

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

Quantum splines for non-linear approximations

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:

Macaluso, A., Clissa, L., Lodi, S., Sartori, C. (2020). Quantum splines for non-linear approximations [10.1145/3387902.3394032].

Availability:

This version is available at: <https://hdl.handle.net/11585/765841> since: 2020-07-13

Published:

DOI: <http://doi.org/10.1145/3387902.3394032>

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (<https://cris.unibo.it/>).
When citing, please refer to the published version.

(Article begins on next page)

This is the final peer-reviewed accepted manuscript of:

Antonio Macaluso, Luca Clissa, Stefano Lodi, and Claudio Sartori. 2020. Quantum splines for non-linear approximations. In Proceedings of the 17th ACM International Conference on Computing Frontiers (CF '20). Association for Computing Machinery, New York, NY, USA, 249–252

The final published version is available online at:
<https://doi.org/10.1145/3387902.3394032>

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (<https://cris.unibo.it/>)

When citing, please refer to the published version.

QUANTUM SPLINES FOR NON-LINEAR APPROXIMATIONS

Antonio Macaluso

Department of Computer Science and Engineering
University of Bologna, Italy
antonio.macaluso2@unibo.it

Luca Clissa

Department of Physics and Astronomy
University of Bologna, Italy
Istituto Nazionale di Fisica Nucleare (INFN), Italy
luca.clissa2@unibo.it

Stefano Lodi

Department of Computer Science and Engineering
University of Bologna, Italy
stefano.lodi@unibo.it

Claudio Sartori

Department of Computer Science and Engineering
University of Bologna, Italy
claudio.sartori@unibo.it

ABSTRACT

Quantum Computing offers a new paradigm for efficient computing and many AI applications could benefit from its potential boost in performance. However, the main limitation is the constraint to linear operations that hampers the representation of complex relationships in data. In this work, we propose an efficient implementation of quantum splines for non-linear approximation. In particular, we first discuss possible parametrisations, and select the most convenient for exploiting the HHL algorithm to obtain the estimates of spline coefficients. Then, we investigate QSpline performance as an evaluation routine for some of the most popular activation functions adopted in ML. Finally, a detailed comparison with classical alternatives to the HHL is also presented.

Keywords Quantum Computing · Quantum AI · Machine Learning · Neural Networks · HHL

1 Introduction

Quantum Computing (QC) is an emerging field with the potential to revolutionise the world of information technology by leveraging quantum mechanics to endow quantum machines with tremendous computing power. This would enable the solution of problems not feasible to address with classical devices. Of course, these premises are hugely appealing for many real-world applications, especially when coming to the adoption of Machine Learning and Deep Learning models in the domain of Artificial Intelligence. However, the operations on quantum states are required to be linear and unitary under the laws of quantum mechanics [1]. This limitation deprives these techniques of their main strength, i.e., the ability to accurately describe complex non-linear relations.

1.1 Background

Spline functions are smoothing methods suitable for modelling the relationships between variables, typically adopted either as a visual aid in data exploration or for estimation purposes [2]. The underpinning idea is to use linear models in which the input features are augmented with the so-called *basis expansions*. These consist of transformations of the original variables and serve to introduce non-linearity. Technically, splines are constructed by dividing the sample data into sub-intervals delimited by breakpoints, also referred to as knots. A fixed degree polynomial is then fitted in each of the segments, thus resulting in a piecewise polynomial regression. Formally, in the case of a 1-dimensional input vector \mathbf{x} , we can express its relationship with a target variable \mathbf{y} in terms of an order- M spline with knots $\{\xi_k\}_{k=1,\dots,K}$:

$$\mathbf{y}_{n_{\text{obs}} \times 1} = \mathbf{N}_{n_{\text{obs}} \times (M+K)} \boldsymbol{\theta}_{(M+K) \times 1} + \boldsymbol{\epsilon}_{n_{\text{obs}} \times 1}, \quad (1)$$

