly related work, where a, however, simple graphical representation is involved. On the other hand, there is only one work that we are aware of, which visualizes technical processes within a solver, i.e., our visualization of piecewise linear interface calculation inherent to two-phase flow simulation [7]. That work, however, does also not investigate residual fields, and rather focuses on geometric and quantitative analysis.

Another related field, where solver-specific representation is taken into account, is the visualization of higher-order data, such as in the case of discontinuous Galerkin [5, 10], finite-element [9], or particle-partition of unity [12] simulation. Similar to these approaches is the incorporation of simulation models into interpolation techniques to accomplish model-consistent interpolation [11].

Various approaches have been presented so far in the general field of in-situ visualization. Kress [8] can provide an introduction to the topic. Contributions in this field range, e.g., from particle-based simulations [13] to climate research [14].

## 3 Method

Our overall approach consists of several building blocks, which we present here in the order of increasing solver process detail. These building blocks complement each other, and make up our overall approach for in-situ analysis of residual fields. Section 3.1 gives some background on typical solver processes, establishes terminology with respect to solver iterations and physical time, and details the concept of residual fields. Since direct investigation of these data, e.g., as animations of residual fields, would lead to issues with perception, exploration, and I/O bandwidth, we present in Section 3.2 aggregation of residual fields. Because such aggregations provide an overview of the convergence behavior of simulations, but at the same time suffer from temporal "averaging", we complement them with the concept of residual curves, presented in Section 3.3. These curves are obtained by dimensionality reduction of sets of residual fields—either representing sets of solver iterations or sets of physical time steps—and enable qualitative analysis of such sets. For detailed inspection of the sets, the approach is complemented with residual stacks (Section 3.4), which provide them in space-time or space-iteration representation. Finally, in-situ context is discussed in Section 3.5.

### 3.1 Solvers and Residual Fields

The addressed solvers for field-based simulations discretize the space domain $\Omega \subset \mathbb{R}^n$ into cells defined by nodes $\mathbf{x}_i \in \Omega$, and the time domain $\mathrm{T} \subset \mathbb{R}$ into physical time steps $t_j \in \mathrm{T}$. Each physical time step gives rise to (a set of) fields, such as a velocity $\mathbf{u}(\mathbf{x}, t_j)$ and a pressure $p(\mathbf{x}, t_j)$ field (Figure 2(b) and (c)). Note that we exemplify our approach by means of 2D flow simulations, i.e., $n = 2$, position $\mathbf{x} \in \Omega \subset \mathbb{R}^2$, and $\mathbf{u} : \Omega \times \mathrm{T} \to \mathbb{R}^2$.

Each of these physical time steps (i.e., fields) is the result of a process that minimizes some norm of the residuals. Since many underlying problems are inherently nonlinear, this minimization is often accomplished by iterative techniques.

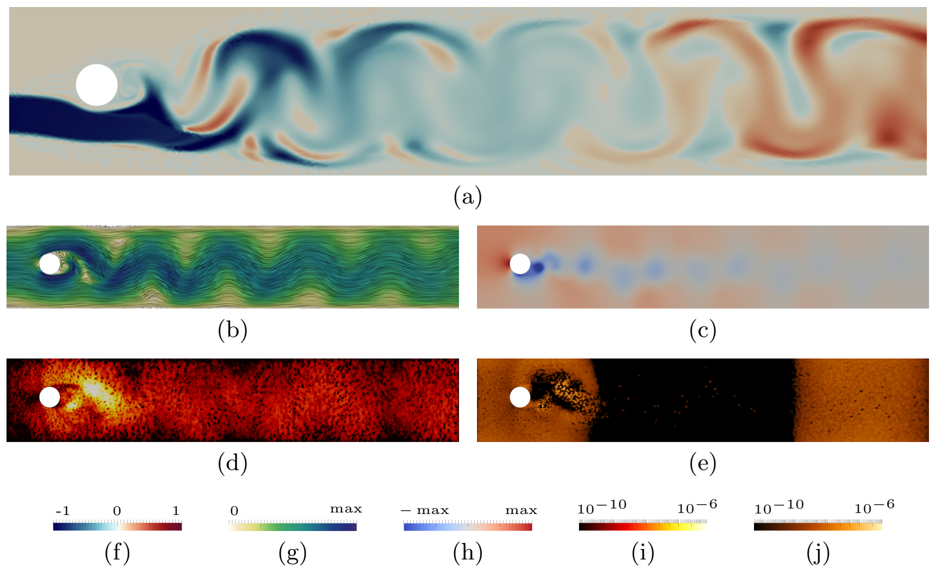
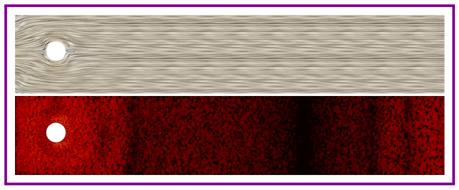


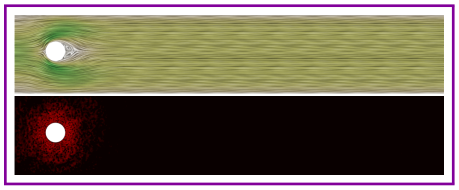
**Fig. 2.** 2D flow around a cylinder (white), inlet on the left, outlet on the right. Last physical time step ($t_{112} = 0.7$ s) of Kármán Run I dataset, before simulation diverges. (a) Visualization by a tracer $c$ (seeded at dark region on left boundary). (b) Velocity $\mathbf{u}$ by line integral convolution [2], with color-coded magnitude. (c) Pressure field $p$. (d) Residual field of velocity $\mathbf{r}(\mathbf{u})$ indicates severe problems just behind the cylinder. (e) Residual field of pressure $r(p)$ exhibits some high-valued noise just behind cylinder. (f)–(j) Respective color maps where max represents the maximum value in the field.

