

Deep Unfolding for Fast Linear Massive MIMO Precoders under a PA Consumption Model

Thomas Feys*, Xavier Mestre[†], Emanuele Peschiera*, François Rottenberg*

*KU Leuven, ESAT-WaveCore, Dramco, 9000 Ghent, Belgium

[†]ISPIC, Centre Tecnologic Telecomunicacions Catalunya, Barcelona, Spain.

Abstract—Massive multiple-input multiple-output (MIMO) precoders are typically designed by minimizing the transmit power subject to a quality-of-service (QoS) constraint. However, current sustainability goals incentivize more energy-efficient solutions and thus it is of paramount importance to minimize the consumed power directly. Minimizing the consumed power of the power amplifier (PA), one of the most consuming components, gives rise to a convex, non-differentiable optimization problem, which has been solved in the past using conventional convex solvers. Additionally, this problem can be solved using a proximal gradient descent (PGD) algorithm, which suffers from slow convergence. In this work, in order to overcome the slow convergence, a deep unfolded version of the algorithm is proposed, which can achieve close-to-optimal solutions in only 20 iterations as compared to the 3500 plus iterations needed by the PGD algorithm. Results indicate that the deep unfolding algorithm is three orders of magnitude faster than a conventional convex solver and four orders of magnitude faster than the PGD.

Index Terms—deep unfolding, massive MIMO, PA efficiency, precoders, proximal gradient descent

I. INTRODUCTION

A. Problem Formulation

Reducing carbon emissions and energy consumption is more than ever a priority, as it is currently put forward by both Europe’s Green Deal [1] and the United Nations Sustainable Development Goals [2]. Nevertheless, the estimated electricity consumption and carbon footprint of the wireless communications sector continues to rise [3]. As such, when looking at the development of 6G, energy-reducing techniques are of the utmost importance. In current 5G networks, massive MIMO is one of the key enablers [4]. Massive MIMO uses clever precoding schemes to spatially multiplex many users, leading to higher spectral efficiency [5]. These precoders are typically obtained by minimizing the transmit power subject to a QoS constraint (e.g., a per-user signal-to-interference-plus-noise ratio (SINR) constraint). However, it is known that the power consumed by the PAs, one of the most consuming components, does not scale linearly with the transmit power [6], [7]. As such, minimizing the PAs consumed power, rather than the

transmit power, can lead to significant energy savings [8]. Moreover, [9] and [8] have shown, for single-user and multi-user systems respectively, that this typically leads to sparse solutions in the number of activated antennas. This can further be exploited by deactivating the RF chains of the unused antennas. Additionally, from [8], [9], it is clear that minimizing the consumed power leads to a convex optimization problem, with the objective function being a composite of a convex, differentiable function and a convex, non-differentiable function. In [8], this problem is solved for a multi-user system using convex solvers. However, when aiming at real-time operation, faster algorithms are required. Given the form of the objective function, a proximal gradient descent (PGD) algorithm can be used to find the global optimum. Unfortunately, PGD suffers from slow convergence, requiring many iterations, which makes the use for real-time applications such as precoding difficult. Therefore, in this work, a deep unfolded version of the PGD algorithm is proposed, which drastically speeds up the optimization, delivering close-to-optimal approximations, in only 20 iterations, as compared to the 5000 needed by PGD.

B. Deep Unfolding

In recent years, algorithm unrolling, also known as deep unfolding, has provided many promising results in signal and image processing [10]. For instance, the technique has been successfully applied in compressed sensing, sparse coding, image denoising, detection and channel decoding in massive MIMO and many more application domains [10]–[12]. Deep unfolding makes the connection between iterative algorithms and deep neural networks. In this scheme, each iteration of an iterative algorithm is represented as a layer in a neural network. This is done for a fixed number of layers/iterations to form a neural network of fixed size. Running the neural network emulates the execution of the traditional algorithm for a finite number of iterations. Furthermore, the main benefit over the traditional algorithm is the fact that algorithm-specific parameters are learned, using backpropagation and stochastic gradient descent-based optimization, while for the traditional algorithm these parameters need to be manually tuned. Learning these parameters in each layer/iteration, produces algorithms that are highly optimized for the problem at hand, and can as such provide faster convergence rates [11]. Many precoding problems lead to iterative optimization algorithms. Consequently, some works have proposed the use of deep unfolding to solve these problems. For instance, in [13] a

This work was made possible by a mobility grant provided by FWO (filenumber V424222N) and a short-term scientific mission grant by COST ACTION CA20120 INTERACT. All code is available at: github.com/thomasf10/DeepUnfoldingMIMOPrecoder.

