Appendix

Available dataset and code The code corresponding to the two-dimensional solid mechanics case (`Tensile2d`) described in Section 4.1 is available at `https://gitlab.com/drti/mmgp` [3]. A documentation is available at `https://mmgp.readthedocs.io/` [2], where details are provided on how to download the dataset `Tensile2d` and reproduce the corresponding numerical experiments.

Details regarding the datasets are provided in Appendix A. Morphing strategies and dimensionality reduction techniques are described in Appendices B and C. Details about the GNNs baselines are given in Appendix D. Finally, additional results about the considered experiments are gathered in Appendix E.

A Datasets

This section provides additional details regarding the synthetic datasets `Tensile2d` and `Rotor37`. Regarding the `AirfRANS` dataset, the reader is referred to [14].

A.1 Rotor37 dataset

Examples of input geometries are shown in Figure 6 together with the associated output pressure fields. While the geometrical variabilities are moderate, it can be seen that they have a significant impact on the output pressure field. A design of experiment for the input parameters of this problem are generated with maximum projection LHS method [43]. For each input mesh and set of input parameters, a three-dimensional aerodynamics problem is solved with RANS, as illustrated in, *e.g.* [8]. The output scalars of the problem are obtained by post-processing the three-dimensional velocity.

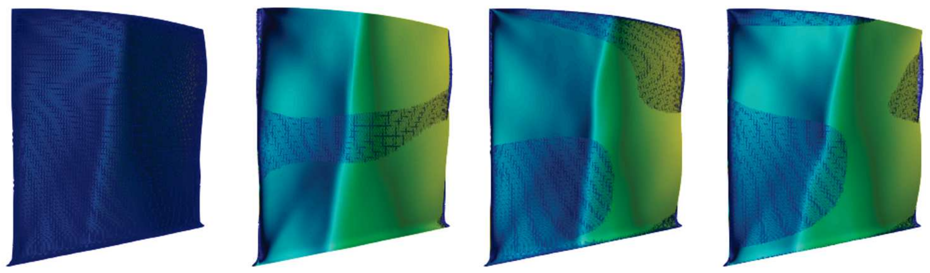

Figure 6: (`Rotor37`) Four geometries with their corresponding output pressure fields. The first panel shows the mesh, and the second to last panels show a superposition of the corresponding geometry and the mesh of the first one.

A.2 Tensile2d dataset

Examples of input geometries are shown in Figure 7. A two-dimensional boundary value problem in solid mechanics is considered, under the assumption of small perturbations (see, *e.g.* [12]). The partial differential equation is supplemented with Dirichlet and Neumann boundary conditions: the displacement on the lower boundary is fixed, while a uniform pressure is applied at the top. The input parameters of the problem are chosen to be the magnitude of the applied pressure and 5 parameters involved in the elasto-visco-plastic constitutive law of the material [50]. The outputs of the problem are chosen as the components of the displacement field, u and v, the entries of the Cauchy stress tensor, σ_{11}, σ_{22}, σ_{12}, and the cumulative plastic strain p. We also consider 4 output scalars obtained by post-processing the fields of interest: the maximum plastic strain $p_{\max}$ accross the geometry, the maximum vertical displacement $v_{\max}$ at the top of the geometry, and the maximum normal stress $\sigma_{22}^{\max}$ and Von Mises stress $\sigma_v^{\max}$ accross the geometry. It is worth emphasizing that the cumulative plastic strain p is challenging to predict, as illustrated in Section E.

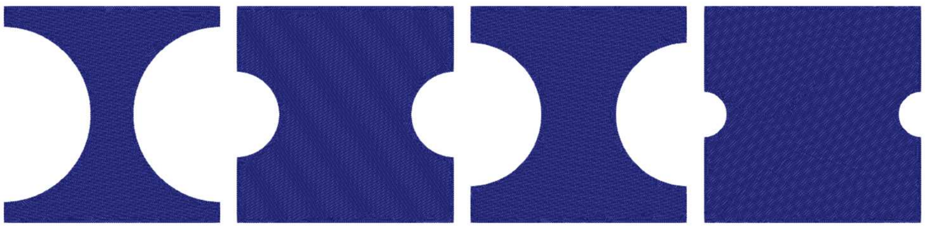

Figure 7: (`Tensile2d`) Illustration of the four input meshes that are used in Figure 5.

B Morphing strategies

In this section, we briefly describe the Tutte's barycentric mapping [71] and the radial basis function morphing [9, 21] used in the considered experiments.

Tutte's barycentric mapping. For this method, we are limited to connected triangular surface meshes of fixed topology, either in a 2D or in a 3D ambient space. Tutte's barycentric mapping starts by setting the value of the displacement of the boundary points of the mesh (usually onto the unit disk), and solve for the value at all the remaining nodes of the mesh. The physical features available on the mesh, and inherited from the problem, are used in the specification of the displacement of the boundary nodes.

We recall that $\mathbf{x}_I$, $I = 1 \dots N$, denote the mesh nodes coordinates. We assume that the numbering of the nodes starts with the interior points of the mesh $1 \dots N_{\text{int}}$, and ends with the N_b nodes on its boundary $N_{\text{int}} + 1 \dots N$. The morphed mesh node coordinates are denoted by $\overline{\mathbf{x}}_I$, $I = 1 \dots N$. The coordinates of the boundary of the morphed mesh being known, we denote $\overline{\mathbf{x}}_{b_I} = \overline{\mathbf{x}}_{I+N_{\text{int}}}$, $I = 1 \dots N_b$. Then the following sparse linear system is solved for the morphing of the interior points:

$$\overline{\mathbf{x}}_I - \frac{1}{d(I)} \sum_{J \in \mathcal{N}(I) \cap [\![1, N_{\text{int}}]\!]} \overline{\mathbf{x}}_J = -\frac{1}{d(I)} \sum_{J \in \mathcal{N}(I) \cap [\![N_{\text{int}}, N]\!]} \overline{\mathbf{x}}_{b_{J-N_{\text{int}}}},$$

where $\mathcal{N}(I)$ and $d(I)$ are respectively the neighbors and the number of neighbors of the node I in the graph (or in the mesh following its connectivity).

In the 2D solid mechanics case `Tensile2d`, we know the rank of the point separating the left and the bottom faces which we map onto the point $(0, 1)$ of the target unit disk. The linear density of nodes on the boundary of the target unit disk is chosen to be the same as the one of the mesh sample (relative to the length of the boundary), see Figure 8 for an illustration.

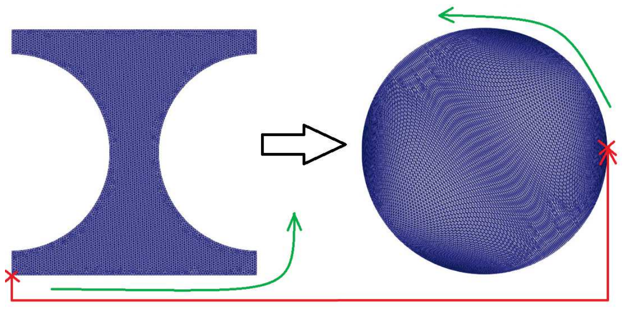

Figure 8: (`Tensile2d`) Illustration of the Tutte's barycentric mapping used in the morphing stage.