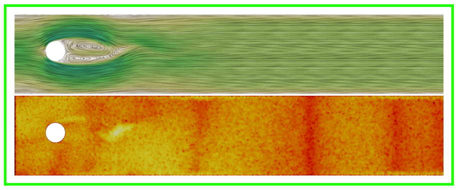
In other words, each physical time step $t_j$ is typically the result of a sequence of iterations. We name them *solver iterations* and denote them $t_j^k$, i.e., the $k^{th}$ solver iteration for physical time step $j$. The solver stops iteration when the total residual drops below a user-defined threshold. Thus, at the example of the velocity field for physical time step $t_j$, the first intermediate solution is $\mathbf{u}(\mathbf{x}, t_j^1)$, and assuming that 11 iterations are needed to reach the total residual threshold, the resulting solution is $\mathbf{u}(\mathbf{x}, t_j) := \mathbf{u}(\mathbf{x}, t_j^{11})$.

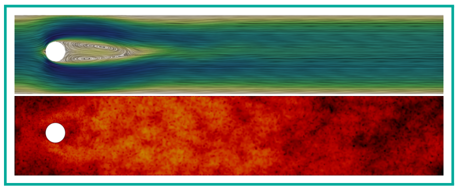
By mapping the residuals back to the domain, we obtain, on the one hand, for each physical time step $t_j$ the corresponding residual fields $\mathbf{r}(\mathbf{u}(\mathbf{x}, t_j))$ and $r(p(\mathbf{x}, t_j))$, and on the other hand, for each solver iteration $t_j^k$, the residual fields $\mathbf{r}(\mathbf{u}(\mathbf{x}, t_j^k))$ and $r(p(\mathbf{x}, t_j^k))$. Notice that the residual fields inherit the dimensionality from the quantity that is being solved, e.g., the residual field of velocity is a vector field, too. For the remaining part of this paper, we focus on velocity residual fields, and defer investigation of the pressure residual to future work.

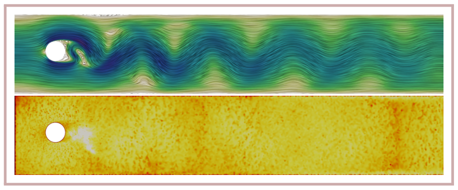
Figure 3 provides selected velocity residual fields for two different simulation runs, a diverging one (Kármán Run I) and a non-diverging one (Kármán Run II). One can see, that residual fields for these simulations have a rather noisy structure in space and time, but also that the magnitude at the last physical time step ($t_{112}$) that still converged in Kármán Run I is overall high, and that it


(a) $t_9 \approx 0.05$ s


(b) $t_{160} = 1$ s


(c) $t_{65} \approx 0.4$ s


(d) $t_{560} = 3.5$ s


(e) $t_{111} \approx 0.7$ s

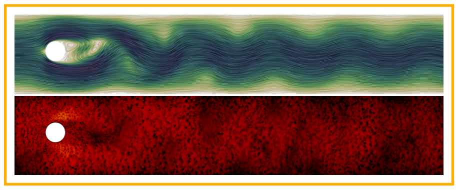

(f) $t_{1600} = 10$ s

**Fig. 3.** Exemplary fields for diverging Kármán Run I (left column) and non-diverging Run II (right column), with velocity $\mathbf{u}(\mathbf{x}, t_j)$ (top, color map from Figure 2) and residual field $\|\mathbf{r}(\mathbf{u}(\mathbf{x}, t_j))\|$ (bottom, color map from Figure 4). The colored frames correspond to the clusters of solver behavior in Figure 5 (ellipses with corresponding colors).

is particularly high directly behind the obstacle (Figure 3(e)). Thus, this region may be considered the "ignition spark" of solver divergence in the computation of the subsequent time step.

### 3.2 Aggregated Residual Fields

Although residual fields can provide insights in solver dynamics (as will be further investigated in Section 4), their visual analysis at full time resolution would suffer from perceptual issues, and—even more important—it would exceed the available bandwidth (and storage) if all residual fields for all physical time steps would be communicated. On the other hand, it is a common approach in field-based numerical simulation to discard most physical time steps and store, e.g., only every 100$^{\text{th}}$ for later analysis. However, assuming that the available bandwidth would allow us to also communicate every corresponding 100$^{\text{th}}$ residual field for analysis (which might, however, often not be feasible, in particular in high-scale computing), the behavior of the solver during the 99 time steps in between would still remain unknown. This would be in particular a problem because the probability that the solver experiences issues during the 99 discarded time steps is much higher than during the communicated one.

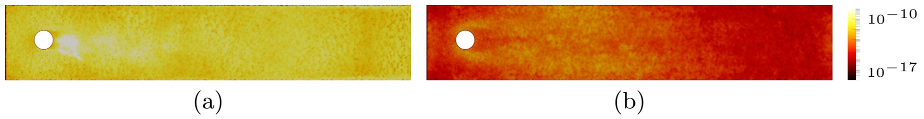


**Fig. 4.** Maximum-aggregated velocity residual field $r^{\mathcal{R}_\mathbf{u}}_{\max}(\mathbf{x})$ for time steps $t_1$–$t_{111}$ of diverging the Kármán Run I (a), and for time steps $t_1$–$t_{1600}$ of the non-diverging Run II (b). In the diverging case, residual magnitude is overall higher, and particularly high just behind the obstacle. The non-diverging case, in contrast, exhibits low residual magnitude behind the cylinder, and in general a more uniform distribution.

