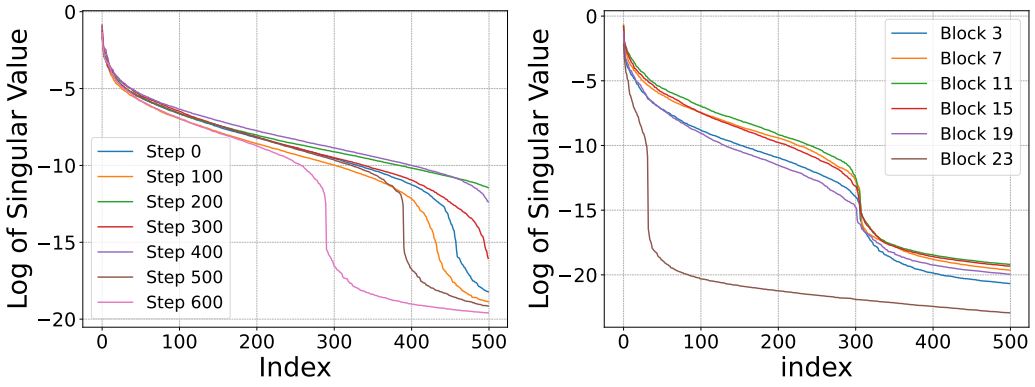
8. Appendix

8.1. Additional Results

Empirical Evidence of BP Gradients Converging to Subspaces

Previous studies have shown that BP gradients during fine-tuning of LLMs rapidly converge to low-dimensional subspaces. Building on this, our empirical investigation using the OPT-1.3B architecture on the SST-2 dataset uncovers a persistent low-rank structure in gradient matrices throughout the optimization process. As shown in Fig. 3, singular value decomposition (SVD) of gradient matrices across layers consistently reveals pronounced spectral decay, with only a small subset of singular values dominating the gradient spectrum during LLM fine-tuning with SGD.

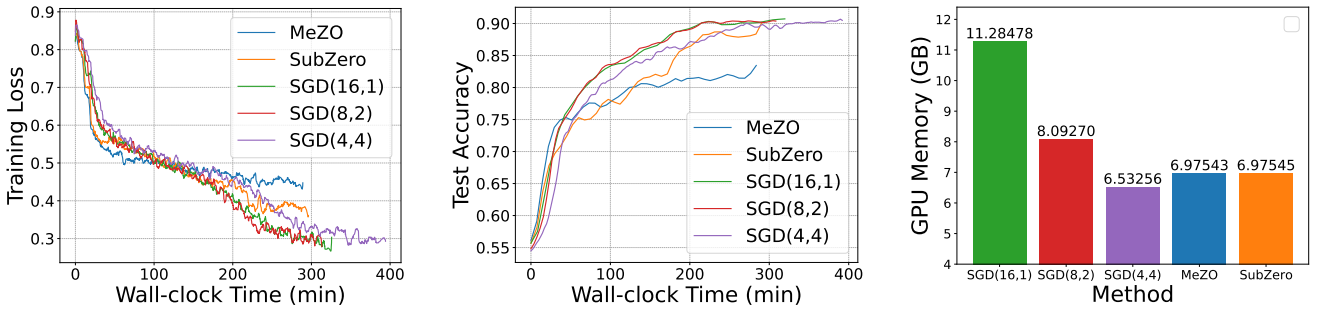


(a) Singular values of `k_proj` matrix for Layer 22. (b) Singular value of `q_proj` matrix graident over step 100.

Figure 3. Low-rank structures in gradient matrices during LLM fine-tuning. (a) illustrates the temporal evolution of singular values for key projections within a single layer, showing consistent spectral decay across training steps. (b) presents the cross-layer comparison at step 100, revealing layer-invariant low-rank patterns in query projection gradients.

More Comparisons

In the main manuscript, we use the identical batch size for FO and ZO optimizers. Here, we adjust SGD with gradient accumulation to match the memory usage of ZO optimizers, and then compare their convergence speed and performance. The experimental settings are the same as those in Fig. 1, and the experimental results are shown in Fig. 4. With similar memory usage, SubZero attains a convergence rate nearly on par with SGD, surpasses MeZO, and achieves test accuracy comparable to that of SGD.



(a) Training Loss (b) Test Accuracy (c) Memory Cost

Figure 4. Visualization of training loss, test accuracy, and peak total GPU memory usage with OPT-1.3B on SST-2 in prompt tuning scheme. SGD(BS, GA) refers to SGD with a batch size of BS and GA times of gradient accumulation. All ZO methods utilize a batch size of 16, while SGD(BS, GA) applies gradient accumulation to ensure its memory usage aligns with that of the ZO optimizers. All methods are executed for 20K steps.

