In Situ Monitoring and Steering Deep Learning Training from Numerical Simulations in ParaView-Catalyst

Alejandro Ribes, François Mazen, Lucas Meyer

In Situ Monitoring and Steering Deep Learning Training from Numerical Simulations in ParaView-Catalyst

Alejandro Ribés
EDF Lab Paris-Saclay
Palaiseau, France
alejandro.ribes@edf.fr

François Mazen
Kitware Europe
Lyon, France
francois.mazen@kitware.com

Lucas Meyer
Univ. Grenoble Alpes, Inria, CNRS, LIG
Grenoble, France
lucas.meyer@inria.fr

ABSTRACT

In the context of numerical simulation, a surrogate model approximates the outputs of a solver with a low computational cost. In this article, we present an In Situ visualization prototype, based on Catalyst 2, for monitoring the training of surrogate models based on Deep Neural Networks. We believe that In Situ monitoring can help solve a fundamental problem of this kind of training: standard metrics, such as the Mean Squared Error, do not convey enough information on which simulation aspects are harder to learn. Our prototype allows the interactive visualization of the current state of convergence of a physical quantity spatial field, complementing the traditional loss function value curve. We enable the steering of physical parameters during the training process, for interactive exploration. We also allow the user to influence the learning process in real-time by changing the learning rate. Results are illustrated on a Computational Fluids Dynamics use case.

CCS CONCEPTS

• **Computing methodologies** → *Simulation tools*; *Semi-supervised learning settings*; **Scientific visualization**.

KEYWORDS

In Situ Visualization, Computational Steering, Deep Learning, Surrogate Models, ParaView, Catalyst

ACM Reference Format:
Alejandro Ribés, François Mazen, and Lucas Meyer. 2022. In Situ Monitoring and Steering Deep Learning Training from Numerical Simulations in ParaView-Catalyst. In *ISAV'22 In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV'22), November 13, 2022, Dallas, TX, USA*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3426462.3426468

1 INTRODUCTION

In the context of numerical simulation, a surrogate model approximates the outputs of a solver with a low computational cost. Solvers of differential equations, for instance those based on finite element methods, often require long runs. Thus, they are not well-suited for real-time applications, prediction, or the resolution of inverse problems requiring multiple executions. Surrogates can be deep-learning models that learn from simulation results and/or experimental data. We call these surrogates *Deep Surrogates*. They have recently appeared in the technical literature and are gaining a lot of attention from industry and academics.

Designing and training *Deep Surrogates* can be very challenging for several reasons. The surrogate model has to operate on irregular meshes and spatiotemporal processes. Moreover, surrogates are often applied to parameterized Partial Differential Equations (EDP), which are EDPs that depend on parameters (such as boundary or initial conditions, material properties, and geometric configurations).

In this article, we present an In Situ visualization prototype, based on Catalyst 2, for monitoring the training of *Deep Surrogates*. The aim is to provide a tool that helps in the training of these surrogates. We believe that In Situ monitoring can help solve a fundamental problem of this kind of training: standard metrics, such as Mean Squared Error, do not convey enough information on which aspects of the simulation are harder to learn.

We illustrate the use of our prototype on the so-called Von Karman Vortex Street, which is a popular use case found in computational fluid dynamics (CFD). We train on this use case an existing *Deep Surrogate* that is described in detail in reference [21]. It consists of a deep neural network that takes as input a time step and one physical parameter. The output of the network is a prediction corresponding to the velocity field of the fluid. The surrogate outputs all the values of a discretized field. Training is achieved with backpropagation of the loss function, which corresponds to the mean square error relative to the original simulation. This training process is monitored and steered by the use of Catalyst 2.

The rest of this article is organized as follows. In section 2 we summarize important developments that have been performed by the in situ community. Section 3 describes the two-dimensional Von Karman Vortex Street use case that is used to illustrate our results. Section 4 briefly introduces the *Deep Surrogate* which training is monitored and steered. Section 5 outlines the problems induced by the integral loss functions when used for spatiotemporal and parametric simulations. Section 6 introduces our contribution concerning the monitor and steering of the training of a *Deep Surrogate*. Finally, a short conclusion and a bibliography section end up the article.