This motivates aggregation of residual fields, i.e., the combination of the information from sets of residual fields. In the abovementioned scenario, we could combine the residual fields for the 99 time steps that are discarded. For example, for a set $\mathcal{R}_\mathbf{u}$ of velocity residual fields, we define the maximum-aggregated residual field $r^{\mathcal{R}_\mathbf{u}}_{\max}(\mathbf{x})$ as follows:

$$r^{\mathcal{R}_\mathbf{u}}_{\max}(\mathbf{x}) := \max_{\mathbf{r}_j \in \mathcal{R}_\mathbf{u}} \|\mathbf{r}_j(\mathbf{x})\|. \tag{1}$$

Assume that the solver fails to converge, e.g., at time step 50, in such a scenario. In this case, this time step will typically exhibit very high residual field magnitude (Figure 2(d)), and thus it would dominate $r^{\mathcal{R}_\mathbf{u}}_{\max}(\mathbf{x})$. Therefore, in case of solver divergence, we omit the diverging time step from $\mathcal{R}_\mathbf{u}$, and communicate both the aggregated $r^{\mathcal{R}_\mathbf{u}}_{\max}(\mathbf{x})$, as well as the residual field of the diverging time step for analysis. The maximum-aggregated residual field enables the identification of spatial regions that exhibited low residual magnitude over the entire set, or regions that exhibited at least for one moment large residual magnitude.

Since aggregation is a local operation, and since the aggregated field is of the size of a single residual field and thus (the respective part) should typically fit into a compute node's memory, it lends itself well for in-situ visualization. As motivated above, aggregated residual fields can be computed for series of physical time steps that are not communicated for analysis, and in case of solver problems, they can give insight about the reasons of the problems. In Figure 4(a), i.e., the diverging Kármán Run I example, $r^{\mathcal{R}_\mathbf{u}}_{\max}(\mathbf{x})$ shows that the region behind the cylinder exhibited (at least for short times) large velocity residual magnitude, whereas in the upstream region of the cylinder, residual magnitude has always been lower. This, for example, indicates that the solver problem is not related to the inlet, but is rather related to the obstacle. In contrast, for the non-diverging Run II, Figure 4(b) shows that the velocity residual has always been low behind the obstacle during the aggregated time interval, and that velocity residual magnitude is overall lower and more spatially uniform than in Figure 4(a). Such distributions are typical for simulation runs that do not exhibit problems.

## 3.3 Residual Curves

So far, we introduced residual fields and their aggregation. Whereas these approaches can be used to obtain detailed insight into the dynamics of a solver

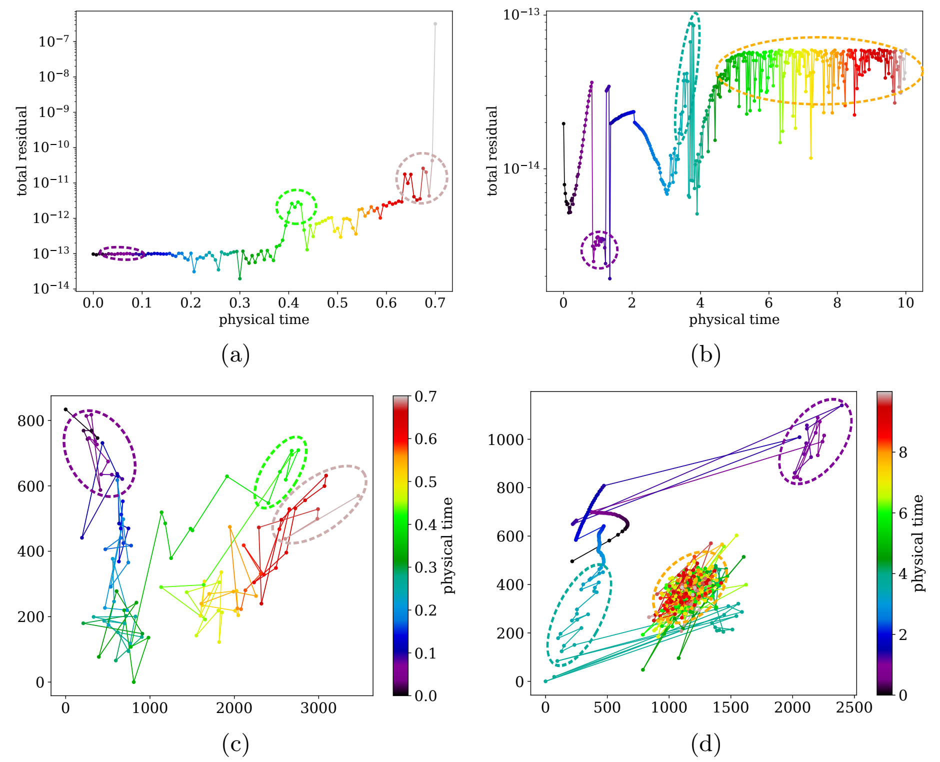


**Fig. 5.** Traditional total residual plots (top) and our residual curves (bottom), for the diverging Kármán Run I (left) and non-diverging Run II (right), w.r.t. physical time steps $t_j$. The residual curves reveal clusters of simulation behavior, some indicated with ellipses, in which residual fields exhibit similarities, see Figure 3 for respective residual fields. The non-diverging run exhibits more compact clusters and a more regular curve.

with respect to its residuals, their rich structure impedes their application for monitoring, quick qualitative overview, and context. This motivates our next complementing concept, which we denote *residual curves*.

We obtain these curves by computing from each residual field a vector

$$\rho_l(t_j) := \ln \|\mathbf{r}(\mathbf{u}(\mathbf{x}_l, t_j))\|, \tag{2}$$

where $\mathbf{x}_l$ is the respective node of the simulation grid. Note that we employ logarithmic mapping of the individual residual element magnitude to avoid clutter, i.e., to obtain more expressive residual curves.