where θ is the vector of coefficients attached to the basis expansions, n_{obs} is the sample size, ϵ is a random error term and the design matrix N contains $M + K$ basis functions defined as follows:

$$h_j(x) = x^{j-1}, \quad j = 1, \dots, M \quad (2)$$

$$h_{M+k} = (x - \xi_k)_+^{M-1}, \quad k = 1, \dots, K. \quad (3)$$

Notice that the formulation above includes M basis functions that determine the order- M polynomial to be fitted in each segment. The additional K basis introduce continuity constraints on the spline and its derivatives up to order $M - 2$. The goal is then to find the optimal set of parameters θ that minimises the residual sum of squares (RSS), with a *ridge regularisation* of the curvature acting as a roughness penalty:

$$\text{Score}(\theta, \eta) = (\mathbf{y} - N\theta)^T (\mathbf{y} - N\theta) + \eta \theta^T \Omega_{(M+K) \times (M+K)} \theta, \quad (4)$$

where Ω is a diagonal matrix containing the partial second derivatives. The solution to (4) can easily seen to be:

$$\hat{\theta} = (N^T N + \eta \Omega)^{-1} N^T \mathbf{y}. \quad (5)$$

1.2 Contribution

In this work, we demonstrate how to adopt quantum splines for approximating several popular activation functions commonly employed in Neural Networks. In practice, we followed two different strategies. The *hybrid* approach computes quantum estimates of the spline coefficients via the Harrow Hassidim Lloyd (HHL) quantum algorithm. Then a classical device is used to evaluate the activation functions. The *full* quantum approach, instead, takes care of the evaluation process end-to-end, with an additional circuit that reads the HHL estimates and evaluates the function. An in-depth discussion of the HHL algorithm efficiency with respect to classical alternatives is also treated.

1.3 Related Works

One of the major issues for building a quantum Neural Network is the implementation of a non-linear activation function on a quantum system. Indeed, the operations on quantum states are required to be linear and unitary under the laws of quantum mechanics, as discussed in [1]. The most promising attempt in this direction is described in [3], where the authors used the repeat-until-success technique to design a quantum circuit for reproducing a non-linear activation function. The biggest limitation is that this function requires input in the range $[0, \frac{\pi}{2}]$, which is a severe constraint for real-world problems. A step forward was made in [4], where domain restrictions were removed and the activation gate parameters were learned during training.

Regarding matrix inversion on quantum devices, the HHL algorithm [5] is the standard reference. The limitation of using HHL in real applications was discussed in [6]. Therefore, the best way to look at it is as a template for other quantum algorithms. In [7] the authors study the advantage of HHL in electromagnetic scattering cross-section, showing that it can potentially achieve exponential speed-up to arbitrary specified problems.

2 Quantum Spline

The final goal of this work is to investigate non-linear approximation by means of a quantum spline (QSpline). In particular, we want to demonstrate its applicability as an evaluation routine by fitting the spline to some widespread activation functions adopted in Neural Networks. To this end, we consider *relu*, *elu*, *tanh* and *sigmoid* and we use a linear spline with no derivability constraints. Also, no roughness penalisation is applied since we would like to mimic the target function as closely as possible. Regarding the choice of knots, 20 equally spaced breakpoints were selected over the interval $(-1, 1)$.

2.1 Implementation

While the formulation in terms of truncated basis functions described in Section 1.1 is conceptually simple, its numerical and computational properties are not very attractive. For this reason, in practice we adopt a *B-splines* parametrisation [8]. This allows generating a block design matrix where the *sparsity* is constant and depends on the degree of the polynomial fitted in each local interval. Given a sequence of knots $\xi_1, \xi_2, \dots, \xi_K$, we fit a line in each interval $[\xi_k, \xi_{k+1}]_{k=1, \dots, K-1}$ without derivability constraints. In matrix form:

$$\tilde{\mathbf{y}} = S\beta \rightarrow \begin{pmatrix} \tilde{y}_1 \\ \tilde{y}_2 \\ \dots \\ \tilde{y}_K \end{pmatrix} = \begin{pmatrix} S_1 & 0 & \dots & 0 \\ 0 & S_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & S_K \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \dots \\ \beta_K \end{pmatrix}, \quad (6)$$

where \tilde{y}_k contains the activation function evaluations in ξ_k and ξ_{k+1} , β_k s are the spline coefficients and $S_{(2K) \times (2K)}$ is a block diagonal matrix with each block S_k that represents the basis expansions in the k -th interval. Solving the linear system in Eq. (6) requires using HHL to invert the matrix S . Nonetheless, we can exploit the fact that the inverse of a block diagonal matrix is still block diagonal, with the correspondent inverse matrices in each block. This implies we can solve K 2×2 quantum linear systems $S_k |\beta_k\rangle = |\tilde{y}_k\rangle$ instead of a single one for the entire function. This little trick permits to overcome the practical limitations of the available quantum simulators, thus enabling the calculation of the spline coefficients through quantum simulations.

In particular, the computation of the full QSpline is performed in three steps. First, the HHL computes the spline coefficients for the k -th interval:

$$S_k |\beta_k\rangle = |\tilde{y}_k\rangle \xrightarrow{\text{HHL}} |\beta_k\rangle \simeq S_k^{-1} |\tilde{y}_k\rangle. \quad (7)$$

Second, $|\beta_k\rangle$ interacts with the quantum state encoding the input in the k -th interval via quantum interference. The scalar product between $|\beta_k\rangle$ and $|x_{i,k}\rangle$ is computed using the swap-test [9]:

$$|\beta_k\rangle |x_{i,k}\rangle |0\rangle \xrightarrow{\text{swap-test}} |e_1\rangle |e_2\rangle |f_{i,k}\rangle. \quad (8)$$

At this point, the amplitudes of the quantum state $|f_{i,k}\rangle$ embed the estimate of the activation function evaluated in $x_{i,k}$. Third, $|f_{i,k}\rangle$ is measured to get the probability of state $|0\rangle$. This depends on the dot product between β_k and $x_{i,k}$ as follows:

$$|f_{i,k}\rangle = \sqrt{p_0} |0\rangle + \sqrt{p_1} |1\rangle, \quad (9)$$

where:

$$p_0 = \frac{1}{2} + \frac{|\langle \beta_k | x_{i,k} \rangle|^2}{2} = \frac{1}{2} + \frac{|f_{i,k}|^2}{2}. \quad (10)$$

Finally, the activation function estimate in correspondence of $x_{i,k}$ is retrieved by back-transforming Eq. (10) to get $f_{i,k}$.

Notice that, the estimates are intrinsically bounded in the interval $[0, 1]$ since they are encoded as the amplitude of a quantum state. For this reason, the target activation functions are first scaled (i.e. $f_{i,k} \rightarrow f_{i,k}^* \subseteq [0, 1]$). The QSpline is then trained on $f_{i,k}^*$, and the results are finally back-transformed to get the original $f_{i,k}$.

3 Results

The results of hybrid QSpline are illustrated in Fig. 1.

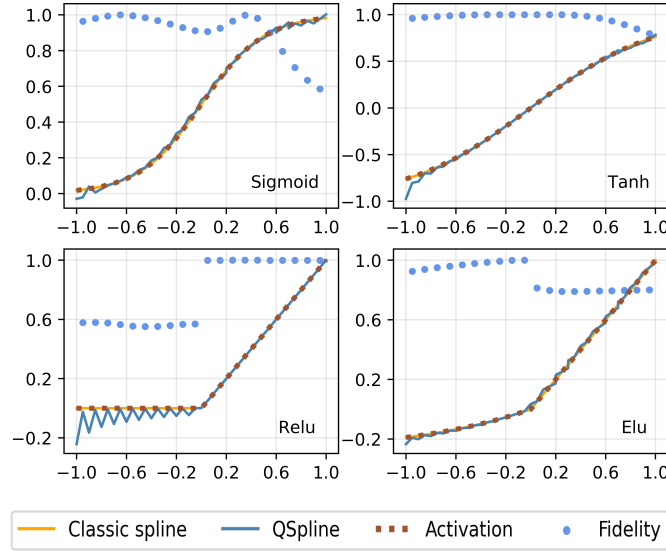


Figure 1: Hybrid QSpline.