2 RELATED WORK

Before dedicated visualization libraries existed, in situ visualization was possible but very cumbersome, examples are the visualization

ISAV'22, November 13, 2022, Dallas, TX, USA

ACM ISBN 978-1-4503-8812-2/20/11…$15.00
https://doi.org/10.1145/3426462.3426468

of large-scale combustion simulations in 2010 [31] or the visualization of a trillion particle simulation in 2012 [9]. This is the reason why finding standardized solutions for the struggling case of large simulations has historically been driving the in situ community.

Catalyst [3] and Libsim [17] are examples of these standardized solutions. These libraries are compiled and linked to the solver and thus access the same computer resources and memory addresses as the simulations (see for instance [25] or [10] for examples concerning computational fluid dynamics). They indeed alleviate the I/O bottleneck but this tightly-coupled paradigm also presents some disadvantages, for instance an error in the visualisation library could propagate and block the solver. Thus the in situ community developed the so-called loosely-coupled paradigm, also commonly called in transit processing, which can be defined (from [16]) as when the simulation transfers data over the network to a separate set of processing or visualization nodes. Reference [22] gives some examples of in transit visualizations.

Another aspect of in situ visualization that has received attention is the visualization pipeline, usually defined in a python script, that needs to be defined prior to the simulation run. This introduces rigidity in the methodology, in the sense that the visualization operations must be a priori defined, preventing the post-hoc exploratory analysis of the simulation results. Several strategies have been proposed to alleviate this problem. One possible solution is the use of steering for interacting with scientific simulations while they are executing [15]. This solution introduces flexibility because the visualization operations can be changed while the simulation is being performed. This approach is implemented in Catalyst and Libsim. Another popular solution is Cinema [1], where in situ visualization tools create an "image database" as a way to save a highly compressed sample of the simulation results. Then, a post-hoc viewer allows the user to browse and interact with the collection of precomputed images, possibly changing some rendering parameters.

In recent past years, in situ visualization became more and more mature. Then authors started to be interested in how to use the in situ paradigm to perform not only visualization but another kind of analysis on the simulation outputs. This requires the development of shared data models that can be used for both simulation and visualization/analysis alike [6]. Efforts to standardize the data models have been performed, important examples are Alpine [19], ADIOS [20], and SENSEI [4]. This naturally leads to the interaction of in situ techniques with computation workflows. Decaf [13], Damaris [12], FlowVR [14] or DataSpace [11] are examples of frameworks designed to support flexible workflows that can combine data processing and visualization, being in situ, in transit or a mix of both.

Also recently, some works of the in situ community have focused on processing online the data produced by multiple-simulations runs. For instance, the Melissa framework for enabling large-scale sensitivity analysis from multiple-simulations runs [30]. In situ techniques appear promising to deal with ensemble-based important applications such as uncertainty quantification, data assimilation, or complex optimization [26].

Connecting CFD with machine learning has received renewed attention with the emergence of deep learning [8, 29]. The goal is often to augment or supplant classical solvers for improved performance in terms of computing speed, error, and resolution. The seminal work of Raissi et al. introduced physics-informed neural networks (PINNs) [24]. This article widely cited article has given rise to many works proposing extensions and improvements to the approach.

Given the importance of meshes in numerical simulation, it is not surprising that mesh-based deep learning techniques have appeared in this context. Dealing with irregular meshes is challenging but can be addressed through some variations of the so-called Graph Neural Network (GNN) formalism [5, 7]. Deep-Mind recently introduced the MeshGraphNets framework for learning mesh-based simulations [23, 28], which uses graph neural networks. We use a *Deep Surrogate* that belongs to this family of methods. It is described in detail in reference [21].

In this article, we present a prototype for ensemble-based Deep-Learning training monitoring, where Catalyst-based In Situ visualization and interactive steering are used.

3 USE CASE: A VON KARMAN VORTEX STREET

The so-called Von Karman Vortex Street is a popular use case found in computational fluid dynamics (CFD). In this article, we present results on a two-dimensional Von Karman Vortex Street, which represents a flow past a circular cylinder. This use case presents the advantage of being compact in mesh size but also complex in its dynamical behavior. Several streams are observable, depending on the Reynolds number, which for this particular case is defined as:

$$\mathfrak{R} = \frac{Ud}{v} \tag{1}$$

where $\mathfrak{R}$ is the Reynolds number, U is the upstream velocity, d is the diameter of the cylinder, and v is the viscosity of the fluid. The Reynolds number is the dimensionless number that determines the stability and the flow type. We note that to a given $\mathfrak{R}$ correspond several combinations of the variables, the diameter, the velocity and the viscosity.