From [27, corollary 2], the morphing described above is called a parametrization, and defines an isomorphic deterministic transformation of the considered triangular surface mesh $\mathcal{M}$, into a plane triangular mesh $\overline{\mathcal{M}}$ of the unit disk. Notice that although these morphing techniques are called "mesh parametrization", this do not mean that we need to know any parametrization of the shape: these are deterministic transformations of the meshes, requiring no other information than the nodes locations and the triangles connectivities.

This method is taken from the computer graphics community and has been improved over the years. In [78], a quality indicator called stretch metric is optimized during an iterative procedure, to obtain more regular morphed mesh. Recently in [31], a procedure was proposed to drastically improve mesh parametrization, even in difficult cases where some triangles are overlapping. It should be noted that such iterative procedures come with the additional cost of solving a series of sparse linear systems.

Radial Basis Function morphing. In the same fashion as Tutte's barycentric mapping, RBF morphing methods start by setting the value of the displacement at some particular nodes of the mesh (here the boundary points of the mesh, but interior points can be considered as well with RBF), and solve for the location at all the remaining nodes of the mesh. The physical features available on the mesh are also used in the specification of the displacement of the boundary points. RBF morphing methods are compatible with 2D and 3D structured and unstructured meshes, do not require any mesh connectivity information, and can be easily implemented in parallel for partitioned meshes.

We use the RBF morphing method as proposed in [21]. Once the mapping for the N_b boundary points of ranks $N_{\text{int}} + 1 \ldots N$ is fixed, then the interior points $1 \leq I \leq N_{\text{int}}$ are mapped such as

$$\overline{\mathbf{x}}_I = \sum_{J=1}^{N_b} \alpha_J \phi(\|\mathbf{x}_I - \mathbf{x}_{b_J}\|), \quad 1 \leq I \leq N_{\text{int}},$$

where ϕ is a radial basis function with compact support and α_J are determined by the interpolation conditions. More precisely, we choose the radial basis function with compact support $\phi(\xi) = (1-\xi)^4 (4\xi + 1)$ and support radius equal to half of the mesh diameter, and interpolation conditions means that the morphing is known at the boundary points:

$$\mathbf{M}_{\text{RBF}} \alpha = \overline{\mathbf{x}}_b,$$

where $\mathbf{M}_{\text{RBF}_{I,J}} = \phi(\|\mathbf{x}_{b_I} - \mathbf{x}_{b_J}\|)$, $1 \leq I, J \leq N_b$.

For the `AirfRANS` dataset, we make use of the physical properties of the boundary condition to morph each mesh onto the first geometry of the training set. Referring to Figure 9 (bottom), we know which nodes lie the external boundary (in black), airfoil extrado (in red), airfoil intrado (in blue) and which nodes define the leading and trailing points (green crosses). We choose to keep the points at the external boundary fixed (zero mapping), map the leading and trailing edge to the ones of the mesh of the first training sample, and map the points on the extrado and intrado along the ones of the mesh of the first training sample while conserving local node density (relative to the length of the boundary). A zoom of the RBF morphing close to the airfoil for test sample 787 is illustrated in Figure 10.

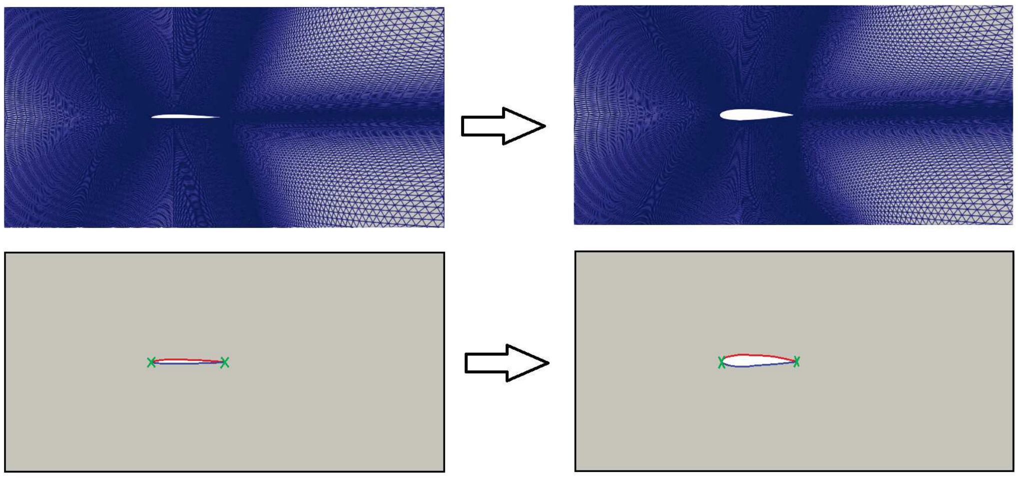

Figure 9: (`AirfRANS`) RBF morphing for test sample 787; (top) complete mesh morphing, (bottom) illustration of the mapping of the boundary points.

Notice that while Tutte's barycentric mapping requires solving a sparse linear system of rather large size N_{int}, RBF morphing requires solving a dense linear system of smaller size N_b. RBF morphing methods dealing with complex non-homogeneous domains have been proposed in [15].

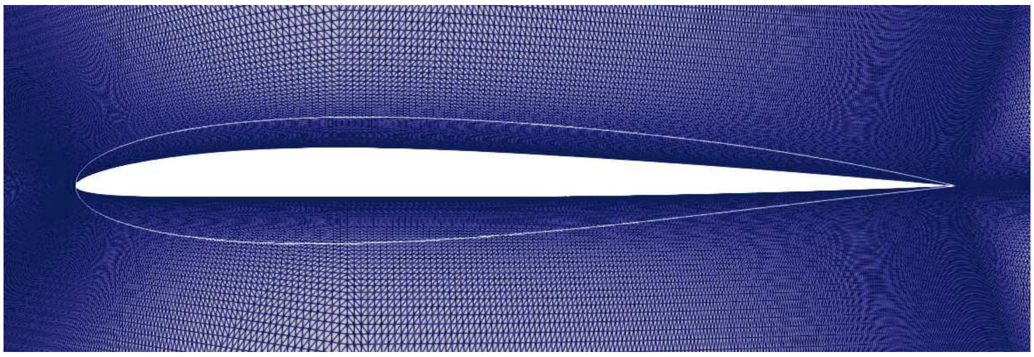
Figure 10: (`AirfRANS`) Zoom of the RBF morphing close to the airfoil for test sample 787.

Other methods. In [23], the morphing is computed by means of solving an elastic problem. See also [6, 70] for literature reviews on mesh morphing methods. Mesh deformation algorithms compatible with topology changes have been proposed [80].

