Mechanical and Aerospace Engineering

NSTX-U feedforward shape control and neural net dynamics modeling

J. Wai[1], M.D. Boyer[2], E. Kolemen[1,2]

[1]*Princeton University, USA*
[2]*Princeton Plasma Physics Laboratory, USA*

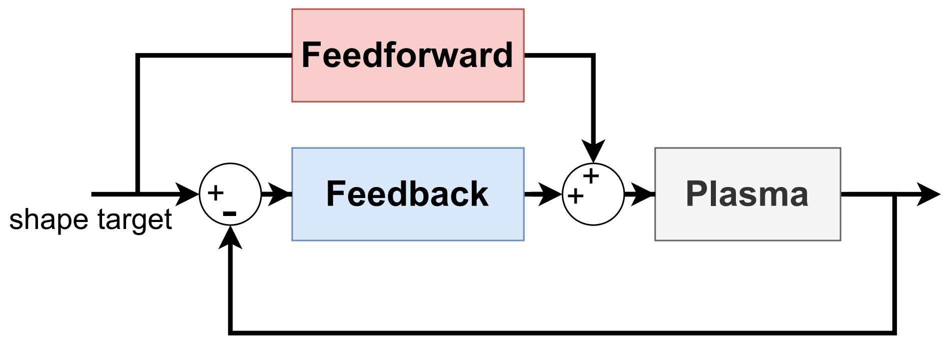
E-mail: jwai@princeton.edu

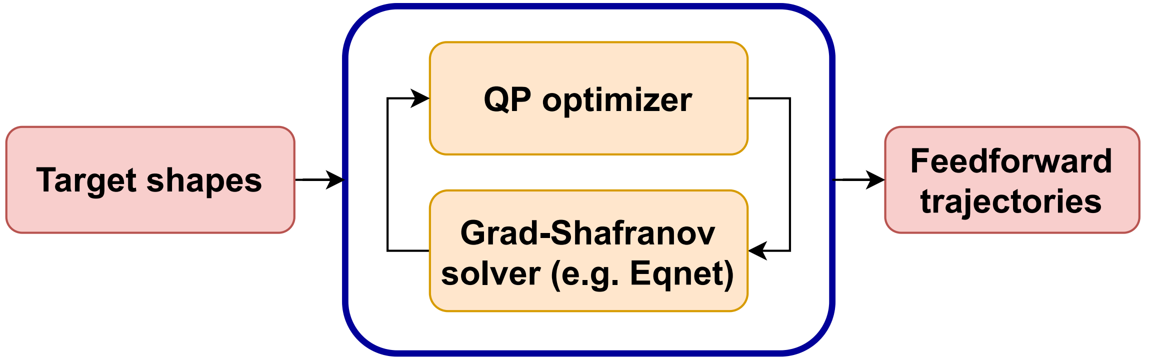
APS-DPP Nov 8-12, 2021

Executive summary (1 of 2)

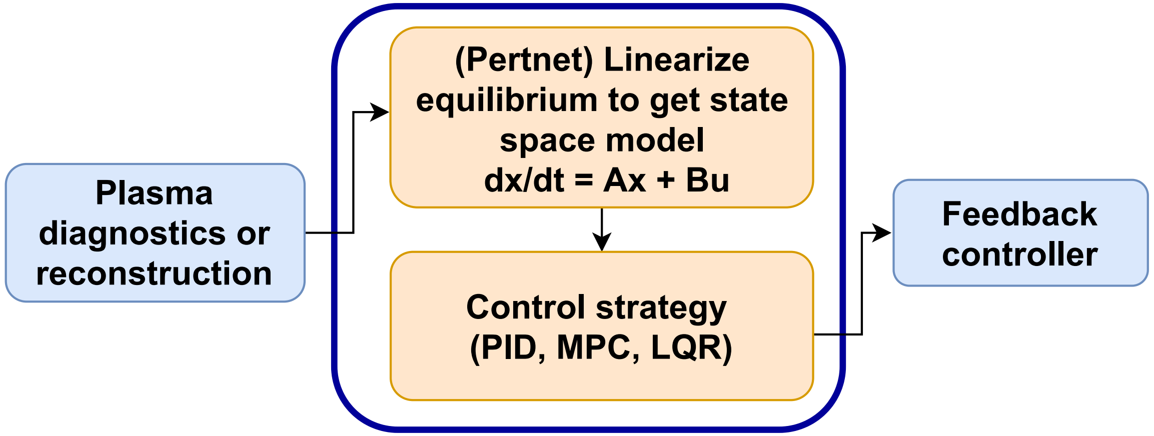
- Upgrading the NSTX-U shape controller to use **feedback and feedforward**, due to performance issues last campaign.

- Designed optimizer and neural nets that will enable fast feedforward and feedback control design.





Between shots



Between shots / during shot

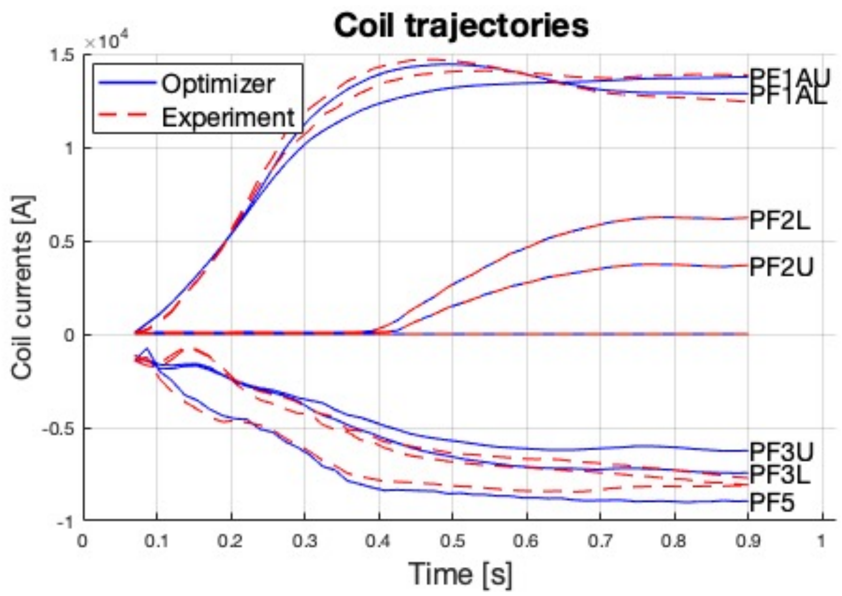
Executive summary (2 of 2)

Shape optimizer uses input shapes to estimate coil current trajectories.