Figure 1 shows an image extracted from a Von Karman simulated using Code-Saturne [2], which is an open-source CFD solver designed to solve the Navier-Stokes equations, with a focus on incompressible or dilatable flows and advanced turbulence modeling. We can observe a circular object inside a 2-dimensional rectangular domain with top and bottom walls. Water is injected on the left and exits the domain on the right, the letter u indicates velocity, U is the input velocity perpendicular to the left boundary of the domain.

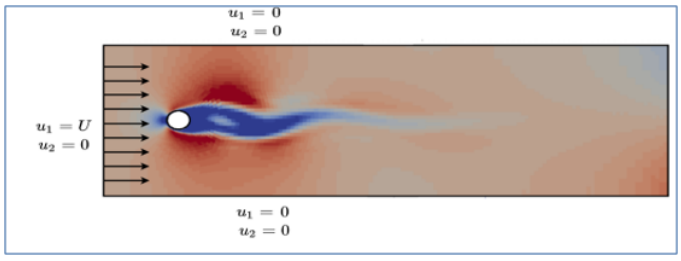

Figure 1: Snapshot extracted from a Von Karman Vortex Street simulated using Code-Saturne [2].

In order to train neural networks on this use case, we used the Code-Saturne solver to generate a dataset of 20 simulations (2GB per simulation). The 20 simulations ran with an irregular mesh of 7361 cells consisting of 15176 points, with an input velocity randomly sampled between 2 and 2.2 (Reynolds numbers between 3000 and 3300). Each simulation runs for 2000 time steps. We remark that this 20 simulations correspond to an ensemble, in the sense of references [26] and [30]. This ensemble is indexed by two parameters, the time ($t \in [0, 2000]$) and the input velocity of the water coming from the right ($U \in [2, 2.2]$).

4 A DEEP LEARNING SURROGATE

A **Surrogate model** approximates the results of a numerical simulation with a low computational cost. In this paper, we use a *Deep Surrogate*, which corresponds to a surrogate model that is built from a neural network. This *Deep Surrogate* is described in detail in reference [21]. For the use case presented in section 3, it consists of a deep neural network that takes as input a time step t, and one physical parameter U. The output of the network is a prediction $\hat{u}$, corresponding to the velocity field of the fluid. A depiction of this *Deep Surrogate* can be seen in Figure 2.

We remark that the surrogate outputs all the values of a discretized field. Thus $\hat{u}$ is an array of size C, where C is in this case the number of cells of the mesh used by the simulation. If we want to visualize $\hat{u}$, this array should be associated with the list of cells of the original mesh.

Figure 2: A deep neural network takes as input a time step t and velocity U. It outputs a prediction $\hat{u}$, corresponding to the discretized velocity field of the fluid.

5 THE PROBLEM: A SCALAR LOSS FUNCTION

As stated in section 4, the training of a *Deep Surrogate* is achieved with backpropagation of the loss $\mathcal{L}$, which in this specific case corresponds to the mean square error relative to the original simulation u: $\mathcal{L} = \|\hat{u} - u\|_2^2$, where $\hat{u}$ is the estimated velocity field of the fluid and u is the reference velocity field. A first important remark about $\mathcal{L}$ is that this loss is going to reduce a spatiotemporal and parametric simulation to a single scalar value. In the use case presented in section 3, physical quantities are associated to the cell centers of the mesh elements. Computing this loss function means adding all cell values in the mesh, for each time step and each value of the upstream velocity. Thus $\mathcal{L}$ corresponds to:

$$\mathcal{L} = \frac{1}{S}\sum_{s=1}^{S}\frac{1}{T}\sum_{t=1}^{T}\frac{1}{C}\sum_{c=1}^{C}\|\hat{u}(s,t,c) - u(s,t,c)\|_2^2 \qquad (2)$$

where S is the number of values of velocity U, T is the number of time steps and C is the number of cells. From section 3, they take values $S = 20$, $T = 2000$ and $C = 7361$.