C Dimensionality reduction

The principal component analysis can be replaced by more effective dimensionality reduction techniques such as the snapshot-POD. The latter is a variant where the underlying ℓ^2-scalar product used to compute the coefficients of the empirical covariance matrices is replaced by the $L^2(\overline{\mathcal{M}}^c)$-inner product. Define the symmetric positive-definite matrix $\mathbf{M} \in \mathbb{R}^{N_c \times N_c}$, such that

$$M_{IJ} = \int_{\overline{\mathcal{M}}^c} \overline{\varphi}^c_I(\overline{\mathbf{x}}) \overline{\varphi}^c_J(\overline{\mathbf{x}}) d\overline{\mathbf{x}} \,.$$

In general, a quadrature formula, in the form of a weighted sum over function evaluations on the common mesh, is chosen such that the integral are computed exactly for functions in the span of the finite element basis. Then, the empirical covariance matrix is computed as

$$\left((\widetilde{\mathbf{U}}^i_k)^T \mathbf{M} \widetilde{\mathbf{U}}^j_k \right)_{i,j} = \sum_{I,J=1}^{N_c} \int_{\overline{\mathcal{M}}^c} \overline{\mathcal{U}}^i_k(\overline{\mathbf{x}}^c_I) \overline{\varphi}^c_I(\overline{\mathbf{x}}) \overline{\mathcal{U}}^j_k(\overline{\mathbf{x}}^c_J) \overline{\varphi}^c_J(\overline{\mathbf{x}}) d\overline{\mathbf{x}} = \int_{\overline{\mathcal{M}}^c} P(\overline{\mathcal{U}}^i_k)(\overline{\mathbf{x}}) P(\overline{\mathcal{U}}^j_k)(\overline{\mathbf{x}}) d\overline{\mathbf{x}} \,,$$

which corresponds to the continuous formula for the computation of the correlations of the fields of interest transported and interpolated on the common morphed mesh. Hence, the empirical covariance matrix can take into account any heterogeneity of the common morphed mesh, which may occur after morphing. The same construction can be made for the spatial coordinate field, while its derivation is more technical, because it involves vector fields instead of scalar fields. The computation of the empirical covariance matrix can be easily be parallelized on numerous computer nodes, provided that the common morphed mesh has been partitioned in subdomains, which enable efficient dimensionality reduction for meshes up to millions of degrees of freedom, see [20].

Other linear or nonlinear dimension reduction techniques can be considered, like mRMR feature selection [22, 62], kernel-PCA [68] or neural network-based autoencoders [48].

D Architectures and hyperparameters of GNN-based baselines

D.1 Graph convolutional neural network

A graph convolutional neural network (GCNN) [67] has been implemented using `PyTorch Geometric` [26] with the Graph U-Net [29] architecture and the following specifications: (i) top_k pooling [29, 47] layers with a pooling ratio of 0.5 to progressively aggregate information over nodes of the graph, (ii) feature sizes progressively increased after each top_k pooling, *i.e.*, 16, 32, 64, 96 and 128, (iii) between each pooling, residual convolution blocks [40] are added to combine two consecutive normalization-activation-convolution layers, (iv) BatchNorm [42] layers are introduced, and (v) LeakyReLU [57] activations are used with slope of 0.1 on negative values.

A weighted multi-loss $\mathcal{L}$ that combines scalars and fields is used, and defined as

$$\mathcal{L}\left(\left(\mathbf{U},\mathbf{w}\right),\left(\mathbf{U}',\mathbf{w}'\right)\right)=\lambda_{\mathrm{scalars}}\mathcal{L}_{\mathrm{MSE}}\left(\mathbf{w},\mathbf{w}'\right)+\lambda_{\mathrm{fields}}\sum_{k=1}^{d}\mathcal{L}_{\mathrm{MSE}}\left(\mathbf{U}_k,\mathbf{U}'_k\right),$$

where $\lambda_{\mathrm{scalars}}$ and $\lambda_{\mathrm{fields}}$ are two positive hyperparameters. For gradient descent, an Adam optimizer [45] is used with a cosine-annealing learning rate scheduler [55]. The following hyperparameters are optimized by grid search: (i) the learning rate, 13 values between 1.0 and 0.0001, (ii) the weight $\lambda_{\mathrm{field}}\in\{1,10,100,1000\}$, and (iv) the type of convolution, chosen between *GATConv* [72], *GeneralConv* [79], *ResGatedGraphConv* [16] and *SGConv* [77]. There are many other hyperparameters that could be tuned, as the number of layers or the number of features on each layer. The chosen hyperparameters are summarized in Table 4 for each experiment. In the case of the `Rotor37` problem, the outwards normals to the surface of the compressor blade are added as input features to input graphs. Similarly, for the `Tensile2d` and `AirfRANS` problems, the signed distance function is added as an input feature.

Table 4: Chosen hyperparameters for the GCNN architectures.

Dataset	Learning rate	λ_{field}	Convolution
`Rotor37`	0.02	10.0	`GeneralConv`
`Tensile2d`	0.01	100.0	`GeneralConv`
`AirfRANS`	0.005	10.0	`GeneralConv`

D.2 MeshGraphNets

The MGN model [63] is taken from Nvidia's `Modulus` [4] package that implements various deep surrogate models for physics-based simulations. The same set of hyperparameters is used for all the considered regression problems, which is chosen after conducting a grid search over the learning rate, the number of hidden nodes and edges features, the number of processor steps. The learning rate is set to 0.001, the numbers of hidden features `hidden_dim_node_encoder`, `hidden_dim_edge_encoder`, and `hidden_dim_node_decoder` are all set to 16. The number of processor steps is chosen as 10. The rest of the MGNs hyperparameters are left to the default values used in the `Modulus` package. The batch size is set to 1, the activation is chosen as the LeakyReLU activation with a 0.05 slope, and $1,000$ epochs are performed for training the network. For scalar outputs, a readout layer taken from [46] is added to the model. The input nodes features are given by the spatial coordinates of the nodes, and possible additional fields such as the signed distance function (for the `Tensile2d` and `AirfRANS` problems), or the outward normals (for the `Rotor37` problem). Given two node coordinates $\mathbf{x}_i$ and $\mathbf{x}_j$, the edges features are chosen as $\exp(-\|\mathbf{x}_i-\mathbf{x}_j\|_2^2/(2h^2))$, where h denotes the median of the edge lengths in the mesh.

For each considered regression problem, it is found that it is more effective to train two MGNs models, one dedicated to handling output fields and the other specialized for output scalars, respectively. Nevertheless, better hyperparameter tuning and more effective readout layers could lead to different conclusions regarding this matter.

D.3 Training on `AirfRANS`

As mentioned in Section 4.1, training GNNs on the `AirfRANS` dataset is computationally expensive due to the sizes of the input meshes. For this reason, the GNN-based baselines are trained on the `AirfRANS-remeshed` dataset (see Table 1) obtained by coarsening the input meshes (see Figure 11) and the associated output fields. Once trained, predictions on the initial fine meshes are obtained through finite element interpolation. It should be underlined that this strategy may hinder the performance of the GNN-based baselines, as the reconstructed fields are obtained by finite element interpolation.

E Additional results

This section gathers additional results about the experiments considered in Section 4.

