Efficient Error Certification for Physics-Informed Neural Networks

Francisco Eiras [1] **Adel Bibi** [1] **Rudy Bunel** [2] **Krishnamurthy Dj Dvijotham** [2] **Philip H.S. Torr** [1]
M. Pawan Kumar [2]

Abstract

Recent work provides promising evidence that Physics-Informed Neural Networks (PINN) can efficiently solve partial differential equations (PDE). However, previous works have failed to provide guarantees on the *worst-case* residual error of a PINN across the spatio-temporal domain – a measure akin to the tolerance of numerical solvers – focusing instead on point-wise comparisons between their solution and the ones obtained by a solver on a set of inputs. In real-world applications, one cannot consider tests on a finite set of points to be sufficient grounds for deployment, as the performance could be substantially worse on a different set. To alleviate this issue, we establish guaranteed error-based conditions for PINNs over their *continuous* applicability domain. To verify the extent to which they hold, we introduce ∂-CROWN: a general, efficient and scalable post-training framework to bound PINN residual errors. We demonstrate its effectiveness in obtaining tight certificates by applying it to two classically studied PINNs – Burgers' and Schrödinger's equations –, and two more challenging ones with real-world applications – the Allan-Cahn and Diffusion-Sorption equations.

1. Introduction

Accurately predicting the evolution of complex systems through simulation is a difficult, yet necessary, process in the physical sciences. Many of these systems are represented by partial differential equations (PDE) the solutions of which, while well understood, pose a major computational challenge to solve at an appropriate spatio-temporal resolution (Raissi et al., 2019a; Kochkov et al., 2020). Inspired by the success of machine learning in other domains, recent work has attempted to overcome the aforementioned challenge through *physics-informed neural networks* (PINN) (Raissi et al., 2019a; Sun et al., 2020; Pang et al., 2019). For example, the Diffusion-Sorption equation – which has real-world applications in the modeling of groundwater contaminant transport – takes 59.83s to solve per inference point using a classical PDE solver, while inference in its PINN version from Takamoto et al. (2022) takes only 2.7×10^{-3}s, a speed-up of more than 10^4 times.

The parameters of a PINN are estimated by minimizing the residual of the given PDE, together with its initial and boundary conditions, over a set of spatio-temporal training inputs. Its accuracy is then empirically estimated by measuring the solution estimate over a set of discrete input points, and (typically) comparing them to numerical PDE solvers. In other words, most current work on PINNs provides no certified error bounds applicable for *every* input within the domain of the PDE.

While testing on a finite set of points provides a good initial signal on the accuracy of the PINN, such an approach cannot be relied upon in practice, and error certification is needed to understand the quality of the PINN trained (Hillebrecht & Unger, 2022). For example, by estimating the maximum residual error of the Diffusion-Sorption PINN from Takamoto et al. (2022) using 10^4 Monte Carlo samples across the domain we obtain an estimate of 1.1×10^{-3}, whereas the estimate using 10^6 samples is 21.09 – indicating the PINN has failed to learn a continuous function that correctly maps to the solution of the underlying PDE. This empirical difference shows the need for computing certified error bounds to avoid deploying poorly trained PINNs.

We introduce formal, error-based *correctness* conditions for PINNs which require that the residual error is *globally* upper bounded by a tolerance parameter, that is, that the continuous function learned approximates the underlying PDE solution across the domain. To compute this bound and verify the correctness conditions, we build on recent progress in neural network verification. Specifically, we efficiently extend the CROWN framework (Zhang et al., 2018) by deriving linear upper and lower bounds for the various nonlinear terms that appear in PINNs, and devise a novel bound propagation strategy for the task at hand.

Our contributions are threefold. **(i)** We formally define

[1]University of Oxford [2]Google DeepMind. Correspondence to: FE <eiras@robots.ox.ac.uk>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

correctness conditions for general PINNs that approximate continuous solutions of PDEs. **(ii)** We introduce a general, efficient, and scalable post-training *error certification framework* (∂-CROWN) to theoretically verify PINNs over their entire spatio-temporal domains. **(iii)** We demonstrate our post-training framework on two widely studied PDEs in the context of PINNs, Burgers' and Schrödinger's equations (Raissi et al., 2019a), and two more challenging ones with real-world applications, the Allan-Cahn equation (Monaco & Apiletti, 2023) and the Diffusion-Sorption equation (Takamoto et al., 2022).

2. Related work

Since our certification framework for PINNs is based on the verification literature of image classifiers, in this section we explore: related work for PINNs, and previous work on neural network robustness verification.

Physics-informed Neural Networks. Raissi et al. (2019a) introduced PINNs, which leverage automatic differentiation to obtain approximate solutions to the underlying PDE. Since then, a variety of different PINNs have emerged in a range of applications, from fluid dynamics (Raissi et al., 2019b; 2020; Sun et al., 2020; Jin et al., 2021), to meta material design (Liu & Wang, 2019; Fang & Zhan, 2019a; Chen et al., 2020) for different classes of PDEs (Pang et al., 2019; Fang & Zhan, 2019b; Zhang et al., 2020). A few works analyze the convergence of the training process of PINNs under specific conditions (Shin et al., 2020; Wang et al., 2022b). Mishra & Molinaro (2022) approximated the generalization error of various PINNs under specific stability and training process assumptions, and others introduced approximation bounds under regularity assumptions (Ryck & Mishra, 2022; Hillebrecht & Unger, 2022). Our verification framework is applicable to any PINN where the solution is modeled by a fully connected network.

Robustness Verification of Neural Networks. The presence of adversarial examples, *i.e.*, small local input perturbations that lead to large output changes, was established by Szegedy et al. (2013) in image classifiers. As robust classifiers emerged (Madry et al., 2017), so did attempts to certify them formally. Those methods can be divided into *exact*, *i.e.*, complete (Katz et al., 2017; Ehlers, 2017; Huang et al., 2017; Lomuscio & Maganti, 2017; Bunel et al., 2018; De Palma et al., 2021; Ferrari et al., 2022), or *conservative*, *i.e.*, sound but incomplete (Gowal et al., 2018; Mirman et al., 2018; Wang et al., 2018; Wong & Kolter, 2018; Ayers et al., 2020). A promising set of conservative methods poses the problem as a convex relaxation of the original nonlinear network architecture, and solves it using a linear programming solver (Salman et al., 2019; Zhang et al., 2022) or by obtaining closed-form bounds (Zhang et al., 2018; Wang et al., 2021). The latter are especially appealing due to their efficiency. Examples include CROWN (Zhang et al., 2018) and α-CROWN (Xu et al., 2020b). Xu et al. (2020a) extended the linear relaxation framework from Zhang et al. (2018) to general computation graphs, but the purely backward propagation nature makes it potentially less efficient than custom bounds/hybrid approaches (Shi et al., 2020). Our work adapts techniques from verification to certify the *full* applicability domain of PINNs, in a similar fashion to the *global* specification from Müller et al. (2023).

3. Preliminaries

3.1. Notation

Given vector $\mathbf{a} \in \mathsf{R}^d$, $\mathbf{a}_i$ refers to its i-th component. We use $\partial_{\mathbf{x}_i^j} f$ and $\frac{\partial^j f}{(\partial \mathbf{x}_i)^j}$ interchangeably to refer to the j-th partial derivative of a function $f : \mathsf{R}^n \to \mathsf{R}$ with respect to the i-component of its input, $\mathbf{x}_i$. Where it is clear, we use $f(\mathbf{x})$ and f interchangeably. We take $\mathsf{L}^{(i)}_{\mathbf{W},\mathbf{b}}(\mathbf{x}) = \mathbf{W}^{(i)}\mathbf{x} + \mathbf{b}^{(i)}$ to be a function of $\mathbf{x}$ parameterized by weights $\mathbf{W}^{(i)}$ and bias $\mathbf{b}^{(i)}$. We define an L-layer *fully connected neural network* $g : \mathsf{R}^{d_0} \to \mathsf{R}^{d_L}$ for an input $\mathbf{x}$ as $g(\mathbf{x}) = y^{(L)}(\mathbf{x})$ where $y^{(k)}(\mathbf{x}) = \mathsf{L}^{(k)}_{\mathbf{W},\mathbf{b}}(z^{(k-1)}(\mathbf{x}))$, $z^{(k-1)}(\mathbf{x}) = \sigma(y^{(k-1)}(\mathbf{x}))$, $z^{(0)}(\mathbf{x}) = \mathbf{x}$, in which $\mathbf{W}^{(k)} \in \mathsf{R}^{d_k \times d_{k-1}}$ and $\mathbf{b}^{(k)} \in \mathsf{R}^{d_k}$ are the weight and bias of layer k, σ is the nonlinear activation, and $k \in \{1, \dots, L\}$.

3.2. Physics-informed neural networks (PINNs)

We consider general nonlinear PDEs of the form:

$$f(t, \hat{\mathbf{x}}) = \partial_t u(t, \hat{\mathbf{x}}) + \mathcal{N}[u](t, \hat{\mathbf{x}}) = 0, \ \hat{\mathbf{x}} \in \mathcal{D}, t \in [0, T], \tag{1}$$

where f is the residual of the PDE, t is the temporal and $\hat{\mathbf{x}}$ is the spatial components of the input, $u : [0, T] \times \mathcal{D} \to \mathsf{R}$ is the solution, $\mathcal{N}$ is a nonlinear differential operator on u, $T \in \mathsf{R}^+$, and $\mathcal{D} \subset \mathsf{R}^D$. Where possible, for conciseness we will use $\mathbf{x} = (t, \hat{\mathbf{x}})$, for $\mathbf{x} \in \mathcal{C} = [0, T] \times \mathcal{D}$, with $\mathbf{x}_0 = t$.

We assume f is the residual of an R^{th} order PDE where the differential operators of $\mathcal{N}$ applied to u yield the partial derivatives for order $\{0, ..., R\}$ as: $u \in \mathcal{N}^{(0)}$, $\partial_{\mathbf{x}_i} u \in \mathcal{N}^{(1)}$, $\partial_{\mathbf{x}_i^2} u \in \mathcal{N}^{(2)}$, $\dots$, $\partial_{\mathbf{x}_i^R} u \in \mathcal{N}^{(R)}$ for $i \in \{0, \dots, D\}$[1]. With these, we can re-write $f = \mathcal{P}(u, \partial_{\mathbf{x}_0} u, \dots, \partial_{\mathbf{x}_D} u, \dots, \partial_{\mathbf{x}_D^R} u)$, where $\mathcal{P}$ is a nonlinear function of those terms. Furthermore, the PDE is defined under (1) initial conditions, *i.e.*, $u(0, \hat{\mathbf{x}}) = u_0(\hat{\mathbf{x}})$, for $\hat{\mathbf{x}} \in \mathcal{D}$, and (2) general Robin boundary conditions, *i.e.*, $au(t, \hat{\mathbf{x}}) + b\partial_{\mathbf{n}} u(t, \hat{\mathbf{x}}) = u_b(t, \hat{\mathbf{x}})$ for $a, b \in \mathsf{R}$, $\hat{\mathbf{x}} \in \delta\mathcal{D}$ and $t \in [0, T]$, and $\partial_{\mathbf{n}} u$ is the normal derivative at the border with respect to some components of $\hat{\mathbf{x}}$.

[1]For simplicity, we assume $\mathcal{N}$ does not contain any cross-derivative operators, yet an extension would be trivial to derive.

Continuous-time PINNs (Raissi et al., 2019a) result from approximating the solution, $u(\mathbf{x})$, using a neural network parameterized by θ, $u_\theta(\mathbf{x})$. We refer to this network as the *approximate solution*. In that context, the *physics-informed neural network* (or residual) is $f_\theta(\mathbf{x}) = \partial_t u_\theta(\mathbf{x}) + \mathcal{N}[u_\theta](\mathbf{x})$. For example, the one-dimensional Burgers' equation (explored in detail in Section 5) is defined as:

$$f_\theta(\mathbf{x}) = \partial_t u_\theta(\mathbf{x}) + u_\theta(\mathbf{x})\partial_x u_\theta(\mathbf{x}) - (0.01/\pi)\partial_{x^2} u_\theta(\mathbf{x}). \tag{2}$$

Note f_θ has the same order as f, and can be described similarly as a nonlinear function with the partial derivatives applied to u_θ instead of u. Burgers' equation (from above) has one 0^{th} order term (u_θ), two 1^{st} order ones ($\partial_t u_\theta$ and $\partial_x u_\theta$), and a 2^{nd} order partial derivative ($\partial_{x^2} u_\theta$), while $u_\theta(\mathbf{x})\partial_x u_\theta(\mathbf{x})$ is a nonlinear term of the f_θ polynomial.

3.3. Bounding neural network outputs using CROWN (Zhang et al., 2018)

The computation of upper/lower bounds on the output of neural networks over a domain has been widely studied within verification of image classifiers (Katz et al., 2017; Mirman et al., 2018; Zhang et al., 2018). For the sake of computational efficiency, we consider the bounds obtained using CROWN (Zhang et al., 2018)/α-CROWN (Xu et al., 2020b) as the base for our framework.

Take g to be the fully connected neural network (as defined in Section 3.1) we're interested in bounding. The goal is to compute $\max / \min_{x\in\mathcal{C}} g(\mathbf{x})$, where $\mathcal{C}$ is the applicability domain. Typically within verification of image classifiers, $\mathcal{C} = \mathsf{B}^p_{\mathbf{x},\epsilon} = \{\mathbf{x}' : \|\mathbf{x}' - \mathbf{x}\|_p \leq \epsilon\}$, *i.e.*, it is a *local* ℓ_p-ball of radius ϵ around an input $\mathbf{x}$ from the test set.

CROWN solves the optimization problem by *back-propagating* linear bounds of $g(\mathbf{x})$ through each hidden layer of the network until the input is reached. To do so, assuming constant bounds on $y^{(k)}(\mathbf{x})$ are known for $\mathbf{x} \in \mathcal{C}$, *i.e.*, $\forall \mathbf{x} \in \mathcal{C} : y^{(k),L} \leq y^{(k)}(\mathbf{x}) \leq y^{(k),U}$, CROWN relaxes the nonlinearities of each $z^{(k)}$ using a linear lower and upper bound approximation that contains the full possible range of $\sigma(y^{(k)}(\mathbf{x}))$. By relaxing the activations of each layer and back-propagating it through $z^{(k)}$, CROWN obtains a bound on each $y^{(k)}$ as a function of $y^{(k-1)}$. Back-substituting from the output $y^{(L)} = g(\mathbf{x})$ until the input $\mathbf{x}$ results in:

$$\min_{\mathbf{x}\in\mathcal{C}} g(\mathbf{x}) \geq \min_{\mathbf{x}\in\mathcal{C}} \mathbf{A}^L\mathbf{x}+\mathbf{a}^L, \ \max_{\mathbf{x}\in\mathcal{C}} g(\mathbf{x}) \leq \max_{\mathbf{x}\in\mathcal{C}} \mathbf{A}^U\mathbf{x}+\mathbf{a}^U,$$

where $\mathbf{A}^L$, $\mathbf{a}^L$, $\mathbf{A}^U$ and $\mathbf{a}^U$ are computed in polynomial time from $\mathbf{W}^{(k)}, \mathbf{b}^{(k)}$, and the linear relaxation parameters. The solution to the optimization problems above given simple constraints $\mathcal{C}$ can be obtained in closed-form. α-CROWN (Xu et al., 2020b) improves these bounds by optimizing the linear relaxations of $\sigma(y^{(k)}(\mathbf{x}))$ for tightness.

4. ∂-CROWN: Error Certification for General Physics-Informed Neural Networks

Take u_θ to be the learned approximate continuous solution of the PDE f through the PINN f_θ. Previous works deriving from Raissi et al. (2019a) have measured the *correctness* of u_θ empirically by computing the solution error at a set of discrete point compared to that obtained via numerical solvers for f (Takamoto et al., 2022; Monaco & Apiletti, 2023) – a compromise arising from the fact we cannot bound $\|u_\theta - u\|$ for general PDEs across their continuous domain.

To mitigate this issue for continuous-time PINNs, we approach the problem of error bounding by imposing correctness conditions on the *residual* instead of the solution error. By definition, u_θ is a correct solution to the PINN f_θ if 3 conditions are met: ① the norm of the solution error with respect to the initial condition is upper bounded by an acceptable tolerance, ② the norm of the solution error with respect to the boundary conditions is bounded by an acceptable tolerance, and ③ the norm of the residual is bounded by an acceptable tolerance. We define these as PINN *correctness conditions*, and formalize it in Definition 1. Note these conditions are general and, at this point, no assumptions are made about u_θ or the PDE.

Definition 1 (Correctness Conditions for PINNs)**.** *$u_\theta : [0,T]\times\mathcal{D} \to \mathsf{R}$ is a $\delta_0, \delta_b, \varepsilon$-globally correct approximation of the exact solution $u : [0,T] \times \mathcal{D} \to \mathsf{R}$ if:*

① $\max_{\hat{\mathbf{x}}\in\mathcal{D}} |u_\theta(0,\hat{\mathbf{x}}) - u_0(\hat{\mathbf{x}})|^2 \leq \delta_0$,

② $\max_{t\in[0,T],\hat{\mathbf{x}}\in\delta\mathcal{D}} |au_\theta(\mathbf{x}) + b\partial_{\mathbf{n}} u_\theta(\mathbf{x}) - u_b(\mathbf{x})|^2 \leq \delta_b$,

③ $\max_{\mathbf{x}\in\mathcal{C}} |f_\theta(\mathbf{x})|^2 \leq \varepsilon$.

In practice, δ_0, δ_b, and ε correspond to tolerances similar to the ones given by numerical solvers for f. While the residual error upper bound is similar in nature to the empirical errors used to monitor convergence in iterative solvers (e.g., in Krylov subspace methods for linear systems), the bound proposed here corresponds to the error of the continuous approximate solution u_θ instead of the discretized version provided in those solvers. In Section 5, we empirically analyze the connection between residual and solution errors using a numerical solver.

The verification of the conditions from Definition 1 requires bounding: a linear function of u_θ for ①, a linear function of u_θ and $\partial_{\mathbf{n}} u_\theta$ for ②, and the PINN output, f_θ, in ③. To achieve ①, assuming u_θ is a standard fully connected neural network as in Raissi et al. (2019a), we can directly use CROWN/α-CROWN (Zhang et al., 2018; Xu et al., 2020b). However, bounding ② and ③ with a linear function in $\mathbf{x}$ efficiently requires a method to bound linear and nonlinear functions of the partial derivatives of u_θ.

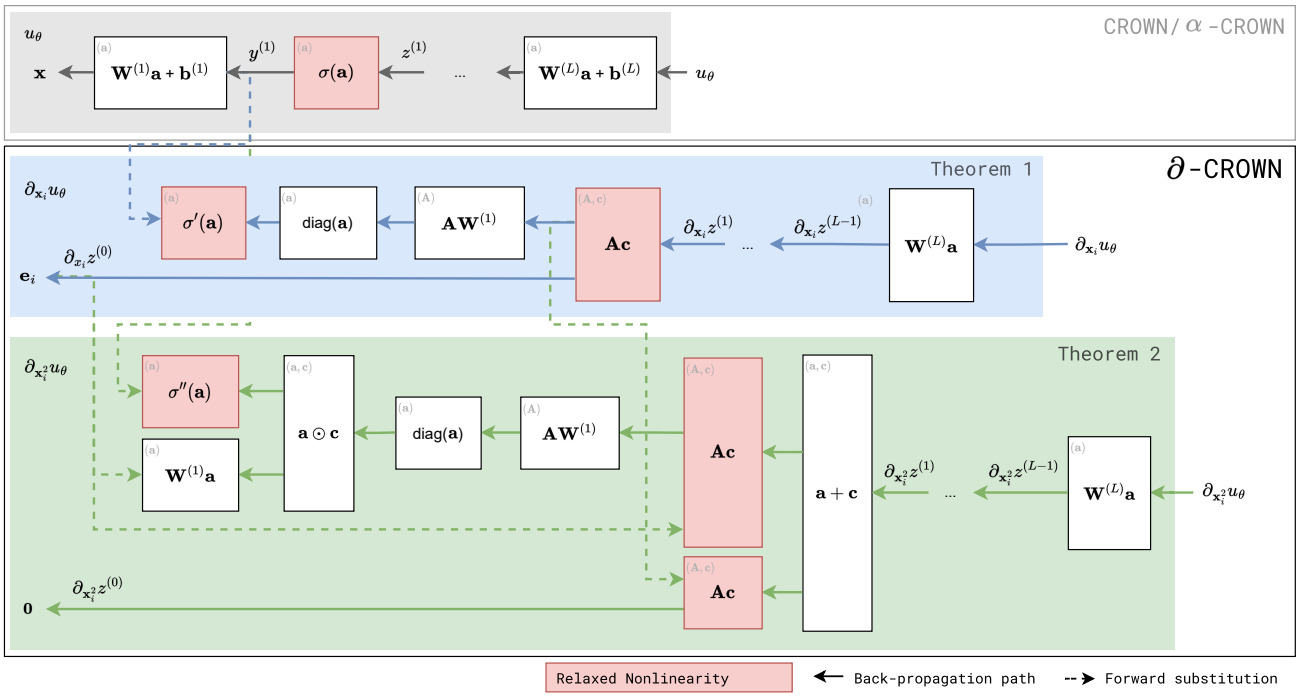


Figure 1: **Bounding Partial Derivatives with ∂-CROWN**: our hybrid scheme for bounding $\partial_{\mathbf{x}_i} u_\theta$ and $\partial_{\mathbf{x}_i^2} u_\theta$ uses back-propagation and forward substitution (inspired by Shi et al. (2020)) to compute bounds in $\mathcal{O}(L)$ instead of the $\mathcal{O}(L^2)$ complexity of full back-propagation as in Xu et al. (2020a).

We propose ∂-CROWN, an efficient framework to: (i) compute closed-form bounds on the partial derivatives of an arbitrary fully-connected network u_θ (Section 4.1), and (ii) bound a nonlinear function of those partial derivative terms, *i.e.*, f_θ (Section 4.2). Throughout this section, we assume $u_\theta(\mathbf{x}) = g(\mathbf{x})$ as defined in Section 3.1, with $d_0 = D + 1$. Formal statements and proofs for the lemmas and theorems presented in this section are in Appendix D.

4.1. Bounding Partial Derivatives of u_θ

The computation of the bounds for the 0^{th} order derivative, *i.e.*, u_θ, and intermediate pre-activations can be computed using CROWN/α-CROWN (Zhang et al., 2018; Xu et al., 2020b). As such, for what follows, we assume that for $\mathbf{x} \in \mathcal{C}$, both the bounds on u_θ and $y^{(k)}$, $\forall k$ are given.

Assumption 1. *The pre-activation layer outputs of u_θ, $y^{(k)} = \mathsf{L}^{(k)}_{\mathbf{W},\mathbf{b}}(z^{(k-1)})$, are lower and upper bounded by linear functions $\mathsf{L}^{(k),L}_{\mathbf{A},\mathbf{a}}(\mathbf{x}) \leq y^{(k)} \leq \mathsf{L}^{(k),U}_{\mathbf{A},\mathbf{a}}(\mathbf{x})$. Moreover, for $\mathbf{x} \in \mathcal{C}$, we have $y^{(k),L} \leq y^{(k)} \leq y^{(k),U}$.*

Note that using CROWN/α-CROWN, $\mathbf{A}^{(k),L}$, $\mathbf{a}^{(k),L}$, $\mathbf{A}^{(k),U}$, $\mathbf{a}^{(k),U}$ are functions of all the previous layers' parameters. For 1^{st} order derivatives, we start by explicitly obtaining the expression of $\partial_{\mathbf{x}_i} u_\theta$.