This paper is presented at VTC2023-Spring, T. Feys, X. Mestre, E. Peschiera, and F. Rottenberg, “Deep Unfolding for Fast Linear Massive MIMO Precoders under a PA Consumption Model,” in *2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*, Florence, Italy, June 2023.

deep unfolding algorithm is proposed to solve the sum-rate maximization problem in multi-user MIMO. Similarly, in [14], a deep unfolding algorithm is proposed that maximizes the sum-rate while being robust against channel estimation errors. Both works consider a transmit power constraint, neglecting the consumed power in their analysis. In this work, we consider the consumed power and minimized it, which results in a power consumption reduction.

C. Contributions

In this work, the design of a zero-forcing (ZF) precoder under a realistic PA consumption model is studied. Hence, the PAs consumed power is minimized, rather than the transmit power. First, we show that this leads to a convex optimization problem, with a composite objective function consisting of a convex, differentiable and a convex, non-differentiable part. Given the form of this cost function, the problem can be solved using a PGD algorithm. To support real-time operations, a deep unfolded version of the PGD algorithm is proposed. It is shown that this unfolded algorithm converges to close-to-optimal solutions, much faster than the traditional PGD.

Notations: Vectors and matrices are denoted by bold lowercase and bold uppercase letters respectively. Superscripts $(\cdot)^*$, $(\cdot)^\top$ and $(\cdot)^H$ stand for the conjugate, transpose, and Hermitian transpose operators respectively. Subscripts $(\cdot)_m$ and $(\cdot)_k$ denote the antenna and user index. The expectation is denoted by $\mathbb{E}\{\cdot\}$, while $\mathbf{D}_\mathbf{a} = \text{diag}(\mathbf{a})$ denotes a diagonal matrix whose diagonal entries are equal to the vector \mathbf{a} . The element at location (i, j) in the matrix \mathbf{A} is indicated as $[\mathbf{A}]_{i,j}$. The symbols $\|\cdot\|_1$, $\|\cdot\|_2$, $\|\cdot\|_{2,1}$ and $\|\cdot\|_F$ indicate the L_1 , L_2 , $L_{2,1}$ and Frobenius norms, respectively. $\mathbf{A}^{1/2}$ denotes the positive square root of \mathbf{A} , namely the only Hermitian positive semidefinite matrix such that $\mathbf{A} = \mathbf{A}^{1/2} \mathbf{A}^{1/2}$.

II. SYSTEM MODEL

A. Signal Model

Throughout this work, a massive MIMO base station (BS) is considered, equipped with M transmit antennas and serving K single-antenna users. The precoded symbol vector $\mathbf{x} \in \mathbb{C}^{M \times 1}$ is given by

$$\mathbf{x} = \mathbf{W}^\top \mathbf{s} \quad (1)$$

where $\mathbf{W} \in \mathbb{C}^{K \times M}$ is the precoding matrix and $\mathbf{s} \in \mathbb{C}^{K \times 1}$ are the transmit symbols, which are assumed to be zero mean, uncorrelated and have unit power. The received signal $\mathbf{r} \in \mathbb{C}^{K \times 1}$ is then

$$\mathbf{r} = \mathbf{H}\mathbf{x} + \boldsymbol{\nu} \quad (2)$$

where $\mathbf{H} \in \mathbb{C}^{K \times M}$ is the channel matrix. The vector $\boldsymbol{\nu} \in \mathbb{C}^{K \times 1}$ contains complex independently and identically distributed (i.i.d.) additive white Gaussian noise (AWGN) with zero mean and variance σ_ν .

