Citation for published version:
Appella, S, Arridge, S, Budd, C, Deveney, T & Kreusser, LM 2024 'Equidistribution-based training of Free Knot Splines and ReLU Neural Networks'.

Publication date:
2024

Link to publication

University of Bath

Alternative formats
If you require this document in an alternative format, please contact:
openaccess@bath.ac.uk

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 23. Apr. 2025

Equidistribution-based training of Free Knot Splines and ReLU Neural Networks

Simone Appella, Simon Arridge, Chris Budd, Teo Deveney, Lisa Maria Kreusser

July 3, 2024

arXiv:2407.02153v1 [cs.LG] 2 Jul 2024

Abstract

We consider the problem of one-dimensional function approximation using shallow neural networks (NN) with a rectified linear unit (ReLU) activation function and compare their training with traditional methods such as univariate Free Knot Splines (FKS). ReLU NNs and FKS span the same function space, and thus have the same theoretical expressivity. In the case of ReLU NNs, we show that their ill-conditioning degrades rapidly as the width of the network increases. This often leads to significantly poorer approximation in contrast to the FKS representation, which remains well-conditioned as the number of knots increases. We leverage the theory of optimal piecewise linear interpolants to improve the training procedure for a ReLU NN. Using the equidistribution principle, we propose a two-level procedure for training the FKS by first solving the nonlinear problem of finding the optimal knot locations of the interpolating FKS. Determining the optimal knots then acts as a good starting point for training the weights of the FKS. The training of the FKS gives insights into how we can train a ReLU NN effectively to give an equally accurate approximation. More precisely, we combine the training of the ReLU NN with an equidistribution based loss to find the breakpoints of the ReLU functions, combined with preconditioning the ReLU NN approximation (to take an FKS form) to find the scalings of the ReLU functions, leads to a well-conditioned and reliable method of finding an accurate ReLU NN approximation to a target function. We test this method on a series or regular, singular, and rapidly varying target functions and obtain good results realising the expressivity of the network in this case.

1 Introduction

Function approximation is one of the major applications of neural nets (NNs), either directly or indirectly, for such applications as function generation, system identification, and the solution of physical problems such as differential equations. For an input vector $\mathbf{x}$ to a NN, the output of the NN can be thought of as a function evaluation $y(\mathbf{x})$ where the corresponding function y is defined on a suitable domain and has the same degree of smoothness as the NN activation functions. The NN output $y(\mathbf{x})$ is trained to approximate some desired target function value $u(\mathbf{x})$. A NN can be regarded as a function generator for instance in the context of neural operators [17], and can be used as means for solving both ordinary and partial differential equations [15], [8].

Approximation theory has long considered the problem of efficiently approximating a univariate target function $u(x), x \in [0,1]$[7, 19]. Examples include the use of basis splines, such as interpolation methods, or the Galerkin methods used in the Finite Element Method. Such classical function approximations are usually *linear* in their coefficients, as the weights $\mathbf{w}$ of a given set of basis functions are optimised. As such the function approximation lies in a linear function space. Typically the basis functions themselves are piece-wise polynomials, defined over a *fixed* set of *knots* $\mathbf{k}$ where the form of the local polynomial approximation changes. These methods have provable accuracy and there exist efficient, and well-conditioned, algorithms for calculating the optimal weights that reliably converge to a unique solution, often either solving linear systems of equations (directly or by iteration), or by a simple quadratic minimisation problem. However, such linear methods often lack accuracy, and may require a large number of coefficients to achieve a good level of approximation for a function with complex, or singular behaviour. In one-dimension significantly greater accuracy, with far fewer numbers of coefficients, can often be achieved by using a nonlinear approximation method [4] such as a Free Knot Spline (FKS), adaptive methods [14]. In contrast to standard spline approximations, in such nonlinear approximations both the weights $\mathbf{w}$ and the knot locations $\mathbf{k}$ are optimised. Even though such methods are nonlinear in the coefficients, effective a-priori and

a-posteriori error estimates and interpolation error estimates have been established in certain cases (such as the FKS), linked to the regularity of the meshes over which the approximations are defined [14]. Neural nets are also nonlinear approximators, with many features in common with a FKS or an adaptive method, and we will make extensive use of this correspondence in this paper.

A key advantage of using a FKS over a standard spline approximation, is that for a fixed number N of knots $\mathbf{k}$ a higher accuracy can be achieved by exploiting the extra degrees of freedom given by moving the knots [6], which is of importance when approximating functions with rapid variation and/or singularities. A special case are linear interpolating FKS (IFKS) where the weights are evaluations of the target function $u(k_i)$ at the knot locations, and as such the optimisation reduces to finding knot locations $\mathbf{k}$ only. Convergence results in various norms for the best interpolating FKS $\Pi_1 u$ to a general target function u as $N \to \infty$ are given in [14] in the context of adaptive mesh generation. In particular, the approximation error between a general target function u and a linear interpolating FKS $\Pi_1 u$ with an optimal choice of N knots satisfies $\|\Pi_1 u - u\|_2 = \mathcal{O}(N^{-2})$. Once the optimal knots $\mathbf{k}$ of the IFKS are known, the problem of finding the optimal weights of the FKS is then a well conditioned (near linear) problem.

When the knots $\mathbf{k}$ are not chosen in an optimal way, such as taking uniform knots for instance, the approximation error to the target function u depends on the smoothness of u. For a smooth function u the same order of convergence is obtained for an interpolating FKS with regular knots or optimal knots. For a rapidly varying function good convergence is only seen when the know spacing is smaller than the smallest length scale of the target function. For a singular function u, a higher asymptotic rate of convergence is obtained as $N \to \infty$ for optimally adapted knots which results in a reduced level of expressivity for the interpolating FKS $\Pi_1 u$ with uniform knots.

In contrast to a FKS, univariate NNs approximate a general target function $u(x)$ through a combination of linear operations and nonlinear/semilinear activation functions which results in function approximations nonlinear in their coefficients. NNs are thus harder to analyse than linear approximation methods and their training can be challenging due to the non-convexity of the associated loss functions, and focusing on the easier case of shallow NNs as opposed to deep NNs can give valuable insights, for instance in terms their landscape [18], their mean-field analysis [21] and their gradient dynamics [22]. A key advantage of NNs is that they are nonlinear approximators with provable expressivity, see [10] and [5] for instance for reviews on the theoretical expressivity of deep NNs. By the Universal approximation theorem [12] any continuous function on a compact set can be approximated by a certain neural network, with a continuous activation function, to any accuracy. In principle a NN approximation also has the attractive feature that it can be 'self-adaptive' [8], so that it may be able to account for singularities and rapid variations in the target function.

A common example for an activation function is the rectified linear unit (ReLU) activation function, defined as $\mathrm{ReLU}(x) = \max(x, 0) = (x)_+$, which is often used in practice due to its efficiency and simplicity. In addition, a key feature of ReLU NNs is that they result in a globally continuous and piecewise linear input-output relation. They are thus formally equivalent to a piecewise linear free knot spline (FKS), with the ReLU NN and FKS describing the same set of functions. For a ReLU NN, the approximation error can be explicitly linked to the depth L and width W of the network, with exponential rates of (error) convergence with increasing depth L [23]. However, even with theoretical expressivity guarantees, this does not guarantee that the optimal solution is found by the training algorithm with the standard L_2^2 loss if the initialisation is distant from the optimal approximation. Examples of the training algorithm failing on even quite regular problems will be given in this paper.

The complementary advantages of FKS and ReLU NNs, and their formal equivalence, motivates us to investigate the relationship between shallow ReLU NNs and splines, and particular the link between the knot points of the FKS and the breakpoints (where the derivative changes discontinuously) of the shallow ReLU NN. Using the spline lens, shallow univariate ReLU NNs have been investigated in terms of their initialization, loss surface, Hessian and gradient flow dynamics [20]. A ReLU NN representation for continuous piecewise linear functions has been studied in [11], with a focus on continuous piecewise linear functions from the finite element method with at least two spatial dimensions. A theoretical study on the expressive power of NNs with a ReLU activation function in comparison to linear spline-type methods is provided in [9]. The connection between NNs and splines have also been considered in other contexts, for instance when learning activation functions in deep spline NNs [1].

1.1 Contributions

In this paper, we analyse the problem of function approximation from the perspective of training both an FKS and a shallow ReLU NN. We show the latter problem is poorly conditioned as the network width W increases, and as a result the training process can lead to a poor approximation of the target function. We propose a novel two-level approach to the nonlinear problem of training (using Adam) the univariate FKS approximation (which involves finding the weights $\mathbf{w}$ and (free) knots $\mathbf{k}$). This problem is firstly tackled by constructing an effective loss function based on the equidistribution principle for the knot locations. Secondly we then solve the provably well conditioned problem of finding the weights. We then show that this approach can be modified to train a shallow ReLU NN. Firstly by finding the breakpoints of the ReLU NN and then the scaling factors of the ReLU functions. For the ReLU NN we show that this problem is ill conditioned for large W, but can be made well conditioned by a linear preconditioning of the ReLU NN (effectively transforming it into an FKS).

Using a variety of (regular, singular and rapidly varying) target functions, we demonstrate numerically that our two-stage training procedure with the novel loss function both the FKS and the shallow ReLU NN can be trained reliably in one spatial dimension to give an highly accurate approximation. In particular if the width of either network is N then we see $\mathcal{O}(1/N^4)$ convergence of the mean square difference between the approximation and the target function This contrasts with the worse performance of applying a standard training procedure to (for example) the ReLU NN, in which we do not see convergence to an accurate solution for any N, or see slow convergence for moderate to large values of N due to ill-conditioning.

1.2 Outline

The paper is structured as follows: in Section 2 we describe deep and shallow ReLU networks, as well as free knot splines (FKS) as nonlinear function approximators. In Section 3, we show the formal equivalence of shallow ReLU NN and the FKS representations and investigate the conditioning of the problem of training the weights and scaling parameters of the respective networks. Section 4 defines the loss functions we consider based on the mean-squared error and an equidistribution-based approach to determine the knots, as well as their training. Numerical results for the one dimensional approximation for a range of target functions are shown in Section 5 where we investigate the impact of the choice of loss function, the quality of the function approximation, the evolution of the knot locations as well as the conditioning and the convergence of the loss training. Finally, in Section 6 we will draw our conclusions based on the numerical results, and suggest areas of further investigation.

2 ReLU NN and free knot splines (FKS) as nonlinear function approximators

In this section, we define one-dimensional ReLU Neural Networks (NN) and free knot splines (FKS).

2.1 Deep and Shallow ReLU Neural Networks (NN)

We start by defining a standard feed forward neural network with a one-dimensional input $x_0 \in \mathbb{R}$ and one-dimensional output $y_{NN}(x_0, \boldsymbol{\theta})$ for some vector of parameters $\boldsymbol{\theta}$, described further below. We assume that this network has a width W and a depth L and uses a rectified linear unit (ReLU) $\sigma(x) = (x)_+$ as activation function which acts element wise on the input vector. Defining the network input by $x_0 \in \mathbb{R}$, the first hidden layer is defined by

$$\boldsymbol{x}_1 = \sigma\left(\boldsymbol{A_0} x_0 + \boldsymbol{b}_0\right) \in \mathbb{R}^W$$

for $\boldsymbol{A}_0, \boldsymbol{b}_0 \in \mathbb{R}^W$ and the NN with L hidden layers is then iteratively defined by

$$\boldsymbol{x}_{k+1} = \sigma\left(A_k \boldsymbol{x}_k + \boldsymbol{b}_k\right) \in \mathbb{R}^W \tag{1}$$

for $k \in \{1, \ldots, L-1\}$, with $A_k \in \mathbb{R}^{W \times W}$ and $\boldsymbol{b}_k \in \mathbb{R}^W$. The NN output is given by

$$y(x, \boldsymbol{\theta}) = \boldsymbol{A}_L \boldsymbol{x}_L + b_L \in \mathbb{R}, \tag{2}$$

with $\boldsymbol{A}_L \in \mathbb{R}^{1 \times W}$ and $b_L \in \mathbb{R}$. In the simplest case we can consider a *shallow* ReLU network with width W and depth $L = 1$, otherwise the network is *deep*. The case $L = 1$ is closest to standard

approximation, but the real power of the NN comes from the expressivity seen when $L > 1$ although as we shall see such networks are harder to train.