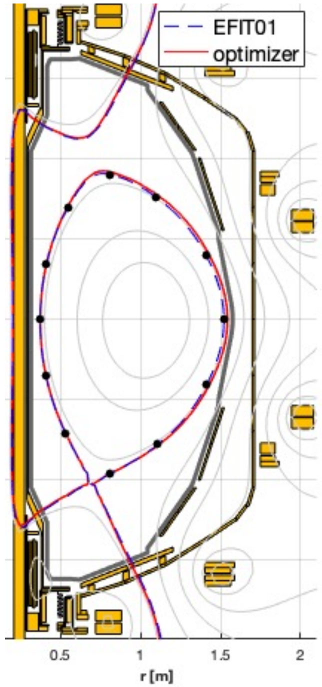
- successfully recreates old shots.
- will improve control, shot planning, and constraint avoidance.

Developed (complementary) shape control neural nets, for fast calculation and simulation.

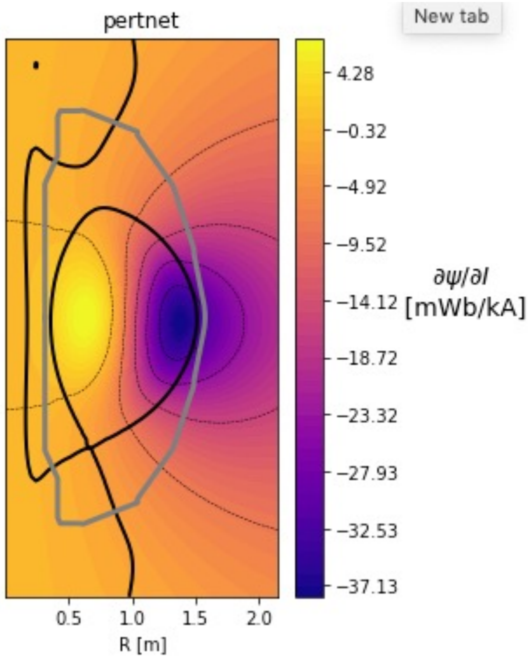
- Eqnet: predicts equilibrium/shape parameters.
- Pertnet: predicts dynamical model



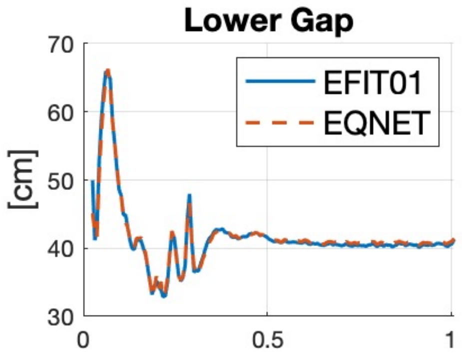
Optimizer successfully recreates coil current trajectories.



Optimizer successfully recreates equilibrium shapes.



Neural net prediction of plasma response (dynamics model)



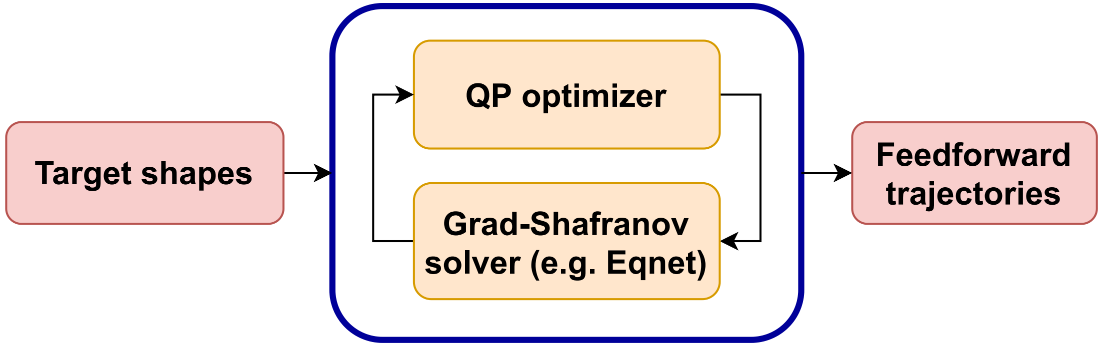
Neural net prediction of shaping parameters.

NSTX-U feedforward shape control

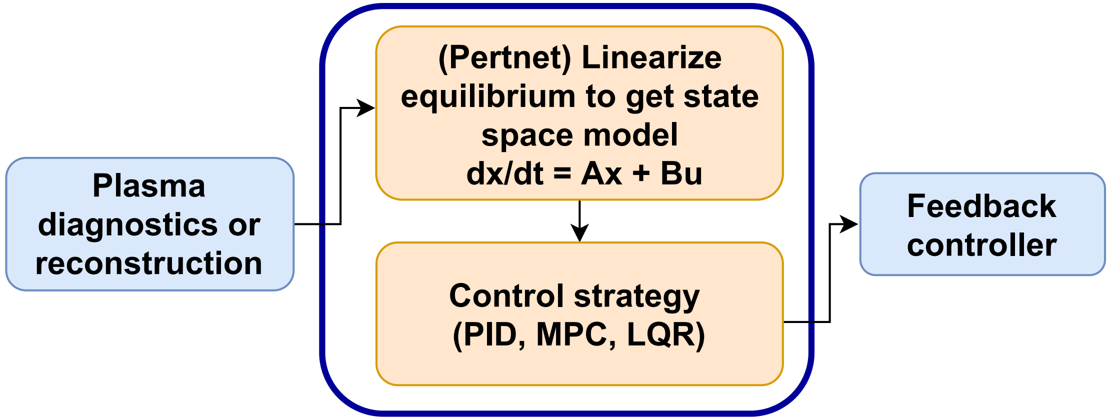
Motivation: upgrade NSTX-U shape control to include feedforward

