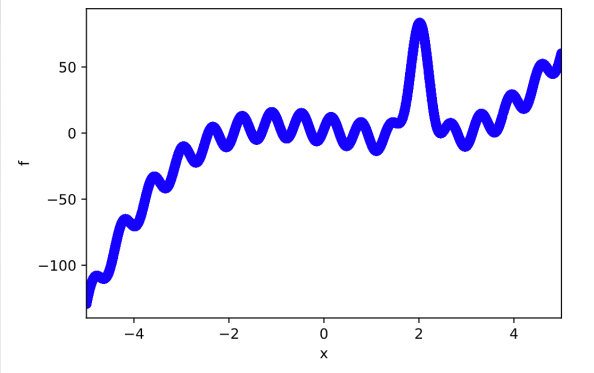
Revision!

1)

DNNs with Tensorflow/Keras



`NN1d.ipynb`

let's return to this example
and try a few things

Exercises

1) edit the code to create an arbitrary archictecture from a list of layer depths. How does the fit depend on the architecture?

2) change the activation function used in the nodes to tanh, what changes in the output? Do you need to change the normalisation?

3) can we change the loss function to prefer certain regions or features?

4) vary the number of input data points. Is there an optimal number?

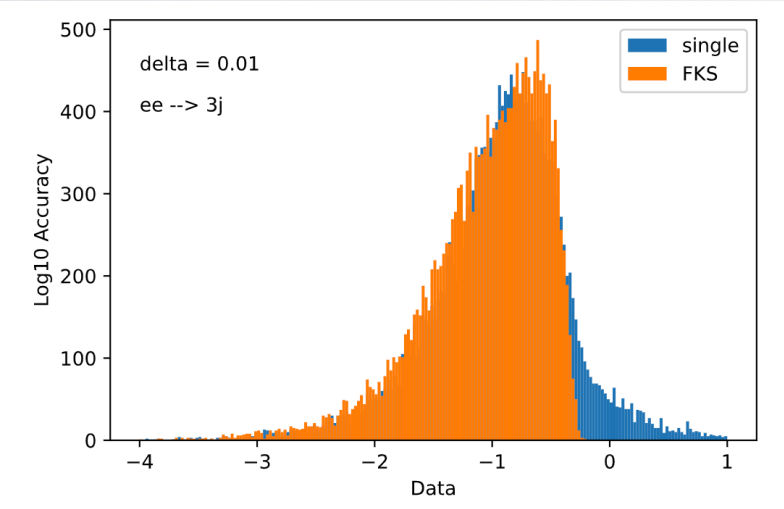
5) Using the template for early stopping try to optimise the number of epochs needed by the network

https://keras.io/api/callbacks/early_stopping/

6) Split the input and output data set according to random shuffle rather than a slice, do you see any affect?

2)

Amplitude Neural Networks



`NNNJet_FKS.ipynb`

(results not as expected….try to see what went wrong)

Exercises

1. Checking the cross section predicitions:
   A. Order the inference test data (i.e. NJ_treevals_test from the data set momenta_test) by the size to see which points give the largest contributions to the cross section.
   B. Compare these points (say the largest 10 values) against the inferred values from the single and FKS ensemble networks.
   C. Compute the cross section from all three approaches and see which one gave the best approximation
2. Higher multiplicity example:
   A. Create a new NJet order file for e+e- -> 5j (e.g. 11 -11 -> 1 -1 21 21 21) and use it to create the contract file for the NJet link (njet.py -o NJ_contract_ee5j_tree.lh).
   B. Run the code again to see how well the network trains - do we need more training data than for lower multiplicities?