B. Power Consumption Model

Given that the transmit symbols are uncorrelated, the transmit power at antenna m is

$$p_m = \mathbb{E}\{|x_m|^2\} = \sum_{k=0}^{K-1} |w_{k,m}|^2 \quad (3)$$

where $x_m = [\mathbf{x}]_m$ is the precoded symbol at antenna m and $w_{k,m} = [\mathbf{W}]_{k,m}$ is the precoding coefficient at antenna m for user k . This gives a total transmit power

$$p_{\text{tx}} = \sum_{m=0}^{M-1} \sum_{k=0}^{K-1} |w_{k,m}|^2 = \|\mathbf{W}\|_F^2. \quad (4)$$

When considering a realistic power consumption model, the transmit power does not scale linearly with the consumed power. In other words, the efficiency of the PA is not fixed but varies with the transmit power, i.e., typically the closer the PA operates to saturation, the higher the efficiency [7]. For instance, for class B amplifiers, the efficiency scales with the square root of the transmit power [6], [9]

$$\eta_m = \eta_{\max} \sqrt{\frac{p_m}{p_{\max}}} \quad (5)$$

where η_m is the efficiency of the m -th PA, η_{\max} is the maximal PA efficiency and p_{\max} is the maximal output power of the PA. The total consumed power by the PAs is then

$$\begin{aligned} p_{\text{cons}} &= \sum_{m=0}^{M-1} \frac{p_m}{\eta_m} = \underbrace{\frac{\sqrt{p_{\max}}}{\eta_{\max}}}_{=\alpha} \sum_{m=0}^{M-1} \sqrt{p_m} \\ &= \alpha \sum_{m=0}^{M-1} \sqrt{\sum_{k=1}^K |w_{k,m}|^2} = \alpha \|\mathbf{W}\|_{2,1}. \end{aligned} \quad (6)$$

III. PROBLEM FORMULATION

Classical precoders such as ZF minimize the transmit power under a QoS constraint (e.g., a per-user SINR constraint) [5]. However, when aiming to reduce the PAs' consumed power, one should minimize the consumed power given in (6) rather than the transmit power in (4). As such, a ZF precoder under the PA consumption model in (6) can be formulated as

$$\begin{aligned} \min_{\mathbf{W}} \quad & \alpha \|\mathbf{W}\|_{2,1} \\ \text{s.t.} \quad & \mathbf{H}\mathbf{W}^\top = \sigma_\nu \mathbf{D}_\gamma^{1/2} \end{aligned} \quad (7)$$

where $\mathbf{D}_\gamma = \text{diag}(\gamma_0, \dots, \gamma_{K-1})$ and γ_k is the target SINR for user k . In this formulation, the consumed power by the PAs is minimized under a SINR constraint per user. The precoder design problem (7) has been analyzed in [8] and can be solved by using convex optimization toolboxes (e.g., CVXPY [15]). Problem (7) is equivalent to

$$\begin{aligned} \min_{\mathbf{W}} \quad & \alpha \|\mathbf{W}\|_{2,1} \\ \text{s.t.} \quad & \left\| \mathbf{H}\mathbf{W}^\top - \sigma_\nu \mathbf{D}_\gamma^{1/2} \right\|_F^2 = 0. \end{aligned} \quad (8)$$

This optimization problem can be reformulated using a Lagrangian function as follows

$$\begin{aligned}\mathcal{L} &= \alpha \|\mathbf{W}\|_{2,1} + \mu \left\| \mathbf{H}\mathbf{W}^\top - \sigma_\nu \mathbf{D}_\gamma^{1/2} \right\|_F^2 \\ &= \mu \left(\lambda \|\mathbf{W}\|_{2,1} + \left\| \mathbf{H}\mathbf{W}^\top - \sigma_\nu \mathbf{D}_\gamma^{1/2} \right\|_F^2 \right)\end{aligned}\quad (9)$$

where $\lambda = \alpha/\mu$ and μ is the Lagrange multiplier.

