Neuro-symbolic hybridization of finite discretization methods for solving partial differential equations

Pouria A. Mistani[†*]
NVIDIA
Santa Clara, CA 95051
pmistani@nvidia.com

Samira Pakravan [†]
University of California
Santa Barbara, CA 93106
spakravan@ucsb.edu

Rajesh Ilango
NVIDIA
Santa Clara, CA 95051
rilango@nvidia.com

Sanjay Choudhry
NVIDIA
Santa Clara, CA 95051
schoudhry@nvidia.com

Frederic G. Gibou
University of California
Santa Barbara, CA 93106
fgibou@ucsb.edu

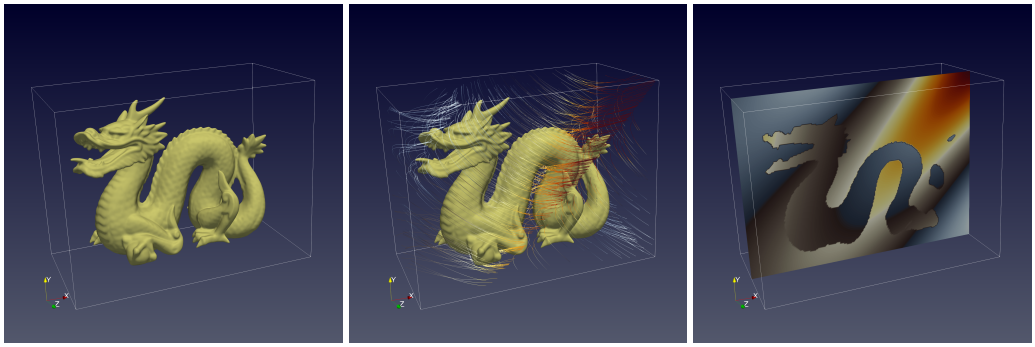

Figure 1: The proposed Neural Bootstrapping Method applied to computing the electrostatic field around a complex charged geometry in a complex dielectric. The neural approach readily enables a 8192^3 effective resolution on a GPU cluster and captures the physically correct discontinuities. Geometry (left), streamlines (center), cross-section showing jump in solution and/or gradient.

Abstract

We present a highly scalable strategy for developing mesh-free hybrid neuro-symbolic partial differential equation (PDE) solvers based on existing mesh-based numerical discretization methods. Particularly, this strategy can be used to efficiently train neural network surrogate models for the solution functions and operators of PDEs while retaining the accuracy and convergence properties of the state-of-the-art numerical solvers. The presented neural bootstrapping method (hereby dubbed NBM) is based on minimizing residuals of discretized PDE system on an array of implicit Cartesian cells at different levels of resolution centered on a set of random collocation points with respect to trainable parameters of the neural network. NBM leverages neural networks to achieve unprecedented resolution and flexibility for solving PDEs that describe complex physical systems.

[*]corresponding author.
[†]These authors contributed equally to this work.

Preprint. Under review.

1 Introduction

Most modern physical and engineering systems are described by partial differential equations on irregular, often moving, boundaries. The difficulties in solving those problems stem from how to approximate the equations, while respecting the physically correct discontinuous nature of the solution across the boundaries. Smoothing strategies are easy to design, but unfortunately introduce unphysical characteristics in the solution and lead to systemic errors.

Since early 1990s, artificial neural networks have been used for solving partial differential equations by (i) mapping the algebraic operations of the discretized PDE systems onto specialized neural network architectures and minimizing the network energy, or (ii) treating the whole neural network as the basic approximation unit whose parameters are adjusted to minimize a specialized error function that includes the differential equation itself with its boundary/initial conditions.

In the first category, neurons output the discretized solution values over a set number of grid points and minimizing the network energy drives the neuronal values towards the solution of the linear system at the mesh points. In this case, the neural network energy is the residual of the finite discretization method summed over all neurons of the network [20]. Although the convergence properties of the finite discretization methods guarantee and control quality of the obtained solutions, the computational costs grow by increasing resolution and dimensionality. Some early examples include [15, 10, 9].

