

Application of Machine Learning for Development of Fast Models for Integrated Modelling

**Sebastian Bodenstein, Anushan Fernando,
Philippe Hamel, Jonathan Citrin**
Google DeepMind, London, UK



Photo by Khyati Trehan for Google DeepMind on Unsplash

Contents



1. The Why and How of "fast" integrated modelling
2. Deep-dive: A neural network surrogate for tokamak turbulence
3. Modern simulation frameworks utilizing ML-surrogates for optimization and control

The Why and How of "fast" integrated modelling

- Use-cases for fast simulation
- (very) brief primer on machine learning and neural nets
- Fusion examples of ML-accelerated applications
 - Superficial overviews with links for further reading



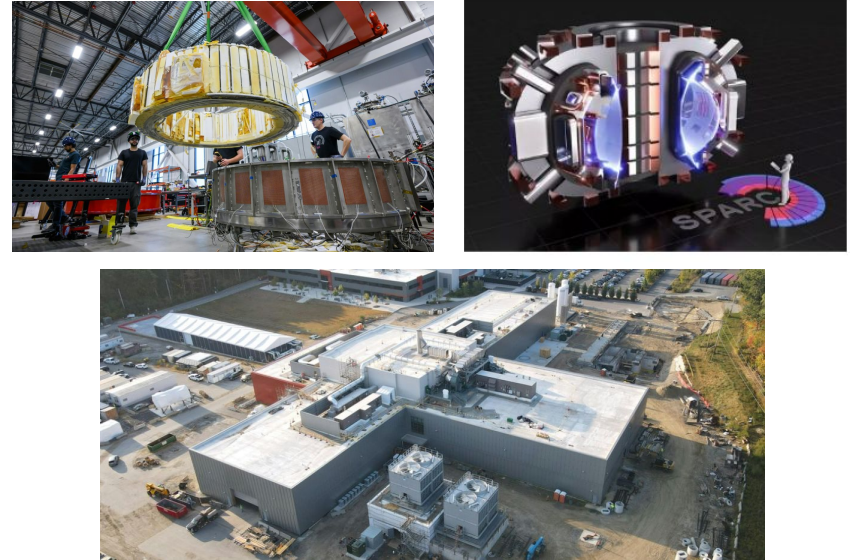
Next generation of tokamak experiments aiming for net fusion gain

ITER: Cadarache, France, standard 5T field



Acknowledgement to ITER Organization

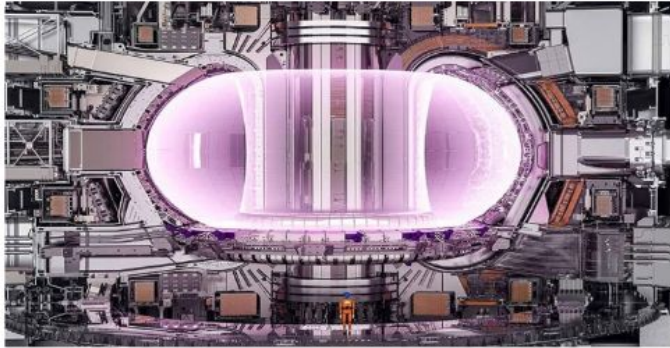
SPARC, Commonwealth Fusion Systems (CFS),
high-field-superconductors. $\sim 12\text{T}$



Acknowledgement to Commonwealth Fusion Systems

Key issue: present-day physics models too slow for routine simulations used for experiment prediction and interpretation

Leap from present-day experiments to reactors requires leap in simulation capabilities



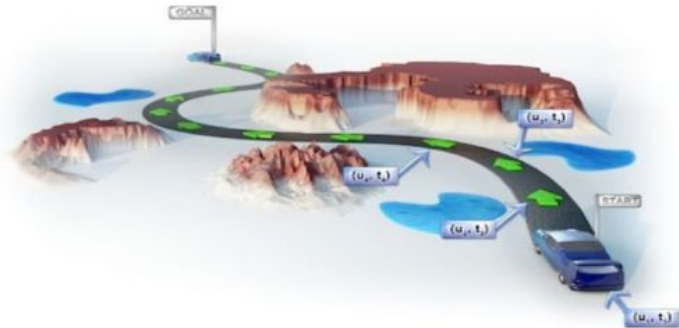
Reduce costs and risks with simulations for

- Experimental preparation including inter-shot
- Performance optimization while avoiding constraints
- Model-based controller design

In next generation devices, fast and accurate simulation a prerequisite for pulse design.

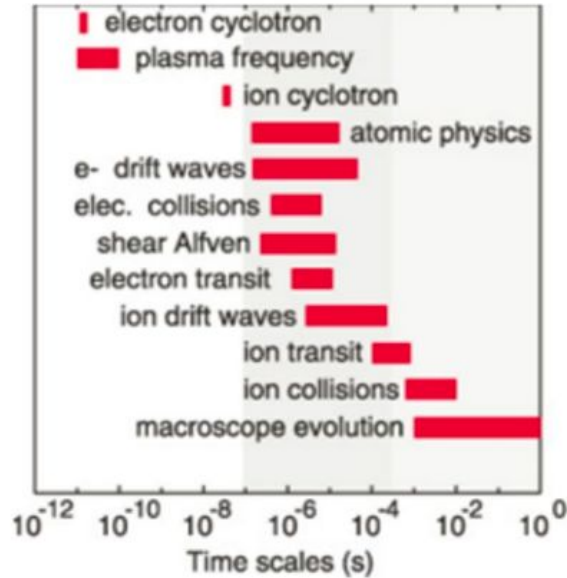
Also key:

- Uncertainty Quantification of simulation outputs
- Calibrating physics models with data when available



Integrated modelling inherently multiscale and multiphysics

Multiple orders of magnitude in spatiotemporal scales between relevant physics processes



Magnetic equilibrium

Plasma collisions

Magnetohydrodynamics

Heating and fuelling

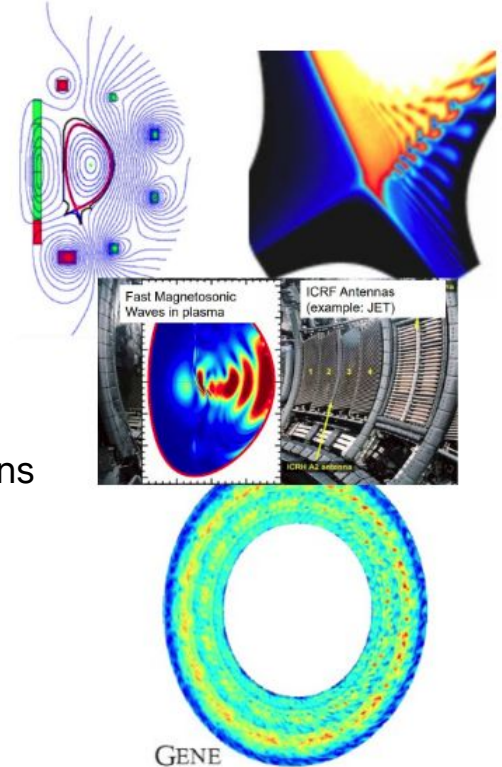
Plasma turbulence

Atomic and molecular interactions

Plasma material interaction

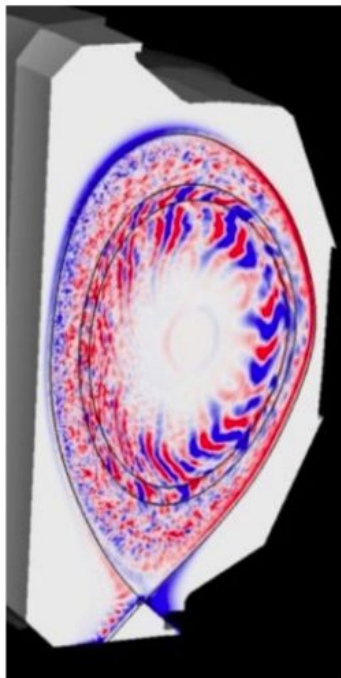
...

...



Modelling fidelity and tractability hierarchy.

Pragmatic modelling demands model reduction

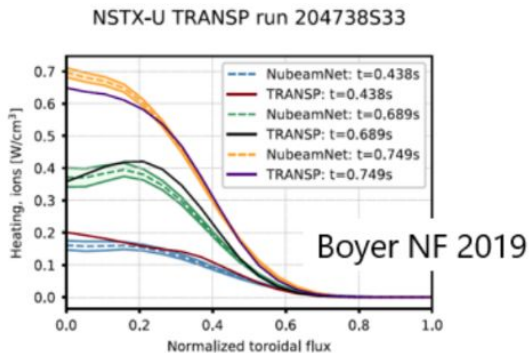


Dominiski Phys. Plasmas 2018

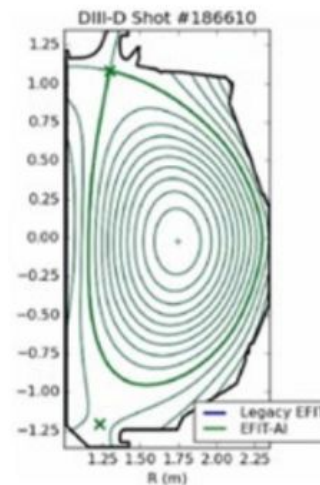
Simulation Class	Method	Fidelity level	Compute burden	Advantages	Challenges
Direct numerical simulation	Coupling high fidelity models	High	Massively parallel and expensive (exascale)	Ultimate ground truth (numerically)	Not pragmatic for most use cases
"Standard" integrated modelling	Plasma transport PDEs + coupled reduced models	Variable (depends on quality of reduced models)	Hours to weeks on single compute node	Suitable for experimental interpretation and extrapolation Community workhorse. Lots of experience and models available	Legacy burden. Tractability and accuracy are conflicting constraints
"Fast" integrated modelling	Plasma transport PDEs + coupled surrogate models	Variable (depends on quality of surrogates)	Faster than realtime to minutes	Suitable for optimization and controller design applications. Surrogates can learn higher fidelity models than "standard"	Need to develop collection of learned surrogates + appropriate framework

Key method for integrated modelling speed is incorporation of learned ML-surrogates (e.g. neural networks) of physics components. More on this later!

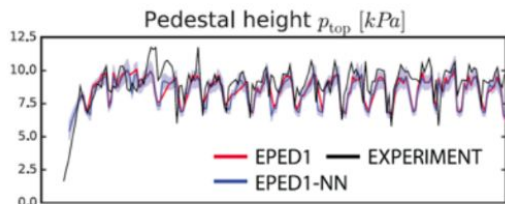
Neutral Beam Injection
and Current Drive



Magnetic
equilibrium

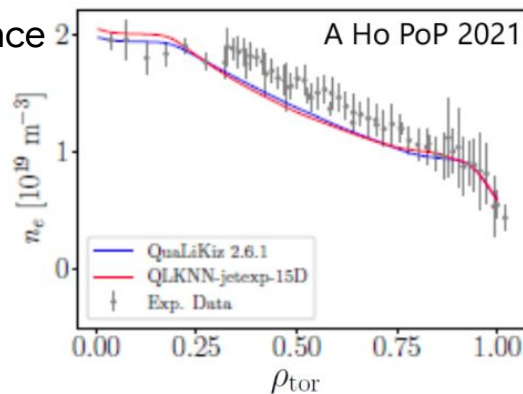


Edge Transport Barriers
(H-mode pedestal)



Meneghini NF 2017

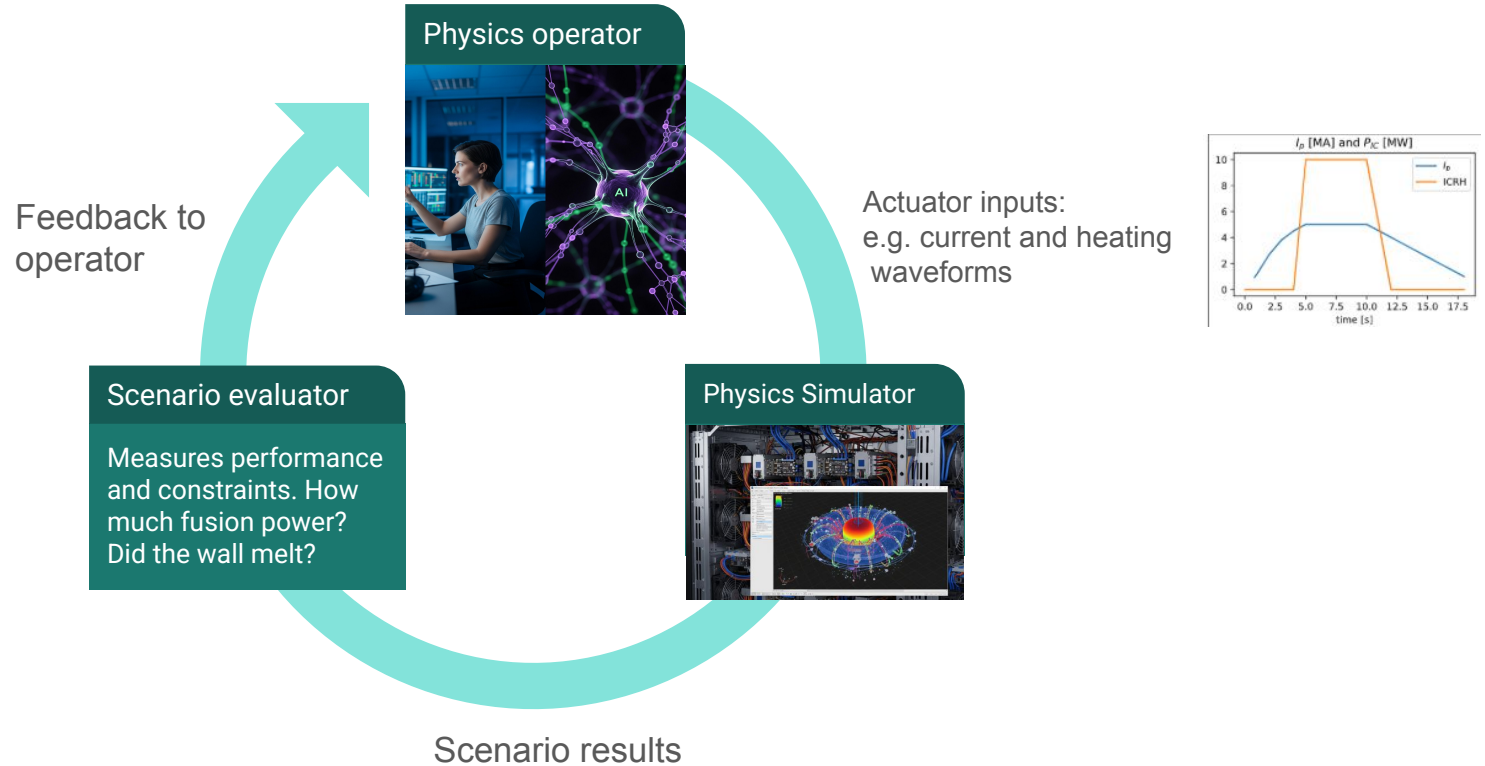
Plasma turbulence



Lao PPCF 2022

Many-query computational workflows where fast-and-accurate simulation is vital:

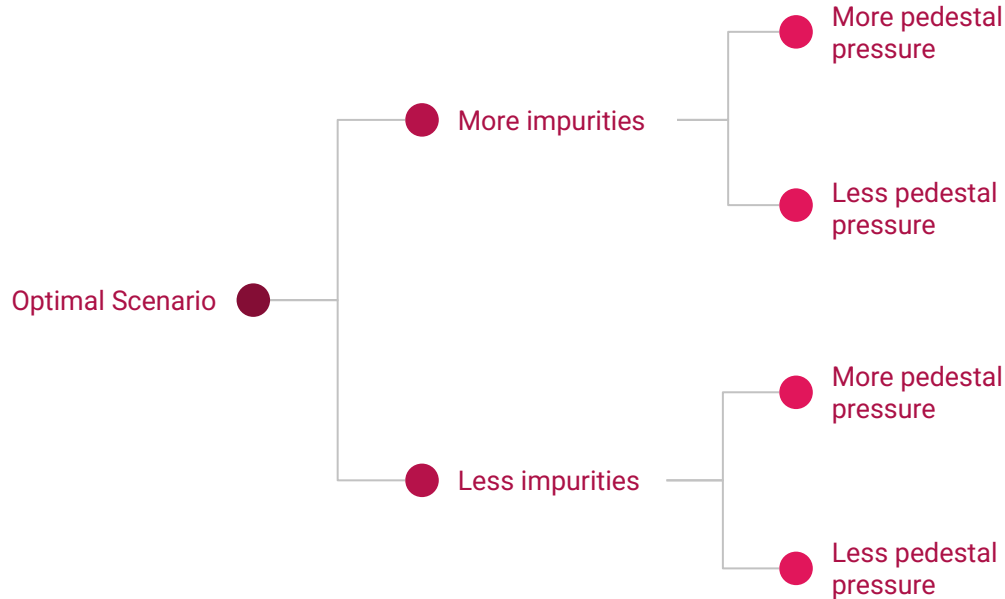
1. Scenario design and optimization



Many-query computational workflows where fast-and-accurate simulation is vital:

2. Uncertainty quantification

For single prediction for optimal trajectory is insufficient. Need "error bars" of simulation

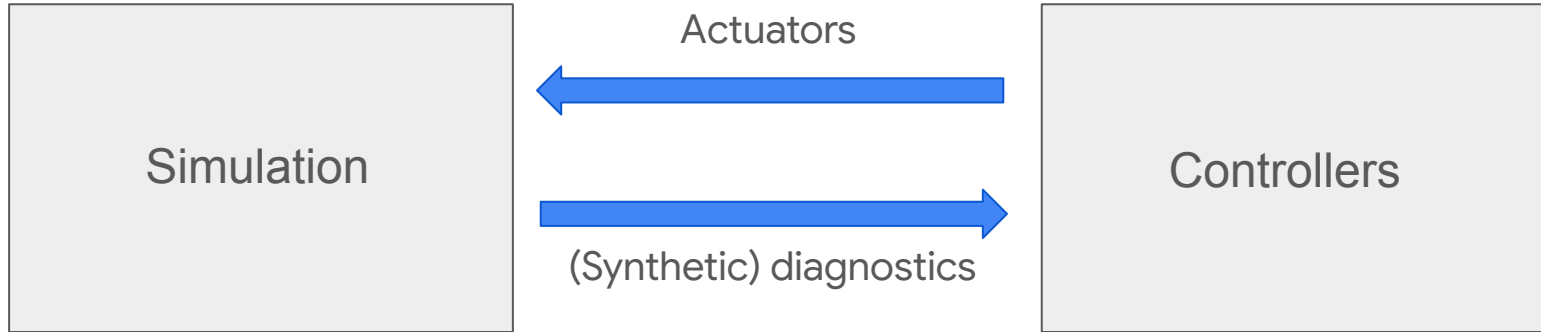


- Output distributions over physics uncertainties
- Inclusion (where possible) of impact of "off-normal-events"
- Quantify tradeoffs between scenario performance and risks
- Various techniques
 - Spawning multiple runs
 - Differentiable simulation
 - Surrogate models output uncertainties
 - ...

Many-query computational workflows where fast-and-accurate simulation is vital

3. Controller design and testing

"Flight simulation". The controllers should not be able to tell if input comes from simulation or real thing



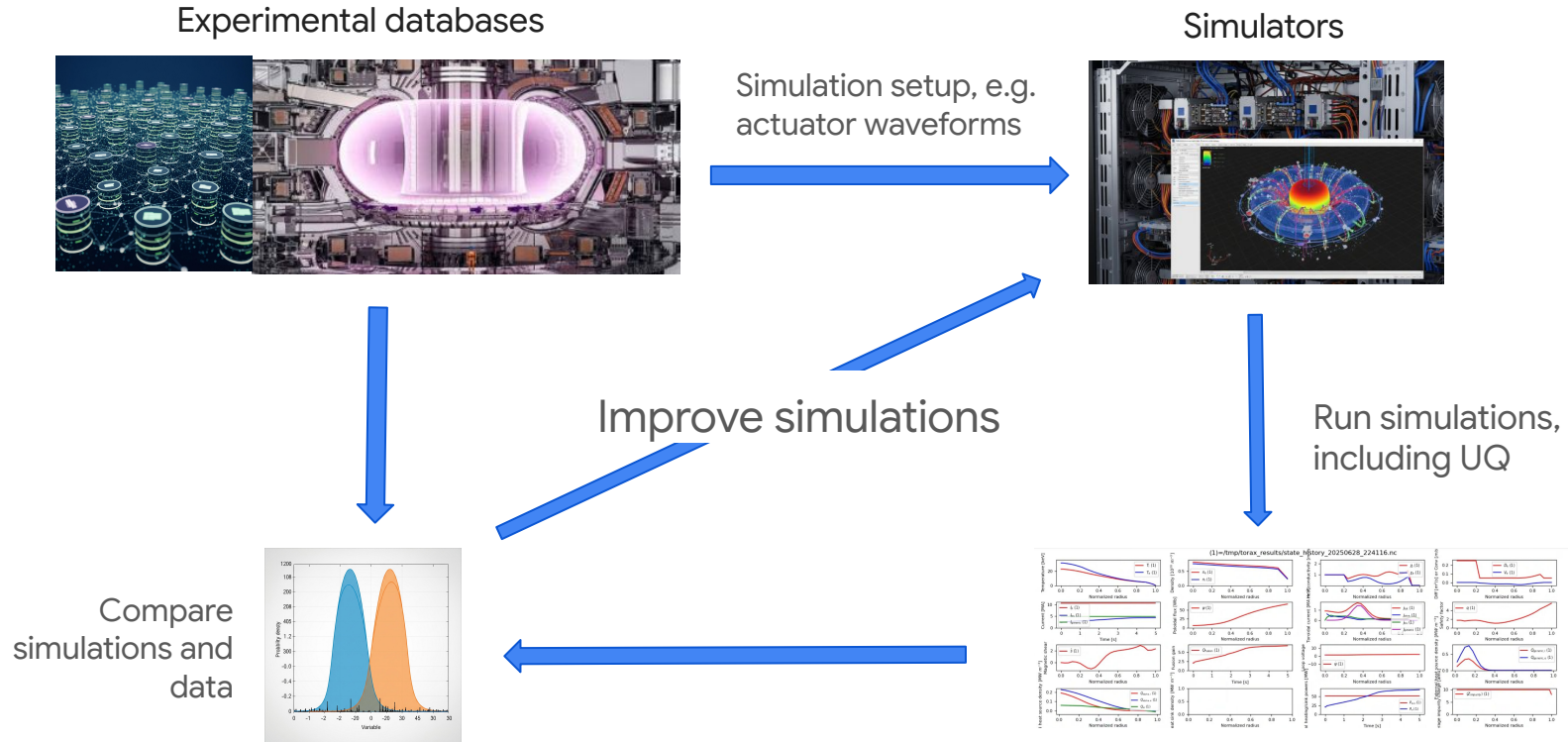
If simulation is fast, can run many times to design and test advanced controllers

If diagnostics are fast, more sensor information is available, unlocking better control schemes. ML can help with realtime diagnostics.

Many-query computational workflows where fast-and-accurate simulation is vital:

4. Large-scale validation and calibration: Learning from data

Improve simulations through combination of better physics models and learned parameters from data.
Hybrid physics+data learned models may generalize better to new regimes than pure data-learned models



(brief!) primer on Machine Learning (ML): a subset of algorithms that improve through experience without explicit instructions

Problem class

Supervised learning (labelled data)

- Regression
- Classification

Unsupervised learning

- Clustering
- Dimension reduction

Reinforcement learning

- Map state to action based on rewards

Example research questions

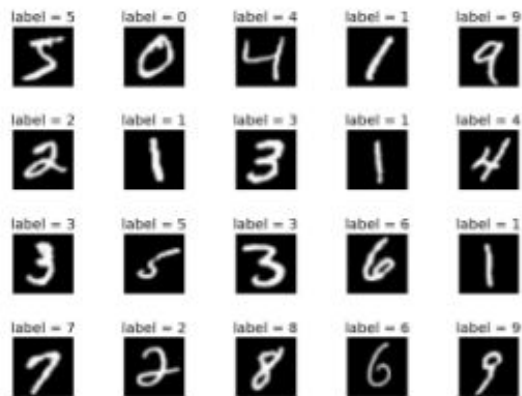
I have done lots of turbulence calculations before.
Can I learn from them and replicate them faster?
(e.g. van de Plassche PoP 2020)

What different classes of disruption does my data describe? (e.g. Murari NF 2021)

I have a target plasma shape. Based on present state, which actuators to tweak to get there and stay there? (e.g. Degraeve Nature 2022)

Examples of supervised learning use-cases for pattern recognition and regression

	Inputs	Nonlinear mapping	Outputs
Handwriting recognition:	x = image		$y(x)$ = character
Molecular dynamics:	x = atomic configuration		$y(x)$ = interatomic-potentials
Drug discovery:	x = molecular composition		$y(x)$ = effectiveness
Materials science:	x = material composition		$y(x)$ = property
Real-time system control:	x = recent measurements		$y(x)$ = prediction of future state
Tokamak turbulence:	x = local plasma parameters		$y(x)$ = flows of heat and particles



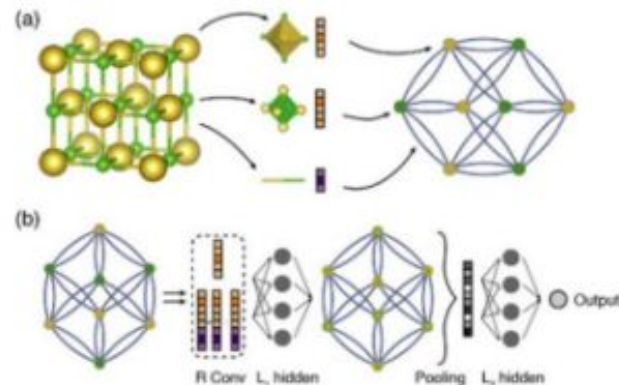
label = $y(\text{pixels})$

MNIST database of handwritten digits



caption = $y(\text{image})$

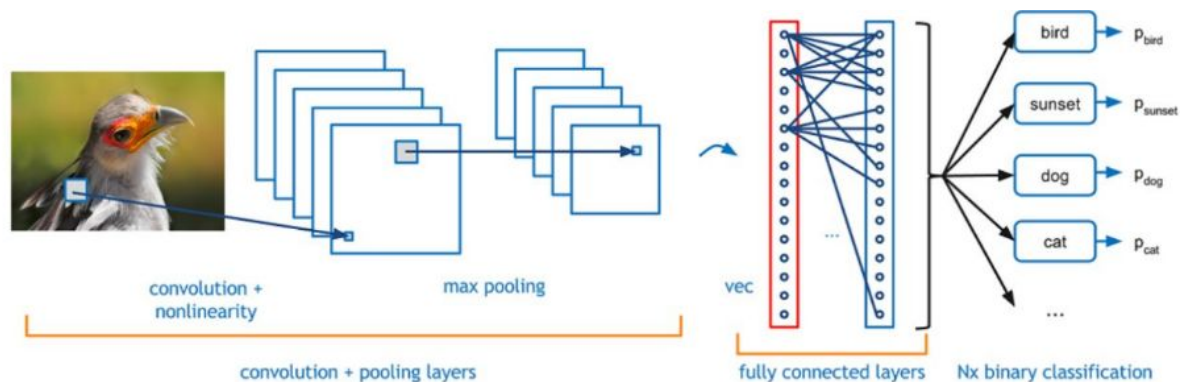
Image: Google Research



energy = $y(\text{atomic-config})$

Image: Mueller J. Chem Phys 2020

Neural networks (in various forms) are specific implementations of nonlinear mappings for ML applications. Essentially, a very powerful and general fit

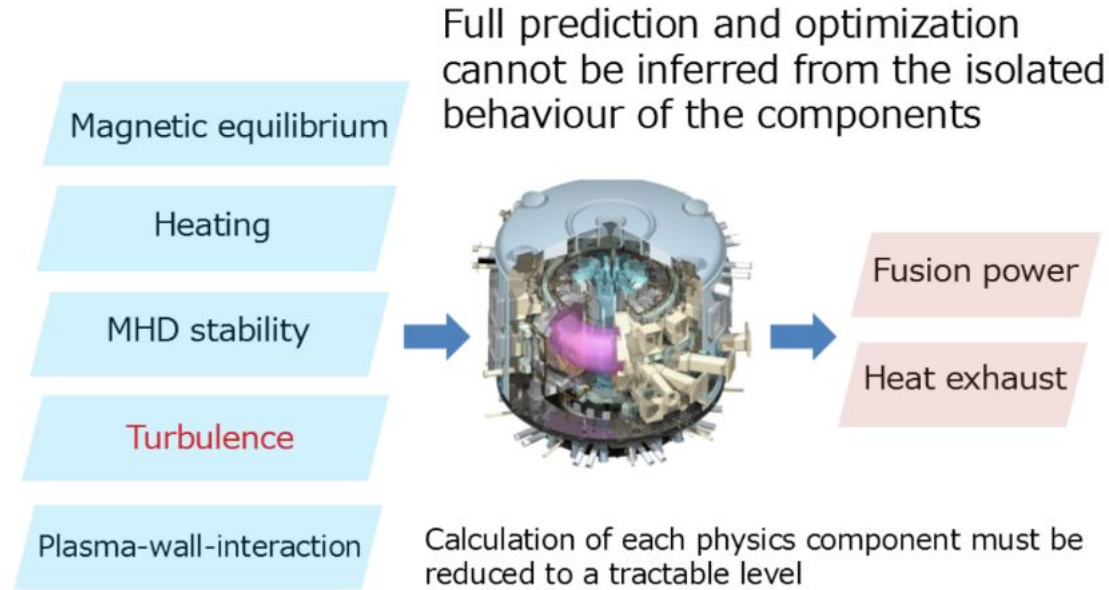


“Convolutional”
neural networks
(CNNs) typically
used for image
pattern recognition

