Differentiable Simulations for Joint Parameterised Optimisation of Antenna Arrays

Niels Skovgaard Jensen
TICRA
Technical University of Denmark
Copenhagen, Denmark
nsje@dtu.dk

Frederik Faye
TICRA
Copenhagen, Denmark
ff@ticra.com

Lasse Hjuler Christiansen
TICRA
Copenhagen, Denmark
lhch@ticra.com

Allan Peter Ensig-Karup
Technical University of Denmark
Kongens Lyngby, Denmark
apek@dtu.dk

Abstract

Many modern antenna systems rely on active electronically scanned array technology to reconfigure antenna systems in real-time. The scanning of arrays requires correct array excitation coefficients. To obtain excitation coefficients that satisfy an ample space of reconfigurability, a considerable number of inverse problems usually need to be solved. This is either done with heuristics, optimisers or through costly physical measurements. In this work, we discuss our experience with implementing a batchable and differentiable antenna array simulation tool and utilising it in combination with neural networks to solve large-scale excitation synthesis using joint parametric optimisation. The method enables us to train neural networks to solve a wide range of beam-shaping problems in an end-to-end manner. We have observed that the method is competitive with direct optimisation methods when a large number of problems need to be solved. However, the current approach does not yet achieve the same solution quality as a well-tuned direct optimisation method.

1 Introduction

A time-harmonic electromagnetic far-field from an antenna can be described as a complex 2-D vector field on a sphere [1]:

$$\boldsymbol{E}(\theta,\phi) = \mathbf{a}_{co}E_{co}(\theta,\phi) + \mathbf{a}_{cross}E_{cross}(\theta,\phi). \tag{1}$$

Here $\theta \in [0, 2\pi[$, $\phi \in [0, \pi[$, $E_{co}, E_{cross} \in \mathbb{C}$, and $\mathbf{a}_{co}$ and $\mathbf{a}_{cross}$ are orthornormal unit vectors dependent on the choice of reference polarisation of the field with the co-polar component being the desired transmitting or receiving polarization.

Multiple antennas can be spatially arranged into an *antenna array*. In that case, the total electromagnetic field is a superposition of the fields of the individual elements, $\boldsymbol{E}_e$, on the form of Equation 1, multiplied by a spatial phase term [2]:

$$\boldsymbol{E}_t(\theta,\phi) = \sum_{n=1}^{N} w_n \boldsymbol{E}_{e,n}(\theta,\phi) \exp\left(j\frac{2\pi}{\lambda}\boldsymbol{r}_n \cdot \boldsymbol{k}(\theta,\phi)\right). \tag{2}$$

Here λ is the wavelength, $\{\boldsymbol{r}_n\}_{n=1}^{N}, \boldsymbol{r}_n \in \mathbb{R}^3$ are the Cartesian positions of the elements, $\boldsymbol{k}(\theta,\phi) \in \mathbb{R}^3$ is a unit vector pointing in direction (θ,ϕ), and $\{w_n\}_{n=1}^{N} \in \mathbb{C}^N$ are the *excitation coefficients* of each of the antenna elements. The embedded element patterns, $\{\boldsymbol{E}_{e,n}\}_{n=1}^{N}$ are highly dependent on antenna design.

Antenna arrays are useful since they can significantly increase the *directivity* of the signal being transmitted, making communications possible over longer distances. Furthermore with sufficiently advanced electronics, they can be used as *active electronically scanned arrays* (AESA)[3], which can change direction of transmission and reception without any physical alterations to the array; even shaping the beampattern of the array $\mathbf{E}_t(\theta, \phi)$ as desired. This is done purely through the choice of the excitation coefficients, $\boldsymbol{w} = \{w_n\}_{n=1}^{N}$. Usually, the excitation coefficients are constrained to be phase-only, i.e. $|w_n| = 1 \ \forall \ n$.

The inverse problem of finding a set of excitation coefficients to fulfil a desired $E_t(\theta, \phi)$ has historically been addressed using heuristic methods or iterative optimisation algorithms [4], one problem at a time. Due to the desire for reconfigurability, it is common to optimise for a large number of beam patterns with different directions and specifications to utilise the array's reconfigurability, and then switch between these solutions through a lookup table. Recent developments and increased interest in machine learning have led to numerous works using supervised learning on databases of pre-optimised excitation coefficients [5], [6], [7], which often require the costly process of pre-solving a large number of array synthesis problems for data generation.

Contrary to a data-driven method, this work uses a batchable and differentiable antenna array simulator, which we have developed in PyTorch[8], to train neural networks to approximate solution manifolds that satisfy a continuum of antenna array synthesis problems using joint parameterised optimisation (JPO) [9] in an end-to-end fashion.