The second strategy proposed by Lagaris *et al.* [19] relies on the function approximation capabilities of the neural networks. Encoding the solution everywhere in the domain within a neural network offers a mesh-free, compact, and memory efficient surrogate model for the solution function that can be utilized in subsequent inference tasks. This method has recently re-emerged as the physics-informed neural networks (PINNs) [31] and is widely used. Despite their advantages, these methods lack controllable accuracy and convergence properties of finite discretization methods, and are biased towards lower frequency features of the solutions [37, 30].

Pursuit of hybrid solvers aims at leveraging the performance gains of neural network inference on modern accelerated hardware with the guaranteed accuracy of finite discretization methods. The hybridization efforts are algorithmic or architectural.

One important algorithmic method is the deep Galerkin method (DGM) [34] that is a neural network extension of the mesh-free Galerkin method where the solution is represented as a deep neural network rather than a linear combination of basis functions. The mesh-free nature of DGM, that stems from the underlying mesh-free Galerkin method, enables solving problems in higher dimensions by training the neural network model to satisfy the PDE operator and its initial and boundary conditions on a randomly sampled set of points rather than on an exponentially large grid. Although the number of points is huge in higher dimensions, the algorithm can process training on smaller batches of data points sequentially. Besides, second order derivatives in PDEs are calculated by a Monte Carlo method that retain scaling to higher dimensions. Another important algorithmic method is the deep Ritz method for solving variational problems [38] that implements a deep nerual network approximation of the trial function that is constrained by numerical quadrature rule for the variational functional, followed by stochastic gradient descent.

Architectural hybridization methods are based on differentiable numerical linear algebra. One emerging class involves implementing differentiable finite discretization solvers and embedding them in the neural network architectures that enable application of end-to-end differentiable gradient based optimization methods. Recently, differentiable solvers have been developed in JAX [7] for fluid dynamic problems, such as `Phi-Flow` [17], `JAX-CFD` [18], and `JAX-FLUIDS` [2]. These methods are suitable for inverse problems where an unknown field is modeled by the neural network, while the model influence is propagated by the differentiable solver into a measurable residual [29, 12, 24]. We also note the classic strategy for solving inverse problems is the adjoint method to obtain the gradient of the loss without differentiation across the solver [1]; however, deriving analytic expression for the adjoint equations is tedious, should be repeated after modification of the problem or its loss function, and can become impractical for multiphysics problems. Other important utilities of differentiable solvers are to model and correct for the solution errors of finite discretization methods [36], learning and controling PDE systems [13, 16].

Neural networks are not only universal approximators of continuous functions, but also of nonlinear operators [8]. Although this fact has been leveraged using data-driven strategies for learning differential operators by many authors [23, 3, 22, 21], current authors have demonstrated utility of differentiable solvers to effectively train nonlinear operators without any data in a completely physics-driven fashion, see section on learning the inverse transforms in [29].

In this work we propose a novel framework for solving PDEs based on deep neural networks by lifting any existing mesh-based finite discretization method off of its underlying grid and extend it into a mesh-free method that can be applied to high dimensional problems on unstructured random points in an embarrasingly parallel fashion. In addition, discontinuous solutions can be readily considered.

2 Problem statement

In order to illustrate our approach, we consider a closed irregular interface (Γ) that partitions the computational domain (Ω) into interior (Ω^-) and exterior (Ω^+) subdomains; *i.e.*, $\Omega = \Omega^- \cup \Gamma \cup \Omega^+$. We are interested in the solutions $\mathrm{u}^\pm \in \Omega^\pm$ to the following class of linear elliptic problems in $\mathbf{x} \in \Omega^\pm$:

$$\begin{aligned} k^\pm u^\pm - \nabla \cdot (\mu^\pm \nabla u^\pm) &= f^\pm, && \mathbf{x} \in \Omega^\pm \\ [u] &= \alpha, && \mathbf{x} \in \Gamma \\ [\mu \partial_{\mathbf{n}} u] &= \beta, && \mathbf{x} \in \Gamma \end{aligned}$$

Here $f^\pm = f(\mathbf{x} \in \Omega^\pm)$, $\mu^\pm = \mu(\mathbf{x} \in \Omega^\pm)$ and $k^\pm$ are the spatially varying source term, diffusion coefficients, and coefficients, respectively in the two domains. For simplicity, we consider Dirichlet boundary conditions at the boundary of a cubic domain $\Omega = [-\mathrm{L}/2, \mathrm{L}/2]^3$, noting that other boundary conditions can be readily considered.