<https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/>

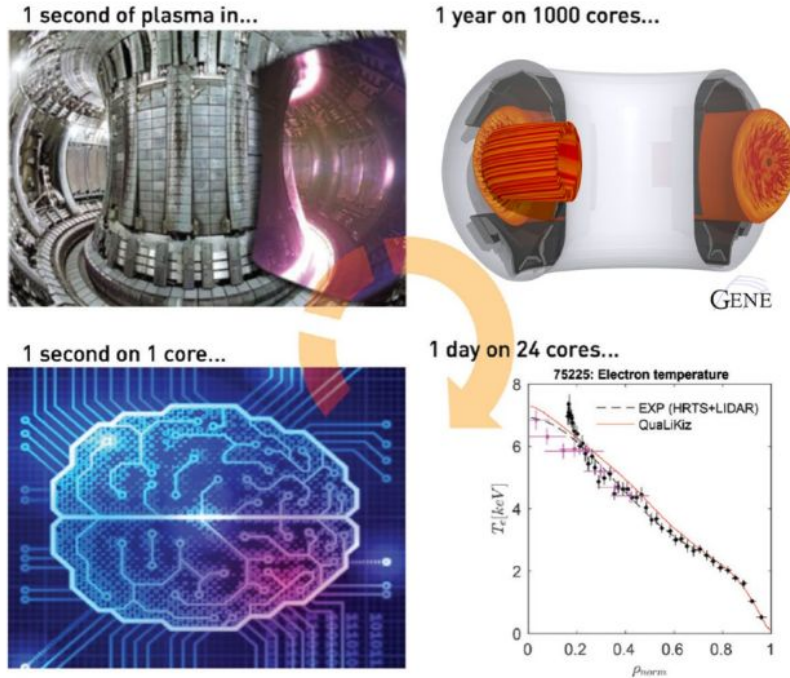
- Each step of the way are linear + nonlinear mathematical operations with (many!) free weights
- Based on a large pool of examples (training data), weights are modified to maximize match to training data. Global optimisation problem. Tour-de-force of calculus
- Immense progress in past 15 years due to:
 - Ability to handle big data
 - Advances in compute power (e.g. multinode GPU/TPU accelerators)
 - Improved optimisation algorithms, innovative network topologies
 - Appropriate software frameworks abstracting away complexity (e.g. JAX, PyTorch)

Integrated tokamak simulation is a multiphysics system with complex emergent behaviour. "Fast and accurate" constraint applies for each physics component



Machine learned surrogate models can circumvent the constraints of tractability and accuracy

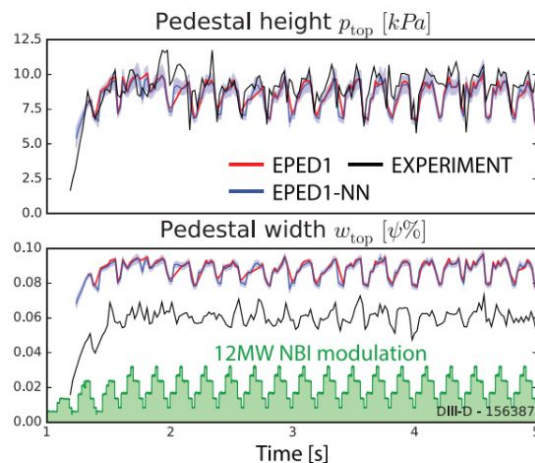
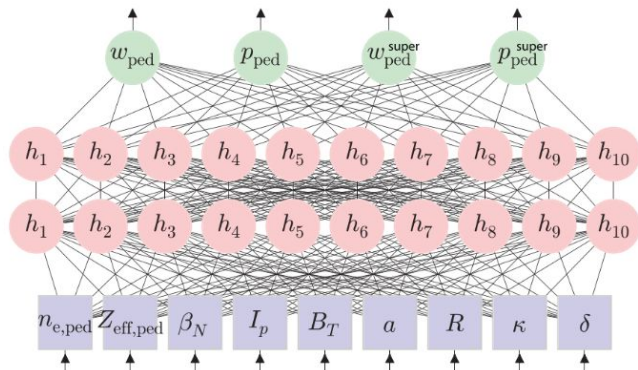
Basic surrogate modelling recipe the same for different underlying physics components in integrated modelling, using standard supervised learning techniques



1. Physics (reduced order) model verified against high-fidelity simulations and validated against experiments
2. Use HPC to generate large datasets of model calculations in relevant tokamak parameters
3. Neural networks used to learn the model nonlinear input-output static map, based on the training set
4. Use the trained NN as a fast surrogate model for tokamak simulation. Typically orders of magnitude faster than original model

Examples of supervised learning use-cases for faster integrated modelling

1. Pedestal height and width



- Learned surrogates of well-established EPED model [Snyder PoP 2009] for pedestal pressure and width.
- Standard multi-layer-perceptron (MLP) topology (more on this later)
- Accelerates H-mode modelling

EPED-NN: Menighini et al Nucl. Fusion 2017

EuroPED-NN: Panera Alvarez et al, PPCF 2024

Gillgren et al, NF 2022

From Menighini et al, NF 2017

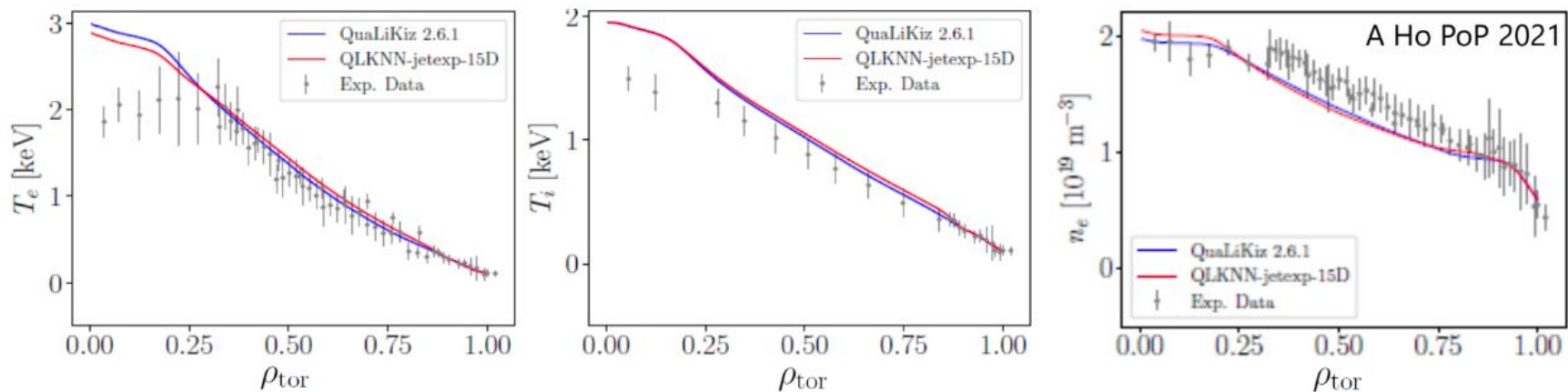
Examples of supervised learning use-cases for faster integrated modelling

2. Surrogate modelling of reduced turbulence model databases (deep dive later)

Comparing reduced turbulence model QuaLiKiz and Qualikiz-NN (QLKNN) for a test JET scenario

JINTRAC-QuaLiKiz: simulation took 4 hours with 16 cores

JINTRAC-QLKNN-jetexp-15D: simulation took 5 minutes with 1 core (and QLKNN not the bottleneck)



TGLF neural networks: Meneghini NF 2017, NF 2020

QuaLiKiz neural networks: Citrin NF 2015, van de Plassche PoP 2020, Ho PoP 2021, Hamel 2025

Gaussian process regression of GS2 (microtearing): Hornsby et al PoP 2024

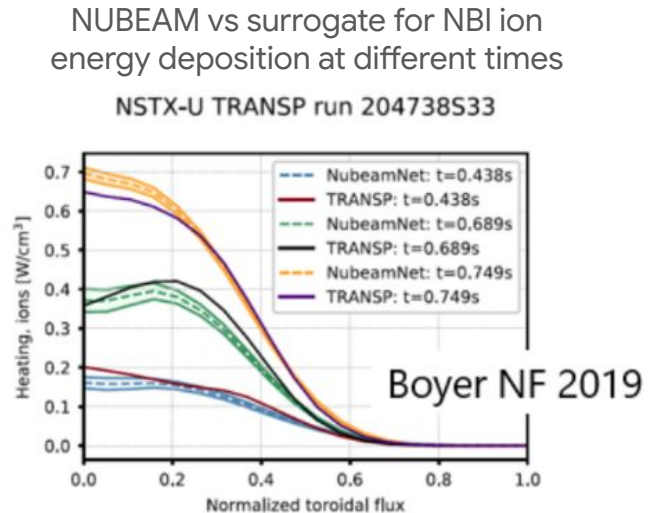
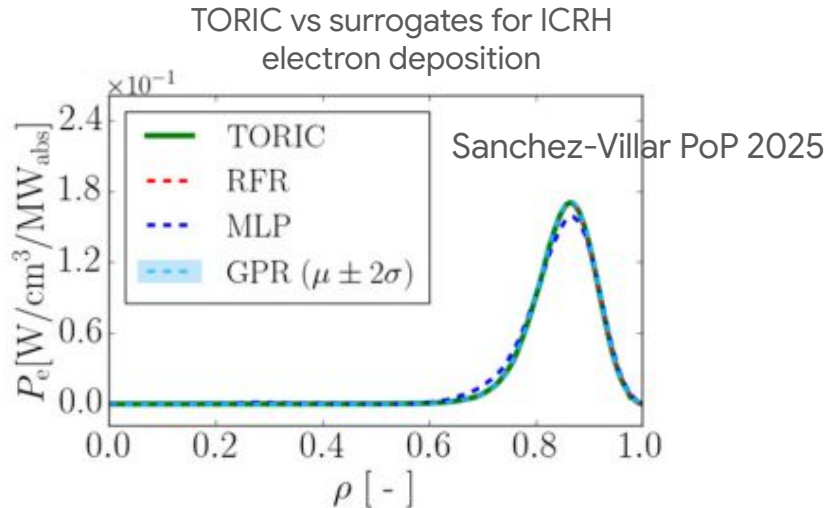
NN regression of quasilinear GENE for ITER: Citrin et al PoP 2023

Examples of supervised learning use-cases for faster integrated modelling

3. Heating and current drive

For heating codes, inputs and outputs are 1D profiles. Techniques for dimensionality reduction (e.g. analytical parameterizations, Principal Component Regression) used to reduce surrogate input + output vector sizes.

- TORIC ICRH heat deposition surrogates: random forests, MLPs, Gaussian Process Regression
 - Sanchez-Villar et al, PoP 2025, NF 2024, Wallace et al APS 2024
- NUBEAM neutral beam injection (NBI) surrogates: Boyer NF 2019, Morosohk FED 2021, Wang FED 2023
- TORBEAM ECRH surrogate: Rothstein PPCF 2025



Examples of supervised learning use-cases for faster integrated modelling

4. MHD prediction for disruption predictions and integrated modelling

- Learned surrogate of tearing mode predictions
 - Disruption precursors: to be avoided!
- Hybrid data and model driven approaches promising for disruption prediction [Piccione NF 2020, DECAF]
- MHD predictors used in RL schemes for tearing mode avoidance [Seo Nature 2024]

Tearing mode predictors

Seo et al, IJCNN 2023

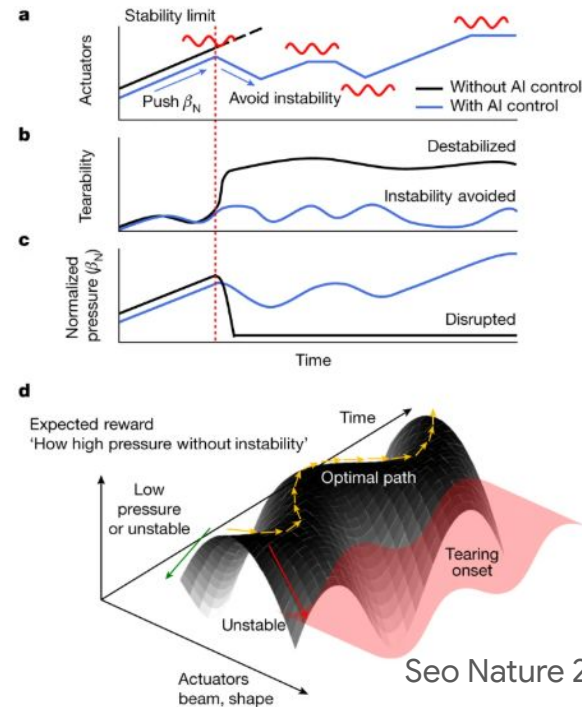
Xu et al, PoP 2024

MHD identification

Long NF 2025

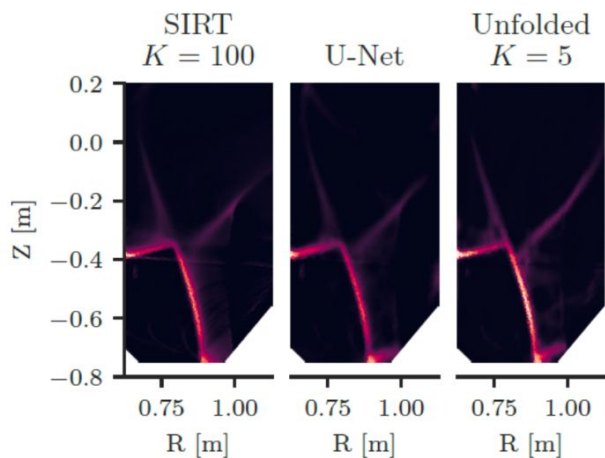
Kong PPCF 2023

Fig. 2: Illustration of the tokamak control by the AI tearing-avoidance system and the plasma responses.

