Neural Networks for Structured Grid Generation

Bari Khairullin, Sergey Rykovanov, Rishat Zagidullin *
Artificial Intelligence & Supercomputing Laboratory
Skolkovo Institute of Science and Technology
Skolkovo, Bolshoy Boulevard 30, bld. 1, Russia
inbox@skoltech.ru

Abstract

Numerical solutions of partial differential equations (PDEs) on regular domains provide simplicity as we can rely on the structure of the space. We investigate a novel neural network (NN) - based approach to generate 2-dimensional body-fitted curvilinear coordinate systems (BFCs) that allow to stay on regular grids even when the complex geometry is considered. We describe a feed-forward neural network (FNN) as a geometric transformation that can represent a diffeomorphism under certain constraints and approximations, followed by the ways of training it to create BFCs. We show that the optimization system is similar to a physics informed neural network (PINN) based solution of Winslow equations. Unlike in classical BFC generation, FNN provides a differentiable mapping between spaces, and all the Jacobian matrices may be obtained exactly at any given point. Also, it allows to change an interior nodes distribution without the need of recreating the whole mapping.

1 Introduction

When solving PDEs with numerical methods, one usually considers discretized problems on approximations of original domains. There are many algorithms to generate structured and unstructured grids, which are then used along with the Finite Difference (FDM), Finite Volume (FVM) or Finite Elements (FEM) methods. Structured meshes require additional attention when dealing with curved boundaries. One of the most common techniques to handle them is a staircase approximation, however, it has been shown that it may generate significant computational errors, especially in Finite-Difference Time-Domain schemes (FDTD) (Häggblad & Runborg, 2014). Ease of use and computational efficiency of FDM contributed to the development of BFCs, octree meshing (Tu et al., 2005), etc. Unstructured grids provide better control over boundaries and are usually used with FEMs (Benito et al., 2007) and FVMs (Dalal et al., 2008), however, structured grids are still widely used with FDMs, as well as with cutting-edge Physics-Informed Geometry-Adaptive Convolutional Neural Networks (Gao et al., 2021).

BFCs try to overcome the boundary problem by finding a proper map x from a rectangular grid $\Omega_C \subset \mathbb{R}^d$ (computational space) to a given domain $\Omega_P \subset \mathbb{R}^d$ (physical space), such that $x(\partial\Omega_C) = \partial\Omega_P$, x is bijective, smooth and meets certain quality conditions. Afterwards, one can reformulate the initial PDE to be solved in Ω_C with boundaries lying on a rectangular grid. Many algorithms were built to create such a map (Hinz et al., 2018), however, there is no general algorithm that works for any given non-convex domain, which is discussed in the paper Azarenok (2009). In 2022, Xinhai Chen, Tiejun Li, et al. (Chen et al., 2022) were the first to present a way to generate structured meshes with the use of unsupervised NNs called MGNet. They construct a BFC as a $\mathcal{C}^\infty\left(\mathbb{R}^2, \mathbb{R}^2\right)$ function represented by a neural network of the architecture shown in the Figure 1. The fully-connected neural network (FCNN) blocks are the ones containing weights $W_l \in \mathrm{Mat}_{\dim(x_{l+1})\times\dim(x_l)}$, biases $b_l \in \mathrm{Mat}_{\dim(x_{l+1})\times 1}$, and the transformation between two consecutive layers is given via the recurrent formula:

$$x_{l+1} = \sigma\left(W_l x_l + b_l\right) \tag{1}$$

*bari.khairullin@skoltech.ru

with an activation function $\sigma = \tanh$. The optimization process employs non-convex optimization algorithms like stochastic gradient descent or quasi-Newton methods. A significant benefit of this approach is that, following the training phase, mesh refinement can be achieved with just a single forward pass. This is because the mapping function learns to represent the entire geometric transformation, rather than merely its discretized version.

