Extended Abstract

Motivation Dynamic beam steering—redirecting an electromagnetic signal toward an arbitrary target angle on demand—is a foundational capability for radar, communications, and sensing. Reconfigurable plasma metamaterials demonstrated at the Stanford Plasma Physics Laboratory offer a compelling substrate: a lattice of gas-discharge rods whose permittivities, and hence the field transformation the device implements, can be reconfigured electronically. Existing inverse-design approaches Rodríguez and Cappelli (2023); Rodríguez et al. (2021), however, produce a static device optimized for a single target angle, and re-running inverse design for each new target costs hundreds of high-dimensional FDFD simulations per query—incompatible with real-time retargeting. We replace per-target inverse design with a closed-loop control policy that, given the current permittivity configuration ε and a target angle θ, proposes a perturbation δ that steers the beam toward the queried target.

Method We cast dynamic beam steering as a goal-conditioned reinforcement learning problem with a reward $r(s, a \mid \theta)$ parameterized by the target angle. The pipeline has two phases. *Phase 1* runs evolution strategies (ES) in permittivity space to solve per-angle inverse design for ten training angles. Because each FDFD solve reports power at all receiver ports, we relabel every solve across all goals—turning a modest number of solves into a goal-conditioned training corpus at no extra simulation cost—and use it to pretrain a fast surrogate of the solver and a goal-conditioned critic. *Phase 2* trains a retargeting policy by ES on policy parameters, warm-started by behavior cloning on the Phase 1 buffer. Within each step the surrogate and critic compare candidate actions cheaply and the solver confirms only the committed action, so each step costs a single solve while policy improvement is driven by realized FDFD performance.

Implementation The pipeline runs in a `ceviche` FDFD environment modeling the SPPL device: a 10×10 rod array on a 22 mm pitch, a 6 GHz line source ($\lambda \approx 50$ mm) injected through a reflective horn, and 30 boundary receivers spanning a $\sim 180^\circ$ output fan. A single solve is one environment transition. Phase 1 is parallelized per angle with checkpoint logging, as each solve takes $\sim$ 2–3 minutes.

Results Phase 1 yields a ten-angle memory bank of steered configurations and the multi-goal-relabeled replay buffer. Evaluated on receiver angles held out from training, the closed-loop policy generalizes to unseen targets, and policy architecture proves decisive: a convolutional policy that respects the rod lattice reaches a mean target-power fraction of 0.65 on unseen angles, against 0.16 for an otherwise-identical multilayer perceptron—the latter barely above the uniform-spread floor.

Discussion Since the two policies differ only in architecture, the gap isolates spatial inductive bias as the operative factor: convolutional weight sharing captures the locality of element-to-field coupling that a flat policy must relearn from limited data and largely fails to. For control over physical arrays, matching the policy to the spatial structure of the medium is a prerequisite for transfer, not a tuning detail.

Conclusion We present a closed-loop RL system that replaces per-target inverse design with a single goal-conditioned controller, trained by a two-phase ES curriculum that amortizes inverse design at modest simulation cost. These results establish the control machinery and demonstrate angular generalization within one frequency regime; extending it across the electromagnetic spectrum by reparameterizing the substrate is the natural next step.

Reinforcement Learning for Dynamic Beam Steering in Plasma Metamaterials

Susan Zhang
Department of Mathematics
Stanford University
susanz10@stanford.edu

Abstract

We study reinforcement learning for dynamic beam steering in a reconfigurable plasma metamaterial: a 10×10 lattice of gas-discharge rods whose permittivities are tuned to steer an injected 6 GHz microwave beam toward a chosen receiver port. Classical inverse design solves this one target at a time, spending hundreds of finite-difference frequency-domain (FDFD) simulations per angle and producing a fixed device that cannot retarget on demand. We instead pose steering as a closed-loop control problem and learn a single goal-conditioned policy that maps the current permittivity field and a desired output direction to an incremental reconfiguration. A two-phase curriculum makes this tractable: a static phase solves a small set of training angles with evolution strategies (ES) while relabeling every FDFD solve across all output ports—turning each solve into many goal-conditioned training examples at no additional simulation cost—and a closed-loop phase refines the policy by ES, using a learned surrogate and critic to compare candidate actions cheaply while the solver confirms only the committed step. The trained controller steers to receiver angles held out from training, generalizing and attaining a mean target-power fraction of 0.65 on unseen angles.