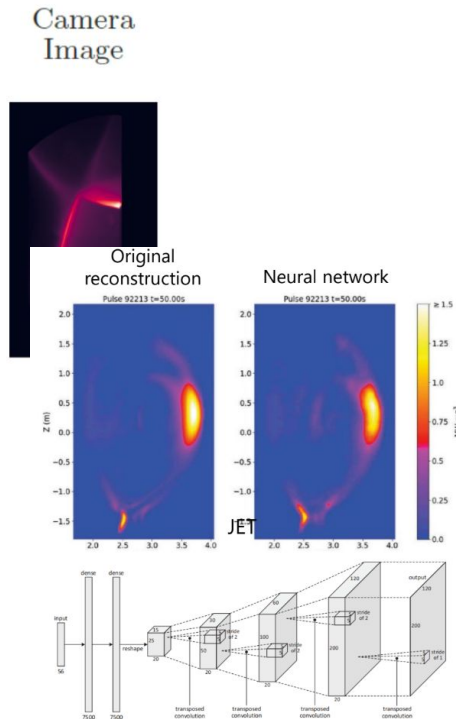


Supervised learning can also accelerate diagnostics, providing more information to controllers and enabling development of more robust advanced control

Machine learning enhanced tomographic reconstruction of multispectral imaging on TCV with U-Nets



L. van Leeuwen, PPCF 2025

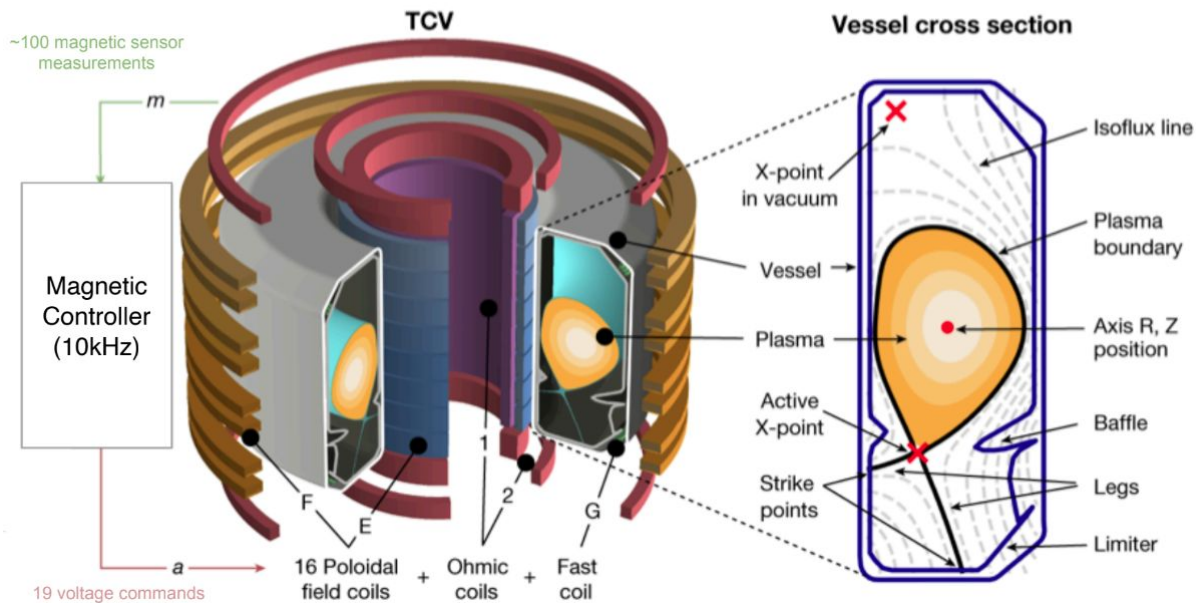


Ferreira FST 2018

ML can accelerate diagnostic analysis for controllers, and high-throughput scientific analysis

- Tomographic inversion
 - Soft X-rays, deconvolutional NN (1D chords \rightarrow 2D image) Ferreira et al FST 2018. Factor 100 million speed up!
 - Multispectral imaging with U-Nets, van Leeuwen et al PPCF 2025
 - Gaussian Process Tomography for error estimates (Matos RSI 2020)
- Equilibrium reconstruction
 - EFIT-NN, NF 2019
 - Eqnet, Wai NF 2022
 - Rutigliano PPCF 2025

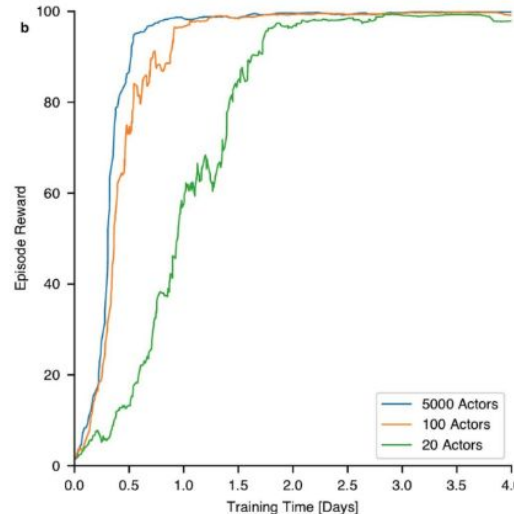
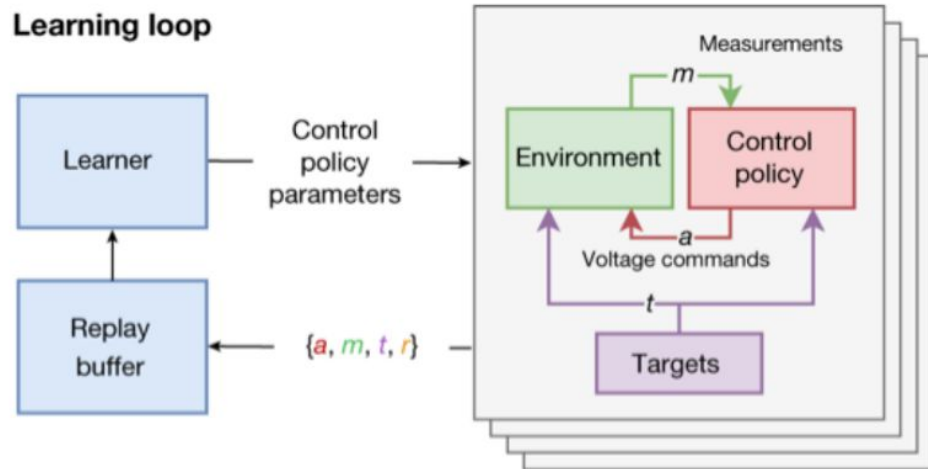
Example of RL interacting with integrated modelling for scenario optimization + control. Here focusing solely on the magnetic equilibrium control problem



Control plasma shape and position

- Plasma subject to pressure gradient and magnetic fields forces
- External actuators: magnetic coil set
- Partial state of plasma inferred from magnetic sensors
- Fast magnetic equilibrium codes available and can be used as environment for training agent with RL (more general than traditional controllers)

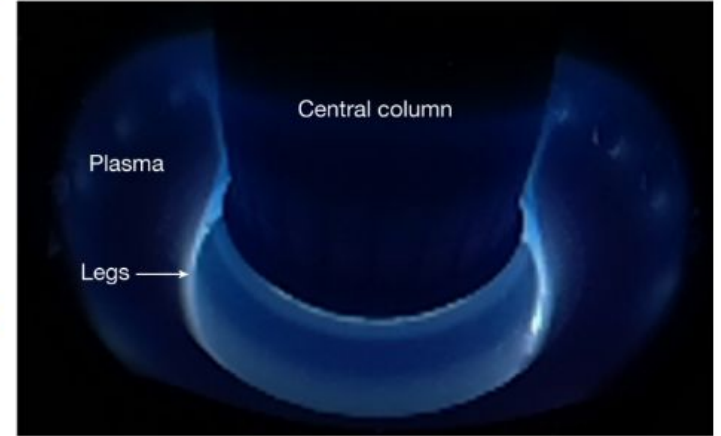
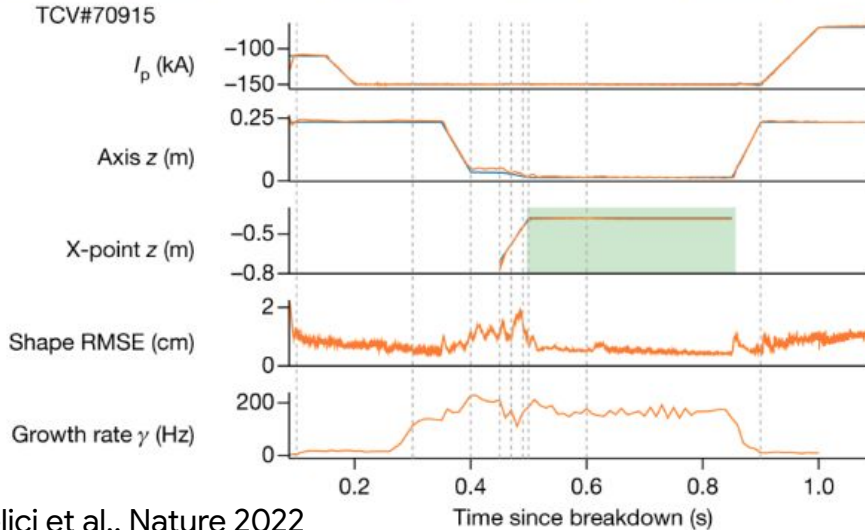
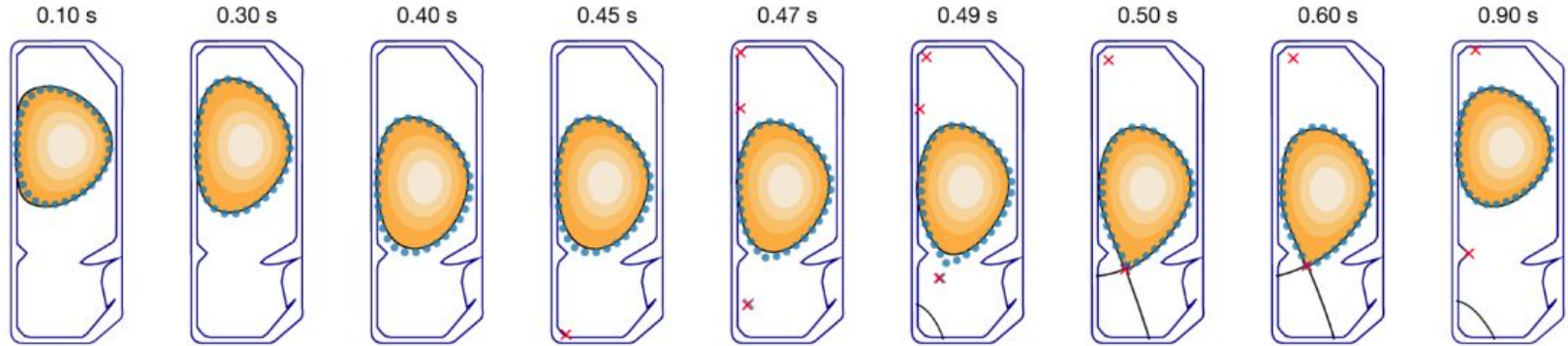
The learning setup



Algorithm details

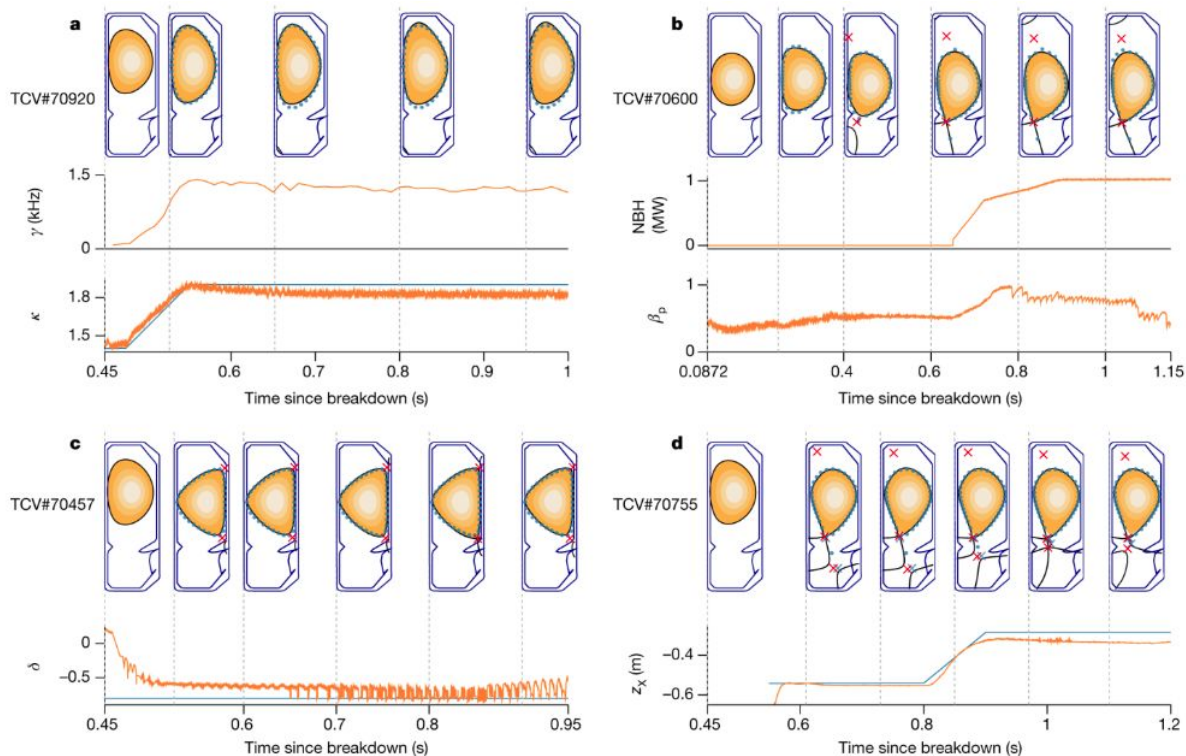
- MPO: off-policy RL algorithm
 - Learner (critic) is large recurrent network (LSTMs) and learned the value (Q) function generated by actors interacting with simulation environment with policy π
 - Actors iteratively learn policy π by sampling actions from the learned Q function
 - Policy is simple MLP for fast realtime implementation. 19 degrees of freedom (inputs) for TCV
- Distributed implementation: many actors in parallel, results fed to replay buffer continuously for learning
- Actor (policy) sees only synthetic diagnostics. Critic can have access to full state

Result - demonstration shot



Inside view at 0.6 s

Various plasma shapes controlled in TCV with RL



- Next step: extending simulation environment with more multiphysics integrated modelling
- Multiphysics simulation with multiple ML-surrogates can provide sufficient speed and accuracy.
- Appropriate software frameworks help. More on this later.

Further topics out of scope of this lecture

Neural operator networks

Learn PDE solutions from models or data, and make faster predictions, examples:

- Fourier Neural Operators emulating MHD simulations of plasma blobs, and time-series prediction of experimental data [Gopakumar et al, NF 2024]
- Neural Green Operators for PDE acceleration [Melchers et al, <https://arxiv.org/pdf/2406.01857>]

Bayesian Optimization Workflows

When optimizing an expensive "black box" function, e.g. gradient-flux matching for a nonlinear gyrokinetic code, build cheap surrogates "on-the-fly" and use those to accelerate optimization

- PORTALS framework , Rodriguez-Fernandez et al NF 2024. Key technique for enabling ultra-high-fidelity integrated modelling, using expensive turbulence (and other) models.