Throughout the work, we develop further the main concepts from (Hauser & Ray, 2017), allowing us to treat FNNs as geometric transformations over space. Consider a set of coordinate systems f on a d-dimensional manifold $\mathcal{M}$ parametrised by $l \in [0, 1]$ that are connected through a differential equation for some $k \in \mathbb{N}$, $T(\cdot, l) \in \mathcal{C}^\infty(\mathcal{M})$ as follows:

$$\frac{d^k f(l)}{dl^k} = T(f(l), l). \tag{2}$$

It can be discretized using the forward-difference scheme to

$$f_{n+1} = T(f_n, n)\,\Delta l^k + \sum_{i=1}^{k} (-1)^{i+1} \binom{k}{i} f_{n+i-k}, \tag{3}$$

where $f_n = f(n\Delta l)$ is a sequence of coordinate systems defined recursively, $\Delta l = 1/N, \quad n \in \overline{0..N}$. If one restricts the T-function, this formula can be treated as a layer-wise transformation of a constant-width residual neural network with $T(f_n, n) = \sigma(W_n f_n + b_n)$. By treating the whole transformation from the layer 0 to the layer N as a combination of transformations between layers, the whole Jacobian matrix becomes a product of consecutive layer-wise Jacobians, establishing differentiable structures between input and output data. Additional routine is needed to handle initial conditions for f, e.g., provide f_0 for $k = 1$, further set $f_1 = f_0$ for $k = 2$, etc. We will call coordinate systems curvilinear $\xi = \left(\xi^1 \dots \xi^d\right)^\top \in \Omega_C$ and Cartesian $x = \left(x^1 \dots x^d\right)^\top \in \Omega_P$ for f_0 and f_N respectively.

2 Structured Grid Generation

2.1 Non-PINN approach

First, let us address the problem from a non-PINN side, i.e. without mesh-loss term. Consider the network (3). One can construct a point-wise loss-function, e.g.

$$L_{\text{boundary}} = \frac{1}{B} \sum_i \| x\left(\xi_{(i)}\right) - x_{(i)} \|_2^2, \tag{4}$$

$$\xi_{(i)} \in \partial\Omega_C, \quad x_{(i)} \in \partial\Omega_P, \quad i = \overline{1..B}, \tag{5}$$

that is minimized through changes in W_n and b_n to match predicted boundary with a physical one. The bigger the value of layers N, the smaller changes a neural network has to learn in a transition between two consecutive layers. Despite the fact that the neural network is considered to be of constant width, one can create an embedding for the input layer:

$$\xi_{(i)} = \underbrace{\left(\xi^1_{(i)} \dots \xi^d_{(i)}, 0 \dots 0\right)}_{d'} \tag{6}$$

and pass it through the NN of width d'. The last layer will represent a projection onto a d-dimensional hypersurface. This approach may seem beneficial due to bigger expressiveness of the NN (Lu et al., 2017), but it tends to generate non-bijective maps on the projection step.

Even if transitions between layers are small, they are not necessarily non-degenerate. To constraint the map, additional relations on weights are needed. Below we consider a neural network with $d = 2$, $k = 1$:

$$f_{n+1} = f_n + \Delta l \sigma\left(W_n f_n + b_n\right), \tag{7}$$

$$J_{n+1} = I + \Delta l D_n W_n, \tag{8}$$

$$D_n = \begin{pmatrix} \dot{\sigma}^1 \left(W_n f_n + b_n\right) & 0 \\ 0 & \dot{\sigma}^2 \left(W_n f_n + b_n\right) \end{pmatrix}. \tag{9}$$

The goal here is to keep the Jacobian determinant non-negative, which will prevent foldings of the mesh. Zero values are allowed to handle cases where boundaries are not smooth and corners are not matched.

$$\det J_{n+1} = 1 + \operatorname{tr}(D_n W_n)\Delta l + \det D_n \det W_n \Delta l^2 \geq 0. \tag{10}$$

If we bound components $0 \leq \dot{\sigma}^i \leq 1$, for example, by taking tanh, Softplus, SiLU, etc. as an activation function, and use the Cauchy-Schwarz inequality, we may estimate the lower bound of the determinant in terms of weights only:

$$\det J_{n+1} \geq 1 - \underbrace{\left(\Delta l \sqrt{2 \sum_i \left(W_{n,ii}\right)^2} - \Delta l^2 \mathcal{H}(\det W_n) \det W_n \right)}_{\mu(W_n)}, \tag{11}$$

where $\mathcal{H}$ is a Heaviside step function. Then the relation $\mu\left(W_n\right) \leq 1$ ensures (10). The pseudo-code to implement such a constraint is written in the algorithm 1.

