Realtime capable quasilinear gyrokinetic modelling using neural networks

K.L. van de Plassche[1,2], J. Citrin[2], C. Bourdelle[3], V.I. Dagnelie[2], A. Ho[2]

[1] *University of Technology Eindhoven, PO Box 513, 5600 MB Eindhoven, The Netherlands*

[2] *DIFFER, PO Box 6336, 5600 HH Eindhoven, The Netherlands*

[3] *CEA, IRFM, F-13108 Saint-Paul-lez-Durance, France.*

Nonlinear gyrokinetic turbulence simulations are increasingly validated by experiments. Yet, they are too computationally expensive for applications such as offline scenario optimization and real-time supervision and control. Reduced linear modules provide speedup of up to 7 orders of magnitude, at the cost of reduced complexity of the underlying physical model. For example, 1s of JET evolution demands ~2×10^3 turbulent flux calculations, resulting in ~10 hour simulation time on 10 cores for a second of plasma evolution for the QuaLiKiz [1, 2] quasilinear transport code. However, we need even more speedup to reach the real-time regime of $10^{-3}s$ per flux calculation. In this study, we propose to use neural networks to emulate QuaLiKiz. This final step gives another six orders of magnitude speedup, which bridges the gap to realtime modelling. We base our work on the successful proof of concept by J. Citrin *et al.*[3], in which a multilayer perceptron neural network was able to reproduce QuaLiKiz heat fluxes as a function of ion temperature gradient, ion-electron temperature ratio, safety factor and magnetic shear. In this work, we expand this 4D input space to 7D by adding electron temperature gradient, density gradient and minor radius. A dataset with these dimensions plus two additional dimensions, collisionally and Zeffx, has been made. The input space can be extended further with a new linear quench rule by V. Dagnelie to include rotation as 10th dimension [4]. Heat fluxes and particle diffusion and pinch for both ions and electrons for ITG, ETG and TEM regime are the outputs contained in the dataset.

Training Method

The first step of Neural Network training is acquiring a training set. We have generated a 9D hyperrectangle of $3 \cdot 10^8$ flux evaluations within experimentally relevant ranges using the QuaLiKiz code. The ranges can be found in table 1. The dataset is available for visual inspection online [5]. It was generated using 1.3 MCPh using the Edison supercomputer of the Berkeley National Energy Research Scientific Computing Center.

variable	# points	min	max
$k_\theta \rho_s$	18	0.1	36
$\frac{R}{L_{T_i}}$	12	0	14
$\frac{R}{L_{T_e}}$	12	0	14
$\frac{R}{L_n}$	12	-5	6
q_x	10	0.66	15
s	10	-1	5
ε	8	0.03	0.33
$\frac{T_i}{T_e}$	7	0.25	2.5
ν^*	6	1×10^{-5}	1
Z_{eff}	5	1	3
Total	3×10^8	≈ 1.3MCPUh	

Table 1: *Input ranges of dataset generated from QuaLiKiz data.*

After generation, the dataset need to be filtered before a network is trained. Points where we feel QuaLiKiz is not yet sufficiently validated, as well as points with a too high flux are filtered out. Then, the filtered dataset is split in three sets: train, test and validation. The training set is used to train the weights and biases of the neural network itself, while the validation set is used to tune the hyper-parameters (for example, learning rate, early stop condition). Finally, the test set is used to determine the final goodness of the fit.

The neural network training was performed using the TensorFlow framework [6]. TensorFlow allows one to define a data flow graph, in which nodes are mathematical operations and the edges are tensors, containing the inputs and outputs for these operations. The data graph is defined in Python, while the actual operations are performed in a high-performance language, usually FORTRAN or C. Many operations are also generalized to use the CUDA or OpenCL framework, allowing code to run on GPUs. This level of abstraction allows for rapid prototyping and analysis in Python, while still being able to do fast neural network training. As such, we have set up a Python framework to quickly test different kinds of neural networks and training strategies.

Our current neural networks use three hidden layers of 30 nodes. Although two layers are generally sufficient, we found higher convergence rates with three layers. A sigmoid, the tanh, was used as activation function. The network are trained using the L-BFGS algorithm, using a loss function that includes the mean square error and the L2-loss. The L2-loss term prevents overfitting.

Threshold Mismatch

An important effect we found to be important for physically consistent integrated modelling during our training is *threshold mismatch*[7]. When training directly on the ion and electron heatflux, one generally tries to minimize mean square error. Although this results in fits that generalize the training set quite well, it does not capture important small features, as it is an average quantity. A feature that is particularly important is the threshold, which is when turbulent transport starts having a non-negligible strength. Turbulent transport shows threshold behaviour in many dimensions, here we focus on the temperature threshold of ITG turbulence. If, for example, the threshold for the ions lies higher than for the electrons, the electron temperature will 'run away', resulting in a higher electron temperature. This artificially changes the system that is simulated, resulting in a non-physical system. In our work, we try to solve this by fitting on combined ion and electron fluxes. For example, fits on $^{q_i}/_{q_e}$ and $q_i + q_e$ have successfully been made for total flux and separate ITG, TEM and ETG fluxes. However, more validation has to be done to determine the best method to remove the threshold mismatch.