In this representation we define the set of parameters $\boldsymbol{\theta} = \{A_k, \boldsymbol{b}_k \colon \ k = 0, \ldots, L\}$. For a given range of parameters $\boldsymbol{\theta}$ we define $\Gamma_{W,L}$ to be the set of all possible such functions $y(x)$. Observe that $\Gamma_{W,L}$ is *not* a linear vector space and that A_k and $\boldsymbol{b}_k$ are not sparse in general.

We will apply this NN to approximate a function $u \colon [0,1] \to \mathbb{R}$ by finding an appropriate choice of $\boldsymbol{\theta}$ where we will not assume any regularity properties of u at present. We assume that there exists a batch of training data $\{y_{x_i}\}_{i=0}^{s-1} \subset \mathbb{R}^s$ at training points $\{x_i\}_{i=0}^{s-1} \subset [0,1]$ of size s. Our aim is to use the NN so that the function $y(x)$ approximates the target $u(x)$ in the interval $[0,1]$. We use the standard means of training a NN to do this, including the use of a squared-error loss function and an optimiser such as Adam [16]. However, as we will see, crucial to the effectiveness of the training procedure is the use of a good initialisation to start the approximation, an effective loss function, and a careful preconditioning of the problem. We shall demonstrate this using the examples in this paper, and will also show how such a good initialisation, loss function and preconditioning can be constructed.

2.2 Linear spline approximations

Let $N \in \mathbb{N}$ be given and let

$$0 = k_0 < k_1 < \ldots k_i < \ldots < k_{N-1} = 1$$

be a set of N *knots* in the interval $[0,1]$. For $i = 0, \ldots, N-1$, we define a *linear spline* $\phi_i(x)$ to be the unique piecewise linear function so that

$$\phi_i(k_i) = 1, \quad \text{and} \quad \phi_i(k_j) = 0 \quad \text{if} \quad i \neq j.$$

Each spline ϕ_i is weighted by some weight w_i, and we denote by

$$\boldsymbol{\psi}_{FKS} = \{(w_i, k_j) \colon i = 0, \ldots, N-1, \ j = 1, \ldots, N-2\}$$

the set of parameter values. Note that we exclude k_0 and k_{N-1} in the definition of $\boldsymbol{\psi}_{FKS}$ as they are set as 0 and 1, respectively.

Definition 1 (Free knot splines (FKS)) *A* free knot spline (FKS) *is a piecewise linear approximation to the function $u(x)$ defined over the set of linear splines and is given by*

$$y(x, \boldsymbol{\psi}_{FKS}) = \sum_{i=0}^{N-1} w_i \phi_i(x), \tag{3}$$

where we assume that we are free to choose both the knots k_i, $i = 1, \ldots, N-2$, in the linear splines ϕ_i and the weights $w_i, i = 0 \ldots N-1$.

We denote by $\Sigma_{N,1}$ the set of all piecewise linear functions of the form (3). Observe that the FKS has $2N-2$ degrees of freedom ($N-2$ for the knots and N for the weights). Like the NN space $\Gamma_{W,L}$, the approximation space $\Sigma_{N,1}$ is nonlinear. Similarly to the NN we can train a FKS to approximate a function. This is a nonlinear, non-convex problem to find the full set of weights $\mathbf{w}_i$ and knots $\mathbf{k}_i$.

In the usual case of piecewise linear approximation (used for example in the Finite Element method), the values of the knots k_i are *fixed* and we refer to such approximations as either *fixed knot piecewise linear approximations* or *regular approximations*. The best function approximation is given by the conditioned problem of finding the optimal choice of *weights* w_i for $i = 0, \ldots, N-1$ and we denote by $\boldsymbol{\psi}_{PWL} = \{w_i \colon i = 0, \ldots, N-1\}$ the associated parameter values.

Definition 2 (Fixed knot piecewise linear interpolant (PWL)) *Such a* fixed knot piecewise linear approximation *is a piecewise linear approximation to the function $u(x)$ defined over the set of linear splines and is given by*

$$y(x, \boldsymbol{\psi}_{PWL}) = \sum_{i=0}^{N-1} w_i \phi_i(x) \tag{4}$$

where we assume that we are free to choose the coefficients w_i, but the knots k_i, $i = 1, \ldots, N-2$, in the linear splines ϕ_i are assumed to be fixed.

Note that (4) is a linear function of the coefficients w_i and the set of all such possible functions is a linear vector space. The values of w_i can be found simply by minimising the least squares loss. Equivalently, a linear matrix problem of the form $\boldsymbol{P}\boldsymbol{w} = \boldsymbol{q}$ for the coefficients $\boldsymbol{w} = (w_i)_{i=0}^{N-1} \in \mathbb{R}^N$ can be solved, where the (well conditioned) sparse matrix $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ and vector $\boldsymbol{q} \in \mathbb{R}^n$ are given. Whilst this is a simple process and widely used, for example in the Finite Element Method, such approximations lack expressivity, especially if the target function $u(x)$ is singular, or has natural length scales which are smaller than the smallest separation between the knots. The additional degrees of freedom given by choosing the knots k_i in the FKS will (as we shall see) lead to a very significant increase in the accuracy of the approximation.

A distinguished subset of the set of free knot splines are the *interpolating free knot splines* where the set of knots k_i for $i = 1, \ldots, N-2$ can be freely chosen, but the weights are given by $w_i = u(k_i)$ We denote the associated parameter values by $\boldsymbol{\psi}_{IFKS} = \{k_i \colon i = 1, \ldots, N-2\}$.

Definition 3 (Interpolating free knot splines (FKS)) *The* interpolating free knot splines (FKS) $y(x, \boldsymbol{\psi}_{IFKS}) = \Pi_1 u(x)$ *for the target function* $u(x)$ *is given by*

$$y(x, \boldsymbol{\psi}_{IFKS}) \equiv \Pi_1 u(x) = \sum_{i=0}^{N-1} u(k_i)\phi_i(x), \tag{5}$$

where we assume that we are free to choose the knots $k_i,\ i = 1, \ldots, N-2$*, in the linear splines* ϕ_i*.*

The interpolating FKS is easier to construct than the best FKS approximation for $u(x)$ as the coefficients w_i of the FKS are given directly by $w_i = u(k_i)$. It is thus not as expressive as the best FKS, but has (as we will see) the same rate of convergence as $N \to \infty$ for classes of singular function, and it is a much better approximation in general than a fixed knot piecewise linear approximation. Observe that finding the interpolating FKS still requires solving a nonlinear problem for the knots. We will make extensive use of this function as a way of both assessing the accuracy of a more general approximation to u and of guiding the approximation procedure. We shall demonstrate that $\Pi_1 y$ is a very good function to take as the initial start of the optimisation procedure for finding either the best ReLU NN approximation or the best FKS approximation to u. Indeed, if the *optimal* set K of knot points is constructed for $\Pi_1 u$ then this appears from calculations to be very close to the optimal set of knot points for the FKS, and equivalently the breakpoints of the ReLU NN representations.

3 Relations between the ReLU NN and the FKS representations and the associated conditioniing of the training problem

Any ReLU NN gives a function $y(x)$ which is piecewise linear, and is smooth (linear) everywhere apart from at a set of breakpoints where it changes gradient and has undefined curvature. These points are equivalent to the knot points k_i of the FKS if they lie in the range of values of $x \in [0, 1]$ over which the target function $u(x)$ is defined. In this sense the ReLU NN and the FKS are formally equivalent. However the map between $\boldsymbol{\theta}$ and $\{k_i \colon i = 0, \ldots, N-1\}$ is complex and is related to the solution of a system of algebraic equations [5]. More precisely, the authors show that a NN of width W and depth L is formally equivalent to a FKS of size N (where N is the number of breakpoints) for $N < (W+1)^L$, and hence they can have the same expressivity, even though this is rarely seen in practice. Due to the linear dependencies between the linear pieces, the NN space $\Gamma_{W,L}$ is much smaller than the FKS space $\Sigma_{N,1}$ with N satisfying $N < (W+1)^L$.

3.1 Equivalence of shallow ReLU NN and the FKS representations

Both a ReLU NN and an FKS are piecewise linear functions of x and hence we can use the methodology of training one to inform the training of the other. To do this we now show how the coefficients of a shallow ReLU and FKS are related. For a shallow network with $L = 1$ and $N = W$, we have breakpoints in the ReLU representation, which are equivalent to knots in the FKS at

$$k_i = -\frac{(\boldsymbol{b}^0)_i}{(\boldsymbol{A}_0)_i}, \quad i = 0, \ldots, W-1 \quad \text{if} \quad k_i \in [0, 1], \tag{6}$$

where $(\boldsymbol{b}^0)_i$ and $(\boldsymbol{A}_0)_i$ denote the ith components of vectors $\boldsymbol{b}_0$ and $\boldsymbol{A}_0$. For any ordered set of knots k_i, with $0 = k_0 < k_1 < \ldots < k_{N-2} < k_{N-1} = 1$ we can represent the piece-wise linear basis functions $\phi_i(x)$ of the FKS using the formula presented in [5] so that

$$\phi_i(x) = \alpha_i ReLU(x - k_{i-1}) - \beta_i ReLU(x - k_i) + \gamma_i ReLU(x - k_{i+1}) \tag{7}$$

for $i = 1, \ldots, N-2$, where

$$\alpha_i = \frac{1}{k_i - k_{i-1}}, \quad \beta_i = \frac{k_{i+1} - k_{i-1}}{(k_{i+1} - k_i)(k_i - k_{i-1})}, \quad \gamma_i = \frac{1}{k_{i+1} - k_i}. \tag{8}$$

Similarly

$$\phi_0(x) = ReLU(k_1 - x)/k_1, \quad \phi_{N-1}(x) = ReLU(x - k_{N-2})/(1 - k_{N-2}).$$

This implies that FKS representation in terms of the basis functions can also be giveen in terms of ReLU functions as follows

$$y(x, \boldsymbol{\psi}_{FKS}) = \sum_{i=0}^{N-1} w_i \phi_i(x) \equiv w_0 ReLU(k_1 - x)/k_1 + \sum_{i=0}^{N-2} c_i ReLU(x - k_i)$$

where the coefficients c_i are rational functions of the knots k_i and weights w_i. Taking careful note of the representation of the basis functions at the two ends of the interval we obtain:

$$\begin{aligned} c_0 &= \alpha_1 w_1, \\ c_1 &= \gamma_1 w_2 - \beta_1 w_1, \\ c_i &= \gamma_i w_{i+1} - \beta_i w_i + \alpha_i w_{i-1}, \quad i = 2, \ldots, N-2. \end{aligned} \tag{9}$$

Hence the two representations of the functions by the ReLU NN and by FKS are formally equivalent, with the equivalence of the ReLU breakpoints with knot points, and with the *tri-diagonal linear* map (9) between the weights $\mathbf{w}$ and scaling constants $\mathbf{c}$. Hence (in principle) both networks are equally *expressive*. However, the information from the respective parameter sets $\boldsymbol{\theta}$ and $\boldsymbol{\psi}_{FKS}$ is encoded differently. and is critical for their training and in particular for the well-posedness of the solution representation and the associated training algorithm. In general, the coefficients w_i of the FKS are $\mathcal{O}(1)$, and may be monotone and of constant sign. However, if, for example, $k_i - k_{i-1}$ is small (which is often the case if we wish to approximate a function with a high gradient or high curvature) then the coefficients in (8) may well be large, and of non-constant sign, even if $u(x)$ has constant sign. As a consequence, the ReLU NN representation of a function may be ill-conditioned, even if the FKS representation is well-conditioned.

