Reproducing results in
Probing the solar coronal magnetic field with physics-informed neural networks
(Jarolim et al. 2022, preprint)

using

Neural Network Force-Free magnetic field extrapolation - NF2
(https://github.com/RobertJaro/NF2)

Mingyu Jeon

Methods

```
bin = 2
spatial_norm = 160
height = 160
b_norm = 2500
d_slice = [66, 658, 9, 377] # crop

dim = 256

lambda_div = 0.1
lambda_ff = 0.1
iterations = 10e4
iterations = int(iterations)
decay_iterations = 5e4
decay_iterations = int(decay_iterations)
batch_size = 1e4
batch_size = int(batch_size)
log_interval = 1e4
log_interval = int(log_interval)
validation_interval = 1e4
validation_interval = int(validation_interval)
potential = True
```

We apply our method (using $\lambda_{div/ff} = 0.1$) to 601 consecutive magnetograms at a time cadence of 12 minutes, covering the time period from 2011-02-12 00:00 to 2011-02-17 00:00. In Fig. 3, we show the cor-

For our model training we use the Adam optimizer with beta of (0.9, 0.999) and apply gradient clipping of 0.1. We train our model for 100,000 iterations until we reach convergence. We found that prioritizing the boundary condition at the beginning of the training leads to a faster convergence. We set $\lambda_{B0} = 1000$ and exponentially decay it to 1 over the first 50,000 iterations. Similarly, we exponentially decay the learning rate from $5e-4$ to $5e-5$ over the first 70,000 iterations. The choice of λ_{ff} and λ_{div} is discussed in the main text.

use 100,000 iterations
instead of 70,000 iterations

```
158         scheduler = ExponentialLR(opt, gamma=(5e-5 / 5e-4) ** (1 / total_iterations))
```

= 100,000

Probably it is the reason my result's energy is higher than others because only this parameter is different from the preprint.

```
126             lambda_B = 1000 if decay_iterations else 1
127             history = {'iteration': [], 'height': [],
128                        'b_loss': [], 'divergence_loss': [], 'force_loss': [], 'sigma_angle': []}
129
130         self.opt = opt
131         self.start_iteration = start_iteration
132         self.history = history
133         self.lambda_B = lambda_B
134         self.lambda_B_decay = (1 / 1000) ** (1 / decay_iterations) if decay_iterations is not None else 1
```

```
209             # update training parameters
210             if self.lambda_B > 1:
211                 self.lambda_B *= self.lambda_B_decay
212             if scheduler.get_last_lr()[0] > 5e-5:
213                 scheduler.step()
```

https://github.com/RobertJaro/NF2/blob/main/nf2/train/trainer.py

Methods

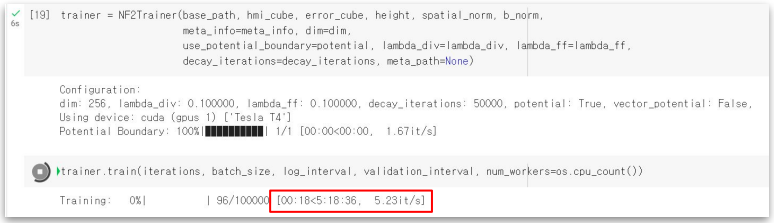
For our experiments we used four NVIDIA V100 GPUs. The initial simulation is performed in about 50 minutes (about 2 hours on a single V100 GPU). The consecutive simulations require approximately 1.2 minutes per extrapolation (including data pre-processing and calculating the potential field boundary). This

Tesla V100-SXM2-32GB	Preprint
Tesla T4	Colab (usually)

https://browser.geekbench.com/opencl-benchmarks

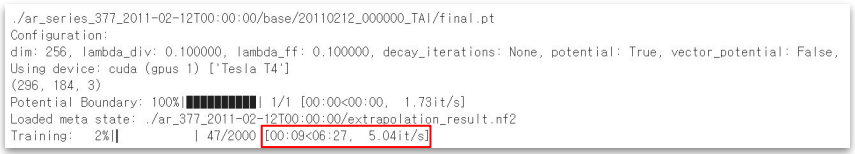
~ x2.3 => 2h * 2.3 ~ 4.6h

Initial training ~ 5.5h



Series training ~ 6.5min per extrapolation

Total ~ 600*6.5min ~ 65h



Methods

Resuming initial training using checkpoint.pt for Colab

```
[ ]  trainer = NF2Trainer(base_path, hmi_cube, error_cube, height, spatial_norm, b_norm,
                          meta_info=meta_info, dim=dim,
                          use_potential_boundary=potential, lambda_div=lambda_div, lambda_ff=lambda_ff,
                          decay_iterations=decay_iterations, meta_path=None)

     Configuration:
     dim: 256, lambda_div: 0.100000, lambda_ff: 0.100000, decay_iterations: 50000, potential: True, vector_potential: False,
     Using device: cuda (gpus 1) ['Tesla T4']
     Potential Boundary: 100%|██████████| 1/1 [00:02<00:00,  2.12s/it]
     Resuming training from iteration 40000

[ ]  trainer.train(iterations, batch_size, log_interval, validation_interval, num_workers=os.cpu_count())

     Training:  17%|██        | 9999/60000 [33:45<2:50:07,  4.90it/s][Iteration 050000/060000] [B-Field: 0.00115485; Div: 0.00468535; For: 0.00783048] [0:33:45.940693]
```

Free energy ~ 3.5min

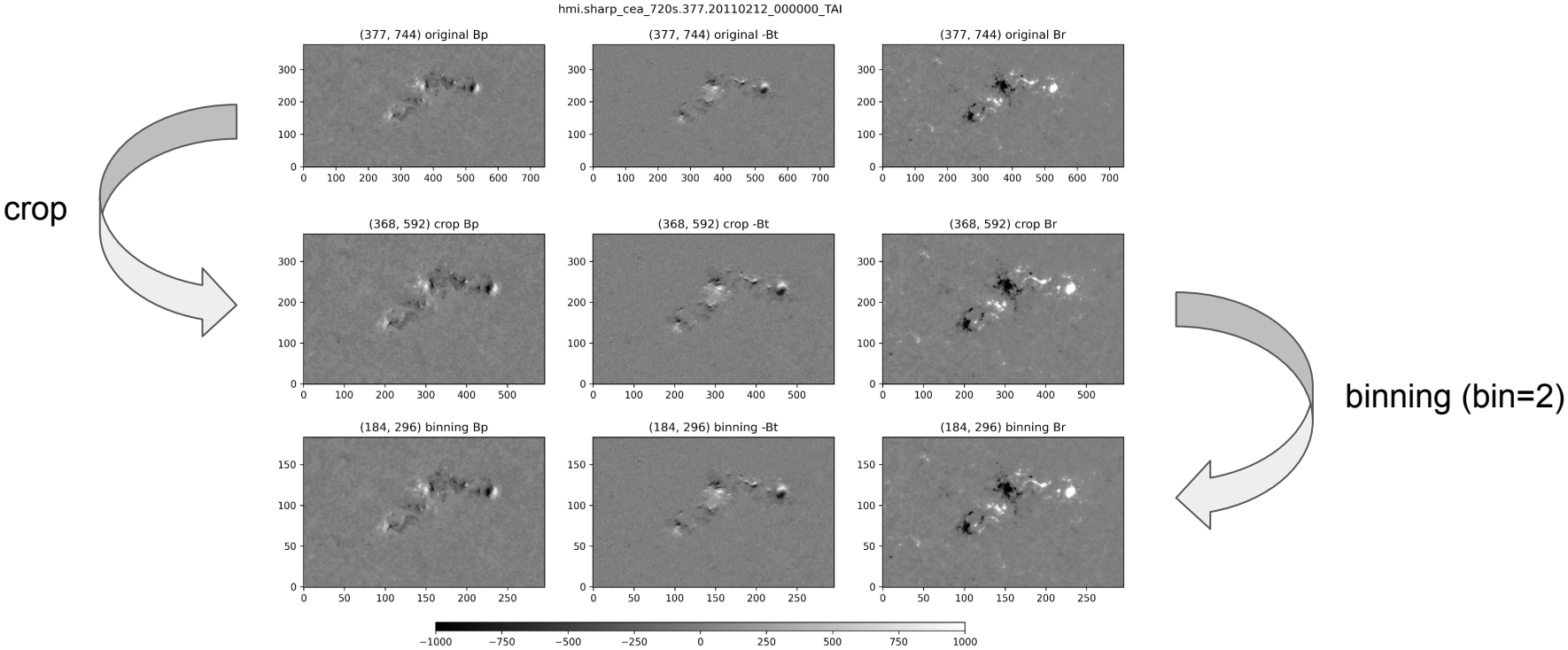
Total ~ 600*3.5min ~ 35h

```
Potential Field: 100%|██████████| 8715/8715 [03:41<00:00, 39.33it/s]
./ar_series_377_2011-02-12T00:00:00/base/20110215_233600_TAI/extrapolation_result.nf2
./ar_series_377_2011-02-12T00:00:00/eval_free_energy/20110215_233600_TAI.npy
100%|██████████| 8715/8715 [00:09<00:00, 872.19it/s]
Potential Field: 100%|██████████| 8715/8715 [03:41<00:00, 39.28it/s]
./ar_series_377_2011-02-12T00:00:00/base/20110215_234800_TAI/extrapolation_result.nf2
./ar_series_377_2011-02-12T00:00:00/eval_free_energy/20110215_234800_TAI.npy
100%|██████████| 8715/8715 [00:10<00:00, 848.37it/s]
Potential Field: 100%|██████████| 8715/8715 [03:40<00:00, 39.47it/s]
./ar_series_377_2011-02-12T00:00:00/base/20110216_000000_TAI/extrapolation_result.nf2
./ar_series_377_2011-02-12T00:00:00/eval_free_energy/20110216_000000_TAI.npy
100%|██████████| 8715/8715 [00:09<00:00, 959.51it/s]
Potential Field: 100%|██████████| 8715/8715 [03:41<00:00, 39.28it/s]
./ar_series_377_2011-02-12T00:00:00/base/20110216_001200_TAI/extrapolation_result.nf2
./ar_series_377_2011-02-12T00:00:00/eval_free_energy/20110216_001200_TAI.npy
100%|██████████| 8715/8715 [00:10<00:00, 852.87it/s]
Potential Field: 100%|██████████| 8715/8715 [03:41<00:00, 39.39it/s]
./ar_series_377_2011-02-12T00:00:00/base/20110216_002400_TAI/extrapolation_result.nf2
./ar_series_377_2011-02-12T00:00:00/eval_free_energy/20110216_002400_TAI.npy
100%|██████████| 8715/8715 [00:10<00:00, 819.73it/s]
Potential Field:  38%|████      | 3308/8715 [01:23<02:16, 39.49it/s]
```