IV. PROXIMAL GRADIENT DESCENT

The Lagrangian function in (9) is of the form

$$\mathcal{L} = \lambda g(\mathbf{W}) + f(\mathbf{W}) \quad (10)$$

with $f(\mathbf{W})$ a convex, differentiable function and $g(\mathbf{W})$ a convex, non-differentiable function. Given this form, the problem can be solved using PGD [16]. This algorithm first performs a gradient update with respect to the differentiable function $f(\mathbf{W})$. Next, the proximal operator of $\lambda g(\mathbf{W})$ is computed on the result of the gradient update. The general formulation of the iterative proximal algorithm is given by

$$\mathbf{W}^{(i+1)} = \mathbf{prox}_{\lambda \eta g} \left(\mathbf{W}^{(i)} - \eta \nabla f(\mathbf{W}^{(i)}) \right) \quad (11)$$

where η is the gradient step size and $\mathbf{prox}_{\lambda \eta g}(\cdot)$ is the proximal operator of the function $\lambda g(\cdot)$. Since \mathbf{W} is a complex matrix, $\nabla f(\mathbf{W})$ is defined as $\nabla f(\mathbf{W}) = \partial f(\mathbf{W})/\partial \mathbf{W}^* = \mathbf{W}\mathbf{H}^\top \mathbf{H}^* - \sigma_\nu \mathbf{D}_\gamma^{1/2} \mathbf{H}^*$. Additionally, the proximal operator is only defined for real matrices, and consequently the real and imaginary parts of the complex matrix are concatenated in order to produce the following algorithm

$$\begin{aligned}\mathbf{V} &= \mathbf{W}^{(i)} - \eta \left(\mathbf{W}^{(i)} \mathbf{H}^\top \mathbf{H}^* - \sigma_\nu \mathbf{D}_\gamma^{1/2} \mathbf{H}^* \right) \\ \begin{bmatrix} \mathbf{W}_{\Re}^{(i+1)} \\ \mathbf{W}_{\Im}^{(i+1)} \end{bmatrix} &= \mathbf{prox}_{\lambda \eta \|\cdot\|_{2,1}} \left(\begin{bmatrix} \mathbf{V}_{\Re} \\ \mathbf{V}_{\Im} \end{bmatrix} \right).\end{aligned}\quad (12)$$

Here, λ is dependent on the Lagrange multiplier μ and controls the trade-off between minimizing $g(\mathbf{W})$ and $f(\mathbf{W})$. Furthermore, convergence is guaranteed if the step size for the gradient update is $\eta \leq 1/L$, where L is the Lipschitz constant of $\nabla f(\mathbf{W}^{(i)})$, which is equivalent to an upper bound on the largest eigenvalue of $\mathbf{H}^\top \mathbf{H}^*$ [16].

In order to define the proximal operator of the $L_{2,1}$ norm, we use the fact that the $L_{2,1}$ norm is a separable function, as it can be expressed as the L_1 norm over the column-wise L_2 norm. Hence, using the property of proximal operators for separable functions [17], the proximal of the $L_{2,1}$ norm simply becomes a series of L_2 -proximals on each column of \mathbf{W} . As such, the proximal operator for column m of \mathbf{W} (denoted as \mathbf{w}_m) can be expressed as

$$\begin{aligned}\mathbf{prox}_{\lambda \eta \|\cdot\|_{2,1}}(\mathbf{w}_m) &= \left(1 - \frac{\lambda \eta}{\max(\|\mathbf{w}_m\|_2, \lambda \eta)} \right) \mathbf{w}_m \quad \forall m \\ &= \begin{cases} \left(1 - \frac{\lambda \eta}{\|\mathbf{w}_m\|_2} \right) \mathbf{w}_m & \text{if } \|\mathbf{w}_m\|_2 \geq \lambda \eta \\ 0 & \text{if } \|\mathbf{w}_m\|_2 < \lambda \eta \end{cases}.\end{aligned}\quad (13)$$

V. DEEP UNFOLDED PROXIMAL GRADIENT DESCENT

The unfolded proximal gradient descent (UfPGD) algorithm follows a similar structure as the algorithm in (12). The algorithm is unrolled for a fixed number of iterations I . As such, a neural network consisting of I layers is constructed, with each layer performing the following operations

$$\begin{aligned}\mathbf{V} &= \mathbf{W}^{(i)} - \eta^{(i)} \left(\mathbf{W}^{(i)} \mathbf{H}^\top \mathbf{H}^* - \sigma_\nu \mathbf{D}_\gamma^{1/2} \mathbf{H}^* \right) \\ \begin{bmatrix} \mathbf{W}_{\Re}^{(i+1)} \\ \mathbf{W}_{\Im}^{(i+1)} \end{bmatrix} &= \mathbf{prox}_{\lambda^{(i)} \eta^{(i)} \|\cdot\|_{2,1}} \left(\begin{bmatrix} \mathbf{V}_{\Re} \\ \mathbf{V}_{\Im} \end{bmatrix} \right).\end{aligned}\quad (14)$$