Lemma 1 (Expression for $\partial_{\mathbf{x}_i} u_\theta$)**.** *For $i \in \{1, \dots, d_0\}$, the partial derivative of u_θ with respect to $\mathbf{x}_i$ can be computed recursively as $\partial_{\mathbf{x}_i} u_\theta = \mathbf{W}^{(L)} \partial_{\mathbf{x}_i} z^{(L-1)}$ for:*

$$\partial_{\mathbf{x}_i} z^{(k)} = \partial_{z^{(k-1)}} z^{(k)} \partial_{\mathbf{x}_i} z^{(k-1)}, \quad \partial_{\mathbf{x}_i} z^{(0)} = \mathbf{e}_i,$$

for $k \in \{1, \dots, L-1\}$, and where $\partial_{z^{(k-1)}} z^{(k)} = \mathrm{diag}\left[\sigma'\left(y^{(k)}\right)\right] \mathbf{W}^{(k)}$.

Using this result, we can efficiently linearly lower and upper bound $\partial_{\mathbf{x}_i} u_\theta$.

Theorem 1 (Informal, ∂-CROWN Linear Bounding $\partial_{\mathbf{x}_i} u_\theta$)**.** *There exist two linear functions $\partial_{\mathbf{x}_i} u^U_\theta$ and $\partial_{\mathbf{x}_i} u^L_\theta$ such that, $\forall \mathbf{x} \in \mathcal{C}$ it holds that $\partial_{\mathbf{x}_i} u^L_\theta \leq \partial_{\mathbf{x}_i} u_\theta \leq \partial_{\mathbf{x}_i} u^U_\theta$, where the linear coefficients can be computed recursively in closed-form in $\mathcal{O}(L)$ time as a function of $\mathbf{W}^{(k)}$, $\mathbf{A}^{(k),L}$, $\mathbf{a}^{(k),L}$, $\mathbf{A}^{(k),U}$, $\mathbf{a}^{(k),U}$, $\mathbf{y}^{(k),L}$, and $\mathbf{y}^{(k),U}$.*

The formal statement of Theorem 1 and expressions for $\partial_{\mathbf{x}_i} u^L_\theta$ and $\partial_{\mathbf{x}_i} u^U_\theta$ are provided in Appendix D.3. Note that this bound is not computed using fully backward propagation as in Xu et al. (2020a). Instead we use a *hybrid* scheme in the spirit of Shi et al. (2020) for the sake of efficiency. We perform backward propagation to compute $\partial_{z^{(k-1)}} z^{(k)}$ as a function of $y^{(k)}$, and forward-substitute the pre-computed CROWN bounds $\mathsf{L}^{(k),L}_{\mathbf{A},\mathbf{a}}(\mathbf{x}) \leq y^{(k)} \leq \mathsf{L}^{(k),U}_{\mathbf{A},\mathbf{a}}(\mathbf{x})$ at that point instead of fully backward propagating which would have $\mathcal{O}(L^2)$ complexity. This induces a significant speed-up while achieving tight enough bounds. Figure 1 showcases the back-propagation and forward substitution paths for

bounding $\partial_{\mathbf{x}_i} u_\theta$ in blue. Similarly to CROWN with the activation σ, this bound requires relaxing $\sigma'(y^{(k)})$.

Similarly, we can linearly bound $\partial_{\mathbf{x}_i^2} u_\theta$, a requirement to bound f_θ in 2^{nd} order PINNs.

Lemma 2 (Expression for $\partial_{\mathbf{x}_i^2} u_\theta(\mathbf{x})$)**.** *For $i \in \{1, \dots, d_0\}$, the second partial derivative of u_θ with respect to $\mathbf{x}_i$ can be computed recursively as $\partial_{\mathbf{x}_i^2} u_\theta = \mathbf{W}^{(L)} \partial_{\mathbf{x}_i^2} z^{(L-1)}$ where:*

$$\partial_{\mathbf{x}_i^2} z^{(k)} = \partial_{x_i z^{(k\ \ 1)}} z^{(k)} \partial_{\mathbf{x}_i} z^{(k-1)} + \partial_{z^{(k\ \ 1)}} z^{(k)} \partial_{\mathbf{x}_i^2} z^{(k-1)},$$

and $\partial_{\mathbf{x}_i^2} z^{(0)} = \mathbf{0}$, for $k \in \{1, \dots, L-1\}$, with $\partial_{\mathbf{x}_i} z^{(k-1)}$ and $\partial_{z^{(k\ \ 1)}} z^{(k)}$ as per in Lemma 1, and $\partial_{x_i z^{(k\ \ 1)}} z^{(k)} = \text{diag}\left[\sigma''\left(y^{(k)}\right)\left(\mathbf{W}^{(k)} \partial_{\mathbf{x}_i} z^{(k-1)}\right)\right] \mathbf{W}^{(k)}$.

Theorem 2 (Informal, ∂-CROWN Linear Bounding $\partial_{\mathbf{x}_i^2} u_\theta$)**.** *Assume that through a previous bounding of $\partial_{\mathbf{x}_i} u_\theta$, we have linear lower and upper bounds on $\partial_{\mathbf{x}_i} z^{(k-1)}$ and $\partial_{z^{(k\ \ 1)}} z^{(k)}$. There exist two linear functions $\partial_{\mathbf{x}_i^2} u_\theta^U$ and $\partial_{\mathbf{x}_i^2} u_\theta^L$ such that, $\forall \mathbf{x} \in \mathcal{C}$ it holds that $\partial_{\mathbf{x}_i^2} u_\theta^L \leq \partial_{\mathbf{x}_i^2} u_\theta \leq \partial_{\mathbf{x}_i^2} u_\theta^U$, where the linear coefficients can be computed recursively in closed-form in $\mathcal{O}(L)$ time as a function of $\mathbf{W}^{(k)}$, $\mathbf{A}^{(k),L}$, $\mathbf{a}^{(k),L}$, $\mathbf{A}^{(k),U}$, $\mathbf{a}^{(k),U}$, $\mathbf{y}^{(k),L}$, $\mathbf{y}^{(k),U}$, and the parameters of the linear lower and upper bounds on $\partial_{\mathbf{x}_i} z^{(k-1)}$ and $\partial_{z^{(k\ \ 1)}} z^{(k)}$.*

The formal statement of Theorem 2 and expressions for $\partial_{\mathbf{x}_i^2} u_\theta^L$ and $\partial_{\mathbf{x}_i^2} u_\theta^U$ are in Appendix D.4. As with the first derivative, this bound requires a relaxation of $\sigma''(y^{(k)})$. Note that this also follows a hybrid computation scheme, with the back-propagation and forward substitution paths for bounding $\partial_{\mathbf{x}_i^2} u_\theta$ computations shown in green in Figure 1.

Assuming $\mathcal{C} = \{\mathbf{x} \in \mathsf{R}^{d_0} : \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U\}$, we can obtain closed-form expressions for constant global bounds on the linear functions $\partial_{\mathbf{x}_i} u_\theta^U$, $\partial_{\mathbf{x}_i} u_\theta^L$, $\partial_{\mathbf{x}_i^2} u_\theta^U$, $\partial_{\mathbf{x}_i^2} u_\theta^L$, which we formulate and prove in Appendix D.5. While here we only compute the expression for the second derivative with respect to the same input, it would be trivial to extend it to cross derivatives (*i.e.*, $\partial_{\mathbf{x}_i \mathbf{x}_j} u_\theta$ for $i \neq j$).

4.2. Bounding f_θ

With the partial derivative terms bounded, to bound f_θ, we use McCormick envelopes (McCormick, 1976) to obtain linear lower and upper bound functions $f_\theta^L \leq f_\theta \leq f_\theta^U$: $f_\theta^U = \mu_0^U + \mu_1^U u_\theta + \sum_{j=1}^r \sum_{\partial_{\mathbf{x}_i^j} \in \mathcal{N}^{(j)}} \mu_{j,i}^U \partial_{\mathbf{x}_i^j} u_\theta$, and $f_\theta^L = \mu_0^L + \mu_1^L u_\theta + \sum_{j=1}^r \sum_{\partial_{\mathbf{x}_i^j} \in \mathcal{N}^{(j)}} \mu_{j,i}^L \partial_{\mathbf{x}_i^j} u_\theta$, where μ_0^U, μ_1^U, and $\mu_{i,j}^U$ are functions of the global lower and upper bounds of u_θ and $\partial_{\mathbf{x}_i^j} u_\theta$. In the example of Burgers' equation (Equation 2), $f_\theta^U = \mu_0^U + \mu_1^U u_\theta + \mu_{1,0}^U \partial_{\mathbf{x}_0} u_\theta + \mu_{1,1}^U \partial_{\mathbf{x}_1} u_\theta + \mu_{2,1}^U \partial_{\mathbf{x}_1^2} u_\theta$ (and similarly for f_θ^L with μ^L).

To get f_θ^U and f_θ^L as linear functions of $\mathbf{x}$, we replace u_θ and $\partial_{\mathbf{x}_i^j} u_\theta$ with the lower and upper bound linear expressions from Section 4.1, depending on the sign of the coefficients μ^U and μ^L. As in Section 4.1, since $\mathcal{C} = \{\mathbf{x} \in \mathsf{R}^{d_0} : \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U\}$ we can then solve $\max_{\mathbf{x} \in \mathcal{C}} f_\theta^U$ and $\min_{\mathbf{x} \in \mathcal{C}} f_\theta^L$ in closed-form (see Appendix D.5), obtaining constant bounds for f_θ in $\mathcal{C}$. We explore the overall complexity of running ∂-CROWN to bound f_θ in Appendix E, and define it generally as $\mathcal{M}$ for the sake of further complexity analysis.

Algorithm 1 Greedy Input Branching

Input: function h, input domain $\mathcal{C}$, # splits N_b, # empirical samples N_s, # branches per split N_d
Result: lower bound h_{lb}, upper bound h_{ub}

1 $\mathcal{B}, \mathcal{B}_\Delta = \emptyset, \emptyset$
2 $\hat{h}_{lb}, \hat{h}_{ub} = \min \backslash \max\, h(\text{SAMPLE}(\mathcal{C}, N_s))$
3 $h_{lb}, h_{ub} = \partial\text{-CROWN}\ (h, \mathcal{C})$
4 $\mathcal{B}[\mathcal{C}] = (h_{lb}, h_{ub})$
5 $\mathcal{B}_\Delta[\mathcal{C}] = \max(\hat{h}_{lb} - h_{lb}, h_{ub} - \hat{h}_{ub})$
6 **for** $i \in \{1, \dots, N_b\}$ **do**
7 $\mathcal{C}_i = \mathcal{B}.\text{POP}(\arg\max_{\mathcal{C}'} \mathcal{B}_\Delta[\mathcal{C}'])$
8 **foreach** $\mathcal{C}' \in \text{DOMAINSPLIT}(\mathcal{C}_i, N_d)$ **do**
9 $h'_{lb}, h'_{ub} = \partial\text{-CROWN}\ (h, \mathcal{C}')$
10 $\mathcal{B}[\mathcal{C}'] = (h'_{lb}, h'_{ub})$
11 $\mathcal{B}_\Delta[\mathcal{C}'] = \max(\hat{h}_{lb} - h'_{lb}, h'_{ub} - \hat{h}_{ub})$
12 $h_{lb}, h_{ub} = \min_{\mathcal{C}'} \mathcal{B}_0[\mathcal{C}'], \max_{\mathcal{C}'} \mathcal{B}_1[\mathcal{C}']$
13 **return** h_{lb}, h_{ub}

4.3. Tighter Bounds via Greedy Input Branching

Using ∂-CROWN we can compute a bound on a nonlinear function of the derivatives of u_θ, which we will generally refer to as h, for $\mathbf{x} \in \mathcal{C}$. However, given the approximations introduced by the relaxations, it is likely these bounds will be too loose compared to the true values of h to be useful.

To improve them, we introduce *greedy input branching* (Algorithm 1). We start by computing empirical estimates of the min/max value of h across the domain (L2), and the ∂-CROWN bounds over the full domain (L3), storing the latter in the certified bounds list, $\mathcal{B}$, (L4) and the max difference between empirical and certified in the list $\mathcal{B}_\Delta$ (L5). For N_b iterations (*branchings*), we take $\mathcal{C}_i$ as the interval with the highest difference between empirical and certified values (L7). We then split it into N_d pieces using DOMAINSPLIT, compute the new certificates for those smaller sub-domains $\mathcal{C}'$ (L9), and add those certified bounds and their error w.r.t. the empirical estimate of the bounds to $\mathcal{B}$ (L10) and $\mathcal{B}_\Delta$ (L12), respectively. Finally, the tighter lower and upper bounds are then the minimum lower bound and the maximum upper bound in $\mathcal{B}$, respectively (L12). A more detailed step-by-step description of the algorithm is given in Appendix H.

Table 1: **Certifying with ∂-CROWN**: empirical lower bounds (l_b) computed using Monte Carlo (MC) samples (10^4 and 10^6 points), and certified upper bounds (u_b) using ∂-CROWN with greedy input branching for ① initial conditions, ② boundary conditions, and ③ residual condition for (a) Burgers (Raissi et al., 2019a), (b) Schrödinger (Raissi et al., 2019a), (c) Allen-Cahn (Monaco & Apiletti, 2023), and (d) Diffusion-Sorption (Takamoto et al., 2022) equations.

		Empirical l_b		Certified u_b
		MC max (10^4)	MC max (10^6)	∂-CROWN u_b (time [s])
(a) Burgers	① $\|u_\theta(0,x) \quad u_0(x)\|^2$	1.59 10 ^{6}	1.59 10 ^{6}	2.63 10 ^{6} (116.5)
	② $\|u_\theta(t, \quad 1)\|^2$	8.08 10 ^{8}	8.08 10 ^{8}	6.63 10 ^{7} (86.7)
	② $\|u_\theta(t,1)\|^2$	6.54 10 ^{8}	6.54 10 ^{8}	9.39 10 ^{7} (89.8)
	③ $\|f_\theta(\mathbf{x})\|^2$	1.23 10 ^{3}	1.80 10 ^{2}	1.03 10 ^{1} (2.8 10^5)
(b) Schrödinger	① $\|u_\theta(0,x) \quad u_0(x)\|^2$	7.06 10 ^{5}	7.06 10 ^{5}	8.35 10 ^{5} (305.2)
	② $\|u_\theta(t,5) \quad u_\theta(t, \quad 5)\|^2$	7.38 10 ^{7}	7.38 10 ^{7}	5.73 10 ^{6} (545.4)
	② $\|\partial_x u_\theta(t,5) \quad \partial_x u_\theta(t, \quad 5)\|^2$	1.14 10 ^{5}	1.14 10 ^{5}	5.31 10 ^{5} (2.4 10^3)
	③ $\|f_\theta(\mathbf{x})\|^2$	7.28 10 ^{4}	7.67 10 ^{4}	5.55 10 ^{3} (1.2 10^6)
(c) Allen-Cahn	① $\|u_\theta(0,x) \quad u_0(x)\|^2$	1.60 10 ^{3}	1.60 10 ^{3}	1.61 10 ^{3} (52.7)
	② $\|u_\theta(t, \quad 1) \quad u_\theta(t,1)\|^2$	5.66 10 ^{6}	5.66 10 ^{6}	5.66 10 ^{6} (95.4)
	③ $\|f_\theta(\mathbf{x})\|^2$	10.74	10.76	10.84 (6.7 10^5)
(d) Diffusion-Sorption	① $\|u_\theta(0,x)\|^2$	0.0	0.0	0.0 (0.2)
	② $\|u_\theta(t,0) \quad 1\|^2$	4.22 10 ^{4}	4.39 10 ^{4}	1.09 10 ^{3} (72.5)
	② $\|u_\theta(t,1) \quad D\partial_x u_\theta(t,1)\|^2$	2.30 10 ^{5}	2.34 10 ^{5}	2.37 10 ^{5} (226.4)
	③ $\|f_\theta(\mathbf{x})\|^2$	1.10 10 ^{3}	21.09	21.34 (2.4 10^6)

As the number of splits, N_b, increases, so does the tightness of our global bounds. For small dimensional spaces, it suffices to split each branch $\mathcal{C}_i$ into $N_d = 2^{d_0}$ equal branches. Note that in higher dimensional spaces, a non-equal splitting function, DOMAINSPLIT, can lead to improved convergence to the tighter bounds. The time complexity of greedy input branching is $\mathcal{O}(N_b N_d \mathcal{M})$, where $\mathcal{M}$ is the complexity of bounding each branch.

5. Experiments

The aim of this experimental section is to (i) showcase that the Definition 1 certificates obtained with ∂-CROWN are tight compared to empirical errors computed with a large number of samples (Section 5.1), (ii) highlight the relationship of our residual-based certificates and the commonly reported solution errors (Section 5.2), (iii) compare the efficiency of our method to an alternative bound propagation one (Section 5.3), and (iv) qualitatively analyze the importance of greedy input branching in the success of our method (Section 5.4). On top of these results, in Appendix A we study how the training method from Shekarpaz et al. (2022) can lead to a reduction in empirical and certified errors, and in Appendix B we showcase how ∂-CROWN can be used to identify failures in PINN training.

5.1. Certifying with ∂-CROWN

To achieve (i), we apply our post-training certification framework ∂-CROWN to two widely studied PINNs from Raissi et al. (2019a), Burgers' and Schrödinger's equations, as well as to the more complex Allen-Cahn's equation from Monaco & Apiletti (2023), and the Diffusion-Sorption equation from Takamoto et al. (2022). These PINNs were chosen for the experimental section as they are well established from previous literature in the field, and either code or trained models were available from that previous work. While we considered other suitable higher dimensional PINNs, such as several of the Navier-Stokes equations from Jin et al. (2021), or the Gray-Scott system from Giampaolo et al. (2022), neither training code nor the pre-trained models were released that allow us to apply ∂-CROWN.

Since u_θ for these PINNs use $\sigma = \tanh$ activations, we need to be able to linearly relax σ' and σ'' given pre-activation bounds. We propose a practical relaxation in Appendix F, highlighting its effeciency compared to a simple baseline in Appendix F.1. All timing results were obtained on a MacBook Pro with a 10-core M1 Max CPU. Visualizations of a fine-grained discretization of the solution and residual error landscapes is provided in Figure 4 in the Appendix.

Burgers' Equation This one-dimensional PDE is used in several areas of mathematics, fluid dynamics, nonlinear acoustics, gas dynamics and traffic flow, and is derived

from the Navier-Stokes equations for the velocity field by dropping the pressure gradient (Raissi et al., 2019a). It is defined on a temporal domain $t \in [0, 1]$ and spatial domain $x \in [-1, 1]$ as:

$$\partial_t u(t,x) + u(t,x)\partial_x u(t,x) - (0.01/\pi)\partial_{x^2} u(t,x) = 0, \tag{3}$$

for $u(0,x) = -\sin(\pi x)$, $u(t,-1) = u(t,1) = 0$. The solution $u_\theta : \mathsf{R}^2 \to \mathsf{R}$ is modeled by an 8-hidden layer, 20 neurons per layer network (Raissi et al., 2019a). The training process took ~ 13.35 minutes, and resulted in a mean ℓ_2 solution error of $6.1 \cdot 10^{-4}$.

Schrödinger's Equation Schrödinger's equation is a classical field equation used to study quantum mechanical systems. In Raissi et al. (2019a), Schrödinger's equation is defined with the temporal domain $t \in [0, \pi/2]$ and spatial domain $x \in [-5, 5]$ as:

$$i\,\partial_t u(t,x) + 0.5\,\partial_{xx} u(t,x) + |u(t,x)|^2 u(t,x) = 0, \tag{4}$$

where $u : [0, \pi/2] \times \mathcal{D} \to \mathsf{C}$ is a complex-valued solution, for initial conditions $u(0,x) = 2\,\mathrm{sech}(x)$, and periodic boundary conditions $u(t,-5) = u(t,5)$ and $\partial_x u(t,-5) = \partial_x u(t,5)$. As in Raissi et al. (2019a), $u_\theta : \mathsf{R}^2 \to \mathsf{R}^2$ is a 5-hidden layer, 100 neurons per layer network. The training took ~ 23.67 minutes, and resulted in a mean ℓ_2 solution error of $1.74 \cdot 10^{-3}$.

Allan-Cahn Equation The Allan-Cahn equation is a form of reaction-diffusion equation, describing the phase separation in multi-component alloy systems (Monaco & Apiletti, 2023). In 1D, it is defined on a temporal domain $t \in [0, 1]$ and spatial domain $x \in [-1, 1]$ as:

$$\partial_t u(t,x) + \rho u(t,x)(u^2(t,x) - 1) - \nu \partial_{x^2} u(t,x) = 0, \tag{5}$$

for $\rho = 5$, $\nu = 10^{-4}$, and $u(0,x) = x^2 \cos(\pi x)$, $u(t,-1) = u(t,1)$. The solution $u_\theta : \mathsf{R}^2 \to \mathsf{R}$ is modeled by an 6-hidden layer, 40 neurons per layer network, and due to its complexity, it is trained using the Causal training scheme from Monaco & Apiletti (2023). The training process took ~ 18.56 minutes, and resulted in a mean ℓ_2 solution error of $7.9 \cdot 10^{-3}$.

Diffusion-Sorption The diffusion-sorption equation models a diffusion system which is retarded by a sorption process, with one of the most prominent applications being groundwater contaminant transport (Takamoto et al., 2022). In (Takamoto et al., 2022), the equation is defined on a temporal domain $t \in (0, 500]$ and spatial domain $x \in (0, 1)$ as:

$$\partial_t u(t,x) - D/R(u(t,x))\partial_{x^2} u(t,x) = 0, \tag{6}$$

where $D = 5 \times 10^{-4}$ is the effective diffusion coefficient, and $R(u(t,x))$ is the retardation factor representing the sorption that hinders the diffusion process (Takamoto et al., 2022). In particular, we consider $R(u(t,x)) = 1 + {}^{(1-\phi)}/_{(\phi)}\rho_s k n_f u^{n_f - 1}(t,x)$, where $\phi = 0.29$ is the porosity of the porus medium, $\rho_s = 2880$ is the bulk density, $k = 3.5 \times 10^{-4}$ is the Freundlich's parameter, and $n_f = 0.874$ is the Freundlich's exponent. The initial and boundary conditions are defined as $u(0,x) = 0$, $u(t,0) = 0$ and $u(t,1) = D\partial_x u(t,1)$. The solution $u_\theta : \mathsf{R}^2 \to \mathsf{R}$ is modeled by a 7-hidden layer, 40 neurons per layer network, and we obtain the trained parameters from Takamoto et al. (2022). The mean ℓ_2 solution error is $9.9 \cdot 10^{-2}$.

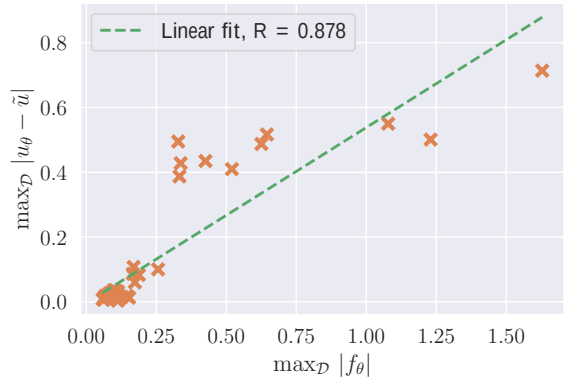


Figure 2: **Residual and solution errors**: connection of the maximum residual error ($\max_{\mathcal{S}^0} |f_\theta|$) and the maximum solution error, $\max_{\mathcal{S}^0} |u_\theta - \tilde{u}|$, for networks at different epochs of the training process (in orange).