By looking at equation 2, it is clear that the scalar value $\mathcal{L}$ can represent a very complex underlying behavior of the simulations. Moreover, the same value of $\mathcal{L}$ can be computed from simulation ensembles of very different nature. However, $\mathcal{L}$ is the information used to update the weights of a neural network that builds a *Deep Surrogate*. By now, we have just described a very general problem appearing in learning-based systems when applied to spatiotemporal processes, this problem becomes even more difficult when parameters are added. Training neural networks on such ensembles often fail. As an example, we consider an ensemble of simulations that present near zero cell values in most spatial locations but some non-zero behavior, which is localized in important areas. Learning from such an ensemble will most probably fail because the learning-based system mostly learns from the numerous zero-valued cells. The imbalance of the distribution of values that compose the loss function appears in numerous situations, not only in the case of zero-values cells. In general, the average quadratic error and other metrics, integrated into a spatio-temporal simulation, do not guarantee that the error will be well distributed in time and space.

Most of the time the monitoring of the training of a neural network simply consists of the visualization of a graph showing the value of the loss function. Decisions such as modifying the learning rate or stopping the process are taken by looking at the evolution of this diagram. However, when looking at equation 2, it is evident that a loss function graph is a too simplified view of what happens during the learning.

The work described in this article aims to palliate the just described problem of the loss function imbalance and lack of information. For this, In Situ visualisation of the spatiotemporal and parametric simulations will be performed.

6 MONITORING AND STEERING

As already stated in section 5, the current monitoring tools for the training of neural networks cannot display significant enough informative metrics for the case described in section 3. Thus, we propose a workflow that exploits the scientific visualization capacities of Paraview to better understand the training process. For the von Karman use case, we can qualitatively determine where the training succeeds and where it fails by visually comparing the *groundtruth* (the reference velocity field of the fluid u) to the predictions of the model (the estimated velocity field $\hat{u}$), for fixed values of $velocity$ and $time$.

Figure 3 shows a schematic view of the implemented workflow. It consists of four steps repeated in a loop for each training iteration:

(1) Update the neural network weights and bias.
(2) Run inferences for a specific set of inputs, in this case $velocity$ and $time$.
(3) Send the estimated velocity field $\hat{u}$ (the output of the *Deep Surrogate*) to ParaView, which associates $\hat{u}$ to the simulation mesh and renders it.
(4) Steer the parameters $velocity$, $time$, and the learning rate.

This loop allows the visualization of the current state of convergence for the spatial velocity field $\hat{u}$, as a complement of the traditional loss function value curve. The right side of Figure 3 shows a snapshot of ParaView performing this visualization. However, the spatial visualization only takes care of the space, represented by c in $\hat{u}(s,t,c)$. In order to visualize the remaining s and t, we allow the steering of $velocity$ and $time$, thus the user can interactively

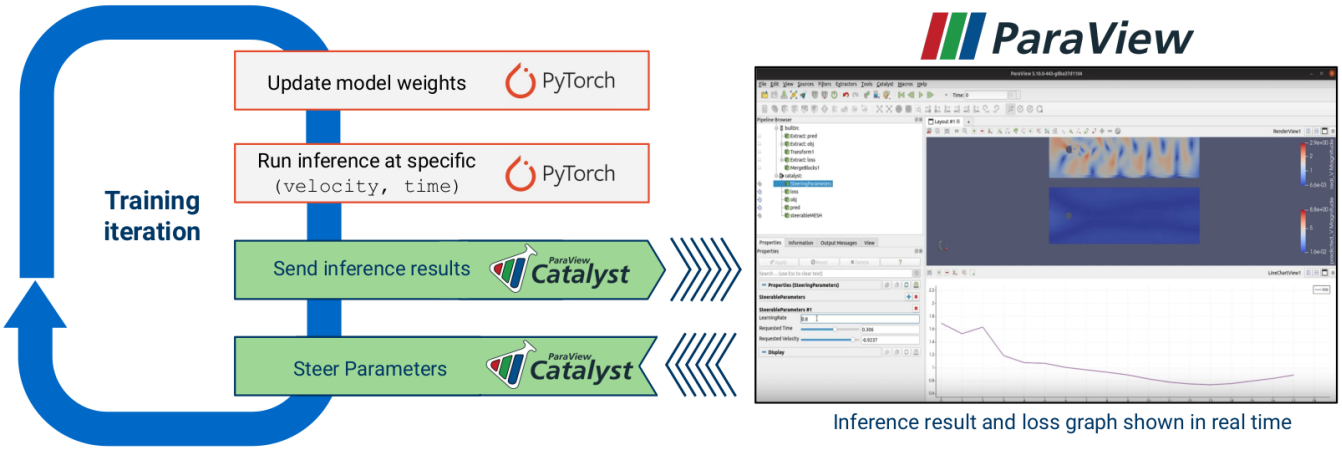

Figure 3: Monitoring and Steering Workflow of the Training.