This way, each residual field consisting of $m$ nodes provides an $m$-dimensional vector $\boldsymbol{\rho}$, which in turn represents a point in $m$-dimensional space. As a consequence, series of residual fields represent polylines in this $m$-dimensional space. We take all points $\boldsymbol{\rho}$ of such a series, and apply dimensionality reduction by means of principal component analysis (PCA). That is, we project the polylines to 2D space spanned by the two major PCA eigenvectors.

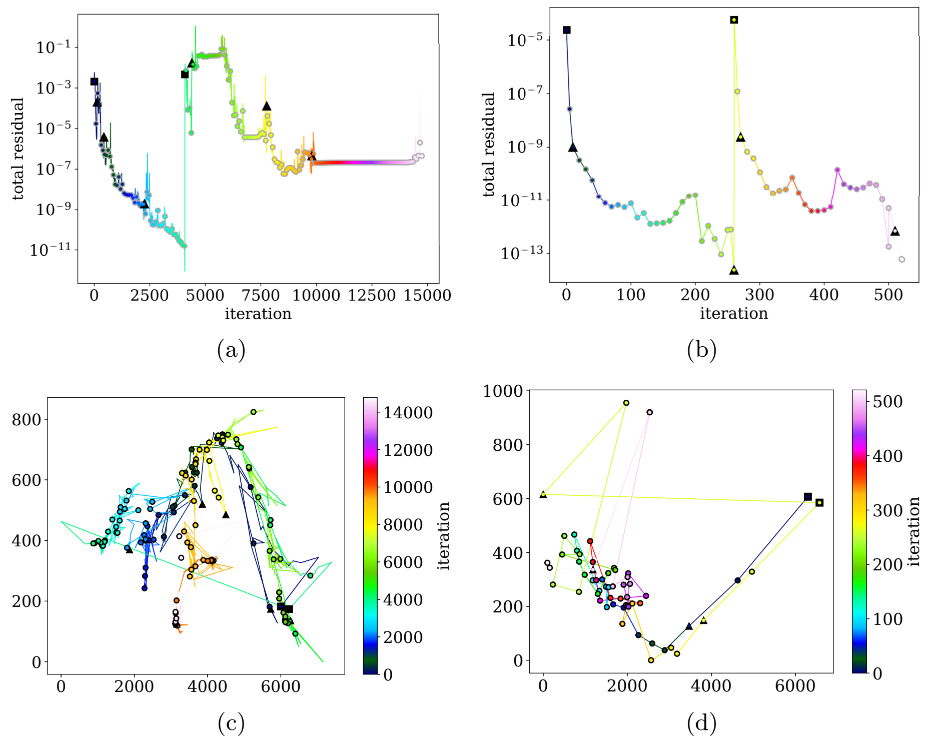


**Fig. 6.** Total residual plots (top) and residual curves (bottom) for the diverging Kármán Run I (left) and non-diverging Run II (right) w.r.t. solver iterations $t_j^k$ for the computation of physical time step $t_{112}$ (left) and physical time step $t_{1600}$ (right). For the diverging run, the residual curve exhibits several phases, each with substantial perturbations. For the non-diverging run, the shape of the residual curve is more ordered, and both stages (starting at black square glyphs) exhibit a similar behavior.

Remember that residual fields can be computed for both series of physical time steps $t_j$ and series of solver iterations $t_j^k$. Whereas series of physical time steps (e.g., the discarded ones) provide analysis of solver behavior over physical time, one can investigate the computation of a single physical time step by means of series of solver iterations. Thus, respective residual curves can provide an overview of the dynamics during both physical time steps and solver iterations. We visualize the residual curves in their 2D projection, with a color map that encodes the order of the individual residuals, to provide a notion of succession. Figure 5 gives an example for residual curves with respect to physical time, whereas Figure 6 provides an example for residual curves of solver iterations.

In Figure 5, we can identify clusters w.r.t. simulation behavior, which are marked by ellipses and exemplified in Figure 3. Residual fields within such clusters turn out to look similar. In Figure 6, which visualizes solver iterations, we additionally indicate the start of each iteration stage with a black box glyph,

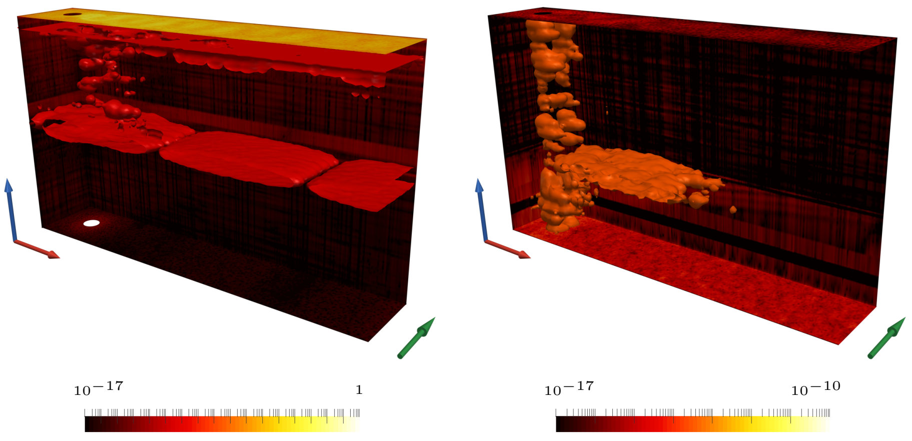


**Fig. 7.** Isosurfaces of physical-time residual stacks for diverging Kármán Run I (a) and non-diverging Run II (b). The structure at the center of (a) corresponds to the light green peak/ellipse in Figure 5(a) and (c). The dominant structure at the lower third of (b) corresponds to the teal green peak/ellipse in Figure 5(b) and (d). As in all stacks, $x$-axis by red arrow, $y$-axis by green, and physical time / solver iteration by blue.

and the start of each Newton iteration with a triangle glyph. This nicely reveals the similarity of solver behavior in the two iteration stages in this example.