This class of problems not only captures the difficulties of solving partial differential equations on irregular domain with discontinuous solutions, but also illustrate their multiscale nature that requires a nonuniform sampling in space. In addition, this system of equations is important on its own as it arises ubiquitously in describing diffusion dominated processes in physical systems and in the life sciences, where sharp and irregular interfaces regulate transport across regions with different properties. Examples include Poisson-Boltzmann equation for describing electrostatic properties of membranes, colloids and solvated biomolecules with jump in dielectric permitivities [33, 25], in electroporation of cell aggregates with nonlinear membrane jump conditions [27], or epitaxial growth in fabrication of opto-electronic devices [26]. Other important applications are found in solidification of multicomponent alloys [35, 6], directed self-assembly of diblock copolymers for next generation lithography [14, 28, 5], multiphase flows with and without phase change.

3 Scalable & Mesh-Freeing Neuro-Symbolic PDE Solver

Neural networks are used as surrogates for the solution function that are iteratively adjusted to minimize discretization residuals at a set of randomly sampled points and at arbitrary resolutions. The key idea is that neural networks can be evaluated over vertices of any discretization stencils centered at any point in the domain effectively emulating the effect of any structured mesh without ever materializing the mesh. Therefore, we use neural networks to bootstrap mesh based finite discretization (FD) methods to compile mesh free numerical methods. We call this the Neural Bootstrapping Method (NBM), and illustrate it in figure 2.

FD methods offer guaranteed accuracy and controllable convergence properties for the training of neural network surrogate models of physical systems. NBM offers a straightforward path for applying mesh-based FD methods on unstructured random points. This is an important ability for augmenting observational data in the training pipelines. Beside its mesh-freeing advantage, NBM is a highly parallelizable strategy for FD methods. Pointwise nature of its kernels is ideally suited for GPU-accelerated computing paradigm. Multi-GPU parallel solution of PDE systems is reduced to the much simpler problem of data-parallel training using existing machine learning frameworks. Data parallelism involves distributing collocation points across multiple processors to compute gradient updates and then aggregating these locally computed updates [32].

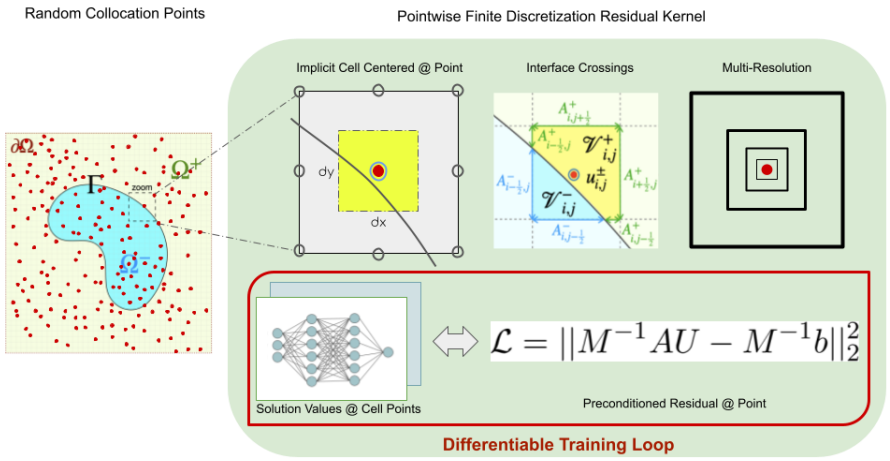

Figure 2: Neural Bootstrapping Method (NBM). NBM kernels compute residual contribution by each collocation point per GPU thread. Kernel operations involve implicit cells at different resolutions according to the bootstrapped finite discretization method. Geometric information for interface-cell crossings is computed by a level-set interpolant defined on a much lower resolution grid. Pointwise residuals are locally preconditioned based on the geometry of the interface crossing the implicit cells.