- NSTX-U isoflux shape control algorithm relies entirely on feedback, which caused some difficulties during previous campaign.
  - Oscillations and tracking errors
  - Shot-to-shot tuning of PID gains

- Control could be improved by using feedback to adjust target coil currents around a reference current trajectory
- Optimizer can also be used to plan around constraints, minimize OH usage and extend shot, etc.
- Neural nets developed for *fast* implementations



Between shots



Between shots / during shot

Feedforward shape design algorithm

Step 1: user inputs

Targets:
-boundary (Rb,Zb)(t)
-touch or x-point (Rbdef, Zbdef)(t)
-plasma current Ip(t)

Initial currents (Ic,Iv,Ip)(t=t0)

Non-inductive current $I_{p,NI}$

Parameters for (step 2) estimating plasma resistance Rp(t) and plasma flux $\psi_{pla}(t)$.
- for example: Rp(t), Wmhd(t), li(t)

Step 2: estimate plasma properties (Rp, ψ_{pla})

Estimate Rp(t):
- direct user input of Rp or Te
- multishot average of resistivity η(t) (==current method)
- transport modeling with actuators (not implemented)

Estimate the plasma $\psi_{pla}(t)$:
- Several methods implemented that trade off speed, accuracy, and flexibility. For details see appendix.

A. custom semi-fixed boundary solver

B. gsdesign

C. eqnet

Feedforward shape design algorithm

Step 3: formulate and solve optimization for coil currents

- The shape control model uses a circuit equation for the coil, vessel, and plasma current dynamics. This can be rearranged into state-space form.
  - v = voltages, R = resistance, I = currents, M = mutual inductance, ψ = flux

$$v_i = R_i I_i + \sum_j \left(M_{ij} \dot{I}_j + \frac{\partial \psi_{i,plasma}}{\partial I_j} \dot{I}_j \right)$$

$$\begin{aligned} \dot{\mathbf{I}} &= \mathbf{A}(t)\mathbf{I} + \mathbf{B}(t)\mathbf{v} \\ \mathbf{A}(t) &:= -(\mathbf{M} + \mathbf{X}(t))^{-1}\mathbf{R} \\ \mathbf{B}(t) &:= (\mathbf{M} + \mathbf{X}(t))^{-1} \end{aligned}$$

$$X_{ij} := \frac{\partial \psi_{i,plasma}}{\partial I_j}$$

- Define our outputs y as the currents and isoflux shape errors.

$$x := [I_v \;\; I_p], \; u := I_c$$

$$y = y_e + \delta y$$

$$y = \begin{bmatrix} I_c \\ I_v \\ I_p \\ \psi|_{R_b, Z_b} - \psi|_{R_{bdef}, Z_{bdef}} \\ \nabla\psi|_{R_{bdef}, Z_{bdef}} \end{bmatrix}$$

- Optimize a constrained quadratic cost function of the outputs **y** versus targets **r**. Outputs at future times are predicted via the dynamics model.

$$x_{k+1} = Ax_k + Bu_k$$

$$\delta y_k = C\delta x_k + D\delta u_k$$

$$\text{minimize } J = \sum_{k=1}^{N} (y_k - r_k)^T Q (y_k - r_k) + \Delta y_k^T Q_v \Delta y_k$$

Feedforward shape design algorithm

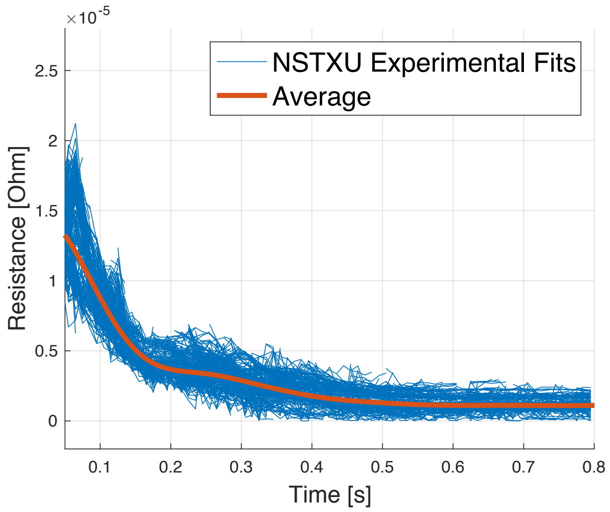
Step 4: Iterate

- In plasma estimation step (step2), solutions implicitly depend on the total flux/applied flux/coil currents.
- If we have reasonable knowledge of plasma parameters(t), then don't need to iterate.
- However, if estimates are not known then iteration likely needed. For example, if we explicitly evolve profile dynamics (not yet implemented) then need to account for coil current trajectories as profile actuators.
- Example: can use Li as a fitting parameter in estimating ψ_{pla}, but also depends on OH/Ip ramp rate.