The key difference between (14) and (12) is the fact that, in the unfolded iteration, the parameters $\lambda^{(i)}$ and $\eta^{(i)}$ are learned parameters that are optimized for each iteration/layer. More specifically, the parameters $\lambda^{(i)}$ and $\eta^{(i)}$ are learned by minimizing a certain cost function $C(\mathbf{W}^{(I)})$, where $\mathbf{W}^{(I)}$ is the output of the neural network (i.e., the result at the I -th layer). For instance, this can be done by using stochastic gradient descent in the following manner

$$\lambda_{\text{new}}^{(i)} = \lambda^{(i)} - \beta \frac{\partial C(\mathbf{W}^{(I)})}{\partial \lambda^{(i)}} \quad \forall i \in [1, 2, \dots, I] \quad (15)$$

$$\eta_{\text{new}}^{(i)} = \eta^{(i)} - \beta \frac{\partial C(\mathbf{W}^{(I)})}{\partial \eta^{(i)}} \quad \forall i \in [1, 2, \dots, I] \quad (16)$$

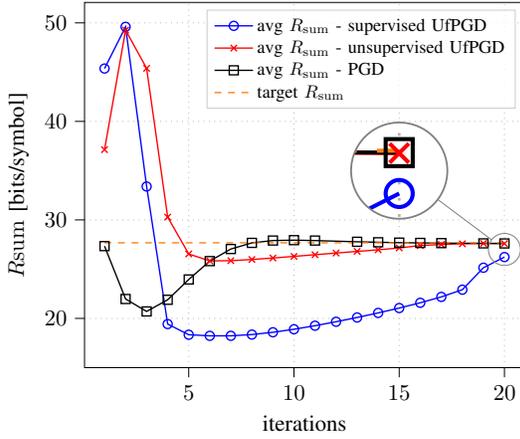
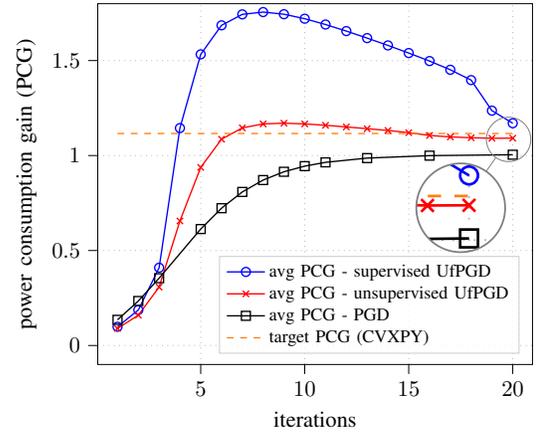
where $\partial C(\mathbf{W}^{(I)})/\partial \eta^{(i)}$ and $\partial C(\mathbf{W}^{(I)})/\partial \lambda^{(i)}$ can be computed using backpropagation, and β is the learning rate. The cost function can be defined in two ways, namely in a supervised or unsupervised manner. When training in a supervised way, the cost function is simply the mean square error (MSE) between the output of the neural network $\mathbf{W}^{(I)}$ and the known ground truth label for the optimal precoding matrix \mathbf{W}_{gt}

$$C(\mathbf{W}^{(I)}) = \left\| \mathbf{W}^{(I)} - \mathbf{W}_{\text{gt}} \right\|_F^2. \quad (17)$$

The ground truth labels can be obtained using convex solvers such as CVXPY [15]. When using the supervised learning approach, the MSE acts as a surrogate for the real cost function we want to optimize, which is the Lagrangian in (9). Conversely, one can also train the network in an unsupervised manner by replacing this surrogate loss function with the real cost function which is defined as

$$C(\mathbf{W}^{(I)}) = \lambda \left\| \mathbf{W}^{(I)} \right\|_{2,1} + \left\| \mathbf{H} \left(\mathbf{W}^{(I)} \right)^\top - \sigma_\nu \mathbf{D}_\gamma^{1/2} \right\|_F^2. \quad (18)$$