Algorithm 1 Algorithm to constraint Jacobians

```
W ← W + ΔW                                   ▷ train step
for W_n in W do
    count ← 0
    while μ(W_n) > 1 do
        if count < N_max then
            W_n ← W_n − α ∂μ/∂W_n (W_n)
            count ← count+1
        else
            W_n ← β · W_n
        end if
    end while
end for
```

Here $\alpha > 0$ is a parameter of gradient descent, and $0 < \beta < 1$ is a coefficient for robust algorithm convergence. Note that biases b_n are optimized during training, too, but they do not contribute to the constraint.

2.2 PINN-approach

To control interior grid points, one may consider additional loss function that measures the deviation of mapping from being conformal on randomly sampled interior points distribution, e.g., Liao, Winslow, area orthogonality, etc. (Khattri, 2007).

$$\|g_{ij}\| = J^\top J \tag{12}$$

is called a metric tensor, and one of the advantages of neural networks is that they can handle gradients of g_{ij} with respect to weights and inputs exactly. In two dimensions, the Winslow integrand is approximated on Ω_C by:

$$L_{\text{int}} \approx \frac{1}{P} \sum_p \left(\frac{g_{11} + g_{22}}{\sqrt{\det g}} \right)_{(p)}, \tag{13}$$

$$(g_{ij})_{(p)} = g_{ij}(\xi_{(p)}), \quad \xi_{(p)} \in \Omega_C \setminus \partial\Omega_C, \quad i = \overline{1..P} \tag{14}$$

with the total loss

$$L = L_{\text{boundary}}(x, \xi) + \epsilon L_{\text{int}} \left(\frac{\partial x}{\partial \xi} \right). \tag{15}$$

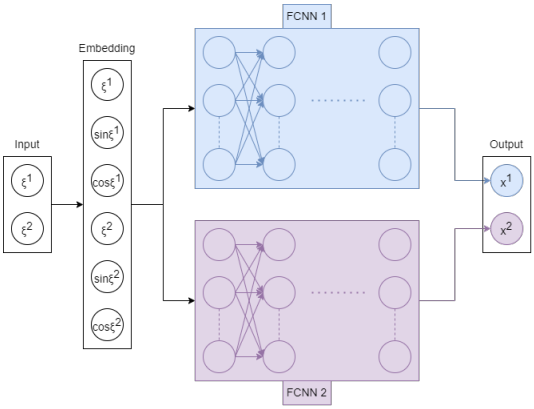


Figure 1: MGNet architecture without excitation blocks.

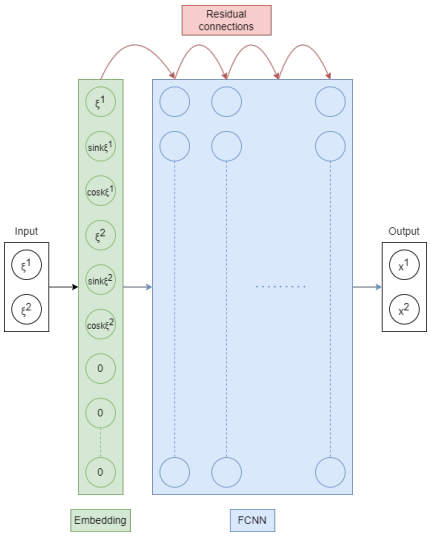


Figure 2: Proposed network architecture.

Here we propose to use the neural network architecture shown in Figure 2. First, it embeds the input space to have several Fourier modes

$$\left(\cos(\xi^1), \cos(2\xi^1), \dots \cos(k\xi^1), \sin(\xi^1), \dots, \cos(\xi^2) \dots\right), \tag{16}$$

as well as additional zero-nodes to match the size of hidden layers. Second, it passes values with $\mathcal{C}^1$-smooth structure, projecting to two-dimensional space at the last transformation. The main idea behind this approach is to initialize all the weights of the FCNN with zeros, and weights of the last transformation with values representing first two coordinates of the last hidden layer:

$$f_N = \begin{pmatrix} \mathrm{I}_{2\times 2} \\ 0_{\dim(f_{N-1})\times 2} \end{pmatrix} f_{N-1} + 0_{2\times 1}. \tag{17}$$

This way, the initial mapping before training is an identity map. Following that it is non-degenerate by construction, the loss (13) will have only positive values of the Jacobian determinant, preventing foldings. This is ensured due to the denominator of the loss, which approaches zero as a folding occurs. Additionally, bigger orders of smoothness $\mathcal{C}^k$ are possible by copying an embedding layer $k-1$ times.