The vanilla LoRA is fine-tuned by Adam. We compare SubZero with SGD in the FT and LoRA schemes with vanilla LoRA using the pretrained OPT-1.3B model on SST-2. For Adam and SubZero with SGD, we apply the constant learning rate

schedule. The results are given in Table 9. We can see that SubZero with SGD in the FT scheme outperforms vanilla LoRA in both test accuracy and memory usage. SubZero with SGD in the LoRA scheme also achieves comparable test accuracy while maintaining minimal memory usage.

Table 9. Comparison with vanilla LoRA using the pretrained OPT-1.3B model on SST-2.

Method	Test Accuracy(%)	Total Memory (GB)
LoRA (Adam)	93.2	10.75
SubZero (FT)	**93.4**	6.88
SubZero (LoRA)	92.9	**6.80**

As described in Sec. 6.3, we compare the memory consumption and wall-clock time of ZO methods (MeZO and SubZero), SGD, and inference-only approaches (zero-shot and in-context learning (ICL)) using OPT-13B (see Table 10). Since inference-only methods do not involve fine-tuning, they have zero wall-clock time and their memory usage reflects only the inference load. For fine-tuning, all methods were run for 20K steps. The ZO methods, including SubZero, achieved over a 1.8× reduction in memory usage compared to SGD. Notably, SubZero's memory footprint closely aligns with MeZO's, while offering improved performance. We use per-layer weight updates for MeZO and SubZero (see Appendix 8.2), resulting in nearly identical memory usage for FT and LoRA schemes when one decimal place is reserved.

Although SubZero introduces computational and memory overhead due to QR decomposition when generating projection matrices, our empirical analysis reveals strictly bounded resource costs across all tested OPT model scales (see Table 11). Specifically, the additional time overhead remains below 8.5% (peaking at 8.36% for the 6.7B model), while the memory overhead stays under 1.8%, even under bfloat16 precision. This indicates that the computational cost of QR decomposition becomes asymptotically negligible as model complexity increases, thereby establishing SubZero's practical scalability.

Table 10. Memory usage (GB) and wall-clock time (minutes) of fine-tuning OPT-13B, with SGD's batch size being 8 for SQuAD and 16 for other tasks.

Task	SST-2		WIC		SQuAD	
Method	**Mem.**	**Time**	**Mem.**	**Time**	**Mem.**	**Time**
Zero-shot/ICL	24.2	0	24.8	0	27.2	0
SGD(FT)	48.9	190.3	48.9	257.3	122.7	623.7
MeZO(FT)	26.1	324.9	26.6	370.5	37.4	670.2
SubZero(FT)	26.5	337.3	27.1	385.3	37.8	690.5
MeZO(LoRA)	26.1	123.9	26.6	171.6	37.4	476.7
SubZero(LoRA)	26.1	130.3	26.6	179.7	37.4	486.5

Table 11. Peak memory usage (GB) and wall-clock time (seconds) for fine-tuning OPT models on the SST-2 dataset. All models fine-tuned on SST-2 for 20K steps. Precision strategy: FP32 for models $\leq$ 6.7B, BF16 for $\geq$ 6.7B.

OPT series	hidden size	MeZO Mem.	SubZero Mem.	Mem. Overhead (%)	MeZO Time	SubZero Time	Time Overhead (%)
1.3B	2048	6.80	6.88	**+1.18%**	11214.95	11683.39	**+4.18%**
2.7B	2560	12.40	12.60	**+1.61%**	21578.56	22335.05	**+3.51%**
6.7B	4096	13.98	14.20	**+1.57%**	9832.69	10654.59	**+8.36%**
13B	5120	26.08	26.53	**+1.73%**	18667.50	19245.10	**+3.09%**

Integration with Other ZO Optimizers

SubZero is orthogonal to many ZO methods and can be combined with them to further boost performance, such as the momentum mechanism in ZO-AdaMU [28], the sparsity pruning in S-MeZO [39], and the second-order information in HiZOO [68]. Here, we report the experimental results of integrating SubZero with ZO-AdaMU, as shown in Table 12 and

Fig. 5. This integration not only achieves faster convergence but also significantly reduces the memory overhead associated with ZO-AdaMU's momentum mechanism.