1 Introduction

Metamaterials realize effective electromagnetic properties through subwavelength structuring rather than bulk material choice. A *plasma* metamaterial replaces fixed inclusions with gas-discharge rods whose permittivity depends on electron density, and therefore on an applied bias voltage. This makes the device *reconfigurable*: the same physical slab can implement many different field transformations by re-programming its rods. A natural application is beam steering—configuring the array so that a wave injected from a fixed source emerges focused toward a chosen output direction.

Prior work by Rodriguez and Cappelli Rodríguez and Cappelli (2022) demonstrates inverse design of plasma metamaterials for beam steering, optimizing a rod configuration for a single, *static* target direction. In control terms this is an *open-loop* design: the configuration is computed offline and applied as a fixed setting, independent of the system's evolving state. We instead study *dynamic* beam steering—changing the output direction on demand—where the target is reissued over time and the array must retarget from whatever configuration it currently holds. This is inherently a *closed-loop* control problem: a feedback policy that, at each step, observes the current state and chooses an action, rather than a one-shot design.

The bottleneck is design speed. Computing the rod configuration that steers the beam to a particular angle is a high-dimensional inverse problem. Gradient-based inverse design Rodríguez et al. (2021) can solve it, but each evaluation requires a full electromagnetic simulation, and a single target may take hundreds to thousands of solves. A deployed device, by contrast, must retarget quickly

and repeatedly: the field starts in one steered configuration and the user requests a different angle. Re-running inverse design online is infeasible.

We therefore pose dynamic beam steering as a sequential decision problem and learn a closed-loop controller $\pi_\phi(\varepsilon, \theta) \to \delta$ that maps the current normalized permittivity field $\varepsilon \in [-1, 1]^{100}$ and a target receiver index θ to an incremental perturbation δ, applied as $\varepsilon \leftarrow \text{clip}(\varepsilon + \delta, -1, 1)$. At deployment the controller retargets the beam with a sequence of cheap forward passes; inverse design is done once offline and is absorbed into the policy weights to warm start the online policy.

2 Related Work

Plasma metamaterial inverse design. Rodriguez et al. Rodríguez et al. (2021); Rodríguez and Cappelli (2022, 2023) establish gradient-based inverse design for plasma metamaterial waveguides, demultiplexers, and beam-steering devices using automatic differentiation through FDFD simulation. Their framework achieves near-optimal single-target performance, but produces a fixed device per target: the optimization is not shared across steering angles, and the resulting designs are not reconfigurable without re-running the full inverse-design loop. In practice each inverse-design iteration is dominated by the FDFD solve (~ 40 s in the previous SPPL setup with 8×8 rod array at lower grid resolution, which translates to $\sim 2 - 3$ mins with our current design parameters) and does not parallelize beyond the solver's ~ 16-core limit, so this per-target cost cannot be reduced simply by adding compute—underscoring why re-optimizing on demand is impractical at deployment.

Gradient-free inverse design for plasma arrays. Ertan (2025) Ertan (2025) applied Evolution Strategies to the same plasma-rod inverse-design setting and established two relevant negative results: a single static design cannot serve multiple steering angles (ES-Multi), and a cue-conditional generator trained end-to-end collapses to a subset of target angles (ES+NN). We read these as empirical confirmation that a linear, passive substrate with a single fixed permittivity assignment cannot implement multi-target routing, motivating our shift to a dynamically reconfigured substrate with a learned update policy. That study also yields near-lossless single-angle redirection solutions—a strong single-target baseline we aim to match while extending to on-demand multi-angle steering.