Here we bootstrap the numerical scheme proposed by [4], hence the loss function is

$$\mathcal{L} = \Bigg|\Bigg| \sum_{s=-,+} k^s_{i,j} u^s_{i,j} |\mathcal{V}^s_{i,j}| - \sum_{s=-,+} \Bigg(\mu^s_{i-\frac{1}{2},j} A^s_{i-\frac{1}{2},j} \frac{u^s_{i-1,j} - u^s_{i,j}}{\Delta x} + \mu^s_{i+\frac{1}{2},j} A^s_{i+\frac{1}{2},j} \frac{u^s_{i+1,j} - u^s_{i,j}}{\Delta x} + \mu^s_{i,j-\frac{1}{2}} A^s_{i,j-\frac{1}{2}} \frac{u^s_{i,j-1} - u^s_{i,j}}{\Delta y} + \mu^s_{i,j+\frac{1}{2}} A^s_{i,j+\frac{1}{2}} \frac{u^s_{i,j+1} - u^s_{i,j}}{\Delta y} \Bigg) - \sum_{s=-,+} f^s_{i,j} |\mathcal{V}^s_{i,j}| - \int_{\Gamma \cap \mathcal{V}_{i,j}} \beta d\Gamma \Bigg|\Bigg|_2^2$$

Operations in differentiable NBM kernels are strictly local. NBM starts by placing implicit compute cells of a specified resolutions at the collocation point. At the presence of discontinuities a separate coarse mesh encapsulates an interpolant for the level-set function whose intersection with the NBM cell is calculated to obtain geometric information for the FD kernel and preconditioner (denoted M^{-1}). FD kernel is applied on the compute cell where the solution values are evaluated by the neural network. Each kernel contributes a local L^2-norm residual $r_p = ||M^{-1}AU - M^{-1}b||$ at one point p. Preconditioning helps to balance relative magnitude of contributions from all points before aggregating residuals to form a global loss value. Finally, gradient based optimization methods used in machine learning are applied to adjust neural network parameters. The automatic differentiation loop passes across the NBM kernels.

4 Numerical results

4.1 Convergence

We consider a sphere centered at the origin with radius 1/2 in a domain $[0, 1]^3$, an exact solution $u^-(x, y, z) = e^z$ inside and $u^+(x, y, z) = \cos(x)\sin(y)$ outside. In addition to the jump in solution, we also consider a jump in the variable diffusion coefficient to be $\mu^-(x, y, z) = y^2 \ln(x + 2) + 4$ inside and $\mu^+(x, y, z) = e^{-z}$ outside the sphere. Table 1 reports convergence results for the solution, which illustrates the convergence in the L^∞-norm.

4.2 Complex geometry and physics

We simulate a Poisson problem with discontinuities on the Dragon problem presented in [11]. In this case we used the signed-distance function produced by SDFGen, and initiated an interpolant based on its values. An exact solution $u^-(x, y, z) = \sin(2x)\cos(2y)e^z$ inside and $u^+(x, y, z) =$

Table 1: We report L^∞-norm error and root-mean-squared-error (RMSE) of the solution field evaluated in the domain. Rightmost column reports the overall time to solution for our JAX implementation which constitutes $10,000$ epochs in each case and the initial compilation time of jaxpressions. A pair of neural networks, each with 5 hidden layers of 10 neurons wide, have 982 total trainable parameters. In each case GPU compute occupancy is at 100% on a single NVIDIA RTX A6000 GPU.

regress ∂_n	RMSE		L^∞		GPU Statistics	
$N_{x,y,z}$	Solution	Order	Solution	Order	t (sec/epoch)	VRAM (GB)
2^3	3.7×10^{-2}	-	3.25×10^{-1}	-	0.0306	1.05
2^4	7.1×10^{-3}	2.38	1.10×10^{-1}	1.56	0.056	1.72
2^5	5.9×10^{-3}	0.27	8.36×10^{-2}	0.4	0.053	2.15
2^6	4.1×10^{-3}	0.53	6.44×10^{-2}	0.38	0.287	5.57
2^7	2.64×10^{-3}	0.64	3.53×10^{-2}	0.87	2.125	32.1

$\left[16\left(\frac{y-x}{3}\right)^5 - 20\left(\frac{y-x}{3}\right)^3 + 5\left(\frac{y-x}{3}\right)\right] \ln(x+y+3)\cos(z)$ outside with variable diffusion coefficients $\mu^-(x,y,z) = 10\left[1+0.2\cos(2\pi(x+y))\sin(2\pi(x-y))\cos(z)\right]$ inside and $\mu^+(x,y,z) = 1$ outside. The results are shown in figure 1, with L^∞-norm of 0.5 and RMSE of 0.06 after 1000 epochs on multi-resolutions $64^3, 128^3, 256^3, 512^3$.