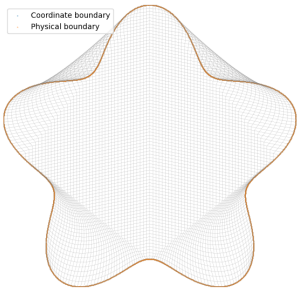
One question to address, is why using residual connections between all of two consecutive layers as it is possible to reconstruct an initial identity map by connecting, for example, the last layer with the first one only. The answer lies in the fact that if we do the latter, then

$$f_n = f_0 + \mathrm{Net}\left(\theta_{W,b}\right) \tag{18}$$

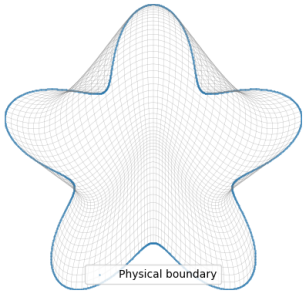
and the neural network has to learn a big deviation from the identity map, which still may provide foldings due to discretization of optimization algorithms. In contrast, the proposed network learns small deviations from identity map from layer to layer, sharing better control over regularity of grid.

2.3 Numerical experiments

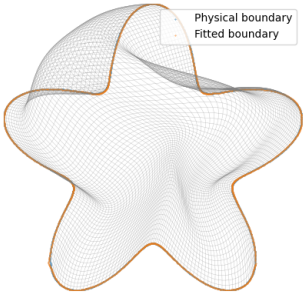
It can be seen in the Figure 3 that algebraic interpolaters failed to fit the star shape. Without additional control of an interior point distribution, neural networks tend to generate degenerate maps (Figure 3c), and while the non-PINN method provided some bijective solution, its convergence at boundaries and overall mesh are of the poor quality. The best solution was obtained by a PINN method with 2 fourier modes (i.e. additional 8 dimensions).



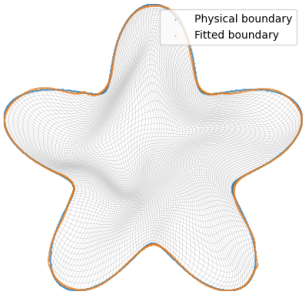
(a) scipy LinearNDInterpolator.



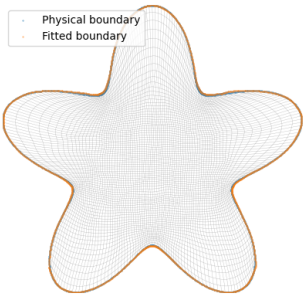
(b) Transfinite interpolation.



(c) Embedding without interior loss.

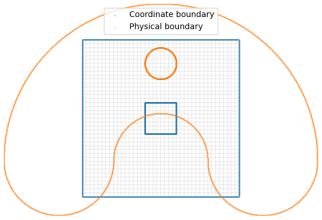


(d) Non-PINN method.

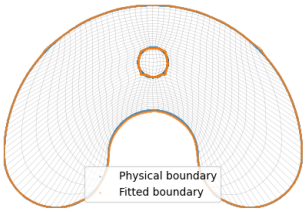


(e) PINN method with 2 fourier modes and interior loss.

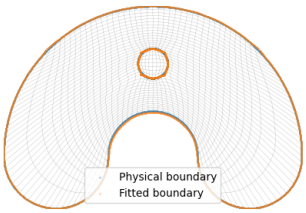
Figure 3: Flower dataset fits comparison.



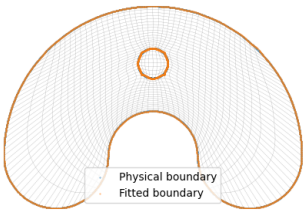
(a) Iterations number: 0.



(b) Iterations number: 1200.