∂-CROWN Error Certification We obtain certified bounds on the PINN errors for the conditions of Definition 1 using ∂-CROWN. We report in Table 1 our verification of the initial conditions ① using $N_b = 5k$ splits, boundary conditions ② using $N_b = 5k$ splits, and the certified bounds on the residual condition ③ using $N_b = 2M$ splits. We observe that ∂-CROWN upper bounds approach the empirical error lower bounds obtained through high-density sampling – showcasing tightness – while providing a guarantee on the continuous solution.

5.2. Empirical relation of $|f_\theta|$ and $|u_\theta - u|$

One question that might arise from our certification procedure is the relationship between the PINN residual error, $|f_\theta|$, and the solution error with respect to true solution u, $|u_\theta - u|$, across the domain. By definition, achieving a low $|f_\theta|$ implies u_θ is a valid solution for the PDE (assuming boundary and initial conditions also hold), but there is no formal guarantee related to $|u_\theta - u|$ within our framework.

Obtaining a bound on $|u_\theta - u|$ is typically a non-trivial task given u might not be unique, and does not necessarily exhibit an analytical solution. And while some recent works perform this analysis for specific PDEs by exploit-

Table 2: **Efficiency of ∂-CROWN**: comparison of ∂-CROWN (Ours), Interval Bound Propagation (IBP) and LiRPA upper bounds obtained with greedy input branching (for N_b branches) in Burgers' equation for fixed runtime limits (150s, 100s, or 10^4s). Lower is better.

	Ours (N_b)	IBP (N_b)	LiRPA (N_b)
$\|u_\theta(0,x)\|^2$ (150s)	2.63 10^{-6} (10^4)	4.12 10^{-3} (10^5)	**2.23 10^{-6}** (10^4)
$\|u_\theta(t,\ 1)\|^2$ (100s)	6.63 10^{-7} (10^4)	1.23 10^{-5} (10^5)	**6.34 10^{-7}** (10^4)
$\|u_\theta(t,1)\|^2$ (100s)	9.39 10^{-7} (10^4)	5.69 10^{-5} (10^5)	**9.12 10^{-7}** (10^4)
$\|f_\theta(x,t)\|^2$ (10^4s)	**1.30 10^{1}** (1.3 10^{5})	2.78 10^{3} (5 10^{6})	1.78 10^{2} (1.9 10^{4})

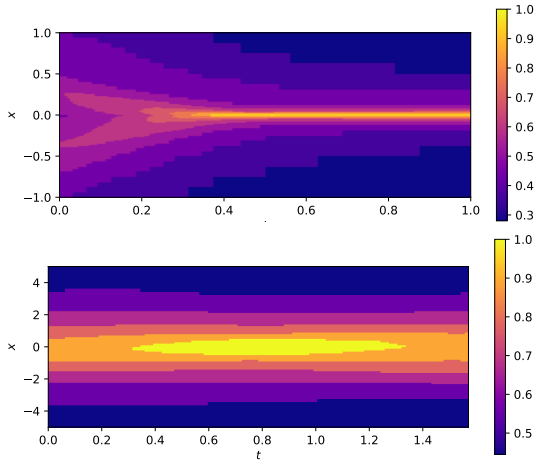


Figure 3: **Branching densities**: relative density of the input branching distribution obtained via Algorithm 1 applied to Burgers' (top) and Schrödinger's (bottom) equations.

ing their structure and/or smoothness properties (Mishra & Molinaro, 2022; Ryck & Mishra, 2022; Wang et al., 2022a), these methods typically suffer from scalability and bound tightness issues. As such, we perform an empirical analysis on Burgers' equation using a numerical, finite-difference solver to obtain $\tilde{u}(\mathbf{x})$ for sampled points $\mathbf{x}$. We randomly sample 10^6 domain points ($\mathcal{S}'$), and compute the maximum residual error, $\max_{\mathbf{x}\in\mathcal{S}'}|f_\theta(\mathbf{x})|$, and the empirical maximum solution error, $\max_{\mathbf{x}\in\mathcal{S}'}|u_\theta(\mathbf{x})-\tilde{u}(\mathbf{x})|$, for networks obtained at different epochs of the training process. We report the results in Figure 2, with each point corresponding to an instance of a network. As expected, there is a correlation between these errors obtained using a numerical solver, suggesting a similar correlation holds for $|u_\theta - u|$.

5.3. On the efficiency of ∂-CROWN

To the best of our knowledge, ∂-CROWN is the first framework designed to bound the errors of general PINNs. To highlight its efficiency, we compare is bounding performance to that of Interval Bound Propagation (IBP) (Gowal et al., 2018; Mirman et al., 2018) and LiRPA (Xu et al., 2020a) for fixed runtime limits in Burgers' equation. IBP is fast yet yields loose bounds, whereas LiRPA's full backpropagation mechanism makes it slower despite having potentially tighter bounds. The results are presented in Table 2, clearly showcasing how ∂-CROWN achieves a balance between speed (branching more than LiRPA yet less than IBP) and tightness (outperforming both methods in the tightness of the residual bounds). Note that both ∂-CROWN and LiRPA are reduced to CROWN in the initial and boundary conditions, and as such the minor differences in bounds in those cases can be attributed to implementation.

5.4. On the importance of greedy input branching

A key factor in the success of ∂-CROWN in achieving tight bounds of the residual is the greedy input branching procedure from Algorithm 1. To illustrate the fact that a uniform sampling strategy would be significantly more computationally expensive, we plot in Figure 3 the relative density of branches (*i.e.*, the percentage of branches per unit of input domain) in the case of Burgers' and Schrödinger's equations. As can be observed, there are clear imbalances at the level of the branching distribution – with areas away from relative optima of u_θ being relatively under sampled yet achieving tight bounds – showcasing the efficiency of our strategy.

6. Discussion and Limitations

We show that ∂-CROWN is able to obtain tight upper bounds on the correctness conditions established in Definition 1. We highlight in the case of the Diffusion-Sorption equation that relying on empirical lower bound estimates can be misleading – using 10^4 MC samples puts the maximum residual error at 1.10×10^{-3}, while 10^6 samples give an estimate of 21.09 –, motivating the need for ∂-CROWN to obtain guarantees across the continuous domain. Note that the absolute values of the residual errors can be seen as a function of the PDE itself, and thus cannot be directly compared across different PINNs. However, in Appendix B we effectively show how ∂-CROWN bounds can be used to detect failure cases in PINN training, highlighting another potential use of our framework on top of certifying well-trained ones.

One of the limitations of our method is unquestionably the running time, particularly for residual verification. This mostly comes down to the high number of branchings required as a result of the relative looseness of the ∂-CROWN bounds on each individual subdomain. The looseness of the bounds is likely worsened for higher-order PDEs with similar solution networks, since the PINN residual can be

viewed in that case as a depth-wise extension of the original network (following Figure 1) which, as widely observed in the network verification community, degrades the tightness of the bounds for incomplete verifiers (Wang et al., 2018) (see Appendix I). A similar argument can be made for higher dimensionality PINNs that *require larger solution networks* (unlike those, e.g., in Jin et al. (2021); Giampaolo et al. (2022), which we omit from this work for the reasons in Section 5.1). In these cases it is likely that one will need (i) tighter relaxations of the nonlinearities of the networks, and (ii) more efficient branching methods that allow us to compensate for the tightness loss in deeper networks.

For future work, it would be interesting to further study the connection between PINN *correctness* errors as per Definition 1 and solution errors, potentially connecting them for specific classes of PDEs by expanding the work of Ryck & Mishra (2022).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgments

FE is supported by EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines and Systems [EP/S024050/1] and Five AI Limited. PHST is supported by UKRI grant: Turing AI Fellowship EP/W002981/1, and by the Royal Academy of Engineering under the Research Chair and Senior Research Fellowships scheme.

References

Ayers, E. W., Eiras, F., Hawasly, M., and Whiteside, I. Parot: A practical framework for robust deep neural network training. In *NASA Formal Methods Symposium*, pp. 63–84. Springer, 2020.

Balcan, M.-F., Dick, T., Sandholm, T., and Vitercik, E. Learning to branch. In *International conference on machine learning*, pp. 344–353. PMLR, 2018.

Bunel, R. R., Turkaslan, I., Torr, P., Kohli, P., and Mudigonda, P. K. A unified view of piecewise linear neural network verification. *Advances in Neural Information Processing Systems*, 31, 2018.

Chen, Y., Lu, L., Karniadakis, G. E., and Dal Negro, L. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Optics express*, 28(8): 11618–11633, 2020.

De Palma, A., Bunel, R., Desmaison, A., Dvijotham, K., Kohli, P., Torr, P. H., and Kumar, M. P. Improved branch and bound for neural network verification via lagrangian decomposition. *arXiv preprint arXiv:2104.06718*, 2021.

Ehlers, R. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pp. 269–286. Springer, 2017.

Fang, Z. and Zhan, J. Deep physical informed neural networks for metamaterial design. *IEEE Access*, 8:24506–24513, 2019a.

Fang, Z. and Zhan, J. A physics-informed neural network framework for pdes on 3d surfaces: Time independent problems. *IEEE Access*, 8:26328–26335, 2019b.

Ferrari, C., Muller, M. N., Jovanovic, N., and Vechev, M. Complete verification via multi-neuron relaxation guided branch-and-bound. *arXiv preprint arXiv:2205.00263*, 2022.

Giampaolo, F., De Rosa, M., Qi, P., Izzo, S., and Cuomo, S. Physics-informed neural networks approach for 1d and 2d gray-scott systems. *Advanced Modeling and Simulation in Engineering Sciences*, 9(1):1–17, 2022.

Gowal, S., Dvijotham, K., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T., and Kohli, P. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

Hillebrecht, B. and Unger, B. Certified machine learning: A posteriori error estimation for physics-informed neural networks. *arXiv preprint arXiv:2203.17055*, 2022.

Huang, X., Kwiatkowska, M., Wang, S., and Wu, M. Safety verification of deep neural networks. In *International conference on computer aided verification*, pp. 3–29. Springer, 2017.

Jin, X., Cai, S., Li, H., and Karniadakis, G. E. Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, 426:109951, 2021.

Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. Reluplex: An efficient smt solver for verifying deep neural networks. In *International conference on computer aided verification*, pp. 97–117. Springer, 2017.

Kim, J., Lee, K., Lee, D., Jhin, S. Y., and Park, N. Dpm: a novel training method for physics-informed neural networks in extrapolation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 8146–8154, 2021.

Kochkov, D., Sanchez-Gonzalez, A., Smith, J. A., Pfaff, T. J., Battaglia, P., and Brenner, M. Learning latent field dynamics of pdes. In *Third Workshop on Machine Learning and the Physical Sciences (NeurIPS 2020)*, 2020.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.

Liu, D. and Wang, Y. Multi-fidelity physics-constrained neural network and its application in materials modeling. *Journal of Mechanical Design*, 141(12), 2019.

Lomuscio, A. and Maganti, L. An approach to reachability analysis for feed-forward relu neural networks. *arXiv preprint arXiv:1706.07351*, 2017.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

McCormick, G. P. Computability of global solutions to factorable nonconvex programs: Part i—convex underestimating problems. *Mathematical programming*, 10(1): 147–175, 1976.

Mirman, M., Gehr, T., and Vechev, M. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning*, pp. 3578–3586. PMLR, 2018.

Mishra, S. and Molinaro, R. Estimates on the generalization error of physics-informed neural networks for approximating pdes. *IMA Journal of Numerical Analysis*, 2022.

Monaco, S. and Apiletti, D. Training physics-informed neural networks: One learning to rule them all? *Results in Engineering*, 18:101023, 2023.

Müller, M. N., Fischer, M., Staab, R., and Vechev, M. Abstract interpretation of fixpoint iterators with applications to neural networks. *Proceedings of the ACM on Programming Languages*, 7(PLDI):786–810, 2023.

Pang, G., Lu, L., and Karniadakis, G. E. fpinns: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, 41(4):A2603–A2626, 2019.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019a.

Raissi, M., Wang, Z., Triantafyllou, M. S., and Karniadakis, G. E. Deep learning of vortex-induced vibrations. *Journal of Fluid Mechanics*, 861:119–137, 2019b.

Raissi, M., Yazdani, A., and Karniadakis, G. E. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

Ryck, T. D. and Mishra, S. Generic bounds on the approximation error for physics-informed (and) operator learning. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=bF4eYy3LTR9.

Salman, H., Yang, G., Zhang, H., Hsieh, C.-J., and Zhang, P. A convex relaxation barrier to tight robustness verification of neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Shekarpaz, S., Azizmalayeri, M., and Rohban, M. H. Piat: Physics informed adversarial training for solving partial differential equations. *arXiv preprint arXiv:2207.06647*, 2022.

Shi, Z., Zhang, H., Chang, K.-W., Huang, M., and Hsieh, C.-J. Robustness verification for transformers. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJxwPJHFwS.

Shin, Y., Darbon, J., and Karniadakis, G. E. On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type pdes. *arXiv preprint arXiv:2004.01806*, 2020.

Sun, L., Gao, H., Pan, S., and Wang, J.-X. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361:112732, 2020.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Takamoto, M., Praditia, T., Leiteritz, R., MacKinlay, D., Alesiani, F., Pflüger, D., and Niepert, M. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35: 1596–1611, 2022.

Wang, C., Li, S., He, D., and Wang, L. Is l^2 physics-informed loss always suitable for training physics-informed neural network? *arXiv preprint arXiv:2206.02016*, 2022a.

Wang, S., Chen, Y., Abdou, A., and Jana, S. Mixtrain: Scalable training of verifiably robust neural networks. *arXiv preprint arXiv:1811.02625*, 2018.

Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C.-J., and Kolter, J. Z. Beta-crown: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. *arXiv preprint arXiv:2103.06624*, 2021.

Wang, S., Yu, X., and Perdikaris, P. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022b.

Wong, E. and Kolter, Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pp. 5286–5295. PMLR, 2018.

Xu, K., Shi, Z., Zhang, H., Wang, Y., Chang, K.-W., Huang, M., Kailkhura, B., Lin, X., and Hsieh, C.-J. Automatic perturbation analysis for scalable certified robustness and beyond. *Advances in Neural Information Processing Systems*, 33:1129–1141, 2020a.

Xu, K., Zhang, H., Wang, S., Wang, Y., Jana, S., Lin, X., and Hsieh, C.-J. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. *arXiv preprint arXiv:2011.13824*, 2020b.

Zhang, D., Guo, L., and Karniadakis, G. E. Learning in modal space: Solving time-dependent stochastic pdes using physics-informed neural networks. *SIAM Journal on Scientific Computing*, 42(2):A639–A665, 2020.

Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., and Daniel, L. Efficient neural network robustness certification with general activation functions. *Advances in neural information processing systems*, 31, 2018.

Zhang, H., Wang, S., Xu, K., Li, L., Li, B., Jana, S., Hsieh, C.-J., and Kolter, J. Z. General cutting planes for bound-propagation-based neural network verification. *Advances in neural information processing systems*, 35:1656–1670, 2022.

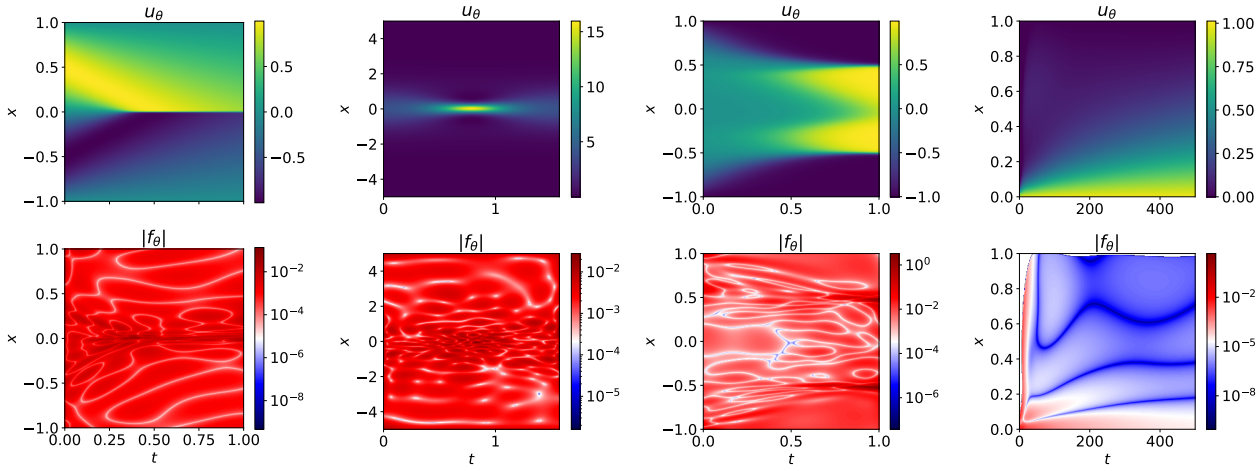


Figure 4: **Certifying with ∂-CROWN**: visualization of the time evolution of u_θ, and the residual errors as a function of the spatial temporal domain (log-scale), $|f_\theta|$, for **(a)** Burgers' equation (Raissi et al., 2019a), **(b)** Schrödinger's equation (Raissi et al., 2019a), **(c)** Allan-Cahn's equation (Monaco & Apiletti, 2023), and **(d)** the Diffusion-Sorption equation (Takamoto et al., 2022).

A. Reducing empirical and certified errors through Physics-Informed Adversarial Training

The goal of reducing the solution errors obtained by PINNs has been the research focus of several previous works (Kim et al., 2021; Krishnapriyan et al., 2021; Shekarpaz et al., 2022). To observe the effects of one of these different training schemes on the verified correctness certification of PINNs, we consider Physics-informed Adversarial Training (PIAT) (Shekarpaz et al., 2022). The procedure consists in replacing the residual loss term from Raissi et al. (2019a) with an adversarial version inspired by Madry et al. (2017). While this procedure leads to improvements in the example PINNs from Shekarpaz et al. (2022) and using our own implementation in Burgers' equation, we were unable to stably train Schrödinger's equation using PIAT. Since Schrödinger's equation is not considered in Shekarpaz et al. (2022), we only show PIAT results for Burgers' equation.

We solve the inner optimization problem using 5 PGD steps (Madry et al., 2017), and for $\epsilon = 0.05$ and a step size of 1.25ϵ. To improve convergence, we warm start PIAT training using a standard training solution after 6,000 L-BFGS iterations. The results in Table 3 show that as expected PIAT improves both empirical and certified residual bounds.

Table 3: **PIAT on Burgers' equation**: Monte Carlo sampled maximum values (10^6 samples in 0.21s) and upper bounds computed using ∂-CROWN with N_b branchings for ① initial conditions ($t = 0$, $x \in \mathcal{D}$, $N_b = 5k$), ② boundary conditions ($t \in [0, T]$, $x = -1 \vee x = 1$, $N_b = 5k$), and ③ residual norm ($t \in [0, T]$, $x \in \mathcal{D}$, $N_b = 125k$), for a PINN trained using PIAT from Shekarpaz et al. (2022).

		MC - max	∂-CROWN- u_b (time [s])
PIAT Burgers (Shekarpaz et al., 2022)	① $\lvert u_\theta(0,x) \quad u_0(x)\rvert^2$	7.40 10 ^{6}	8.18 10 ^{6} (90.9)
	② $\lvert u_\theta(t, \quad 1)\rvert^2$	2.31 10 ^{7}	3.32 10 ^{7} (49.4)
	$\lvert u_\theta(t,1)\rvert^2$	8.41 10 ^{8}	1.39 10 ^{7} (48.5)
	③ $\lvert f_\theta(\mathbf{x})\rvert^2$	3.60 10 ^{3}	2.39 10 ^{2} (2.8 10^5)

Certification convergence in PIAT vs. standard training The regularization provided by adversarial training often leads to verification algorithms converging faster to tighter lower and upper bounds. We investigate whether this is the case with ∂-CROWN's greedy branching strategy by comparing the *relative convergence* (*i.e.*, the deviation between the upper bound and the empirical maximum, $|f_\theta|^U - \max_{\mathcal{D}^0} |f_\theta|$) for the first $125k$ splits of PINNs trained in the standard and PIAT cases. The results presented in Figure 5 show that adversarial training leads to quicker convergence, requiring a lower number of branches to reach the same error when compared to standard. This suggests that our method, while already efficient, would

Table 4: **Failure identification using residual bounds**: empirical analysis of the connection between the residual bounds obtained by ∂-CROWN and the maximum solution error computed with respect to a numerical solver, u, over a sampled dataset $\mathcal{D}'$. The range of the solution values over the samples in $\mathcal{D}'$ are included for ease of comparison.

	Residual ∂-CROWN u_b	Max solution error ($\max_{D'} \lvert u_\theta - u \rvert$)	Solution range ($\min$ / $\max_{D'} u_\theta$)
Burgers	1.80×10^{-2}	3.78×10^{-3}	$[-1, 1]$
Schrödinger	7.67×10^{-4}	7.05×10^{-5}	$[1.82 \times 10^{-4}, 15.98]$
Allen-Cahn	10.76	0.86	$[-1, 1]$
Diffusion-Sorption	21.09	0.99	$[0, 1]$

benefit from smarter training strategies that lead to lower residual errors.

B. ∂-CROWN for Failure Identification

In Section 5.2 we establish the empirical correlation between residual and solution errors for PINNs at different training stages (Figure 2). While comparing PINN errors for different PDEs is not easy due to residual scaling factors, note from Table 1 that the errors obtained for Burgers' and Schrödinger's equations are orders of magnitude lower than the ones for the Allen-Cahn and Diffusion-Sorption equations. Even with different residual tolerances, this would suggest the maximum solution error of the latter, harder to train PINNs should be higher.

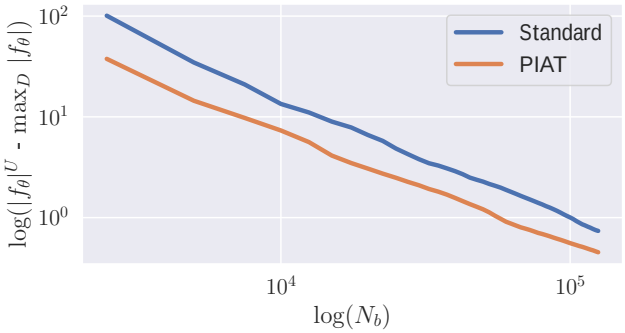


Figure 5: **Certification Convergence**: log-log plot of the relative convergence of ∂-CROWN certification for a standard trained PINN (in blue) and PIAT (in orange).

Table 4 presents the residual bounds obtained using ∂-CROWN as well as the maximum solution error with respect to a numerical solver for each of the four PINNs studied, which empirically reinforces that correlation. E.g., Burgers' equation has a maximum solution error of 3.78×10^{-3}, which is significantly lower than the trained Allen-Cahn PINN at 0.86, as expected from the residual bounds of 1.80×10^{-2} and 10.76, respectively. This contextualizes the results of Table 1 and showcases our framework can identify weaker models.