Finally, in order to ensure stability during the training, the values of $\lambda^{(i)}$ and $\eta^{(i)}$ are projected into their desired intervals. First, $\lambda^{(i)}$ should be in $[0, +\infty]$, leading to the following projection $\lambda_{\text{projected}}^{(i)} = \max(0, \lambda^{(i)})$. Second, for the classical PGD $\eta^{(i)} = 1/L$, where L is an upper bound on the largest eigenvalue of $\mathbf{H}^\top \mathbf{H}^*$. This upper bound can be approximated by the upper limit of the support of the Marchenko-Pastur law, which is (up to scaling) the asymptotic eigenvalue distribution for matrices with i.i.d., zero mean and unit variance entries.

(a) Average R_{sum} 

(b) Average power consumption gain

Fig. 1: Sum rate and power consumption gain (PCG) convergence in function of the number of iterations/layers of PGD, unfolding with supervised training and unfolding with unsupervised training. The results are averaged over 5000 channel realizations taken from the test set. Note that the target R_{sum} and PCG (dashed) are obtained using CVXPY.

This approximation is given by $\tilde{L} = (\sqrt{K} + \sqrt{M})^2$ [18]. This bound holds with probability one for sufficiently large K and M . Hence, the value of $\eta^{(i)}$ is always projected into the following interval $[1/(2\tilde{L}), 1/\tilde{L}]$, where the projection is defined as $\eta_{\text{projected}}^{(i)} = \min(\max(\eta^{(i)}, 1/(2\tilde{L})), 1/\tilde{L})$.

VI. SIMULATIONS

A. Training and Hyperparameters Selection

In this section, simulations are performed to compare the performance of the unfolded and the classical PGD algorithm. For the following simulations, the training set consists of 10^5 Rayleigh fading channels sampled from a complex normal distribution with zero mean and variance one, i.e., $[\mathbf{H}]_{k,m} \sim \mathcal{CN}(0, 1)$. The hyperparameters of the unfolding algorithm, such as the learning rate, batch size, etc. were tuned on a validation set of 5000 Rayleigh fading channels. Finally, all simulation results were obtained on an independent test set of 5000 Rayleigh fading channels. For training, the Adam optimizer [19] is used with a batch size of 64 and a learning rate of 0.001. The unfolded algorithm consists of $I = 20$ unfolded iterations/layers and was trained for 200 epochs, with early stopping if the validation loss did not further decrease.

B. Performance Evaluation Metrics

To compare the performance of the unfolded and classical algorithms, two metrics are considered. First, the sum rate

$$R_{\text{sum}} = \sum_{k=0}^{K-1} \log_2(1 + \text{SINR}_k) \quad (19)$$

where SINR_k is the SINR of user k . Given a per-user SINR constraint of 10 dB and $K = 8$ users, the target sum rate the algorithm should achieve is 27.68 bits/symbol. Second, the power consumption gain (PCG) is defined as

$$\text{PCG} = \frac{p_{\text{cons}}^{\text{ZF}}}{p_{\text{cons}}^{\text{ZF-eff}}} \quad (20)$$

where $p_{\text{cons}}^{\text{ZF}}$ is the consumed power of the ZF precoder, and $p_{\text{cons}}^{\text{ZF-eff}}$ is the consumed power of the proposed power efficient precoder (7), the consumed power is computed using (6).

C. Simulation Results

To compare the convergence of PGD and UfPGD, the sum rate and PCG are evaluated at each iteration/layer. This is done for $K = 8, M = 64$, a target per-user SINR of $\gamma_k = 10$ dB and a noise variance of $\sigma_v = 1$. Note that $\lambda = 1/15$ for both the PGD algorithm and the unsupervised cost function. Additionally, both PGD and UfPGD are initialized with the conjugate of the channel, i.e., $\mathbf{W}^{(0)} = \mathbf{H}^*$.