The quantum splines perform very well in reproducing the activation curves. The agreement is almost perfect for the sigmoid and the hyperbolic tangent, with slight deviations at the boundaries of the estimation interval. The situation is

similar for Relu and Elu, although apparently some systematic error is introduced in the negative part of the x domain. A possible explanation can be given by looking at the blue dots representing the fidelity of the quantum algorithm, that measures the discrepancy between the HHL output and the real solution of the system. In fact, larger deviations appear in areas where the fidelity is low, thus suggesting the HHL implementation may behave poorly in our setting.

A further attempt was then made with a subset of activation curves exploiting a *full* quantum circuit. The results are reported in Fig. 2. In this case, the goodness of fit is definitely worse, with an almost systematic overestimate of the

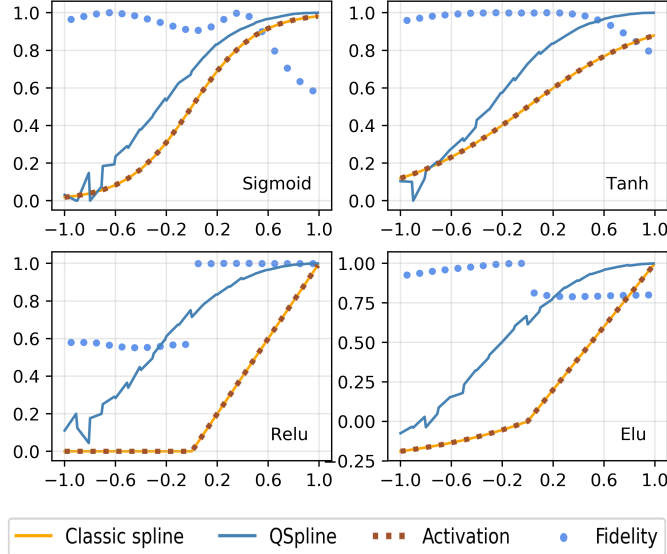


Figure 2: Full QSpline.

target curves. This may be due to the adoption of a second circuit for evaluation on top of the estimation one. Therefore, having two measurement phases introduce more uncertainty which is then propagated and produce a larger error. Nonetheless, notice that the QSpline is likewise capable of reproducing the non-linear behaviour of the curves, which is anyway the most relevant characteristic under investigation. For this reason, we interpret the results as promising, although they necessitate for more tuning. A quantitative assessment of the goodness of fit of both strategies is reported in Table 1.

Activation function	RSS (classic)	RSS (hybrid)	RSS (quantum)	Average Fidelity
Sigmoid	3.3×10^{-5}	0.01	0.75	0.90
Tanh	1.2×10^{-5}	0.06	1.12	0.96
Relu	7.6×10^{-31}	0.14	8.16	0.78
Elu	5.9×10^{-7}	0.12	7.06	0.88

Table 1: Approximation metrics. The table shows the Residual Sum of Squares (RSS) of both quantum and classical splines with respect to the true activation functions. Quantum approaches are indicated as *hybrid* and *quantum*. The average fidelity of the HHL is also reported.