3 Method

We treat dynamic beam steering as a goal-conditioned control problem. A reconfigurable plasma-metamaterial array of 100 elements is described by a configuration vector $\varepsilon \in [-1, 1]^{100}$, which sets the permittivity of each element. Given a configuration, a finite-difference frequency-domain (FDFD) solver computes how much power reaches each of ten output angles. The goal is to find configurations that concentrate power at a chosen target angle. The agent learns a policy $\pi_\phi(a \mid s, \theta)$ that maps the current rod permittivity state and target cue to a permittivity perturbation, alongside a goal-conditioned Q-critic $Q_\psi(s, a \mid \theta_{target})$ used to sample candidate actions and score policy rollouts during training. The state, action, and reward are defined as the following:

State (s): state vector $\varepsilon \in [-1, 1]^{100}$, where ε_i is the normalized permittivity value of rod i.

Action (a): perturbation $a = \text{clip}(s + \delta, [-1, 1])$ where $\delta = \sigma\xi$ and $\xi \sim \mathcal{N}(0, I_{100})$

Reward (r): $r(s, a \mid \theta) = P_\theta(s) - \lambda_{\text{crosstalk}}(s) \cdot \sum_{\theta' \neq \theta} P_{\theta'}(s) - \lambda_{\text{loss}}(s, a) - \lambda_{\text{energy}}(s, a)$, where:

- $\lambda_{\text{crosstalk}}(s) = 1 - \left(P_\theta(s) / \left(P_\theta(s) + \sum_{\theta' \neq \theta} P_{\theta'}(s) \right) \right)$ penalizes for low signal-to-noise ratio at state s
- $\lambda_{\text{loss}}(s, a) = \Delta P_{\text{loss}}(s, a) = P_{\text{loss}}(a) - P_{\text{loss}}(s)$ is the change in dispersive loss caused by action a, and
- $\lambda_{\text{energy}}(s, a)$: $\Delta E_{\text{rods}}(s, a) = E_{\text{rods}}(a) - E_{\text{rods}}(s)$ is the change in rod-applied energy caused by action a

Phase 1 solves the steering problem for a fixed set of angles and, along the way, trains two helper networks: a fast surrogate that imitates the solver, and a critic that scores how good an action is for a given goal. Phase 2 reuses those networks to train a policy that steers the array in a closed loop—retargeting it to new angles on the fly, without re-solving from scratch.

3.1 Phase 1: Static Design and Network Pretraining

Phase 1 handles ten target angles one at a time. For each angle, we search for a configuration that delivers nearly all the power to that angle, using Evolution Strategies (ES). ES works without gradients: at each step it tries a batch of randomly perturbed configurations, evaluates each one, and nudges the configuration toward the perturbations that scored best. We use mirrored sampling—each random perturbation is tried in both directions—and rank the results rather than using raw scores, which makes the update robust to outliers. The search continues until the target angle receives nearly all the power.

The key efficiency comes from how we use each solve. Every FDFD solve reports the power at *all ten* angles, not just the one we are currently optimizing for. A configuration that was tried while aiming at one angle is therefore also a free measurement of how that configuration performs for the other nine. We record each solve as ten separate training examples, one per angle. This costs nothing extra and turns a modest number of solves into a training set ten times larger.

We use these recorded examples to train the two helper networks while the ES search runs. The surrogate learns to predict the solver's power output from a configuration. The critic learns to predict, for any goal angle, how good a given action is. Both are trained only on real solver data, so by the end of Phase 1 the surrogate is a faithful stand-in for the solver and the critic can score actions against any of the training goals. We keep the best configuration found for each angle in a memory bank, which gives Phase 2 a set of known good starting points.

3.2 Phase 2: Closed-Loop Steering

Phase 2 trains the policy that performs dynamic steering. The policy takes the array's current configuration and a desired target angle, and returns an incremental adjustment to the configuration, then evaluates on the next state. We warm start the policy by behavior cloning on Phase 1 trajectories. The policy is then refined by ES perturbations on policy parameters to optimize for signal at the target port.

Each training iteration runs a batch of steering attempts in parallel. Every attempt starts from one beam direction and is asked to move the array to a *different* target angle—drawing from the memory bank of converged configurations from Phase 1 to train for retargeting dynamics.

