Many thanks to all the reviewers for their time and attention to our work!

For Reviewer 1: We certainly agree that our techniques are modifications of prior ideas, but we feel that significant insight was required to develop and apply them in our particular manner. This is hard to prove as, once written down, the approach is quite simple (although this is of course actually a positive trait). A difficulty here is that one might be tempted to believe that at least our Theorem 6 can be easily obtained directly from prior results, and it is only after trying a bit (as we did for quite some time) that one concludes that things are not so easy. For some more evidence in this regard, and for your question about applying [17] in constrained settings, please see our response to Reviewer 3 on how the naive approach can fail. We'd also like to stress that obtaining Theorem 7 is perhaps a better exemplar since there isn't even an obvious-but-secretly-problematic approach until viewed under our lens.

We also agree it would be better to be scale-free as well as parameter-free, but as you state this is not possible in general. In fact, it seems even the compromise of adding a penalty of $O(\|\mathring{w}\|^3 + \sqrt{T})$ cannot be achieved without sacrificing the second-order gradient statistics in the bounds [Mhammedi&Koolen 2020]. That said, we agree that finding what is achievable here is a good problem - in particular we suspect this may be possible in the constrained setting subject to an additive penalty of $O(GD)$, where G is the maximum Lipschitz constant and D is the diameter of the domain.

For Reviewer 2: Regarding efficiency: The parameter-free scaling algorithm can run in $O(d)$ time per update as it just needs the inner-product $\langle g_t, x_t \rangle$ to compute its loss. The FTRL algorithm requires the same matrix manipulations that a corresponding unconstrained algorithm would require to get the same regret bound, and so for the full-matrix algorithms here this involves a $O(d^2)$ rank-one update step, and the final combined algorithm requires a projection step that may depend on the domain. Concretely, the algorithm of Theorem 7 runs as fast as full-matrix AdaGrad.

For Reviewer 3: We appreciate your comments, and the references you suggest are highly relevant - in particular the recent FreeGrad algorithm of [Mhammedi&Koolen 2020] is a better comparison in the unconstrained case than the full-matrix algorithm of [17].

In regards to novelty, we must disagree here. It is certainly very reasonable to suspect that the unconstrained-to-constrained technique suggested by [17] can be used to immediately convert an algorithm like full-matrix FreeGrad into a constrained algorithm - we also thought this would work at first! **However, it does not**. As it turns out, the fact that this approach is not as easy as it seems was actually our original motivation for working on this problem!

Using the constraint-set reduction of [17] in concert with an unconstrained algorithm like FreeGrad is problematic because the reduction changes the losses supplied to FreeGrad, which will in turn change its regret guarantee. This is because the regret bound of FreeGrad depends in a delicate manner on the values of the g_t. If the original losses are g_t, and the gradients supplied to FreeGrad are $\tilde{g}_t$, the final regret will be $\tilde{O}\left(\sqrt{\tilde{r}\sum_{t=1}^T \langle \tilde{g}_t, \mathring{w}\rangle^2}\right)$, where $\tilde{r}$ is the rank of the $\tilde{g}_t$. It is not clear what the relationship is between this quantity and the desired bound $\tilde{O}\left(\sqrt{r\sum_{t=1}^T \langle g_t, \mathring{w}\rangle^2}\right)$.

Here is a sketch of what one might try, and where it goes wrong. To start, we need to pick a norm to use with the constraint set reduction. The natural choice here is $\|\cdot\|_{G_T}$, so let us say we already know the matrix G_T ahead of time, and use this norm so that we have $\|\tilde{g}_t\|_{G_T^{-1}} \le \|g_t\|_{G_T^{-1}}$. This is unrealistic, but even with this extra power it is not clear that things work: using the most obvious algebraic path of bounding $\sum_{t=1}^T \langle \mathring{w}, \tilde{g}_t\rangle^2 \le \sum_{t=1}^T \|\mathring{w}\|_{G_T}^2 \|\tilde{g}_t\|_{G_T^{-1}}^2$ yields a bound like $\tilde{O}\left(\|\mathring{w}\|_{G_T}\sqrt{r\sum_{t=1}^T \|g_t\|_{G_T^{-1}}^2}\right)$. Now, the $\sum_{t=1}^T \|g_t\|_{G_T^{-1}}^2$ term may add an extra dependence on r that prevents us from achieving the desired bound. To make this work, we need to guarantee that the matrix $\sum_{t=1}^T g_t g_t^\top$ induces the same norm up to an absolute constant as the matrix $\sum_{t=1}^T \tilde{g}_t \tilde{g}_t^\top$, and it is not clear that this will hold. We are aware that for the case of Metagrad-style bounds [van Erven&Koolen 2016], the work of [Mhammedi et. al 2019] invokes some clever algebra to show that the reduction works as-is, but their technique does not seem to apply to get the bounds we are looking for. In contrast, our approach does not encounter these issues, and so we feel justified in saying that we are in fact the first to provide an algorithm achieving the desired full-matrix bound in the constrained setting.

Moreover, even if there were some unknown algebraic trick that would make the prior reduction apply, **our approach has obvious additional benefits**: we can easily obtain the optimally-tuned full-matrix AdaGrad-style bound that is not clear how to obtain (even in the unconstrained case) using other techniques.

We hope this addresses your concern.