4 Computational Efficiency

Here we discuss in details the computational cost of the quantum estimation phase, also drawing a theoretical comparison with the classical baselines. In general, matrix inversion can be accomplished in polynomial time on classical devices [10, 11, 12]. However, when several favourable assumptions hold it is possible to reduce computational costs. In particular, the *Conjugate Gradient* algorithm [13, 14] allows solving a linear system with a complexity equal to

$\mathcal{O}(s\sqrt{\kappa n}/\log(\epsilon))$, where n is the system size, κ is the system condition number, s the matrix sparsity (i.e. the maximum number of non-zero matrix elements of \mathbf{A} in any given row or column), and ϵ is the desired error tolerance.

The reference quantum technique is the HHL algorithm. HHL is a method for approximately preparing a quantum superposition of the form $|x\rangle$, where x is the solution to a linear system $\mathbf{A}x = b$, \mathbf{A} is a hermitian design matrix and b is encoded in amplitudes of $|b\rangle$. From a computational point of view, this requires an amount of time that grows roughly like $\mathcal{O}(s^2\kappa^2\log(n)/\epsilon)$ (cfr. Table 2 for a comparison between HHL and classical counterparts). The algorithm scales logarithmically with respect to the size of the matrix, which means it has an exponential advantage when compared to classical alternatives. However, its complexity is polynomial in s and κ , which means we have to introduce constraints on the condition number and the sparsity not to destroy the computational advantage of the HHL. This makes the previous comparison unfair since we cannot make assumptions about the design matrix in general.

Gauss Jordan elimination	Strassen	Coppersmith	Conjugate Gradient	HHL
$\mathcal{O}(n^3)$	$\mathcal{O}(n^{2.8})$	$\mathcal{O}(n^{2.37})$	$\mathcal{O}(sn\sqrt{\kappa}/\log(\epsilon))$	$\mathcal{O}(s^2\kappa^2\log(n)/\epsilon)$

Table 2: Comparison of algorithms computational costs.

In our case, we can observe that an order- M penalised spline has very desirable properties. First of all, the ridge penalisation described in Eq. (5) guarantees a lower condition number compared to the unconstrained design matrix. Specifically, Casella [15] showed that the higher the penalisation, the lower the condition number is. In fact, the condition number of a matrix, $\kappa(\mathbf{A})$, is equal to $\frac{|\lambda_{\max}(\mathbf{A})+\eta|}{|\lambda_{\min}(\mathbf{A})+\eta|}$, where λ_{\min} and λ_{\max} are the smallest and the biggest eigenvalues respectively, and η is the amount of penalisation. Clearly, the more η increases, the more it dominates the fraction, eventually tending to 1 for large penalisations. Furthermore, regarding splines as piecewise polynomials defined on contiguous segments allows building a sparse design matrix \mathbf{A} , whose blocks describe the approximation in the corresponding intervals. This implies that the sparsity amounts to the number of basis functions, $M + K$. In light of the two properties above, splines are an ideal setting for an efficient application of the HHL algorithm.

Figure 3 illustrates a theoretical comparison of HHL computational costs with respect to the classical counterparts. The

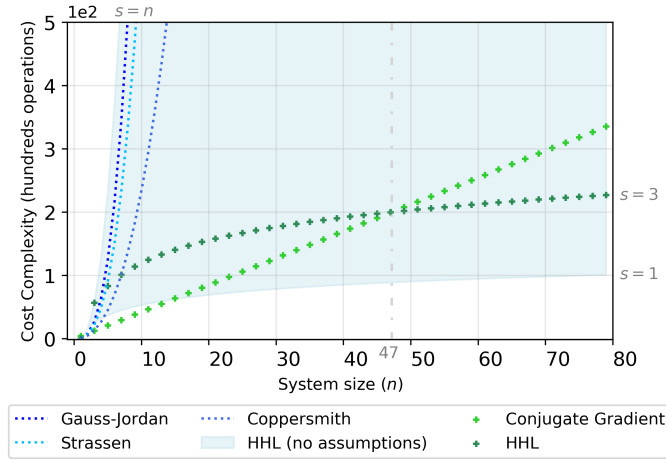


Figure 3: Cost complexity as a function of the size, n , of the \mathbf{A} matrix. The green curves represent HHL and Conjugate Gradient for fixed s, κ , while the blue ones are referred to matrix inversion with no assumptions. The light blue shaded area illustrates the performance of HHL as the sparsity varies (κ fixed).