C. Ablation on N_b

We use $N_b = 2M$ for all the PINNs evaluated in this paper. A high number of branchings is required to obtain the tight bounds presented in Table 1. To justify that need, we have added plots of the variation of the obtained residual bound for Burgers' and Schrödinger' equations in Figure 6. Generally for both these PINNs we only get closer than one order of magnitude from the empirical estimates (considering the empirical MC sampled errors from Table 1) by using around $2M$ branches.

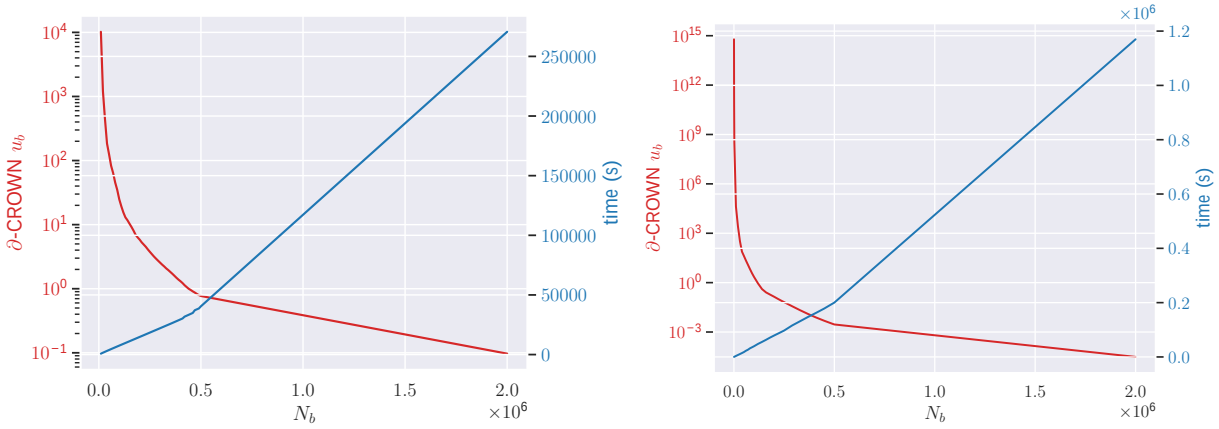
D. Proofs of partial derivative computations

D.1. Proof of Lemma 1: computing $\partial_{\mathbf{x}_i} u_\theta$

Let us now derive $\partial_{\mathbf{x}_i} u_\theta(\mathbf{x})$ for a given $i \in \{1, ..., n_0\}$. Starting backwards from the last layer and applying the chain rule we obtain:

$$\partial_{\mathbf{x}_i} u_\theta(\mathbf{x}) = \frac{\partial y^{(L)}}{\partial z^{(L-1)}} \cdot \frac{\partial z^{(L-1)}}{\partial z^{(L-2)}} \cdot \ldots \cdot \frac{\partial z^{(1)}}{\partial \mathbf{x}} \cdot \frac{\partial \mathbf{x}}{\partial x_i}$$

Given that $\partial_{\mathbf{x}_i} x = \mathbf{e}_i$ and $\frac{\partial y^{(L)}}{\partial z^{(L-1)}} = \mathbf{W}^{(L)}$, all that's left to compute to obtain the full expression is $\frac{\partial z^{(k)}}{\partial z^{(k-1)}}$, $k \in$



(a) (b)

Figure 6: **Ablation on** N_b: comparison of the residual error bounds ($|f_\theta|^2$) and runtime performance of our framework, ∂-CROWN on (a) Burgers' equation and (b) Schrödinger's equation.

$\{L-1, ..., 1\}$. Note that, for simplicity of the expressions, $z^{(0)} = \mathbf{x}$. For every element $j \in \{1, ..., d_k\}$ of $z^{(k)}$ denoted by $z_j^{(k)}$, we have:

$$\frac{\partial z_j^{(k)}}{\partial z^{(k-1)}} = \sigma'\left(\mathbf{W}_{[j,:]}^{(k)} z^{(k-1)} + \mathbf{b}_j^{(k)}\right) \mathbf{W}_{[j,:]}^{(k)}$$

where $\mathbf{W}_{j,:}^{(k)}$ denotes the j-th row of $\mathbf{W}^{(k)}$, and $\mathbf{b}_j^{(k)}$ the j-th element of $\mathbf{b}$. Thus, the final expression can be obtained by stacking the columns of the previous expression to obtain the full Jacobian:

$$\frac{\partial z^{(k)}}{\partial z^{(k-1)}} = \text{diag}\left[\sigma'\left(\mathbf{W}^{(k)} z^{(k-1)} + \mathbf{b}^{(k)}\right)\right] \cdot \mathbf{W}^{(k)}$$

This concludes the proof.

D.2. Proof of Lemma 2: computing $\partial_{\mathbf{x}_i^2} u_\theta$

Given the result obtained in Appendix D.1, let us now derive $\partial_{\mathbf{x}_i^2} u_\theta(\mathbf{x})$ for a given $i \in \{1, ..., d_0\}$. Starting backwards from the last layer of $\partial_{\mathbf{x}_i} u_\theta$ and applying the chain rule we obtain:

$$\partial_{\mathbf{x}_i^2} u_\theta = \frac{\partial}{\partial x_i}\left(\frac{\partial y^{(L)}}{\partial z^{(L-1)}} \cdot \frac{\partial z^{(L-1)}}{\partial z^{(L-2)}} \cdot ... \cdot \frac{\partial z^{(1)}}{\partial \mathbf{x}} \cdot \frac{\partial \mathbf{x}}{\partial x_i}\right) = \mathbf{W}^{(L)} \partial_{\mathbf{x}_i^2} z^{(L-1)}$$

Now the same can be applied to $\partial_{\mathbf{x}_i^2} z^{(L-1)}$, and in general to $\partial_{\mathbf{x}_i^2} z^{(k)}$ to obtain:

$$\partial_{\mathbf{x}_i^2} z^{(k)} = \frac{\partial}{\partial x_i}\left(\frac{\partial z^{(k)}}{\partial z^{(k-1)}} \partial_{\mathbf{x}_i} z^{(k-1)}\right) = \frac{\partial^2 z^{(k)}}{\partial x_i \partial z^{(k-1)}} \partial_{\mathbf{x}_i} z^{(k-1)} + \frac{\partial z^{(k)}}{\partial z^{(k-1)}} \partial_{\mathbf{x}_i^2} z^{(k-1)},$$

forming a recursion which can be taken until the first layer of $\partial_{\mathbf{x}_i} u_\theta$, *i.e.*,:

$$\partial_{\mathbf{x}_i^2} z^{(1)} = \frac{\partial}{\partial x_i}\left(\frac{\partial z^{(1)}}{\partial \mathbf{x}} \cdot \mathbf{e}_i\right) = \frac{\partial^2 z^{(1)}}{\partial x_i \partial \mathbf{x}} \cdot \mathbf{e}_i.$$

With the computation of $\partial_{\mathbf{x}_i} u_\theta$, both $\partial_{\mathbf{x}_i} z^{(k-1)}$ and $\frac{\partial z^{(k)}}{\partial z^{(k-1)}}$ are known. As such, the only missing pieces in the general recursion is the computation of $\frac{\partial^2 z^{(k)}}{\partial x_i \partial z^{(k-1)}}$. Recall from the previous section that $\frac{\partial z^{(k)}}{\partial z^{(k-1)}} = \text{diag}\left[\sigma'\left(\mathbf{W}^{(k)} z^{(k-1)} + \mathbf{b}^{(k)}\right)\right] \mathbf{W}^{(k)}$. As such:

$$\frac{\partial^2 z^{(k)}}{\partial x_i \partial z^{(k-1)}} = \frac{\partial}{\partial x_i}\left(\text{diag}\left[\sigma'\left(\mathbf{W}^{(k)} z^{(k-1)} + \mathbf{b}^{(k)}\right)\right] \mathbf{W}^{(k)}\right).$$

Following the element-wise reasoning from above, we have that:

$$\begin{aligned}\frac{\partial^2 z_j^{(k)}}{\partial x_i \partial z^{(k-1)}} &= \sigma''\left(\mathbf{W}_{j,:}^{(k)} z^{(k-1)} + \mathbf{b}_j^{(k)}\right) \frac{\partial}{\partial x_i}\left(\mathbf{W}_{j,:}^{(k)} z^{(k-1)} + \mathbf{b}_j^{(k)}\right) \mathbf{W}_{j,:}^{(k)} \\ &= \sigma''\left(\mathbf{W}_{j,:}^{(k)} z^{(k-1)} + \mathbf{b}_j^{(k)}\right)\left(\mathbf{W}_{j,:}^{(k)} \frac{\partial z^{(k-1)}}{\partial x_i}\right) \mathbf{W}_{j,:}^{(k)}\end{aligned}$$

Stacking as in the previous case, we obtain:

$$\frac{\partial^2 z^{(k)}}{\partial x_i \partial z^{(k-1)}} = \text{diag}\left[\sigma''\left(\mathbf{W}^{(k)} z^{(k-1)} + \mathbf{b}^{(k)}\right)\left(\mathbf{W}^{(k)} \partial_{\mathbf{x}_i} z^{(k-1)}\right)\right] \mathbf{W}^{(k)},$$

completing the derivation of $\partial_{\mathbf{x}_i^2} u_\theta(\mathbf{x})$.

D.3. Theorem 1: Formal Statement and Proof

Theorem 1 (∂-CROWN: linear lower and upper bounding $\partial_{\mathbf{x}_i} u_\theta$)**.** *For every $j \in \{1, \dots, d_L\}$ there exist two functions $\partial_{\mathbf{x}_i} u_{\theta,j}^U$ and $\partial_{\mathbf{x}_i} u_{\theta,j}^L$ such that, $\forall \mathbf{x} \in \mathcal{C}$ it holds that $\partial_{\mathbf{x}_i} u_{\theta,j}^L \leq \partial_{\mathbf{x}_i} u_{\theta,j} \leq \partial_{\mathbf{x}_i} u_{\theta,j}^U$, with:*

$$\begin{aligned}\partial_{\mathbf{x}_i} u_{\theta,j}^U &= \phi_{0,j,i}^{(1),U} + \sum_{r=1}^{d_0} \phi_{1,j,r}^{(1),U} \mathbf{x} + \phi_{2,j,r}^{(1),U} \\ \partial_{\mathbf{x}_i} u_{\theta,j}^L &= \phi_{0,j,i}^{(1),L} + \sum_{r=1}^{d_0} \phi_{1,j,r}^{(1),L} \mathbf{x} + \phi_{2,j,r}^{(1),L}\end{aligned}$$

where for $p \in \{0, 1, 2\}$, $\phi_{p,j,r}^{(1),U}$ and $\phi_{p,j,r}^{(1),L}$ are functions of $\mathbf{W}^{(k)}$, $y^{(k),L}$, $y^{(k),U}$, $\mathbf{A}^{(k),L}$, $\mathbf{A}^{(k),U}$ $\mathbf{a}^{(k),L}$, and $\mathbf{a}^{(k),U}$, and can be computed using a recursive closed-form expression in $\mathcal{O}(L)$ time.

Proof: Assume that through the computation of the previous bounds on u_θ, the pre-activation layer outputs of u_θ, $y^{(k)}$, are lower and upper bounded by linear functions defined as $\mathbf{A}^{(k),L}\mathbf{x} + \mathbf{a}^{(k),L} \leq y^{(k)} \leq \mathbf{A}^{(k),U}\mathbf{x} + \mathbf{a}^{(k),U}$ and $y^{(k),L} \leq y^{(k)} \leq y^{(k),U}$ for $x \in \mathcal{C}$.

Take the upper and lower bound functions for $\partial_{\mathbf{x}_i} u_\theta$ as $\partial_{\mathbf{x}_i} u_\theta^U$ and $\partial_{\mathbf{x}_i} u_\theta^L$, respectively, and the upper and lower bound functions for $\partial_{\mathbf{x}_i} z^{(k)}$ as $\partial_{\mathbf{x}_i} z^{(k),U}$ and $\partial_{\mathbf{x}_i} z^{(k),L}$, respectively. For the sake of simplicity of notation, we define $\mathbf{B}^{(k),+} = \mathbb{I}\left(\mathbf{B}^{(k)} \geq 0\right) \odot \mathbf{B}^{(k)}$ and $\mathbf{B}^{(k),-} = \mathbb{I}\left(\mathbf{B}^{(k)} < 0\right) \odot \mathbf{B}^{(k)}$.

Working backwards from $\partial_{\mathbf{x}_i} u_\theta$, we apply the same idea from CROWN (Zhang et al., 2018):

$$\begin{aligned}\partial_{\mathbf{x}_i} u_\theta^U &= \mathbf{W}^{(L),+} \partial_{\mathbf{x}_i} z^{(L-1),U} + \mathbf{W}^{(L),-} \partial_{\mathbf{x}_i} z^{(L-1),L} \\ \partial_{\mathbf{x}_i} u_\theta^L &= \mathbf{W}^{(L),+} \partial_{\mathbf{x}_i} z^{(L-1),L} + \mathbf{W}^{(L),-} \partial_{\mathbf{x}_i} z^{(L-1),U}\end{aligned} \tag{7}$$

We continue to apply this backwards propagation to $\partial_{\mathbf{x}_i} z^{(L-1)}$ to obtain $\partial_{\mathbf{x}_i} z^{(L-1),U}$ and $\partial_{\mathbf{x}_i} z^{(L-1),L}$. Recall that $\partial_{\mathbf{x}_i} z^{(k)} = \partial_{z^{(k-1)}} z^{(k)} \partial_{\mathbf{x}_i} z^{(k-1)}$, that is, for $j \in \{1, \dots, d_k\}$ we have $\partial_{\mathbf{x}_i} z_j^{(k)} = \partial_{z^{(k-1)}} z_{j,:}^{(k)} \partial_{\mathbf{x}_i} z^{(k-1)} = \sum_{n=1}^{d_{k-1}} \partial_{z^{(k-1)}} z_{j,n}^{(k)} \partial_{\mathbf{x}_i} z_n^{(k-1)}$.

We resolve the bilinear dependencies of each $\partial_{\mathbf{x}_i} z_j^{(k)}$ by relaxing it using a convex combination of the upper and lower bounds obtained by the McCormick envelopes of the product. Assuming that $\partial_{z^{(k-1)}} z_{j,n}^{(k),L} \leq \partial_{z^{(k-1)}} z_{j,n}^{(k)} \leq \partial_{z^{(k-1)}} z_{j,n}^{(k),U}$ and $\partial_{\mathbf{x}_i} z_n^{(k-1),L} \leq \partial_{\mathbf{x}_i} z_n^{(k-1)} \leq \partial_{\mathbf{x}_i} z_n^{(k-1),U}$, we have that:

$$\begin{aligned}\partial_{\mathbf{x}_i} z_j^{(k)} \leq \partial_{\mathbf{x}_i} z_j^{(k),U} &= \sum_{n=1}^{d_{k-1}} \alpha_{0,j,n}^{(k)} \partial_{\mathbf{x}_i} z_n^{(k-1)} + \alpha_{1,j,n}^{(k)} \partial_{z^{(k-1)}} z_{j,n}^{(k)} + \alpha_{2,j,n}^{(k)} \\ \partial_{\mathbf{x}_i} z_j^{(k)} \geq \partial_{\mathbf{x}_i} z_j^{(k),L} &= \sum_{n=1}^{d_{k-1}} \beta_{0,j,n}^{(k)} \partial_{\mathbf{x}_i} z_n^{(k-1)} + \beta_{1,j,n}^{(k)} \partial_{z^{(k-1)}} z_{j,n}^{(k)} + \beta_{2,j,n}^{(k)},\end{aligned} \tag{8}$$

for:

$$\begin{aligned}
\alpha^{(k)}_{0,j,n} &= \eta^{(k)}_{j,n}\partial_{z^{(k\ \ 1)}} z^{(k),U}_{j,n} + \left(1-\eta^{(k)}_{j,n}\right)\partial_{z^{(k\ \ 1)}} z^{(k),L}_{j,n} \\
\alpha^{(k)}_{1,j,n} &= \eta^{(k)}_{j,n}\partial_{\mathbf{x}_i} z^{(k-1),L}_{n} + \left(1-\eta^{(k)}_{j,n}\right)\partial_{\mathbf{x}_i} z^{(k-1),U}_{n} \\
\alpha^{(k)}_{2,j,n} &= -\eta^{(k)}_{j,n}\partial_{z^{(k\ \ 1)}} z^{(k),U}_{j,n}\partial_{\mathbf{x}_i} z^{(k-1),L}_{n} - \left(1-\eta^{(k)}_{j,n}\right)\partial_{z^{(k\ \ 1)}} z^{(k),L}_{j,n}\partial_{\mathbf{x}_i} z^{(k-1),U}_{n} \\
\beta^{(k)}_{0,j,n} &= \zeta^{(k)}_{j,n}\partial_{z^{(k\ \ 1)}} z^{(k),L}_{j,n} + \left(1-\zeta^{(k)}_{j,n}\right)\partial_{z^{(k\ \ 1)}} z^{(k),U}_{j,n} \\
\beta^{(k)}_{1,j,n} &= \zeta^{(k)}_{j,n}\partial_{\mathbf{x}_i} z^{(k-1),L}_{n} + \left(1-\zeta^{(k)}_{j,n}\right)\partial_{\mathbf{x}_i} z^{(k-1),U}_{n} \\
\beta^{(k)}_{2,j,n} &= -\zeta^{(k)}_{j,n}\partial_{z^{(k\ \ 1)}} z^{(k),L}_{j,n}\partial_{\mathbf{x}_i} z^{(k-1),L}_{n} - \left(1-\zeta^{(k)}_{j,n}\right)\partial_{z^{(k\ \ 1)}} z^{(k),U}_{j,n}\partial_{\mathbf{x}_i} z^{(k-1),U}_{n},
\end{aligned}$$

where $\eta^{(k)}_{j,n}$ and $\zeta^{(k)}_{j,n}$ are convex coefficients that can be set as hyperparameters, or optimized for as in α-CROWN (Xu et al., 2020b).

To continue the backward propagation, we now need to bound the components of $\partial_{z^{(k\ \ 1)}} z^{(k)}$. Recall from Lemma 1 that $\partial_{z^{(k\ \ 1)}} z^{(k)} = \text{diag}\left[\sigma'\left(y^{(k-1)}\right)\right]\mathbf{W}^{(k)}$, and $\partial_{z^{(k\ \ 1)}} z^{(k)}_{j,:} = \sigma'\left(y^{(k-1)}_j\right)\mathbf{W}^{(k)}_{j,:}$ for $j \in \{1,\dots,d_k\}$.

Since $y^{(k),L}_j \le y^{(k)}_j \le y^{(k),U}_j$, we can obtain a linear upper and lower bound relaxation for $\sigma'\left(y^{(k)}_j\right)$, such that $\gamma^{(k),L}_j\left(y^{(k)}_j + \delta^{(k),L}_j\right) \le \sigma'\left(y^{(k)}_j\right) \le \gamma^{(k),U}_j\left(y^{(k)}_j + \delta^{(k),U}_j\right)$. With this, we can proceed to bound $\partial_{z^{(k\ \ 1)}} z^{(k)}_{j,:}$ as:

$$\begin{aligned}
\partial_{z^{(k\ \ 1)}} z^{(k)}_{j,:} &\le \underbrace{\left(\gamma^{(k),U}_j \mathbf{W}^{(k),+}_{j,:} + \gamma^{(k),L}_j \mathbf{W}^{(k),-}_{j,:}\right)}_{\iota^{(k)}_{0,j,:}} y^{(k)}_j + \underbrace{\left(\gamma^{(k),U}_j \delta^{(k),U}_j \mathbf{W}^{(k),+}_{j,:} + \gamma^{(k),L}_j \delta^{(k),L}_j \mathbf{W}^{(k),-}_{j,:}\right)}_{\iota^{(k)}_{1,j,:}} \\
\partial_{z^{(k\ \ 1)}} z^{(k)}_{j,:} &\ge \underbrace{\left(\gamma^{(k),L}_j \mathbf{W}^{(k),+}_{j,:} + \gamma^{(k),U}_j \mathbf{W}^{(k),-}_{j,:}\right)}_{\lambda^{(k)}_{0,j,:}} y^{(k)}_j + \underbrace{\left(\gamma^{(k),L}_j \delta^{(k),L}_j \mathbf{W}^{(k),+}_{j,:} + \gamma^{(k),U}_j \delta^{(k),U}_j \mathbf{W}^{(k),-}_{j,:}\right)}_{\lambda^{(k)}_{1,j,:}}
\end{aligned} \tag{9}$$

At this point, one could continue the back-substitution process using the bounds from CROWN (Zhang et al., 2018). However, for the sake of efficiency, we use instead the pre-computed inequalities from propagating bounds through u_θ: $\mathbf{A}^{(k),U}\mathbf{x} + \mathbf{a}^{(k),U} \le y^{(k)} \le \mathbf{A}^{(k),L}\mathbf{x} + \mathbf{a}^{(k),L}$. Substituting this in Equation 9, we obtain:

$$\begin{aligned}
\partial_{z^{(k\ \ 1)}} z^{(k),U}_{j,:} &= \underbrace{\left(\iota^{(k),+}_{0,j,:}\mathbf{A}^{(k),U}_{j,:} + \iota^{(k),-}_{0,j,:}\mathbf{A}^{(k),L}_{j,:}\right)}_{\iota^{(k)}_{2,j,:}}\mathbf{x} + \underbrace{\iota^{(k),+}_{0,j,:}\mathbf{a}^{(k),U}_{j} + \iota^{(k),-}_{0,j,:}\mathbf{a}^{(k),L}_{j} + \iota^{(k)}_{1,j,:}}_{\iota^{(k)}_{3,j,:}} \\
\partial_{z^{(k\ \ 1)}} z^{(k),L}_{j,:} &= \underbrace{\left(\lambda^{(k),+}_{0,j,:}\mathbf{A}^{(k),L}_{j,:} + \lambda^{(k),-}_{0,j,:}\mathbf{A}^{(k),U}_{j,:}\right)}_{\lambda^{(k)}_{2,j,:}}\mathbf{x} + \underbrace{\lambda^{(k),+}_{0,j,:}\mathbf{a}^{(k),L}_{j} + \lambda^{(k),-}_{0,j,:}\mathbf{a}^{(k),U}_{j} + \lambda^{(k)}_{1,j,:}}_{\lambda^{(k)}_{3,j,:}}
\end{aligned} \tag{10}$$