## 3.4 Residual Stacks

So far, we have seen that residual curves can provide a qualitative overview of sequences of physical time steps and sequences resulting from solver iterations. However, once, e.g., a cluster in these curves has been identified, a more detailed analysis of the residual fields is required. One could color-map each residual field and compose the respective sub-sequences into animations, the observation of these animations would, however, involve perceptual difficulties. For time-dependent 2D simulations, space-time representation, i.e., treatment of the time axis as an additional spatial dimension, is a common approach that at least partially avoids these issues. We name the resulting representations of sequences of residual fields *residual stacks*. Figure 7 provides physical-time residual stacks for the Kármán Run I and II cases, i.e., stacked with respect to physical time steps $t_k$. Note that in all visualizations of the stacks, the $x$-axis is visualized in red, the $y$-axis green, and the physical time or solver iteration axis blue.

Whereas physical-time residual stacks serve well for understanding long-term behavior of a solver, e.g., to understand how residual structure develops over physical time and eventually leads to divergence, solver-iteration residual stacks provide the analog with respect to solver iterations. Since the residual fields tend to be rather noisy in space and (iteration) time, we employ smoothing prior to, e.g., isosurface extraction.

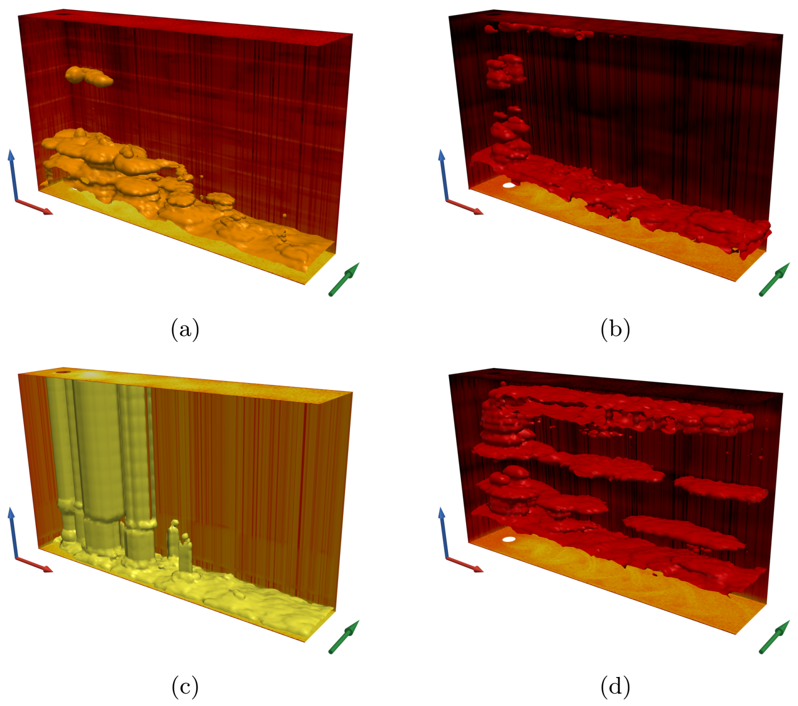


**Fig. 8.** Solver-iteration residual stacks for stage one (top) and two (bottom) of diverging Kármán Run I (left, $t_{112} = 0.7$ s) and non-diverging Run II (right, $t_{1600} = 10$ s). Whereas the first stage exhibits similar structure in both runs, the second stage shows very high residuals for the diverging run, which eventually leads to divergence. Please refer to Figure 7(a) for the color maps.

In Figure 5, we have identified clusters for both simulation runs (light green ellipse in the diverging, and teal green in the non-diverging case) which correspond to a temporal peak in the total residual plots. Using residual stacks, these clusters can be investigated by means of isosurfaces. We observe a large red structure at the center of Figure 7(a), and a cloud-shaped horizontal structure at the lower third of Figure 7(b), which is related to the structure in Figure 3(d).

In Figure 8, we analyze the computation of the last physical time step $t_{112}$, which did not converge, in the diverging run, and the last physical time step $t_{1600}$ that was computed in the non-diverging run. Since the underlying solver performs the computation of a physical time step in two stages, we provide a solver-iteration residual stack for both stages separately. While the first stage of the diverging run (Figure 8(a)) still converged, we can see the cylindrical structures of too high residual in the vicinity of the obstacle during the second stage (Figure 8(c)). The non-diverging run (Figure 8(b) and (d)), in contrast, exhibits structures similar to those of the first stage in the diverging run. Additionally, there seems to be more solver dynamics in the second stage of the non-diverging run, which might relate to the used higher-order time integration scheme.

The approach of residual field stacking provides quite detailed insight into residual dynamics, and motivates further analysis with feature extraction techniques. On the other hand, the runtime and memory overhead, especially for the iteration stacks, is large compared to the other building blocks of our approach.

### 3.5 In-Situ Application

So far, we have identified aggregated residual fields (Section 3.2) to fit well into in-situ environments because their computation is local and well-suited for, e.g., domain decomposition. However, residual curves (Section 3.3) and residual stacks (Section 3.4) are global constructs in space-(iteration-)time, therefore require entire sets of residual information, and are thus not straightforward to employ in in-situ contexts.

In our implementation, we follow a sliding-window approach for these constructs. That is, we maintain the residual fields (i.e., physical-time residual stacks) of a fixed number of most recent physical time steps, as well as a fixed number of most recent solver-iteration residual stacks. These data are, however, not communicated for analysis by default. Only if issues are determined, either because the simulation diverged, or based on the regularly communicated aggregated residual fields, the user can request these data for computation of residual curves as well as composition of residual stacks.