Example: recreate NSTX-U 204660

User inputs

- Target boundary and Ip obtained from experimental EFIT01 equilibria
- Ip, W_{MHD}, and boundary given as inputs to semi-fixed boundary solver for step 2

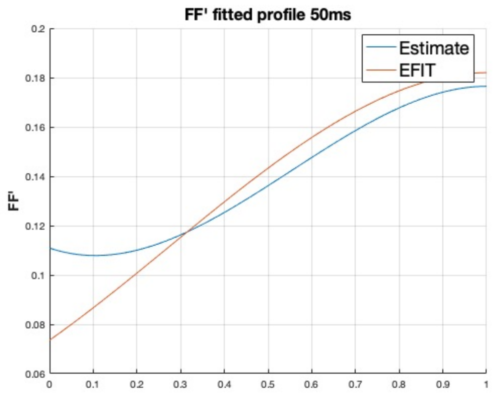


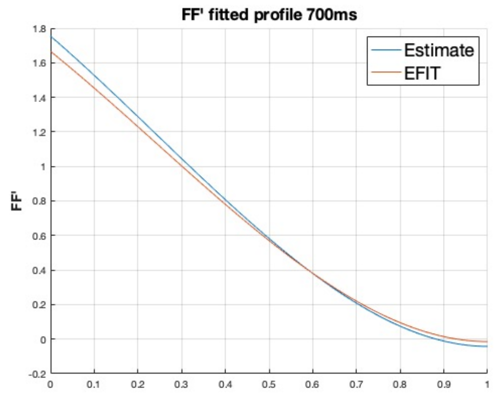
$$R_p = 1/I_p\left(V_{loop} - L_p\dot{I}_p\right)$$

Fig 1: plasma resistances fits from 2015-16 campaign

Estimate Rp and ψ_{pla}

- Rp: plasma resistance is estimated as average resistance from campaign.
- ψ_{pla}: semi-fixed boundary solver is a simple & fast implementation that scales averaged EFIT01 FF’ and P’ profiles to match Ip and W_{MHD}.
- (This can be a gross approximation, but optimizer is fairly robust to internal profile details.)





Figs 2 and 3: Examples of FF’ fitting approximations found by solver. Optimizer is robust against this level of approximation.