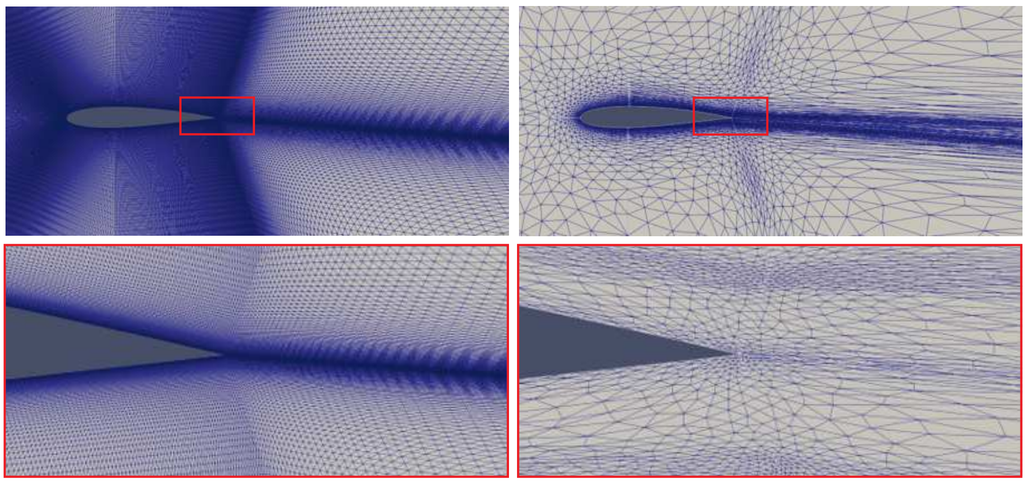

Figure 11: (AirfRANS) Example of an original mesh from the dataset (left) and the corresponding coarsened mesh in the AirfRANS-remeshed dataset (right).

E.1 Out-of-distribution inputs

Figures 12, 13, and 14 show histograms of the logarithm of the predictive variance for the output scalars of interest on different sets of samples. The aim is to empirically assess if the MMGP model is able to identify out-of-distribution (OOD) inputs by attributing higher predictive variances. In the case of the Rotor37 problem, three OODs samples are generated such that the support of the covariates (μ_1, μ_2, and $\mathcal{M}$) are disjoint with the support of the training distributions. It can be seen that the variances of the OOD samples are higher than the ones of the in-distribution samples. Similar observations are made for the Tensile2d and AirfRANS problems. While such an analysis can help to identify OOD inputs, it should be underlined that the predictive uncertainties of Gaussian processes are only valid under the Gaussian a priori assumption, which may not be verified in practice. For instance, the ellipsoid geometry has a similar variance as the in-distribution samples in Figure 13.

E.2 Predicted output fields

Tensile2d dataset. For reproducibility matters, we mention that for the field p and the scalar $p_{\max}$, the denominators in the formulae RRMSE_f and RRMSE_s has been replaced by 1 when its value is below $1e-6$ for preventing division by zero, which corresponds to replacing the relative error by the absolute error for samples that do not feature plastic behaviors.

In Figures 15-18, we illustrate the MMGP prediction, variance and relative error for all the considered fields: u, v, p (evrcum), σ_{11}, σ_{12} and σ_{22}, for respectively the first training inputs, first test inputs, and two out-of-distribution geometries (ellipsoid and wedge). In particular, the wedge cut-off geometry features stress concentrations that are not present in the training set. We notice that the predictions for selected train and test inputs (Figures 15 and 16) are accurate, with small relative errors and relatively small predictive variances, except for some small areas where the considered fields have larger magnitudes. As expected, the predictions for the ellipsoid and wedge cases (Figures 17 and 18) are less accurate than for in-distribution shapes, but the predictive variances are larger, which confirms that MMGP informs that, locally, the prediction cannot be trusted. This phenomenon is particularly strong for the wedge case, that largely differs from the training set shapes.

In Figures 19 and 20, we consider all the output interest fields. The 2D domain is visualized in 3D, in the form of three surfaces: a transparent blue for the 0.025-quantile, a white for the reference prediction and a transparent red for the 0.975-quantile. The point-wise 95% confidence interval is the distance (along the out-of-plane axis) between the transparent blue and red surfaces. We notice that the 95% confidence intervals are very small for the train and test inputs, larger for the ellipsoid case, and much larger for the wedge case (in particular σ_{22}). Not surprisingly, for the OOD shapes

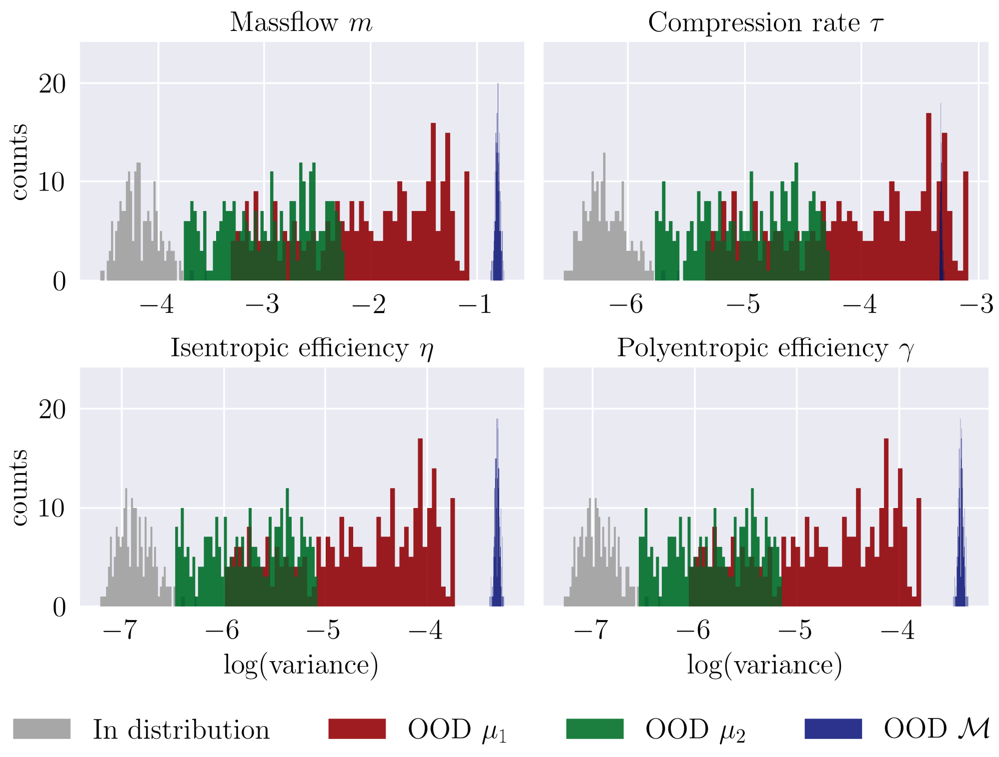

Figure 12: (`Rotor37`) Histograms of log(variance) of MMGP predictions for the output scalars of interest on four sets: in grey the testing set (in distribution), in green and red respectively two sets where the input pressure μ_1 and rotation speed μ_2 are taken OOD, and in blue a set of geometry taken OOD.

ellipsoid and wedge, some surfaces intersect, meaning that, locally, the reference solution is not inside the 95% confidence interval.

In Figure 21, we illustrate the finite element error occurring when predicting σ_{11} with respect to the 95% confidence interval for samples taken from the training and testing sets. We notice that on the training set, the finite element error magnitude is comparable to the 95% confidence interval, which is very small on training samples. On the testing set, the 95% confidence interval is larger, and the finite element error magnitude can be neglected.

AirfRANS dataset. Figures 22 and 23 illustrate reference, MMGP prediction and relative errors for the fields of interest u, v and p on respectively test sample 430 and train sample 93. In the first row, zooms are provided close to the trailing edge to illustrate the accuracy of the prediction in the thin boundary layer. Relative errors have larger magnitudes on spatially restricted areas. We notice that on train sample 93, the areas with low relative error are larger than for test sample 430.