Method	Memory (GB)	Accuracy (%)
MeZO	**26.62**	60.0
SubZero	27.06	60.8
ZO-AdaMU	50.77	60.7
SubZO-AdaMU	27.07	**61.1**

Table 12. Peak memory usage (GB) and test accuracy (%) for fine-tuning OPT-13B models on the WIC dataset.

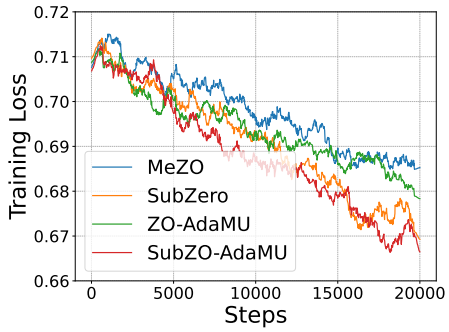


Figure 5. Training loss curves of Table 12.

ZO-AdaMU accelerates convergence by introducing stochastic momentum, a mechanism that is orthogonal to SubZero. In all experiments, we use a shared learning rate and perturbation scale. SubZero-specific hyperparameters, such as rank and update interval, are set as detailed in Table 3, while all momentum-related settings remain consistent with the original ZO-AdaMU paper [28]. Notably, vanilla ZO-AdaMU requires storing the full momentum history, leading to substantial memory overhead. In contrast, when combined with SubZero, only low-dimensional momentum $\boldsymbol{M}_i^t \in \mathbb{R}^{r\times r}$ needs to be maintained, i.e.,

$$\boldsymbol{M}_i^t = \beta_i^t \boldsymbol{M}_i^{t-1} + (1-\beta_i^t)\boldsymbol{Z}_i^t, \tag{17}$$

$$\boldsymbol{W}_i^t = \boldsymbol{W}_i^{t-1} - \eta_i^t \boldsymbol{U}_i \boldsymbol{M}_i^t \boldsymbol{V}_i^T. \tag{18}$$

When the subspaces are updated at time steps t such that $t \mod F \equiv 0$ (as specified in Algorithm 3), the old momentum needs to be projected from the old subspaces $\boldsymbol{U}^{t-1}$ and $\boldsymbol{V}^{t-1}$ onto the new subspaces $\boldsymbol{U}^t$ and $\boldsymbol{V}^t$. This projection ensures the optimizer's continuity and maintains the effectiveness of the momentum mechanism. The projection is formulated as the following optimization problem:

$$\min_{M} \|\boldsymbol{U}^{t-1}\boldsymbol{M}^{t-1}(\boldsymbol{V}^{t-1})^T - \boldsymbol{U}^t \boldsymbol{M}(\boldsymbol{V}^t)^T\|. \tag{19}$$

More Ablation Studies

We investigate the effects of random seed, batch size, and combination with Adam for SubZero. We also provide more results on the reshaping strategy.

We first fine-tune the OPT-1.3B model on the SST-2 dataset in prompt tuning scheme with three random seeds. We present the results in Table 13, and hyperparameters are presented in Table 18. For various random seeds, the variance of MeZO is quite large, whereas the variance of SubZero is small, and its average performance is superior.

Then we examine the impact of batch size for ZO optimizers using the RoBERTa-large model on SST-2 in full-parameter tuning scheme. The results are shown in Table 14. The training epochs are 100K in Table 6, while they are 20K in Table 14. The remaining hyperparameters are consistent with Table 6, as detailed in Appendix 8.4. For ZO optimizers, a large batch size always gets better performance. Across various batch sizes, SubZero demonstrates better fine-tuning performance compared to MeZO.

Table 13. The impact of random seed with the pretrained OPT-1.3B model on SST-2 in prompt tuning scheme.

Seed	42	0	1234	AVG.
MeZO	85.9	83.3	80.7	83.3
SubZero	**89.1**	**89.4**	**89.2**	**89.2**

Next, we assess the impact of the Adam optimizer. We fine-tune the OPT-1.3B model on the SST-2 dataset, and the experimental results are displayed in Table 15. For ZO optimizers with Adam, we perform a grid search on the hyperparameters

Table 14. The impact of batch size with the pretrained RoBERTa-large model in full-parameter tuning scheme.