In practice, we can use Equation 10 to compute the required $\partial_{z^{(k\ \ 1)}} z^{(k),L}_{j,n}$ and $\partial_{z^{(k\ \ 1)}} z^{(k),U}_{j,n}$ for the McCormick relaxation that leads to Equation 8. By back-substituting the result of Equation 10 in Equation 8, we obtain an expression for the upper and lower bounds on $\partial_{\mathbf{x}_i} z^{(k)}_j$ that only depends on $\partial_{\mathbf{x}_i} z^{(k-1)}$ and $\mathbf{x}$:

$$\begin{aligned}
\partial_{\mathbf{x}_i} z^{(k),U}_j &= \sum_{n=1}^{d_{k\ \ 1}} \alpha^{(k)}_{0,j,n}\partial_{\mathbf{x}_i} z^{(k-1)}_n + \alpha^{(k)}_{3,j,n}\mathbf{x} + \alpha^{(k)}_{4,j,n} \\
\partial_{\mathbf{x}_i} z^{(k),L}_j &= \sum_{n=1}^{d_{k\ \ 1}} \beta^{(k)}_{0,j,n}\partial_{\mathbf{x}_i} z^{(k-1)}_n + \beta^{(k)}_{3,j,n}\mathbf{x} + \beta^{(k)}_{4,j,n},
\end{aligned} \tag{11}$$

where:

$$\begin{aligned}
\alpha^{(k)}_{3,j,n} &= \alpha^{(k),+}_{1,j,n}\iota^{(k)}_{2,j,n} + \alpha^{(k),-}_{1,j,n}\lambda^{(k)}_{2,j,n}, \qquad & \alpha^{(k)}_{4,j,n} &= \alpha^{(k),+}_{1,j,n}\iota^{(k)}_{3,j,n} + \alpha^{(k),-}_{1,j,n}\lambda^{(k)}_{3,j,n} + \alpha^{(k)}_{2,j,n} \\
\beta^{(k)}_{3,j,n} &= \beta^{(k),+}_{1,j,n}\lambda^{(k)}_{2,j,n} + \beta^{(k),-}_{1,j,n}\iota^{(k)}_{2,j,n}, \qquad & \beta^{(k)}_{4,j,n} &= \beta^{(k),+}_{1,j,n}\lambda^{(k)}_{3,j,n} + \beta^{(k),-}_{1,j,n}\iota^{(k)}_{3,j,n} + \alpha^{(k)}_{2,j,n}
\end{aligned}$$

Given Equation 11, we now have a recursive expression for each of the blocks that compose the computation of $\partial_{\mathbf{x}_i} u_\theta$, which allows us to obtain a closed form expression for $\partial_{\mathbf{x}_i} u_\theta^U$ and $\partial_{\mathbf{x}_i} u_\theta^L$ by applying recursive back-substitution starting with Equation 7. Let us begin by performing back-substitution to the result in Equation 11 for layer $L-1$:

$$\partial_{\mathbf{x}_i} z_j^{(L-1),U} = \sum_{n=1}^{d_L{}_2} \alpha_{0,j,n}^{(L-1)} \partial_{\mathbf{x}_i} z_n^{(L-2)} + \alpha_{3,j,n}^{(L-1)} \mathbf{x} + \alpha_{4,j,n}^{(L-1)} \tag{12}$$

$$= \sum_{n=1}^{d_L{}_2} \alpha_{0,j,n}^{(L-1)} \left(\sum_{r=1}^{d_L{}_3} \mu_{0,n,r}^{(L-2)} \partial_{\mathbf{x}_i} z_r^{(L-3)} + \mu_{3,n,r}^{(L-2)} \mathbf{x} + \mu_{4,n,r}^{(L-2)} \right) + \alpha_{3,j,n}^{(L-1)} \mathbf{x} + \alpha_{4,j,n}^{(L-1)} \tag{13}$$

$$= \sum_{n=1}^{d_L{}_2} \alpha_{0,j,n}^{(L-1)} \left(\sum_{r=1}^{d_L{}_3} \mu_{0,n,r}^{(L-2)} \partial_{\mathbf{x}_i} z_r^{(L-3)} \right) + \alpha_{0,j,n}^{(L-1)} \left(\sum_{r=1}^{d_L{}_3} \mu_{3,n,r}^{(L-2)} \mathbf{x} + \mu_{4,n,r}^{(L-2)} \right) + \alpha_{3,j,n}^{(L-1)} \mathbf{x} + \alpha_{4,j,n}^{(L-1)} \tag{14}$$

$$= \sum_{r=1}^{d_L{}_3} \left(\sum_{n=1}^{d_L{}_2} \alpha_{0,j,n}^{(L-1)} \mu_{0,n,r}^{(L-2)} \right) \partial_{\mathbf{x}_i} z_r^{(L-3)} + \tag{15}$$

$$+ \left(\sum_{n=1}^{d_L{}_2} \alpha_{0,j,n}^{(L-1)} \left(\mu_{3,n,r}^{(L-2)} \mathbf{x} + \mu_{4,n,r}^{(L-2)} \right) + \frac{1}{d_{L-3}} \left(\alpha_{3,j,n}^{(L-1)} \mathbf{x} + \alpha_{4,j,n}^{(L-1)} \right) \right) \tag{16}$$

$$= \sum_{r=1}^{d_L{}_3} \rho_{0,j,r}^{(L-2)} \partial_{\mathbf{x}_i} z_r^{(L-3)} + \left(\sum_{n=1}^{d_L{}_2} \alpha_{0,j,n}^{(L-1)} \mu_{3,n,r}^{(L-2)} + \frac{1}{d_{L-3}} \alpha_{3,j,n}^{(L-1)} \right) \mathbf{x} + \tag{17}$$

$$+ \left(\sum_{n=1}^{d_L{}_1} \alpha_{0,j,n}^{(L-1)} \mu_{4,n,r}^{(L-2)} + \frac{1}{d_{L-3}} \alpha_{4,j,n}^{(L-1)} \right) \tag{18}$$

$$= \sum_{r=1}^{d_L{}_3} \rho_{0,j,r}^{(L-2)} \partial_{\mathbf{x}_i} z_r^{(L-3)} + \rho_{1,j,r}^{(L-2)} \mathbf{x} + \rho_{2,j,r}^{(L-2)}, \tag{19}$$

where:

$$\rho_{0,j,r}^{(L-2)} = \sum_{n=1}^{d_L{}_2} \alpha_{0,j,n}^{(L-1)} \mu_{0,n,r}^{(L-2)}$$

$$\rho_{1,j,r}^{(L-2)} = \sum_{n=1}^{d_L{}_2} \alpha_{0,j,n}^{(L-1)} \mu_{3,n,r}^{(L-2)} + \frac{1}{d_{L-2}} \alpha_{3,j,n}^{(L-1)}$$

$$\rho_{2,j,r}^{(L-2)} = \sum_{n=1}^{d_L{}_2} \alpha_{0,j,n}^{(L-1)} \mu_{4,n,r}^{(L-2)} + \frac{1}{d_{L-2}} \alpha_{4,j,n}^{(L-1)},$$

and:

$$\mu_{p,n,:}^{(L-2)} = \begin{cases} \alpha_{p,n,:}^{(L-2)} & \text{if } \alpha_{0,j,n}^{(L-1)} \geq 0 \\ \beta_{p,n,:}^{(L-2)} & \text{if } \alpha_{0,j,n}^{(L-1)} < 0 \end{cases} \quad , p \in \{0, 3, 4\}$$

As in CROWN (Zhang et al., 2018), given we have put Equation 19 in the same form as Equation 12, we can now apply this argument recursively using the $\rho^{(k)}$ and $\mu^{(k)}$ coefficients to obtain:

$$\partial_{\mathbf{x}_i} z_j^{(L-1),U} = \rho_{0,j,i}^{(1)} + \sum_{r=1}^{d_0} \rho_{1,j,r}^{(1)} \mathbf{x} + \rho_{2,j,r}^{(1)},$$

where:

$$\rho_{0,j,r}^{(k-1)} = \begin{cases} \alpha_{0,j,r}^{(k)} & \text{if } k = L \\ \sum_{n=1}^{d_k\ _1} \rho_{0,j,n}^{(k)} \mu_{0,n,r}^{(k-1)} & \text{if } k \in \{2, \dots, L-1\} \end{cases}$$
$$\rho_{1,j,r}^{(k-1)} = \begin{cases} \alpha_{3,j,r}^{(k)} & \text{if } k = L \\ \sum_{n=1}^{d_k\ _1} \rho_{0,j,n}^{(k)} \mu_{3,n,r}^{(k-1)} + \frac{1}{d_k\ _2} \rho_{1,j,n}^{(k)} & \text{if } k \in \{2, \dots, L-1\} \end{cases}$$
$$\rho_{2,j,r}^{(k-1)} = \begin{cases} \alpha_{4,j,r}^{(k)} & \text{if } k = L \\ \sum_{n=1}^{d_k\ _1} \rho_{0,j,n}^{(k)} \mu_{4,n,r}^{(k-1)} + \frac{1}{d_k\ _2} \rho_{2,j,n}^{(k)} & \text{if } k \in \{2, \dots, L-1\} \end{cases},$$

and:

$$\mu_{p,n,:}^{(k-1)} = \begin{cases} \alpha_{p,n,:}^{(k-1)} & \text{if } \rho_{0,j,n}^{(k)} \geq 0 \\ \beta_{p,n,:}^{(k-1)} & \text{if } \rho_{0,j,n}^{(k)} < 0 \end{cases}, p \in \{0, 3, 4\}$$

And following the same recursive argument:

$$\partial_{\mathbf{x}_i} z_j^{(L-1),L} = \tau_{0,j,i}^{(1)} + \sum_{r=1}^{d_0} \tau_{1,j,r}^{(1)} \mathbf{x} + \tau_{2,j,r}^{(1)},$$

where:

$$\tau_{0,j,r}^{(k-1)} = \begin{cases} \beta_{0,j,r}^{(k)} & \text{if } k = L \\ \sum_{n=1}^{d_k\ _1} \tau_{0,j,n}^{(k)} \omega_{0,n,r}^{(k-1)} & \text{if } k \in \{2, \dots, L-1\} \end{cases}$$
$$\tau_{1,j,r}^{(k-1)} = \begin{cases} \beta_{3,j,r}^{(k)} & \text{if } k = L \\ \sum_{n=1}^{d_k\ _1} \tau_{0,j,n}^{(k)} \omega_{3,n,r}^{(k-1)} + \frac{1}{d_k\ _2} \tau_{1,j,n}^{(k)} & \text{if } k \in \{2, \dots, L-1\} \end{cases}$$
$$\tau_{2,j,r}^{(k-1)} = \begin{cases} \beta_{4,j,r}^{(k)} & \text{if } k = L \\ \sum_{n=1}^{d_k\ _1} \tau_{0,j,n}^{(k)} \omega_{4,n,r}^{(k-1)} + \frac{1}{d_k\ _2} \tau_{2,j,n}^{(k)} & \text{if } k \in \{2, \dots, L-1\} \end{cases},$$

and:

$$\omega_{p,n,:}^{(k-1)} = \begin{cases} \beta_{p,n,:}^{(k-1)} & \text{if } \tau_{0,j,n}^{(k)} \geq 0 \\ \alpha_{p,n,:}^{(k-1)} & \text{if } \tau_{0,j,n}^{(k)} < 0 \end{cases}, p \in \{0, 3, 4\}$$

With these expressions, we can compute the required $\partial_{\mathbf{x}_i} z_n^{(k-1),L}$ and $\partial_{\mathbf{x}_i} z_n^{(k-1),U}$ which we assumed to be known to derive Equation 8.

Finally, by back-propagating the bounds starting from Equation 7, we get:

$$\begin{aligned}
\partial_{\mathbf{x}_i} u_{\theta,j}^U &= \sum_{n=1}^{d_L\ _1} \mathbf{W}_{j,n}^{(L),+} \left(\sum_{r=1}^{d_L\ _2} \alpha_{0,n,r}^{(L-1)} \partial_{\mathbf{x}_i} z_{[r]}^{(L-2)} + \alpha_{3,n,r}^{(L-1)} \mathbf{x} + \alpha_{4,n,r}^{(L-1)} \right) + \\
&\quad + \mathbf{W}_{j,n}^{(L),-} \left(\sum_{r=1}^{d_L\ _2} \beta_{0,n,r}^{(L-1)} \partial_{\mathbf{x}_i} z_{[r]}^{(L-2)} + \beta_{3,n,r}^{(L-1)} \mathbf{x} + \beta_{4,n,r}^{(L-1)} \right) \\
&= \sum_{r=1}^{d_L\ _2} \left(\sum_{n=1}^{d_L\ _1} \mathbf{W}_{j,n}^{(L),+} \alpha_{0,n,r}^{(L-1)} + \mathbf{W}_{j,n}^{(L),-} \beta_{0,n,r}^{(L-1)} \right) \partial_{\mathbf{x}_i} z_{[r]}^{(L-2)} + \\
&\quad + \left(\sum_{n=1}^{d_L\ _1} \mathbf{W}_{j,n}^{(L),+} \alpha_{3,n,r}^{(L-1)} + \mathbf{W}_{j,n}^{(L),-} \beta_{3,n,r}^{(L-1)} \right) \mathbf{x} + \left(\sum_{n=1}^{d_L\ _1} \mathbf{W}_{j,n}^{(L),+} \alpha_{4,n,r}^{(L-1)} + \mathbf{W}_{j,n}^{(L),-} \beta_{4,n,r}^{(L-1)} \right) \\
&= \sum_{r=1}^{d_L\ _2} \phi_{0,j,r}^{(L-1),U} \partial_{\mathbf{x}_i} z_n^{(L-2),U} + \phi_{1,j,r}^{(L-1),U} \mathbf{x} + \phi_{2,j,r}^{(L-1),U},
\end{aligned}$$

where:

$$\phi_{0,j,r}^{(L-1),U} = \sum_{n=1}^{d_{L-1}} \mathbf{W}_{j,n}^{(L),+} \alpha_{0,n,r}^{(L-1)} + \mathbf{W}_{j,n}^{(L),-} \beta_{0,n,r}^{(L-1)}$$
$$\phi_{1,j,r}^{(L-1),U} = \sum_{n=1}^{d_{L-1}} \mathbf{W}_{j,n}^{(L),+} \alpha_{3,n,r}^{(L-1)} + \mathbf{W}_{j,n}^{(L),-} \beta_{3,n,r}^{(L-1)}$$
$$\phi_{2,j,r}^{(L-1),U} = \sum_{n=1}^{d_{L-1}} \mathbf{W}_{j,n}^{(L),+} \alpha_{4,n,r}^{(L-1)} + \mathbf{W}_{j,n}^{(L),-} \beta_{4,n,r}^{(L-1)}.$$

From this, using the same back-propagation logic as in the derivations of $\partial_{\mathbf{x}_i} z_n^{(k-1),L}$ and $\partial_{\mathbf{x}_i} z_n^{(k-1),U}$, we can obtain:

$$\partial_{\mathbf{x}_i} u_{\theta,j}^U = \phi_{0,j,i}^{(1),U} + \sum_{r=1}^{d_0} \phi_{1,j,r}^{(1),U} \mathbf{x} + \phi_{2,j,r}^{(1),U}, \tag{20}$$

where:

$$\phi_{0,j,r}^{(k-1),U} = \begin{cases} \sum_{n=1}^{d_{k-1}} \mathbf{W}_{j,n}^{(k),+} \alpha_{0,n,r}^{(k-1)} + \mathbf{W}_{j,n}^{(k),-} \beta_{0,n,r}^{(k-1)} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} \phi_{0,j,n}^{(k),U} \upsilon_{0,n,r}^{(k-1)} & \text{if } k \in \{2, \dots, L-1\} \end{cases}$$
$$\phi_{1,j,r}^{(k-1),U} = \begin{cases} \sum_{n=1}^{d_{k-1}} \mathbf{W}_{j,n}^{(k),+} \alpha_{3,n,r}^{(k-1)} + \mathbf{W}_{j,n}^{(k),-} \beta_{3,n,r}^{(k-1)} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} \phi_{0,j,n}^{(k),U} \upsilon_{3,n,r}^{(k-1)} + \frac{1}{d_{k-2}} \phi_{1,j,n}^{(k),U} & \text{if } k \in \{2, \dots, L-1\} \end{cases}$$
$$\phi_{2,j,r}^{(k-1),U} = \begin{cases} \sum_{n=1}^{d_{k-1}} \mathbf{W}_{j,n}^{(k),+} \alpha_{4,n,r}^{(k-1)} + \mathbf{W}_{j,n}^{(k),-} \beta_{4,n,r}^{(k-1)} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} \phi_{0,j,n}^{(k),U} \upsilon_{4,n,r}^{(k-1)} + \frac{1}{d_{k-2}} \phi_{2,j,n}^{(k),U} & \text{if } k \in \{2, \dots, L-1\} \end{cases},$$

and:

$$\upsilon_{p,n,:}^{(k-1)} = \begin{cases} \alpha_{p,n,:}^{(k-1)} & \text{if } \phi_{0,j,n}^{(k),U} \geq 0 \\ \beta_{p,n,:}^{(k-1)} & \text{if } \phi_{0,j,n}^{(k),U} < 0 \end{cases}, p \in \{0, 3, 4\}$$

And similarly for the lower bound:

$$\partial_{\mathbf{x}_i} u_{\theta,j}^L = \phi_{0,j,i}^{(1),L} + \sum_{r=1}^{d_0} \phi_{1,j,r}^{(1),L} \mathbf{x} + \phi_{2,j,r}^{(1),L}, \tag{21}$$

where:

$$\phi_{0,j,r}^{(k-1),L} = \begin{cases} \sum_{n=1}^{d_{k-1}} \mathbf{W}_{j,n}^{(k),+} \beta_{0,n,r}^{(k-1)} + \mathbf{W}_{j,n}^{(k),-} \alpha_{0,n,r}^{(k-1)} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} \phi_{0,j,n}^{(k),L} \chi_{0,n,r}^{(k-1)} & \text{if } k \in \{2, \dots, L-1\} \end{cases}$$
$$\phi_{1,j,r}^{(k-1),L} = \begin{cases} \sum_{n=1}^{d_{k-1}} \mathbf{W}_{j,n}^{(k),+} \beta_{3,n,r}^{(k-1)} + \mathbf{W}_{j,n}^{(k),-} \alpha_{3,n,r}^{(k-1)} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} \phi_{0,j,n}^{(k),L} \chi_{3,n,r}^{(k-1)} + \frac{1}{d_{k-2}} \phi_{1,j,n}^{(k),L} & \text{if } k \in \{2, \dots, L-1\} \end{cases}$$
$$\phi_{2,j,r}^{(k-1),L} = \begin{cases} \sum_{n=1}^{d_{k-1}} \mathbf{W}_{j,n}^{(k),+} \beta_{4,n,r}^{(k-1)} + \mathbf{W}_{j,n}^{(k),-} \alpha_{4,n,r}^{(k-1)} & \text{if } k = L \\ \sum_{n=1}^{d_{k-1}} \phi_{0,j,n}^{(k),L} \chi_{4,n,r}^{(k-1)} + \frac{1}{d_{k-2}} \phi_{2,j,n}^{(k),L} & \text{if } k \in \{2, \dots, L-1\} \end{cases},$$

and:

$$\chi_{p,n,:}^{(k-1)} = \begin{cases} \beta_{p,n,:}^{(k-1)} & \text{if } \phi_{0,j,n}^{(k),L} \geq 0 \\ \alpha_{p,n,:}^{(k-1)} & \text{if } \phi_{0,j,n}^{(k),L} < 0 \end{cases}, p \in \{0, 3, 4\}.$$

D.4. Theorem 2 Formal Statement and Proof

Theorem 2 (∂-CROWN: linear lower and upper bounding $\partial_{\mathbf{x}_i^2} u_\theta$)**.** *Assume that through a previous computation of bounds on $\partial_{\mathbf{x}_i} u_\theta$, the components of that network required for $\partial_{\mathbf{x}_i^2} u_\theta$, i.e., $\partial_{\mathbf{x}_i} z^{(k-1)}$ and $\partial_{z^{(k-1)}} z^{(k)}$, are lower and upper bounded by linear functions. In particular, $\mathbf{C}^{(k),L}\mathbf{x}+\mathbf{c}^{(k),L} \leq \partial_{\mathbf{x}_i} z^{(k-1)} \leq \mathbf{C}^{(k),U}\mathbf{x}+\mathbf{c}^{(k),U}$ and $\mathbf{D}^{(k),L}\mathbf{x}+\mathbf{d}^{(k),L} \leq \partial_{z^{(k-1)}} z^{(k)} \leq \mathbf{D}^{(k),U}\mathbf{x}+\mathbf{d}^{(k),U}$.*

For every $j \in \{1,\dots,d_L\}$ there exist two functions $\partial_{\mathbf{x}_i^2} u^U_{\theta,j}$ and $\partial_{\mathbf{x}_i^2} u^L_{\theta,j}$ such that, $\forall \mathbf{x} \in \mathcal{C}$ it holds that $\partial_{\mathbf{x}_i^2} u^L_{\theta,j} \leq \partial_{\mathbf{x}_i^2} u_{\theta,j} \leq \partial_{\mathbf{x}_i^2} u^U_{\theta,j}$. These functions can be written as:

$$\partial_{\mathbf{x}_i^2} u^U_{\theta,j} = \psi^{(1),U}_{0,j,i} + \sum_{r=1}^{d_0} \psi^{(1),U}_{1,j,r}\mathbf{x} + \psi^{(1),U}_{2,j,r}$$
$$\partial_{\mathbf{x}_i^2} u^L_{\theta,j} = \psi^{(1),L}_{0,j,i} + \sum_{r=1}^{d_0} \psi^{(1),L}_{1,j,r}\mathbf{x} + \psi^{(1),L}_{2,j,r}$$

where for $p \in \{0,1,2\}$, $\psi^{(1),U}_{p,j,r}$ and $\psi^{(1),L}_{p,j,r}$ are functions of $\mathbf{W}^{(k)}$, $y^{(k),L}$, $y^{(k),U}$, $\mathbf{A}^{(k),L}$, $\mathbf{A}^{(k),U}$ $\mathbf{a}^{(k),L}$, $\mathbf{a}^{(k),U}$, $\mathbf{C}^{(k),L}$, $\mathbf{C}^{(k),U}$ $\mathbf{c}^{(k),L}$, $\mathbf{c}^{(k),U}$, $\mathbf{D}^{(k),L}$, $\mathbf{D}^{(k),U}$ $\mathbf{d}^{(k),L}$, and $\mathbf{d}^{(k),U}$, and can be computed using a recursive closed-form expression in $\mathcal{O}(L)$ time.

Proof: Assume that through the computation of the previous bounds on u_θ, the pre-activation layer outputs of u_θ, $y^{(k)}$, are lower and upper bounded by linear functions defined as $\mathbf{A}^{(k),L}\mathbf{x}+\mathbf{a}^{(k),L} \leq y^{(k)} \leq \mathbf{A}^{(k),U}\mathbf{x}+\mathbf{a}^{(k),U}$ and $y^{(k),L} \leq y^{(k)} \leq y^{(k),U}$ for $\mathbf{x} \in \mathcal{C}$. Additionally, we consider also that through a previous computation of bounds on $\partial_{\mathbf{x}_i} u_\theta$, the components of that network required for $\partial_{\mathbf{x}_i^2} u_\theta$, *i.e.*, $\partial_{\mathbf{x}_i} z^{(k-1)}$ and $\partial_{z^{(k-1)}} z^{(k)}$ are lower and upper bounded by linear functions. In particular, $\mathbf{C}^{(k),L}\mathbf{x}+\mathbf{c}^{(k),L} \leq \partial_{\mathbf{x}_i} z^{(k-1)} \leq \mathbf{C}^{(k),U}\mathbf{x}+\mathbf{c}^{(k),U}$ and $\mathbf{D}^{(k),L}\mathbf{x}+\mathbf{d}^{(k),L} \leq \partial_{z^{(k-1)}} z^{(k)} \leq \mathbf{D}^{(k),U}\mathbf{x}+\mathbf{d}^{(k),U}$.