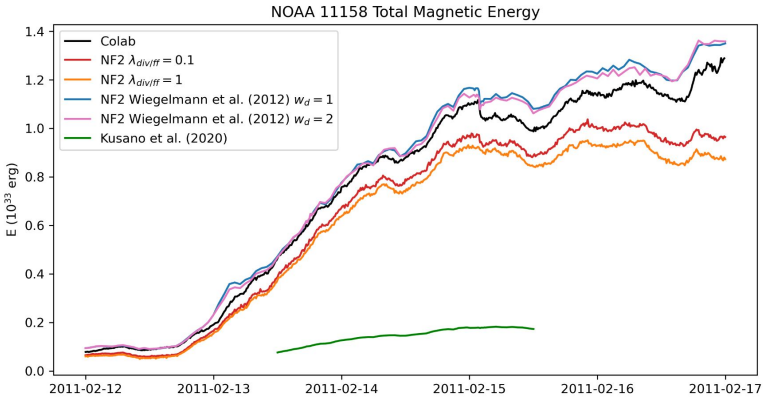
NOAA 11158

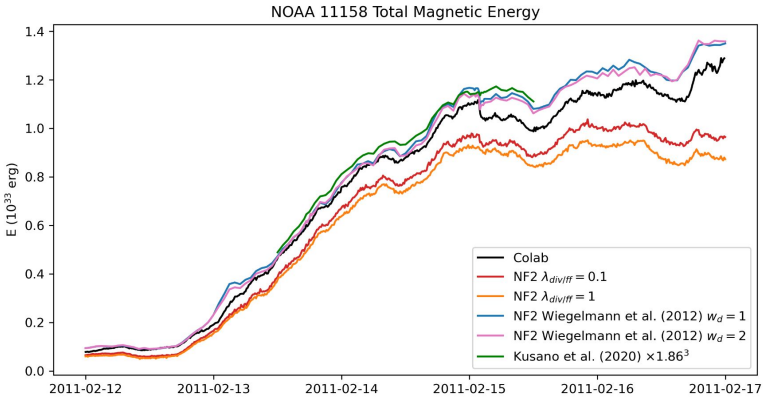
2011-02-12 00:00:00

For observations of NOAA 11158 (SHARP 377), we crop the active region patches to x [66:658] and y [9:377] pixels (prior to the binning), in correspondence with the optimization-based (3, see Appendix A) NLFF modeling used for comparison (for a similar model time series see, e.g., 27).



NOAA 11158

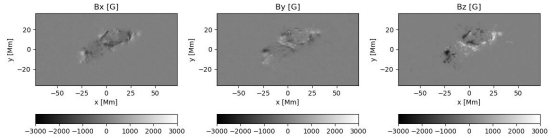


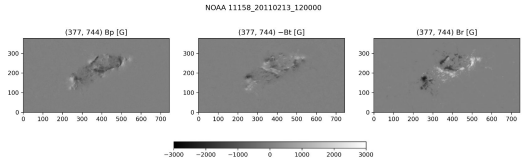


Colab → dV = (360e5 * 2)**3

NF2 → data from csv files provided by authors
https://kanzelhohe.uni-graz.at/nf2/

Kusano et al. 2020 → dV = dx*dy*dz
https://hinode.isee.nagoya-u.ac.jp/nlfff_database/





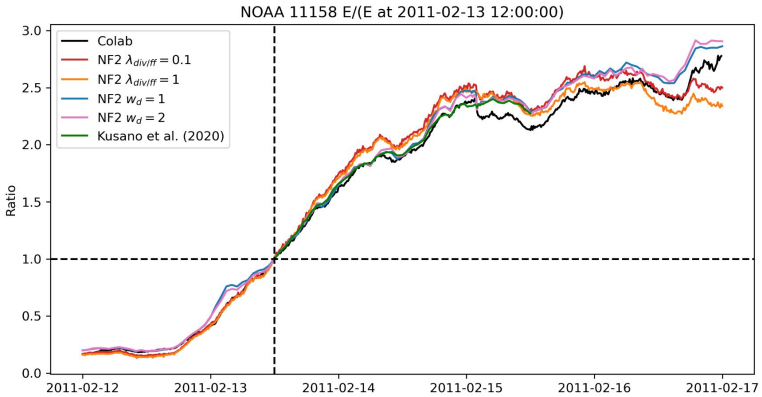
A physics-based method that can predict imminent large solar flares
K Kusano, T Iju, Y Bamba, S Inoue - Science, 2020 - science.org
Solar flares are highly energetic events in the Sun's corona that affect Earth's space weather. The mechanism that drives the onset of solar flares is unknown, hampering efforts to forecast them, which mostly rely on empirical methods. We present the κ-scheme, a physics-based model to predict large solar flares through a critical condition of magnetohydrodynamic instability, triggered by magnetic reconnection. Analysis of the largest (X-class) flares from 2008 to 2019 (during solar cycle 24) shows that the κ-scheme predicts most imminent large …
☆ 저장 99 인용 48회 인용 관련 학술자료 전체 6개의 버전

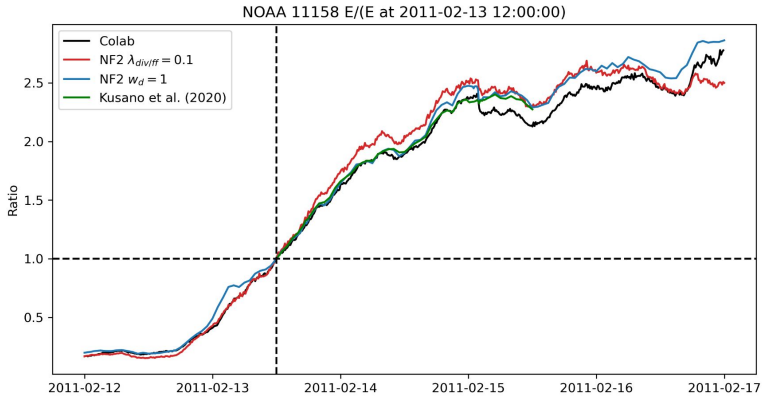
SHARP Xsize = 744 pixel * 0.36 Mm/pixel ~ 268 Mm
Kusano Xsize ~ 144 Mm