2 Joint Parameterised Optimisation of Antenna Arrays

Adapting the notation from [9], we formulate our problem as a joint parametrised optimisation (JPO):

$$\boldsymbol{w}_i^* = \boldsymbol{w}_i(\varphi^*) \quad \varphi^* = \mathrm{argmin}_{\varphi} \sum_{i=1}^{n} \mathcal{L}(\mathcal{F}(\mathcal{N}(\gamma_i|\varphi)|\gamma_i), E_i^*(\gamma_i)). \tag{3}$$

Fig. 1 shows a graphical view of the process. Here $F(w_n|\gamma_i)$ is our differentiable simulator, γ_i are the variables of the simulation, such as positions and directions of targets, and $E_i^*(\gamma_i)$ are the desired beam-pattern levels based on these variables. A neural network $\mathcal{N}_\varphi : \gamma_i \to \boldsymbol{w}_i$ predict excitation coefficients for a given set of desired targets. This is conducted in a batch-wise fashion, allowing us to approximate the gradient of the solution manifold using stochastic gradient descent. We directly constrain $|w_n| = 1$ during training. The advantage of this setup on the antenna array synthesis problems comes from multiple factors:

1. Often, a large number of inverse problems need to be solved for an AESA system, since the goal is to make the system reconfigurable, i.e for different desired steering angles, field patterns, and nulling directions. A large number of highly similar inverse problems must be solved to span this solution space. This makes JPO's ability to approximate continuous solution manifolds advantageous.
2. For certain goals, the problem can be highly non-convex, making it very difficult to reliably use higher-order optimisers without repeatedly getting stuck at local minima. This further increases the number of optimisations that need to be run.

In our experience, the primary difficulty of an efficient implementation of JPO in PyTorch comes from implementing the simulator, F and goal-manager $\mathcal{G}_M$, in a way such that we can maximise the amount of computation being done with GPU-accelerated matrix multiplications and at the same time maintain framework adaptability for different goals, simulation setups and advanced training techniques. In the next section, we provide an overview of the framework's key components.

3 A Batchable and Differentiable Simulator for Antenna Arrays

Returning to Fig. 1, we will now expand on the components of the computational setup:

Batchable Differentiable Simulator, F: For idealised arrays where $E_e(\theta, \phi) = 1$, we can express the majority of the simulation as a single large tensor product in PyTorch. Thus, the majority of

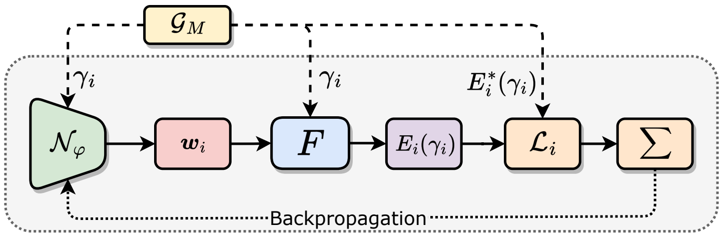

Figure 1: Joint parameterised optimisation of antenna arrays. The goal-manager $\mathcal{G}_M$ distributes batches of identifiers, γ_i and targets $E^*(\gamma_i)$. The neural network $\mathcal{N}_\varphi$ predicts a batch of excitation coefficient vectors, which our differentiable solver $F(\boldsymbol{w}_i|\gamma_i)$ then provides results for.

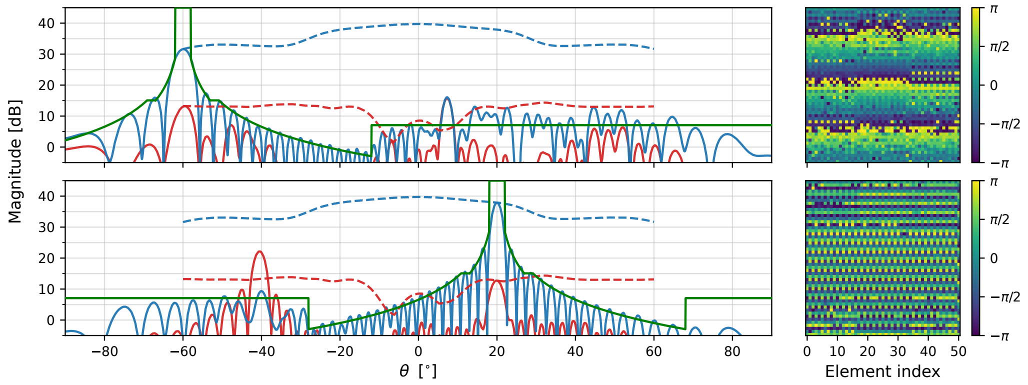

Figure 2: Sample beamforming results for two positions of θ (-60, and 20 degrees) at a fixed ϕ angle for a single beamforming calculation. The co-polar and cross-polar results are plotted in blue and red, respectively, and are presented against the gain mask in green. The dashed lines show the magnitude at the peak as the beam is steered in dB (dB $= 20\log_{10}(|E|)$). The excitation phase for the 51×51 antenna array, as predicted by the neural network, is displayed as a colour matrix plot on the right of each beam position.