Take the upper and lower bound functions for $\partial_{\mathbf{x}_i^2} u_\theta$ as $\partial_{\mathbf{x}_i^2} u^U_\theta$ and $\partial_{\mathbf{x}_i^2} u^L_\theta$, respectively, and the upper and lower bound functions for $\partial_{\mathbf{x}_i^2} z^{(k)}$ as $\partial_{\mathbf{x}_i^2} z^{(k),U}$ and $\partial_{\mathbf{x}_i^2} z^{(k),L}$, respectively. For the sake of simplicity of notation, we define $\mathbf{B}^{(k),+} = \mathsf{I}\left(\mathbf{B}^{(k)} \geq 0\right) \odot \mathbf{B}^{(k)}$ and $\mathbf{B}^{(k),-} = \mathsf{I}\left(\mathbf{B}^{(k)} < 0\right) \odot \mathbf{B}^{(k)}$.

Note that, unless explicitly mentioned otherwise, the non-network variables (denoted by Greek letters, as well as bold, capital and lowercase letters) used here have no relation to the ones from Appendix D.3.

Starting backwards from $\partial_{\mathbf{x}_i^2} z^{(k)}$, we have that:

$$\partial_{\mathbf{x}_i^2} z^{(k)}_j = \sum_{n=1}^{d_{k-1}} \partial_{\mathbf{x}_i z^{(k-1)}} z^{(k)}_{j,n} \partial_{\mathbf{x}_i} z^{(k-1)}_n + \partial_{z^{(k-1)}} z^{(k)}_{j,n} \partial_{\mathbf{x}_i^2} z^{(k-1)}_n.$$

Given the transitive property of the sum operator, we can bound $\partial_{\mathbf{x}_i^2} z^{(k)}_j$ by using a McCormick envelope around each of the multiplications. Assuming that for all $j \in \{1\dots,d_k\}, n \in \{1\dots,d_{k-1}\}$: $\partial_{\mathbf{x}_i z^{(k-1)}} z^{(k),L}_{j,n} \leq \partial_{\mathbf{x}_i z^{(k-1)}} z^{(k)}_{j,n} \leq \partial_{\mathbf{x}_i z^{(k-1)}} z^{(k),U}_{j,n}$, $\partial_{\mathbf{x}_i} z^{(k-1),L}_n \leq \partial_{\mathbf{x}_i} z^{(k-1)}_n \leq \partial_{\mathbf{x}_i} z^{(k-1),U}_n$, $\partial_{z^{(k-1)}} z^{(k)}_{j,n} \leq \partial_{z^{(k-1)}} z^{(k)}_{j,n} \leq \partial_{z^{(k-1)}} z^{(k)}_{j,n}$, and $\partial_{\mathbf{x}_i^2} z^{(k-1),L}_n \leq \partial_{\mathbf{x}_i^2} z^{(k-1)}_n \leq \partial_{\mathbf{x}_i^2} z^{(k-1),U}_n$, we obtain:

$$\begin{aligned}
\partial_{\mathbf{x}_i^2} z^{(k)}_j \quad \partial_{\mathbf{x}_i^2} z^{(k),U}_j &= \sum_{n=1}^{d_{k-1}} \alpha^{(k)}_{0,j,n} \partial_{\mathbf{x}_i} z^{(k-1)}_n + \alpha^{(k)}_{1,j,n} \partial_{\mathbf{x}_i z^{(k-1)}} z^{(k)}_{j,n} + \alpha^{(k)}_{2,j,n} \partial_{\mathbf{x}_i^2} z^{(k-1)}_n + \alpha^{(k)}_{3,j,n} \partial_{z^{(k-1)}} z^{(k)}_{j,n} + \alpha^{(k)}_{4,j,n} \\
\partial_{\mathbf{x}_i^2} z^{(k)}_j \quad \partial_{\mathbf{x}_i^2} z^{(k),L}_j &= \sum_{n=1}^{d_{k-1}} \beta^{(k)}_{0,j,n} \partial_{\mathbf{x}_i} z^{(k-1)}_n + \beta^{(k)}_{1,j,n} \partial_{\mathbf{x}_i z^{(k-1)}} z^{(k)}_{j,n} + \beta^{(k)}_{2,j,n} \partial_{\mathbf{x}_i^2} z^{(k-1)}_n + \beta^{(k)}_{3,j,n} \partial_{z^{(k-1)}} z^{(k)}_{j,n} + \beta^{(k)}_{4,j,n}
\end{aligned} \tag{22}$$

for:

$$\alpha^{(k)}_{0,j,n} = \eta^{(k)}_{j,n}\partial_{\mathbf{x}_i z^{(k-1)}} z^{(k),U}_{j,n} + \left(1 - \eta^{(k)}_{j,n}\right)\partial_{\mathbf{x}_i z^{(k-1)}} z^{(k),L}_{j,n} \qquad \alpha^{(k)}_{1,j,n} = \eta^{(k)}_{j,n}\partial_{\mathbf{x}_i} z^{(k-1),L}_n + \left(1 - \eta^{(k)}_{j,n}\right)\partial_{\mathbf{x}_i} z^{(k-1),U}_n$$

$$\alpha^{(k)}_{2,j,n} = \gamma^{(k)}_{j,n}\partial_{z^{(k-1)}} z^{(k),U}_{j,n} + \left(1 - \gamma^{(k)}_{j,n}\right)\partial_{z^{(k-1)}} z^{(k),L}_{j,n} \qquad \alpha^{(k)}_{3,j,n} = \gamma^{(k)}_{j,n}\partial_{\mathbf{x}_i^2} z^{(k-1),L}_n + \left(1 - \gamma^{(k)}_{j,n}\right)\partial_{\mathbf{x}_i^2} z^{(k-1),U}_n$$

$$\begin{aligned}\alpha^{(k)}_{4,j,n} = {} & \eta^{(k)}_{j,n}\partial_{\mathbf{x}_i z^{(k-1)}} z^{(k),U}_{j,n}\partial_{\mathbf{x}_i} z^{(k-1),L}_n \qquad \left(1 - \eta^{(k)}_{j,n}\right)\partial_{\mathbf{x}_i z^{(k-1)}} z^{(k),L}_{j,n}\partial_{\mathbf{x}_i} z^{(k-1),U}_n + \\ & \gamma^{(k)}_{j,n}\partial_{z^{(k-1)}} z^{(k),U}_{j,n}\partial_{\mathbf{x}_i x_i} z^{(k-1),L}_n \qquad \left(1 - \gamma^{(k)}_{j,n}\right)\partial_{z^{(k-1)}} z^{(k),L}_{j,n}\partial_{\mathbf{x}_i x_i} z^{(k-1),U}_n\end{aligned}$$

$$\beta^{(k)}_{0,j,n} = \zeta^{(k)}_{j,n}\partial_{\mathbf{x}_i z^{(k-1)}} z^{(k),L}_{j,n} + \left(1 - \zeta^{(k)}_{j,n}\right)\partial_{\mathbf{x}_i z^{(k-1)}} z^{(k),U}_{j,n} \qquad \beta^{(k)}_{1,j,n} = \zeta^{(k)}_{j,n}\partial_{\mathbf{x}_i} z^{(k-1),L}_n + \left(1 - \zeta^{(k)}_{j,n}\right)\partial_{\mathbf{x}_i} z^{(k-1),U}_n$$

$$\beta^{(k)}_{2,j,n} = \delta^{(k)}_{j,n}\partial_{z^{(k-1)}} z^{(k),L}_{j,n} + \left(1 - \delta^{(k)}_{j,n}\right)\partial_{z^{(k-1)}} z^{(k),U}_{j,n} \qquad \beta^{(k)}_{3,j,n} = \delta^{(k)}_{j,n}\partial_{\mathbf{x}_i^2} z^{(k-1),L}_n + \left(1 - \delta^{(k)}_{j,n}\right)\partial_{\mathbf{x}_i^2} z^{(k-1),U}_n$$

$$\begin{aligned}\beta^{(k)}_{4,j,n} = {} & \zeta^{(k)}_{j,n}\partial_{\mathbf{x}_i z^{(k-1)}} z^{(k),L}_{j,n}\partial_{\mathbf{x}_i} z^{(k-1),L}_n \qquad \left(1 - \zeta^{(k)}_{j,n}\right)\partial_{\mathbf{x}_i z^{(k-1)}} z^{(k),U}_{j,n}\partial_{\mathbf{x}_i} z^{(k-1),U}_n + \\ & \delta^{(k)}_{j,n}\partial_{z^{(k-1)}} z^{(k),L}_{j,n}\partial_{\mathbf{x}_i^2} z^{(k-1),L}_n \qquad \left(1 - \delta^{(k)}_{j,n}\right)\partial_{z^{(k-1)}} z^{(k),U}_{j,n}\partial_{\mathbf{x}_i^2} z^{(k-1),U}_n,\end{aligned}$$

where $\eta^{(k)}_{j,n}$, $\gamma^{(k)}_{j,n}$, $\zeta^{(k)}_{j,n}$ and $\delta^{(k)}_{j,n}$ are convex coefficients that can be set as hyperparameters, or optimized for as in α-CROWN (Xu et al., 2020b).

For the next step of the back-propagation process, we now need to bound $\partial_{\mathbf{x}_i} z^{(k-1)}_n$, $\partial_{\mathbf{x}_i z^{(k-1)}} z^{(k)}_{j,n}$, and $\partial_{z^{(k-1)}} z^{(k)}_{j,n}$, so as to eventually be able to write $\partial_{\mathbf{x}_i^2} z^{(k)}_j$ as a function of simply $\partial_{\mathbf{x}_i^2} z^{(k-1)}_n$ and $\mathbf{x}$. As per our assumptions at the beginning of this section, for the sake of computational efficiency we take $\partial_{\mathbf{x}_i} z^{(k-1)}_n$ and $\partial_{z^{(k-1)}} z^{(k)}_{j,n}$ from the computation of the bounds of $\partial_{\mathbf{x}_i} u_{\theta,j}$, and thus assume we have a linear upper and lower bound function of $\mathbf{x}$. This leaves us with $\partial_{\mathbf{x}_i z^{(k-1)}} z^{(k)}_{j,n}$ to bound as a linear function of $\mathbf{x}$.

Note that, as per Lemma 2, $\partial_{\mathbf{x}_i z^{(k-1)}} z^{(k)}_{j,n} = \sigma''\left(\mathbf{y}^{(k)}_j\right)\left(\mathbf{W}^{(k)}_{j,:}\partial_{\mathbf{x}_i} z^{(k-1)}\right)\mathbf{W}^{(k)}_{j,n}$. Since $\left(\mathbf{W}^{(k)}_{j,:}\partial_{\mathbf{x}_i} z^{(k-1)}\right) = \sum_{n=1}^{d_{k-1}} \mathbf{W}^{(k)}_{j,n}\partial_{\mathbf{x}_i} z^{(k-1)}_n$, and $\mathbf{C}^{(k),U}_{n,:}\mathbf{x} + \mathbf{c}^{(k),U}_n \leq \partial_{\mathbf{x}_i} z^{(k-1)}_n \leq \mathbf{C}^{(k),L}_{n,:}\mathbf{x} + \mathbf{c}^{(k),L}_n$ (from the assumptions above), we can write:

$$\mathbf{W}^{(k)}_{j,:}\partial_{\mathbf{x}_i} z^{(k-1)} \leq \underbrace{\left(\sum_{n=1}^{d_{k-1}} \mathbf{W}^{(k),+}_{j,n}\mathbf{C}^{(k),U}_{n,:} + \mathbf{W}^{(k),-}_{j,n}\mathbf{C}^{(k),L}_{n,:}\right)}_{\mathbf{E}^{(k),U}_j}\mathbf{x} + \underbrace{\left(\sum_{n=1}^{d_{k-1}} \mathbf{W}^{(k),+}_{j,n}\mathbf{c}^{(k),U} + \mathbf{W}^{(k),-}_{j,n}\mathbf{c}^{(k),L}\right)}_{\mathbf{e}^{(k),U}_j}$$

$$\mathbf{W}^{(k)}_{j,n}\partial_{\mathbf{x}_i} z^{(k-1)} \geq \underbrace{\left(\sum_{n=1}^{d_{k-1}} \mathbf{W}^{(k),+}_{j,n}\mathbf{C}^{(k),L}_{n,:} + \mathbf{W}^{(k),-}_{j,n}\mathbf{C}^{(k),U}_{n,:}\right)}_{\mathbf{E}^{(k),L}_j}\mathbf{x} + \underbrace{\left(\sum_{n=1}^{d_{k-1}} \mathbf{W}^{(k),+}_{j,n}\mathbf{c}^{(k),L}_n + \mathbf{W}^{(k),-}_{j,n}\mathbf{c}^{(k),U}_n\right)}_{\mathbf{e}^{(k),L}_j}.$$

We define $\theta^{(k),U}_j = \max_{\mathbf{x}\in\mathcal{C}} \mathbf{E}^{(k),U}_j\mathbf{x} + \mathbf{e}^{(k),U}_j$ and $\theta^{(k),L}_j = \min_{\mathbf{x}\in\mathcal{C}} \mathbf{E}^{(k),L}_j\mathbf{x} + \mathbf{e}^{(k),L}_j$. As with the first derivative case, since $y^{(k),L}_j \leq y^{(k)}_j \leq y^{(k),U}_j$, we can obtain a linear upper and lower bound relaxation for $\sigma''\left(y^{(k)}_j\right)$, such that $\lambda^{(k),L}_j\left(y^{(k)}_j + \mu^{(k),L}_j\right) \leq \sigma''\left(y^{(k)}_j\right) \leq \lambda^{(k),U}_j\left(y^{(k)}_j + \mu^{(k),U}_j\right)$, as well as the values $\iota^{(k),L}_j \leq \sigma''\left(y^{(k)}_j\right) \leq \iota^{(k),U}_j$. By considering the assumption that $\mathbf{A}^{(k),U}_{j,:}\mathbf{x} + \mathbf{a}^{(k),U}_j \leq y^{(k)} \leq \mathbf{A}^{(k),L}_{j,:}\mathbf{x} + \mathbf{a}^{(k),L}_j$, we can obtain:

$$\sigma''\left(y^{(k)}_j\right) \leq \underbrace{\left(\lambda^{(k),U,+}_j\mathbf{A}^{(k),U}_{j,:} + \lambda^{(k),U,-}_j\mathbf{A}^{(k),L}_{j,:}\right)}_{\mathbf{H}^{(k),U}_j}\mathbf{x} + \underbrace{\left(\lambda^{(k),U,+}_j\mathbf{a}^{(k),U}_j + \lambda^{(k),U,-}_j\mathbf{a}^{(k),L}_j + \lambda^{(k),U}_j\mu^{(k),U}_j\right)}_{\mathbf{h}^{(k),U}_j}$$

$$\sigma''\left(y^{(k)}_j\right) \geq \underbrace{\left(\lambda^{(k),L,+}_j\mathbf{A}^{(k),L}_{j,:} + \lambda^{(k),L,-}_j\mathbf{A}^{(k),U}_{j,:}\right)}_{\mathbf{H}^{(k),L}_j}\mathbf{x} + \underbrace{\left(\lambda^{(k),L,+}_j\mathbf{a}^{(k),L}_j + \lambda^{(k),L,-}_j\mathbf{a}^{(k),U}_j + \lambda^{(k),L}_j\mu^{(k),L}_j\right)}_{\mathbf{h}^{(k),L}_j}.$$

This allows us to relax $\sigma''\left(y_j^{(k)}\right)\left(\mathbf{W}_{j,:}^{(k)}\partial_{\mathbf{x}_i} z^{(k-1)}\right)$ using McCormick envelopes:

$$\sigma''\left(y_j^{(k)}\right)\left(\mathbf{W}_{j,:}^{(k)}\partial_{\mathbf{x}_i} z^{(k-1)}\right) \leq \nu_{0,j}^{(k),U}\left(\mathbf{W}_{j,:}^{(k)}\partial_{\mathbf{x}_i} z^{(k-1)}\right) + \nu_{1,j}^{(k),U}\sigma''\left(y_j^{(k)}\right) + \nu_{2,j}^{(k),U}$$
$$\sigma''\left(y_j^{(k)}\right)\left(\mathbf{W}_{j,:}^{(k)}\partial_{\mathbf{x}_i} z^{(k-1)}\right) \geq \nu_{0,j}^{(k),L}\left(\mathbf{W}_{j,:}^{(k)}\partial_{\mathbf{x}_i} z^{(k-1)}\right) + \nu_{1,j}^{(k),L}\sigma''\left(y_j^{(k)}\right) + \nu_{2,j}^{(k),L},$$

for:

$$\nu_{0,j}^{(k),U} = \rho_j^{(k)}\iota_j^{(k),U} + \left(1-\rho_j^{(k)}\right)\iota_j^{(k),L} \qquad \nu_{1,j,n}^{(k),U} = \rho_j^{(k)}\theta_j^{(k),L} + \left(1-\rho_j^{(k)}\right)\iota_j^{(k),U}$$
$$\nu_{2,j}^{(k),U} = -\rho_j^{(k)}\iota_j^{(k),U}\theta_j^{(k),L} - \left(1-\rho_j^{(k)}\right)\iota_j^{(k),L}\theta_j^{(k),U}$$

$$\nu_{0,j}^{(k),L} = \tau_j^{(k)}\iota_j^{(k),L} + \left(1-\tau_j^{(k)}\right)\iota_j^{(k),U} \qquad \nu_{1,j}^{(k),L} = \tau_j^{(k)}\theta_j^{(k),L} + \left(1-\tau_j^{(k)}\right)\theta_j^{(k),U}$$
$$\nu_{2,j}^{(k),L} = -\tau_j^{(k)}\iota_j^{(k),L}\theta_j^{(k),L} - \left(1-\tau_j^{(k)}\right)\iota_j^{(k),U}\theta_j^{(k),U},$$

where $\rho_j^{(k)}$ and $\tau_j^{(k)}$ are convex coefficients that can be set as hyperparameters, or optimized for as in α-CROWN (Xu et al., 2020b). By replacing this multiplication in the expression from Lemma 2, we bound $\partial_{\mathbf{x}_i z^{(k\ \ 1)}} z_{j,n}^{(k)}$ as:

$$\partial_{\mathbf{x}_i z^{(k\ \ 1)}} z_{j,n}^{(k)} \leq \upsilon_{0,j,n}^{(k),U}\left(\mathbf{W}_{j,:}^{(k)}\partial_{\mathbf{x}_i} z^{(k-1)}\right) + \upsilon_{1,j,n}^{(k),U}\sigma''\left(y_j^{(k)}\right) + \upsilon_{2,j}^{(k),U}$$
$$\partial_{\mathbf{x}_i z^{(k\ \ 1)}} z_{j,n}^{(k)} \geq \upsilon_{0,j,n}^{(k),L}\left(\mathbf{W}_{j,:}^{(k)}\partial_{\mathbf{x}_i} z^{(k-1)}\right) + \upsilon_{1,j,n}^{(k),L}\sigma''\left(y_j^{(k)}\right) + \upsilon_{2,j}^{(k),L},$$

for:

$$\upsilon_{i,j,n}^{(k),U} = \nu_{i,j}^{(k),U}\mathbf{W}_{j,n}^{(k),+} + \nu_{i,j}^{(k),L}\mathbf{W}_{j,n}^{(k),-}, \quad \upsilon_{i,j,n}^{(k),L} = \nu_{i,j}^{(k),L}\mathbf{W}_{j,n}^{(k),+} + \nu_{i,j}^{(k),U}\mathbf{W}_{j,n}^{(k),-} \qquad i \in \{0,1,2\}.$$

By replacing the lower and upper bounds for $\sigma''(y_j^{(k)})$ and $\left(\mathbf{W}_{j,:}^{(k)}\partial_{\mathbf{x}_i} z^{(k-1)}\right)$ in the previous inequality, we obtain the expression:

$$\partial_{\mathbf{x}_i z^{(k\ \ 1)}} z_{j,n}^{(k)} \leq \mathbf{M}_{j,n}^{(k),U}\mathbf{x} + \mathbf{m}_{j,n}^{(k),U}$$
$$\partial_{\mathbf{x}_i z^{(k\ \ 1)}} z_{j,n}^{(k)} \geq \mathbf{M}_{j,n}^{(k),L}\mathbf{x} + \mathbf{m}_{j,n}^{(k),L},$$

for:

$$\begin{aligned}
\mathbf{M}_{j,n}^{(k),U} =& \upsilon_{0,j,n}^{(k),U,+}\mathbf{E}_j^{(k),U} + \upsilon_{0,j,n}^{(k),U,-}\mathbf{E}_j^{(k),L} + \upsilon_{1,j,n}^{(k),U,+}\mathbf{H}_j^{(k),U} + \upsilon_{1,j,n}^{(k),U,-}\mathbf{H}_j^{(k),L}\\
\mathbf{m}_{j,n}^{(k),U} =& \upsilon_{0,j,n}^{(k),U,+}\mathbf{e}_j^{(k),U} + \upsilon_{0,j,n}^{(k),U,-}\mathbf{e}_j^{(k),L} + \upsilon_{1,j,n}^{(k),U,+}\mathbf{h}_j^{(k),U} + \upsilon_{1,j,n}^{(k),U,-}\mathbf{h}_j^{(k),L} + \upsilon_{2,j,n}^{(k),U}\\
\mathbf{M}_{j,n}^{(k),L} =& \upsilon_{0,j,n}^{(k),L,+}\mathbf{E}_j^{(k),L} + \upsilon_{0,j,n}^{(k),L,-}\mathbf{E}_j^{(k),U} + \upsilon_{1,j,n}^{(k),L,+}\mathbf{H}_j^{(k),L} + \upsilon_{1,j,n}^{(k),L,-}\mathbf{H}_j^{(k),U}\\
\mathbf{m}_{j,n}^{(k),L} =& \upsilon_{0,j,n}^{(k),L,+}\mathbf{e}_j^{(k),L} + \upsilon_{0,j,n}^{(k),L,-}\mathbf{e}_j^{(k),U} + \upsilon_{1,j,n}^{(k),L,+}\mathbf{h}_j^{(k),L} + \upsilon_{1,j,n}^{(k),L,-}\mathbf{h}_j^{(k),U} + \upsilon_{2,j,n}^{(k),L}.
\end{aligned}$$

