Emulating Coronal Field Models with Physics-Informed Neural Nets

#031

N. H. Mathews and B. J. Thompson

How Do We Include Neural Nets in a Physical Model Workflow?

- Neural nets tend not to have strong convergence guarantees with respect to training or testing data. We cannot be confident even well-trained neural nets will model physical systems self-consistently.
  - Some varieties can provide reasonably realistic solutions to even complicated equations very quickly once trained.
  - This makes them the perfect tool to fill any need for fast, approximate solutions to nonlinear differential equations.
- Exactly such a need exists in inverse modeling.
  - Solving an inverse problem requires many evaluations of a parameterized forward model at different points in the parameter space.
  - Synthetic observations developed based on those model solutions are then compared to true observations, and the realization of the model which minimizes the difference between the two sets of observations, or the cost function, is determined to be the true state of the magnetic field.
- [Mathews, Flyer and Gibson 2022] presents a forward, plasma-parameterized magnetohydrostatic (MHS) solver that could be used in such a framework, but it is too slow to make the actual inversion feasible.
  - Simply speaking, it would take too long to run the model as many times as would be required to build an accurate realization of the true magnetic field that minimizes the cost function for a given true observation set.
- So we use the neural net first, constructing many quick, approximate MHS solutions to sample the parameter space.
  - Near where we expect the cost function minimum to be, we then run the full MHS model only a few times.

Neural Net Architecture

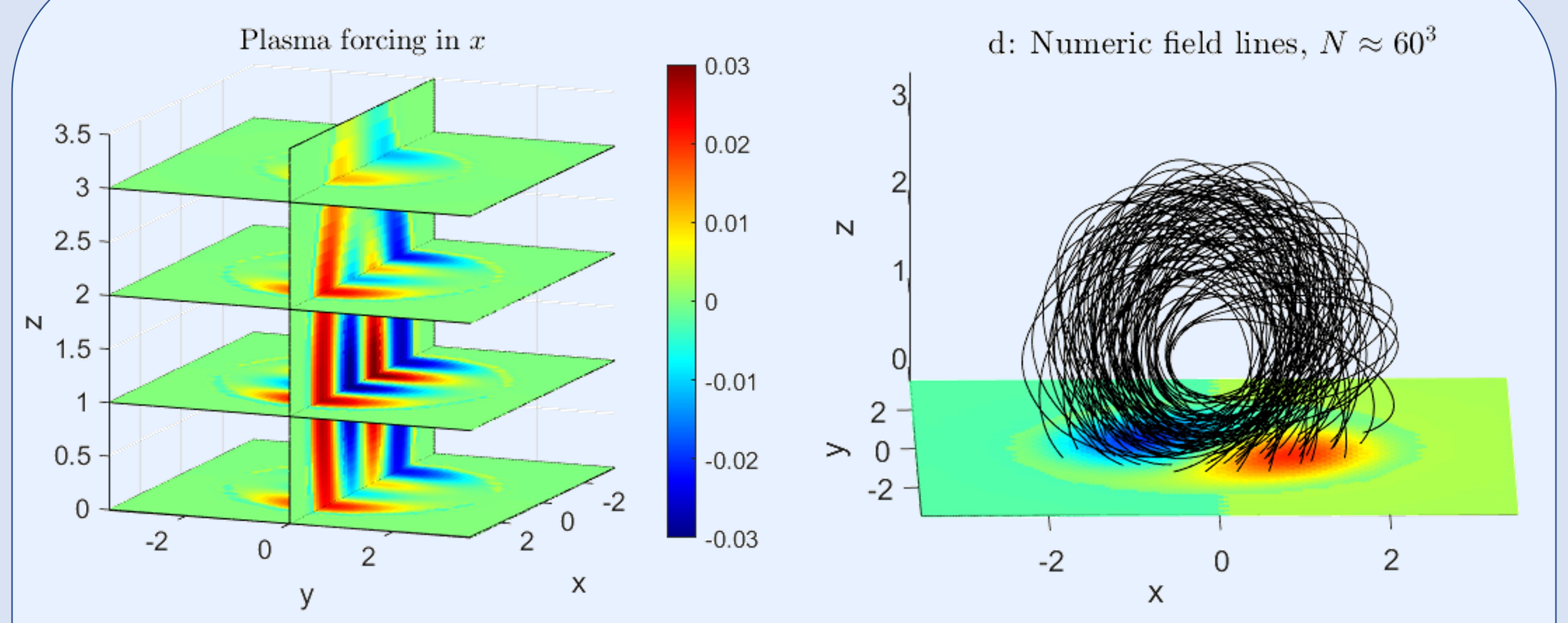
- Our neural net is built out of a series of convolutional Fourier layers, c.f. [Li et al 2021].
  - To preserve the meshfree property of the [Mathews, Flyer and Gibson 2022] solver and also allow propagation of the spatial derivatives through the layers for the Physics-Informed training component, we use the Non-Uniform Fast Fourier Transform, c.f. [Barnett, Magland and af Klinteberg 2019].
- The first convolutional layer propagates boundary data to the 3D nodepoints, and then that representation is further convolved with plasma forcing data to determine the magnetic field.