Example: recreate NSTX-U 204660

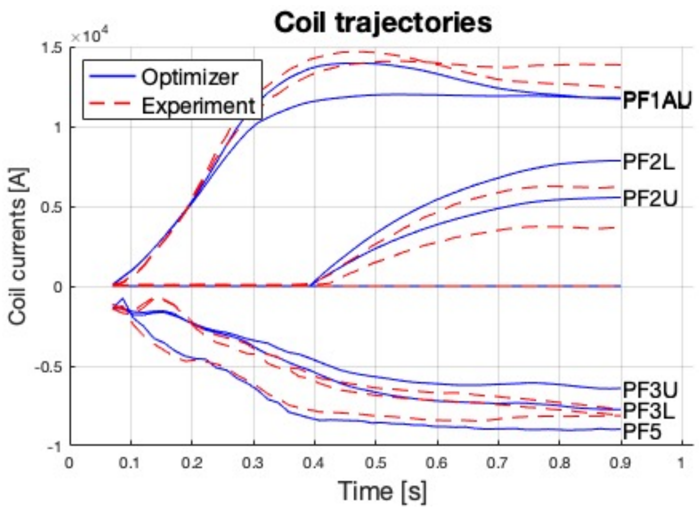


Fig 1A: coil currents, PF2 unconstrained

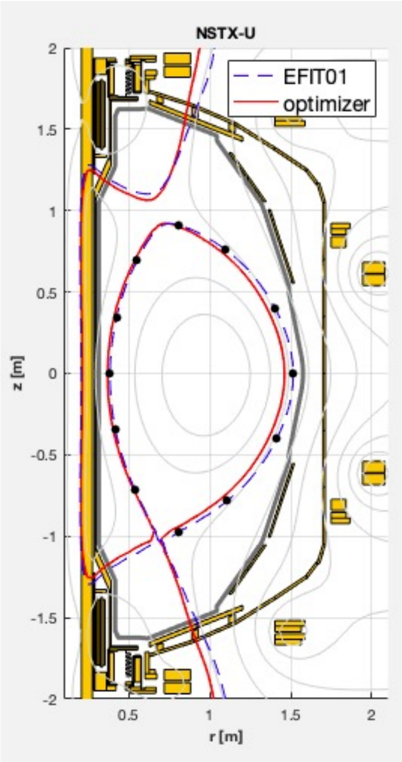


Fig 2A: Equilibrium at final time, PF2 unconstrained

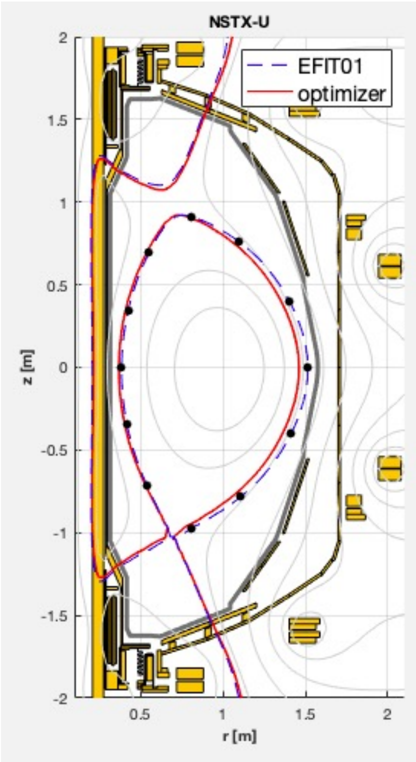


Fig 2B: Equilibrium at final time, PF2 constrained

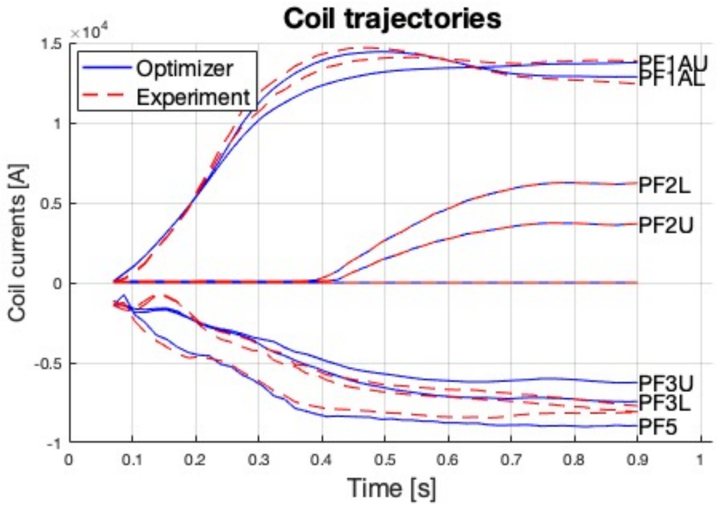


Fig 1B: coil currents, PF2 constrained

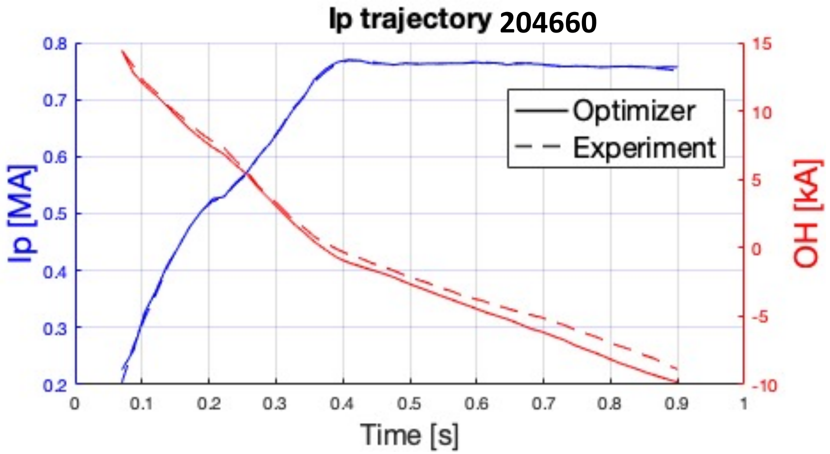
- Optimizer initially finds a different set of coil currents (**Fig 1A**) than was obtained in experiment. This can be interpreted as a “lower cost” solution to the shape optimization.
- If PF2U/L are constrained to experimental values, other coils converge towards experiment (**Fig 1B**)

Both optimization examples have similar shape errors, which are attributed to the ψ_{pla} estimate. (If exact FF’/P’ profiles are given, shape error → 0.0cm)

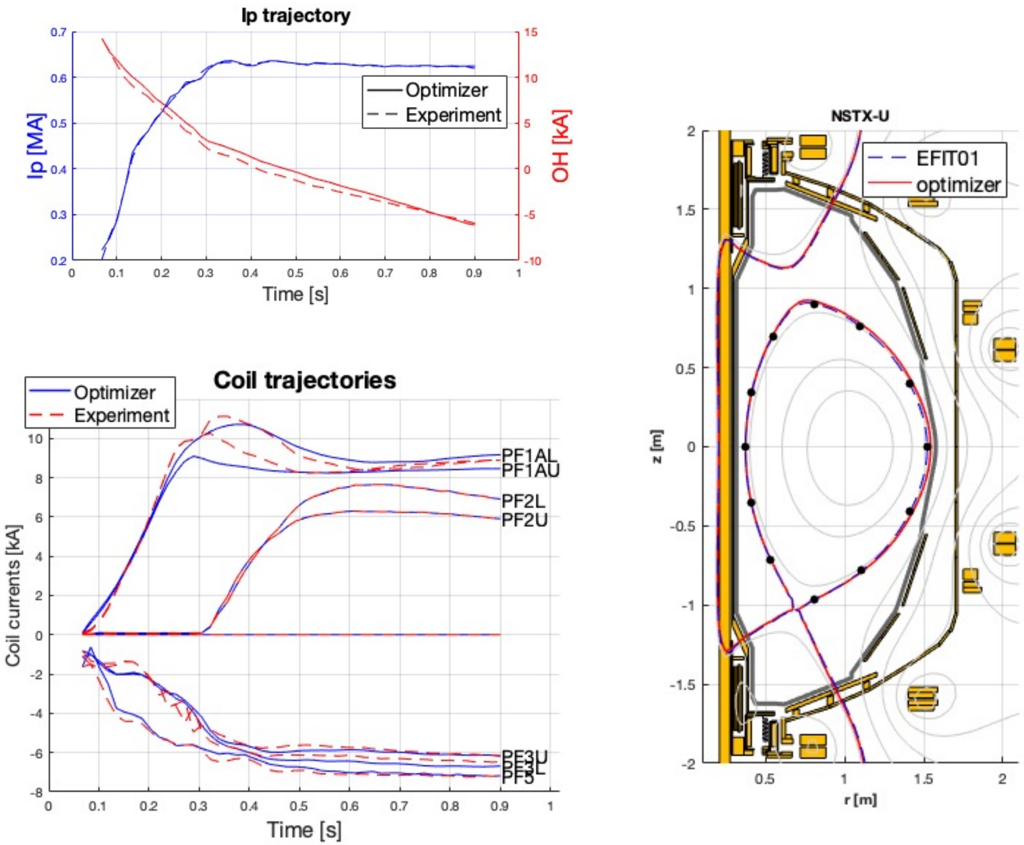
Example: recreate NSTX-U 204660 / 204069

- Ip/OH match with experiment well, but some deviation due to Rp approximation (good enough for feedforward).

- Relationship is sensitive to Rp, and in practice would be improved by more detailed modeling or reference shots. However, shaping is not overly sensitive to OH.



Another example: Ip, coil trajectories, and final shape for 204069:



Feedforward shape designs: next steps

Next steps:

- quantify sensitivity of feedforward solutions to assumptions
- Test the combined feedforward + feedback system with gsevolve [Welander, 2019] simulations. How accurate does the feedforward need to be?