Within an attempt, we keep the solver out of the inner loop. At each step the policy proposes several candidate adjustments. Rather than simulating each one, we predict its outcome with the cheap surrogate and score that prediction with the critic, then commit to the best candidate. Only the committed adjustment is checked against the real solver, with a single FDFD solve. In other words, the surrogate does the comparing and the solver does the confirming—so each step costs exactly one solve no matter how many candidates were considered. The attempt ends once the target angle receives nearly all the power.

Every confirmed step feeds back into the system. We use it to keep the surrogate accurate and to correct the critic wherever its prediction disagreed with what the solver actually returned. Finally, we update the policy. Each steering attempt is scored by the total reward it actually achieved—measured by the solver, not the surrogate—and ES updates pushes the policy toward the variants that performed best.

Table 1: Phase-2 policy architectures (default configuration, 10×10 array, 30 goals). "ES dim" is the parameter count searched by ES

Architecture	Parameters (ES dim)	Inductive bias / conditioning
MLP (baseline)	183,652	flatten + $(\sin, \cos)$ goal
CNN	125,697	CoordConv + FiLM + global context (GELU)

4 Experimental Setup

4.1 SPPL hardware setup as the simulation environment

Our environment models the plasma metamaterial device at the Stanford Plasma Physics Laboratory (SPPL), specified in `pm_setup.py`. The device is a 10×10 array of gas-discharge plasma rods on a 22 mm pitch; each rod's relative permittivity is set by its electron density through a Drude response, and therefore by an applied bias, making the array electronically reconfigurable. A 6 GHz microwave source ($\lambda \approx 50$ mm) is injected from a waveguide on the left edge, with reflective walls forming a horn that directs the beam into the array; thirty receiver ports tile the bottom, right, and top boundaries—a $\sim 180^\circ$ fan of steerable output directions (Fig. 1). We simulate the device with `ceviche` FDFD over a compiled permittivity canvas: a single FDFD solve is one environment transition, and the policy controls the rod permittivities $\varepsilon \in [-1, 1]^{10\times 10}$. The full geometry→canvas compilation and all physical parameters are documented in Appendix A.1 and Table 3.

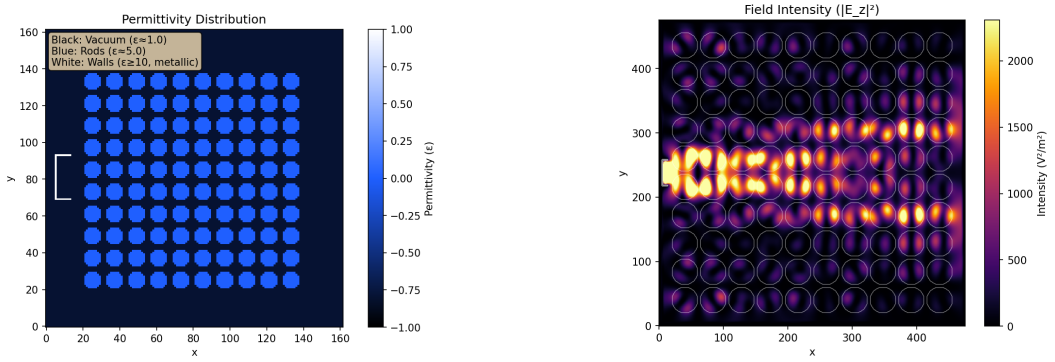

Figure 1: Left: the 10×10 plasma-rod permittivity canvas with the left-wall line source and the boundary receivers. Right: the steady-state $|E_z|^2$ field intensity from an FDFD solve. The policy controls the rod permittivities; the reward takes as input power at the boundary receivers.

4.2 Evaluation:

We compare two goal-conditioned networks $\pi_\phi(\varepsilon, \theta) \to \delta$ with an identical interface (Table 1). The MLP flattens ε and concatenates the goal encoding—a baseline that discards the rod lattice. The CNN treats ε as an image with 3×3 convolutions (near-neighbor rod coupling), CoordConv channels to encode the fixed source/receiver geometry, FiLM goal conditioning, and a global-context branch so each rod sees the globally-emergent beam direction; all activations are GELU.