In Table 5, we compare MMGP and our trained GCNN and MGN, as well as the four models trained in [14], for the scalars of interest drag coefficient C_D and lift coefficient C_L, computed by post-processing the predicted fields instead of directly predicting them as output scalars. This post-processing consists in integrating the reference and predicted wall shear stress (from the velocity) and pressure fields around the surface of the airfoil. The models from [14] are a MLP (a classical Multi-Layer Perceptron), a GraphSAGE [38], a PointNet [64] and a Graph U-Net [29] and the corresponding results are taken from [14, Table 19] ("full dataset" setting that we considered in this work). Refer to [14, appendix L] for a description of the used architecture. The limits of this comparison are (i) the mesh supporting the fields are not the same (they have been coarsen in [14] by process different from ours), (ii) the scalar integration routine are not identical (we integrate using

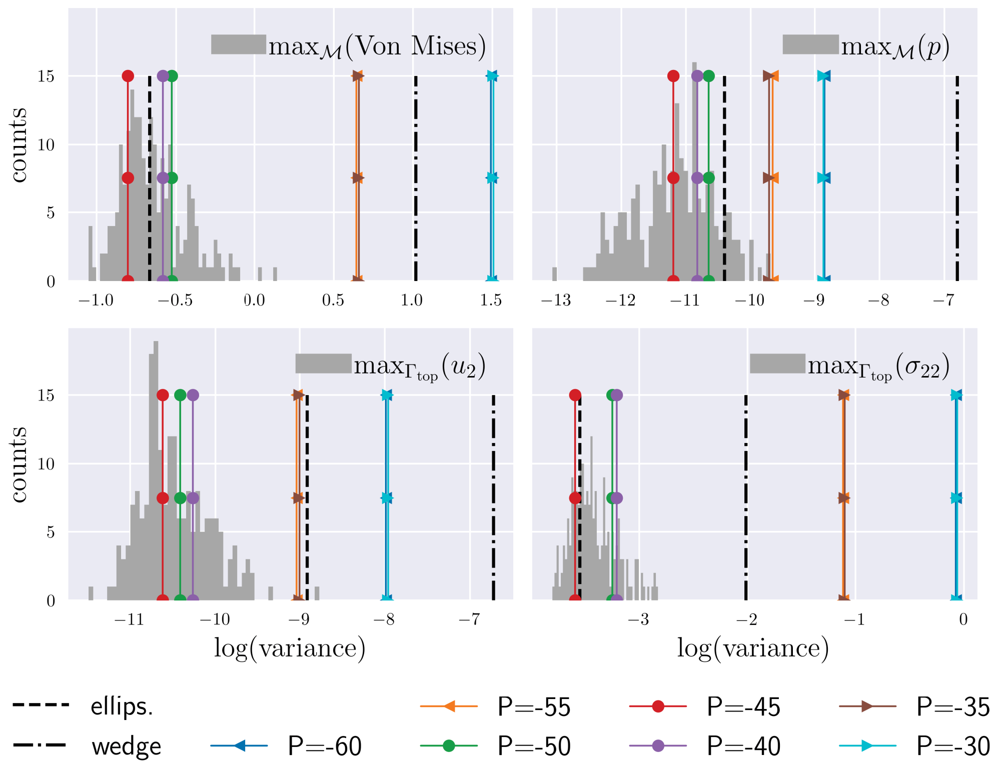

Figure 13: (`Tensile2d`) Histograms of log(variance) of MMGP predictions in grey for the output scalars of interest on the testing set (in distribution). The variance of the MMGP prediction is identified for various configurations: the ellipsoid and wedge cases (where all the nongeometrical parameters are taken at the center of the training intervals), and 5 settings where the same geometry is taken in the testing set and the input pressure varies (the training interval is $[-50, -40]$).

finite element representations). Within these limits, MMGP appears competitive with respects to the models of [14] and our trained GCNN and MGN.

Table 5: (`AirfRANS`) Relative errors (Spearman's rank correlation) for the predicted drag coefficient C_D (ρ_D) and lift coefficient C_L (ρ_D) for the four models of [14, Table 19], as well as GCNN, MGN and MMGP. These scalars of interest are computed as a postprocessing of the predicted fields (best is **bold**).

Model	Relative error		Spearman's correlation	
	C_D	C_L	ρ_D	ρ_L
MLP	6.2e+0 (9e-1)	2.1e-1 (3e-2)	0.25 (9e-2)	0.9932 (2e-3)
GraphSAGE	7.4e+0 (1e+0)	1.5e-1 (3e-2)	0.19 (7e-2)	0.9964 (7e-4)
PointNet	1.7e+1 (1e+0)	2.0e-1 (3e-2)	0.07 (6e-2)	0.9919 (2e-3)
Graph U-Net	1.3e+1 (9e-1)	1.7e-1 (2e-2)	0.09 (5e-2)	0.9949 (1e-3)
GCNN	3.6e+0 (7e-1)	2.5e-1 (4e-2)	0.002 (2e-1)	0.9773 (4e-3)
MGN	3.3e+0 (6e-1)	2.6e-1 (8e-2)	0.04 (3e-1)	0.9761 (5e-3)
MMGP	**7.6e-1** (4e-4)	**2.8e-2** (4e-5)	**0.71** (1e-4)	**0.9992** (2e-6)

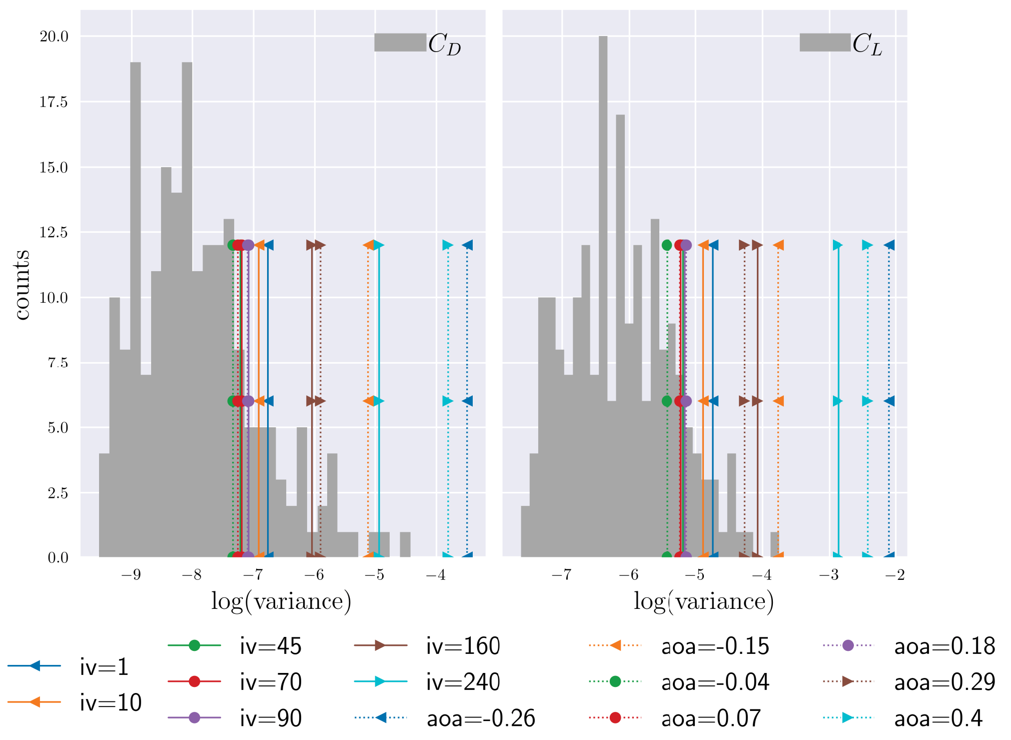

Figure 14: (`AirfRANS`) Histograms of log(variance) of MMGP predictions in grey for the output scalars of interest on the testing set (in distribution). The variance of the MMGP prediction is identified for various configurations, where the same geometry is taken in the testing set and the inlet velocity (iv) and angle of attack (aoa) varies (only iv=45, 70, 90 and aoa=-0.04, 0.07, 0.18 are in the training intervals).