## 4 Results

We start with some details on the implementation and a performance analysis (Section 4.1). Then, we describe the two simulation runs that were used to present our technique (Section 4.2), followed by an experiment investigating the impact of grid resolution on residual fields (Section 4.3), and some experiments on mesh refinement based on residual fields (Section 4.4).

### 4.1 Implementation and Timings

The simulation code underlying our experiments is based on DUNE [1], which we extended to access and export the residual fields and our derived representations. In our experiments, we export $\mathbf{u}(\mathbf{x}, t_j)$ and $\mathbf{r}(\mathbf{u}(\mathbf{x}, t_j))$ every fourth time step, and $\mathbf{r}(\mathbf{u}(\mathbf{x}, t_j^k))$ at every tenth iteration. The aggregation of $r_{\max}^{\mathcal{R}_\mathbf{u}}(\mathbf{x})$ is computed also within the solver, but from consecutive time steps. The overhead of our technique can be obtained from Table 1—and the absolute overhead is rather small. For larger grids, however, exporting the solver iterations becomes expensive, especially due to the memory overhead that grows with the number of iterations. In our prototype, the computation of the residual curves was accomplished using a separate Python program. For the PCA, the sklearn Python library was used. Note that our naive implementation writes and reads thousands of residual fields (one for each solver iteration) in this case, and could be optimized substantially. Still, the method was, in our experiments, fast enough to be used for monitoring.

### 4.2 Kármán Runs

This experiment consists of two simulation runs (Kármán Run I and Kármán Run II) of a 2D flow around a cylinder, which exhibits vortex shedding, and is used to demonstrate the interplay of the presented building blocks. It is simulated with a physical time step size of 0.00625 s, a triangular simulation grid consisting of 17552 nodes and 34422 cells, and features an inlet along its left boundary, and an outlet along its right boundary (Figure 2). The two simulation runs differ only with respect to inlet velocity magnitude ($\|\mathbf{u}_{\text{in}}\| = 40.5$ m/s for Run I and $\|\mathbf{u}_{\text{in}}\| = 1.5$ m/s for Run II), causing Run I to diverge at time step $t_{112}$, whereas Run II does not diverge during the simulated 1600 time steps.

### 4.3 Mesh Resolution Experiment

To gain more insight on the interpretation of residual fields, we investigate different simulation grid resolutions (19766 cells, 4738 cells, 986 cells, and 444 cells), with the same relative refinement around the obstacle as for the Kármán Run I and II. The basic setup is the same as described in Section 4.2, with $\|\mathbf{u}_{\text{in}}\| = 1.5$ m/s. As can be seen from Figure 9, reducing the resolution increases the overall residual field magnitude. However, at the lowest resolution, residual magnitude drops again. This seems to be related to the fact that this resolution does not exhibit vortex shedding anymore, i.e., it results in a quasi-stationary solution. The corresponding residual curves exhibit nice tight clusters for the highest resolution, whereas they exhibit a more irregular behavior for the medium resolution. Notice also that for larger cell sizes or higher inlet velocities (Figure 4), the solver shows residual peaks behind the obstacle, which may be related to too large time step size with respect to the cell size. In contrast, Figure 9 shows a rather uniform residual distribution, which indicates a well chosen ratio between cell size and inlet velocity. The simulation with the grid consisting of 19766 cells shows results similar to those of Kármán Run II, and is therefore not depicted.

**Table 1.** Performance measurements. Number of nodes (nodes) and cells (cells) of respective simulation grids (dataset), time spent for simulation without visualization part (sim. only), time used for computing maximum-aggregated residual field ($r^{\mathcal{R}_\mathbf{u}}_{\max}$), time spent for physical-time residual stack composition ($\mathbf{r}(\mathbf{u}(\mathbf{x}, t_j))$) and solver-iteration residual stack composition ($\mathbf{r}(\mathbf{u}(\mathbf{x}, t_j^k))$), followed by time spent for residual curve (RC) computation of physical time series (RC $t_j$), and solver iterations (RC $t_j^k$). All simulations perform 1600 physical time steps. (This is the reason why the Kármán Run I was excluded from the comparison).

| dataset | nodes | cells | sim. only | $r^{\mathcal{R}_\mathbf{u}}_{\max}$ | $\mathbf{r}(\mathbf{u}(\mathbf{x}, t_j))$ | $\mathbf{r}(\mathbf{u}(\mathbf{x}, t_j^k))$ | RC $t_j$ | RC $t_j^k$ |
|---|---|---|---|---|---|---|---|---|
| Section 4.3 | 2496 | 4738 | 1 h 07 m | $<$ 1 m | $<$ 1 m | 10 m | 37 s | 8 s |
| Section 4.3 | 10142 | 19766 | 18 h 57 m | 4 m | 3 m | 2 h 04 m | 2 m 20 s | 37 s |
| Kár. Run II | 17552 | 34422 | 23 h 06 m | 2 h 04 m | 1 h 59 m | 10 h 27 m | 3 m 38 s | 59 s |