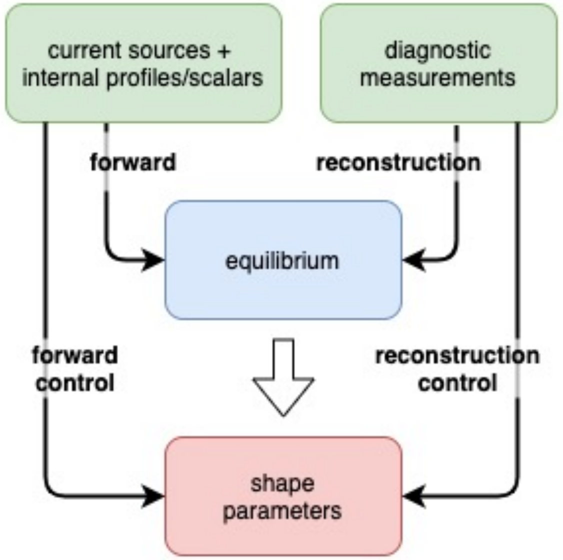
General thoughts:

- System takes input profiles and optimizes coil trajectories and flux maps, i.e. it decouples the profile optimization from shape optimization.
- On first principles this may not seem like a good idea, but results look promising for NSTX-U. There is a practical advantage since it is difficult to solve both optimizations simultaneously
  - Profile optimizers often use prescribed shapes ([Felici, 2012], [Teplukhina, 2017])
  - Most self-consistent coupled simulators not designed for optimization

Neural net equilibrium and dynamics modelling

Neural nets for shape control

- **Eqnet:**
  - neural net equilibrium solver
  - several modes of operation. Predict flux surfaces or shaping parameters from diagnostics or profiles.
  - Useful for, e.g., estimating $\boldsymbol{\psi}_{\mathbf{pla}}$ for the shape optimizer.



Eqnet inputs and outputs

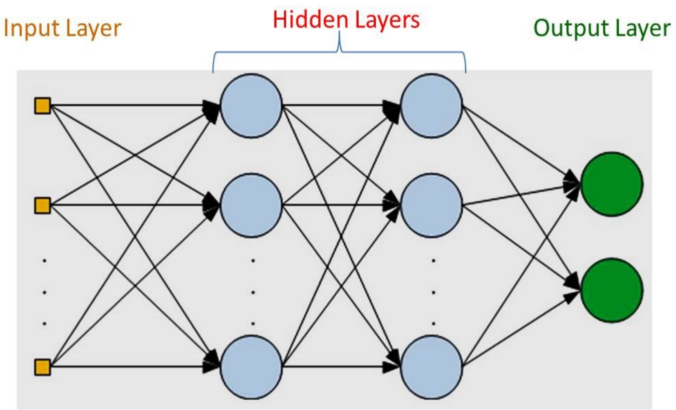
- **Pertnet:**
  - predicts the non-rigid plasma response $\frac{\partial \psi_{plasma}}{\partial I}$ which is a nonlinear term in the shape control model (how the plasma redistributes in response to coil current perturbation)
  - trained on gspert code outputs [Welander, 2005]

$$v_i = R_i I_i + \sum_j \left(M_{ij} \dot{I}_j + \frac{\partial \psi_{i,plasma}}{\partial I_j} \dot{I}_j \right)$$

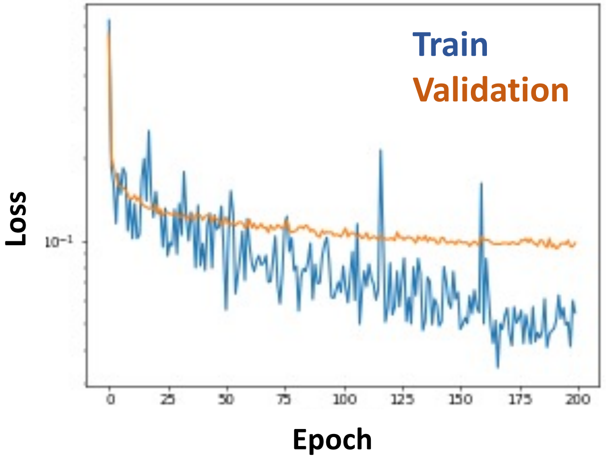
$$\dot{x} = A(t)x(t) + B(t)u(t)$$
$$y(t) = C(t)x(t) + D(t)u(t)$$

Neural net architectures

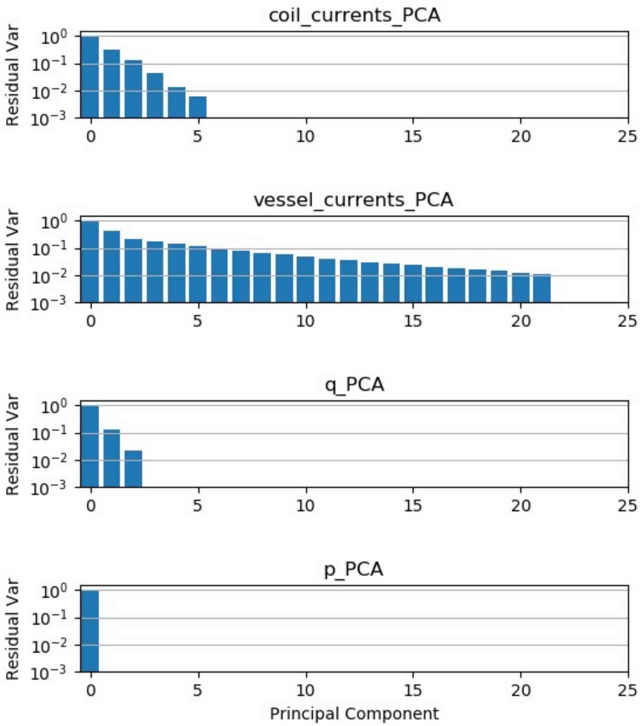
- Both NNs use multilayer perceptron (MLP) architectures with PCA reduction of inputs and outputs to capture 99.9% variance.
- Train, validation, test split is 80-10-10 by shot number
- Trained in PyTorch using a gridscan for hyperparameters
  - Eqnet: 6 hidden layers, size 800, ELU activation, no dropout
  - Pertnet: 3 hidden layers, size 200, ReLU activation, 10% dropout