4.3 Parallel scaling on network of GPUs

Put the weak and strong scaling.

5 Discussion and conclusions

We developed a differentiable GPU-based framework for solving partial differential equations with jump conditions across irregular interfaces in three spatial dimensions. Solutions in each domain are represented by a simple multi-layer perceptron (MLP) and Cartesian grid points of the underlying numerical discretization scheme are treated as collocation points for optimizing the unknown parameters of the MLPs.

More here.

The following describes the formatting of a NeurIPS paper:

6 Submission of papers to NeurIPS 2022

Please read the instructions below carefully and follow them faithfully.

6.1 Style

Papers to be submitted to NeurIPS 2022 must be prepared according to the instructions presented here. Papers may only be up to **nine** pages long, including figures. Additional pages *containing only acknowledgments and references* are allowed. Papers that exceed the page limit will not be reviewed, or in any other way considered for presentation at the conference.

The margins in 2022 are the same as those in 2007, which allow for $\sim$15% more words in the paper compared to earlier years.

Authors are required to use the NeurIPS LaTeX style files obtainable at the NeurIPS website as indicated below. Please make sure you use the current files and not previous versions. Tweaking the style files may be grounds for rejection.

6.2 Retrieval of style files

The style files for NeurIPS and other conference information are available on the World Wide Web at

`http://www.neurips.cc/`

The file `neurips_2022.pdf` contains these instructions and illustrates the various formatting requirements your NeurIPS paper must satisfy.

The only supported style file for NeurIPS 2022 is `neurips_2022.sty`, rewritten for LaTeX 2_ε. **Previous style files for LaTeX 2.09, Microsoft Word, and RTF are no longer supported!**

The LaTeX style file contains three optional arguments: `final`, which creates a camera-ready copy, `preprint`, which creates a preprint for submission to, e.g., arXiv, and `nonatbib`, which will not load the `natbib` package for you in case of package clash.

Preprint option If you wish to post a preprint of your work online, e.g., on arXiv, using the NeurIPS style, please use the `preprint` option. This will create a nonanonymized version of your work with the text "Preprint. Work in progress." in the footer. This version may be distributed as you see fit. Please **do not** use the `final` option, which should **only** be used for papers accepted to NeurIPS.

At submission time, please omit the `final` and `preprint` options. This will anonymize your submission and add line numbers to aid review. Please do *not* refer to these line numbers in your paper as they will be removed during generation of camera-ready copies.

The file `neurips_2022.tex` may be used as a "shell" for writing your paper. All you have to do is replace the author, title, abstract, and text of the paper with your own.

The formatting instructions contained in these style files are summarized in Sections 7, 8, and 9 below.

7 General formatting instructions

The text must be confined within a rectangle 5.5 inches (33 picas) wide and 9 inches (54 picas) long. The left margin is 1.5 inch (9 picas). Use 10 point type with a vertical spacing (leading) of 11 points. Times New Roman is the preferred typeface throughout, and will be selected for you by default. Paragraphs are separated by ½ line space (5.5 points), with no indentation.

The paper title should be 17 point, initial caps/lower case, bold, centered between two horizontal rules. The top rule should be 4 points thick and the bottom rule should be 1 point thick. Allow ¼ inch

space above and below the title to rules. All pages should start at 1 inch (6 picas) from the top of the page.

For the final version, authors' names are set in boldface, and each name is centered above the corresponding address. The lead author's name is to be listed first (left-most), and the co-authors' names (if different address) are set to follow. If there is only one co-author, list both author and co-author side by side.

Please pay special attention to the instructions in Section 9 regarding figures, tables, acknowledgments, and references.

8 Headings: first level

All headings should be lower case (except for first word and proper nouns), flush left, and bold.

First-level headings should be in 12-point type.

8.1 Headings: second level

Second-level headings should be in 10-point type.

8.1.1 Headings: third level

Third-level headings should be in 10-point type.

Paragraphs There is also a `\paragraph` command available, which sets the heading in bold, flush left, and inline with the text, with the heading followed by 1 em of space.