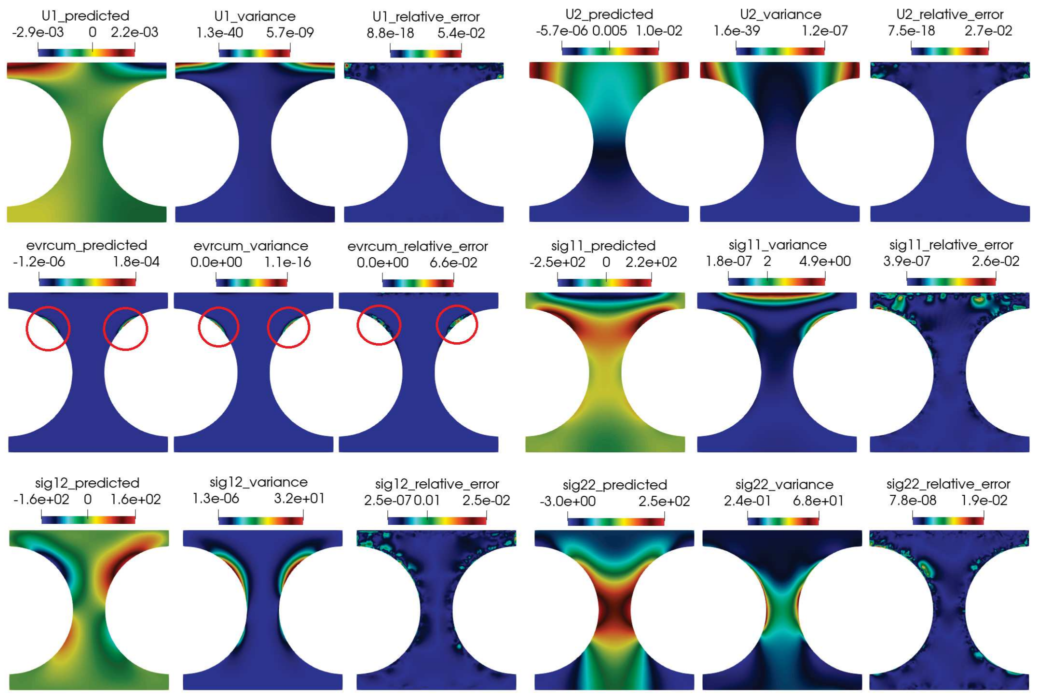

Figure 15: (`Tensile2d`) MMGP prediction for the first training input, where: U_1, U_2, evrcum, sig11, sig22, and sig12 correspond to u, v, p, σ_{11}, σ_{22}, and σ_{12}, respectively.

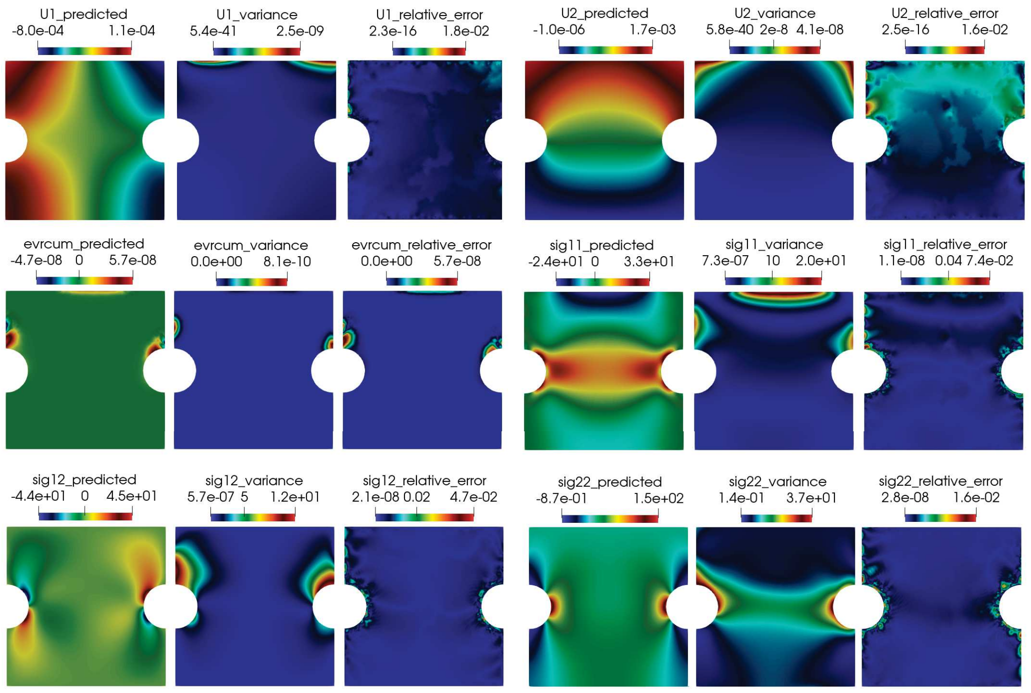

Figure 16: (`Tensile2d`) MMGP prediction for the first test input, where: U_1, U_2, evrcum, sig11, sig22, and sig12 correspond to u, v, p, σ_{11}, σ_{22}, and σ_{12}, respectively.