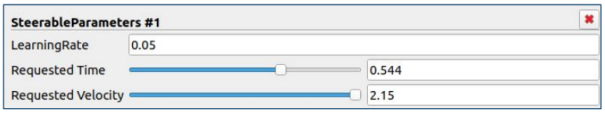

Figure 4: ParaView widget for interactively steering the parameters.

explore the whole ensemble. Furthermore, we also allow the user to influence the learning process in real-time by changing the learning rate. Figure 4 shows the ParaView widget for performing the steering interactively.

The implementation of this pipeline has been performed with Catalyst 2, the last available version of Catalyst [3]. In the context of this article, the aim is to infer a velocity field for a specific time step and the input water velocity of a Von Karman simulation, and then send the result with Catalyst so that ParaView can display it in real-time on a predefined appropriate mesh. During the training of the surrogate model, the user can interactively explore the results of the model inference (using a remote ParaView) or save these results for subsequent analysis (thanks to a disk backup made with Paraview Desktop). The data is sent through a Conduit [18] node in the form of a generic, hierarchical, and human-readable format called *Mesh Blueprint*, without the need for manually generated VTK data structures.

Figure 5 offers a comparison between the results of the reference simulation (*groundtruth*, top) and the inference of the surrogate model during training (bottom). We show four moments of training, which are chronologically ordered from left to right and top to bottom:

(1) During the first training iterations not much has been learned by the model.
(2) After a few epochs, the model understands that a seemingly continuous trail is forming behind the obstacle.
(3) Training continues, and the trail fades out as vortices start forming. The vortices at the back, farther away from the obstacle, are easier for the model to learn.
(4) After enough time, the model has learned each vortex and can predict the result of the simulation with high accuracy.

These visualizations make it possible to qualitatively characterize the convergence of the model. We observe that the neural network converges spatially in a non-uniform way towards the reference solution.

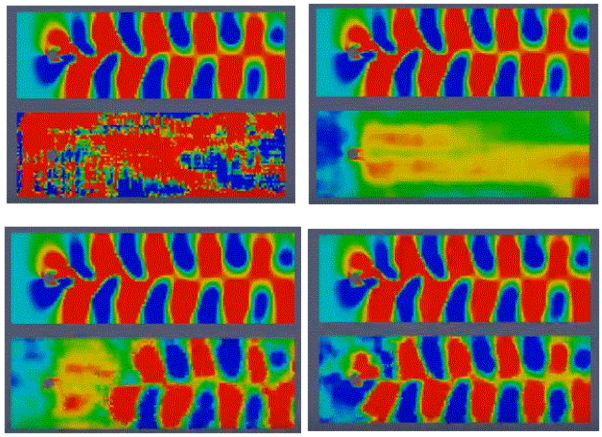

Figure 5: Evolution of the training. Each image presents a comparison between the results of the reference simulation (groundtruth, top) and the inference of the surrogate model during training (bottom). We show four moments of the training, which are chronologically ordered from left to right and top to bottom.

7 CONCLUSION

The prototype that has been described in this short article is our first attempt to monitor the training of Deep Neural Networks from an ensemble of simulations. We consider this preliminary work as a proof of concept. We plan to extend and test the presented

approach. A first step will be to evaluate it on other use cases of different complexity and size.

We believe that we have proved the added value of monitoring and steering the training process of a *Deep Surrogate*. Currently, the monitoring of this kind of training mostly consists of the visualization of a graph showing the scalar value of the loss function. Decisions such as modifying the learning rate or stopping the process are taken by looking at the evolution of this diagram. Using ParaView and Catalyst we can visualize the current state of convergence of a physical quantity spatial field, as a complement of the traditional loss function value curve.