implementation focus has been on constructing matrices before multiplication. In short, we try to express as much of the geometry creation and angular sampling as projections from unit grids. If in memory, they are quickly projected to a batch of geometries. For more complicated arrays, where we desire all effects to be accounted for, we use a commercial EM simulation software built on a fast-direct solver to extract the true embedded element pattern E_e of each antenna element [10], [11]. This process can take hours to days for large, state-of-the-art array structures on modern workstations. We then use an interpolating reduced-order model of the embedded element matrix during training, which allows us to fit it into GPU memory and ensure that we can evaluate each embedded element, $E_e(\theta, \phi)$, continuously across the spherical coordinates (θ, ϕ) without a large computational or memory shuffling overhead.

Goalmanager, G_M: From an adaptability perspective, the most critical aspect of software implementation is the handling of goals. We have created a bottom-up structure where all our goals are independently sampling their parameterisations. These parameterisations are then packed in the 'goal-manager' and distributed to the neural network if relevant, i.e they are a variable of the optimisation, or only provided to the simulator if they are a constant of the specific simulation setup. During loss calculations, each goal has its own target and calculates its residual individually, which is then scalarized in the goal manager. This also easily allows for advanced scalarization methods, such as ConFIG [12], which can help balance training when competing goals are defined in a multi-objective optimisation setting. The goal manager also keeps track of how to pack the tensor inputs to the neural network and unpack the resulting far-fields from the simulator.

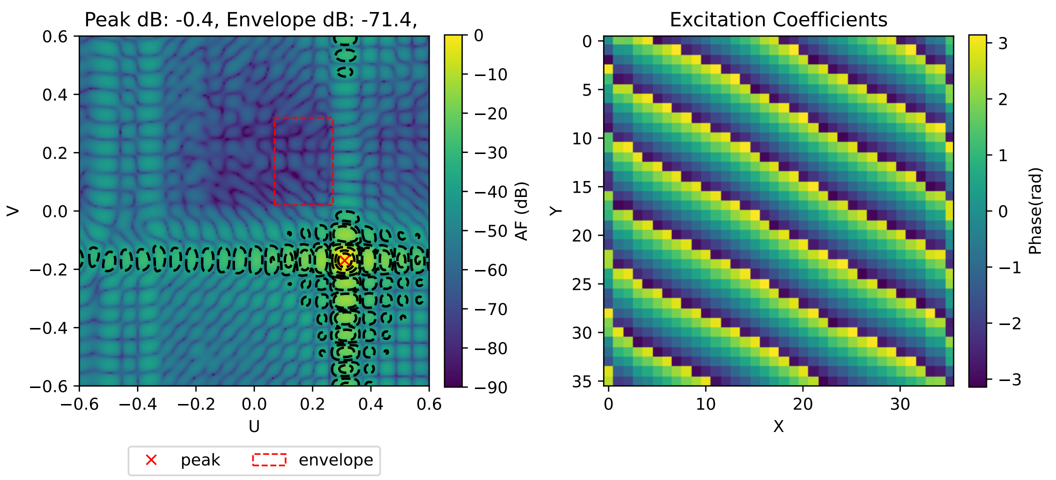

Figure 3: **Left**: Trained model maximising transmitted power towards a specified direction (red cross) and minimising power within a parameterised variable envelope. Plotted on spherical projection $U = \sin\theta\cos\phi$, $V = \sin\theta\sin\phi$. Plot is normalised such that 0 dB is the maximum achievable. **Right**: Phase of output of neural network plotted by the index of the square array geometry

4 Results

We now show examples of using the batchable simulator and the JPO method to train networks to approximate a solution manifold of excitation coefficients for two different problems:

Complying to gain mask: In Fig. 2 we see the results using JPO to comply with the ETSI EN 303978 gain-mask standard over a steering range of $\theta, \phi \in [-60°, 60°]$. The resulting trained neural network is a CNN-style network that predicts 51 × 51 normalised complex-valued excitation coefficients for a given desired steering angle. Compliance with the gain mask in green is almost upheld, and the minor deviations can be easily made compliant with a few iterations of a direct optimiser. Training time for this network was approximately 7 minutes, and inference time is around 1ms per synthesised excitation on an RTX 4090, enabling real-time prediction when embedded in an antenna system.