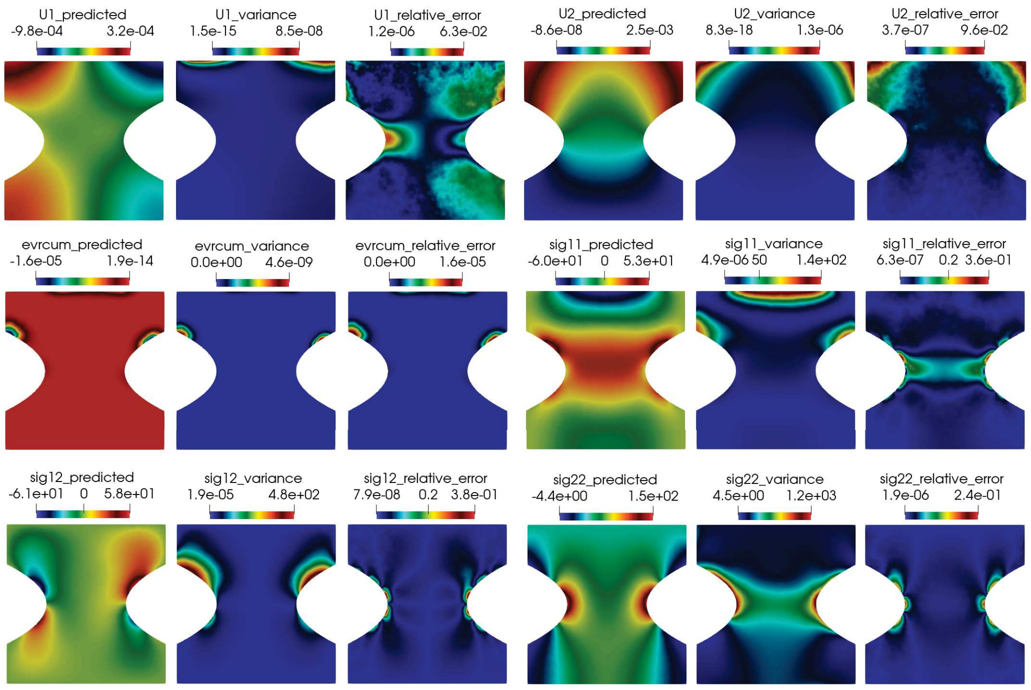

Figure 17: (`Tensile2d`) MMGP prediction for an OOD ellipsoid geometry, where: U_1, U_2, evrcum, sig11, sig22, and sig12 correspond to u, v, p, σ_{11}, σ_{22}, and σ_{12}, respectively.

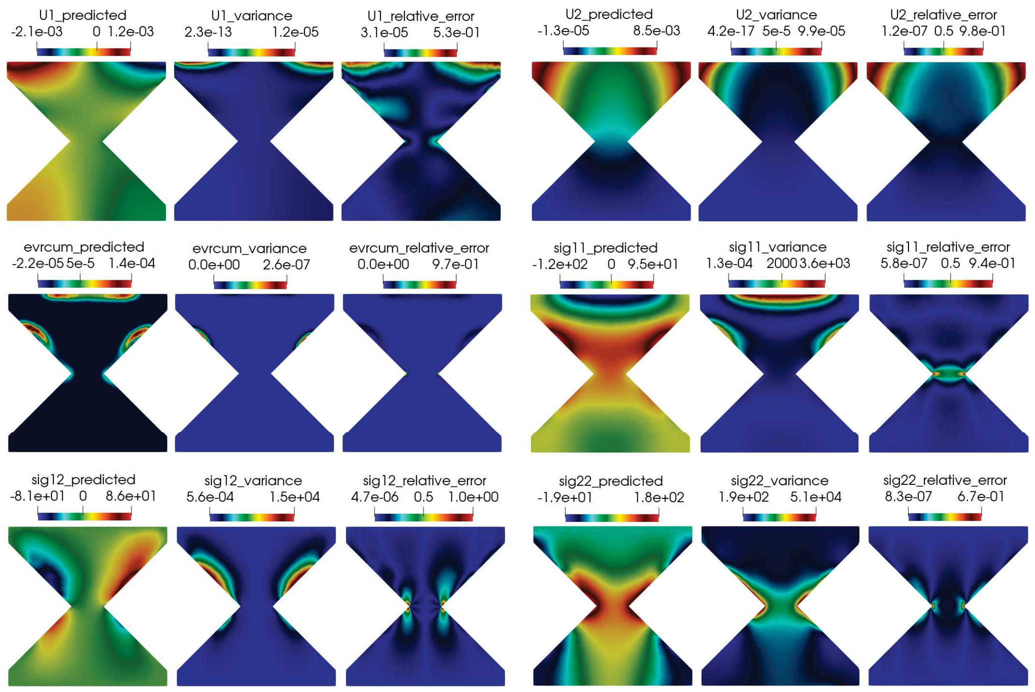

Figure 18: (`Tensile2d`) MMGP prediction for an OOD wedge cut-off geometry, where: U_1, U_2, evrcum, sig11, sig22, and sig12 correspond to u, v, p, σ_{11}, σ_{22}, and σ_{12}, respectively.

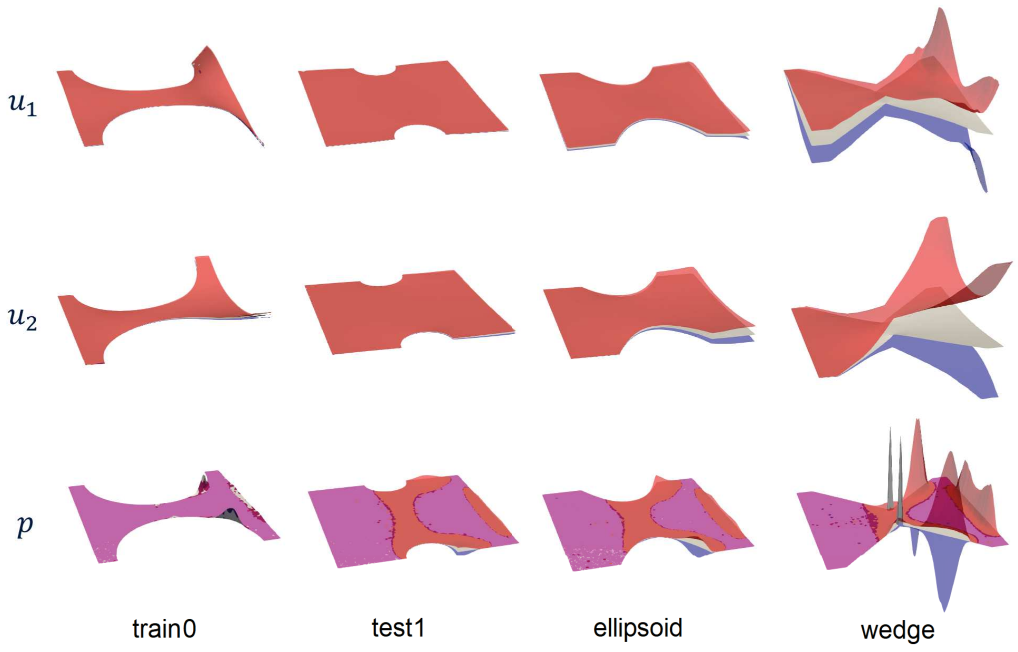

Figure 19: (`Tensile2d`) MMGP: train0, test1, ellipsoid and wedge cases, confidence intervals for u, v and p visualized as surfaces (for each field, the deformation factor is taken identical through the cases).