We evaluate the trained controller along two axes:

Signal isolation at the target receiver. Steering succeeds only if power is both *delivered to* the target receiver and *isolated from* the others. We report (i) the transmission to the target, $P_\theta / P_{\text{source}}$—the fraction of injected power reaching the target port—and (ii) the port isolation, the target fraction $P_\theta / \sum_j P_j$ together with the ratio of target power to the strongest off-target port. A policy that raises P_θ while leaking comparable power to neighboring ports does not steer; high isolation, not raw target power, is the operative success criterion, and it is what the crosstalk term in the reward targets.

Generalization to unseen angles. The 10 Phase-1 angles cover only a third of the 30 receiver ports. Because the goal is encoded as a continuous semicircle feature, the policy can be queried at directions it never trained on. We therefore evaluate retargeting to 10 new held-out receiver indices and report isolation as a function of angular distance from the nearest training angle, testing whether the controller *interpolates* across the receiver fan rather than memorizing the ten trained directions.

5 Results

We evaluate Phase 2 on ten receiver angles held out from the Phase 1 training set, measuring the fraction of total received power delivered to the target port, $P_\theta / \sum_j P_j$, after 150 iterations. This

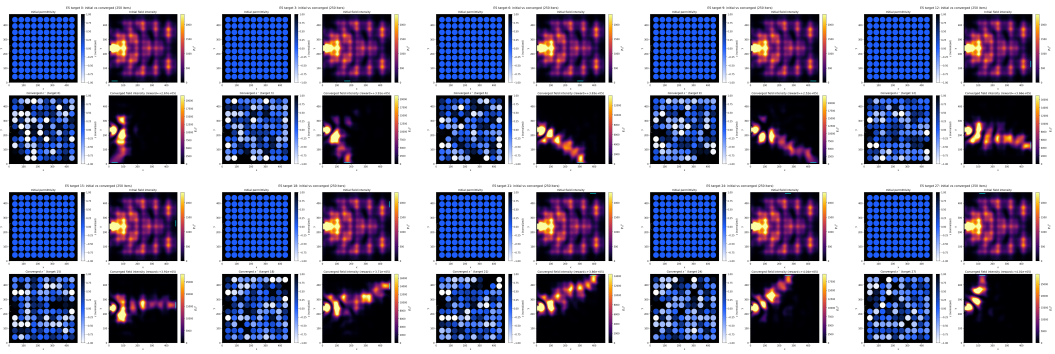

Figure 2: Phase-1 inverse design (uniform-init ES) for ten training angles (receiver indices $0, 3, 6, \ldots, 27$; left-to-right, top row then bottom). Each panel shows the before/after permittivity and the resulting $|E_z|^2$ field: from a common uniform initialization, ES perturbs the permittivity values of the 10×10 rod array differently for each target receiver to redirect power toward the correct port.

Table 2: Phase 2 on 10 unseen receiver angles (θ not included in the Phase-1 training set). We compare $P_\theta / \sum_j P_j$ (i.e. signal-to-noise ratio) at each target-port θ after 150 iterations for CNN and MLP.

Receiver port	CNN	MLP
1	0.697	0.257
4	0.706	0.149
7	0.680	0.162
10	0.439	0.115
13	0.666	0.179
16	0.656	0.062
19	0.481	0.250
22	0.715	0.136
25	0.725	0.160
28	0.707	0.144
Mean $\pm$ SD	**0.647 $\pm$ 0.102**	**0.161 $\pm$ 0.058**

quantity is the natural figure of merit for the routing objective: a value approaching one indicates that the policy has concentrated essentially all radiated power at the intended port, while a uniform spread across the ten ports would yield 0.1. Table 2 reports results for two policy parameterizations, a convolutional network (CNN) and a multilayer perceptron (MLP).

The CNN policy generalizes to unseen angles substantially better than the MLP. It achieves a mean target-power fraction of 0.647 ± 0.102 against the MLP's 0.161 ± 0.058—a fourfold difference, and one that separates a policy that reliably routes power to the correct port from one that performs little better than chance. The MLP's mean of 0.161 is close to the 0.1 uniform-spread floor, indicating that it largely fails to concentrate power at target angles it was not trained on; its best case (0.257 at port 1) does not reach the CNN's worst.