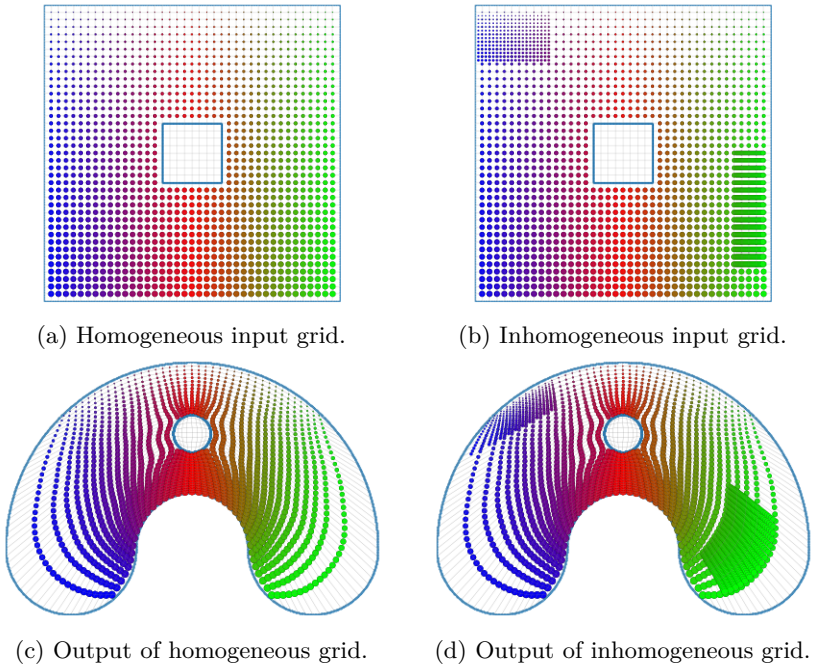


(c) Iterations number: 9800.



(d) Iterations number: 11000.

Figure 4: Multi connected dataset training process.



(a) Homogeneous input grid.

(b) Inhomogeneous input grid.

(c) Output of homogeneous grid.

(d) Output of inhomogeneous grid.

Figure 5: Example of mesh refinement in input and output spaces.

3 Conclusions

It was shown that a neural network can be used for body-fitted curvilinear coordinate system generation with further application to finite-difference solvers of partial differential equations. While usually neural networks are considered black-box functions, several methods can be used to constraint Jacobians through weights control and mesh loss functions. Benefits of such a grid generation relative to discrete schemes of Winslow equations and others, are that it allows to vary the interior points distribution on the computational domain and compute metric tensors exactly, leading to better representations of differential operators. The use of BFC can be justified, for example, in inverse problem to find estimations of parameters on sparse grids, because it is fast and captures the boundary much better than a staircase approximation.

Still, the grid generator is far from being robust in terms of convergence in a soft loss setup. Further investigation implies the analysis of hard-constrained PINN solvers and interpretations of neural network in that case. Also, such features as 3D domain processing and time-dependent boundaries are to be investigated and still remain an open question.

References

Boris Nikolaevich Azarenok. Generation of structured difference grids in two-dimensional nonconvex domains using mappings. Computational Mathematics and Mathematical Physics, 49:797–809, 2009.

Josip Basic, Nastia Degiuli, and Dario Ban. A class of renormalised meshless laplacians for boundary value problems. Journal of computational physics, 354:269–287, 2018.

Yoshua Bengio and Yann LeCun. Scaling learning algorithms towards AI. In Large Scale Kernel Machines. MIT Press, 2007.

JJ Benito, F Urena, and L Gavete. Solving parabolic and hyperbolic equations by the generalized finite difference method. Journal of computational and applied mathematics, 209(2):208–233, 2007.

Xinhai Chen, Tiejun Li, Qian Wan, Xiaoyu He, Chunye Gong, Yufei Pang, and Jie Liu. Mgnet: a novel differential mesh generation method based on unsupervised neural networks. Engineering with Computers, 38(5):4409–4421, 2022.

Amaresh Dalal, V Eswaran, and G Biswas. A finite-volume method for navier-stokes equations on unstructured meshes. Numerical Heat Transfer, Part B: Fundamentals, 54(3): 238–259, 2008.

Han Gao, Luning Sun, and Jian-Xun Wang. Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain. Journal of Computational Physics, 428:110079, 2021.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. Deep learning, volume 1. MIT Press, 2016.

Jon Häggblad and Olof Runborg. Accuracy of staircase approximations in finite-difference methods for wave propagation. Numerische Mathematik, 128:741–771, 2014.