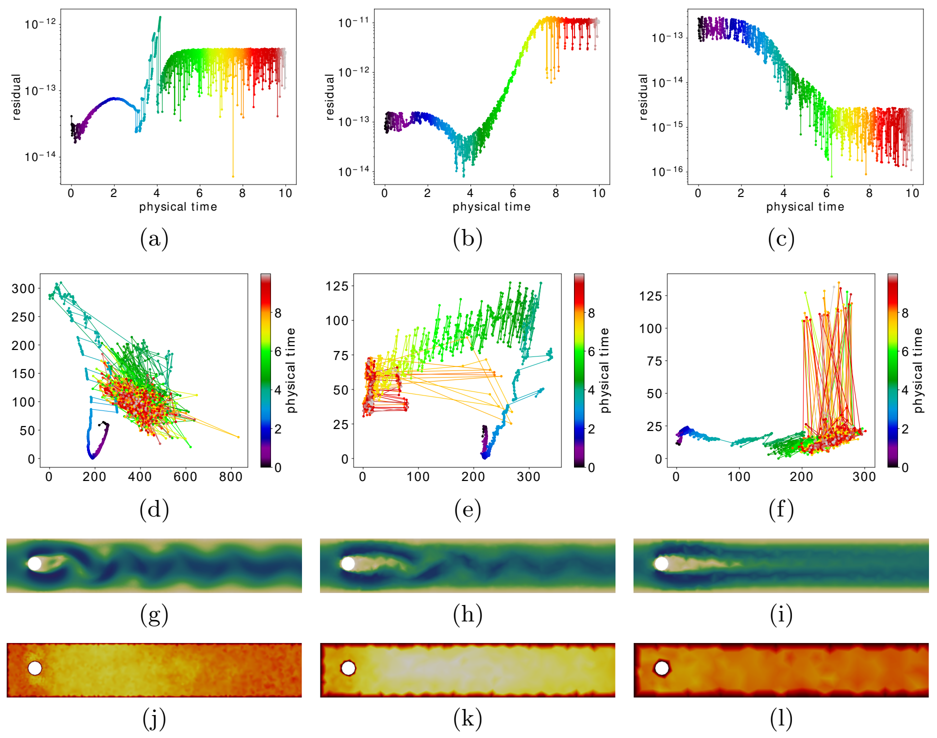


**Fig. 9.** Mesh Resolution Experiment, with varying mesh resolution: 4738 cells (left column), 896 cells (middle column), and 444 cells (right column). (a)–(c) Total residual plots. (d)–(f) Residual curves. (g)–(i) Velocity $\mathbf{u}(\mathbf{x}, t_{1600})$ (color map from Figure 2(g)). (j)–(l) Maximum-aggregated velocity residual $\mathbf{r}^{\mathcal{R}_\mathbf{u}}_{\max}(\mathbf{x})$ (color map from Figure 4). Residuals increase with decreasing mesh resolution, until a resolution is reached that does not exhibit vortex shedding, but nevertheless shows a drop in residual magnitude. Our residual curves, in contrast, show increasing irregularity from (d) to (e).

## 4.4 Grid Refinement Experiment

In this last experiment, we want to investigate grid refinement based on residual fields. As discussed in Section 4.3, higher grid resolutions with too large time step sizes tend to result in larger residuals. This motivates the investigation of residual fields for grid refinement, and at the same time provides a better understanding of residual field behavior. Due to the rather uniform residual magnitude distribution in the Mesh Resolution Experiment, we choose a similar setup with uniform triangulation, however with a modified domain exhibiting a narrow passage (a slit), inlet velocity 0.25 m/s, with maximum velocity $\|\mathbf{u}_{\max}\| = 2.05$ m/s at the slit, and a corresponding time step of 0.0015 s. Figure 10 shows the results of this experiment. As one would expect, all residual fields for higher mesh resolution than the lowest resolution grid (Figure 10(a)) show smaller residual magnitudes, whereas for similar mesh resolutions, the results differ less. Fig-