Steering with parameterized no-signal zone: In Fig 3 show the results of optimizing an idealized array, meaning $|E_{e,n}(\theta, \phi)| = 1 \;\forall\; n$, arranged in a square with 0.5λ spacing. A neural network is trained to maximise the transmitted power in a given direction and minimise the transmitted power in a parameterised 'no-go zone'. The area is parameterised for variable size and position, as larger areas will inevitably lead to bigger tradeoff costs for the main steering beam. The network is trained to steer both the no-go zone and the main beam in all combinations of $U, V \in [-0.6, 0.6]$, and we have used an adaptive multi-objective scalarization technique to balance the wildly different loss terms [12]. This example also highlights one of the difficulties of the process, as we have competing goals that are defined over the same space; many of the requested simulation objectives are not achievable, which adds noise to the gradient during training.

5 Conclusion

While this method does not currently provide solutions of the same quality as highly tuned direct optimisation algorithms, in our experience, it has proven to be an effective method for solving a large number of array synthesis problems simultaneously. This also means that if the solutions are not satisfactory, they are often excellent starting points for higher-order optimisation methods, which are otherwise prone to getting stuck at local minima.

Networks like this can also be used for error mitigation; training on possible antenna element failure modes can ensure that an array remains optimally operational, even when parts of the array fail. Potential extensions of the framework to other areas are also possible. Combining multiple receivers and transmitters into directional arrays is also utilised in the audio domain. Where ultrasound, microphone arrays and sonar all use identical concepts. Thus, similar setups for audio-array synthesis should be equally viable.

Acknowledgments and Disclosure of Funding

Parts of this work are based on a collaboration between TICRA and the European Space Agency (ESA) carried out under the ESA contract No. 4000139341/22/NL/GLC/ov.

References

[1] B. Constantine, *Advanced Engineering Electromagnetics*, 2nd ed. John Wiley & Sons, 2012.

[2] B. Constantine, *Modern Antenna Handbook*. John Wiley & Sons, 2011.

[3] A. K. Mishra, "AESA radar and its application," *Proceedings of the 2018 International Conference On Communication, Computing and Internet of Things, IC3IoT 2018*, pp. 205–209, Jul. 2018, ISBN: 9781538624586 Publisher: Institute of Electrical and Electronics Engineers Inc. DOI: `10.1109/IC3IOT.2018.8668101`.

[4] R. J. Mailloux, *Phased array antenna handbook (3rd edition)*, 2018.

[5] D. Roy et al., "Going beyond RF: A survey on how AI-enabled multimodal beamforming will shape the NextG standard," *Computer Networks*, vol. 228, p. 109 729, Jun. 2023, ISSN: 13891286. DOI: `10.1016/j.comnet.2023.109729`.

[6] J. Han, X. Wang, Y. Wei, and Y. Zhang, "Antenna Array Pattern Synthesize Based On Convolutional Neural Network," in *2022 IEEE 10th Asia-Pacific Conference on Antennas and Propagation (APCAP)*, Nov. 2022, pp. 1–2. DOI: `10.1109/APCAP56600.2022.10069786`.

[7] H. Al Kassir et al., "Antenna Array Beamforming Based on Deep Learning Neural Network Architectures," in *2022 3rd URSI Atlantic and Asia Pacific Radio Science Meeting (AT-AP-RASC)*, May 2022, pp. 1–4. DOI: `10.23919/AT-AP-RASC54737.2022.9814201`.

[8] J. Ansel et al., "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation," en, in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, La Jolla CA USA: ACM, Apr. 2024, pp. 929–947, ISBN: 979-8-4007-0385-0. DOI: `10.1145/3620665.3640366`.

[9] P. Holl and N. Thuerey. "The Unreasonable Effectiveness of Solving Inverse Problems with Neural Networks." arXiv: `2408.08119 [cs]`, pre-published.

[10] M. H. Gæde, M. S. Andersen, A. Limkilde, O. Borries, and J. S. Hesthaven, "A Fast Direct Solver for Higher Order Discretizations of Integral Equations," en, in *2024 IEEE International Symposium on Antennas and Propagation and INC/USNC-URSI Radio Science Meeting (AP-S/INC-USNC-URSI)*, Firenze, Italy: IEEE, Jul. 2024, pp. 497–498, ISBN: 979-8-3503-6990-8. DOI: `10.1109/AP-S/INC-USNC-URSI52054.2024.10686430`.

[11] M. Zhou et al., "Analysis and design of large array antennas using a fast direct solver," in *2025 19th European Conference on Antennas and Propagation (EuCAP)*, 2025, pp. 1–5. DOI: `10.23919/EuCAP63536.2025.10999264`.

[12] Q. Liu, M. Chu, and N. Thuerey, *ConFIG: Towards Conflict-free Training of Physics Informed Neural Networks*, arXiv:2408.11104 [cs], Mar. 2025. DOI: `10.48550/arXiv.2408.11104`. Accessed: Oct. 1, 2025.