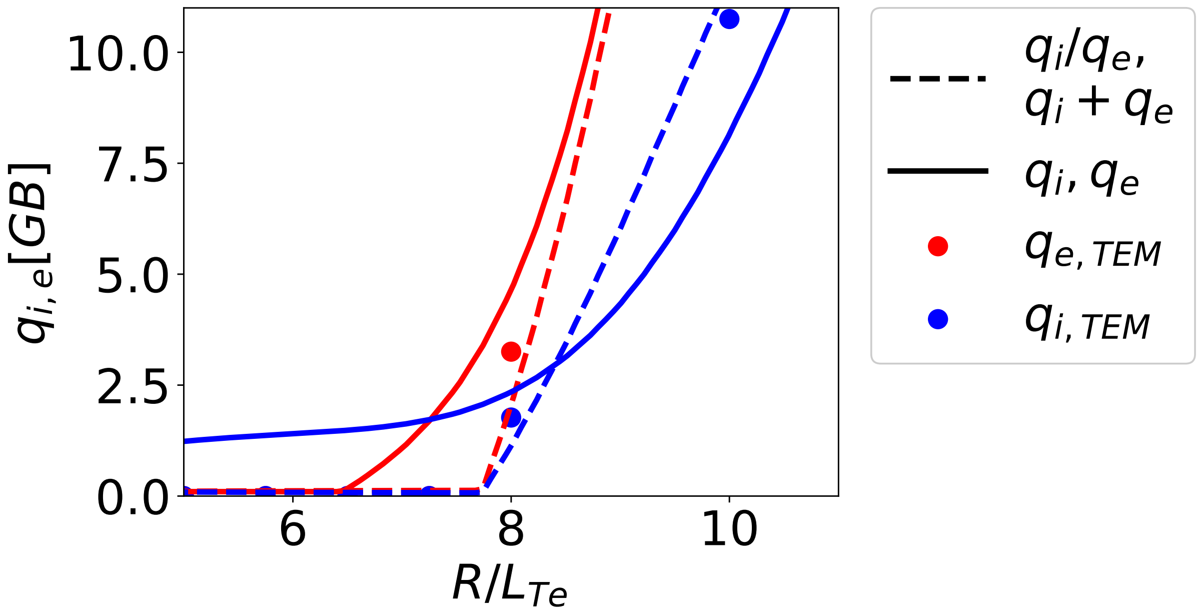

Figure 1: *An example of threshold mismatch. One can see the two different styles of neural network fits. Dots are QuaLiKiz output, and the solid lines represent a fit directly on flux. The dashed line is a fit on* $^{q_i}/_{q_e}$ *and* $q_i + q_e$*. Ions in blue and electrons in yet. Note the exact match of threshold with the dashed lines.*

Goodness of Neural Networks

Finding a good metric for comparing neural network performance is hard. The usual practice of stating the root mean square error as catch-all for neural network performance is not sufficient to cover the relevant physics. To get a better view of the performance we instead propose to use kernel density estimating (KDE). KDE is a non-parametric way to estimate population density, and is generalization of the histogram. Using a KDE gives a better view of where the error comes from, and from which part of the population.

Conclusion and Outlook

In this paper we have presented a large-scale database of $3 \cdot 10^8$ ITG, TEM, ETG heat and particle fluxes using the QuaLiKiz code. Initial 7D neural network fits look promising, but a more rigid approach of validation is needed. We noted that for out purposes, a simple RMS error is not enough.

In future work, the neural networks will be extended to 9D. This will be done while keeping physical constraints in mind. The networks will also be extended to 10D by using a ExB shear suppression model. Finally, the trained networks will be validated in the real-time capable tokamak simulator suite RAPTOR [7, 8, 9]

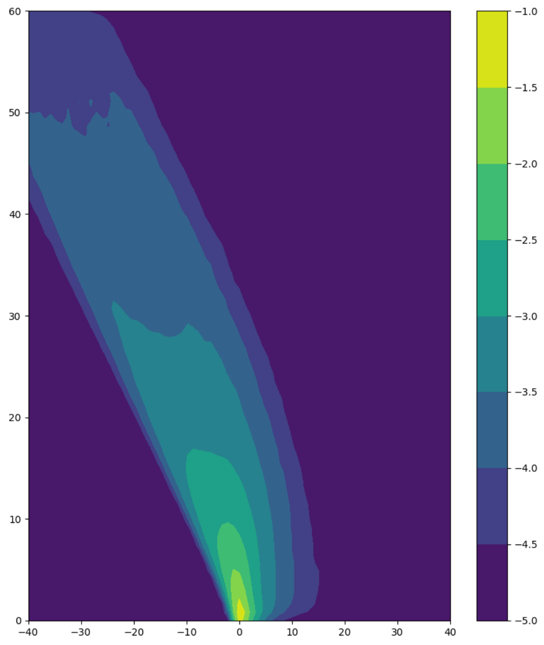

Figure 2: *Kernel density estimate of residual. Colours are plotted on a logarithmic axis, and represent the population density. The x-axis shows the residual ($y_{real} - y_{NN}$), and the y-axis shows total flux. Here we can clearly see that most of the population has both a low flux and a low residual. The distribution of errors is also non-symmetric, so the RMS error does not describe the underlying distribution fully.*

References

[1] C. Bourdelle *et al.* PPCF **58** 014036 (2016)
[2] C. Bourdelle *et al.*, this conference (EPS Belfast 2017, P4.167)
[3] J. Citrin *et al.* Nucl. Fusion **55** 092001 (2015)
[4] V.I. Dagnelie *et al.*, this conference (EPS Belfast 2017, P5.183)
[5] K.L. van de Plassche `http://dataslicer.qualikiz.com`
[6] M. Abadi *et al.* Tensorflow `http://tensorflow.org`
[7] J. Citrin *et al.*, this conference (EPS Belfast 2017, P5.160)
[8] F. Felici and O. Sauter, Plasma Physics and Controlled Fusion **54**, 2 (2012)
[9] A. Ho *et al.*, this conference (EPS Belfast 2017, P5.173)