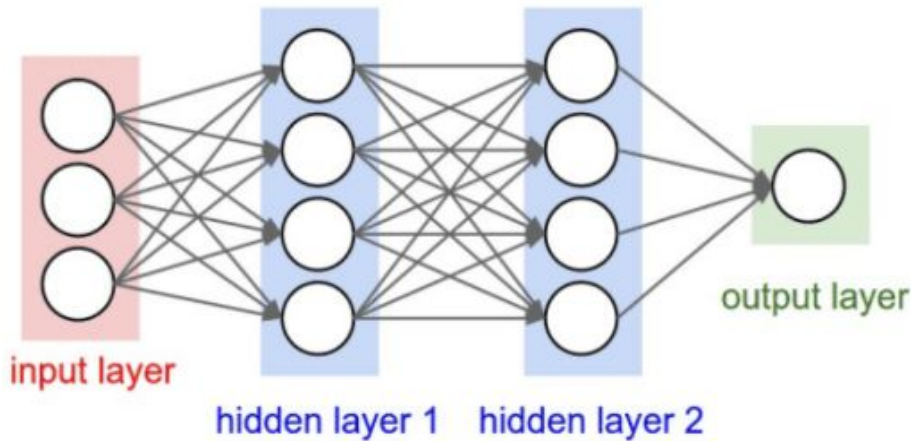
Further reviews

Anirudh et al, "2022 review of data-driven plasma science", doi: 10.1109/TPS.2023.3268170

Pavone et al, 2023, "Machine learning and Bayesian Inference in Nuclear Fusion Research: an overview", doi: 10.1088/1361-6587/acc60f

Schissel et al, 2025, "Digital Twins for Fusion Research", doi: 10.1063/5.0273586

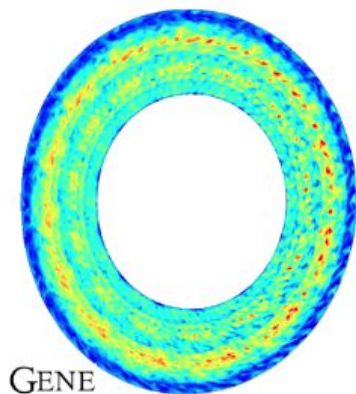
Deep-dive: A neural network surrogate for tokamak turbulence



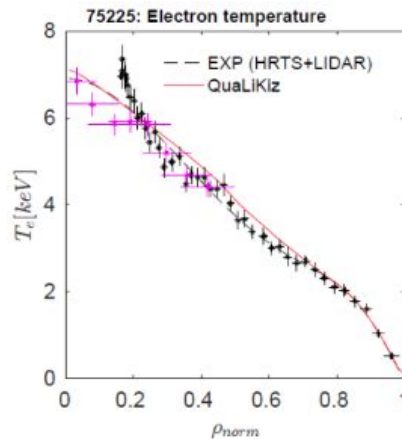
2

Build datasets of verified reduced order models, or mixed multifidelity datasets with high and reduced fidelity, and fit ML-surrogates to them

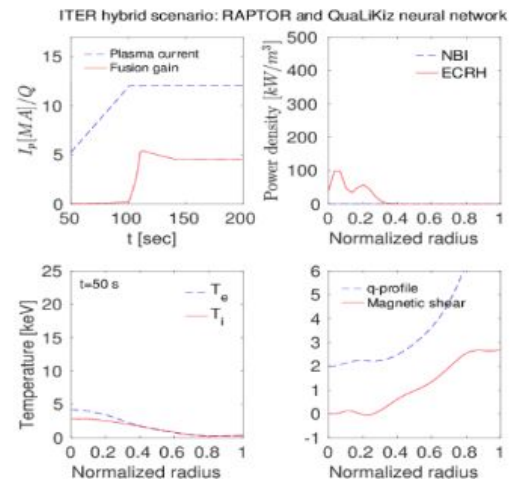
Bridging ~12 orders of magnitude in calculation speed



Expensive nonlinear turbulence modelling.
~100 million CPU hours per full-pulse sim
e.g. GENE [Jenko PoP 2000]
CGYRO [Candy JCP 2016]



Reduced order physics model. Quasilinear approximation
~100 CPU hours per full-pulse sim
e.g. QuaLiKiz [Bourdelle PPCF 2016]
TGLF [Staebler PoP 2007]



Learned surrogate models.
Realtime capable.
Shallow fully connected NNs typically sufficient

Approaches to training set generation:

1. Brute force lattice bounded by extremes found in experiments

Dataset used in QLKNN10D model
van de Plassche PoP 2020: $3 \cdot 10^8$ QuaLiKiz runs

Quantity	Range	# points
Wavenumbers (ion + electron scale) [$k_{\theta} \rho_s$]	0.1 - 36	18
Ion temperature gradient [R/L_{T_i}]	0 - 14	12
Electron temperature gradient [R/L_{T_e}]	0 - 14	12
Density gradient [$R/L_{n_e} (\equiv R/L_{n_i})$]	-5 - 6	12
Magnetic pitch angle [q]	0.66 - 15	10
Magnetic pitch angle shear [\hat{s}]	-1 - 5	10
Normalized radius [r/R]	0.03 - 0.33	8
Temperature ratio [T_i/T_e]	0.25 - 2.5	7
Collisionality [ν^*]	10^{-5} - 1	6
Impurity content [Z_{eff}]	1 - 3	5

QLKNN10D dataset on Zenodo:
doi.org/10.5281/zenodo.349065

More recent QuaLiKiz dataset

- $\sim 10^9$ points with up-to-date QuaLiKiz version
- Includes impurity density information and data relevant in L-mode near-edge

<https://zenodo.org/records/8017522>
<https://zenodo.org/records/8106431>

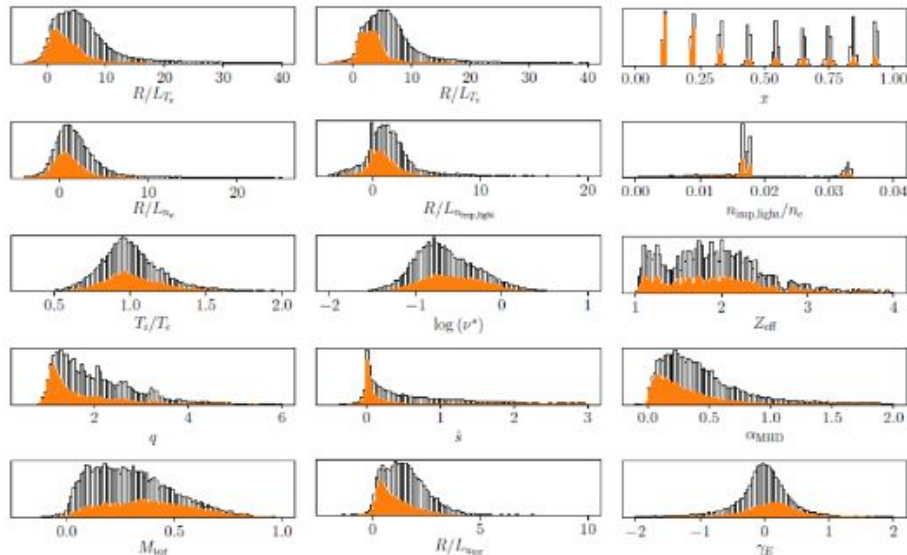
Used to fit new QLKNN_7_11 model [Hamel 2025]
https://github.com/google-deepmind/fusion_surrogates

- Lattice trainings sets can generalize to tokamak parameters outside of present-day devices
 - At the cost of reduced input dimensionality due to "curse of dimensionality"
- We clearly know beforehand when NN is extrapolating
- In practice, better not to uniformly fill lattice, but to generate in parallel to model training and stop generating when new data no longer informative

Approaches to training set generation:

2. Restrict training set to experimentally relevant sub-space

Aaron Ho Nucl. Fusion 2019
Aaron Ho PoP 2021



Database of ~2000 JET shots, ~7 timeslices each

Automated data fitting pipeline with Gaussian Process Regression (GPR)

Sampled for reduced turbulence model inputs.
Enrich by varying gradient quantities with fit errors

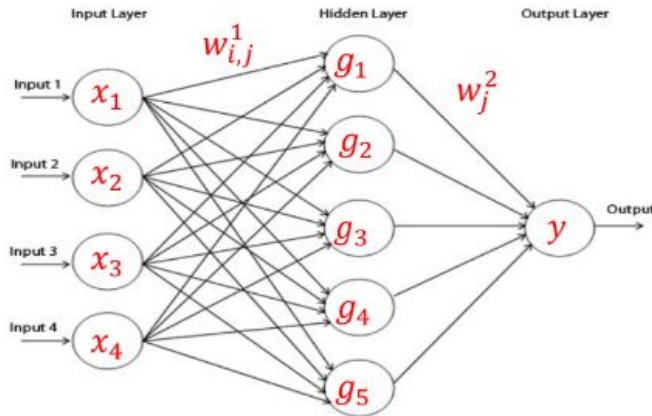
~ 21 million input-output mappings

- Includes more physics: plasma rotation, impurity species, additional magnetic geometry inputs
 - Caveat. Sampling only space populated by existing experiments restricts exploration of new regimes and devices.
- Use methods for extrapolation detection, e.g. analysing variance of ensemble of networks, noise contrastive priors.
- In practice: best to use **Active Learning** techniques for efficient sampling of datapoints, especially key for expensive ground-truth evaluation. See Zanisi et al NF 2024 for QLKNN active learning demo.

Simple feed forward neural networks sufficient for quasilinear transport model surrogate generation: FFNNs, a.k.a. "Multiple Layer Perceptrons" (MLPs)

INPUT (at local plasma position)

- Temperature gradient
- Density gradient
- Magnetic field geometry
- Ion to electron temp ratio
- etc.



OUTPUTS
Particle fluxes
Heat fluxes
etc.

$$y = \sum w_j^2 g_j \left(\sum w_{i,j}^1 x_i \right)$$

With, e.g. $g(x) = \tanh(x)$

Optimize weights $w_{i,j}$ by minimizing cost function: $C = \sum_N (t_N - y_N)^2 + \lambda \sum (w_{ij})^2$

t_N are target values from generated dataset ("training set")

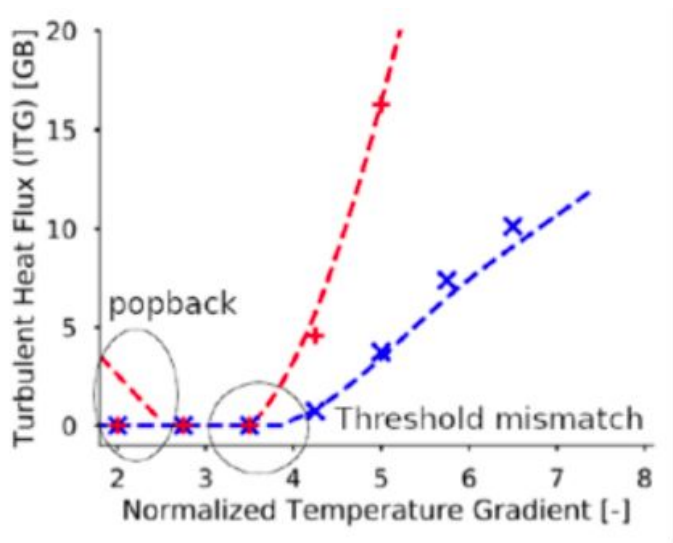
λ is the (L2) regularization factor.

Additional advantage: provides an analytical formula with analytical derivatives.

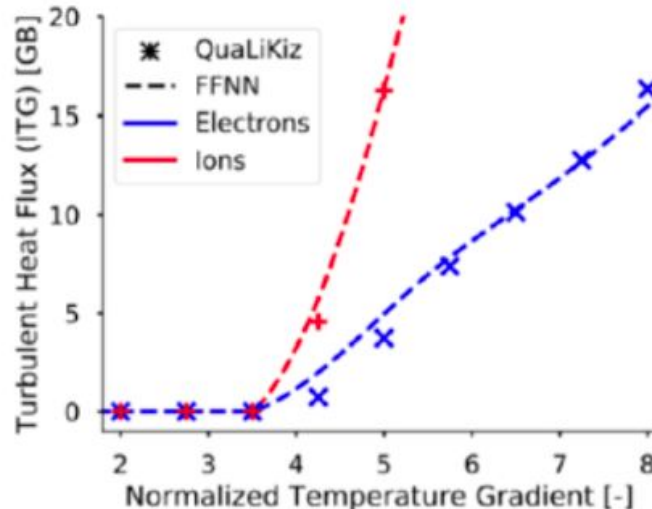
Turbulent flux derivatives with respect to inputs important for implicit timestep PDE solvers and trajectory optimization

Incorporate prior physics knowledge into NN training: e.g. tokamak turbulence transport has sharp “critical gradient threshold”

- Global statistics (MSE) can be less important than local features (critical gradient instability threshold)
- Same threshold for all transport channels (ion heat, electron heat, particles) essential
- Achieved by customising training targets and optimisation cost function with physics constraints



Bad fit



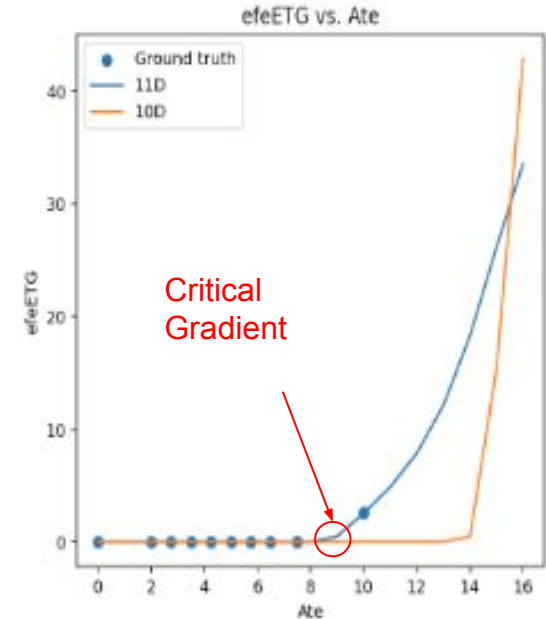
Good fit

Various underlying instability "modes" combine to describe turbulent fluxes. Each have their own thresholds and can be separated in output

- Ion Temperature Gradient (ITG) - most important for reactor-relevant regimes
 - Driven by ion temperature gradient
 - Leading transport: ion heat flux
 - Low frequency
- Trapped Electron Modes (TEM)
 - Driven by electron density gradient
 - Leading transport: electron heat flux
 - Intermediate frequency
- Electron Temperature Gradient (ETG)
 - Driven by electron temperature gradient
 - Leading transport: electron heat flux (does not drive ion heat flux or particle flux)
 - High frequency
- Modes not modelled by QLKNN
 - Micro-Tearing Modes (MTM)
 - Kinetic Ballooning Modes (KBM)

Critical gradient model

- All fluxes caused by a given mode are null below a certain threshold of the driving gradient
- This needs to be enforced in the surrogate model
- QLKNN trick:
 - Predict **leading flux** for each mode
 - Non-leading flux are predicted as a **ratio**
 - Enforces that zero leading flux \rightarrow zero secondary fluxes



Single neural network with multiple outputs. Turbulent fluxes are done in post-processing combining these outputs

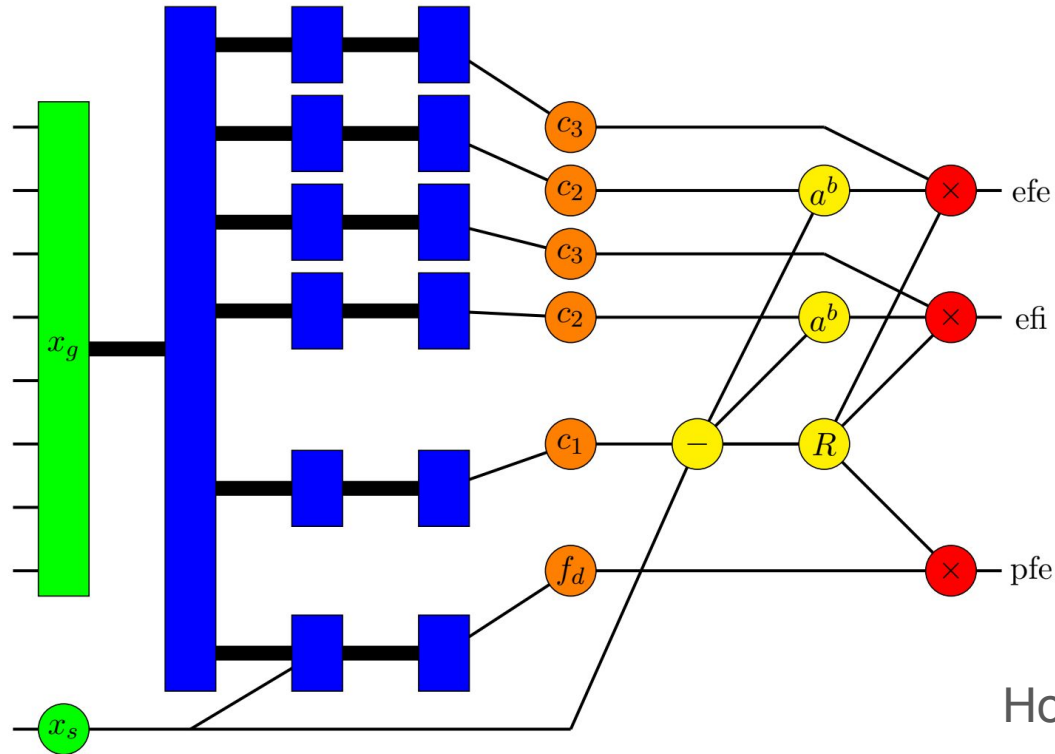
- itgleading (ion heat flux - ITG)
- itgqediv (electron heat flux ratio - ITG)
- itgpfediv (electron particle flux ratio - ITG)
- temleading (electron heat flux -TEM)
- temqidiv (ion heat flux ratio -TEM)
- tempfediv (electron particle flux ratio - TEM)
- etgleading (electron heat flux - ETG)
- gamma_max (maximum growth rate on ion-Larmor-radius-scales,
to be used for rotation shear turbulence suppression model)