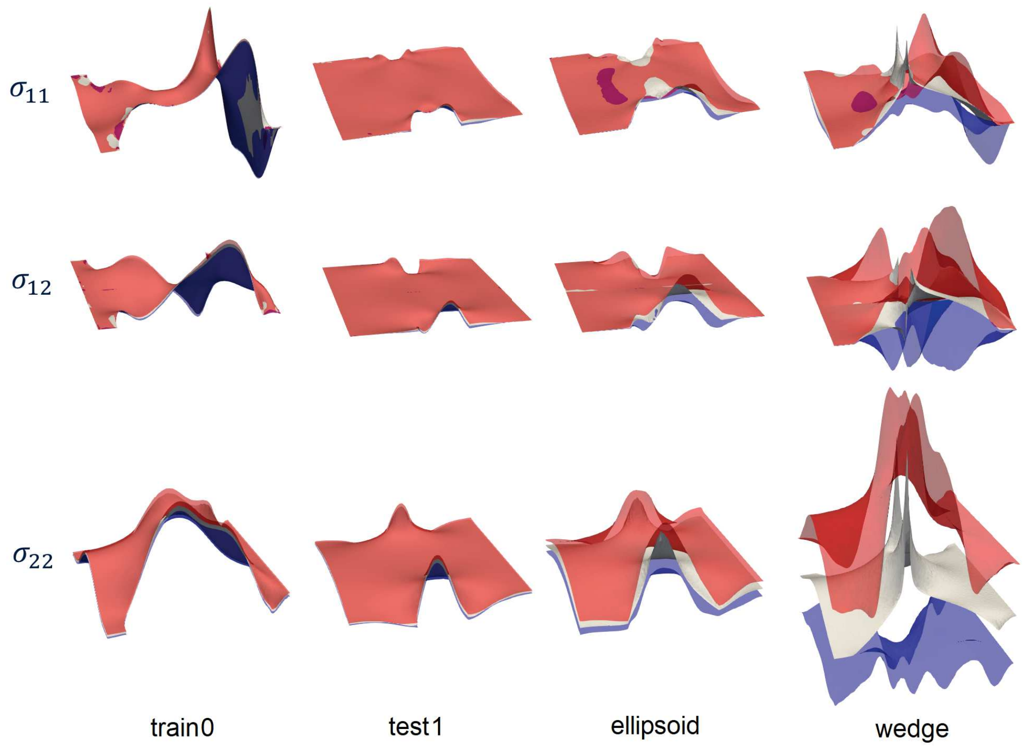

Figure 20: (`Tensile2d`) MMGP: train0, test1, ellipsoid and wedge cases, confidence intervals for σ_{11}, σ_{12} and σ_{22} visualized as surfaces (for each field, the deformation factor is taken identical through the cases).

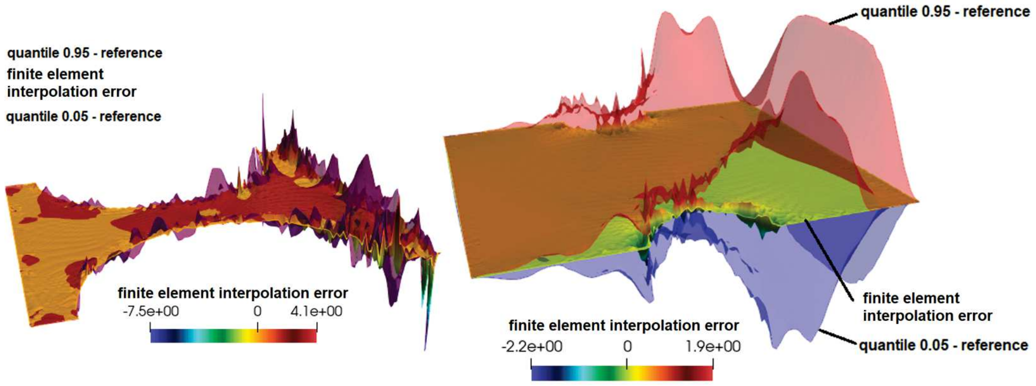
quantile 0.95 - reference
finite element interpolation error
quantile 0.05 - reference
finite element interpolation error
-7.5e+00
0
4.1e+00
quantile 0.95 - reference
finite element interpolation error
quantile 0.05 - reference
finite element interpolation error
-2.2e+00
0
1.9e+00

Figure 21: (`Tensile2d`) Finite element interpolation error for the prediction of σ_{11} compared to the 95% confidence interval for a sample from (left): the training set, (right) the testing set.

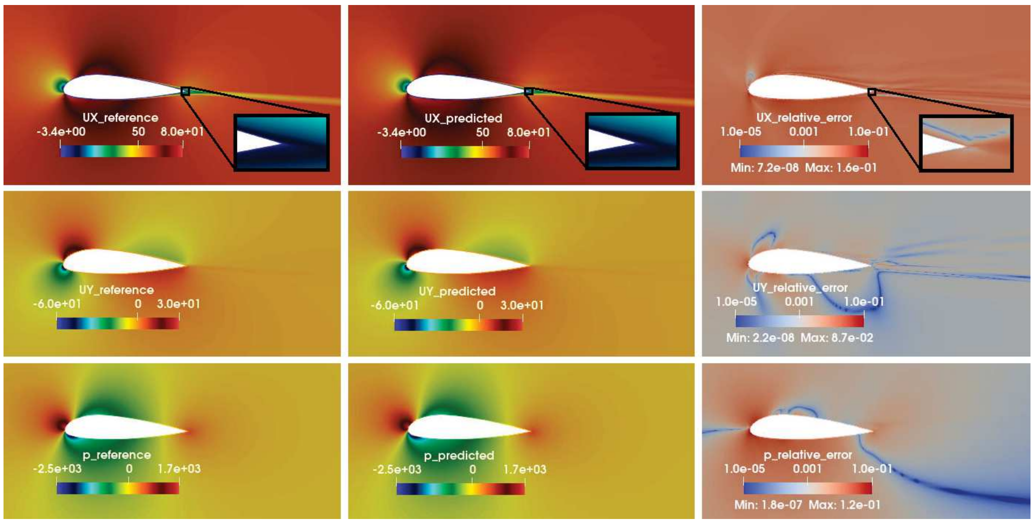

Figure 22: (AirfRANS) Test sample 430, fields of interest u (UX), v (UY) and p: (left) reference, (middle) MMGP prediction, (right) relative error.

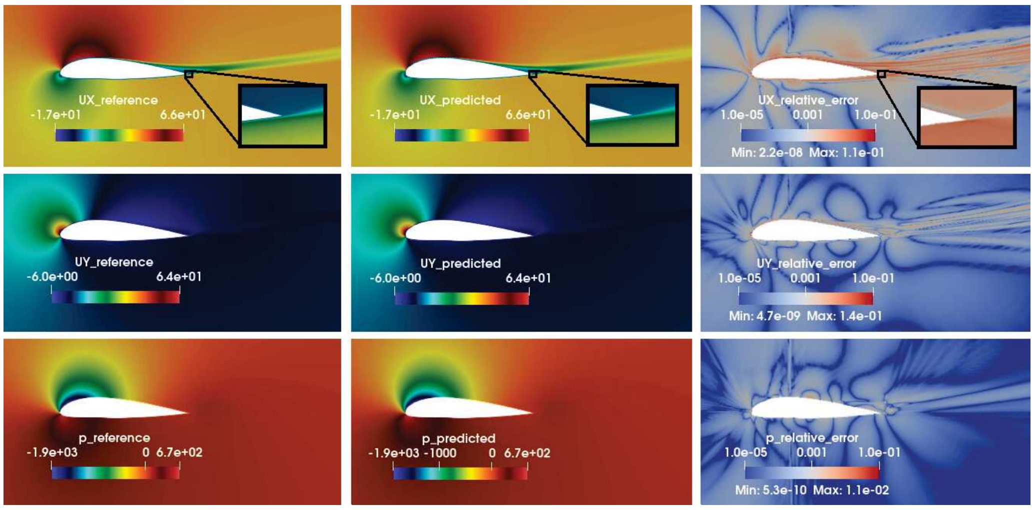

Figure 23: (AirfRANS) Train sample 93, fields of interest u (UX), v (UY) and p: (left) reference, (middle) MMGP prediction, (right) relative error.