As an example, we consider the interpolating FKS $\Pi_1 u$ on the set of knot points for which $w_i = u(k_i)$. From (9) we obtain (after some manipulation) that if $k_{i+1} - k_{i-1}$ is small then

$$c_i \approx \frac{k_{i+1} - k_{i-1}}{2}\, u''(k_i), \quad i > 0.$$

In the case of $w_0 = u(0) = 0$ it also follows that

$$c_0 = \frac{(w_1 - w_0)}{k_1} \approx u'(0).$$

For the target function $u(x) = x(1-x)$ for which the optimal set of approximating knots is uniform with spacing h, i.e. $h = k_{i+1} - k_i$, we have $c_0 \approx 1$ and $c_i \approx -h$ for $i > 0$. Observe that there is a large jump from c_0 to c_1 leading to a poor balancing of the ReLU NN representation, while the FKS representation is better balanced. If, instead, we assume that the knot points have a density of $1/|u''|$ then the values of c_i are approximately constant and the ReLU NN representation is evenly balanced. This observation links the coefficients of the ReLU NN to the equidistribution condition we investigate in Section 4.2.

3.2 Conditioning of the training of the shallow ReLU NN and FKS representations

Using the results of the previous sub-section we now compare the conditioning of the problem of calculating the scaling factors $c_i, i = 0 \ldots N-2$ and the weights $w_i, i = 1 \ldots N-1$ of the shallow

ReLU NN and the FKS representations respectively (to give the best L_2 approximation of the function $u(x)$) on the assumption that the knot locations k_i are known. We show that the FKS problem (with the localised support basis functions) is well conditioned for all N whereas the ReLU NN problem (for which basis functions have global support) is increasingly ill-conditioned for larger values of N.

It follows from (9) that

$$\boldsymbol{c} = L\boldsymbol{w} \tag{10}$$

for the tri-diagonal linear operator L and we can compare the conditioning of the two problems by studying the properties of the matrix L. This matrix has a specific structure which we can exploit. It follows from (9) that

$$c_i = \frac{w_{i+1} - w_i}{k_{i+1} - k_i} - \frac{w_i - w_{i-1}}{k_i - k_{i-1}} \tag{11}$$

for $i = 2, \ldots, N-2$. Hence, the vectors $\boldsymbol{c} = (c_i)_{i=1}^{N-2}, \mathbf{w}^* = (w_i)_{i=1}^{N-2} \in \mathbb{R}^{N-2}$ are related via

$$\mathbf{c} = T\mathbf{w}^* + \gamma_{N-2} w_{N-1} \mathbf{e}_{N-2}, \tag{12}$$

where $\mathbf{e}_{N-2} = (0, 0, 0, \ldots 1)^T \in R^{N-2}$ and $T \in R^{(N-2)\times(N-2)}$ is the symmetric tri-diagonal matrix with β_i on the leading diagonal, and α_i and γ_i on the upper and lower first diagonals. Observe that to find $\mathbf{w}$ from $\mathbf{c}$ (and hence to invert the linear operator L) we can let w_{N-1} be initially unknown, invert T using the Thomas algorithm, to find $\mathbf{w}^*$ in terms of w_{N-1} and then fixing w_{N-1} from the relation

$$w_1 = c_0/\alpha_1.$$

Evidently the conditioning of L is then completely determined by the conditioning of T.

3.3 Uniformly spaced knots

We can make very precise estimates of the conditioning of the problem of finding $\mathbf{w}$ and $\mathbf{c}$ in this case. Suppose that we have uniformly spaced knots $k_i = i/(N-1)$, then (11) reduces to

$$c_i = (N-1)(w_{i+1} - 2w_i + w_{i-1})$$

The linear operator T then becomes the tri-diagonal Toeplitz matrix

$$T = (N-1)\begin{pmatrix} -2 & 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 1 & -2 & 1 & 0 & & & & \vdots \\ 0 & 1 & -2 & 1 & \ddots & & & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & 0 & \vdots \\ \vdots & & & \ddots & 1 & -2 & 1 & 0 \\ \vdots & & & & 0 & 1 & -2 & 1 \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & 1 & -2 \end{pmatrix} \in \mathbb{R}^{N-2\times N-2}.$$

We can then apply standard theory of $M \times M$ Toeplitz matrices of the tri-diagonal form taken by T. This theory implies that the eigenvalues of T are given by

$$\lambda_k = -2(M-1) + 2(M-1)\cos(\pi k/(M+1)) \in (-4(N-1), 0)$$

for $k = 1, \ldots, M$. In particular, we have $|\lambda_1| \leq \ldots \leq |\lambda_M|$, and for large M we then have

$$\begin{aligned} \lambda_1 &= -2(M-1) + 2(M-1)\cos\left(\frac{\pi}{M+1}\right) \\ &\approx -2(M-1) + 2(M-1)\left(1 - \frac{\pi^2}{2(M+1)^2}\right) = -\frac{\pi^2(M-1)}{(M+1)^2}. \end{aligned}$$

Similarly, for large M,

$$\lambda_M = -2(M-1) + 2(M-1)\cos\left(\frac{M\pi}{M+1}\right)$$
$$\approx -2(M-1) + 2(M-1)\left(-1 + \frac{\pi^2}{2(M+1)^2}\right) = -4(M-1) - \frac{\pi^2(M-1)}{(M+1)^2} \approx -4(M-1),$$

implying that the condition number $\kappa(T)$ satisfies

$$\kappa(T) = \left|\frac{\lambda_M}{\lambda_1}\right| \approx \frac{4(M+1)^2}{\pi^2} + 1, \quad \text{with} \quad M = N-2. \tag{13}$$

For the FKS representation of the approximation to the target function, we aim to solve the problem for the weights $\mathbf{w}$ given by

$$M\boldsymbol{w} = \boldsymbol{u},$$

where $M_{i,j} = \langle \phi_i, \phi_j \rangle$, giving

$$M = \frac{1}{6(N-1)} \begin{pmatrix} 4 & 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 1 & 4 & 1 & 0 & & & & \vdots \\ 0 & 1 & 4 & 1 & \ddots & & & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & 0 & \vdots \\ \vdots & & & \ddots & 1 & 4 & 1 & 0 \\ \vdots & & & & 0 & 1 & 4 & 1 \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & 1 & 4 \end{pmatrix} \in \mathbb{R}^{N\times N}.$$

This is also a tri-diagonal Toeplitz matrix. Hence, by the standard theory it has eigenvalues given by

$$6(N-1)\mu_k = 4 + 2\cos(\pi k/(N+1)) \in (2,6) \quad k = 1, \ldots, N.$$

Note that $\mu_N \leq \ldots \leq \mu_1$ with $6(N-1)\mu_N > 2$, $6(N-1)\mu_1 < 6$ and condition number $\kappa(M) = \mu_1/\mu_N < 3$. As $N \to \infty$ we have $6(N-1)\mu_N \to 2$, $6(N-1)\mu_1 \to 6$ and

$$\kappa(M) \to 3.$$

The associated normal equations for $\mathbf{w}$ are given by

$$M^T M\boldsymbol{w} = M^T\boldsymbol{u}.$$

The boundedness of $\kappa(M)$ in the limit of large N lies at the heart of the regularity, and eases the construction of the FKS approximation from the normal equations in this limit.

We now consider the related problem of finding the coefficients $\mathbf{c}$ of the ReLU approximation. This is of course closely linked to the problem of finding the FKS coefficients $\mathbf{w}$. For simplicity we will consider the case where $w_0 = w_{N-1} = 0$ so that the operators L and T are the same. From (10), we then obtain

$$M^T M T^{-1}\boldsymbol{c} = M^T\boldsymbol{u}$$

and multiplication by $(T^{-1})^T$ yields

$$\tilde{M}^T\tilde{M}\boldsymbol{c} = \tilde{M}^T\boldsymbol{u}$$

for the matrix

$$\tilde{M} = MT^{-1}.$$

To calculate the condition number of the matrix $\tilde{M}$ we note, that as the matrices T and M have exactly the same tri-diagonal Toeplitz structure, it follows from standard theory that they have

identical eigenvectors $\mathbf{e}_k$, $k = 1 \dots N$ with the eigenvector $\mathbf{e}_k$ having corresponding eigenvalues λ_k and μ_k for the matrices T and M respectively. It follows immediately, that the eigenvectors of the matrix $\tilde{M}$ are also given by the vectors $\mathbf{e}_k$ with corresponding eigenvalues

$$\nu_k = \mu_k/\lambda_k.$$

Observe that for large N we have $\nu_1 \to -6N/\pi^2$ and $\nu_N \to -1/2N$. Hence we deduce that as $N \to \infty$

$$\kappa(\tilde{M}) = \nu_1/\nu_N \to \frac{12\ N^2}{\pi^2}.$$

In particular, this shows that $\kappa(\tilde{M}) = \mathcal{O}(N^2)$ for $N \gg 1$. When solving the normal equations for the weights $\boldsymbol{c}$ of the ReLU network, we must consider the condition number κ of the matrix $\tilde{M}\tilde{M}^T$ which from the above calculation satisfies $\kappa = \mathcal{O}(N^4)$. This is very large even for moderate values of N such as $N = 64$, and is huge compared with the $\mathcal{O}(1)$ condition number for the FKS approximation. Classical optimisation methods (such as BFGS) have convergence rate dependent on

$$(\sqrt{\kappa} - 1)/(\sqrt{\kappa} + 1) = 1 - \mathcal{O}(1/N^2)$$

This leads to very slow convergence for large N. We see presently that the performance of the Adam optimiser is similarly very slow on this problem.

3.4 Non-uniformly spaced knots

In this case the matrices T and M are still symmetric and tri-diagonal, but are not constant along their diagonals. The (stiffness) matrix M takes values $(k_{i+1} - k_{i-1})/3$ along its leading diagonal, and values $(k_i - k_{i-1})/6$ and $(k_{i+1} - k_i)/6$ along its lower and upper diagonals, and thus is diagonally dominant. A simple application of Gershgorin's circle theorem to estimate the location of its eigenvalues λ_j implies that

$$\frac{1}{6}\min(k_{i+i} - k_{i-1}) \le \lambda_j \le \frac{1}{2}\max(k_{i+i} - k_{i-1}).$$

Hence (consistent with the estimate on a uniform mesh)

$$\kappa(M) < 3\ \frac{\max(k_{i+i} - k_{i-1})}{\min(k_{i+i} - k_{i-1}).}$$

Note that this estimate for the condition number is smallest for uniform knots, and increases if the knot spacing is non-uniform.

In contrast, each row of the matrix T has the respective coefficients:

$$\alpha_i = \frac{1}{k_i - k_{i-1}}, \quad -\beta_i = -\frac{1}{k_i - k_{i-1}} - \frac{1}{k_{i+1} - k_i}, \quad \gamma_i = \frac{1}{k_{i+1} - k_i}.$$

Observe that $\alpha_i - \beta_i + \gamma_i = 0$ for each i. It follows that the matrix T is near singular; indeed if $\mathbf{f} = (1, 1, 1, 1, , \dots 1)^T$ then

$$T\mathbf{f} = \mathbf{g} = (\gamma_1 - \beta_1, 0, \dots, 0, \alpha_{N-2} - \beta_{N-2})^T.$$

Hence

$$\|T^{-1}\| > \|f\|/\|g\| \to \infty, \quad \text{as} \quad N \to \infty.$$

The condition number of T, and hence of MT^{-1} thus increases without bound as N increases. Numerical experiments strongly indicate that, as in the case of the matrix M, the condition number increases more rapidly for non-uniform knots than uniform knots.

4 Loss functions and the associated training methods

Having seen the expressivity of the ReLU NN and of the FKS and the formal equivalence of the (shallow) ReLU NN and the FKS approximation in Section 2, we consider different loss functions in this section which can be used to train these to find the best approximation of the given function u. We will describe several *loss functions* approximating the L_2^2 error, which, whilst formally equivalent, lead to different training behaviours. The training will be done through a (first order) gradient-based approach and using the Adam optimiser with a suitable initial start. This procedure is chosen as it is a very common way of training a machine learning network.