Training and Evaluation

- Trained on MSE and stability loss
- Stability loss enforce zero flux in the stable region
- Output leading fluxes are clipped to zero at inference time (no negative leading fluxes)
- Alternatively, we can incorporate physics in the network structure

$$\text{Loss} = \begin{cases} \frac{1}{N} \sum_{i=1}^N (Y_i - y_i)^2, & \text{if } Y_i \neq 0 \text{ (MSE Loss),} \\ \lambda_{\text{stab}} \frac{1}{N} \sum_{i=1}^N \max(0, y_i + c_{\text{stab}}), & \text{if } Y_i = 0 \text{ (Stability Loss)} \end{cases}$$

Alternative: Embed physics in the network structure

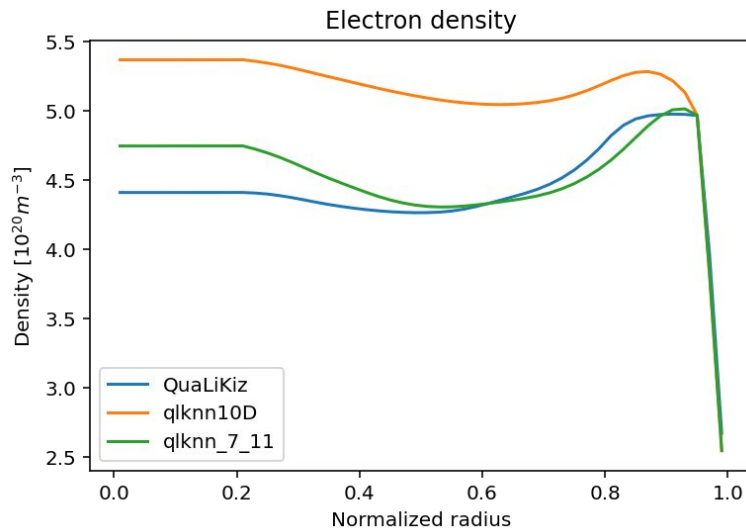
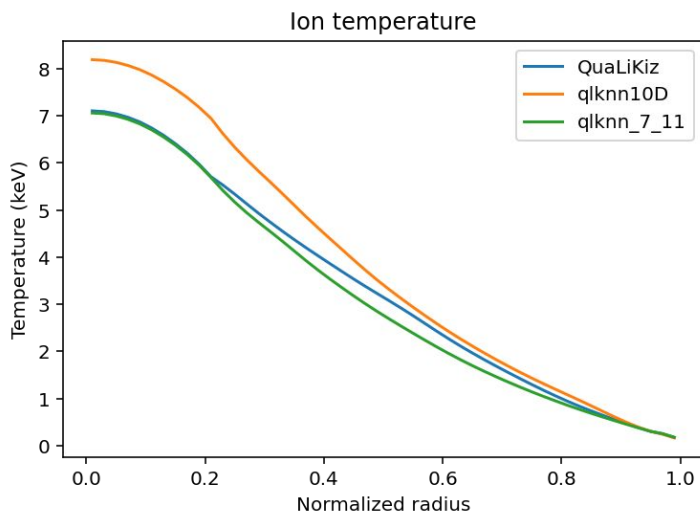


Horn P. 2021

Evaluation

- Best loss is not necessarily the best model for integrated modelling
- Integrated modelling involves an iterative PDE solver
- Expectation of smoothness in the output domain
- Speed is also a consideration
- We ran full integrated modelling (TORAX) simulations as part of validation
- Metrics
 - **Smoothness of spatial and temporal temperature profiles**
 - **Number of TORAX nonlinear solver iterations**
 - **Total simulation time**
- Post-training evaluation comparing dozens of scenarios

Sample result of new QLKNN_7_11 surrogate, compared to ground truth and previous best surrogate

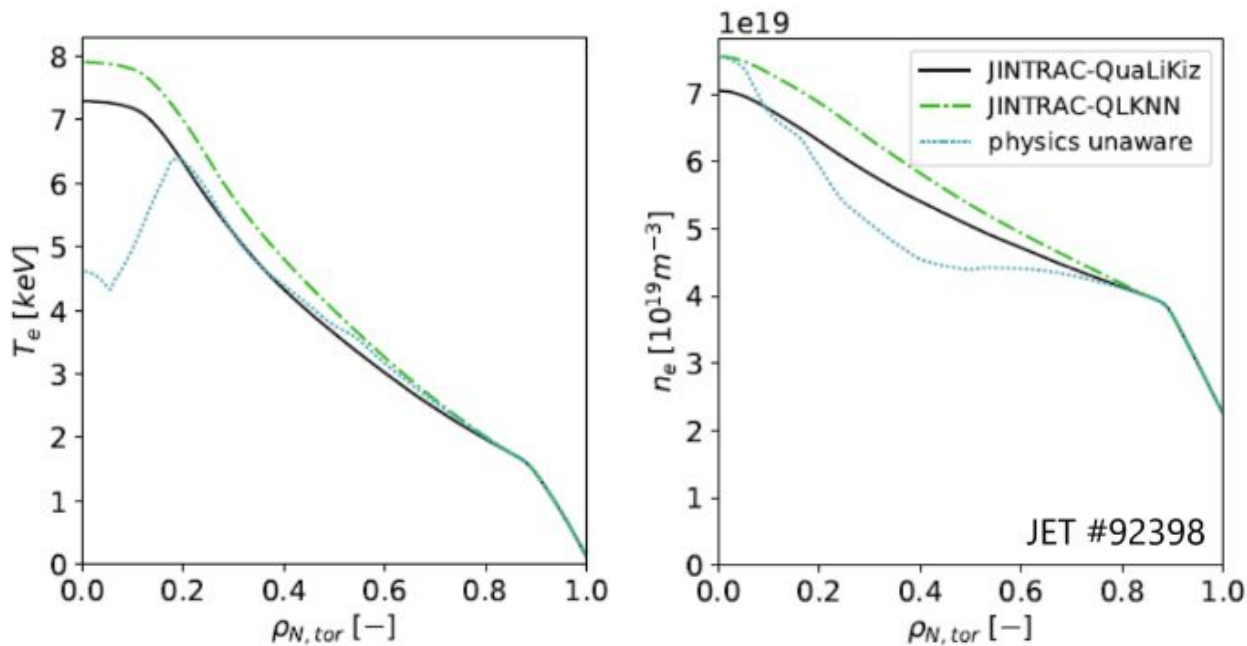


Improved agreement to ground-truth compared to previous general QuaLiKiz surrogate QLKNN10D in multiple regimes: H-mode, L-mode, ITER-like, SPARC-like, up to LCFS, various collisionalities

QLKNN_7_11 model released open-source: https://github.com/google-deepmind/fusion_surrogates/

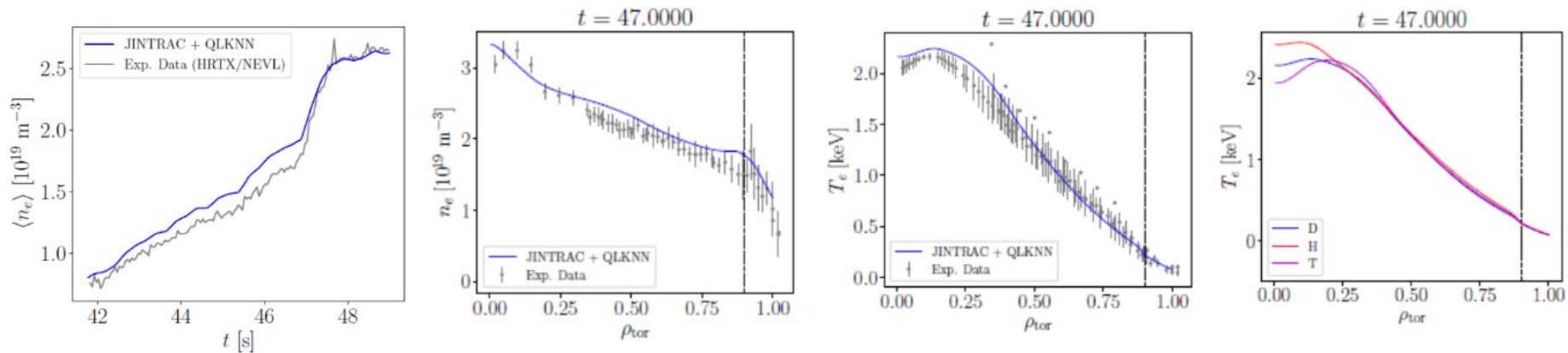
Naive NN fit with RMS only leads to poor results in integrated modelling

NO special cost function, NO special train targets, *same RMS of fit!*



JINTRAC-QLKNN-hyper-10D vs a *network* trained on the full fluxes at $t=22.75s$ [K.L. van de Plassche et al. PoP 2020]

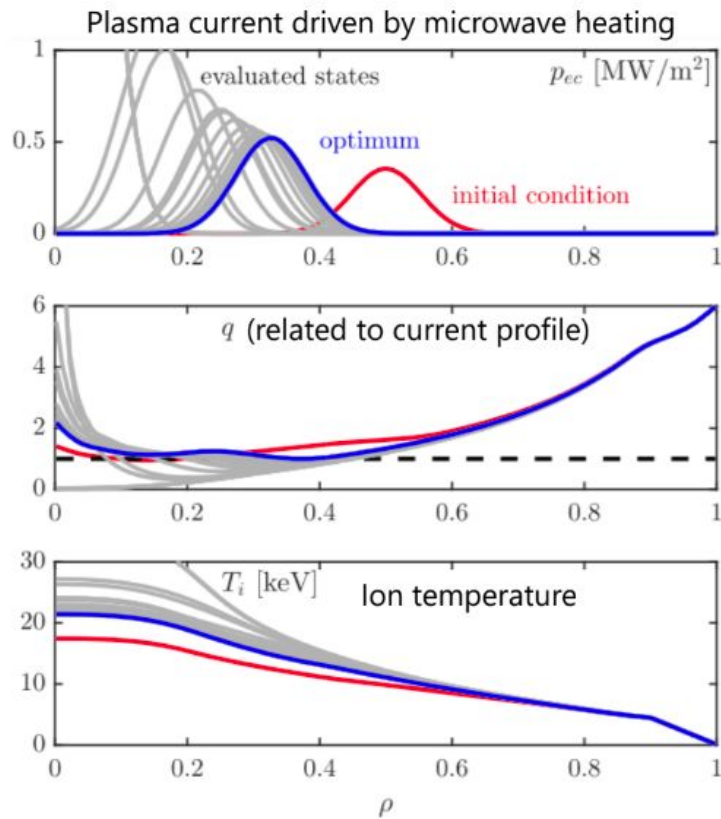
Applications: QLKNN-jetexp-15D model demonstrated current ramp-up optimization of a JET high-performance scenario in Tritium



- Good agreement with deuterium-reference in time-dependent simulation from early ramp-up
- Electron temperature hollowing in inner-core predicted. Worsened hollowing with heavier isotope also predicted
 - Can be deleterious for operations. Leads to plasma current profile that destabilizes MHD \rightarrow disruption
- Modelling predicted necessary increase in density in tritium scenario for controlling the temperature profile, in line with subsequent experiments. Modelling shed light on underlying mechanisms [A Ho APS invited 2021, Nucl. Fusion 2023]
- Each run ~2 hours instead of 2 weeks! Learned surrogate a key enabling factor in reaching scientific goals

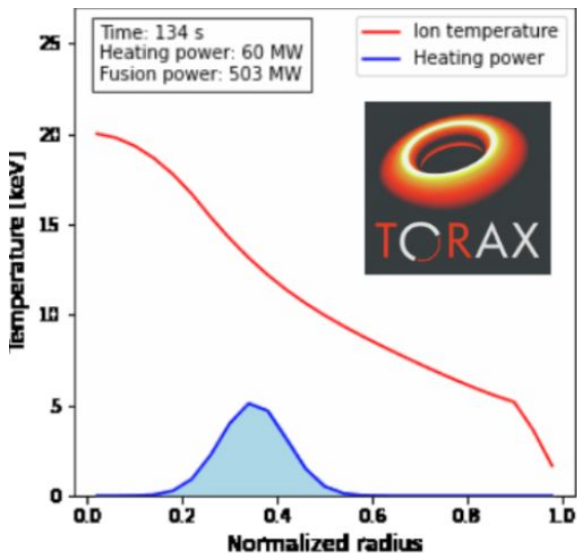
Applications:

Optimization demonstration for a ITER high performance scenario



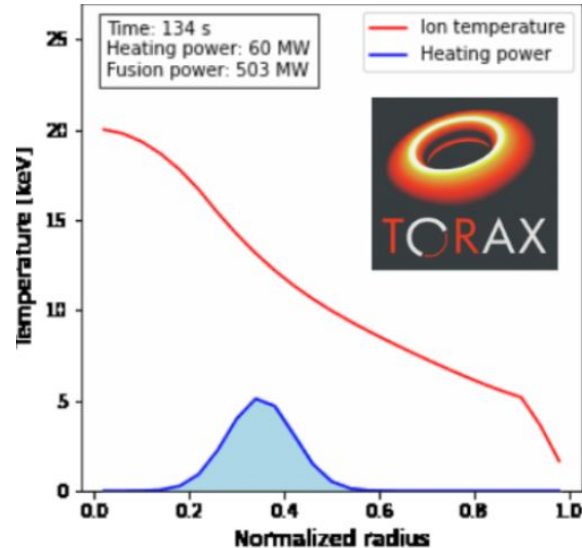
- QKNN-hyper-10D implemented within RAPTOR control-oriented tokamak simulator
- Stationary-state solver in RAPTOR demonstrated ITER high-performance scenario optimization [S. van Mulders Nucl. Fusion 2021]
- Constrained optimization for maximal fusion power, scanning over microwave heating source launcher angles
 - Leads to modification in plasma current profile
 - Impacts turbulence and confinement
- Optimization routine delivers solution in few minutes. Would take weeks/months with full QuaLiKiz

Modern simulation frameworks utilizing ML-surrogates for optimization and control








3

TORAX is our differentiable tokamak core transport simulator built in Python using JAX, solving for core temperatures, density, and current diffusion



TORAX motivated by requirements for pulse simulation, planning, and controller design tasks

-  Fast and accurate forward modelling
-  Differentiable for accurate nonlinear PDE solvers, gradient-driven optimization and data-driven model parameter identification
-  Easy incorporation of physics model ML-surrogates; higher fidelity simulation without sacrificing speed
-  Modularity for coupling within various workflows, additional physics models, and controllers
-  Scalable for large-scale validation and UQ

TORAX motivated by requirements for pulse simulation, planning, and controller design tasks

Builds on ideas developed over many years in the fusion community for tokamak pulse planning, optimization, and control:

RAPTOR [Felici PPCF 2012]
METIS [Artaud NF 2018]
COTSIM [Pajares NF 2021]
FENIX [Fable PPCF 2022]

But in Python using JAX which delivers new capabilities and facilitates others

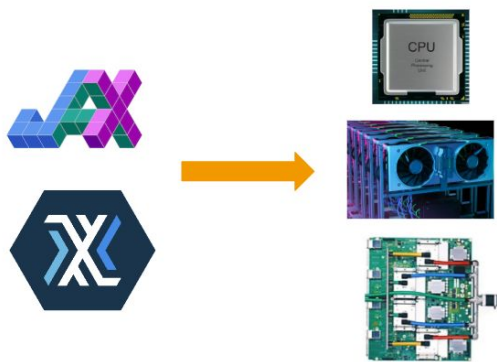
- [Auto](#)differentiation, fast compiled code, easy ML-surrogate coupling
- Aimed for SciML workflows with hybrid model + data-driven simulation and optimization.