Multilayer perceptron



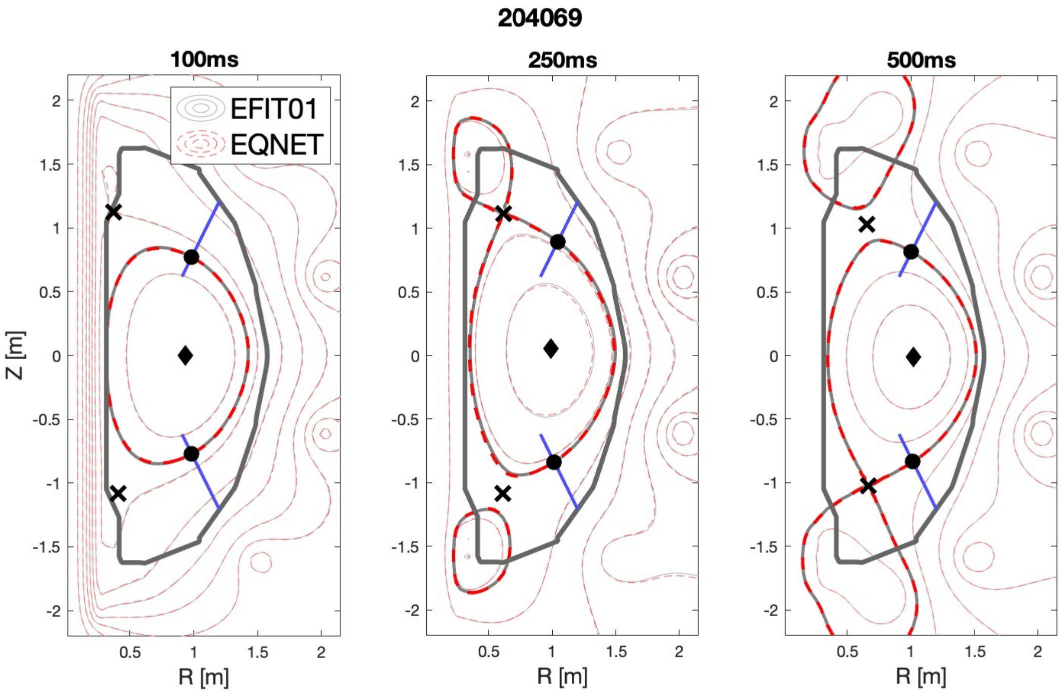
Training loss curve



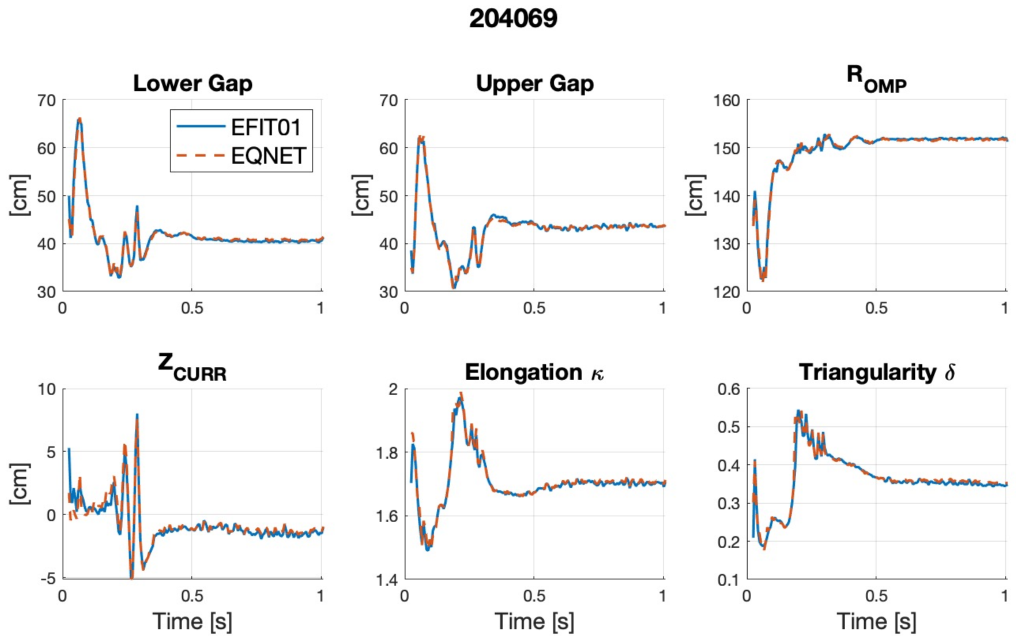
PCA components for a subset of the input variables

Eqnet results

- Eqnet results (from validation dataset)