Michael Hauser and Asok Ray. Principles of riemannian geometry in neural networks. Advances in neural information processing systems, 30, 2017.

Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. Neural Computation, 18:1527–1554, 2006.

Jochen Hinz, Matthias Möller, and Cornelis Vuik. Elliptic grid generation techniques in the framework of isogeometric analysis applications. Computer Aided Geometric Design, 65: 48–75, 2018.

Sanjay Kumar Khattri. Grid generation and adaptation by functionals. Computational & Applied Mathematics, 26:235–249, 2007.

Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. Advances in neural information processing systems, 30, 2017.

Tiankai Tu, David R O'Hallaron, and Omar Ghattas. Scalable parallel octree meshing for terascale applications. In SC'05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing, pp. 4–4. IEEE, 2005.

Appendix

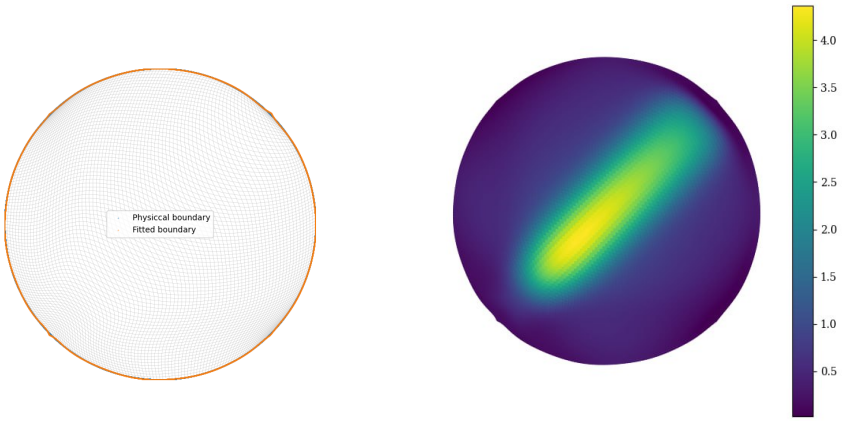
PDE solutions

Consider the Helmholtz equation with zero Dirichlet boundary conditions in a unit circle:

$$\Delta u + u = v, \tag{19}$$

$$u \in \Omega_P = \mathcal{B}(0,1) \quad u(\partial\Omega_P) = 0. \tag{20}$$

If $v(r,\phi) = J_1\left(\sqrt{\lambda_2^{(1)}}r\right)\cos(\phi)$, the exact solution is known to be $u = \frac{v}{1-\lambda_2^{(1)}}$.



(a) Grid lines of constant ξ^1, ξ^2. (b) Metric tensor determinant distribution.

Figure 6: NN generated BFC for the unit circle. NN configuration: 10 hidden layers, tanh activation function, $k = 1$ smoothness, no grid controlling term. Loss on boundary: 1.18e-06.

One of the advantages of generating grid with neural networks is that it gives freedom in a choice of gridsize on the computational domain. The grid with $h_x = h_y = h = 1/100$ with $98^2 = 9604$ unknowns was taken. Also, it was compared to the classic FD solver with a staircase approximation of the boundary on a flat space with 10364 unknowns. The results are given in the Figure 7. The BFC algorithm outperforms the classic one even with smaller average density of interior points.

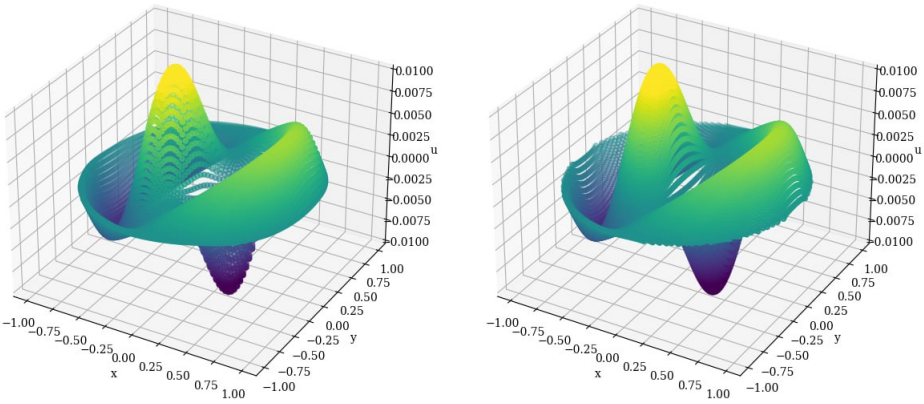
Another example to compare both methods is a heat equation on an hourglass-shaped domain.