See also POPSIM [Wang, Nature. Comm. Physics. 2025]

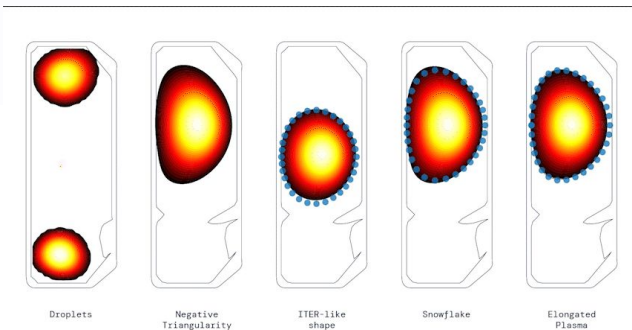
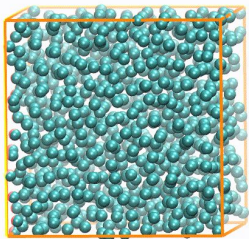
- Portable, modular, extensible

JAX enables fast compiled and differentiable simulation with NumPy-like API

- Python library originally developed by Google, is "NumPy on the CPU, GPU, TPU" (<https://jax.readthedocs.io/>)
 - Originally developed for AI/ML. Increasing applied for scientific computing.
- Uses an updated version of Autograd to automatically differentiate NumPy-like code
 - Automatic transformation of functions to their *analytic* derivatives
- Uses XLA (Accelerated Linear Algebra) JIT (just in time) compiler for speed
 - Transforms functions to fast compiled versions
 - Abstracts away parallelization
 - Same code can run seamlessly on CPU or accelerators



Usages

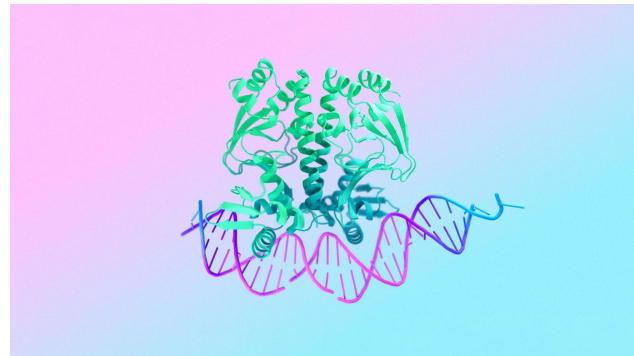


Gemini

Whisper JAX

This repository contains optimised JAX code for OpenAI's [Whisper Model](#), largely built on the 🤗 Hugging Face Transformers Whisper implementation. Compared to OpenAI's PyTorch code, Whisper JAX runs over **70x** faster, making it the fastest Whisper implementation available.

The JAX code is compatible on CPU, GPU and TPU, and can be run standalone (see [Pipeline Usage](#)) or as an inference endpoint (see [Creating an Endpoint](#)).



Basic introduction to the Python JAX library



Simple autodiff example

jnp is "JAX NumPy"; grad is imported from JAX

```
def sum_logistic(x):  
    return jnp.sum(1.0 / (1.0 + jnp.exp(-x)))  
  
x_small = jnp.arange(3.)  
derivative_fn = grad(sum_logistic)  
print(derivative_fn(x_small))
```

```
[0.25      0.19661194 0.10499357]
```

Similarly

- `jax.jit` transforms functions to compiled version
- `jax.vmap` transforms function to parallelized version

TORAX governing equations: set of 1D flux-surface-averaged **nonlinear** transport PDEs

Ion and electron heat equations

$$\frac{3}{2}V'^{-5/3} \left(\frac{\partial}{\partial t} - \frac{\dot{\Phi}_b}{2\Phi_b} \frac{\partial}{\partial \hat{\rho}} \right) [V'^{5/3} n_i T_i] =$$
$$\frac{1}{V'} \frac{\partial}{\partial \hat{\rho}} \left[\chi_i n_i \frac{g_1}{V'} \frac{\partial T_i}{\partial \hat{\rho}} - g_0 q_i^{\text{conv}} T_i \right] + Q_i$$

$$\frac{3}{2}V'^{-5/3} \left(\frac{\partial}{\partial t} - \frac{\dot{\Phi}_b}{2\Phi_b} \frac{\partial}{\partial \hat{\rho}} \right) [V'^{5/3} n_e T_e] =$$
$$\frac{1}{V'} \frac{\partial}{\partial \hat{\rho}} \left[\chi_e n_e \frac{g_1}{V'} \frac{\partial T_e}{\partial \hat{\rho}} - g_0 q_e^{\text{conv}} T_e \right] + Q_e$$

Electron density equation

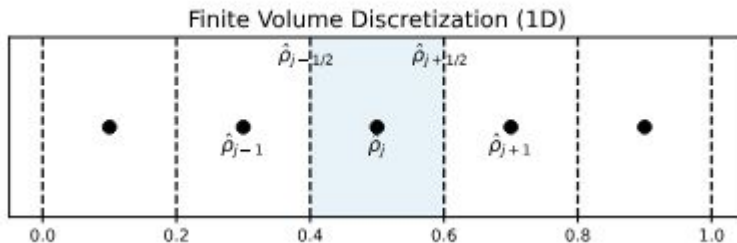
$$\left(\frac{\partial}{\partial t} - \frac{\dot{\Phi}_b}{2\Phi_b} \frac{\partial}{\partial \hat{\rho}} \right) [n_e V'] =$$
$$\frac{\partial}{\partial \hat{\rho}} \left[D_e \frac{g_1}{V'} \frac{\partial n_e}{\partial \hat{\rho}} - g_0 V_e n_e \right] + V' S_n$$

Current diffusion equation

$$\frac{16\pi^2 \sigma_{||} \mu_0 \hat{\rho} \Phi_b^2}{F^2} \left(\frac{\partial \psi}{\partial t} - \frac{\dot{\rho} \Phi_b}{2\Phi_b} \frac{\partial \psi}{\partial \hat{\rho}} \right) =$$
$$\frac{\partial}{\partial \hat{\rho}} \left(\frac{g_2 g_3}{\hat{\rho}} \frac{\partial \psi}{\partial \hat{\rho}} \right) - \frac{8\pi^2 V' \mu_0 \Phi_b}{F^2} \langle j_{ni} \rangle$$

- Moment equations of underlying kinetic equations
 - Toroidal symmetry + flux-surface averaging reduces to 1D
 - Timescale separation of turbulence, sources: function calls of reduced order models ideally verified against high-fidelity
- ψ (poloidal magnetic flux) boundary condition options
 - Neumann (prescribed total current)
 - Dirichlet for next timestep (edge V_{loop})
- Any subset of these equations are evolved
- Non-evolved profiles, initial conditions, source parameters, etc, set by
 - Config (xarray, numpy, python primitives, IMAS)
 - Other models, controllers, in between PDE steps within flexible composable workflows using TORAX API

Spatial discretization: finite-volume-method with bespoke JAX fvm package



- TORAX JAX 1D finite-volume-method package inspired by FiPy¹ and uses similar API
 - Constrained to solving convection-diffusion equations
- For (particle) convection, power-law α -weighting scheme based on Péclet number
- Constructs nonlinear/linear systems of equations for solvers based on solver method
 - Predictor corrector: implicit PDE where nonlinearity of PDE coefficients treated by fixed point iteration
 - Newton-Raphson nonlinear solver: iterative root finding for PDE residual

¹<https://www.ctcms.nist.gov/fipy/>

Newton-Raphson illustration: simple example with heat diffusion

$$\frac{\partial T_k}{\partial t} = \frac{\partial}{\partial r} \left(\chi(T_{k+1}) \frac{\partial T_{k+1}}{\partial r} \right) + S(T_{k+1})$$

Fully implicit simple nonlinear diffusion equation

$$\frac{\vec{T}_{k+1} - \vec{T}_k}{\Delta t} - \bar{A}(\chi(T_{k+1}))\vec{T}_{k+1} - \vec{S}(T_{k+1}) \equiv \vec{R}(T_{k+1}, T_k) = 0$$

Discretize and define nonlinear system of equations to be solved (residual)

$$\vec{R}(T_{old}, T_k) + \bar{J}|_{(\vec{T}_{old})} (\vec{T}_{new} - \vec{T}_{old}) = 0$$

Newton-Raphson: starting from initial guess of T_{old} (e.g. T_k), iteratively solve linear system for T_{new} , until $R(T_{new}, T_k)$ within tolerance

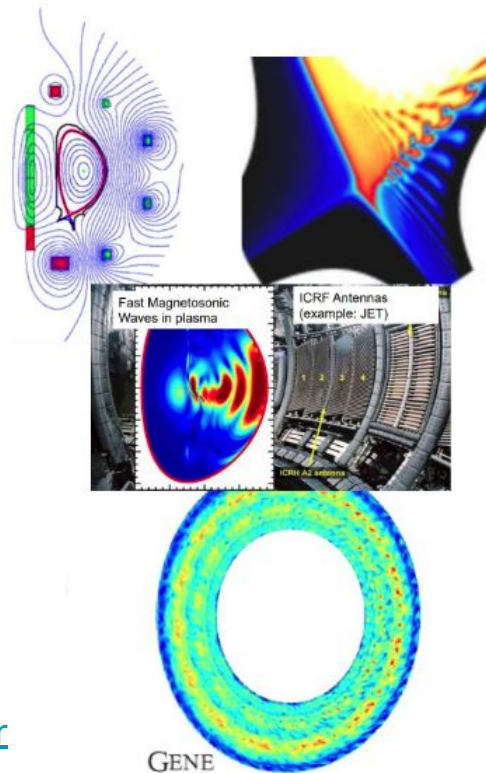
All the physics goes into the residual function, and the Jacobian is JAX magic ✨

jacobian = jax.jacfwd(residual)

- TORAX has simple linesearch to ensure good Newton steps (physical T_{new}), as well as Δt backtracking if no convergence

Presently implemented physics models/couplings (non-exhaustive list)

- ML-surrogates where available
 - Turbulence: QLKNN [van de Plassche 2020, Hamel 2025*]
 - ICRH heating: TORIC-NN for SPARC [Wallace APS 24]
- Fast analytical models where appropriate
 - Bootstrap current, neoclassical transport, ECCD
 - Semi-empirical transport models (e.g. BgB)
 - Mavrin polynomial fits to ADAS for line radiation, Bremsstrahlung
 - Fusion power, ion-electron collisional exchange, Ohmic power
 - Sawteeth
- Pre-calculated sequence of geometry inputs: wrappers for CHEASE, MEQ, EQDSK, (IMAS)
- Collaborations key
 - TORAX aims to be a natural target platform for community-wide ML-surrogates
 - ONNX data storage for portability
 - https://torax.readthedocs.io/en/v1.0.0/interfacing_with_surrogates.html
 - Design focus on modularity



* https://github.com/google-deepmind/fusion_surrogates (recent release)

TORAX open-source with permissive Apache 2.0 license

- Open source launch (v0.1.0) in June 2024
 - <https://github.com/google-deepmind/torax>
 - Notebook tutorials available
 - Latest release: v1.0.3
- Technical report on v0.1.0 + up-to-date online documentation
 - <https://arxiv.org/abs/2406.06718>
 - torax.readthedocs.io

On PyPI: `pip install torax`

arXiv > physics > arXiv:2406.06718

Physics > Plasma Physics

[Submitted on 10 Jun 2024 (v1), last revised 12 Jun 2024 (this version, v2)]

TORAX: A Fast and Differentiable Tokamak Transport Simulator in JAX

Jonathan Citrin, Ian Goodfellow, Akhil Raju, Jeremy Chen, Jonas Degraeve, Craig Donner, Federico Felici, Philippe Hamel, Andrea Huber, Dmitry Nikulin, David Pfau, Brendan Tracey, Martin Riedmiller, Pushmeet Kohli

We present TORAX, a new, open-source, differentiable tokamak core transport simulator implemented in Python using the JAX framework. TORAX solves the coupled equations for ion heat transport, electron heat transport, particle transport, and current incorporating modular physics-based and ML models. JAX's just-in-time compilation ensures fast runtimes, while its automatic differentiation capability enables gradient-based optimization workflows and simplifies the use of Jacobian-based PDE solver surrogates of physics models is greatly facilitated by JAX's intrinsic support for neural network development and inference. TORAX is verified against the established RAPTOR code, demonstrating agreement in simulated plasma profiles. TORAX provides a versatile tool for accelerating research in tokamak scenario modeling, pulse design, and control.