SHARP Ysize = 377 pixel * 0.36 Mm/pixel ~ 136 Mm
Kusano Ysize ~ 73 Mm

=> bin ~ 1.86

NOAA 11158

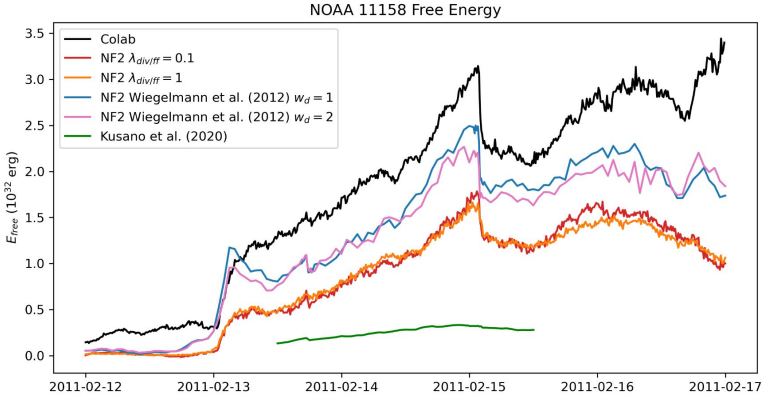


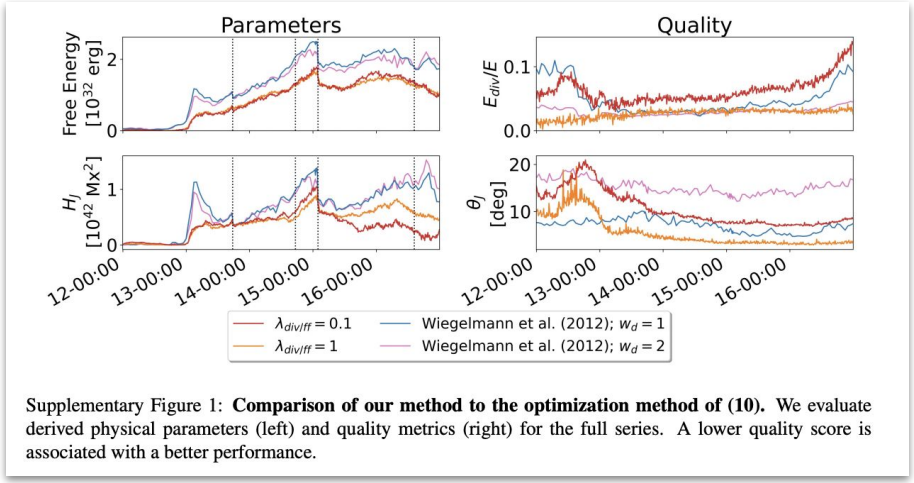


Ratio = energy / (energy at 2011-02-13 12:00:00)

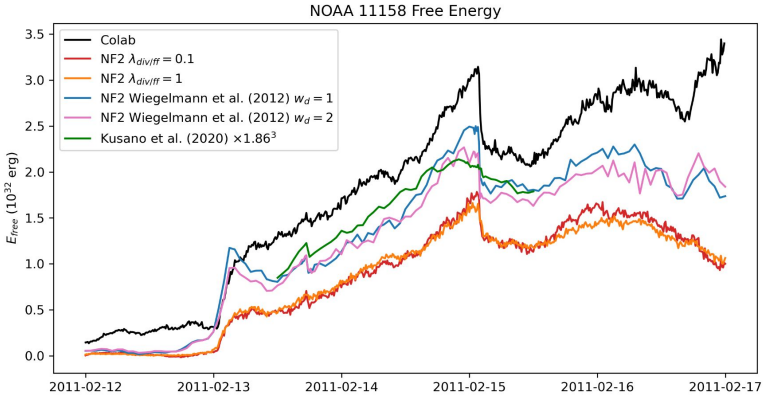
2011-02-13 12:00:00 = the first time in Kusano's data

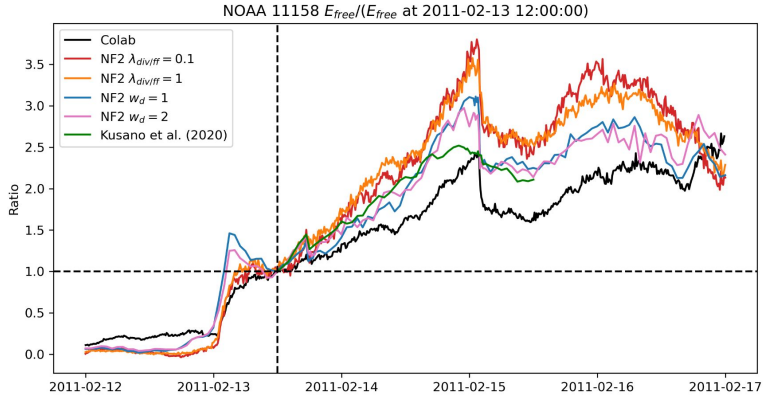
NOAA 11158





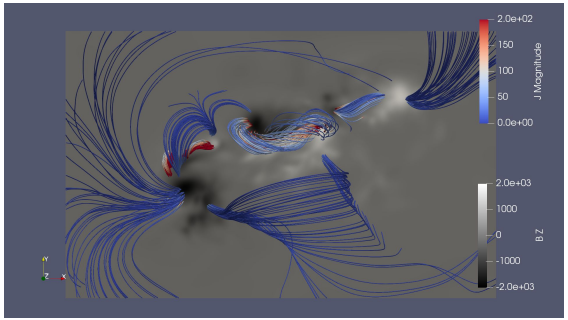
Supplementary Figure 1: **Comparison of our method to the optimization method of (10).** We evaluate derived physical parameters (left) and quality metrics (right) for the full series. A lower quality score is associated with a better performance.



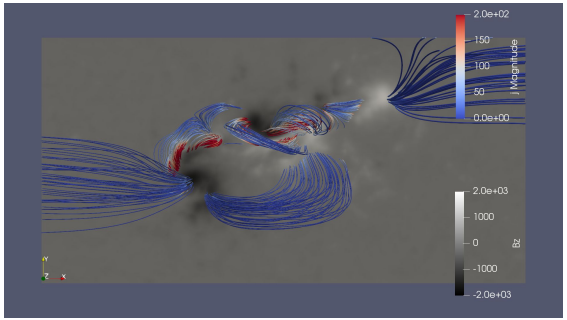


NOAA 11158

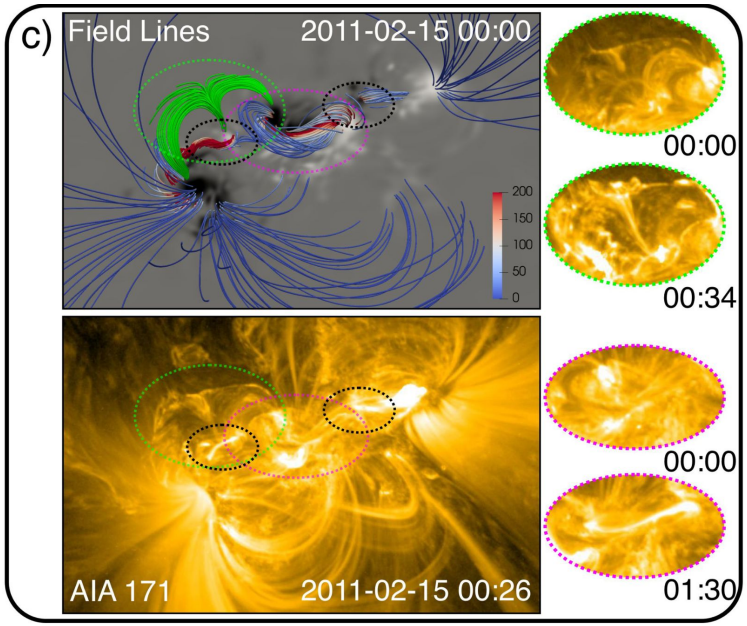
Colab



Kusano

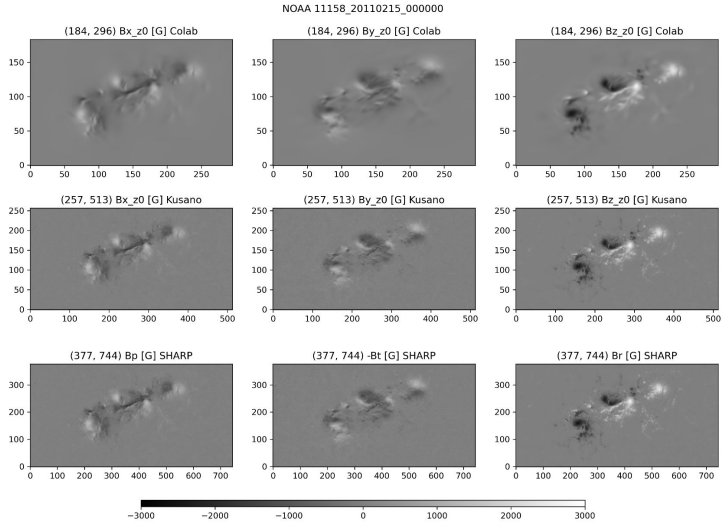


ParaView
https://www.paraview.org/



magnetic energy depletion, as estimated by our model, is in agreement with the position of the flare event. c) Magnetic field line traces of the central components of the active region. The colors of the field lines indicate the current density. The two flux ropes associated to pores are encircled in black, the central flux rope in pink, and the eruptive structure associated to the null point in green. A movie of the full series is included in the supplementary material (Movie 1).