4.1 General loss functions for the ReLU NN, the linear spline approximations, and their training

To unify notation, we denote the ReLU NN or the FKS approximation by $y(x)\colon [0,1] \to \mathbb{R}$ and the associated parameter values by $\mathbf{\Theta}$. Further, we denote the target function by $u\colon [0,1] \to \mathbb{R}$. We consider the loss function $\mathcal{L}_2^2$ based directly on the *mean-squared approximation error* of the NN. As both $y(x)$ and $u(x)$ are given for all $x \in [0,1]$ we have that the $\mathcal{L}_2^2$-error is given by

$$\mathcal{L}_2^2(\mathbf{\Theta}) = \int_0^1 (y(x,\mathbf{\Theta}) - u(x))^2 \, dx. \tag{14}$$

In practice, we cannot evaluate (14) for a general function and we have to consider various approximations to it. For this, we consider quadrature points $\{x_k\}_{k=0}^{s-1} \subset [0,1]$ for some large parameter $s \in \mathbb{N}\backslash\{0\}$ which can either be randomly generated with uniform distribution or regularly distributed over the domain $[0,1]$.

Assume that points $\{x_k\}_{k=0}^{s-1} \subset [0,1]$ are ordered so that $x_i < x_{i+1}$. An approximation to $\mathcal{L}_2^2$ in (14) is given by

$$L_2^2(\mathbf{\Theta}) = \sum_{i=0}^{s-1} (x_{i+1} - x_i)\Big(y(x_i,\mathbf{\Theta}) - u(x_i)\Big)^2. \tag{15}$$

Note that minimising (15) is a highly non-convex problem, and the minimisation process generally leads to sub-optimal solutions. A similar loss function is often used in the *Deep Ritz Method (DRM)* [8]. Similarly to the DRM, we will consider both the case where $\{x_k\}_{k=0}^{s-1}$ are sampled at each iteration or fixed during the training.

The purpose of the training of the ReLU NN or the FKS is to vary the parameters in $\mathbf{\Theta}$ systematically to reduce the loss function. Typically this is done using the Adam optimiser. The parameters in $\mathbf{\Theta}$ are updated. Note that the iterates of $\mathbf{\Theta}$ depend on the choice of the loss function, the optimisation method used and the initial values of $\mathbf{\Theta}$. Note that the minimisation of L_2^2 in (15) reduces to the pointwise error at sample points $\{x_i\}_{i=0}^{s-1}$ and neglects global properties of the function, such as its curvature. This is important when considering equidistributed knots in regions of highest curvature, which contribute to the interpolation error significantly.

4.2 Loss functions for the interpolating FKS

Whilst the interpolating FKS $\Pi_1 u$ is sub-optimal as a general approximating function, it can be considered as an excellent first guess for general optimisation procedures. Finding the interpolating FKS only involves determining the knots k_i for $i = 1, \dots, N-1$ which leads to a simplified approximation process approach for choosing the optimal knots which can then be combined with the notion of equidistribution. Having found the knots, the problem of finding the weights of the optimal FKS is a well conditioned linear problem.

4.2.1 The equidistribution loss function

Provided that the target function $u(x)$ is twice differentiable in (k_i, k_{i+1}), the local interpolation error of the linear interpolant y of u on $[k_i, k_{i+1}]$ satisfies

$$y(x,\mathbf{\Theta}) - u(x) \approx \frac{1}{2}(x - k_i)(k_{i+1} - x)u''(x_{i+1/2}) \tag{16}$$

for any $x \in [k_i, k_{i+1}]$ and some $x_{i+1/2} \in (k_i, k_{i+1})$. Note that this result even holds for the function $u(x) = x^{2/3}$ for which $u''(0)$ is singular. This yields

$$\int_{k_i}^{k_{i+1}} (y(x,\mathbf{\Theta}) - u(x))^2 \, dx \approx \frac{1}{120}(k_{i+1} - k_i)^5 |u''(x_{i+1/2})|^2,$$

implying

$$\mathcal{L}_2^2(\mathbf{\Theta}) = \int_0^1 (y(x,\mathbf{\Theta}) - u(x))^2 \, dx \approx \frac{1}{120} \sum_{i=0}^{N-2} (k_{i+1} - k_i)^5 |u''(x_{i+1/2})|^2. \tag{17}$$

Motivated by (17), we define

$$L_I(\mathbf{\Theta}) = \frac{1}{120} \sum_{i=0}^{N-2} (k_{i+1} - k_i)^5 \, |u''(k_{i+1/2})|^2 \tag{18}$$

where $k_{i+1/2} = k_i + k_{i+1}$ and we obtain $\mathcal{L}_2^2(\mathbf{\Theta}) \approx L_I(\mathbf{\Theta})$. Note that L_I only depends on the free knot locations for the interpolating FKS, and thus can be used to train the knots. However, unlike other loss functions L_I requires knowledge of u'' rather than point values of u and we will assume for the moment that u'' is known.

A powerful result in the form of the equidistribution principle, first introduced by de Boor [2] for solving boundary value problems for ordinary differential equations, gives a way of finding $\mathbf{\Theta}$ so that L_I in (18) is minimised. More precisely, minimising L_I in (18) can be expressed by equidistributing the error over each cell which results in the following lemma (see [14, Chapter 2]):

Lemma 1 (Equidistribution) *The loss L_I in (18) is minimised over all choices of the knots k_i if*

$$\rho_{i+1/2} = (k_{i+1} - k_i)\, |u''(k_{i+1/2})|^{2/5} \quad \textit{for all } i = 0, \ldots, N-2 \tag{19}$$

there is a constant $\sigma > 0$ so that

$$\rho_{i+1/2} = \sigma \quad \textit{for all } i = 0, \ldots, N-2. \tag{20}$$

The equidistribution principle in Lemma 1 replaces a non-convex optimisation problem by one with better convexity properties. A set of knot points satisfying these conditions is said to be *equidistributed*. The degree of equidistribution can then be used as a quality measure of the solution.

The algebraic equations (20) can then be solved directly [3], or iteratively using moving mesh techniques such as [13] for example. As an alternative, we propose training the interpolating FKS directly using an optimisation approach (such as Adam), but enforcing the equidistribution condition (20) directly through an equidistribution-based loss function.

To do this, we compute $\rho_{i+1/2}$ from knots k_i by (19) and set σ as their mean, i.e.

$$\sigma = \frac{1}{N-1} \sum_{i=0}^{N-2} \rho_{i+1/2}.$$

We define the equidistribution loss function L_E by

$$L_E(\mathbf{\Theta}) = \sum_{i=0}^{N-2} (\rho_{i+1/2} - \sigma)^2. \tag{21}$$

Remark 1 *In practice, to ensure regularity when u'' is small we replace the definition of $\rho_{i+1/2}$ in (19) by the regularised version*

$$\rho_{i+1/2} = (k_{i+1} - k_i)\, (\epsilon^2 + u''(k_{i+1/2}))^{1/5} \quad \textit{for all } i = 0, \ldots, N-2.$$

A value of $\epsilon^2 = 0.1$ works well in practice, and we will use it for all further calculations.

4.2.2 Optimal knots for the interpolating FKS

The optimal knots $k_i,\ i = 0, \ldots, N-1$, of L_I for $\Pi_1 u$ can be determined semi-analytically by using quadrature applied to (19) in the one-dimensional setting. Similar to Moving Mesh PDEs [14], we consider the *physical interval* $x \in [0,1]$ to be a map from a *computational interval* $\xi \in [0,1]$ so that $x = x(\xi)$ with $x(0) = 0$ and $x(1) = 1$. For $i = 0, \ldots, N-1$, the *ith* knot point k_i is given by $k_i = x(i/(N-1))$. Provided that N is large we can then use the approximations

$$k_{i+1} - k_i = \frac{1}{N-1} \frac{dx}{d\xi}(\xi_{i+1/2}) \quad \text{and} \quad x_{i+1/2} = x(\xi_{i+1/2})$$

for some $\xi_{i+1/2} \in (i/(N-1), (i+1)/(N-1))$, where $x_{i+1/2} \in (k_i, k_{i+1})$ is defined by (16). The equidistribution condition for L_I requires that $(k_{i+1} - k_i)^5 |u''(x_{i+1/2})|^2$ is constant for $i = 0, \ldots, N-1$ which yields

$$\left(\frac{dx}{d\xi}(\xi_{i+1/2}) \right)^5 \left(u''(x(\xi_{i+1/2})) \right)^2 = D^5 \tag{22}$$

for a suitable constant D. This results in a differential equation for x given by

$$\frac{dx}{d\xi}(\xi) = D\ \left(u''(x(\xi)) \right)^{-2/5}, \quad x(0) = 0, \quad x(1) = 1, \tag{23}$$

where value of D is fixed by the boundary condition so that

$$D = \int_0^1 (u''(x(\xi)))^{2/5}\, d\xi. \tag{24}$$

The optimal knots $k_i = x(i/(N-1))$ of a piecewise linear interpolating FKS on an equidistributed mesh can then be determined from the solution of (23) using (24). We substitute (22) into (18) to obtain

$$L_I(\mathbf{\Theta}) \approx \sum_{i=0}^{N-2} \left(\frac{dx}{d\xi}(\xi_{i+1/2})\right)^5 \frac{(u''(x(\xi_{i+1/2})))^2}{120(N-1)^5} = \sum_{i=0}^{N-2} \frac{D^5}{120(N-1)^5} = \frac{D^5}{120(N-1)^4}$$

up to leading order. Provided that $D = \mathcal{O}(1)$, the discretisation L_I of $\mathcal{L}_2^2$ for an interpolating FKS is $\mathcal{O}(1/N^4)$.

Example 1 *As an example, we consider the target function $u(x) = x^\alpha$ for some $\alpha \in (0,1)$, which has a derivative singularity at $x = 0$. We have $(u''(x))^2 = \alpha^2(1-\alpha)^2 x^{2\alpha-4}$. It follows from* (23) *that the equidistributed set of mesh points satisfies the ODE*

$$\frac{dx}{d\xi}\alpha^{2/5}(1-\alpha)^{2/5}x^{(2\alpha-4)/5} = D, \quad x(0) = 0, \quad x(1) = 1.$$

Integrating with respect to ξ yields

$$\frac{5}{2\alpha+1}\alpha^{2/5}(1-\alpha)^{2/5}(x(\xi))^{(2\alpha+1)/5} = D\xi,$$

implying that

$$x(\xi) = \left(\frac{2\alpha+1}{5}D\xi\right)^{5/(2\alpha+1)} (\alpha(1-\alpha))^{-2/(2\alpha+1)}$$

As $x(1) = 1$, we have

$$x(\xi) = \xi^{5/(2\alpha+1)}, \quad k_i = (i/(N-1))^{5/(2\alpha+1)}, \quad D^5 = \alpha^2(1-\alpha)^2(5/(2\alpha+1))^5.$$

Observe that $D = \mathcal{O}(1)$ so that the discretisation L_I of $\mathcal{L}_2^2$ with optimal knots for an interpolating FKS is $\mathcal{O}(N^{-4})$. For comparison, note that the discretisation L_I of $\mathcal{L}_2^2$ on a uniform set of N knot points of spacing $1/(N-1)$ satisfies

$$L_I(\mathbf{\Theta}) \approx \sum_{i=0}^{N-2} \frac{(u''(x(\xi_{i+1/2})))^2}{120(N-1)^5} = \sum_{i=0}^{N-2} \frac{\alpha^2(1-\alpha)^2\xi_{i+1/2}^{2\alpha-4}}{120(N-1)^5} \tag{25}$$

for an appropriate choice of $\xi_{i+1/2}$. As (25) *is dominated by the contribution in the first cell with $\xi_{1/2}$ of order $\mathcal{O}(1/N)$, this implies that*

$$L_I(\mathbf{\Theta}) = \mathcal{O}(N^{4-2\alpha}N^{-5}) = \mathcal{O}(N^{-1-2\alpha}). \tag{26}$$