In Fig. 1a, the sum rate per iteration is plotted for PGD and UfPGD with supervised and unsupervised training. From this figure it is clear that both PGD and UfPGD with unsupervised training can achieve the target sum rate of 27.68 bits/symbol. When training in a supervised manner, the unfolded algorithm is not able to achieve the goal sum rate. Given that for supervised training the MSE acts as a surrogate for the actual loss function, it is not able to capture the importance of achieving the SINR constraint. When using unsupervised training, the real objective function is optimized, which enables the algorithm to achieve the SINR constraint.

In Fig. 1b, the PCG per iteration is plotted for PGD and UfPGD. This figure shows that after 20 iterations, the PGD algorithm reaches a PCG of approximately one, indicating that it consumes the same amount of energy as the classical ZF precoder. It is widely known that PGD suffers from slow convergence, as is illustrated in Fig. 2, where the PCG is depicted when the algorithm is run for more iterations. Indeed, from this figure, it is clear that PGD does eventually reach the target PCG. However, it needs over 3500 iterations to get close to the optimal solution. This highlights the benefit of using the unfolded algorithm, as after 20 iterations it is close to the optimal solution. This is depicted in Fig. 1b, where after 20 iterations the unsupervised UfPGD algorithm

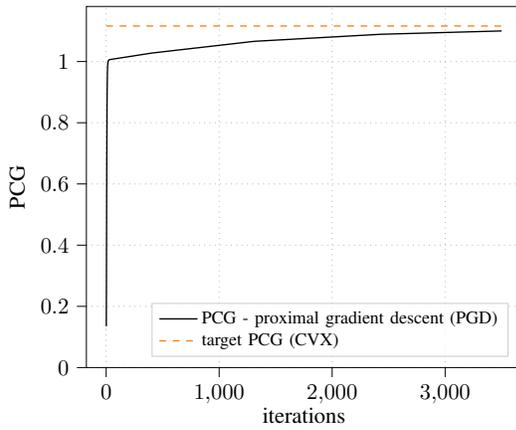


Fig. 2: Power consumption gain (PCG) per iteration of PGD, versus the target PCG of CVXPY. Averaged over 5000 channel realizations.

reaches a $PCG = 1.0915$, where the target value (obtained with CVXPY) is $PCG = 1.1160$.

When comparing the execution time of UfPGD, PGD¹ and CVXPY in Fig. 3, it is clear that the unfolding algorithm has the smallest execution time. This is expected as the unfolding algorithm only needs 20 iterations compared to the 5000 needed by PGD. Additionally, it is clear that general-purpose numerical solvers such as CVXPY cannot reach sufficiently low execution times, as the execution times are three orders of magnitude larger than the unfolding algorithm.

VII. CONCLUSION

In this work, a fast unfolded optimization algorithm is developed, for a massive MIMO precoder that minimizes the consumed power rather than the transmit power. It is shown that the optimization problem gives rise to an objective function that consists of a convex, differentiable and a convex, non-differentiable part. Given this form, the optimization problem is solved using PGD. However, given the slow convergence rate of the PGD algorithm, a deep unfolded version of the algorithm is proposed. The proposed algorithm is highly optimized for the task at hand, which allows it to achieve close-to-optimal solutions in only 20 iterations, as compared to the 3500 plus iterations needed by the PGD algorithm. Finally, the execution time of the proposed algorithm is compared against a conventional numerical convex solver (CVXPY), showing that the proposed algorithm is three orders of magnitude faster.

REFERENCES

- [1] European Commission, “The European Green Deal,” *COM (2019)*, November 2019.
- [2] United Nations, “The 2030 Agenda and the Sustainable Development Goals: An opportunity for Latin America and the Caribbean,” (LC/G.2681-P/Rev.3), Santiago, 2018.
- [3] L. Belkhir and A. Elmehri, “Assessing ICT global emissions footprint: Trends to 2040 & recommendations,” *Journal of Cleaner Production*, vol. 177, pp. 448–463, Mar. 2018.

¹In order to reach the optimal solution, PGD is executed for 5000 iterations.