NOAA 11158



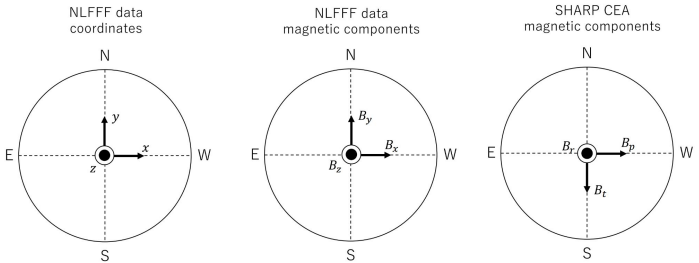
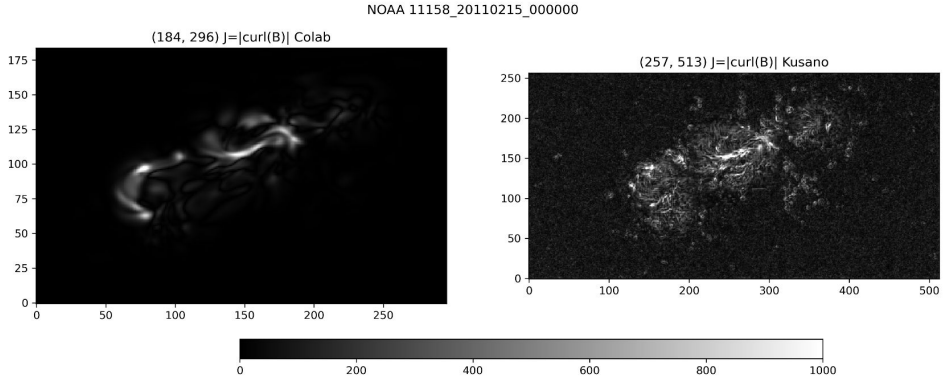
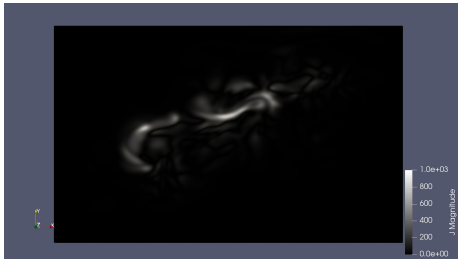


Figure from ISEE NLFFF database homepage (https://hinode.isee.nagoya-u.ac.jp/nlfff_database/) by Kusano et al. (2020)

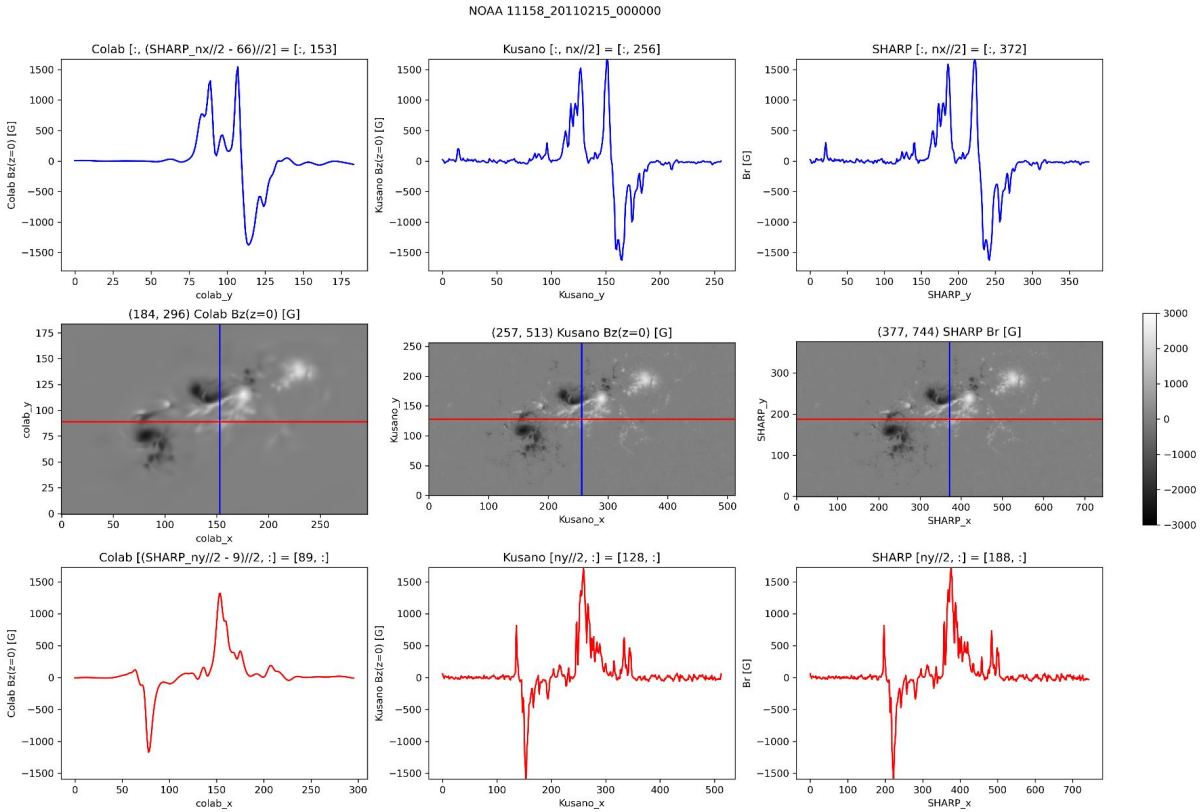


Colab



Kusano

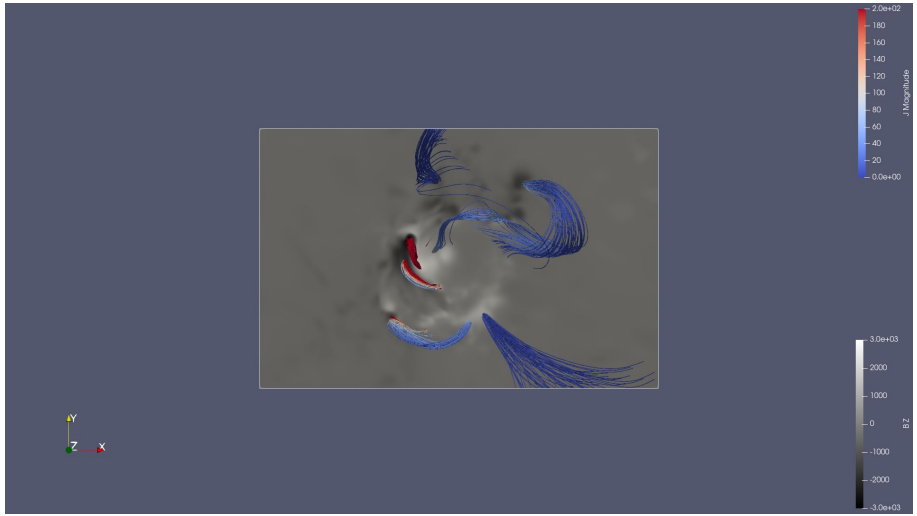
NOAA 11158


NOAA 11158_20110215_000000
Colab [:, (SHARP_nx//2 - 66)//2] = [:, 153]
Kusano [:, nx//2] = [:, 256]
SHARP [:, nx//2] = [:, 372]
Colab Bz(z=0) [G]
Kusano Bz(z=0) [G]
Br [G]
colab_y
Kusano_y
SHARP_y
(184, 296) Colab Bz(z=0) [G]
(257, 513) Kusano Bz(z=0) [G]
(377, 744) SHARP Br [G]
colab_x
Kusano_x
SHARP_x
Colab [(SHARP_ny//2 - 9)//2, :] = [89, :]
Kusano [ny//2, :] = [128, :]
SHARP [ny//2, :] = [188, :]

