Chapter 1

Proposal

The general form of Laplace's equation is as follows. Let $n \geqslant 2$, $\Omega \subseteq \mathbb{R}^n$ a connected, open, bounded Lipschitz domain and $g\colon \partial\Omega \to \mathbb{R}$. Consider Laplace's equation

$$\left.\begin{aligned} \Delta u(\mathbf{x}) &= 0 \qquad (\mathbf{x} \in \Omega), \\ u(\mathbf{x}) &= g(\mathbf{x}) \qquad (\mathbf{x} \in \partial\Omega). \end{aligned}\right\} \tag{1.1}$$

A well known fact from the theory of partial differential equations is that the fundamental solution for the operator $(-\Delta)$ – i.e. the solution to the equation $-\Delta u = \delta_0$ – is:

$$E_n(\mathbf{x}) \coloneqq \begin{cases} -\frac{1}{2\pi} \ln \|\mathbf{x}\|_2, & \text{for } n = 2, \quad \mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}, \\ \frac{1}{(n-2)S_n \|\mathbf{x}\|_2^{n-2}}, & \text{for } n \geqslant 3, \quad \mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}, \end{cases} \tag{1.2}$$

where S_n is the surface area of the n dimensional unit ball.

Article [2] shows a working method to find the solution to the equation (1.1) for a non-trivial $\Omega \subseteq \mathbb{R}^2$ in a couple points $\{\mathbf{x_0}, \ldots, \mathbf{x_{I-1}}\}$ $(I \in \mathbb{N}^+)$, which is as follows:

- take a number of boundary points $\{\mathbf{z_0}, \ldots, \mathbf{z_{K-1}}\} \subset \partial\Omega$, $(K \in \mathbb{N}^+)$,
- and a number of exterior points $\{\mathbf{w_0}, \ldots, \mathbf{w_{L-1}}\} \subset \operatorname{ext}\Omega$ $(L \in \mathbb{N}^+)$.
- Consider the functions $G_{\mathbf{w_l}}(\mathbf{x}) \coloneqq E_2(\mathbf{x} - \mathbf{w_l})$, and
- take a one-layer dense NN whose training set is

$$\Big\{ \underbrace{(G_{\mathbf{w_l}}(\mathbf{z_0}), G_{\mathbf{w_l}}(\mathbf{z_1}), \ldots, G_{\mathbf{w_l}}(\mathbf{z_{K-1}}))}_{\text{input}\coloneqq\xi_\mathbf{l}}, \underbrace{(\Psi_{\mathbf{w_l}}(\mathbf{x_0}), \Psi_{\mathbf{w_l}}(\mathbf{x_1}), \ldots, \Psi_{\mathbf{w_l}}(\mathbf{x_{I-1}}))}_{\text{output}\coloneqq\eta_\mathbf{l}} \Big\}_{l=0}^{L-1}.$$

  That is, the goal is to find a map $\tilde{u}\colon \mathbb{R}^K \to \mathbb{R}^I$ for which $\tilde{u}(\xi_\mathbf{l}) = \eta_\mathbf{l}$ for $l \in \{0, \ldots, L-1\}$

- This is possible using a NN, which will give a reasonably accurate estimation of the values. $\{u(\mathbf{x_i})\}_{i=0}^{I-1}$ given the values $\{g(\mathbf{z_k})\}_{k=0}^{K-1}$ on the boundary in the following way:

$$\tilde{u}\big(\underbrace{g(\mathbf{z_0}), g(\mathbf{z_1}) \ldots, g(\mathbf{z_{K-1}})}_{\text{given boundary values}}\big) \approx \big(\underbrace{u(\mathbf{x_0}), u(\mathbf{x_1}) \ldots, u(\mathbf{x_{I-1}}))}_{\text{approximated values}}\big).$$

In short, this NN is able to learn what the values of a fundamental solution should be in the interior points $\{\mathbf{x_i}\}_{i=0}^{I-1}$, given the function values in the boundary points $\{\mathbf{z_0}, \ldots, \mathbf{z_{K-1}}\}$. From that, one may estimate the solution in the desired points.

Remark 1.1. The indexes run from 0 specifically to be congruent with the notation in the implementation, which is done in the programming language python.

Our first goal is to modify this model to solve the more general Poisson's equation: Given a function $f\colon \Omega \to \mathbb{R}$, solve

$$\left.\begin{aligned} \Delta u(\mathbf{x}) &= f(\mathbf{x}) \qquad (\mathbf{x} \in \Omega), \\ u(\mathbf{x}) &= g(\mathbf{x}) \qquad (\mathbf{x} \in \partial\Omega). \end{aligned}\right\} \tag{1.3}$$

A couple of apriori thoughts:

- it is obvious that the model needs to take into account some values of f inside the domain
- we will probably have to introduce some new functions Ψ for which $\Delta\Psi \neq 0$.
- It is not at all trivial as to what class of functions to pick here. Intuitively, a good candidate for the class of functions to consider could be radial functions, as the differential operator Δ is invariant for orthogonal transformations.
- For the sake of numerical stability, the singularities of all functions – if any – should be in $\operatorname{ext}\Omega$, just like in the original model.
- The span of $\{\Delta\Psi_\alpha\}_{\alpha\in A}$ (for some countable index set A) should be dense in some norm – for instance the $\|.\|_{W^2}$ norm – so that we have a chance to generalize the method for a sufficiently large set of possible f RHS functions. This is a very hard theoretical property, whose verification might fall out of the scope of this thesis.