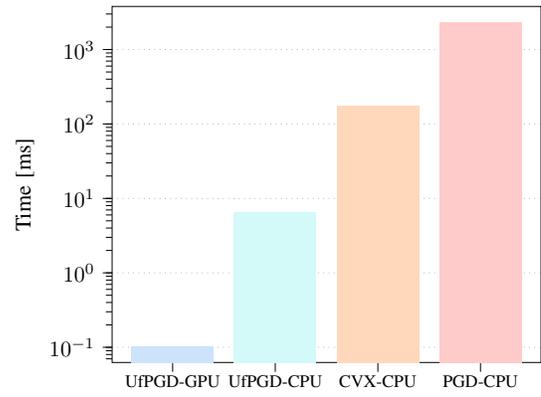


Fig. 3: Execution time of UfPGD on GPU and CPU, CVX and PGD, both on CPU (ran for 5000 iterations per execution). Averaged over 5000 executions.

- [4] E. Björnson, L. Sanguinetti, H. Wymeersch, J. Hoydis, and T. L. Marzetta, “Massive mimo is a reality—what is next?: Five promising research directions for antenna arrays,” *Digital Signal Processing*, vol. 94, pp. 3–20, 2019, special Issue on Source Localization in Massive MIMO.
- [5] E. Björnson, J. Hoydis, and L. Sanguinetti, “Massive MIMO networks: Spectral, energy, and hardware efficiency,” *Foundations and Trends® in Signal Processing*, vol. 11, no. 3-4, pp. 154–655, 2017.
- [6] A. Grebennikov, *RF and Microwave Power Amplifier Design*, 1st ed. NY, USA: McGraw-Hill, 2005.
- [7] S. Mikami, T. Takeuchi, H. Kawaguchi, C. Ohta, and M. Yoshimoto, “An Efficiency Degradation Model of Power Amplifier and the Impact against Transmission Power Control for Wireless Sensor Networks,” in *2007 IEEE Radio and Wireless Symposium*, 2007, pp. 447–450.
- [8] E. Peschiera and F. Rottenberg, “Linear Precoder Design in Massive MIMO under Realistic Power Amplifier Consumption Constraint,” in *42nd WIC Symposium on Information Theory and Signal Processing in the Benelux*, 2022.
- [9] H. V. Cheng, D. Persson, and E. G. Larsson, “Optimal MIMO Precoding Under a Constraint on the Amplifier Power Consumption,” *IEEE Transactions on Communications*, vol. 67, no. 1, pp. 218–229, 2019.
- [10] V. Monga, Y. Li, and Y. C. Eldar, “Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing,” *IEEE Signal Processing Magazine*, vol. 38, no. 2, pp. 18–44, 2021.
- [11] K. Gregor and Y. LeCun, “Learning fast approximations of sparse coding,” in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML’10. Madison, WI, USA: Omnipress, 2010, p. 399–406.
- [12] A. Balatsoukas-Stimming and C. Studer, “Deep unfolding for communications systems: A survey and some new directions,” 2019. [Online]. Available: <https://arxiv.org/abs/1906.05774>
- [13] Q. Hu, Y. Cai, Q. Shi, K. Xu, G. Yu, and Z. Ding, “Iterative algorithm induced deep-unfolding neural networks: Precoding design for multiuser mimo systems,” *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1394–1410, 2021.
- [14] J. Xu, C. Kang, J. Xue, and Y. Zhang, “A fast deep unfolding learning framework for robust mu-mimo downlink precoding,” *IEEE Transactions on Cognitive Communications and Networking*, pp. 1–1, 2023.
- [15] S. Diamond and S. Boyd, “CVXPY: A Python-embedded modeling language for convex optimization,” *J. Mach. Learn. Res.*, vol. 17, no. 83, pp. 1–5, 2016.
- [16] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski, “Optimization with Sparsity-Inducing Penalties,” 2011. [Online]. Available: <https://arxiv.org/abs/1108.0775>
- [17] N. Parikh and S. Boyd, “Proximal Algorithms,” *Found. Trends Optim.*, vol. 1, no. 3, p. 127–239, jan 2014. [Online]. Available: <https://doi.org/10.1561/2400000003>
- [18] V. A. Marčenko and L. A. Pastur, “Distribution of eigenvalues for some sets of random matrices,” *Mathematics of The USSR-sbornik*, vol. 1, pp. 457–483, 1967.
- [19] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014. [Online]. Available: <https://arxiv.org/abs/1412.6980>