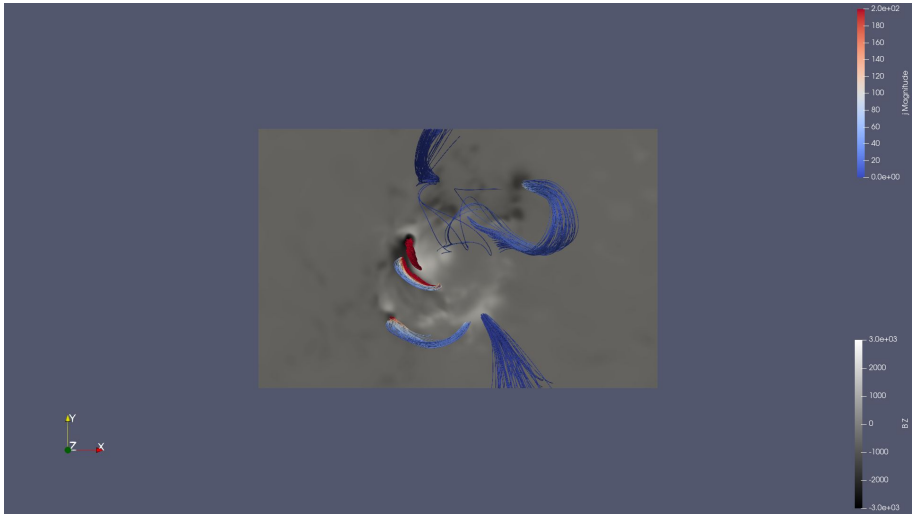
NOAA 12673

2017-09-06 08:36:00

Initial training in Colab ~ 5.5h

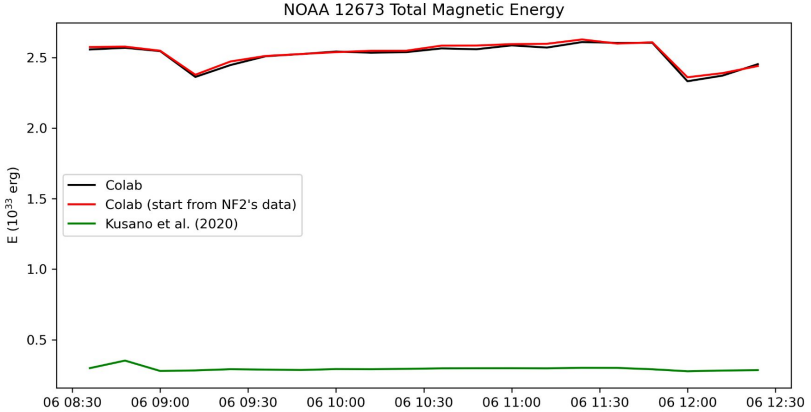


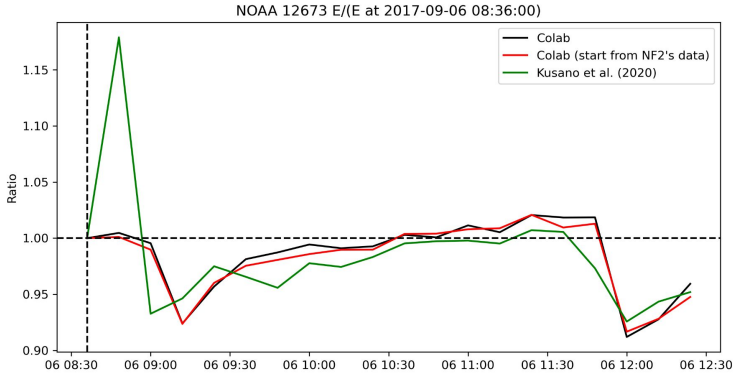
Data provided by authors

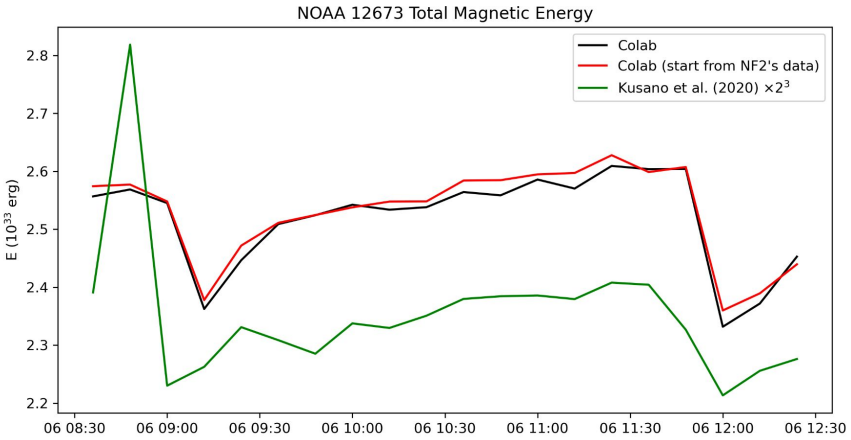


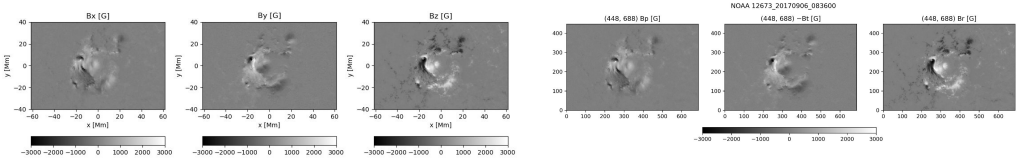
NOAA 12673

Series Training ~ 20*6.5min ~ 2.2h







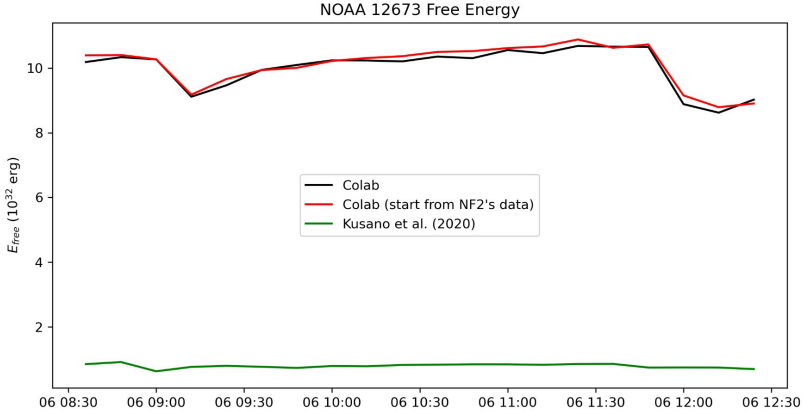


SHARP Xsize = 688 pixel * 0.36 Mm/pixel ~ 248 Mm
Kusano Xsize ~ 123 Mm

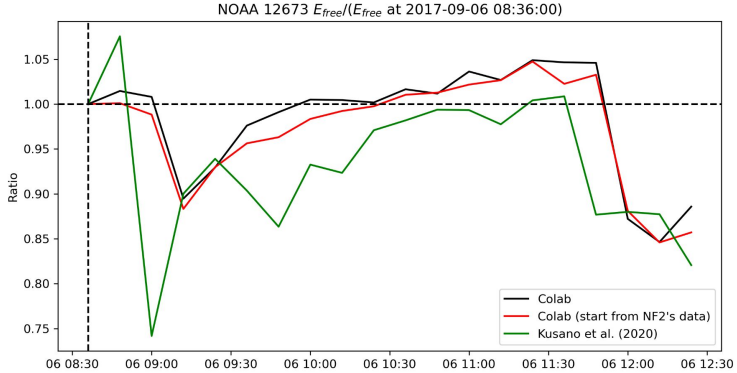
SHARP Ysize = 377 pixel * 0.36 Mm/pixel ~ 161 Mm
Kusano Ysize ~ 80 Mm