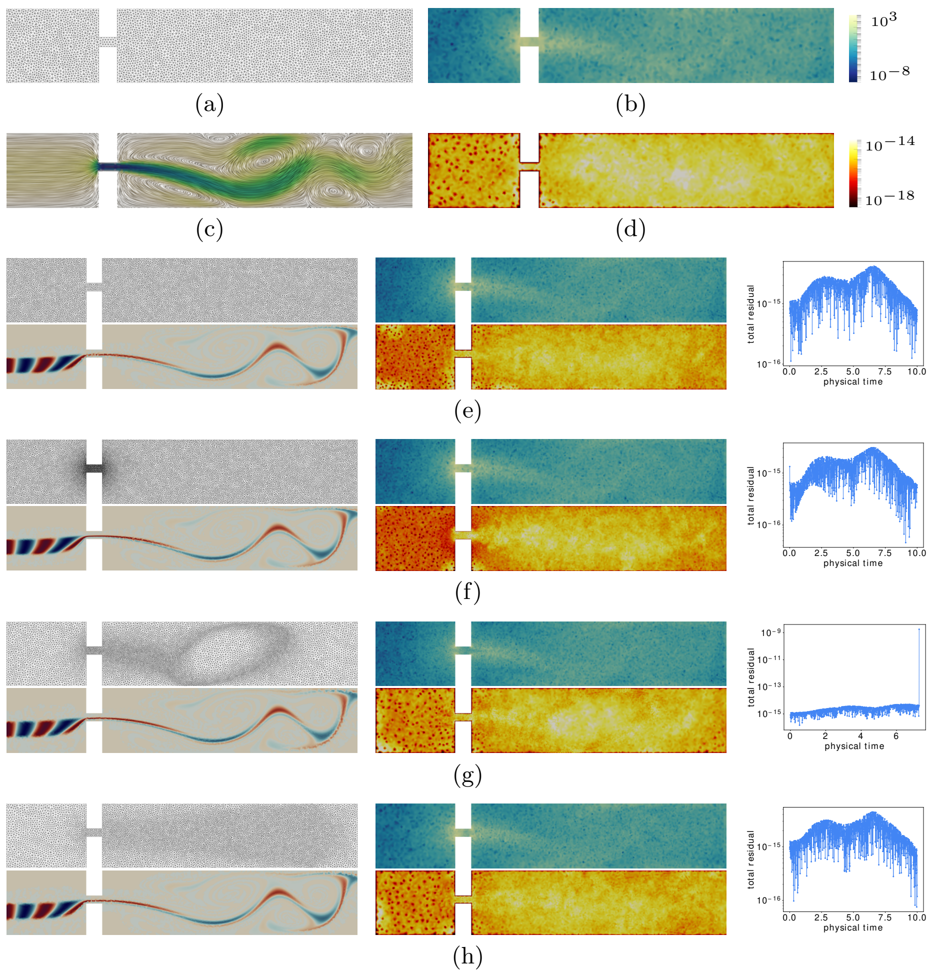


**Fig. 10.** Grid Refinement Experiment at $t_{4840} = 7.26$ s. (a) Grid with approximately 10,000 cells, (b) respective divergence field $\nabla \cdot \mathbf{u}(\mathbf{x}, t_{4840})$, (c) velocity $\mathbf{u}(\mathbf{x}, t_{4840})$, and (d) maximum-aggregated velocity residual $\mathbf{r}^{\mathcal{R}_\mathbf{u}}_{\max}(\mathbf{x})$. (e)–(h) Include tracer $c(\mathbf{x}, t_{4840})$ (red/blue) instead of $\mathbf{u}(\mathbf{x}, t_{4840})$, and total residual plots (right). These grids all have about 20,000 cells, and are adapted (refined) (e) uniformly, (f) along the slit, (g) relative to divergence, and (h) relative to maximum-aggregated velocity residual $\mathbf{r}^{\mathcal{R}_\mathbf{u}}_{\max}(\mathbf{x})$. For color maps of $c$ and $\mathbf{u}$, please refer to Figure 2(f) and (g).

ure 10(e) and (f) show a rather non-uniform residual magnitude distribution, whereas divergence-based refinement (Figure 10(g)) results in a more even distribution of residual magnitude. Our residual magnitude-based refinement (h) is similar to the divergence-based, but exhibits more uniform and overall slightly lower residual magnitude. Nevertheless, thorough investigation of the utility of residual fields for mesh refinement has to be subject of future work.

## 5 Conclusion

In this work, we have presented the concept of residual field visualization, and provided several building blocks whose interplay enables such an analysis in in-situ environments. This first investigation revealed interesting patterns and interrelations in the space-time and space-iteration structure of residual fields, which may provide a basis for future research of more advanced techniques for the analysis of solver behavior. On the other hand, it is clear that, due to the high complexity of solver processes, this first work cannot provide in-depth insights into the novel field of residual field visualization—it is the aim of this work to foster this new topic in visualization research. As future work, we plan to investigate feature extraction from residual fields, as well as more effective in-situ integration of our approach.

**Acknowledgments** This work was supported in part by Forschungsallianz Baden-Württemberg, "Data-Integrated Simulation Science (DISS)", and the Heidelberg Graduate School of Mathematical and Computational Methods for the Sciences (HGS MathComp), founded by DFG grant GSC 220 in the German Universities Excellence Initiative.

## References

1. Blatt, M., Burchardt, A., Dedner, A., Engwer, C., Fahlke, J., Flemisch, B., Gersbacher, C., Gräser, C., Gruber, F., Grüninger, C., Kempf, D., Klöfkorn, R., Malkmus, T., Müthing, S., Nolte, M., Piatkowski, M., Sander, O.: The distributed and unified numerics environment, version 2.4. Archive of Numerical Software **100**(4), 13–29 (2016)
2. Cabral, B., Leedom, L.C.: Imaging vector fields using line integral convolution. In: Proceedings of 20th Annual Conference on Computer Graphics and Interactive Techniques. pp. 263–270 (1993)
3. Dahlquist, G., Björck, Å.: Numerical Methods. Dover Books on Mathematics, Dover Publications (2003)
4. Dahlquist, G.G.: A special stability problem for linear multistep methods. BIT Numerical Mathematics **3**(1), 27–43 (1963)
5. Haimes, R., Liu, E., Kirby, R.M., Nelson, B.: ElVis: A system for the accurate and interactive visualization of high-order finite element solutions. IEEE Transactions on Visualization and Computer Graphics **18**, 2325–2334 (2012)
6. Hubbard, J., Schleicher, D., Sutherland, S.: How to find all roots of complex polynomials by Newton's method. Inventiones mathematicae **146**(1), 1–33 (2001)
7. Karch, G.K., Sadlo, F., Rauschenberger, P., Meister, C., Eisenschmidt, K., Weigand, B., Ertl, T.: Visualization of piecewise linear interface calculation. In: Proceedings of IEEE Pacific Visualization Symposium (PacificVis). pp. 121–128 (2013)
8. Kress, J.: In situ visualization techniques for high performance computing. Tech. rep., University of Oregon (2017)
9. Schollmeyer, A., Froehlich, B.: Direct isosurface ray casting of NURBS-based isogeometric analysis. IEEE Transactions on Visualization and Computer Graphics **20**(9), 1227–1240 (2014)

10. Üffinger, M., Frey, S., Ertl, T.: Interactive high-quality visualization of higher-order finite elements. Computer Graphics Forum **29**(2), 115–136 (2010)
11. Üffinger, M., Sadlo, F., Munz, C.D., Ertl, T.: Toward wall function consistent interpolation of flow fields. In: Short Paper Proceedings of EuroVis 2013. pp. 85–89 (2013)
12. Üffinger, M., Schweitzer, M.A., Sadlo, F., Ertl, T.: Direct visualization of particle-partition of unity data. In: Proceedings of International Workshop on Vision, Modeling and Visualization (VMV). pp. 255–262 (2011)
13. Usher, W., Wald, I., Knoll, A., Papka, M., Pascucci, V.: In situ exploration of particle simulations with CPU ray tracing. Supercomputing Frontiers and Innovations: An International Journal **3**(4), 4–18 (2016)
14. Vetter, Olbrich: Development and integration of an in-situ framework for flow visualization of large-scale, unsteady phenomena in ICON. Supercomputing Frontiers and Innovations: An International Journal **4**(3), 55–67 (2017)