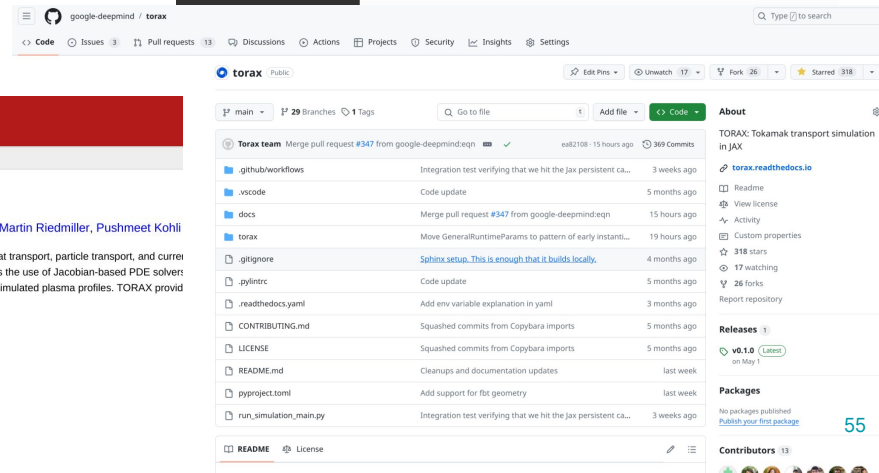
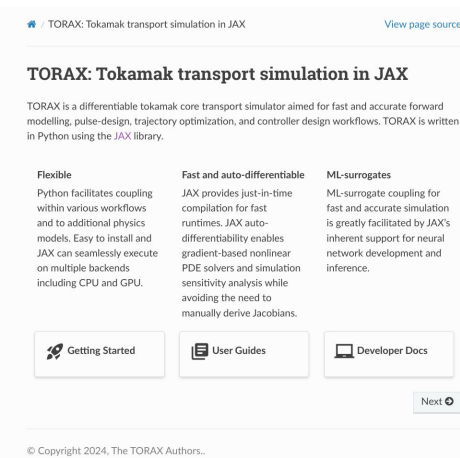
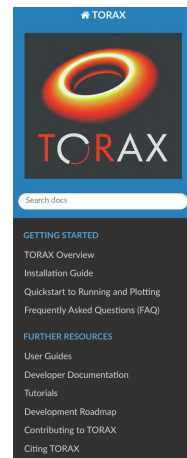
Comments: 16 pages, 7 figures

Subjects: Plasma Physics (physics.plasm-ph)

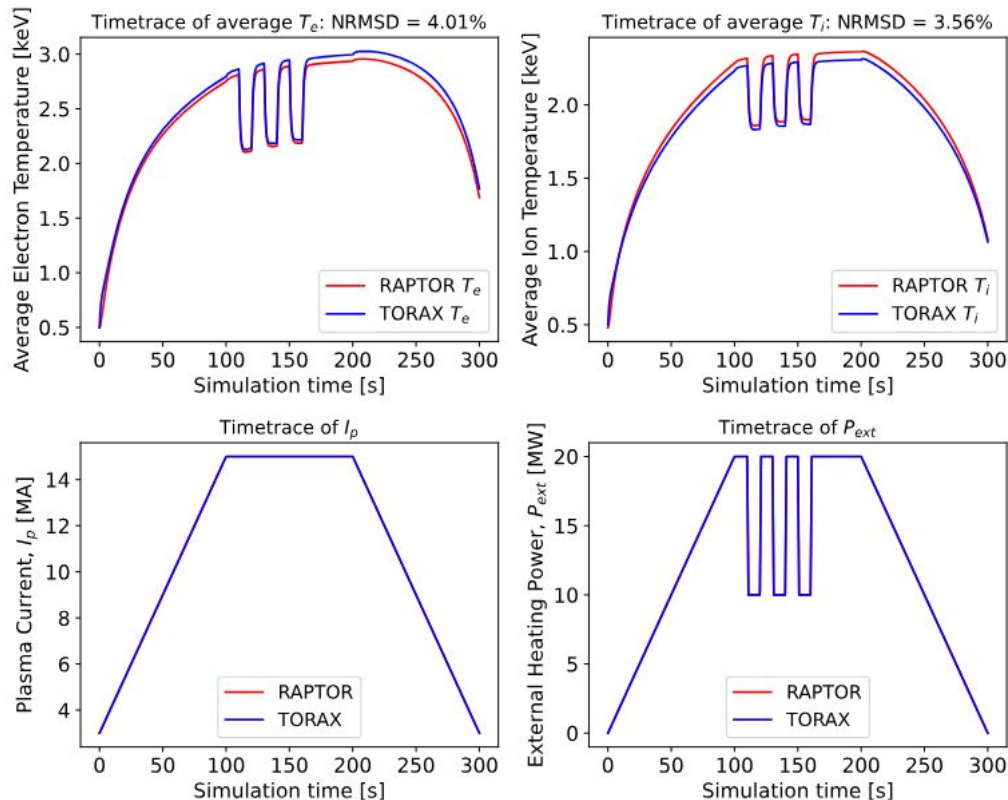
Cite as: arXiv:2406.06718 [physics.plasm-ph]

(or arXiv:2406.06718v2 [physics.plasm-ph] for this version)

<https://doi.org/10.48550/arXiv.2406.06718>



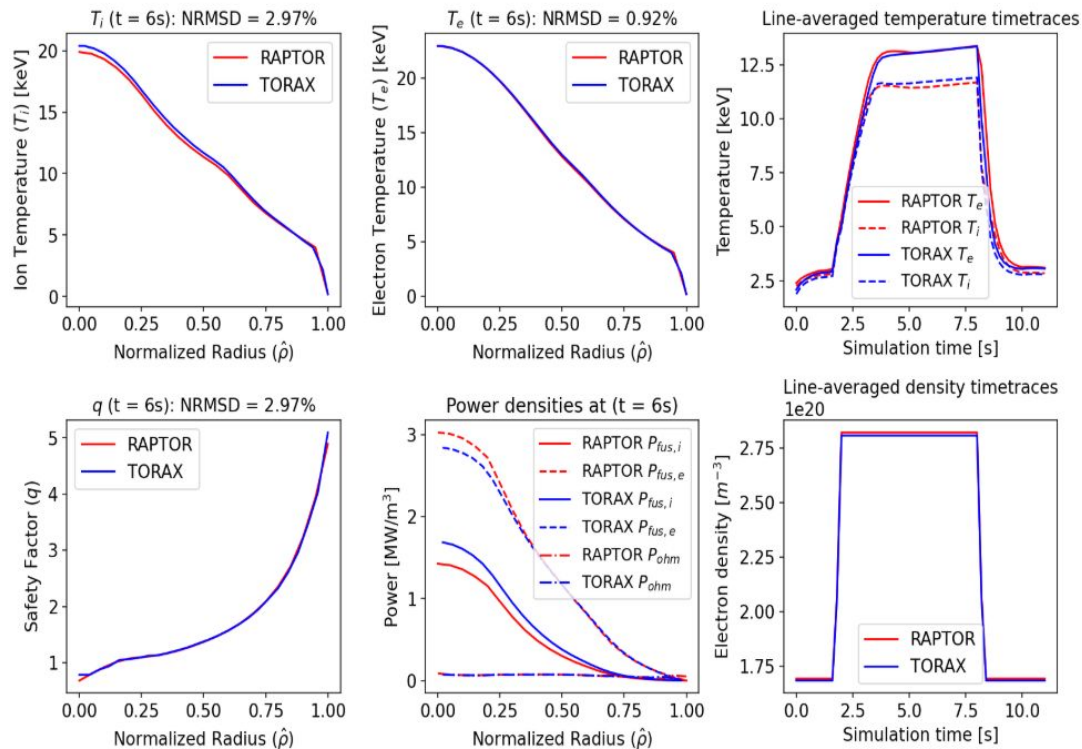
Verification: TORAX vs RAPTOR agreement for ITER-like cases



Modeling settings:

- ITER inspired params
- Nonlinear Newton-Raphson solver
- Heat transport + current diffusion
- 20MW heating: modulated
- Constant transport coefficients

Verification: TORAX vs RAPTOR agreement on SPARC H-mode scenario



Modeling settings:

- SPARC full-pulse scenario [A. Teplukhina CFS]
- Nonlinear Newton-Raphson solver
- Heat transport + current diffusion
- Sequence of magnetic equilibria
- 11MW heating power
- ML-surrogate turbulent transport model (QLKNN10D*)

$\Delta t = 0.2s$: RAPTOR walltime: ~70s, TORAX walltime: ~7s

Initial applications and collaborations

CFS MOSAIC Pulse Planning Workflow:

- GSPulse [Wai APS24]. Optimizes coil currents over full target pulse trajectory
- TORAX for internal plasma dynamics

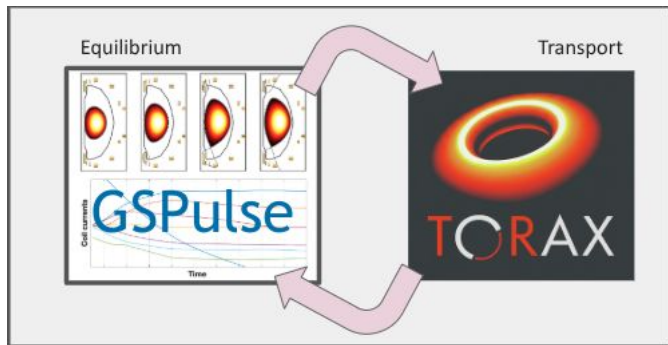


Image from Devon Battaglia

ITER Pulse Design Simulator w/CEA + Ignition Computing

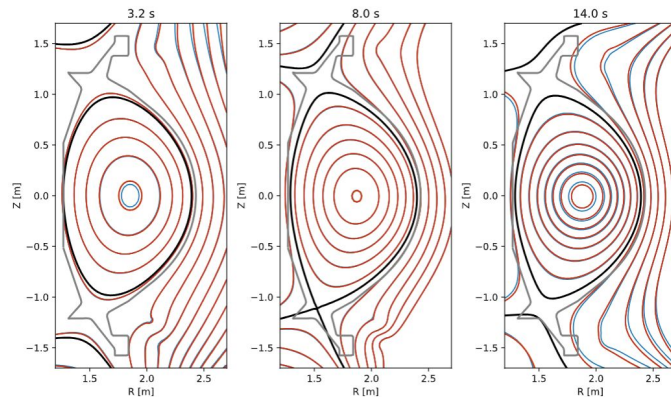
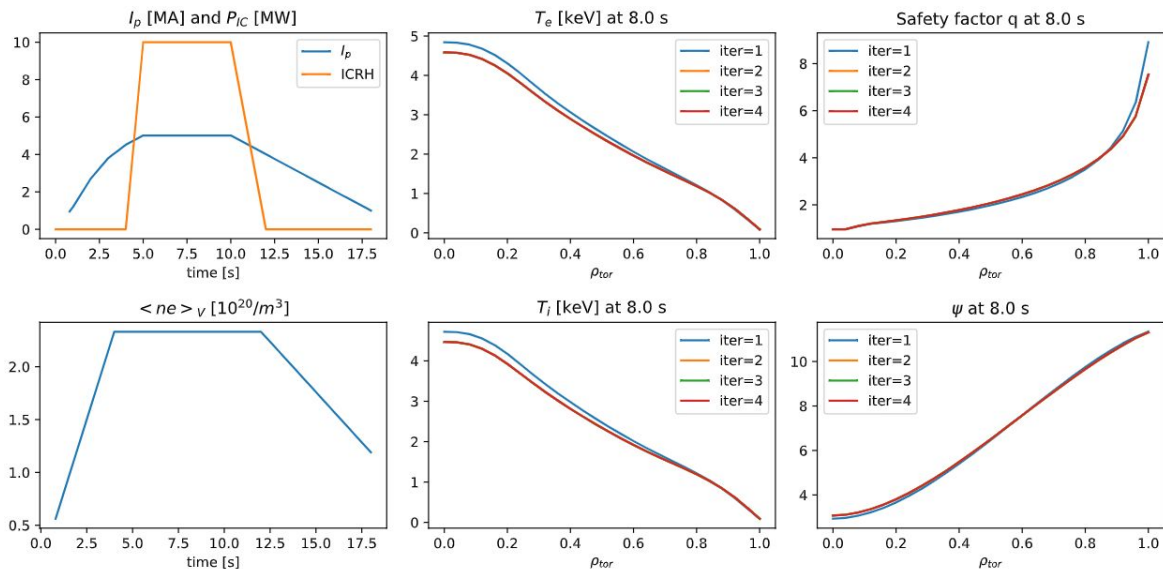
- Coupling to IMAS underway [Schneider, Bellouard, Sanders, van Vugt]

UKAEA

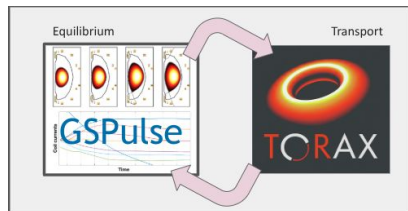
- Progress in preparing TORAX for STEP simulation and benchmarking with JINTRAC [T. Brown, L. Zanisi]

TORAX-GSPulse convergence for self-consistent transport and geometry within the SPARC Pulse Planning Workflow running in MOSAIC

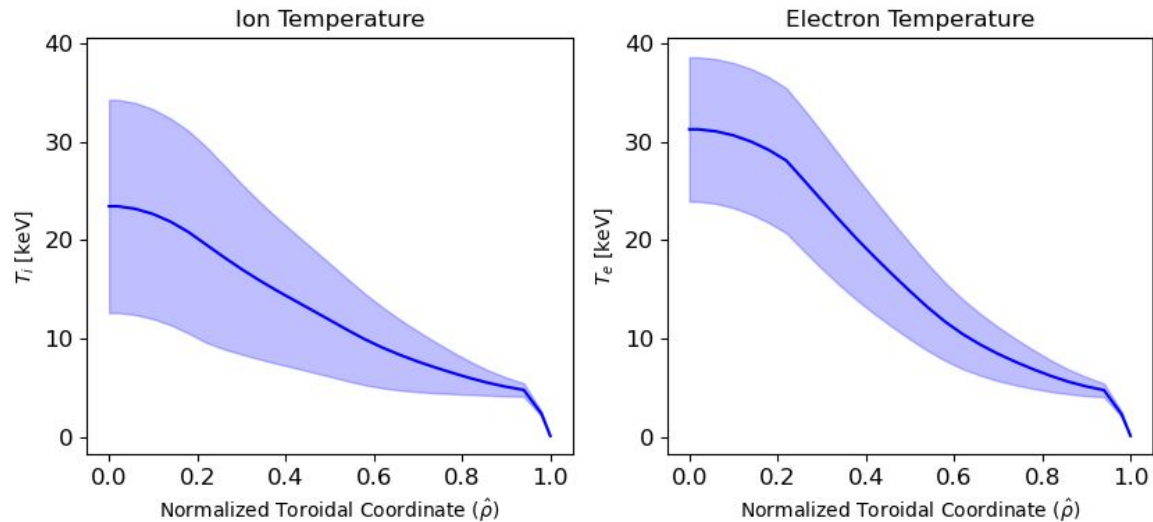
SPARC L-mode D pulse: 10 MW ICRH, single null



Anna Teplukhina [CFS]



Running TORAX at scale for Uncertainty Quantification



~ 80,000 simulations. ~30 mins on ~200 cores

ITER-like hybrid scenario
parameter variations

- $P_{\text{ped}} = [75 - 108] \text{ kPa}$
- $n_{\text{ped}} = [0.5 - 0.72] 10^{20} \text{ m}^{-3}$
- $Z_{\text{eff}} = [1.5-2.5]$
- $n_{\text{W}}/n_{\text{Ne}} = [5\text{e-}4 - 2\text{e-}3]$
- ECRH width = $[0.064-0.1] \rho_{\text{norm}}$
- Bootstrap multiplier = $[0.9-1.1]$
- $\text{chi}_{\text{ion,inner}} = [0.25 - 1] \text{ m}^2/\text{s}$

$$Q = 6.8 \pm 2.9$$

Roadmap: towards higher physics fidelity and pulse planning applications

Physics model developments

- Extended physics
 - Coupling to reduced edge models for heat-exhaust
 - Rotation + ExB shear
 - Multi-ion + impurity transport
- Engaging with wider community on improved ML-surrogates
 - Turbulence
 - Pedestal
 - Magnetic equilibrium
 - Heat/particle sources
 - Plasma edge + wall

Improved numerics+engineering

- Demonstrate "full-sim" differentiable capabilities with flexible API
- Demonstrate large-scale batch simulations on GPU for optimization, UQ, inverse-problem solving

Validation/calibration against data

- Open source datasets for integrated modelling validation
 - Engagement with wider community
- sim2real gap closure through hybrid model + data-driven surrogates



<https://arxiv.org/abs/2406.06718>
<https://github.com/google-deepmind/torax>
<https://torax.readthedocs.io>



Thank you.

Google DeepMind

TORAX team



Sebastian Bodenstein



Jonathan Citrin



Anushan Fernando



Philippe Hamel



Tamara Norman