9 Citations, figures, tables, references

These instructions apply to everyone.

9.1 Citations within the text

The `natbib` package will be loaded for you by default. Citations may be author/year or numeric, as long as you maintain internal consistency. As to the format of the references themselves, any style is acceptable as long as it is used consistently.

The documentation for `natbib` may be found at

`http://mirrors.ctan.org/macros/latex/contrib/natbib/natnotes.pdf`

Of note is the command `\citet`, which produces citations appropriate for use in inline text. For example,

```
\citet{hasselmo} investigated\dots
```

produces

Hasselmo, et al. (1995) investigated…

If you wish to load the `natbib` package with options, you may add the following before loading the `neurips_2022` package:

```
\PassOptionsToPackage{options}{natbib}
```

If `natbib` clashes with another package you load, you can add the optional argument `nonatbib` when loading the style file:

```
\usepackage[nonatbib]{neurips_2022}
```

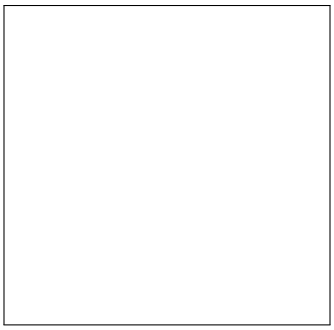

Figure 3: Sample figure caption.

As submission is double blind, refer to your own published work in the third person. That is, use "In the previous work of Jones et al. [4]," not "In our previous work [4]." If you cite your other papers that are not widely available (e.g., a journal paper under review), use anonymous author names in the citation, e.g., an author of the form "A. Anonymous."

9.2 Footnotes

Footnotes should be used sparingly. If you do require a footnote, indicate footnotes with a number[3] in the text. Place the footnotes at the bottom of the page on which they appear. Precede the footnote with a horizontal rule of 2 inches (12 picas).

Note that footnotes are properly typeset *after* punctuation marks.[4]

9.3 Figures

All artwork must be neat, clean, and legible. Lines should be dark enough for purposes of reproduction. The figure number and caption always appear after the figure. Place one line space before the figure caption and one line space after the figure. The figure caption should be lower case (except for first word and proper nouns); figures are numbered consecutively.

You may use color figures. However, it is best for the figure captions and the paper body to be legible if the paper is printed in either black/white or in color.

9.4 Tables

All tables must be centered, neat, clean and legible. The table number and title always appear before the table. See Table 2.

Place one line space before the table title, one line space after the table title, and one line space after the table. The table title must be lower case (except for first word and proper nouns); tables are numbered consecutively.

Note that publication-quality tables *do not contain vertical rules.* We strongly suggest the use of the `booktabs` package, which allows for typesetting high-quality, professional tables:

`https://www.ctan.org/pkg/booktabs`

This package was used to typeset Table 2.

10 Final instructions

Do not change any aspects of the formatting parameters in the style files. In particular, do not modify the width or length of the rectangle the text should fit into, and do not change font sizes (except perhaps in the **References** section; see below). Please note that pages should be numbered.

[3] Sample of the first footnote.

[4] As in this example.

Table 2: Sample table title

	Part	
Name	Description	Size (μm)
Dendrite	Input terminal	$\sim$100
Axon	Output terminal	$\sim$10
Soma	Cell body	up to 10^6

11 Preparing PDF files

Please prepare submission files with paper size "US Letter," and not, for example, "A4."

Fonts were the main cause of problems in the past years. Your PDF file must only contain Type 1 or Embedded TrueType fonts. Here are a few instructions to achieve this.

- You should directly generate PDF files using `pdflatex`.
- You can check which fonts a PDF files uses. In Acrobat Reader, select the menu Files>Document Properties>Fonts and select Show All Fonts. You can also use the program `pdffonts` which comes with `xpdf` and is available out-of-the-box on most Linux machines.
- The IEEE has recommendations for generating PDF files whose fonts are also acceptable for NeurIPS. Please see `http://www.emfield.org/icuwb2010/downloads/IEEE-PDF-SpecV32.pdf`
- `xfig` "patterned" shapes are implemented with bitmap fonts. Use "solid" shapes instead.
- The `\bbold` package almost always uses bitmap fonts. You should use the equivalent AMS Fonts:

  ```
  \usepackage{amsfonts}
  ```

  followed by, e.g., `\mathbb{R}`, `\mathbb{N}`, or `\mathbb{C}` for $\mathbb{R}$, $\mathbb{N}$ or $\mathbb{C}$. You can also use the following workaround for reals, natural and complex:

  ```
  \newcommand{\RR}{I\!\!R} %real numbers
  \newcommand{\Nat}{I\!\!N} %natural numbers
  \newcommand{\CC}{I\!\!\!\!C} %complex numbers
  ```

  Note that `amsfonts` is automatically loaded by the `amssymb` package.