In particular, the approximation error of the optimal knot choice is smaller than the one for the uniform set of knot points for any $\alpha \in (0,1)$. For $\alpha = 2/3$, for instance, (25) *reduces to $k_i = (i/(N-1))^{15/7}$ and by the definition of the weights $w_i = u(k_i)$ for the interpolating FKS we obtain $w_i = (i/(N-1))^{10/7}$. Observe that k_i and w_i are bounded, and that $k_i - k_{i-1}$ rapidly decreases from $i = 1$ to $i = 0$. The associated coefficients c_i of the ReLU NN satisfy* (9) *and are shown for $N = 64$ in Figure 1. As expected, $c_0 > 0 > c_1$ and the values of c_i are large for small i. The coefficients c_i of the ReLU NN expression of the approximation of $u = x^{2/3}$ show rapid changes, and this adds to the ill-conditioning problems for finding the values of c_i identified earlier.*

4.3 Equidistribution-linked loss functions and the related training of ReLU NN and FKS networks

Given the results on the loss for the interpolating FKS in Section 4.2, we consider two further procedures to train either a shallow ReLU NN or a general FKS which combine the usual optimisation with equidistribution and the calculation of the best interpolating FKS. These lead to procedures

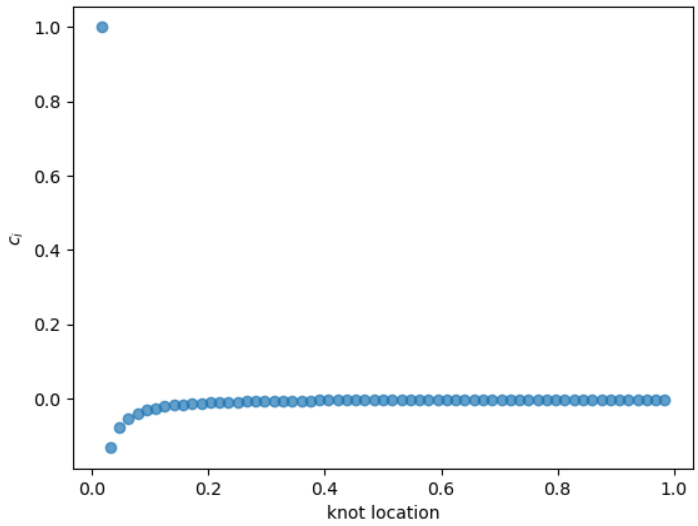


Figure 1: The unbalanced scaling coefficients c_i of the ReLU NN approximation equivalent to the Interpolating FKS for the target function $x^{2/3}$.

which allow the expressivity of these methods to be better realised with close to optimal approximations. We note that both of these methods require extra knowledge about the higher derivatives of the target function.

We introduce the *combined loss function* which combines the L_2^2-approximation error with the equidistribution condition and is given by

$$L_{comb} \equiv L_2^2 + \beta L_E. \tag{27}$$

Observe that if β is small the equidistribution condition acts to regularise the approximation error loss. Conversely when β is large then the approximation error acts to regularise the equidistribution condition. The latter case is surprisingly effective. Often when invoking the equidistribution condition directly in adaptive methods it is found that the resulting mesh is irregular. Adding the approximation error indirectly as a regulariser appears to smooth the resulting knot location.

Method 1: Two-level training

In this approach we first find the knots/breakpoints of the FKS/ReLU NN. We then find the weights/scaling factors. This method is motivated by the observation whilst the interpolating FKS $y(x) = \Pi_1 u(x)$ has quite a large error compared with the optimised solution, it is still quite close to the final solution, has a knot distribution very close to optimal, and has the correct asymptotic convergence properties.

(i) Suppose that we have either a FKS with knots k_i given directly, or a shallow ReLU NN with knots given indirectly by $-b_i/a_i$. We determine the knots k_i of the Interpolating FKS $u(x) = \Pi_1 u(x)$ by minimising the equidistribution loss (21) for the knots k_i. This is done by directly minimising the loss function L_{comb} using Adam with large β (say $\beta = 10$). (Typically we might start with a uniform distribution of knots in the interval $[0, 1]$.)

(ii) Use $\Pi_1 u(x)$ with the calculated knots as the *initial guess* for an optimising procedure based on the loss function L_2^2 in (15) to find the coefficients of either the FKS or the shallow ReLU NN. Or more simply, observing that the optimal knots for the FKS are very close indeed to those for the IFKS, freeze the knot locations and then solve the simple linear problem of finding the optimal weights w_i/scalings c_i.

We will see presently that this method applied as above is effective and well-conditioned in giving an expressive FKS approximation when using an optimisation method such as Adam.

In contrast when applied to the problem of finding the ReLU NN coefficients, we find that whilst part (i) of the two-level training correctly locates the knot points k_i, applying part (ii) is still very

slow for large N due to ill conditioning of the problem of fining the scalings c_i identified in Section 3.2. This is an inherent feature of the ReLU NN architecture we are considering.

A simple resolution is to precondition the problem of finding the ReLU coefficients c_i by applying the transformation (10) after step (i). Hence, given a ReLU NN we consider

Pre-conditioned two-level training of a ReLU NN.

1. Fix $\mathbf{c}$ and use equidistribution based loss to find k_i
2. Apply T^{-1} to determine $\mathbf{w}$ from $\mathbf{c}$ using the procedure outlined in Section 2 in which we find T^{-1} efficiently in $\mathcal{O}(N)$ operations using the well known *Thomas Algorithm*. This converts (preconditions) the ReLU NN optimisation problem to that of finding the optimal FKS coefficients w_i, k_i.
3. Apply (ii) to locate these coefficients.
4. Finally we transform back to find c_i from w_i using (10). As the matrix L is tri-diagonal the operation of multiplication by L is $\mathcal{O}(N)$.

Method 2: Combined training

As an alternative, we consider a combined approach by using the loss function L_{comb} with a smaller β, say $\beta = 0.1$. This procedure can again be used for a shallow ReLU NN or an FKS with some suitable initial conditions. For training the FKS, we use the loss function L_{comb} with the direct definition of k_i, while for training the NN we use the loss function L_{comb} with k_i implicitly defined by (6).

In computations we find that this method works quite well for the FKS, but requires a careful choice of the regularisation parameter β. In practice a value of $\beta \approx 0.1$ seems to be appropriate for calculations. For the ReLU NN problem it suffers from the same ill conditioning problems as the Two-level training method, so preconditoning (as above) must be applied for it to be effective.

4.4 Summary of loss functions

Based on the discussions in the previous subsections, we consider the following loss functions for the NN and the linear spline approximations:

- L_2^2 in (15)
- L_{comb} in (27) with the two training procedures described in Section 4.3

5 Numerical Results

In this section, we give a series of numerical results for the training and convergence of ReLU NNs and different linear spline approximations, including the FKS. We consider a set of target functions using the different loss functions and their training described in Section 4.4. As a summary we find that the FKS trains much faster than the usual ReLU NN architecture, largely due to the ill-conditioning problems identified in Section 2. Optimal results are obtained when training a FKS with the equidistribution-based loss function L_{comb} and two-level training. Similar results are found when using the same method, with preconditioning, to train the ReLU NN.

To make the comparisons between the FKS approximations and the ReLU NN comparable we assume that we have N knots k_i for $i = 0, \ldots, N-1$, with $k_0 = 0$ and $k_{N-1} = 1$. Accordingly, we consider the approximation of five different target functions $u_i(x)$ for $u(x)$ on the interval $[0, 1]$ of increasing complexity:

$$
\begin{aligned}
u_1(x) &= x(1-x), \\
u_2(x) &= \sin(\pi x), \\
u_3(x) &= x^{2/3}, \\
u_4(x) &= \tanh(100(x-0.25)), \\
u_5(x) &= \exp\big(-500(x-0.75)^2\big)\sin(20\pi x).
\end{aligned}
$$

Note that the first two target functions are smooth, the third is singular as $x \to 0$, and the last two examples represent a smoothed step function and an oscillatory function, respectively, both with small length scales. Our interest will be in the importance and speed of the training, the role of the initial start, and the convergence of the resulting approximation as $N \to \infty$. Our extensive experiments on different optimisation methods such as Adam, Gauss-Newton and Nelder-Mead optimisers all led to similar optimal solutions, computation times and training behaviour. Consequently, we focus on the popular Adam optimiser for the remainder of this section.

5.1 Numerical results for linear spline approximation using standard training procedures

As a motivation for this section, in Table 1, we consider the approximation of the singular target function $u(x) = x^{2/3}$ for various linear spline approximations (not yet including the ReLU NN) and different values of N using the Adam optimiser, and compare the performance in terms of the L_2^2 loss function. As expected, the performance of the different interpolants in (a)-(c) in Table 1 is significantly worse than the FKS training in (d)-(g) in Table 1, motivating us to focus on FKS training in the following, when considering linear spline approximants. In terms of the equidistribution-linked loss functions described in Section 4.3, the performance of the combined FKS training in (g) in Table 1 is comparable to the two-level training in (f) in Table 1, though this becomes less stable for $N > 64$ (see Figure 6), which suggests that it is sufficient to consider two-level FKS training for the experiments with equidistribution-linked loss functions. Observe the excellent accuracy $\mathcal{O}(10^{-10})$ of the best approximations.

Linear spline approximant	$N = 16$	$N = 32$	$N = 64$
(a) Interpolant on a uniform mesh	2.18×10^{-5}	3.99×10^{-6}	7.47×10^{-7}
(b) Best least squares approximation on a uniform mesh	3.41×10^{-6}	1.64×10^{-6}	5.24×10^{-7}
(c) Interpolant on an optimal mesh	3.45×10^{-7}	1.90×10^{-8}	1.13×10^{-9}
(d) FKS starting from (a)	8.91×10^{-7}	1.39×10^{-7}	8.08×10^{-8}
(e) FKS starting from (c)	5.42×10^{-8}	3.17×10^{-9}	1.56×10^{-10}
(f) Two-level FKS training	7.48×10^{-8}	3.00×10^{-9}	5.52×10^{-10}
(g) Combined FKS training	1.10×10^{-7}	5.54×10^{-9}	5.95×10^{-10}

Table 1: Comparison of optimal linear spline approximations for the target function $u(x) = x^{2/3}$ in terms of the L_2^2 loss function for different values of N using the Adam optimiser.

5.2 Ill-conditioning of the ReLU NN training

Results on the training of a standard ReLU NN were not included in Table 1, as the performance of the training algorithms for ReLU NNs is significantly worse than those for FKS for these problems. This is due to the ill-conditioning of the training of the scaling coefficients c_i of the ReLU NN representation discussed in Section 2. We illustrate this by looking at the loss as a function of the training iterations in Figure 2 where we consider $N = 64$ and the target function $u_4(x) = \tanh(100(x - 1/4))$. We use Adam with an equidistribution based loss function in the first stage of the two-level method to train both an FKS and a ReLU NN to find the optimal knots k_i. Starting from these knot locations and with an initially random distribution of the scaling coefficients/weights, we then apply the second stage to further train both networks to find the optimal values of the coefficients c_i for the ReLU NN and w_i for the FKS, respectively, using the (combined) loss function (27). It is clear from Figure 2 that the FKS trains more quickly to a much more accurate approximation than the ReLU NN for this target function. This is due to the poor conditioning of the problem for large N associated with the condition number of the normal equations identified in Section 2. The FKS uses linear splines with small support as a basis and the resulting linearisation M of the system used to train the weights is a diagonally dominant matrix with low condition number $\kappa_{FKS} = \mathcal{O}(1)$. In contrast the ReLU functions have global support, and the normal equation matrix is both full and has a high condition number $\kappa_{ReLU} = \mathcal{O}(N^4)$. For example if $N = 64$ then a direct calculation for this specific approximation problem gives $\kappa_{ReLU} = 6.3 \times 10^7$ and $\kappa_{FKS} = 3.7$. Similar poor conditioning is observed for all of the target functions considered.