We allow the steering of physical parameters (in this case $velocity$ and $time$), thus the user can interactively explore the whole ensemble. Furthermore, we also allow the user to influence the learning process in real-time by changing the learning rate.

In summary, we believe that In Situ monitoring and steering can help solve a fundamental problem concerning the training of a *Deep Surrogate*: standard metrics, such as the Mean Squared Error, do not convey enough information on which aspects of the simulation are harder to learn.

We expect, in the near future, to integrate this work into the SALOME Open-Source numerical simulation platform (www.salome-platform.org) [27].

ACKNOWLEDGMENTS

This work has been partly funded by the European Union's Horizon 2020 project REGALE (Open Architecture for Exascale Supercomputers) under grant agreement No 956560. It has also been partly funded by SINCLAIR (Saclay INdustrial Collaborative Laboratory for Artificial Intelligence Research) and Kitware Europe.

REFERENCES

[1] J. Ahrens, S. Jourdain, P. OLeary, J. Patchett, D. H. Rogers, and M. Petersen. 2014. An Image-Based Approach to Extreme Scale in Situ Visualization and Analysis. In *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 424–434.

[2] Frédéric Archambeau, Namane Méchitoua, and Marc Sakiz. 2004. Code Saturne: A finite volume code for the computation of turbulent incompressible flows-Industrial applications. *International Journal on Finite Volumes* (2004).

[3] Utkarsh Ayachit, Andrew Bauer, Berk Geveci, Patrick O'Leary, Kenneth Moreland, Nathan Fabian, and Jeffrey Mauldin. 2015. Paraview catalyst: Enabling in situ data analysis and visualization. In *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*. 25–29.

[4] U. Ayachit, B. Whitlock, M. Wolf, B. Loring, B. Geveci, D. Lonie, and E. W. Bethel. 2016. The SENSEI Generic In Situ Interface. In *2016 Second Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV)*. 40–44.

[5] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261* (2018).

[6] Janine C. Bennett, Hank Childs, Christoph Garth, and Bernd Hentschel. 2019. In Situ Visualization for Computational Science (Dagstuhl Seminar 18271). *Dagstuhl Reports* 8, 7 (2019), 1–43. http://drops.dagstuhl.de/opus/volltexte/2019/10171

[7] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 18–42.

[8] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. 2020. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics* 52 (2020), 477–508.

[9] Surendra Byna, Jerry Chou, Oliver Rubel, Homa Karimabadi, William S Daughter, Vadim Roytershteyn, E Wes Bethel, Mark Howison, Ke-Jou Hsu, Kuan-Wu Lin, et al. 2012. Parallel I/O, analysis, and visualization of a trillion particle simulation. In *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–12.

[10] Jose J Camata, Vitor Silva, Patrick Valduriez, Marta Mattoso, and Alvaro LGA Coutinho. 2018. In situ visualization and data analysis for turbidity currents simulation. *Computers & Geosciences* 110 (2018), 23–31.

[11] Ciprian Docan, Manish Parashar, and Scott Klasky. 2012. DataSpaces: an Interaction and Coordination Framework for Coupled Simulation Workflows. *Cluster Computing* 15, 2 (2012), 163–181.

[12] Matthieu Dorier, Gabriel Antoniu, Franck Cappello, Marc Snir, and Leigh Orf. 2012. Damaris: How to efficiently leverage multicore parallelism to achieve scalable, jitter-free I/O. In *2012 IEEE International Conference on Cluster Computing*. IEEE, 155–163.

[13] Matthieu Dreher and Tom Peterka. 2017. *Decaf: Decoupled dataflows for in situ high-performance workflows*. Technical Report. Argonne National Lab.(ANL), Argonne, IL (United States).

[14] Matthieu Dreher and Bruno Raffin. 2014. A flexible framework for asynchronous in situ and in transit analytics for scientific simulations. In *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE, 277–286.

[15] James A Kohl, Torsten Wilde, and David E Bernholdt. 2006. Cumulvs: Interacting with high-performance scientific simulations, for visualization, steering and fault tolerance. *The International Journal of High Performance Computing Applications* 20, 2 (2006), 255–285.

[16] James Kress, Scott Klasky, Norbert Podhorszki, Jong Choi, Hank Childs, and David Pugmire. 2015. Loosely coupled in situ visualization: A perspective on why it's here to stay. In *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*. 1–6.