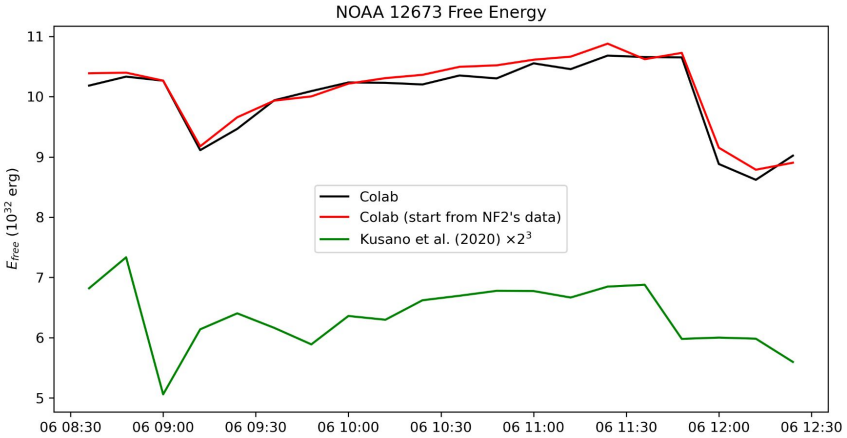
=> bin ~ 2

NOAA 12673



Free energy ~ 20*3.5min ~ 1.2h

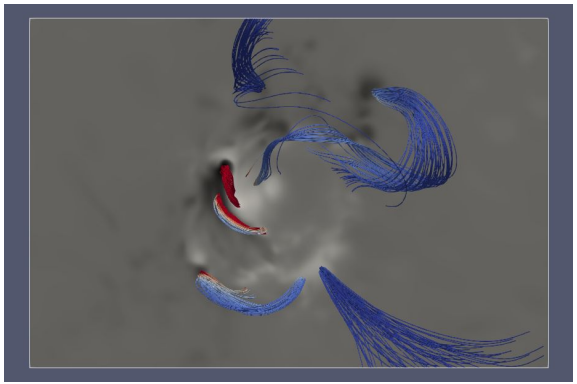




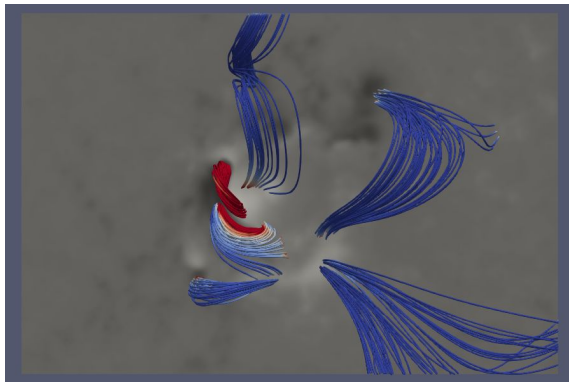
NOAA 12673

2017-09-06 08:36:00

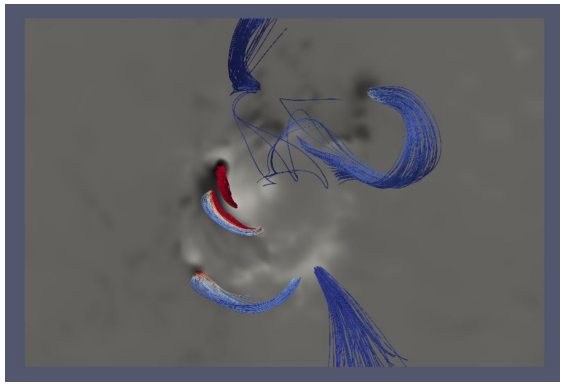
Colab



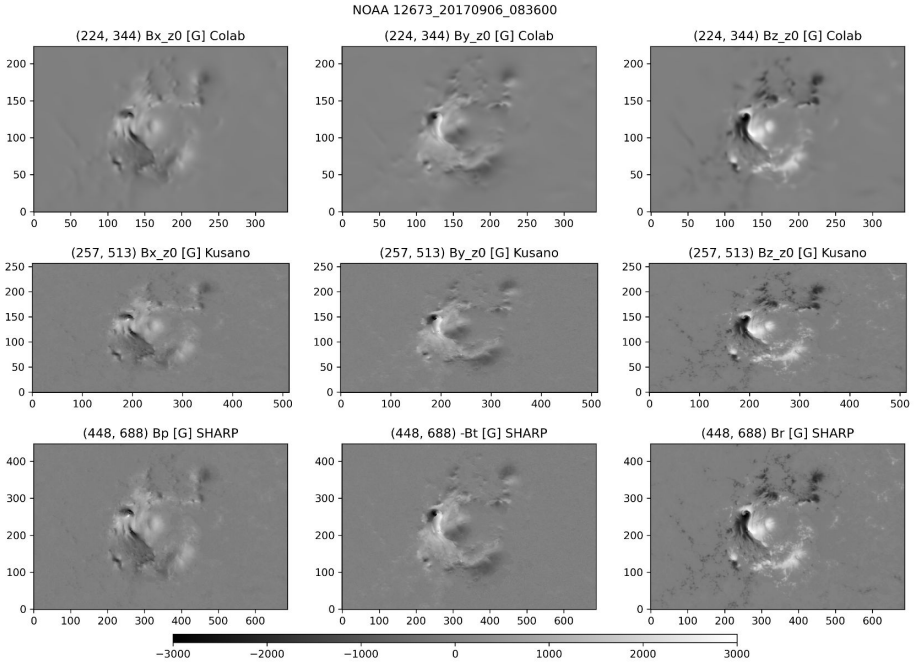
Kusano

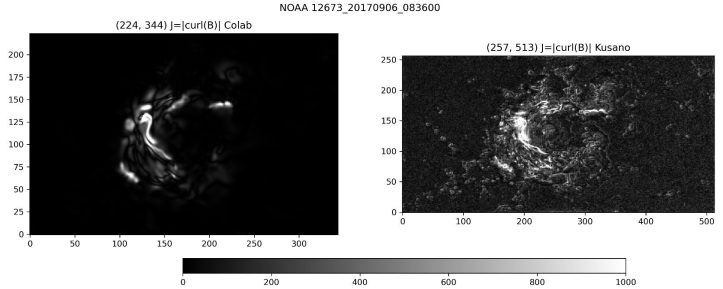


NF2's data

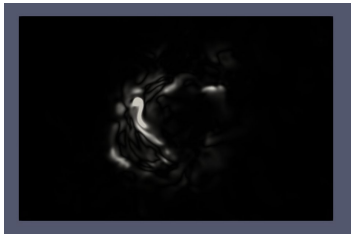


NOAA 12673

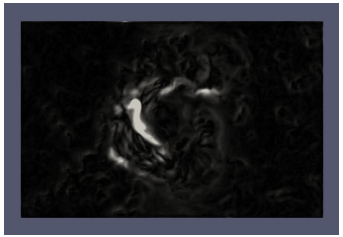




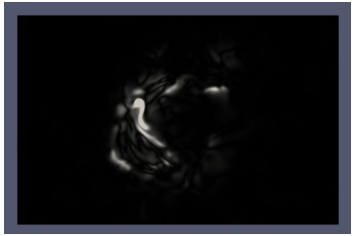
Colab



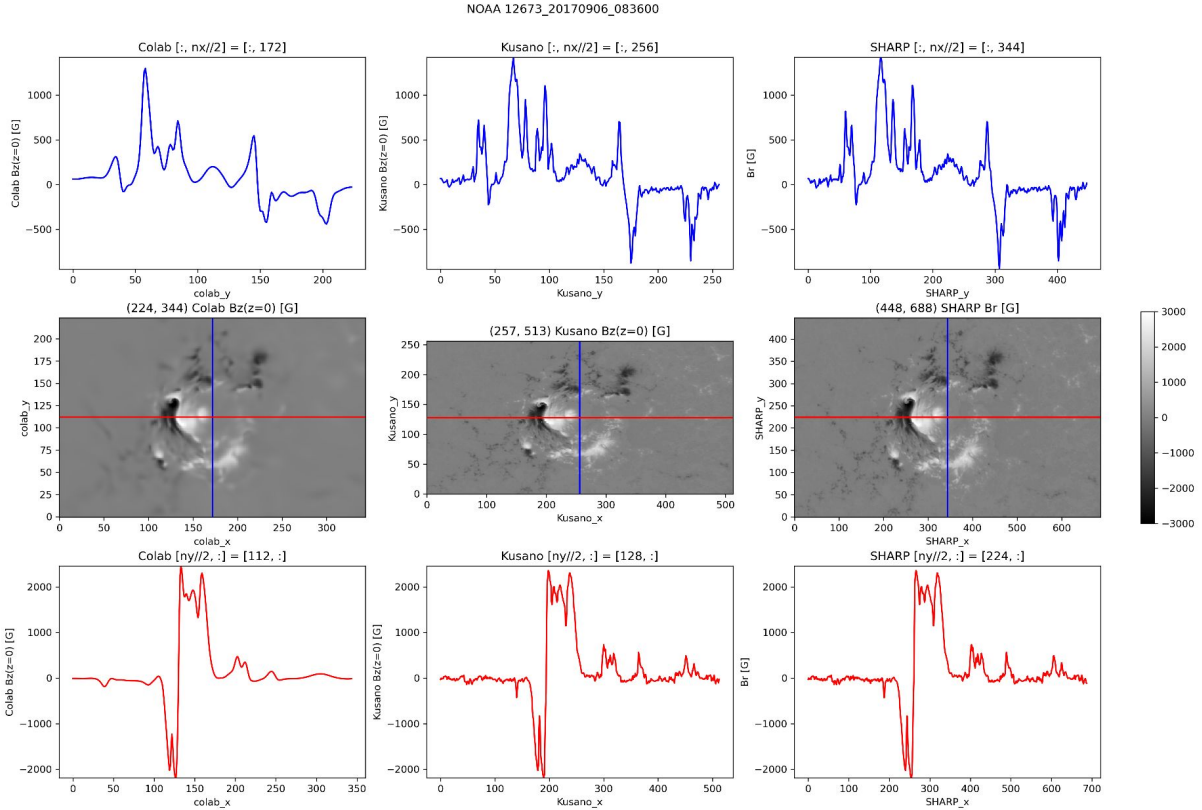
Kusano



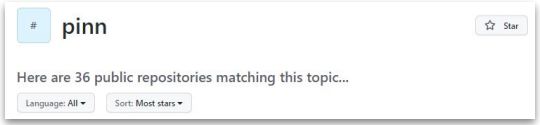
NF2's data



NOAA 12673


NOAA 12673_20170906_083600
Colab [:, nx//2] = [:, 172]
Kusano [:, nx//2] = [:, 256]
SHARP [:, nx//2] = [:, 344]
Colab Bz(z=0) [G]
Kusano Bz(z=0) [G]
Br [G]
colab_y
Kusano_y
SHARP_y
(224, 344) Colab Bz(z=0) [G]
(257, 513) Kusano Bz(z=0) [G]
(448, 688) SHARP Br [G]
colab_x
Kusano_x
SHARP_x
Colab [ny//2, :] = [112, :]
Kusano [ny//2, :] = [128, :]
SHARP [ny//2, :] = [224, :]