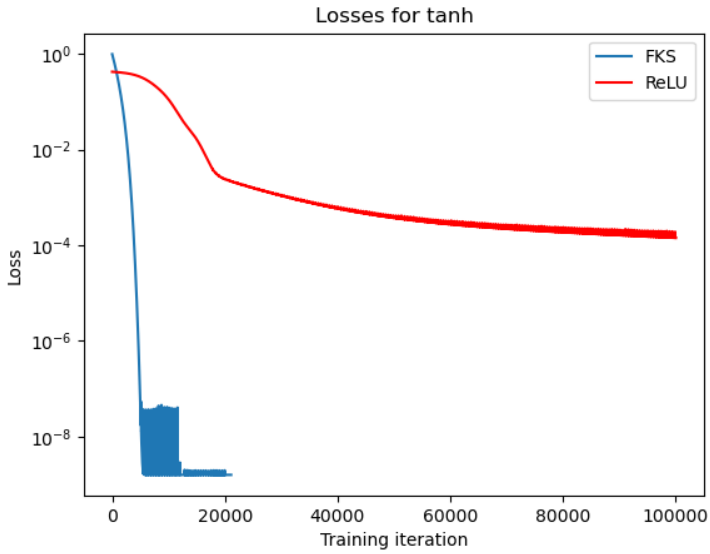


Figure 2: A comparison of the training times and convergence for an FKS and a shallow ReLU NN for the target function $u_4(x) = \tanh(100(x - 1/4))$ with $N = 64$.

5.3 Optimal approximations

To investigate the dependence of the approximation accuracy on N further, we calculate a near optimal piecewise linear approximation to each of the target functions using a moving mesh partial differential equation (MMPDE) method. For each of the target functions u_j we define a monitor function $m_j = (\epsilon_j + u_j''^2)^{1/5}$. The value of the non-negative regulariser ϵ_j is not critical, but has to be chosen with some care when the value of u_j'' varies a lot over the domain to ensure that regions where u_j'' is relatively small still have some knots within them. We take $\epsilon_j = 0$ for $j = 1, 2$ and $\epsilon_j = 1$ for $j = 3, 4$. It then follows that the optimal knot points $k_i, i = 0, \ldots, N-1$, are given by $x(\xi)$ where $\xi = i/(N-1)$ and $x(\xi)$ satisfies the ODE

$$\frac{dx}{d\xi} = \frac{\int_0^1 m_j(\tilde{x})\, d\tilde{x}}{m_j(x)}, \quad x(0) = 0. \tag{28}$$

To find the best knot points k_i for the piecewise linear IFKS $\Pi_1 u_j(x)$ we solve (28) and evaluate the integral in the expression by using a high-order Gear solver with a high tolerance. Having found these points we then use the values $(k_i, \Pi_1 u_j(k_i))$ as the initialisation of the optimisation procedure to find the FKS. Note that one can show as in Example 1 for the target functions $u_1(x) = x(1-x)$ and $u_3(x) = x^{2/3}$ that the optimal knot points are given by

$$k_i = i/(N-1) \quad \text{and} \quad k_i = (i/(N-1))^{15/7},$$

respectively, so no differential equation needs to be solved numerically in these cases to find the optimal knots.

We present results for each of the target functions in Figure 3. For each target function we study the convergence of the approximation by plotting the L_2^2 error as a function of N in the three cases of (i) (blue) the PWL interpolant defined over a uniform mesh, (ii) (orange) the optimal PWL interpolant, and (iii) (green) the FKS fully trained with the knots and weights of the optimal PWL interpolant (obtained using the MMPDE) as the start of the Adam optimisation. In all cases the training was rapid. We see that the FKS (trained with the optimal PWL interpolant as a start) and the optimal PWL interpolant both show $\mathcal{O}(1/N^4)$ convergence of the loss function in each case. The FKS gives the best results as expected, often significantly better than the optimal interpolant. In this context the 'optimal' FKS achieves the best expressivity of piecewise linear approximations which includes the ReLU approximation. The error of the PWL interpolant on a uniform mesh is always much poorer than the others, but often better than the naively trained ReLU network. For the *smooth* target functions u_1, u_2, u_4, u_5 we see $\mathcal{O}(1/N^4)$ convergence for values of N so that $h = 1/N$ is smaller than the smallest length scale of the target function, but with a much larger constant of proportionality than either the optimal FKS or the optimal PWL interpolant. In the

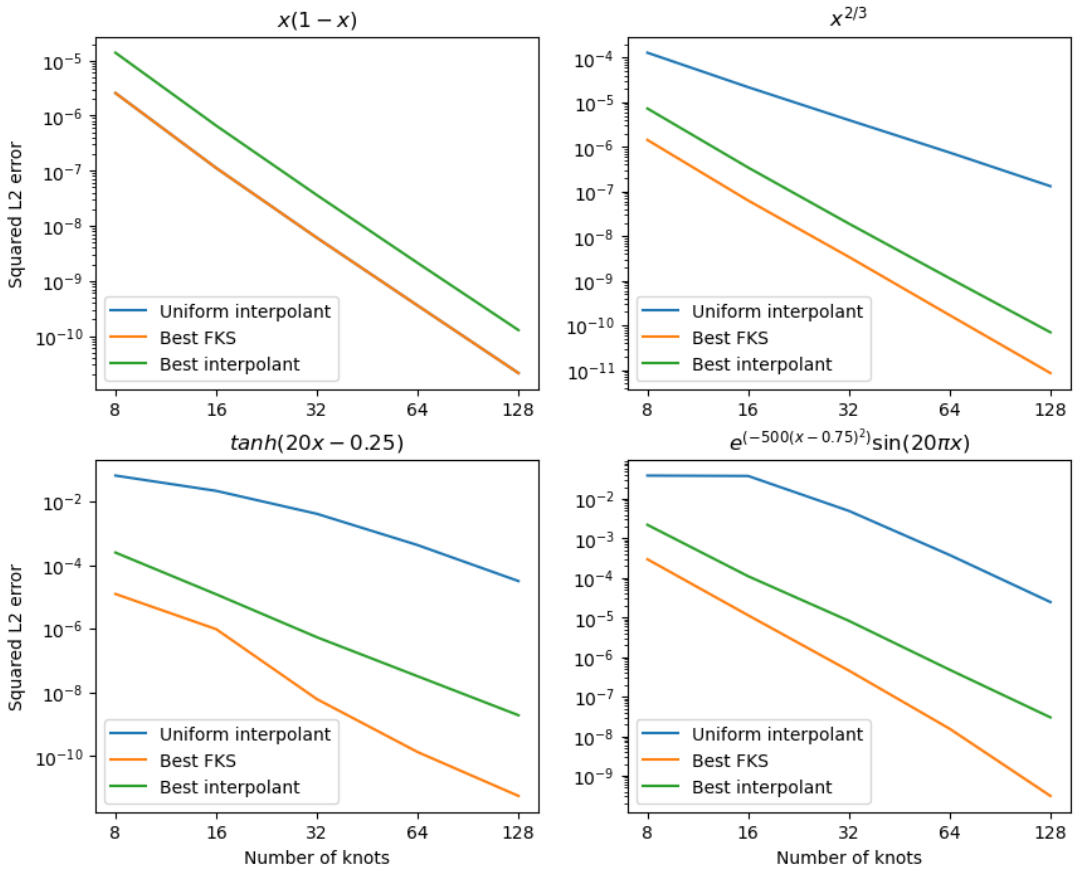


Figure 3: Comparison of L_2^2 for different linear spline approximations as a function of the number of knots: blue - PWL interpolant on a uniform mesh, orange - optimal PWL interpolant, green - optimal FKS.

case of the *singular* target function $u_3(x)$ we see a slower convergence at the theoretical rate of $\mathcal{O}(1/N^{7/3})$.

5.4 Numerical results for the ReLU NN using standard training procedures

In this section, we present numerical results for the 'usual' training of a shallow ReLU NN. For our first example, we consider the results of applying a standard training procedure for a shallow ReLU NN in PyTorch (without any form of pre-conditioning). We take a shallow ReLU NN with the usual L_2^2 loss function given in (15), with width $W = 16$, and consider the approximation of the target functions $u_i(x), i = 2, 3, 5$. For this calculation the default PyTorch parameter initialisation is used, and the location of the implicit knot locations over 50,000 optimisation iterations using Adam with a learning rate of 10^{-3} are illustrated in the top part of Figure 4. The lower part of Figure 4 shows the quality of the function approximation, with the implicit knot locations plotted. We can see that during the course of the training procedure the knots can cross over, can merge and can leave the domain $[0, 1]$.

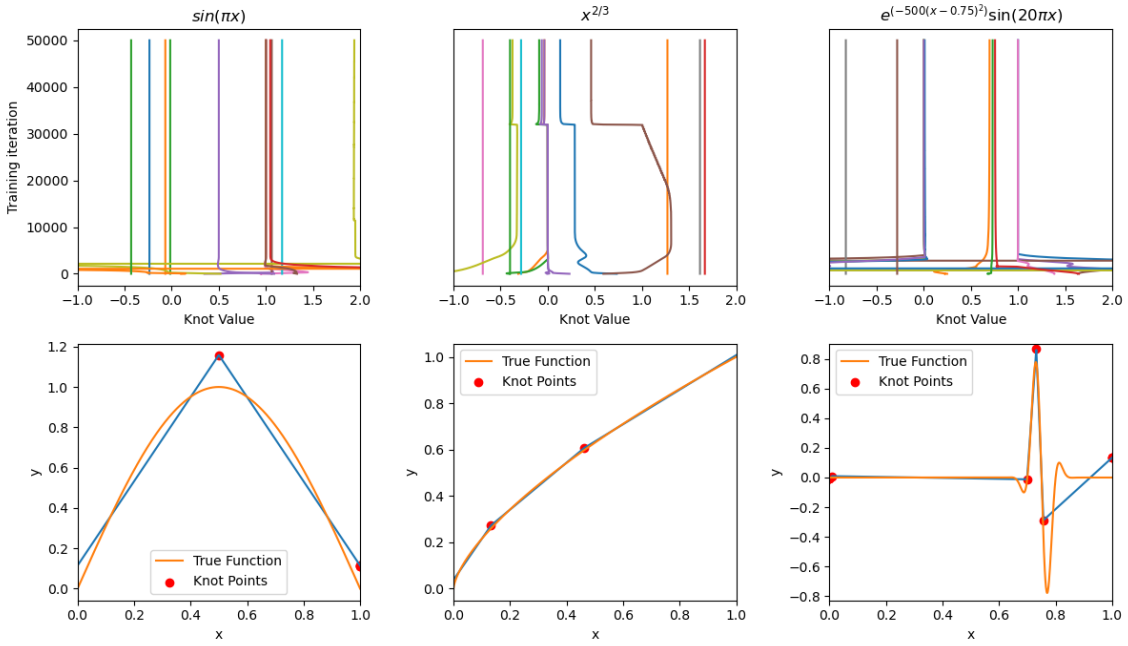


Figure 4: Comparison of (top) Knot evolution with standard Gaussian initialisation and (bottom) the trained approximation with the knot points indicated for different target functions.

We repeat the above but with the constraint on the starting parameters that the initial implicit knot locations given by (6) are all required to lie in the interval $[0, 1]$. The corresponding results are shown in Figure 5. We can see that the results are better than those presented in Figure 4 but are still far from optimal, and show evidence of knot crossing and merging during training.

The resulting values of the L_2^2 loss function in the respective cases are:

$$\text{Random}: \quad u_2 : 3.948 \times 10^{-3}, \quad u_3 : 2.611 \times 10^{-5}, \quad u_5 : 8.524 \times 10^{-3},$$

$$\text{Constrained}: \quad u_2 : 4.77 \times 10^{-6}, \quad u_3 : 2.30 \times 10^{-6}, \quad u_5 : 8.24 \times 10^{-3}.$$

We note that the loss values for the constrained start are in general better than the large loss values we obtain for the random start. However, in the case of u_3, they are still not close to those given by the FKS.