The CNN's performance is also more consistent across the angular range. Its standard deviation of 0.102 reflects a policy that maintains a high target-power fraction at most ports, with two exceptions—ports 10 and 19, at 0.439 and 0.481—pulling down the mean. The remaining eight ports lie between 0.656 and 0.725, a tight band suggesting that the policy has learned a transferable steering strategy rather than memorizing the training angles. The two weaker ports are candidates for closer inspection: their position in the angular range, relative to the Phase 1 training angles, may indicate where interpolation between learned targets is hardest.

The gap between the two parameterizations is the central result. Both policies were trained under an identical curriculum and differ only in architecture, so the disparity isolates the effect of the inductive bias each imposes. The CNN's advantage is consistent with the spatial structure of the problem: the

configuration vector indexes a physical array, and convolutional weight sharing captures the locality of element-to-field interactions in a way that a fully connected MLP, which treats each element independently, does not. This points to architectural inductive bias as a decisive factor in generalizing the steering policy to unseen targets.

6 Discussion

Insights: Our central finding is that a goal-conditioned policy, trained on a fixed set of steering angles, can route power to angles it never saw during training—and that the architecture of the policy is decisive in whether this generalization occurs. The convolutional policy reaches a mean target-power fraction of 0.647 on held-out angles, against 0.161 for an otherwise-identical multilayer perceptron. Because the two policies differ only in architecture, the gap isolates inductive bias as the operative factor: weight sharing over the spatial layout of the array captures the locality of element-to-field coupling, whereas a fully connected policy must relearn that structure from limited data and largely fails to. This suggests that, for control over physical arrays, the match between policy architecture and the spatial structure of the medium is not a tuning detail but a prerequisite for transfer.

Limitations: Two of the ten held-out angles (ports 10 and 19) show markedly lower target-power fractions than the rest. We attribute this tentatively to their position relative to the Phase 1 training angles—generalization is hardest where the policy must interpolate farthest from any seen target—but we have not isolated the cause, and it may instead reflect a property of the field geometry at those angles. Either way, the result indicates that transfer is not uniform across the angular range, which any deployment would need to account for.

The results support that a closed- loop, goal-conditioned policy can generalize beam steering to unseen angles when its architecture respects the spatial structure of the array, and that a two- phase curriculum makes this achievable at modest simulation cost. The path from here to the broader goal—a single controller that transfers across the electromagnetic spectrum by reparameterization—runs through the spectral- transfer experiment named above.

7 Conclusion

We posed dynamic beam steering in a reconfigurable plasma metamaterial as a closed-loop control problem and showed that a goal-conditioned policy can replace per-target inverse design. Where prior work computes a fixed device per steering angle through hundreds of FDFD solves, our two-phase curriculum trains a single controller that retargets on demand, with the static phase doubling as a data generator—relabeling every solve across all output angles to seed the policy at no extra simulation cost.

The controller is able to successfully converges to unseen angles, though the architecture is decisive: a convolutional policy that respects the rod lattice reaches a mean target-power fraction of 0.647 on held-out angles, against 0.161 for an otherwise-identical MLP. The gap isolates inductive bias—matching the policy to the physical structure of the medium is what enables transfer.

These results establish the control machinery and demonstrate angular generalization within a single frequency regime. The motivating premise—one controller steering a substrate retuned across the electromagnetic spectrum by reparameterization—remains to be tested, and the spectral-transfer experiment is the natural next step.

Changes from Proposal Relative to the proposal, the project expanded from a single ES-based Phase-2 policy to a comparative study of architectures.

References

Selin Ertan. 2025. Conditional Beam Steering in Plasma Photonic Crystals via Evolution Strategies. (2025). CS229 Final Report, Stanford University.