$$\frac{\partial u}{\partial t} = \Delta u + v, \tag{21}$$

$$u(t, \partial\Omega_P) = 0, \quad u(0, \Omega_P) = w(\Omega_P), \quad v = 1. \tag{22}$$

To have a reference, a numerical solution with 12158 interior points was used that is based on the meshless Laplace operator discretization given by J.Basic, et al. (2018)Basic et al. (2018). The scheme is implicit and the resulting linear system is sparse. To solve it we use the conjugate gradient algorithm from the Eigen library. Much sparser meshes were used for FD solvers, 1444 and 1452 interior points for BFC and staircase approximations, respectively.

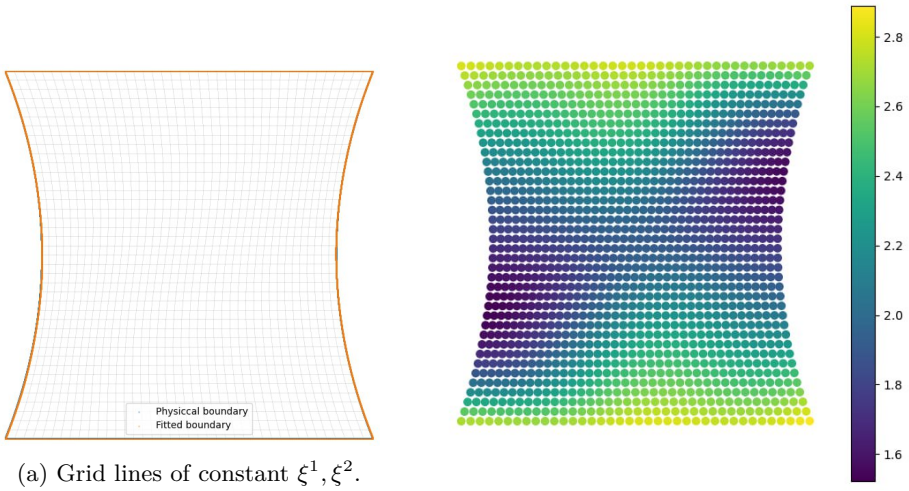
Both simulations converge to a static heat distribution $\Delta u = -1$ if sufficiently small timestep is taken. BFC algorithm again outperforms the classic one, giving smaller errors on the whole simulation time domain.



(a) BFC FD solution with 9604 unknowns, RMSE = 4e-5.

(b) Classic FD on a staircase approximation with 10364 unknowns, RMSE = 1.2e-4.

Figure 7: Comparison of BFC and classic FD solutions. RMSE is computed relative to the exact solution at corresponding nodes.



(a) Grid lines of constant ξ^1, ξ^2.

(b) Metric tensor determinant distribution.

Figure 8: NN generated BFC for an hourglass-shaped domain. NN configuration: 5 hidden layers, tanh activation function, $k = 1$ smoothness, Winslow (13) grid controlling loss, loss on boundary is equal to 3.38e-06.

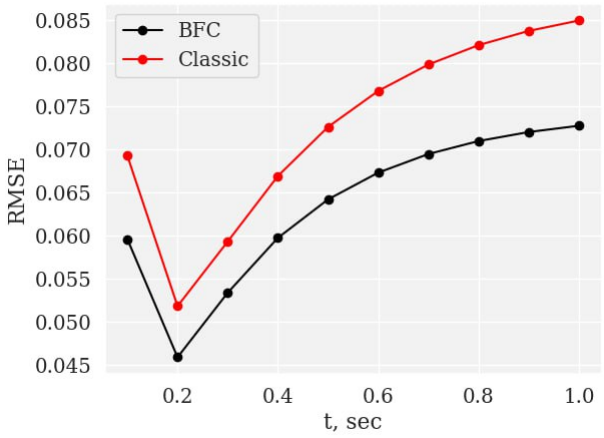


Figure 9: RMSE of FD solutions relative to a reference with respect to the physical time of simulation.