These calculations demonstrate that the standard machine learning based techniques, when used in a simple example of functional approximation, can lead to a significantly sub-optimal solution in each of the three cases of the target functions. This can be seen by the distribution of the knots which is far from optimal. Note that they are irregular and unevenly spaced whereas the optimal knots would be expected to be symmetric, and regularly spaced. Furthermore the knots for the

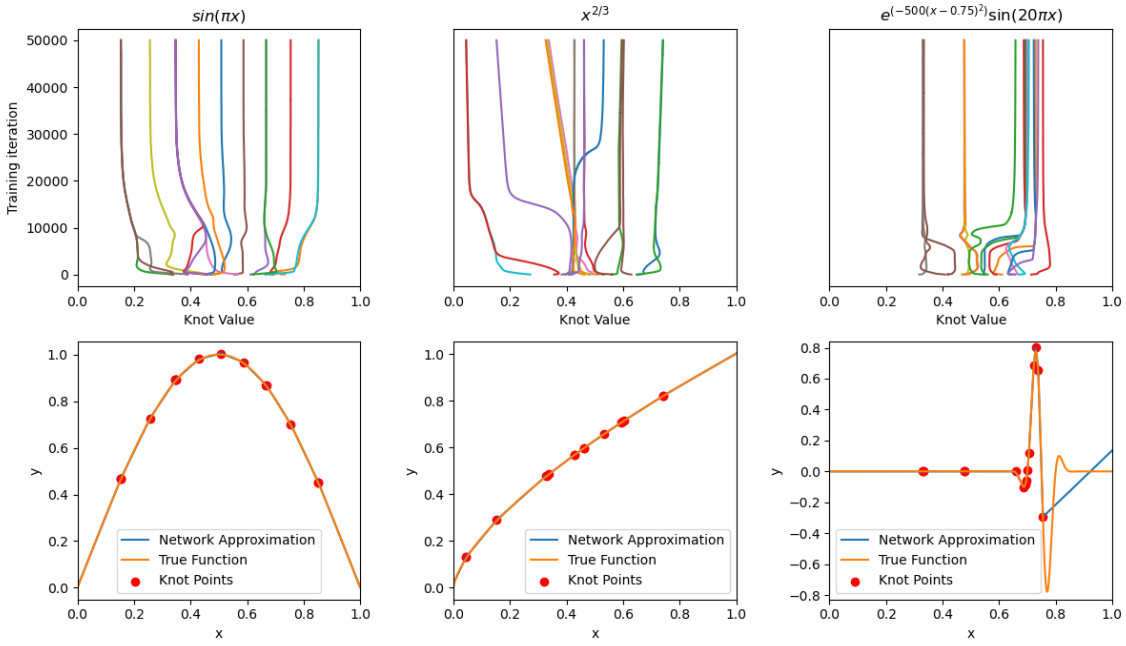


Figure 5: Comparison of (top) Knot evolution, with initialisation in $[0, 1]$ and (bottom) the trained approximation with the knot points indicated for different target functions.

target functions u_3 and u_5 show no sign of clustering close to the singularity at $x = 0$ (in the case of u_3) or the points of rapid variation (in the case of u_5). In all cases the training of the knots is erratic. We conclude that the commonly used training methods and loss functions for a shallow ReLU NN perform badly on even simple target functions. We now see how we can improve on this performance.

5.5 Numerical results for the training ReLU NN and FKS approximations using equidistribution-linked loss functions and pre-conditioning

In this subsection, we present numerical results for training a ReLU NN, a linear spline (without free knots), and a FKS for each of the target functions. In all case we use the Adam optimiser with a learning rate of 10^{-3}. We consider the use of different loss functions, including the equidistribution-linked loss function L_{comb} in (27). We also consider using the two-level training procedure described in Section 4.3 where we first train the knots of the approximation and then the weights (in the case of the ReLU NN with and without pre-conditioning.)

We present the results as follows. For each target function $u_i(x)$ we consider using the following methods: (a) A standard ReLU NN. This is the usual implementation of the ReLU NN approximation in PyTorch as considered in the previous subsection, with random initialisation of the Adam optimiser but with the knot locations constrained to lie in $[0, 1]$. (b) ReLU NN and FKS trained using the standard L_2^2 loss function. (c) ReLU NN and FKS trained using the combined/joint L_2^2 and equidistribution loss function. (d) ReLU NN and FKS trained in a two-level manner (in the case of the ReLU NN without and with pre-conditioning) to first to locate the knots k_i and then the weights (d) best least squares approximation, which takes a uniform knot distribution and optimises the weights (this is classic least squares approximation on a fixed mesh). Apart from case (a) we take the initialisation for the Adam optimiser to be the piecewise linear interpolant of the target function with uniform knots $k_i = i/(N-1), i = 0 \dots N-1$ represented either in the form of the ReLU NN or as a piecewise linear spline approximation.

We show our numerical results for different target functions in Figure 6 in terms of different aspects: (i) The function approximation of the FKS trained using the two-level approach method when $N = 64$. This will consistently turn out to be the best approximation. (ii) The convergence of the knot values during training when using the equidistribution based loss function, where we plot (on the x-axis) the location of the knots as a function of time t on the y-axis. This figure is nearly identical for both the FKS and the ReLU NN approximations and the smooth evolution of the

knots should be compared to erratic behaviour seen in Figures 4 and 5. (iii) The convergence of the L_2^2 approximation error of each method as a function of N,

The *basic ReLU NN* trained using the direct L_2^2 loss function consistently performs badly. Whilst the approximation error does in general decrease with N, it is usually the worst of all the approximations by a significant margin. It also takes a much longer time to train than any other method due to the ill-conditioning of the system.

The *ReLU NN trained using the direct two-level approach without preconditioning* in which the knots k_i are first trained and then the weights are calculated *without* using pre-conditioning, works in general rather better than the basic ReLU NN above, particularly for the lower values of N where conditioning issues are less severe, but is outperformed by the FKS, particularly for larger N where ill-conditioning issues arise. In the two-level approach the knot locations are found accurately in stage (i) of the training using the equidistribution based loss function and are presented in the middle figure. Again, contrast this smooth evolution with the erratic convergence of the knots of the basic ReLU NN presented earlier in Figures 4 and 5. However, having found the knot locations correctly, finding the resulting weights c_i is ill-conditioned, as expected from the results of Section 2. Consequently the convergence to the correct solution is very slow, as was illustrated in Figure 2.

The *ReLU NN trained using the direct two-level approach with preconditioning* behaves essentially identically to the FKS with two-level training, see later.

The *ReLU NN* trained using the joint loss function behaves in a similar manner to the two-stage trained ReLU NN without preconditioning.

The *regular spline* (classical least squares approximation on a uniform mesh of mesh spacing $h = 1/N$ so that there is no training of the knots) works well for the first two smooth target functions with convergence $\mathcal{O}(N^{-4})$. For the singular target function $u_3(x)$ it converges but at the sub-optimal rate of $\mathcal{O}(N^{-7/3})$ given by (26). (The same behaviour is seen for the piecewise linear interpolant on a uniform mesh.) In the case of the smooth target functions $u_4(x)$ and $u_5(x)$ we see sub optimal convergence when $h = 1/N$ is greater than the smallest length scale of the problem (which is $1/100$ for u_3 and $1/\sqrt{500} = 1/22.3$. For larger values of N (and smaller values of h) we see $\mathcal{O}(1/N^4)$ convergence. In all cases the error of the regular spline is much larger than that of the best FKS approximation, but in general better than that of the ReLU NN.

The *full FKS (with free knots and weights) trained with either the joint loss function (knots and weights together with an equidistribution constraint) or the two-level training (first training the knots using equidistribution)* then the weights consistently is the best performing method, alongside the near identical example of the *pre-conditioned two-level training of the ReLU NN* with similar results in both cases. The results show excellent and consistent convergence at a rate of $\mathcal{O}(1/N^4)$. The convergence of the knots in all cases is regular and monotone, indicating a degree of convexity to the problem. The resulting knots are regularly spaced, showing symmetry and clustering where needed. The training converges rapidly in all cases. Note that pre-training the knots using the MMPDE as discussed in Section 5.1 gives a slightly smaller error after optimisation than the best trained FKS/ReLU NN (although the difference is not large). This is to be expected as the MMPDE has employed an accurate solver to determine the knot points.

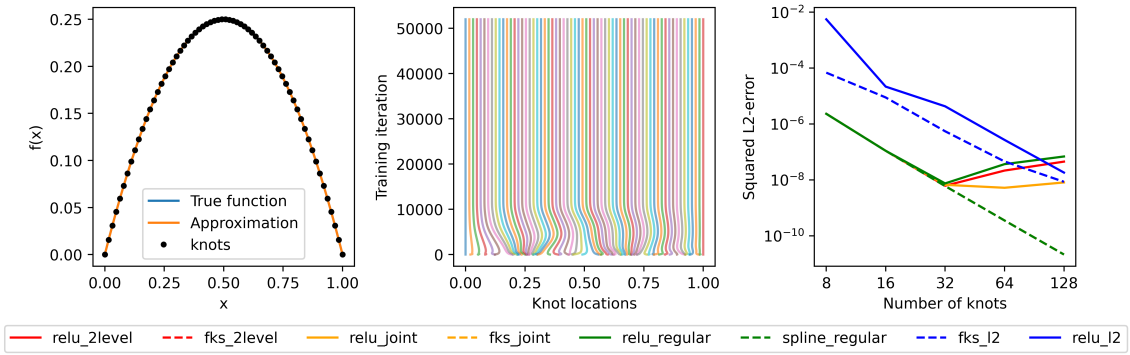
The *FKS trained using the direct L_2^2 loss* function performs better than the remaining approximations, but a lot worse than the optimal FKS. We do not in general see $\mathcal{O}(1/N^4)$ convergence in this case.

5.6 Conclusions from the results

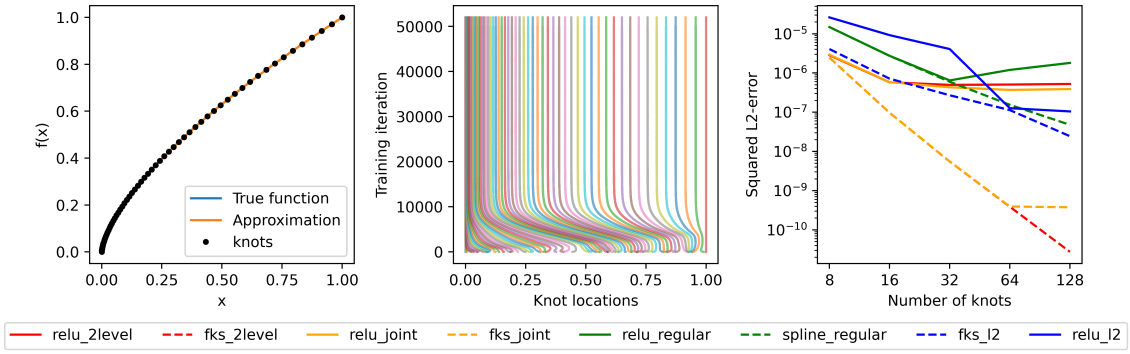
We draw the following conclusions from our numerical calculations.

A naively (in the sense of using the usual training procedures) trained shallow ReLU NN, always gives a poor approximation of the target functions. The knot points are generally badly placed, especially if we start from a random parameter set. Starting from a uniform set of knots gives an improvement, but still leads to a poor approximation.