Training the Neural Net 1: Physics Information

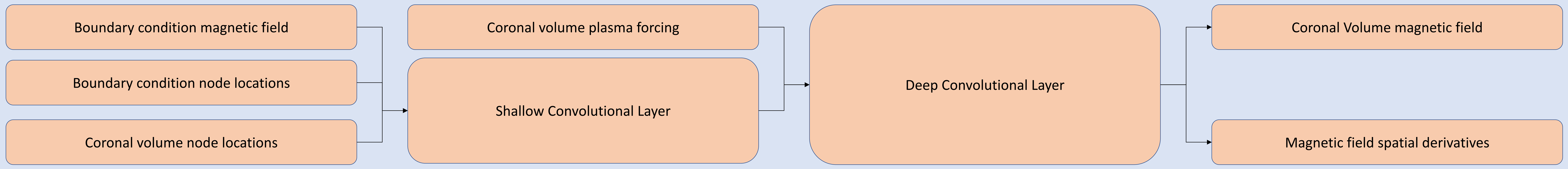
- We build the neural net as one which is physics-informed (Physics-Informed Neural Net, or PINN).
  - By keeping track of the gradients of the output magnetic field with respect to computational node location, we can construct spatial field derivatives, and so compute MHS balance of the magnetic field very cheaply.
  - We can include this difference from MHS balance as an additional objective in the training loop.
- Not only does this provide the neural net optimization scheme with better information as to how to reduce the objective vs ground truth in the boutique training cases, it *also* allows us to train on un-precomputed parameter vectors.
  - We can drastically increase the size of our training dataset, and potentially the results of our model, by using new, semirandom parameter values and setting the objective function to a measure of the MHS residual.

Training the Neural Net 2: Boutique Solutions

- MHS is a difficult partial differential equation for any reasons.
  - It can produce nonunique solutions given boundary conditions if the interior plasma forcing is weak.
  - If we are sampling cost functions produced by two different methods, they may produce very different cost functions even when provided the same parameter vector.
  - Important to not just train the neural net to produce nearly-MHS fields, but rather to emulate the particular MHS solver we intend to use in later refinement.
- We have produced a training dataset from the [Mathews, Flyer and Gibson 2022] solver containing samples with a variety of parameter-space values. The first part of the training data provided to the model consists of this dataset.



Parameters and magnetic field linesfor one such boutique training solution from the [Mathews, Flyer and Gibson 2022] solver


Boundary condition magnetic field
Boundary condition node locations
Coronal volume node locations
Coronal volume plasma forcing
Shallow Convolutional Layer
Deep Convolutional Layer
Coronal Volume magnetic field
Magnetic field spatial derivatives

Intermediate Results

- By constructing the neural net as we have, as a series of convolutional layers, we can deduce scientifically interpretable results from the layer weights
  - This is normally very challenging to do with deep learning approaches.
- The shape of the first convolutional kernel describes the propagation of photospheric boundary information into the upper corona.
- Relative weights in the layer immediately following the concatenation of plasma forcing is indicative of the importance of plasma forcing to the final solution.

Forthcoming Work

- The next step is the actual implementation of the inversion framework, and the use of the neural net as an emulator to enable that implementation.
  - We will use a trust-region-style derivative-free global optimization method, such as that proposed in [Dalmasse et al 2016].
  - The synthetic observables will be computed by FORWARD [Gibson et al 2016].
- This variety of framework was proposed and tested with a simple model in [Dalmasse et al 2019], but the more generalized model we plan here is expected to provide much-improved results that can be generalized to real-sun images.

More Information

Mathews, Flyer and Gibson paper:
https://doi.org/10.1016/j.jcp.2022.111214
Mathews, Flyer and Gibson code:
https://github.com/apt-get-nat/RBF-MHS
Dalmasse et al 2019 paper:
https://doi.org/10.3847/1538-4357/ab1907

Code for this poster is forthcoming, and will be posted at
https://github.com/apt-get-nat/

Digital copy of this poster