If your file contains type 3 fonts or non embedded TrueType fonts, we will ask you to fix it.

11.1 Margins in LaTeX

Most of the margin problems come from figures positioned by hand using `\special` or other commands. We suggest using the command `\includegraphics` from the `graphicx` package. Always specify the figure width as a multiple of the line width as in the example below:

```
\usepackage[pdftex]{graphicx} ...
\includegraphics[width=0.8\linewidth]{myfile.pdf}
```

See Section 4.4 in the graphics bundle documentation (`http://mirrors.ctan.org/macros/latex/required/graphics/grfguide.pdf`)

A number of width problems arise when LaTeX cannot properly hyphenate a line. Please give LaTeX hyphenation hints using the `\-` command when necessary.

Acknowledgments and Disclosure of Funding

Use unnumbered first level headings for the acknowledgments. All acknowledgments go at the end of the paper before the list of references. Moreover, you are required to declare funding

(financial activities supporting the submitted work) and competing interests (related financial activities outside the submitted work). More information about this disclosure can be found at: `https://neurips.cc/Conferences/2022/PaperInformation/FundingDisclosure`.

Do **not** include this section in the anonymized submission, only in the final paper. You can use the `ack` environment provided in the style file to autmoatically hide this section in the anonymized submission.

References

References follow the acknowledgments. Use unnumbered first-level heading for the references. Any choice of citation style is acceptable as long as you are consistent. It is permissible to reduce the font size to `small` (9 point) when listing the references. Note that the Reference section does not count towards the page limit.

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

[2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer–Verlag.

[3] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section 7.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? **[TODO]**
   (b) Did you describe the limitations of your work? **[TODO]**
   (c) Did you discuss any potential negative societal impacts of your work? **[TODO]**
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? **[TODO]**
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? **[TODO]**
   (b) Did you include complete proofs of all theoretical results? **[TODO]**
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? **[TODO]**
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? **[TODO]**

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? **[TODO]**

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? **[TODO]**

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? **[TODO]**

(b) Did you mention the license of the assets? **[TODO]**

(c) Did you include any new assets either in the supplemental material or as a URL? **[TODO]**

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? **[TODO]**

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? **[TODO]**

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? **[TODO]**

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? **[TODO]**

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? **[TODO]**

A Appendix

Optionally include extra information (complete proofs, additional experiments and plots) in the appendix. This section will often be part of the supplemental material.

References

[1] J. Berg and K. Nyström. Neural network augmented inverse problems for pdes. *arXiv preprint arXiv:1712.09685*, 2017.

[2] D. A. Bezgin, A. B. Buhendwa, and N. A. Adams. Jax-fluids: A fully-differentiable high-order computational fluid dynamics solver for compressible two-phase flows. *arXiv preprint arXiv:2203.13760*, 2022.

[3] K. Bhattacharya, B. Hosseini, N. B. Kovachki, and A. M. Stuart. Model reduction and neural networks for parametric pdes. *arXiv preprint arXiv:2005.03180*, 2020.

[4] D. Bochkov and F. Gibou. Solving elliptic interface problems with jump conditions on cartesian grids. *Journal of Computational Physics*, 407:109269, 2020.

[5] D. Bochkov and F. Gibou. A non-parametric shape optimization approach for solving inverse problems in directed self-assembly of block copolymers. *arXiv preprint arXiv:2112.09615*, 2021.

[6] D. Bochkov, T. Pollock, and F. Gibou. Sharp-interface simulations of multicomponent alloy solidification. *arXiv preprint arXiv:2112.08650*, 2021.

[7] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[8] T. Chen and H. Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995.