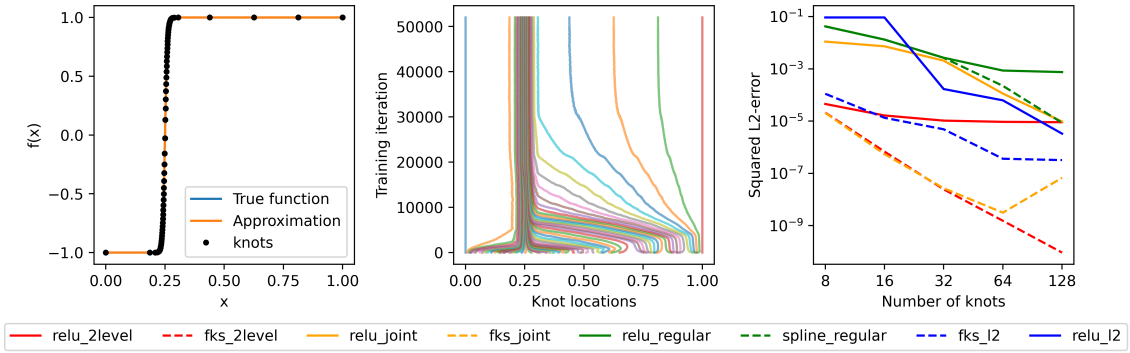
Two-level training (using equidistribution but without using preconditioning) of a ReLU NN is significantly better than non-equidistribution based training, and the knot points are close to optimal locations. However the training of the weights $c_i$is far from optimal due to ill-conditioning for larger



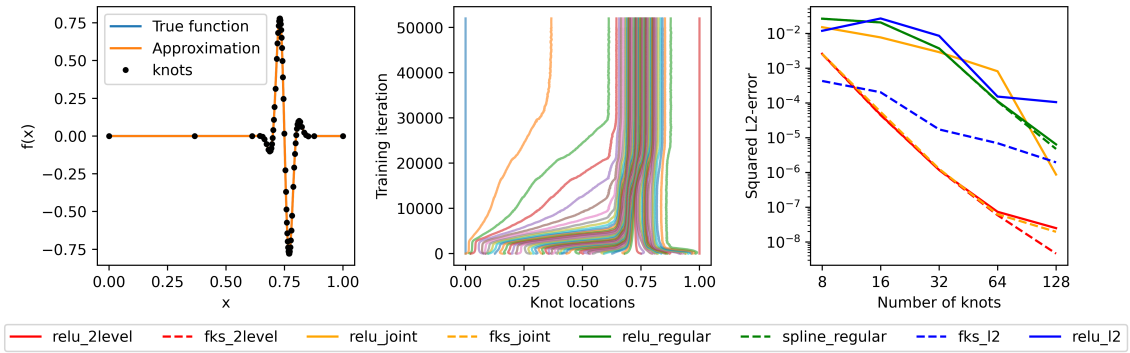
(a) Target function $u_1(x) = x(1-x)$



(b) Target function $u_3(x) = x^{2/3}$



(c) Target function $u_4(x) = \tanh(100(x - 1/4))$



(d) Target function $u_5(x) = \exp(500(x - 3/4))^2) \sin(x)$.

Figure 6: Comparison of (i) Function approximation, (ii) knot evolution, (iii) Convergence for different target functions

values of N, leading to an approximation that is still far from being as expressive as it could be.

Two-level training of an FKS is better than non-equidistribution based training as it is faster and more accurate and gives a good approximation which is very expressive. Preconditioning the ReLU NN in the two-level training (effectively turning it into an FKS) gives almost identical results to using an FKS directly.

The MMPDE approach with a high-order solver is best of all in terms of the approximation and results in an accurate piecewise linear approximation. This shows us that we can train a system get the expressivity we want with a piecewise linear approximation, equivalent to a ReLU NN, but comes at a high computational cost.

5.7 Further numerical experiments

In this paper we have confined the theory, and the earlier numerical experiments, to the case of the shallow ReLU NN. This is done to allow a direct theoretical and numerical comparison with the linear spline FKS. Indeed there is a direct identification of the FKS knots with the shallow ReLU NN breakpoints, and a linear relation between the FKS weights and the shallow ReLU NN scaling coefficients.

However, we make the obvious point that in machine learning applications other activation functions than ReLU functions are often used. Examples being the leaky ReLU, ReLU-cubed, tanh and sigmoid activation functions. Furthermore, approximations using deep networks (with large depth L) are usually considered instead of shallow ones. Apart from shallow networks with leaky ReLU activation, it is difficult to make a direct comparison of any of these with the FKS approximation. (For example the link between the knots of a FKS and the breakpoints of the deep network approximation is very subtle [5].) This is why it is much harder to develop a direct theory in this case, and we have to confine ourselves to a series of numerical comparisons.

As a *first calculation* we considered the training (using Adam) of a shallow NN with tanh activation functions, using the standard methods on the same target functions as before, with the standard L_2^2 loss function. In Table 1 we present the L_2^2 error when training the tanh based single layer network on the same test functions $u_i(x)$ as before with N knots. We also show the total number P of parameters. The results in all cases were very similar (and equally poor) to earlier results when the ReLU activation function was used with the direct training methods. Observe, for example, the especially poor performance on the highly oscillatory test function u_5. In particular such networks suffer from the same problems of ill conditioning as were observed earlier.

N	u_1	u_2	u_3	u_4	u_5	P
8	4.6×10^{-9}	1.70×10^{-6}	3.55×10^{-4}	1.08×10^{-3}	1.77×10^{-2}	25
16	1.56×10^{-9}	1.20×10^{-8}	1.62×10^{-7}	8.01×10^{-4}	1.53×10^{-2}	49
32	1.21×10^{-8}	1.41×10^{-8}	1.50×10^{-7}	6.17×10^{-4}	1.64×10^{-2}	97
64	5.25×10^{-9}	8.88×10^{-9}	2.15×10^{-7}	5.54×10^{-4}	1.77×10^{-2}	193
128	2.21×10^{-8}	8.00×10^{-9}	2.15×10^{-7}	4.40×10^{-4}	1.90×10^{-2}	385

Table 2: The L_2^2 error when training a shallow network using the tanh activation function to approximate the test functions $u_i(x)$.

As a second calculation we consider a 2-layer neural network, with ReLU activation. To make a direct comparison with a FKS with 16 knots which has 49 trainable parameters, we take a width of $W = 5$ such that the 2-layer network has the (comparable) 46 trainable parameters. When trained on the same test functions as before (using Adam) and the L_2^2 loss function we see presented in Table 2 the poor convergence observed for the shallow ReLU NN. The behaviour is rather erratic and is critically dependant on the initial values of the parameters. We illustrate this by giving the variance of the error in this case. We compare the results in Table 2 with those presented in Table 1.

In none of these cases (for example with a direct comparison between Tables 1, 2 and 3, do we see the same performance as the FKS or ReLU NN trained using the (pre-conditioned) two-level training method.

	u_1	u_2	u_3	u_4	u_5
L_2^2 error	4.2×10^{-3}	5.47×10^{-2}	3.45×10^{-2}	1.8×10^{-1}	1.85×10^{-2}
Variance	9.71×10^{-5}	1.42×10^{-2}	1.35×10^{-2}	5.40×10^{-2}	1.09×10^{-4}

Table 3: Results of training a two-layer network with 5 nodes on the test functions.

6 Conclusions

The results of this paper have shown that it is hard to train a shallow ReLU NN to give a particularly accurate approximation for either smooth or singular target functions if a standard method is used. In contrast, using a two-level equidistribution based training approach, combined with pre-conditioning it is certainly possible to train a formally equivalent FKS to give very accurate approximations on the same set of target functions. Hence the high level of expressivity theoretically possible for the ReLU NN approximation is achieved in training by the FKS and the pre-conditioned ReLU NN, but not, in practical training using standard methods, by the standard ReLU NN. The reason for this seems to be two-fold. Firstly, it is necessary to have (in both the FKS and the ReLU NN) control over the knot location using (for example) an equidistribution based loss function. Secondly, the process of training the weights of the FKS is much better conditioned than that of finding the scaling coefficients of the ReLU functions which have a global support. As we have seen, the combination of these two factors leads to poor ReLU NN approximations. Both issues can be overcome when training a ReLU NN by using an equidistribution based loss function and then pre-conditioning the ReLU problem to give it the same structure as an FKS. By doing this we can then reliably realise the full expressivity of the ReLU NN approximation. These results have implications in the use of ReLU NN to approximate functions in the context of a PINN and other aspects of (scientific) machine learning.

In this paper we have deliberately mainly confined ourselves to piecewise linear function approximations in one spatial dimension. This is because we can then make a direct comparison between a shallow ReLU NN and an FKS, while the results will be generalised to higher dimensions and deep networks as part of our future work.

References

[1] P. Bohra et al. "Learning Activation Functions in Deep (Spline) Neural Networks". In: *IEEE Open Journal of Signal Processing* 1 (2020), pp. 295–309.

[2] C. de Boor. "Good Approximation by Splines with Variable Knots". In: *Spline Functions and Approximation Theory: Proceedings of the Symposium held at the University of Alberta, Edmonton May 29 to June 1, 1972*. Ed. by A. Meir and A. Sharma. Basel: Birkhäuser Basel, 1973, pp. 57–72.

[3] C. de Boor. "Good approximation by splines with variable knots. II". In: *Conference on the Numerical Solution of Differential Equations*. Ed. by G. A. Watson. Berlin, Heidelberg: Springer Berlin Heidelberg, 1974, pp. 12–20.

[4] C. De Boor. *A practical guide to splines*. Springer-Verlag, 2001.

[5] R. DeVore, B. Hanin, and G. Petrova. "Neural network approximation". In: *Acta Numerica* 30 (2021), pp. 327–444.

[6] R. A. DeVore. "Nonlinear approximation". In: *Acta Numerica* 7 (1998), pp. 51–150.

[7] R. A. DeVore and G. G. Lorentz. *Constructive Approximation*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 1993.

[8] W. E and B. Yu. "The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems". In: *Communications in Mathematics and Statistics* 6.1 (2018), pp. 1–12.

[9] K. Eckle and J. Schmidt-Hieber. "A comparison of deep networks with ReLU activation function and linear spline-type methods". In: *Neural Networks* 110 (2019), pp. 232–242.

[10] P. Grohs and G. Kutyniok. *Mathematical Aspects of Deep Learning*. Cambridge University Press, 2022.

[11] J. He et al. “ReLU Deep Neural Networks and Linear Finite Elements”. In: *Journal of Computational Mathematics* 38.3 (2020), pp. 502–527.

[12] K. Hornik, M. Stinchcombe, and H. White. “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* 2.5 (1989), pp. 359–366.

[13] W. Huang, Y. Ren, and R. D. Russell. “Moving Mesh Partial Differential Equations (MMPDES) Based on the Equidistribution Principle”. In: *SIAM Journal on Numerical Analysis* 31.3 (1994), pp. 709–730.

[14] W. Huang and R.D. Russell. *Adaptive Moving Mesh Methods*. Applied Mathematical Sciences. Springer New York, 2012.

[15] G. E. Karniadakis et al. “Physics-informed machine learning”. In: *Nature Reviews Physics* 3.6 (2021), pp. 422–440.

[16] D. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations (ICLR)*. San Diega, CA, USA, 2015.

[17] N. Kovachki et al. “Neural Operator: Learning Maps Between Function Spaces With Applications to PDEs”. In: *Journal of Machine Learning Research* 24.89 (2023), pp. 1–97.

[18] S. Mei, A. Montanari, and P.-M. Nguyen. “A mean field view of the landscape of two-layer neural networks”. In: *Proceedings of the National Academy of Sciences* 115.33 (2018), E7665–E7671.

[19] M. J. D. Powell. *Approximation Theory and Methods*. Cambridge University Press, 1981.

[20] J. Sahs et al. “Shallow Univariate ReLU Networks as Splines: Initialization, Loss Surface, Hessian, and Gradient Flow Dynamics”. In: *Frontiers in Artificial Intelligence* 5 (2022).

[21] A. Shevchenko, V. Kungurtsev, and M. Mondelli. “Mean-field Analysis of Piecewise Linear Solutions for Wide ReLU Networks”. In: *Journal of Machine Learning Research* 23.130 (2022), pp. 1–55.

[22] F. Williams et al. “Gradient Dynamics of Shallow Univariate ReLU Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019.

[23] D. Yarotsky. “Error bounds for approximations with deep ReLU networks”. In: *Neural Networks* 94 (2017), pp. 103–114.