Finally in the derivation of $\partial_{\mathbf{x}_i^2} z_j^{(k)}$ as a function of $\mathbf{x}$ and $\partial_{\mathbf{x}_i^2} z^{(k-1)}$, we just have to replace all the quantities in Equation 22 (recalling from the assumptions that $\mathbf{C}^{(k),U}\mathbf{x} + \mathbf{c}^{(k),U} \leq \partial_{\mathbf{x}_i} z^{(k-1)} \leq \mathbf{C}^{(k),L}\mathbf{x} + \mathbf{c}^{(k),L}$ and $\mathbf{D}^{(k),U}\mathbf{x} + \mathbf{d}^{(k),U} \leq \partial_{z^{(k\ \ 1)}} z^{(k)} \leq \mathbf{D}^{(k),L}\mathbf{x} + \mathbf{d}^{(k),L}$) to obtain:

$$\begin{aligned}
\partial_{\mathbf{x}_i^2} z_j^{(k)} \leq \partial_{\mathbf{x}_i^2} z_j^{(k),U} &= \sum_{n=1}^{d_{k\ \ 1}} \alpha_{2,j,n}^{(k)}\partial_{\mathbf{x}_i^2} z_n^{(k-1)} + \alpha_{5,j,n}^{(k)}\mathbf{x} + \alpha_{6,j,n}^{(k)}\\
\partial_{\mathbf{x}_i^2} z_j^{(k)} \geq \partial_{\mathbf{x}_i^2} z_j^{(k),L} &= \sum_{n=1}^{d_{k\ \ 1}} \beta_{2,j,n}^{(k)}\partial_{\mathbf{x}_i^2} z_n^{(k-1)} + \beta_{5,j,n}^{(k)}\mathbf{x} + \beta_{6,j,n}^{(k)},
\end{aligned} \tag{23}$$

where:

$$\alpha_{5,j,n}^{(k)} = \alpha_{0,j,n}^{(k),+}\mathbf{C}_n^{(k),U} + \alpha_{0,j,n}^{(k),-}\mathbf{C}_n^{(k),L} + \alpha_{1,j,n}^{(k),+}\mathbf{M}_{j,n}^{(k),U} + \alpha_{1,j,n}^{(k),-}\mathbf{M}_{j,n}^{(k),L} + \alpha_{3,j,n}^{(k),+}\mathbf{D}_{j,n}^{(k),U} + \alpha_{3,j,n}^{(k),-}\mathbf{D}_{j,n}^{(k),L}$$
$$\alpha_{6,j,n}^{(k)} = \alpha_{0,j,n}^{(k),+}\mathbf{c}_n^{(k),U} + \alpha_{0,j,n}^{(k),-}\mathbf{c}_n^{(k),L} + \alpha_{1,j,n}^{(k),+}\mathbf{m}_{j,n}^{(k),U} + \alpha_{1,j,n}^{(k),-}\mathbf{m}_{j,n}^{(k),L} + \alpha_{3,j,n}^{(k),+}\mathbf{d}_{j,n}^{(k),U} + \alpha_{3,j,n}^{(k),-}\mathbf{d}_{j,n}^{(k),L} + \alpha_{4,j,n}^{(k)}$$
$$\beta_{5,j,n}^{(k)} = \beta_{0,j,n}^{(k),+}\mathbf{C}_n^{(k),L} + \beta_{0,j,n}^{(k),-}\mathbf{C}_n^{(k),U} + \beta_{1,j,n}^{(k),+}\mathbf{M}_{j,n}^{(k),L} + \beta_{1,j,n}^{(k),-}\mathbf{M}_{j,n}^{(k),U} + \beta_{3,j,n}^{(k),+}\mathbf{D}_{j,n}^{(k),L} + \beta_{3,j,n}^{(k),-}\mathbf{D}_{j,n}^{(k),U}$$
$$\beta_{6,j,n}^{(k)} = \beta_{0,j,n}^{(k),+}\mathbf{c}_n^{(k),L} + \beta_{0,j,n}^{(k),-}\mathbf{c}_n^{(k),U} + \beta_{1,j,n}^{(k),+}\mathbf{m}_{j,n}^{(k),L} + \beta_{1,j,n}^{(k),-}\mathbf{m}_{j,n}^{(k),U} + \beta_{3,j,n}^{(k),+}\mathbf{d}_{j,n}^{(k),L} + \beta_{3,j,n}^{(k),-}\mathbf{d}_{j,n}^{(k),U} + \beta_{4,j,n}^{(k)}$$

This forms a recursion of exactly the same form as Equation 11 from Appendix D.3, where only the coefficients of $\partial_{\mathbf{x}_i^2} z_n^{(k-1)}$ and $\mathbf{x}$ are different ($\alpha_{0,j,n}^{(k)}$ in this case is referred by $\alpha_{2,j,n}^{(k)}$, $\alpha_{3,j,n}^{(k)}$ by $\alpha_{5,j,n}^{(k)}$, and $\alpha_{4,j,n}^{(k)}$ by $\alpha_{6,j,n}^{(k)}$, and similarly for the β values). This yields:

$$\partial_{\mathbf{x}_i x_i} z_j^{(L-1),U} = \rho_{0,j,i}^{(1),U} + \sum_{r=1}^{d_0} \rho_{1,j,r}^{(1),U}\mathbf{x} + \rho_{2,j,r}^{(1),U},$$

where:

$$\rho_{0,j,r}^{(k-1),U} = \begin{cases} \alpha_{2,n,r}^{(k)} & \text{if } k = L \\ \sum_{n=1}^{d_k} \rho_{0,j,n}^{(k),U}\mu_{2,n,r}^{(k-1),U} & \text{if } k \in \{2,\dots,L-1\} \end{cases}$$
$$\rho_{1,j,r}^{(k-1),U} = \begin{cases} \alpha_{5,n,r}^{(k)} & \text{if } k = L \\ \sum_{n=1}^{d_k} \rho_{0,j,n}^{(k),U}\mu_{5,n,r}^{(k-1),U} + \frac{1}{d_k}\rho_{1,j,n}^{(k),U} & \text{if } k \in \{2,\dots,L-1\} \end{cases}$$
$$\rho_{2,j,r}^{(k-1),U} = \begin{cases} \alpha_{6,n,r}^{(k)} & \text{if } k = L \\ \sum_{n=1}^{d_k} \rho_{0,j,n}^{(k),U}\mu_{6,n,r}^{(k-1),U} + \frac{1}{d_k}\rho_{2,j,n}^{(k),U} & \text{if } k \in \{2,\dots,L-1\} \end{cases},$$

and:

$$\mu_{p,n,:}^{(k-1),U} = \begin{cases} \alpha_{p,n,:}^{(k-1)} & \text{if } \rho_{0,j,n}^{(k),U} \geq 0 \\ \beta_{p,n,:}^{(k-1)} & \text{if } \rho_{0,j,n}^{(k),U} < 0 \end{cases}, p \in \{2,5,6\}.$$

And following the same argument:

$$\partial_{\mathbf{x}_i} z_j^{(L-1),L} = \rho_{0,j,i}^{(1),L} + \sum_{r=1}^{d_0} \rho_{1,j,r}^{(1),L}\mathbf{x} + \rho_{2,j,r}^{(1),L},$$

where:

$$\rho_{0,j,r}^{(k-1),L} = \begin{cases} \beta_{2,n,r}^{(k)} & \text{if } k = L \\ \sum_{n=1}^{d_k} \rho_{0,j,n}^{(k),L}\mu_{2,n,r}^{(k-1),L} & \text{if } k \in \{2,\dots,L-1\} \end{cases}$$
$$\rho_{1,j,r}^{(k-1),L} = \begin{cases} \beta_{5,n,r}^{(k)} & \text{if } k = L \\ \sum_{n=1}^{d_k} \rho_{0,j,n}^{(k),L}\mu_{5,n,r}^{(k-1),L} + \frac{1}{d_k}\rho_{1,j,n}^{(k),L} & \text{if } k \in \{2,\dots,L-1\} \end{cases}$$
$$\rho_{2,j,r}^{(k-1),L} = \begin{cases} \beta_{6,n,r}^{(k)} & \text{if } k = L \\ \sum_{n=1}^{d_k} \rho_{0,j,n}^{(k),L}\mu_{6,n,r}^{(k-1),L} + \frac{1}{d_k}\rho_{2,j,n}^{(k),L} & \text{if } k \in \{2,\dots,L-1\} \end{cases},$$

and:

$$\mu_{p,n,:}^{(k-1),L} = \begin{cases} \beta_{p,n,:}^{(k-1)} & \text{if } \rho_{0,j,n}^{(k),L} \geq 0 \\ \alpha_{p,n,:}^{(k-1)} & \text{if } \rho_{0,j,n}^{(k),L} < 0 \end{cases}, p \in \{2,5,6\}$$

With these expressions, we can compute the required $\partial_{\mathbf{x}_i^2} z_n^{(k-1),L}$ and $\partial_{\mathbf{x}_i^2} z_n^{(k-1),U}$ which we assumed to be known to derive Equation 22.

Finally, with the exact same argument as in Appendix D.3, we obtain:

$$\partial_{\mathbf{x}_i} u^U_{\theta,j} = \psi^{(1),U}_{0,j,i} + \sum_{r=1}^{d_0} \psi^{(1),U}_{1,j,r}\mathbf{x} + \psi^{(1),U}_{2,j,r},$$

where:

$$\psi^{(k-1),U}_{0,j,r} = \begin{cases} \sum_{n=1}^{d_k} \mathbf{W}^{(k),+}_{j,n}\alpha^{(k-1)}_{2,n,r} + \mathbf{W}^{(k),-}_{j,n}\beta^{(k-1)}_{2,n,r} & \text{if } k = L \\ \sum_{n=1}^{d_k} \psi^{(k),U}_{0,j,n}\psi^{(k-1),U}_{2,n,r} & \text{if } k \in \{2,\dots,L-1\} \end{cases}$$
$$\psi^{(k-1),U}_{1,j,r} = \begin{cases} \sum_{n=1}^{d_k} \mathbf{W}^{(k),+}_{j,n}\alpha^{(k-1)}_{5,n,r} + \mathbf{W}^{(k),-}_{j,n}\beta^{(k-1)}_{5,n,r} & \text{if } k = L \\ \sum_{n=1}^{d_k} \psi^{(k),U}_{0,j,n}\psi^{(k-1),U}_{5,n,r} + \frac{1}{d_{k-2}}\psi^{(k),U}_{1,j,n} & \text{if } k \in \{2,\dots,L-1\} \end{cases}$$
$$\psi^{(k-1),U}_{2,j,r} = \begin{cases} \sum_{n=1}^{d_k} \mathbf{W}^{(k),+}_{j,n}\alpha^{(k-1)}_{6,n,r} + \mathbf{W}^{(k),-}_{j,n}\beta^{(k-1)}_{6,n,r} & \text{if } k = L \\ \sum_{n=1}^{d_k} \psi^{(k),U}_{0,j,n}\psi^{(k-1),U}_{6,n,r} + \frac{1}{d_{k-2}}\psi^{(k),U}_{2,j,n} & \text{if } k \in \{2,\dots,L-1\} \end{cases},$$

and:

$$\psi^{(k-1)}_{p,n,:} = \begin{cases} \alpha^{(k-1)}_{p,n,:} & \text{if } \psi^{(k),U}_{0,j,n} \geq 0 \\ \beta^{(k-1)}_{p,n,:} & \text{if } \psi^{(k),U}_{0,j,n} < 0 \end{cases}, p \in \{2,5,6\}.$$

And similarly for the lower bound:

$$\partial_{\mathbf{x}_i} u^L_{\theta,j} = \psi^{(1),L}_{0,j,i} + \sum_{r=1}^{d_0} \psi^{(1),L}_{1,j,r}\mathbf{x} + \psi^{(1),L}_{2,j,r},$$

where:

$$\psi^{(k-1),L}_{0,j,r} = \begin{cases} \sum_{n=1}^{d_k} \mathbf{W}^{(k),+}_{j,n}\beta^{(k-1)}_{2,n,r} + \mathbf{W}^{(k),-}_{j,n}\alpha^{(k-1)}_{2,n,r} & \text{if } k = L \\ \sum_{n=1}^{d_k} \psi^{(k),L}_{0,j,n}\psi^{(k-1),L}_{2,n,r} & \text{if } k \in \{2,\dots,L-1\} \end{cases}$$
$$\psi^{(k-1),L}_{1,j,r} = \begin{cases} \sum_{n=1}^{d_k} \mathbf{W}^{(k),+}_{j,n}\beta^{(k-1)}_{5,n,r} + \mathbf{W}^{(k),-}_{j,n}\alpha^{(k-1)}_{5,n,r} & \text{if } k = L \\ \sum_{n=1}^{d_k} \psi^{(k),L}_{0,j,n}\psi^{(k-1),L}_{5,n,r} + \frac{1}{d_{k-2}}\psi^{(k),L}_{1,j,n} & \text{if } k \in \{2,\dots,L-1\} \end{cases}$$
$$\psi^{(k-1),L}_{2,j,r} = \begin{cases} \sum_{n=1}^{d_k} \mathbf{W}^{(k),+}_{j,n}\beta^{(k-1)}_{6,n,r} + \mathbf{W}^{(k),-}_{j,n}\alpha^{(k-1)}_{6,n,r} & \text{if } k = L \\ \sum_{n=1}^{d_k} \psi^{(k),L}_{0,j,n}\psi^{(k-1),L}_{6,n,r} + \frac{1}{d_{k-2}}\psi^{(k),L}_{2,j,n} & \text{if } k \in \{2,\dots,L-1\} \end{cases},$$

and:

$$\psi^{(k-1),L}_{p,n,:} = \begin{cases} \beta^{(k-1)}_{p,n,:} & \text{if } \psi^{(k),L}_{0,j,n} \geq 0 \\ \alpha^{(k-1)}_{p,n,:} & \text{if } \psi^{(k),L}_{0,j,n} < 0 \end{cases}, p \in \{2,5,6\}.$$

D.5. Formulation and proof of closed-form global bounds on $\partial_{\mathbf{x}_i} u_\theta$

Lemma 3 (Closed-form global bounds on $\partial_{\mathbf{x}_i} u_\theta$)**.** *For every $j \in \{1,\dots,d_L\}$ there exist two values $\kappa^U_j \in \mathsf{R}$ and $\kappa^L_j \in \mathsf{R}$, such that $\forall \mathbf{x} \in \mathcal{C} = \{\mathbf{x} \in \mathsf{R}^{d_0} : \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U\}$ it holds that $\kappa^L_j \leq \partial_{\mathbf{x}_i} u_{\theta,j} \leq \kappa^U_j$, with:*

$$\kappa^U_j = \mathbf{B}^{U,+}\mathbf{x}^U + \mathbf{B}^{U,-}\mathbf{x}^L + \phi^{(1)}_{0,j,i} + \sum_{r=1}^{d_0} \phi^{(1)}_{2,j,r}$$
$$\kappa^L_j = \mathbf{B}^{L,+}\mathbf{x}^L + \mathbf{B}^{L,-}\mathbf{x}^U + \psi^{(1)}_{0,j,i} + \sum_{r=1}^{d_0} \psi^{(1)}_{2,j,r},$$

where $\mathbf{B}^U = \sum_{r=1}^{d_0} \phi^{(1)}_{1,j,r}$, $\mathbf{B}^L = \sum_{r=1}^{d_0} \psi^{(1)}_{1,j,r}$, and $\mathbf{B}^{\cdot,+} = \mathsf{I}(\mathbf{B}^{\cdot} \geq 0) \odot \mathbf{B}^{\cdot}$ and $\mathbf{B}^{\cdot,-} = \mathsf{I}(\mathbf{B}^{\cdot} < 0) \odot \mathbf{B}^{\cdot}$.

Proof. Take a function $f : \mathsf{R}^{d_0} \to \mathsf{R}$ defined as $f(\mathbf{x}) = \mathbf{v}^\top \mathbf{x} + c$ for $\mathbf{v} \in \mathsf{R}^{d_0}$ and $c \in \mathsf{R}$, as well as a domain $\mathcal{C} = \{\mathbf{x} \in \mathsf{R}^{d_0} : \mathbf{x}^L \leq \mathbf{x} \leq \mathbf{x}^U\}$. Given the perpendicularity of the constraints in $\mathcal{C}$, by separating each component of f we obtain:

$$\max_{\mathbf{x}\in\mathcal{C}} f(\mathbf{x}) = (\mathbf{v}^+)^\top \mathbf{x}^U + (\mathbf{v}^-)^\top \mathbf{x}^L + c, \qquad \min_{\mathbf{x}\in\mathcal{C}} f(\mathbf{x}) = (\mathbf{v}^+)^\top \mathbf{x}^L + (\mathbf{v}^-)^\top \mathbf{x}^U + c,$$

where $\mathbf{v}^+ = \mathsf{I}(\mathbf{v} \geq 0) \odot \mathbf{v}$ and $\mathbf{v}^- = \mathsf{I}(\mathbf{v} < 0) \odot \mathbf{v}$. □

E. On the Complexity of Bounding using ∂-CROWN

The complexity $\mathcal{M}$ of bounding f_θ (or any function of the partial derivatives of u_θ) is contingent on the type of PDE we are bounding.

For simplicity, assume the solution network, u_θ, has L fully connected hidden layers each with d output neurons, and that the relaxation of the activation functions and their derivatives, i.e., σ, σ', ..., can be computed in $\mathcal{O}(1)$. Using CROWN we can bound the output of layer $l \in \{1, \ldots, L\}$ in $\mathcal{O}(ld^2)$, yielding the complexity of bounding the output of u_θ as $\mathcal{O}(L^2d^2)$. With our hybrid scheme of backward propagation within the bounding component ($\partial_{\mathbf{x}\_i} u_\theta$ or $\partial_{\mathbf{x}^2\_i} u_\theta$) and forward substitution for elements from other components (e.g., $y^{(k)}$ in the bounding of $\partial_{\mathbf{x}\_i} u_\theta$, see Equation 10 in Appendix D.3), the complexity of bounding the output of each of these components remains $\mathcal{O}(L^2d^2)$. Following the McCormick envelope bounding described in Section 4.2, to estimate the final complexity we must now assume that the particular structure of f_θ will be linear lower and upper bounded as a function of R partial derivative components (of first or second order). For example, in the case of Burgers' equation, $R = 3$. The final complexity of bounding f_θ can then be written as $\mathcal{O}(RL^2d^2)$.

F. Correctness Certification for PINNs with $\tanh$ activations

∂-CROWN allows one to compute lower and upper bounds on the outputs of $\partial_{\mathbf{x}_i} u_\theta$, $\partial_{\mathbf{x}_i^2} u_\theta$ and f_θ as long as we can obtain linear bounds for u_θ's activations, σ, $\partial_{\mathbf{x}_i} u_\theta$'s activations, σ', and $\partial_{\mathbf{x}_i^2} u_\theta$'s activations, σ'', assuming previously computed bounds on the input of those activations. In this section we explore how to compute those bounds when u_θ has $\tanh$ activations.

Throughout, we assume the activation's input (y) is lower bounded by l_b and upper bounded by u_b (*i.e.*, $l_b \leq y \leq u_b$), and define the upper bound line as $h^U(y) = \alpha^U(y + \beta^U)$, and the lower bound line as $h^L(y) = \alpha^L(y + \beta^L)$. For the sake of brevity, we define for a function $h : \mathsf{R} \to \mathsf{R}$, and points $p, d \in \mathsf{R}$ the function $\tau(h, p, d) = {}^{(h(p)-h(d))}/_{(p-d)} - h'(d)$. This is useful as for a given h and p, if there exists a $d \in [d_l, d_u]$, such that $\tau_{d_l,d_u}(h, p, d) = 0$, then $h'(d)$ is the slope of a tangent line to h that passes through p and d.

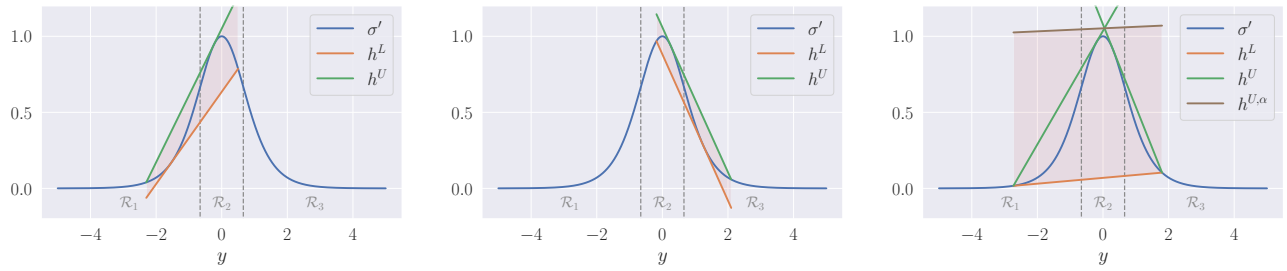
Bounding $\sigma(y) = \tanh(y)$ We follow the bounds provided in CROWN (Zhang et al., 2018), by observing that $\tanh$ is a convex function for $y < 0$ and concave for $y > 0$. For $l_b \leq u_b \leq 0$ we let h^U be the line that connects l_b and u_b, and for an arbitrary $d \in [l_b, u_b]$ we let h^L be the tangent line at that point. Similarly, for $0 \leq l_b \leq u_b$ we let h^L be the line that connects l_b and u_b, and for an arbitrary $d \in [l_b, u_b]$ we let h^U be the tangent line at that point. For the last case where $l_b \leq 0 \leq u_b$, we let h^U be the tangent line at $d_1 \geq 0$ that passes through $(l_b, \sigma(l_b))$, and h^L be the tangent line at $d_2 \leq 0$ that passes through $(u_b, \sigma(u_b))$. Given these bounds were given in Zhang et al. (2018), we omit visual representations of them.

Bounding $\sigma'(y) = 1 - \tanh^2(y)$ The derivative of $\tanh(y)$, $1 - \tanh^2(y)$, is a more complicated function. By inspecting it's derivative, $\sigma''(y) = -2\tanh(y)(1 - \tanh^2(y))$, we conclude that there are two inflection points at $y_1 = \max \sigma''(y)$ and $y_2 = \min \sigma''(y)$, leading to three different regions: $y \in]-\infty, y_1]$ ($\mathcal{R}_1$, the first convex region), $y \in]y_1, y_2]$ ($\mathcal{R}_2$, the concave region), and $y \in]y_2, +\infty[$ ($\mathcal{R}_3$, the second convex region). As a result, there are 6 combinations for the location of l_b and u_b which must be resolved.