Future work - External libraries for PINN



NeuralPde: Automating physics-informed neural networks (pinns) with error approximations
K Zubov, Z McCarthy, Y Ma, F Calisto… - arXiv preprint arXiv …, 2021 - arxiv.org
… In this manuscript we detail the inner workings of **NeuralPDE**.jl … We showcase how **NeuralPDE** uses a purely symbolic … formulated and solved with **NeuralPDE**. Together this manuscript …
☆ 저장 99 인용 36회 인용 관련 학술자료 전체 3개의 버전 »

NeuralPDE.jl: Automatic Physics-Informed Neural Networks (PINNs) ~ 700 stars

NeuralPDE.jl NeuralPDE.jl is a solver package which consists of neural network solvers for partial differential equations using physics-informed neural networks (PINNs).

julia

Features

- Physics-Informed Neural Networks for ODE, SDE, RODE, and PDE solving.
- Ability to define extra loss functions to mix xDE solving with data fitting (scientific machine learning).
- Automated construction of Physics-Informed loss functions from a high-level symbolic interface.
- Sophisticated techniques like quadrature training strategies, adaptive loss functions, and neural adapters to accelerate training.
- Integrated logging suite for handling connections to TensorBoard.
- Handling of (partial) integro-differential equations and various stochastic equations.
- Specialized forms for solving `ODEProblem`s with neural networks.
- Compatibility with Flux.jl and Lux.jl. for all the GPU-powered machine learning layers available from those libraries.
- Compatibility with NeuralOperators.jl for mixing DeepONets and other neural operators (Fourier Neural Operators, Graph Neural Operators, etc.) with physics-informed loss functions.

https://github.com/SciML/NeuralPDE.jl

DeepXDE: A deep learning library for solving differential equations
L Lu, X Meng, Z Mao, GE Karniadakis - SIAM review, 2021 - SIAM
… usage of **DeepXDE** and its customizability, and we also demonstrate the capability of PINNs and the userfriendliness of **DeepXDE** for five different examples. More broadly, **DeepXDE** …
☆ 저장 99 인용 719회 인용 관련 학술자료 전체 13개의 버전

DeepXDE ~ 1500 stars

DeepXDE is a library for scientific machine learning and physics-informed learning. DeepXDE includes the following algorithms:

python™

- **physics-informed neural network (PINN)**
  - **solving different problems**
    - solving forward/inverse ordinary/partial differential equations (ODEs/PDEs) [SIAM Rev.]
    - solving forward/inverse integro-differential equations (IDEs) [SIAM Rev.]
    - fPINN: solving forward/inverse fractional PDEs (fPDEs) [SIAM J. Sci. Comput.]
    - NN-arbitrary polynomial chaos (NN-aPC): solving forward/inverse stochastic PDEs (sPDEs) [J. Comput. Phys.]
    - PINN with hard constraints (hPINN): solving inverse design/topology optimization [SIAM J. Sci. Comput.]
  - **improving PINN accuracy**
    - residual-based adaptive sampling [SIAM Rev., Comput. Methods Appl. Mech. Eng.]
    - gradient-enhanced PINN (gPINN) [Comput. Methods Appl. Mech. Eng.]
    - PINN with multi-scale Fourier features [Comput. Methods Appl. Mech. Eng.]
  - Slides, Video, Video in Chinese
- **(physics-informed) deep operator network (DeepONet)**
  - DeepONet: learning operators [Nat. Mach. Intell.]
  - DeepONet extensions, e.g., POD-DeepONet [Comput. Methods Appl. Mech. Eng.]
  - MIONet: learning multiple-input operators [SIAM J. Sci. Comput.]
  - physics-informed DeepONet [Sci. Adv.]
  - multifidelity DeepONet [Phys. Rev. Research]
  - DeepM&Mnet: solving multiphysics and multiscale problems [J. Comput. Phys., J. Comput. Phys.]
  - Reliable extrapolation [arXiv]
- **multifidelity neural network (MFNN)**
  - learning from multifidelity data [J. Comput. Phys., PNAS]

DeepXDE supports five tensor libraries as backends: TensorFlow 1.x (`tensorflow.compat.v1` in TensorFlow 2.x), TensorFlow 2.x, PyTorch, JAX, and PaddlePaddle. For how to select one, see Working with different backends.

https://github.com/lululxvi/deepxde

Future work - External libraries for PINN

PIXEL: Physics-Informed Cell Representations for Fast and Accurate PDE Solvers

Namgyu Kang[1], **Byeonghyeon Lee**[1], **Youngjoon Hong**[2], **Seok-Bae Yun**[2], **Eunbyung Park**[1, 3*]

[1]Department of Artificial Intelligence [2]Department of Mathematics
[3]Department of Electrical and Computer Engineering
Sungkyunkwan University

PIXEL: Physics-Informed Cell Representations for Fast and Accurate PDE Solvers

N Kang, B Lee, Y Hong, SB Yun, E Park - arXiv preprint arXiv:2207.12800, 2022 - arxiv.org

… **physics-informed** cell representations (coined as **PIXEL**), a … for certain input coordinates in **PIXEL**. This parameter separation … Furthermore, the suggested **PIXEL** is immuned to spectral …

☆ 저장 99 인용 관련 학술자료 전체 4개의 버전 »

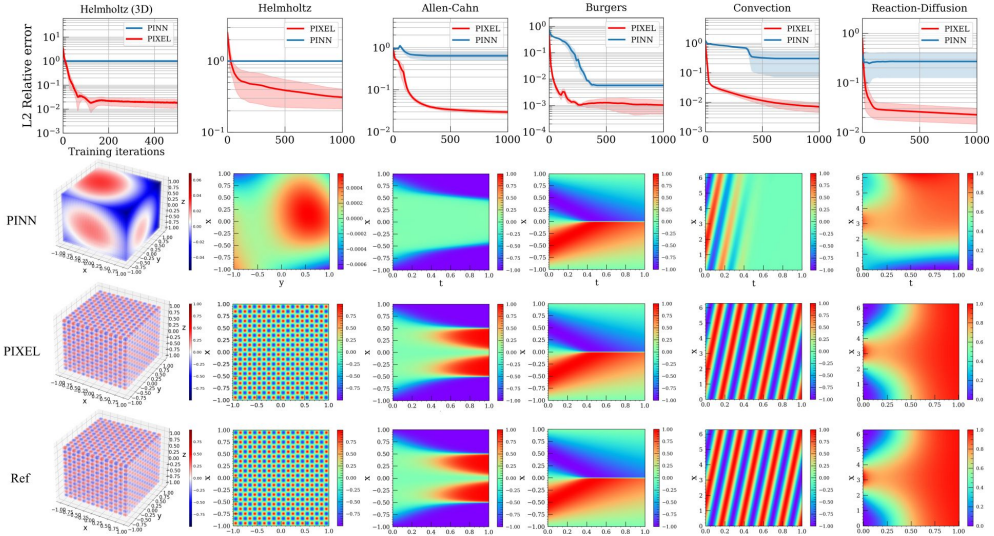


Figure 4: For the forward problem, training loss curves and solutions of various PDEs: We run both PINN and PIXEL 5 times for each PDE experiment, and the shaded areas show 80% confidence interval of 5 different runs with different *random* seeds (100, 200, 300, 400, 500). Each PDE parameters are followed. $\beta = 30$ in convection, $\nu = 3, \rho = 5$ in reaction diffusion, $a_1 = 7, a_2 = 7, a_3 = 7, k = 1$ in Helmholtz (3D), $a_1 = 10, a_2 = 10, k = 1$ in Helmholtz (2D), and $\nu = 0.01/\pi$ in Burgers equations. We showed solution images of the best performing one out of 5 different runs.

https://github.com/NamGyuKang/PIXEL

Future work - Another loss function

https://github.com/RobertJaro/NF2/blob/main/nf2/train/trainer.py

Loss function in NF2

$$L = \lambda_{\text{ff}} L_{\text{ff}} + \lambda_{\text{div}} L_{\text{div}} + \lambda_B \boxed{L_B}$$

```
196            # compute loss
197            (b_diff * self.lambda_B +
198             divergence_loss.mean() * lambda_div +
199             force_loss.mean() * lambda_ff).backward()
```