green curves illustrate the computational costs (in hundreds of operations) of HHL and Conjugate Gradient for fixed sparsity and condition number. In particular, their values were chosen to reflect the case of natural splines with no intercept coefficient ($s = 3$) and an approximately well-conditioned data matrix ($\kappa = 2$). The HHL outperforms the Conjugate Gradient as the number of features becomes larger than 47, that is quite frequent in *Big Data* and *Artificial Intelligence* applications, e.g. bioinformatics, natural language processing and computer vision. A comparison of HHL with matrix inversion algorithms is also conducted when no assumptions are made. The light blue shaded area

depicts the performance of HHL as s varies (κ fixed to 2), while the curves using the blue palette describe classical alternatives. The advantage of HHL is evident as soon as some sparsity is introduced. However, we have also to take into account κ and the higher costs of quantum computation to draw more solid conclusions. Thus the efficiency boost due to the quantum technologies may foster the use of HHL for novel, classically unfeasible applications. For instance, the computational burden of spline functions could be mitigated in this way, paving the way for future studies aimed at performing splines per se (e.g. multidimensional splines), and not just as a mere tool for evaluating non-linear functions.

5 Conclusion

In this work, we demonstrated the adoption of splines for approximating popular activation functions on a quantum device. The preliminary results were promising, although some tuning of the HHL implementation and the circuit for the full quantum approach is required. The quantum spline was able to reproduce the non-linearity of the curves, thus candidating this approach as a building block in the development of Quantum Neural Networks.

Also, we compared the computational complexity of the HHL algorithm against the most adopted classical counterparts, showing why splines may be an ideal setting for leveraging its advantages.

Future studies will be dedicated to improving the full quantum approach, with the possibility of developing a novel and more stable implementation of the HHL routine.

References

- [1] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.
- [2] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.
- [3] Yudong Cao, Gian Giacomo Guerreschi, and Alán Aspuru-Guzik. Quantum neuron: an elementary building block for machine learning on quantum computers. *arXiv preprint arXiv:1711.11240*, 2017.
- [4] Wei Hu. Towards a real quantum neuron. *Natural Science*, 10(3):99–109, 2018.
- [5] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.
- [6] Scott Aaronson. Read the fine print. *Nature Physics*, 11(4):291, 2015.
- [7] B David Clader, Bryan C Jacobs, and Chad R Sprouse. Preconditioned quantum linear system algorithm. *Physical review letters*, 110(25):250504, 2013.
- [8] Carl De Boor, Carl De Boor, Etats-Unis Mathématicien, Carl De Boor, and Carl De Boor. *A practical guide to splines*, volume 27. springer-verlag New York, 1978.
- [9] Harry Buhrman, Richard Cleve, John Watrous, and Ronald de Wolf. Quantum fingerprinting. *Phys. Rev. Lett.*, 87:167902, Sep 2001.
- [10] Peter Bürgisser, Michael Clausen, and Mohammad A Shokrollahi. *Algebraic complexity theory*, volume 315. Springer Science & Business Media, 2013.
- [11] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 1–6, 1987.
- [12] Volker Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969.
- [13] Carlo Ciliberto, Mark Herbster, Alessandro Davide Ialongo, Massimiliano Pontil, Andrea Rocchetto, Simone Severini, and Leonard Wossnig. Quantum machine learning: a classical perspective. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2209):20170551, 2018.
- [14] Jonathan Richard Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain, 1994.
- [15] George Casella. Condition numbers and minimax ridge regression estimators. *Journal of the American Statistical Association*, 80(391):753–758, 1985.