[17] T Kuhlen, R Pajarola, and K Zhou. 2011. Parallel in situ coupling of simulation with a fully featured visualization system. In *Proceedings of the 11th Eurographics Conference on Parallel Graphics and Visualization (EGPGV)*, Vol. 10. Eurographics Association Aire-la-Ville, Switzerland, 101–109.

[18] Lawrence Livermore National Laboratory. 2017. Conduit: Simplified Data Exchange for HPC Simulations. https://software.llnl.gov/conduit/. Accessed: 2022-07.

[19] Matthew Larsen, James Ahrens, Utkarsh Ayachit, Eric Brugger, Hank Childs, Berk Geveci, and Cyrus Harrison. 2017. The ALPINE In Situ Infrastructure: Ascending from the Ashes of Strawman. In *Proceedings of the In Situ Infrastructures on Enabling Extreme-Scale Analysis and Visualization* (Denver, CO, USA) *(ISAV'17)*. Association for Computing Machinery, New York, NY, USA, 42–46.

[20] Jay F. Lofstead, Scott Klasky, Karsten Schwan, Norbert Podhorszki, and Chen Jin. 2008. Flexible IO and Integration for Scientific Codes through the Adaptable IO System (ADIOS). In *Proceedings of the 6th International Workshop on Challenges of Large Applications in Distributed Environments* (Boston, MA, USA) *(CLADE '08)*. Association for Computing Machinery, New York, NY, USA, 15–24.

[21] Lucas Meyer, Louen Pottier, Alejandro Ribes, and Bruno Raffin. 2021. Deep Surrogate for Direct Time Fluid Dynamics. In *NeurIPS 2021 - Thirty-fifth Workshop on Machine Learning and the Physical Sciences*. Vancouver, Canada, 1–7. https://hal.archives-ouvertes.fr/hal-03451432

[22] Kenneth Moreland, Ron Oldfield, Pat Marion, Sebastien Jourdain, Norbert Podhorszki, Venkatram Vishwanath, Nathan Fabian, Ciprian Docan, Manish Parashar, Mark Hereld, et al. 2011. Examples of in transit visualization. In *Proceedings of the 2nd international workshop on Petascal data analytics: challenges and opportunities*. 1–6.

[23] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. 2020. Learning Mesh-Based Simulation with Graph Networks. In *International Conference on Learning Representations*.

[24] M. Raissi, P. Perdikaris, and G.E. Karniadakis. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* 378 (Feb. 2019), 686–707. https://doi.org/10.1016/j.jcp.2018.10.045

[25] Alejandro Ribés, Benjamin Lorendeau, Julien Jomier, and Yvan Fournier. 2015. In-situ visualization in computational fluid dynamics using open-source tools: integration of catalyst into Code_Saturne. In *Topological and Statistical Methods for Complex Data*. Springer, 21–37.

[26] Alejandro Ribés and Bruno Raffin. 2020. The Challenges of In Situ Analysis for Multiple Simulations. In *ISAV'20 In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization* (Atlanta, GA, USA) *(ISAV'20)*. Association for Computing Machinery, New York, NY, USA, 32–37. https://doi.org/10.1145/3426462.3426468

[27] Alejandro Ribés and Adrien Bruneton. 2014. Visualizing results in the SALOME platform for large numerical simulations: An integration of ParaView. In *2014 IEEE 4th Symposium on Large Data Analysis and Visualization (LDAV)*. 119–120. https://doi.org/10.1109/LDAV.2014.7013218

[28] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. 2020. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*. PMLR, 8459–8468.

[29] Rick Stevens, Valerie Taylor, Jeff Nichols, Arthur Barney Maccabe, Katherine Yelick, and David Brown. 2020. *AI for Science*. Technical Report. Argonne National Lab.(ANL), Argonne, IL (United States).

[30] Théophile Terraz, Alejandro Ribés, Yvan Fournier, Bertrand Iooss, and Bruno Raffin. 2017. Melissa: Large Scale In Transit Sensitivity Analysis Avoiding Intermediate Files. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC'17)*. Denver.

[31] H. Yu, C. Wang, R. W. Grout, J. H. Chen, and K. Ma. 2010. In Situ Visualization for Large-Scale Combustion Simulations. *IEEE Computer Graphics and Applications* 30, 3 (2010), 45–57.