Loss for Boundary Condition (z=0)

$$L_B = \begin{cases} 0 & \text{if } |B(z=0) - B_{\text{obs}}| < B_{\text{err}} \\ (|B(z=0) - B_{\text{obs}}| - B_{\text{err}})^2 & \text{otherwise} \end{cases}$$

?

```
186            # compute boundary loss
187            boundary_b = b[:n_boundary_coords]
188            b_diff = torch.clip(torch.abs(boundary_b - b_true) - b_err, 0)
189            b_diff = torch.mean(b_diff.pow(2).sum(-1))
```

?

$$L_{\text{ff}} = \frac{|(\nabla \times \mathbf{B}) \times \mathbf{B}|^2}{|\mathbf{B}|^2 + \epsilon}$$

```
434        force_loss = torch.sum(jxb ** 2, dim=-1) / (torch.sum(b ** 2, dim=-1) + 1e-7)
435        divergence_loss = (dBx_dx + dBy_dy + dBz_dz) ** 2
```

$$L_{\text{div}} = |\nabla \cdot \mathbf{B}|^2$$

Consider a vector field $\boldsymbol{B}(\boldsymbol{x}, t)$ defined in a volume V. The method involves the minimization of the quantity

$$L = \int_V [B^{-2} |(\nabla \times \boldsymbol{B}) \times \boldsymbol{B}|^2 + |\nabla \cdot \boldsymbol{B}|^2] dV \qquad (5)$$

by an evolutionary procedure. It is clear from the form of

An **optimization approach** to **reconstructing force-free fields**
MS Wheatland, PA Sturrock… - The Astrophysical …, 2000 - iopscience.iop.org
… of an initial Beld from a **force-free** and solenoidal state, is presented. The **method** is tested by
… the application of this **method** to the solar case: the **reconstruction** of **force-free** Beldsin the …
☆ 저장 99 인용 455회 인용 관련 학술자료 전체 6개의 버전

https://sprg.ssl.berkeley.edu/~jimm/fff/optimization_fff.html
IDL code in SolarSoftWare (SSW) based on Wheatland et al. 2000

Future work - Another loss function

The Astrophysical Journal, 937:11 (19pp), 2022 September 20 https://doi.org/10.3847/1538-4357/ac8b0e

Reconstruction of Coronal Magnetic Fields Using a Poloidal–Toroidal Representation

Sibaek Yi[1], G. S. Choe[1,2], Kyung-Suk Cho[3], Sami K. Solanki[4], and Jörg Büchner[4,5]
[1] School of Space Research, Kyung Hee University, Yongin 17104, Republic of Korea; gchoe@khu.ac.kr
[2] Department of Astronomy and Space Science, Kyung Hee University, Yongin 17104, Republic of Korea
[3] Korea Astronomy and Space Science Institute, Daejeon 34055, Republic of Korea
[4] Max-Planck-Institut für Sonnensystemforschung, D-37077 Göttingen, Germany
[5] Zentrum für Astronomie und Astrophysik, Technische Universität Berlin, D-10623 Berlin, Germany
Received 2022 May 21; revised 2022 July 25; accepted 2022 August 14; published 2022 September 16

Abstract

A new method for reconstruction of coronal magnetic fields as force-free fields (FFFs) is presented. Our method employs poloidal and toroidal functions to describe divergence-free magnetic fields. This magnetic field representation naturally enables us to implement the boundary conditions at the photospheric boundary, i.e., the normal magnetic field and the normal current density there, in a straightforward manner. At the upper boundary of the corona, a source surface condition can be employed, which accommodates magnetic flux imbalance at the bottom boundary. Although our iteration algorithm is inspired by extant variational methods, it is nonvariational and requires far fewer iteration steps than most others. The computational code based on our new method is tested against the analytical FFF solutions by Titov & Démoulin. It is found to excel in reproducing a tightly wound flux rope, a bald patch, and quasi-separatrix layers with a hyperbolic flux tube.

Unified Astronomy Thesaurus concepts: Solar magnetic fields (1503); Solar corona (1483); Computational methods (1965)

Iteration (not optimization)

$$\nabla^2 B_z^{n+1} = \frac{\partial J_{\|x}{}^n}{\partial y} - \frac{\partial J_{\|y}{}^n}{\partial x}, \tag{39}$$

$$\nabla^2_{xy}\Phi^{n+1} = B_z^{n+1}, \tag{40}$$

$$\nabla^2 J_z^{n+1} = \nabla^2 J_{\|z}{}^n - \frac{\partial}{\partial z}\nabla \cdot \boldsymbol{J}_\|^n$$

$$= \nabla^2_{xy} J_{\|z}{}^n - \frac{\partial}{\partial z}\nabla_{xy} \cdot \boldsymbol{J}_\|^n, \tag{41}$$

$$\nabla^2_{xy} A_z^{n+1} = -J_z^{n+1}, \tag{42}$$

$$B_x = -\frac{\partial}{\partial x}\left(\frac{\partial \Phi}{\partial z}\right) + \frac{\partial A_z}{\partial y}, \tag{26}$$

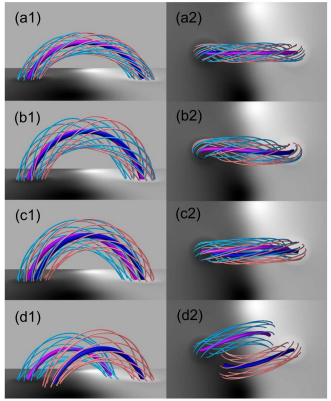
$$B_y = -\frac{\partial}{\partial y}\left(\frac{\partial \Phi}{\partial z}\right) - \frac{\partial A_z}{\partial x}, \tag{27}$$

$$B_z = \frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2}. \tag{28}$$

$$J_x = \frac{\partial}{\partial x}\left(\frac{\partial A_z}{\partial z}\right) + \frac{\partial}{\partial y}(\nabla^2 \Phi), \tag{29}$$

$$J_y = \frac{\partial}{\partial y}\left(\frac{\partial A_z}{\partial z}\right) - \frac{\partial}{\partial x}(\nabla^2 \Phi), \tag{30}$$

$$J_z = -\left(\frac{\partial^2 A_z}{\partial x^2} + \frac{\partial^2 A_z}{\partial y^2}\right). \tag{31}$$



TD (Analytical model)

NFPT (This paper, Yi et al. 2022)

VFVP (Previous code by Yi et al. 2022)

SSW (IDL code based on Wheatland et al. 2000)

Figure 1. Field lines of model TD1. Each row shows a side view and a top view for (a) the analytical model, (b) the numerical solution by our new code nonvariational FFF code in poloidal–toroidal formulation (NFPT),* (c) the numerical solution by our earlier code variational FFF code in vector potential formulation (VFVP),* and (d) the numerical solution by the optimization code in SolarSoftWare (SSW), which employs a potential field solution at the lateral and top boundaries. Field lines in all models are traced from the same footpoints. The black and white brightness in the photosphere represents the polarity of the line-of-sight magnetic field component, i.e., white for positive and black for negative. The thin field lines in magenta are traced from certain positive footpoints of the analytical flux rope and those in cyan from negative footpoints. The thick field lines are drawn from the two footpoints of the magnetic axis (blue from the positive and pink from the negative) of the analytical flux rope. For the analytical model (a) and the numerical model (b), the thick field lines overlap with each other. For the numerical model (c), they are slightly off, but the overall structure shows one flux rope. For the numerical model (d), thick field lines are quite separated and the overall structure indicates the presence of two flux tubes rather than one.

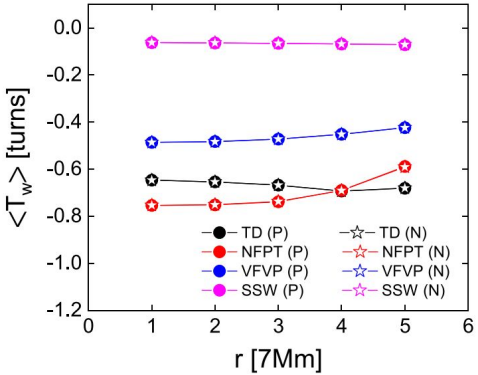


Figure 2. Mean twists of different solutions in units of turns for TD1. Twenty-four field line footpoints are chosen in each circle of radius $r/r_0 = 1, 2, 3, 4$, and 5, where $r_0 = 7$ Mm, in both positive (P) and negative (N) polarity sides, and an average is taken to give the mean twist as a function of r at $z = 0$. The mean twists of field lines traced from the positive footpoints and those from the negative footpoints are almost indistinguishable. The twist of NFPT is quite close to that of the analytical solution (TD). The SSW solution shows the least twist in magnitude.