[9] L. O. Chua and L. Yang. Cellular neural networks: Applications. *IEEE Transactions on circuits and systems*, 35(10):1273–1290, 1988.

[10] L. O. Chua and L. Yang. Cellular neural networks: Theory. *IEEE Transactions on circuits and systems*, 35(10):1257–1272, 1988.

[11] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996.

[12] N. Dal Santo, S. Deparis, and L. Pegolotti. Data driven approximation of parametrized pdes by reduced basis and neural networks. *Journal of Computational Physics*, 416:109550, 2020.

[13] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter. End-to-end differentiable physics for learning and control. *Advances in neural information processing systems*, 31, 2018.

[14] K. Galatsis, K. L. Wang, M. Ozkan, C. S. Ozkan, Y. Huang, J. P. Chang, H. G. Monbouquette, Y. Chen, P. Nealey, and Y. Botros. Patterning and templating for nanoelectronics. *Advanced Materials*, 22(6):769–778, 2010.

[15] D. Gobovic and M. Zaghloul. Design of locally connected cmos neural cells to solve the steady-state heat flow problem. In *Proceedings of 36th Midwest Symposium on Circuits and Systems*, pages 755–757. IEEE, 1993.

[16] P. Holl, V. Koltun, and N. Thuerey. Learning to control pdes with differentiable physics. *arXiv preprint arXiv:2001.07457*, 2020.

[17] P. Holl, V. Koltun, K. Um, and N. Thuerey. phiflow: A differentiable pde solving framework for deep learning via physical simulations. In *NeurIPS Workshop*, volume 2, 2020.

[18] D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21), 2021.

[19] I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

[20] H. Lee and I. S. Kang. Neural algorithm for solving differential equations. *Journal of Computational Physics*, 91(1):110–131, 1990.

[21] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[22] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

[23] L. Lu, P. Jin, and G. E. Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

[24] P. Y. Lu, S. Kim, and M. Soljačić. Extracting interpretable physical parameters from spatiotemporal systems using unsupervised learning. *Physical Review X*, 10(3):031056, 2020.

[25] M. Mirzadeh, M. Theillard, and F. Gibou. A second-order discretization of the nonlinear Poisson-Boltzmann equation over irregular geometries using non-graded adaptive Cartesian grids. *Journal of Computational Physics*, 230(5):2125–2140, Mar. 2011.

[26] P. Mistani, A. Guittet, D. Bochkov, J. Schneider, D. Margetis, C. Ratsch, and F. Gibou. The island dynamics model on parallel quadtree grids. *Journal of Computational Physics*, 361:150–166, 2018.

[27] P. Mistani, A. Guittet, C. Poignard, and F. Gibou. A parallel voronoi-based approach for mesoscale simulations of cell aggregate electropermeabilization. *Journal of Computational Physics*, 380:48–64, 2019.

[28] G. Y. Ouaknin, N. Laachi, K. Delaney, G. H. Fredrickson, and F. Gibou. Level-set strategy for inverse dsa-lithography. *Journal of Computational Physics*, 375:1159–1178, 2018.

[29] S. Pakravan, P. A. Mistani, M. A. Aragon-Calvo, and F. Gibou. Solving inverse-pde problems with physics-aware neural networks. *Journal of Computational Physics*, 440:110414, 2021.

[30] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019.

[31] M. Raissi, P. Perdikaris, and G. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[32] C. J. Shallue, J. Lee, J. Antognini, J. Sohl-Dickstein, R. Frostig, and G. E. Dahl. Measuring the effects of data parallelism on neural network training. *arXiv preprint arXiv:1811.03600*, 2018.

[33] K. A. Sharp and B. Honig. Calculating total electrostatic energies with the nonlinear poisson-boltzmann equation. *Journal of Physical Chemistry*, 94(19):7684–7692, 1990.

[34] J. Sirignano and K. Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

[35] M. Theillard, F. Gibou, and T. Pollock. A sharp computational method for the simulation of the solidification of binary alloys. *Journal of scientific computing*, 63(2):330–354, 2015.

[36] K. Um, R. Brand, Y. R. Fei, P. Holl, and N. Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. *Advances in Neural Information Processing Systems*, 33:6111–6122, 2020.

[37] S. Wang, X. Yu, and P. Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

[38] B. Yu et al. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.