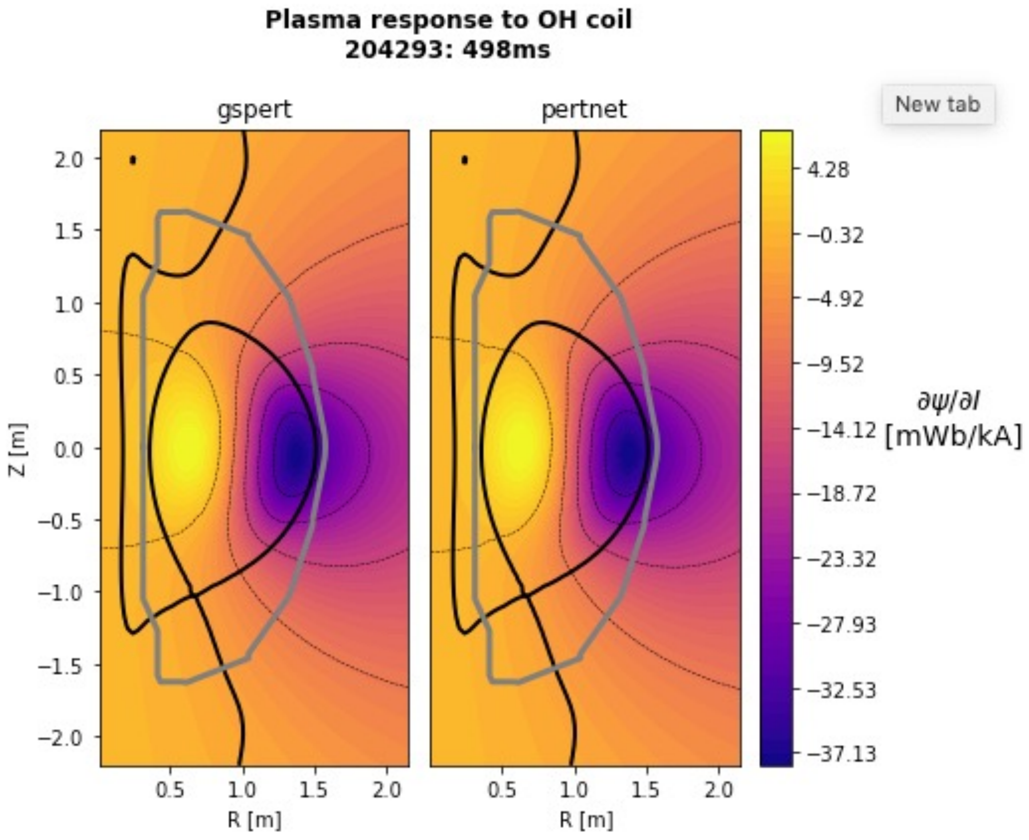
Equilibrium flux predictions capture range of equilibria (limited, USN, LSN)



Good prediction of shaping parameters

Pertnet results

- Pertnet results (from validation dataset)



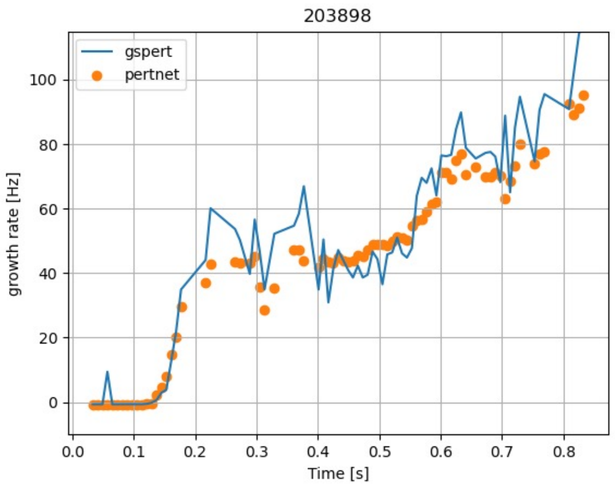
- Growth rate ($\boldsymbol{\gamma}$) can be obtained from flux responses

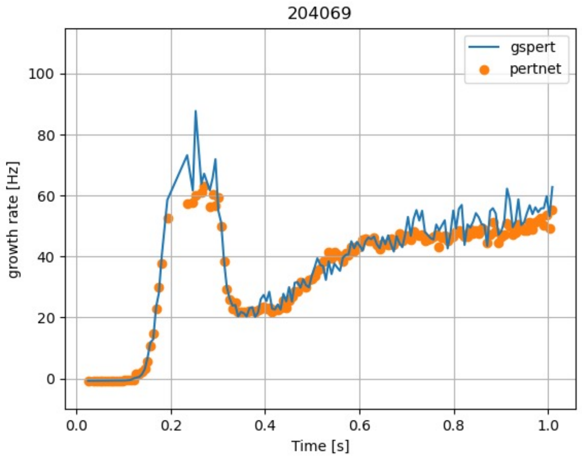
- Flux responses give good estimates of growth rate, even though NN not trained to estimate growth rate directly.

$$X_{cc} = M_{cp} M_{pp}^{-1} \frac{\partial \psi_p}{\partial I_c}$$

$$A = -(M_{cc} + X_{cc})^{-1} R_c$$

$$\gamma = \max \text{ eigenvalue}(\mathrm{A})$$





Conclusion

Conclusions

- Design tool created for optimizing feedforward currents on NSTX-U in order to match desired shape targets.
- Shows good agreement with previous experimental data and can be used to design optimize/design combined feedforward + feedback control algorithms.
- Equilibrium and perturbation neural nets also developed to speed up compute-intense portions of code and implementation is in-progress.

References

A. Welander *et al* 2005 *Fusion Science and Technology*, **47** 763-767

A. Welander *et al* 2019 *Fusion Engineering & Design* **146** 2361-2365

F. Felici and O. Sauter 2012 *Plasma Phys. Control. Fusion* **54** 025002

A.A. Teplukhina *et al* 2017 Plasma Phys. Control. Fusion 59 124004

Acknowledgements:

- This work supported by DOE contract DE-SC0015878 and DE-AC02-09CH11466.

Appendix

Feedforward shape design algorithm: step 2 details

Step 2: estimate plasma flux distribution ψ_{pla}

A. custom semi free-boundary solver

Ip, Wmhd, (Rb,Zb), EFIT01 average profiles P_0', FF_0' → ψ_{pla}

Scales P_0', FF_0' in order to match Ip, Wmhd
~2 iterations is OK

+ fast, reasonably accurate

- need Wmhd a-priori, profile assumption weak

B. gsdesign

Flexible inputs (e.g., Ip, betap, li, boundary) → ψ_{pla}

+ well-established tool, flexibility in inputs, generalizability, accuracy

- speed, profiles (so far, have not been able to figure out some desired behavior for profiles like shift/scale a given FF'. Can shift/scale a linear FF', or match a target profile).

C. Eqnet neural network

Flexible inputs (e.g., Ip, betap, li, boundary) → ψ_{pla}

Have not yet trained for this particular mode of operation, but accurate for similar problems

+ speed, accuracy(?)

- accuracy(?), generalizability, flexibility