Such a well behaved family of functions is currently unknown. Article [] proposes functions of the form $\Psi(r) =$ for similar applications.

It also might prove useful in decreasing the numerical cost of these calculations if we change the structure of the NN from a dense connection to a convolutional one. The intuition behind this is that both the laplacian and the function itself are influenced by its local properties more than its global properties. We might want to experiment with multi-layer constructions and different activation functions. However, these options don't seem to offer any obvious benefits intuitively, they might still prove useful in practice.

This thesis will explore the feasibility of the following approaches:

- Modify said linear NN model by including information from the interior of the domain.
- How to structure the data such that the geometry of the problem is preserved and utilized: experiment with convolutional NN structures.
- Testing and researching in the literature the different possible layer structures, activation function and bias choices suitable for this task.
- Experimenting with different training strategies: variable learning rate, different optimizer methods etc.
- Create a trained model that is as generalized as possible in the following directions: different f and g choices, different choices for Ω and possibly even different choices for differential operators. However, this later task may very well be beyond the scope of this thesis.

Chapter 2

Estimating with a linear NN structure

Let us change the training set and the testing set of the model described in Chapter 1 as follows. Let $\Omega \subseteq \mathbb{R}^n$ be a bounded open Lipschitz-domain. Let

- $X = \{\mathbf{x_i}\}_{i=0}^{I-1} \subset \Omega$ be a set of a few interior points where the solution is to be approximated,
- $Y = \{\mathbf{y_j}\}_{j=0}^{J-1} \subset \overline{\Omega}$ be basis points of
- $\Psi_{\mathbf{y_j}}(\mathbf{x}) \coloneqq \Psi(\mathbf{x} - \mathbf{y_j})$ radial functions,
- $\{\mathbf{z_k}\}_{k=0}^{K-1} \subset \partial\Omega$ boundary points, and finally
- $\{\mathbf{w_l}\}_{l=0}^{L-1} \subset \operatorname{ext}\Omega$ a number of exterior points, which serve as a center of the Green functions
- $G_{\mathbf{w_l}}(\mathbf{x}) = E_n(\mathbf{x} - \mathbf{w_l})$.

Remark 2.1. The restriction that $\mathbf{w_l}$ be exterior points of Ω is required due to the fact that $G_{\mathbf{w_l}}(\mathbf{x})$ has a singularity at $\mathbf{x} = \mathbf{w_l}$.

Remark 2.2. An argument for the inclusion of X in the set T is the following. Assume that we have $u, \tilde{u} \in W^{2,2}(\Omega)$ such that for all $\mathbf{x_i} \in X$ we have $u(\mathbf{x_i}) = \tilde{u}(\mathbf{x_i})$, but $u \neq \tilde{u}$ on Ω. In this case, the algorithm wouldn't know which function to approximate. This might not cause too great fluctuations, but it's worth trying to avoid this issue.

Introduce the notation $T = X \cup Y \coloneqq \{\mathbf{t_m}\}_{m=0}^{M-1}$. Let the elements of the training set be the following:

I Belonging to the points $\mathbf{y_j}$ – for every $j \in \{0, 1, \ldots, J-1\}$ – an input-output pair:

$$\underbrace{(\Delta\Psi_{\mathbf{y_j}}(T), \Psi_{\mathbf{y_j}}(Z))}_{\in\mathbb{R}^{M+K} \text{ input}} \to \underbrace{\Psi_{\mathbf{y_j}}(X)}_{\in\mathbb{R}^{I} \text{ output}},$$

II and belonging to the points $\mathbf{w_l}$ – for every $l \in \{0, 1, \ldots, L-1\}$ – an input-output pair:

$$\underbrace{(\mathbf{0}, G_{\mathbf{w_l}}(Z))}_{\in\mathbb{R}^{M+K} \text{ input}} \to \underbrace{G_{\mathbf{w_l}}(X)}_{\in\mathbb{R}^{I} \text{ output}}.$$

This amounts to a total number of $J + L$ input-output pairs.