The first two cases are the straightforward: if $l_b \in \mathcal{R}_1$ and $u_b \in \mathcal{R}_1$ or $l_b \in \mathcal{R}_3$ and $u_b \in \mathcal{R}_3$, *i.e.*, if both ends are in the same convex region, then we use the same relaxation as in the bounding of $\tanh$ in the convex region - h^U is the line that connects l_b and u_b, while h^L is a tangent line at a point $d \in [l_b, u_b]$. Similarly for the case where $l_b \in \mathcal{R}_2$ and $u_b \in \mathcal{R}_2$, we take the solution from the $\tanh$ concave side and use h^L to be the line that connects l_b and u_b, and h^U to be the tangent line at a point $d \in [l_b, u_b]$. The next case is $l_b \in \mathcal{R}_1$ and $u_b \in \mathcal{R}_2$, *i.e.*, l_b in the first convex region and u_b in the concave one. In this case we use the same bounding as in the $\tanh$ case when $l_b \leq 0 \leq u_b$: h^U is the tangent line at $d_1 \geq y_1$ that passes through $(l_b, \sigma'(l_b))$, and h^L is the tangent line at $d_2 \leq y_1$ that passes through $(u_b, \sigma'(u_b))$. In a similar fashion, for the case

Table 5: **Relaxing** $\sigma'(y) = 1 - \tanh^2(y)$: linear upper and lower bounds for a given l_b and u_b.

l_b	u_b	α^U	β^U	α^L	β^L
R_1 R_3	R_1 R_3	$(\sigma(u_b) \quad \sigma(l_b))/(u_b \quad l_b)$	$\sigma(l_b)/\alpha^U \quad l_b$	$\sigma'(d), d \in [l_b, u_b]$	$\sigma(d)/\alpha^L \quad d$
R_2	R_2	$\sigma'(d), d \in [l_b, u_b]$	$\sigma(d)/\alpha^U \quad d$	$(\sigma(u_b) \quad \sigma(l_b))/(u_b \quad l_b)$	$\sigma(l_b)/\alpha^L \quad l_b$
R_1	R_2	$\sigma'(d_1)$, $\tau_{y_1,u_b}(\sigma', l_b, d_1) = 0$	$\sigma(l_b)/\alpha^U \quad l_b$	$\sigma'(d_2)$, $\tau_{l_b,y_1}(\sigma', u_b, d_2) = 0$	$\sigma(u_b)/\alpha^L \quad u_b$
R_2	R_3	$\sigma'(d_1)$, $\tau_{l_b,y_2}(\sigma', u_b, d_1) = 0$	$\sigma(u_b)/\alpha^U \quad u_b$	$\sigma'(d_2)$, $\tau_{y_2,u_b}(\sigma', l_b, d_2) = 0$	$\sigma(l_b)/\alpha^L \quad l_b$
R_1	R_3	$\alpha\sigma'(d_1) + (1 \quad \alpha)\sigma'(d_2)$, $\tau_{l_b,0}(\sigma', l_b, d_1) = 0$, $\tau_{0,u_b}(\sigma', u_b, d_2) = 0$	$\alpha\beta_1^U + (1 \quad \alpha)\beta_2^U$, $\beta_1^U = \sigma(l_b)/\sigma'(d_1) \quad l_b$, $\beta_2^U = \sigma(u_b)/\sigma'(d_2) \quad u_b$	$\begin{cases} \sigma'(d_3), & l_b \quad u_b \\ \sigma'(d_4), & l_b < u_b \end{cases}$, $\tau_{l_b,y_1}(\sigma', u_b, d_3) = 0$, $\tau_{y_2,u_b}(\sigma', l_b, d_4) = 0$	$\begin{cases} \frac{\sigma'(u_b)}{\sigma'(d_3)} & u_b, \quad l_b \quad u_b \\ \frac{\sigma'(l_b)}{\sigma'(d_4)} & l_b, \quad l_b < u_b \end{cases}$



(a) $l_b \in R_1$ and $u_b \in R_2$ (b) $l_b \in R_2$ and $u_b \in R_3$ (c) $l_b \in R_1$ and $u_b \in R_3$

Figure 7: **Relaxing** $\sigma'(y) = 1 - \tanh^2(y)$: examples of the linear relaxations of σ' for different sets of l_b and u_b.

in which $l_b \in \mathcal{R}_2$ and $u_b \in \mathcal{R}_3$, *i.e.*, l_b in the concave region and u_b in the second convex region, we take the opposite approach: h^U is the tangent line at $d_1 \leq y_2$ that passes through $(u_b, \sigma'(u_b))$, and h^L is the tangent line at $d_2 \geq y_2$ that passes through $(l_b, \sigma'(l_b))$. These two cases are plotted in Figures 7a and 7b.

Finally, we tackle the case where $l_b \in \mathcal{R}_1$ and $u_b \in \mathcal{R}_3$, *i.e.*, where l_b is in the first convex region and u_b is in the second convex region. Given there is a concave region in between them, two valid upper bounds would be the ones considered previously for $l_b \in \mathcal{R}_1$ and $u_b \in \mathcal{R}_2$, and $l_b \in \mathcal{R}_2$ and $u_b \in \mathcal{R}_3$. To obtain these bounds, we shift the upper bound in the first case to 0, and the lower bound in the second case to 0 (see h^U in Figure 7c). As our bounding requires a single h^U, we take a convex combination of the two bounds obtained, $h^{U,\alpha}$. For the lower bound, we use a line that passes by either $(u_b, \sigma'(u_b))$, if $-l_b \geq u_b$, or by $(l_b, \sigma'(l_b))$, otherwise, as well as by a tangent point $d_3 \in \mathcal{R}_1$, if $-l_b \geq u_b$, or by $d_4 \in \mathcal{R}_3$, otherwise. See the line $h^{U,\alpha}$ in Figure 7c for a visual representation.

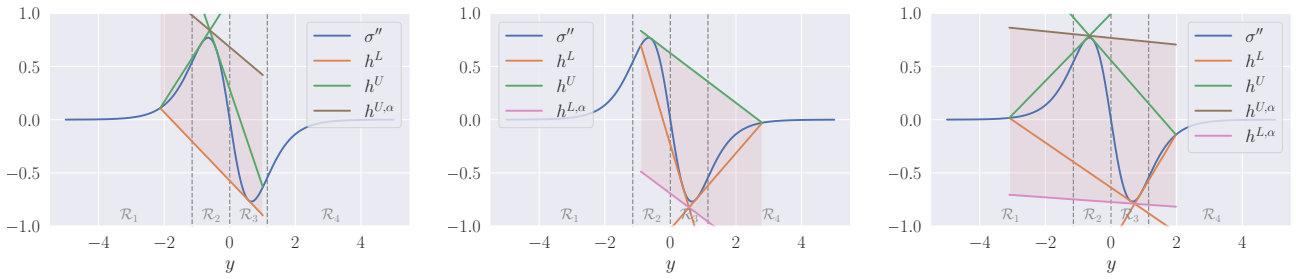
Bounding $\sigma''(y) = -2\tanh(y)\left(1 - \tanh(y)^2\right)$ By inspecting the derivative of σ'', $\sigma''(y) = -2 + 8\tanh^2(y) - 6\tanh^4(y)$, we conclude there are three inflection points for this function, one at $y_1 = \arg\max_{y \leq 0} \sigma'''(y)$, another at $y_2 = 0$, and finally at $y_3 = -y_1$. Take also, for the sake of bounding, $y_{\max} = \arg\max_{y \leq 0} \sigma''(y)$ and $y_{\min} = \arg\min_{y \leq 0} \sigma''(y)$. This leads to four different regions of σ'': $y \in]-\infty, y_1]$ ($\mathcal{R}_1$, the first convex region), $y \in]y_1, y_2]$ ($\mathcal{R}_2$, the first concave region), $y \in]y_2, y_3]$ ($\mathcal{R}_3$, the second convex region), and $y \in]y_3, +\infty[$ ($\mathcal{R}_4$, the second concave region). This leads to 10 combinations for the location of l_b and u_b.

The first four are straightforward: if $l_b \in \mathcal{R}_i$ and $u_b \in \mathcal{R}_i$ for $i \in \{1, \ldots, 4\}$, then we use exactly the same approximations as for σ and σ'', varying only based on the convexity of $\mathcal{R}_i$. Similarly, if $l_b \in \mathcal{R}_i$ and $u_b \in \mathcal{R}_{i+1}$ for $i \in \{1, 2, 3\}$, then we are also in the same situation as the adjacent regions of different convexity from σ', so we use exactly the same relaxation.

We are left with three cases where l_b and u_b are in non-adjacent regions. For $l_b \in \mathcal{R}_1$ and $u_b \in \mathcal{R}_3$, we are in the same scenario as in the bounding of σ', since $\mathcal{R}_1$ and $\mathcal{R}_3$ are convex regions separated by a concave one. In that case we follow the bounding procedure outlined before for σ' - see Figure 8a for an example of it applied in this setting. For the case where

Table 6: **Relaxing** $\sigma''(y) = -2\tanh(y)\left(1-\tanh^2(y)\right)$: linear upper and lower bounds for a given l_b and u_b.

l_b	u_b	α^U	β^U	α^L	β^L
R_1 R_3	R_1 R_3	$(\sigma''(u_b) \quad \sigma''(l_b))/(u_b \quad l_b)$	$\sigma''(l_b)/\alpha^U \quad l_b$	$\sigma'''(d),\ d\ 2\ [l_b, u_b]$	$\sigma''(d)/\alpha^L \quad d$
R_2 R_4	R_2 R_4	$\sigma'''(d),\ d\ 2\ [l_b, u_b]$	$\sigma''(d)/\alpha^U \quad d$	$(\sigma''(u_b) \quad \sigma''(l_b))/(u_b \quad l_b)$	$\sigma''(l_b)/\alpha^L \quad l_b$
R_1	R_2	$\sigma'''(d_1),$ $\tau_{y_1,u_b}(\sigma'', l_b, d_1) = 0$	$\sigma(l_b)/\alpha^U \quad l_b$	$\sigma'''(d_2),$ $\tau_{l_b,y_1}(\sigma'', u_b, d_2) = 0$	$\sigma''(u_b)/\alpha^L \quad u_b$
R_3	R_4	$\sigma'''(d_1),$ $\tau_{y_3,u_b}(\sigma'', l_b, d_1) = 0$	$\sigma''(l_b)/\alpha^U \quad l_b$	$\sigma'''(d_2),$ $\tau_{l_b,y_3}(\sigma'', u_b, d_2) = 0$	$\sigma''(u_b)/\alpha^L \quad u_b$
R_2	R_3	$\sigma'''(d_1),$ $\tau_{l_b,y_2}(\sigma'', u_b, d_1) = 0$	$\sigma''(u_b)/\alpha^U \quad u_b$	$\sigma'''(d_2),$ $\tau_{y_2,u_b}(\sigma'', l_b, d_2) = 0$	$\sigma''(l_b)/\alpha^L \quad l_b$
R_1	R_3	$\alpha\sigma'''(d_1) + (1 \quad \alpha)\sigma'''(d_2),$ $\tau_{l_b,y_{\max}}(\sigma'', l_b, d_1) = 0,$ $\tau_{y_{\max},u_b}(\sigma'', u_b, d_2) = 0$	$\alpha\beta_1^U + (1 \quad \alpha)\beta_2^U,$ $\beta_1^U = \sigma''(l_b)/\sigma'''(d_1) \quad l_b,$ $\beta_2^U = \sigma''(u_b)/\sigma'''(d_2) \quad u_b$	$\sigma'''(d_3),$ $\tau_{y_1,u_b}(\sigma', l_b, d_3) = 0$	$\sigma''(l_b)/\alpha^L \quad l_b$
R_2	R_4	$\sigma'''(d_1),$ $\tau_{l_b,y_2}(\sigma', u_b, d_1) = 0$	$\sigma''(u_b)/\alpha^U \quad u_b$	$\alpha\sigma'''(d_2) + (1 \quad \alpha)\sigma'''(d_3),$ $\tau_{l_b,y_{\min}}(\sigma'', l_b, d_2) = 0,$ $\tau_{y_{\min},u_b}(\sigma'', u_b, d_3) = 0$	$\alpha\beta_1^L + (1 \quad \alpha)\beta_2^L,$ $\beta_1^L = \sigma''(l_b)/\sigma'''(d_2) \quad l_b,$ $\beta_2^L = \sigma''(u_b)/\sigma'''(d_3) \quad u_b$
R_1	R_4	$\alpha\sigma'''(d_1) + (1 \quad \alpha)\sigma'''(d_2),$ $\tau_{l_b,y_{\max}}(\sigma'', l_b, d_1) = 0,$ $\tau_{y_{\max},u_b}(\sigma'', u_b, d_2) = 0$	$\alpha\beta_1^U + (1 \quad \alpha)\beta_2^U,$ $\beta_1^U = \sigma''(l_b)/\sigma'''(d_1) \quad l_b,$ $\beta_2^U = \sigma''(u_b)/\sigma'''(d_2) \quad u_b$	$\alpha\sigma'''(d_3) + (1 \quad \alpha)\sigma'''(d_4),$ $\tau_{l_b,y_{\min}}(\sigma'', l_b, d_3) = 0,$ $\tau_{y_{\min},u_b}(\sigma'', u_b, d_4) = 0$	$\alpha\beta_1^L + (1 \quad \alpha)\beta_2^L,$ $\beta_1^L = \sigma''(l_b)/\sigma'''(d_3) \quad l_b,$ $\beta_2^L = \sigma''(u_b)/\sigma'''(d_4) \quad u_b$



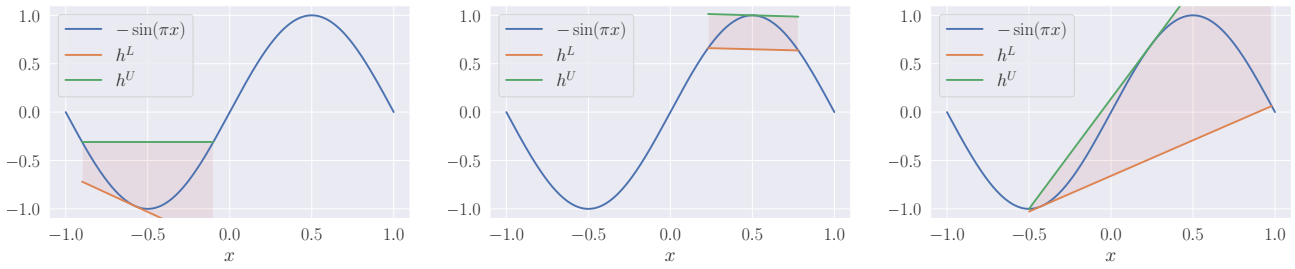
(a) l_b 2 R_1 and u_b 2 R_3 (b) l_b 2 R_2 and u_b 2 R_4 (c) l_b 2 R_1 and u_b 2 R_4

Figure 8: **Relaxing** $\sigma''(y) = -2\tanh(y)\left(1-\tanh^2(y)\right)$: examples of the linear relaxations of σ'' for different sets of l_b and u_b.

$l_b \in \mathcal{R}_2$ and $u_b \in \mathcal{R}_4$, we are in an analogous case where $\mathcal{R}_2$ and $\mathcal{R}_4$ are concave regions separated by a convex one. As such, we consider the two valid lower bounds computed previously for $l_b \in \mathcal{R}_2$ and $u_b \in \mathcal{R}_3$, and $l_b \in \mathcal{R}_3$ and $u_b \in \mathcal{R}_4$. To obtain these bounds, we shift the upper bound in the first case to $\arg\min \sigma''(y)$, and the lower bound in the same case to the same value (see h^L in Figure 8b). As our bounding requires a single h^L, we take a convex combination of the two bounds, $h^{L,\alpha}$. For the upper bound, we simply assume l_b is in a concave region while u_b is in a convex region, and take the tangent at d for $\arg\max \sigma''(y) \geq d \leq 0$ (see h^U in Figure 8b). Finally, we are left with the case where $l_b \in \mathcal{R}_1$ and $u_b \in \mathcal{R}_4$. In that case, we take the upper bound lines from the case where $l_b \in \mathcal{R}_1$ and $u_b \in \mathcal{R}_3$, and the lower bound ones from where $l_b \in \mathcal{R}_2$ and $u_b \in \mathcal{R}_4$. As before, given the requirement of one lower and upper bound functions, we take a convex combination of both in $h^{L,\alpha}$ and $h^{U,\alpha}$, respectively. See Figure 8c for a visual representation.

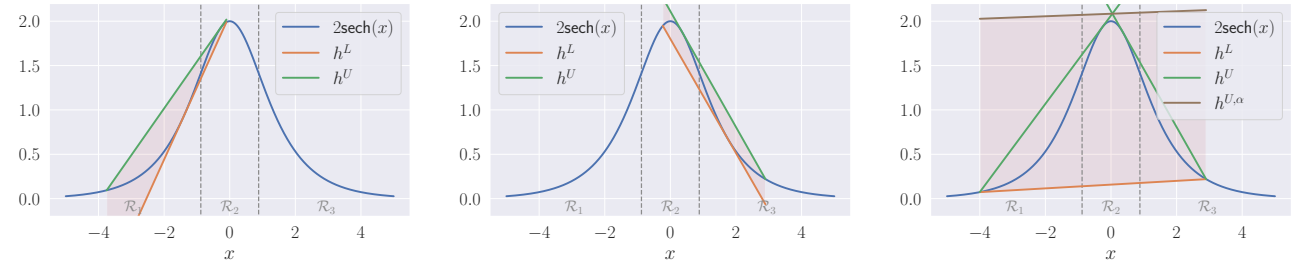
Table 7: **Ablation on our relaxations for the derivatives of** tanh: comparison of the residual upper bounds on Burgers' equation obtained using our relaxations in ∂-CROWN vs using a simple baseline which takes the minimum area in the convex/concave regions and a constant value elsewhere with a time limit of 10^4s.

	Our relaxations u_b		**Simple baseline** u_b	
$jf_\theta(\mathbf{x})j^2$	1.30	10^1	4.34	10^2



(a) l_b 0 and u_b 0 (b) l_b 0 and u_b 0 (c) l_b 0 and u_b 0

Figure 9: **Relaxing** $-\sin(\pi x)$: examples of the linear relaxations for different sets of l_b and u_b.



(a) l_b 0 and u_b 0 (b) l_b 0 and u_b 0 (c) l_b 0 and u_b 0

Figure 10: **Relaxing** 2**sech**(x): examples of the linear relaxations for different sets of l_b and u_b.

F.1. Ablation on σ' and σ'' relaxations for $\tanh$

To understand the effectiveness of proposed the proposed relaxations for σ'' and σ'' for the case of $\tanh$, we compare the performance of ∂-CROWN in bounding the residual of Burgers' equation (with the fixed time limit of 10^4s from Section 5.3), using **Our relaxations** for σ' and σ'', as well as a **Simple baseline** which takes the minimum area in the convex/concave sections and a constant elsewhere. For $\tanh$, both use the same relaxation from Zhang et al. (2018). The comparison results are presented in the Table 7. The tightness difference showcases the efficacy of our proposed nonlinearity relaxations for σ' and σ'' for $\tanh$ activations.

G. Linear lower and upper bounding nonlinear functions

Throughout, we assume the function's input (x) is lower bounded by l_b and upper bounded by u_b (*i.e.*, $l_b \leq x \leq u_b$), and define the upper bound line as $h^U(x) = \alpha^U(x + \beta^U)$, and the lower bound line as $h^L(x) = \alpha^L(x + \beta^L)$. For the sake of brevity, we define for a function $h : \mathsf{R} \to \mathsf{R}$, and points $p, d \in \mathsf{R}$ the function $\tau(h, p, d) = {}^{(h(p)-h(d))}/_{(p-d)} - h'(d)$. This is useful as for a given h and p, if there exists a $d \in [d_l, d_u]$, such that $\tau_{d_l,d_u}(h, p, d) = 0$, then $h'(d)$ is the slope of a tangent line to h that passes through p and d.

G.1. Case study: $-\sin(\pi x)$ for $x \in [-1, 1]$

As in Appendix F, we observe the convexity of the function $-\sin(\pi x)$ for $x \in [-1, 1]$, noticing that the function is convex for $x \leq 0$ and concave for $x \geq 0$. For $l_b \leq u_b \leq 0$ we let h^U be the line that connects l_b and u_b, and for an arbitrary $d \in [l_b, u_b]$ we let h^L be the tangent line at that point. Similarly, for $0 \leq l_b \leq u_b$ we let h^L be the line that connects l_b and u_b, and for an arbitrary $d \in [l_b, u_b]$ we let h^U be the tangent line at that point. For the last case where $l_b \leq 0 \leq u_b$, we let h^U be the tangent line at $d_1 \geq 0$ that passes through $(l_b, \sigma(l_b))$, and h^L be the tangent line at $d_2 \leq 0$ that passes through $(u_b, \sigma(u_b))$. Given the similarity of to the $\tanh$ bounds from Zhang et al. (2018), we omit a summary table, but present 3 examples of the possible cases in Figure 9.

G.2. Case study: $2\text{sech}(x)$ for $x \in [-5, 5]$

We start by observing that the function $2\text{sech}(x)$ is similar to the derivative of $\tanh$, whose relaxation we presented in Appendix F. By inspecting it's derivative, $f'(x) = 2\text{sech}(x)\tanh(x)$, we conclude that there are two inflection points at $x_1 = \max f'(x)$ and $x_2 = \min f'(x)$, leading to three different regions: $x \in]-\infty, x_1]$ ($\mathcal{R}_1$, the first convex region), $x \in]x_1, x_2]$ ($\mathcal{R}_2$, the concave region), and $x \in]x_2, +\infty[$ ($\mathcal{R}_3$, the second convex region). As a result, there are 6 combinations for the location of l_b and u_b which must be resolved. This is exactly the same case as the first derivative of $\tanh$, simply with x_1 and x_2 instead of y_1 and y_2. Due to the similarities, we can use exactly the same relaxations as presented in Table 5. We present visual examples of 3 cases of this relaxation in Figure 10.

H. Further details on Greedy Input Branching

In Section 4.3 we motivated and described at a high-level greedy input branching. In the following we provide a step-by-step analysis of Algorithm 1.

We start by initializing a lower and upper bound list of pairs $\mathcal{B}$ (line 3) as well as a list for storing the maximum error between the empirical and certified bounds $\mathcal{B}_\Delta$ (line 4). To initialize them (line 7 and 8), we first compute the empirical lower and upper bounds across the domain by sampling N_s points within the full domain $\mathcal{C}$ using $\text{SAMPLE}(\mathcal{C}, N_s)$ and evaluating the function h there (line 5) yielding $\hat{h}_{lb}$ and $\hat{h}_{ub}$, as well as the first version of the certified lower and upper bounds using ∂-CROWN on h (line 6) yielding h_{lb}, h_{ub}. Next, we pop from $\mathcal{B}$ and $\mathcal{B}_\Delta$ as C_i the interval which has the maximum error between the empirical and certified bounds (line 10), which we then proceed to split into N_d parts following a policy defined by DOMAINSPLIT (line 11). Importantly, DOMAINSPLIT must be complete, i.e., it must be that $C_i = \cup C'$. For each of those split subdomains $\mathcal{C}'$ we compute new bounds using ∂-CROWN (line 12) and add this subdomain along with its bounds and error to the empirical estimates to $\mathcal{B}$ and $\mathcal{B}_\Delta$, respectively (line 13 and 14). This process is repeated using the updated lists until the branching budget is spent, at which point the global lower bound is the minimum of all of lower bounds in $\mathcal{B}$ (defined as the list $\mathcal{B}_0$), and the global upper bound is the maximum of all upper bounds in $\mathcal{B}$ (defined as the list $\mathcal{B}_1$). These are computed in line 17. This algorithm is greedy as increasing the branching budget is expected to improve the bounds, since ∂-CROWN 's bounds are guaranteed to monotonically decrease with smaller input domains.

I. On Extending ∂-CROWN to higher-order PDEs

In this section we explore the potential of applying ∂-CROWN to higher-order PDEs. We divide the analysis into the theory and experimental challenges, and how these could be mitigated.

Theory. For the purposes of this paper, we only derive first and second partial derivative bounds, yet there is nothing that theoretically limits our method to second-order PDEs. The extension of the theory to third-order PDEs is relatively straightforward, consisting of applying the chain rule to Lemma 2, and following the same backward-forward mechanism in the proof of Theorem 2 (Appendix D.4). We acknowledge that extending it to higher order PDEs leads to a growing computational graph, which can be more difficult to derive.

Experiments. It is likely that the obtained bounds with extensions of ∂-CROWN to higher-order PDEs will be looser due to the growth of the computational graph. However, it is possible to mitigate these issues by designing (i) tighter nonlinearity relaxations and (ii) more efficient branching methods than our greedy branching one.

We perform a qualitative analysis of the greedy branching strategy in Section 5.4, and present in Table 8 the ∂-CROWN u_b difference between using a simple uniform strategy and our greedy input branching in Burgers' equation given a fixed number number of branchings that leads to approximately the same runtime for both methods (10^4s). This highlights the importance of input branching in achieving tight bounds in second-order PINNs. With more efficient branching strategies, such as asymmetrical branching using sampling or through learning following a similar idea to Balcan et al. (2018), these could be significantly improved and applied to higher-order PINNs.

Table 8: **Efficiency of Greedy Input Branching**: comparing greedy input branching to uniform branching in Burgers' equation given an approximate runtime limit of 10^4s in both cases.

	Uniform branching ($N_b = 1.6 \times 10^5$)	Greedy input branching ($N_b = 1.3 \times 10^5$)
$\lvert f_\theta(x,t)\rvert^2$	1.51×10^2	1.30×10^1