Jesse A. Rodríguez, Ahmed I. Abdalla, Benjamin Wang, Beicheng Lou, Shanhui Fan, and Mark A. Cappelli. 2021. Inverse Design of Plasma Metamaterial Devices for Optical Computing. *Physical Review Applied* 16, 1 (2021), 014023. doi:10.1103/PhysRevApplied.16.014023

Jesse A. Rodríguez and Mark A. Cappelli. 2022. Inverse Design of Plasma Metamaterial Devices with Realistic Elements. *Journal of Physics D: Applied Physics* 55, 46 (2022), 465203. doi:10.1088/1361-6463/ac8da1

Jesse A. Rodríguez and Mark A. Cappelli. 2023. Inverse Design and Experimental Realization of Plasma Metamaterials. *Physical Review Applied* 20, 4 (2023), 044017. doi:10.1103/PhysRevApplied.20.044017

A Implementation Details

A.1 The `pm_setup` environment pipeline

The environment is specified declaratively as a set of geometry objects (a design region, a rod grid, a source, and receivers) and then compiled into a single permittivity "canvas"—the relative-permittivity field ε_r that `ceviche` solves on. Table 3 lists the physical parameters of the 10×10 device used throughout. The compile step (`initialize_environment`) runs five conversions, in order:

1. **Allocate the canvas.** A 2D array is filled with the background permittivity (vacuum, $\varepsilon_r = 1$). Its size is the rod-grid extent plus a margin and a perfectly-matched-layer (PML) border on every side ($\approx 476 \times 476$ cells at 0.5 mm resolution).
2. **Stamp the rods.** Each of the 100 lattice sites is rasterized as a filled disk of radius $r/\Delta L$ cells (here 20 cells), set to that rod's permittivity. The rods sit on a 22 mm pitch (20 mm diameter plus a 2 mm wall gap), so the disks are disjoint from one another and from the walls.
3. **Stamp the source.** A vertical line current on the left wall, centered, of length $\ell/\Delta L$ cells, recorded as a binary source mask.
4. **Stamp the source walls.** Reflective metallic plates ($\varepsilon_r = 10^6$) form a horn around the source: a back wall to its left suppresses backward radiation, and two horizontal plates extend rightward from the source endpoints, channeling the injected beam forward into the rod array.
5. **Stamp the receivers.** Thirty boundary masks—one per rod column on the bottom and top, one per rod row on the right—are placed so that the receiver index increases monotonically with steering angle (counter-clockwise from the bottom-left, through the right edge, to the top-left).

Table 3: Physical parameters of the `pm_setup` environment.

Component	Parameter	Value
Grid	rods; radius; pitch	10×10; 10 mm; 22 mm
Source	frequency; wavelength	6 GHz; $\lambda \approx 50$ mm
Receivers	count (bottom/right/top)	30 (10/10/10), $\sim 180^\circ$ fan
Domain	resolution; PML; margin	0.5 mm/px; 10 cells; 20 cells
Walls	plate permittivity	$\varepsilon_r = 10^6$ (metallic)
Canvas	size; array span	$\approx 476^2$ cells; $\approx 3.96\lambda$

Solve and measurement. A simulation step calls `ceviche.fdfd_ez`$(\omega, \Delta L, \varepsilon_r, \mathrm{npml})$ with $\omega = 2\pi f$ and a unit-amplitude current on the source mask, returning the out-of-plane field E_z (the PML absorbs outgoing radiation; multiple sources, if present, superpose coherently). Receiver powers are the squared field integrated over each mask, $P_j = \sum |E_z|^2 \, m_j$. We also read out two auxiliary scalars: a leakage proxy P_{loss} (intensity summed over the PML ring) and a per-rod effort proxy $E_{\text{rods}} = \sum(1 - \varepsilon)/2$. The fixed denominator P_{source} of the source-normalized reward is the total receiver power in free space (rods set to background), computed once.

RL interface. The policy state is the per-rod permittivity field $\varepsilon \in [-1, 1]^{10\times 10}$. Applying an action overwrites only the rod disks of the canvas with $\mathrm{clip}(\varepsilon + \delta, -1, 1)$—the walls, PML, source, and background are untouched, since the disks are disjoint from them—then re-solves and reads P. Because the rod-vector→canvas stamping is a fixed linear scatter (disjoint disks) composed with the differentiable solver, the exact gradient $\partial P/\partial \varepsilon$ is available, which is what the adjoint bridge exploits.