Now let the the test set take the form of

$$((f(T)), (g(Z))) \in \mathbb{R}^{M+K}.$$

For the sake of clarity, it is perhaps worth taking a look at the training inputs and outputs once more as matrices. The input matrix takes the form

$$\begin{pmatrix} \Delta\Psi_{\mathbf{y_0}}(\mathbf{t_0}) & \Delta\Psi_{\mathbf{y_0}}(\mathbf{t_1}) & \dots & \Delta\Psi_{\mathbf{y_0}}(\mathbf{t_{M-1}}) & \Psi_{\mathbf{y_0}}(\mathbf{z_0}) & \Psi_{\mathbf{y_0}}(\mathbf{z_1}) & \dots & \Psi_{\mathbf{y_0}}(\mathbf{z_{K-1}}) \\ \Delta\Psi_{\mathbf{y_1}}(\mathbf{t_0}) & \Delta\Psi_{\mathbf{y_1}}(\mathbf{t_1}) & \dots & \Delta\Psi_{\mathbf{y_1}}(\mathbf{t_{M-1}}) & \Psi_{\mathbf{y_1}}(\mathbf{z_0}) & \Psi_{\mathbf{y_1}}(\mathbf{z_1}) & \dots & \Psi_{\mathbf{y_1}}(\mathbf{z_{K-1}}) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \Delta\Psi_{\mathbf{y_{J-1}}}(\mathbf{t_0}) & \Delta\Psi_{\mathbf{y_{J-1}}}(\mathbf{t_1}) & \dots & \Delta\Psi_{\mathbf{y_{J-1}}}(\mathbf{t_{M-1}}) & \Psi_{\mathbf{y_{J-1}}}(\mathbf{z_0}) & \Psi_{\mathbf{y_{J-1}}}(\mathbf{z_1}) & \dots & \Psi_{\mathbf{y_{J-1}}}(\mathbf{z_{K-1}}) \\ 0 & 0 & \dots & 0 & G_{\mathbf{w_0}}(\mathbf{z_0}) & G_{\mathbf{w_0}}(\mathbf{z_1}) & \dots & G_{\mathbf{w_0}}(\mathbf{z_{K-1}}) \\ 0 & 0 & \dots & 0 & G_{\mathbf{w_1}}(\mathbf{z_0}) & G_{\mathbf{w_1}}(\mathbf{z_1}) & \dots & G_{\mathbf{w_1}}(\mathbf{z_{K-1}}) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & G_{\mathbf{w_{L-1}}}(\mathbf{z_0}) & G_{\mathbf{w_{L-1}}}(\mathbf{z_1}) & \dots & G_{\mathbf{w_{L-1}}}(\mathbf{z_{K-1}}) \end{pmatrix},$$

while the output matrix looks as follows

$$\begin{pmatrix} \Psi_{\mathbf{y_0}}(\mathbf{x_0}) & \Psi_{\mathbf{y_0}}(\mathbf{x_1}) & \dots & \Psi_{\mathbf{y_0}}(\mathbf{x_{I-1}}) \\ \Psi_{\mathbf{y_1}}(\mathbf{x_0}) & \Psi_{\mathbf{y_1}}(\mathbf{x_1}) & \dots & \Psi_{\mathbf{y_1}}(\mathbf{x_{I-1}}) \\ \vdots & \vdots & \ddots & \vdots \\ \Psi_{\mathbf{y_{J-1}}}(\mathbf{x_0}) & \Psi_{\mathbf{y_{J-1}}}(\mathbf{x_1}) & \dots & \Psi_{\mathbf{y_{J-1}}}(\mathbf{x_{I-1}}) \\ G_{\mathbf{w_0}}(\mathbf{x_0}) & G_{\mathbf{w_0}}(\mathbf{x_1}) & \dots & G_{\mathbf{w_0}}(\mathbf{x_{I-1}}) \\ G_{\mathbf{w_1}}(\mathbf{x_0}) & G_{\mathbf{w_1}}(\mathbf{x_1}) & \dots & G_{\mathbf{w_1}}(\mathbf{x_{I-1}}) \\ \vdots & \vdots & \ddots & \vdots \\ G_{\mathbf{w_{L-1}}}(\mathbf{x_0}) & G_{\mathbf{w_{L-1}}}(\mathbf{x_1}) & \dots & G_{\mathbf{w_{L-1}}}(\mathbf{x_{I-1}}) \end{pmatrix}$$

It should be noted that the desired map is between $\mathbb{R}^{M+K}$ and $\mathbb{R}^I$, however it is not obvious whether it is linear or not. A first guess could be that it is, and that we should try to find a suitable map in $\mathcal{L}(\mathbb{R}^{M+K}, \mathbb{R}^I)$. Since $\dim\left(\mathcal{L}(\mathbb{R}^{M+K}, \mathbb{R}^I)\right) = (M+K)\cdot I$, our first attempt should have at least this many adjustable parameters. For this purpose, a fully connected network with no biases and linear activation functions should suffice.

2.1 Preliminary results for the one layer linear model

At this stage the linear model attempts to fit a $\mathcal{L}(\mathbb{R}^{M+K}, \mathbb{R}^I)$ map onto $J + L$ data points. For this to be reasonably efficient and quick, we suspect that $J + L$ and $(M + K) \cdot I$ should be roughly equal.

Some of the numerical experiments shall be included in the presentation, however, further fine tuning of the parameters is still much needed.