Batch Size	Method	**SST-2**	**SST-5**	**SNLI**	**MNLI**	AVG.
16	MeZO	91.7	44.7	77.3	**53.0**	66.7
	SubZero	**91.9**	**45.9**	**77.5**	52.8	**67.0**
32	MeZO	92.9	45.4	78.3	53.2	67.5
	SubZero	**93.0**	**45.5**	**79.6**	**54.0**	**68.0**

and find that keeping the learning rate and perturbation scale consistent with those with SGD resulted in good convergence, as detailed in Table 18. We utilize the linear and the constant learning rate schedules for SGD and Adam, respectively. For all ZO optimizers with SGD, we apply the constant learning rate schedule. For all ZO optimizers with Adam, we test the constant and the cosine annealing schedules. We note that SubZero surpasses MeZO when employing the Adam optimizer with both constant and cosine annealing schedules. Also, Adam does not provide an advantage over SGD for ZO optimization, which aligns with the conclusions of previous studies [22, 65].

Table 15. Comparison of test accuracy (%) for the pretrained OPT-1.3B model fine-tuned on SST-2 with SGD and Adam.

Method	**FT**	**LoRA**	**Prefix**	**Prompt**	AVG.
SGD	**93.2**	93.0	**93.1**	90.7	92.5
Adam	92.6	**93.2**	92.9	**93.3**	**93.0**
MeZO_SGD	92.3	92.8	91.6	85.9	90.7
SubZero_SGD	**93.4**	**92.9**	**92.2**	**89.1**	**91.9**
MeZO_Adam(constant)	92.3	**93.3**	90.7	84.6	90.2
SubZero_Adam(constant)	**93.2**	92.4	**90.9**	**89.3**	**91.5**
MeZO_Adam(cosine)	**91.9**	**93.1**	86.1	78.7	87.5
SubZero_Adam(cosine)	91.7	92.0	**86.6**	**83.4**	**88.4**

Finally, we provide more ablations on the reshaping strategy with OPT-1.3B on Winogrande in the PEFT schemes. The Winogrande dataset [48] is a benchmark for commonsense reasoning and available at https://winogrande.allenai.org/. The results are shown in Table 16. We can set that the reshaping strategy clearly enhances performance, aligning with the conclusion presented in Table 8.

Table 16. Reshaping strategy for non-square matrices with the pretrained OPT-1.3B model fine-tuned on Winogrande in PEFT schemes.

Method	**LoRA**	**Prefix**	**Prompt**	AVG.
SGD	58.3	56.9	58.4	57.9
SubZero(w/o)	56.6	56.6	56.5	56.6
SubZero(w/)	**57.8**	**57.3**	**57.6**	**57.6**

8.2. Implementation Details

We use one A800 GPU with the PyTorch 2.1.0+CUDA 11.8 framework for ZO methods and, if needed, two A800 GPUs for SGD.

The gradient estimation in SubZero is applicable to parameter matrices, while LLMs mainly consist of dense layers. For other trainable parameters, such as biases and layer normalization parameters, we recommend using the gradient estimation in MeZO [40], as these layers contain fewer parameters.

We introduce two useful strategies to implement our SubZero efficiently in memory.

In-place Operation. As indicated in Eqn. (7), directly computing the loss difference ρ requires twice the memory of inference, as it must store both the parameter matrix set $\mathcal{W}$ and the perturbation matrix set $\tilde{\mathcal{Z}}$. To mitigate this, we draw inspiration from MeZO and utilize in-place operations. By employing the random seed trick, we store a random seed to compute ρ (see lines 9-12 in Algorithm 3 and Algorithm 2) and regenerate the low-dimensional perturbation matrices $\boldsymbol{Z}_1, \boldsymbol{Z}_2, \cdots, \boldsymbol{Z}_l$ (see line 15 in Algorithm 3). Consequently, the memory cost for fine-tuning with SubZero is nearly equivalent to that of inference (see Table 2 and Table 10).

Per-layer Weight Update. FO optimizers update all model parameters after BP by storing the entire gradients in memory. In contrast, ZO optimizers like SubZero calculate gradient estimates by first determining the loss value difference from two forward passes, then calculating the gradient estimate for each layer using this difference along with the layer's perturbation. To reduce memory usage during training, we can implement the parameter update with `optimizer.step()` after calculating the gradient estimate for each layer.

SubZero significantly reduces GPU memory consumption with the two implementation strategies. It should note that we use the per-layer weight update strategy for MeZO in all experiments.

To simplify hyperparameter tuning, we employ a norm alignment trick, allowing SubZero to directly utilize hyperparameter settings, such as the learning rate, from MeZO [40]. For a random perturbation matrix $\boldsymbol{Z} \in \mathbb{R}^{m\times n}$, and its low-rank approximation is $\hat{\boldsymbol{Z}} = \boldsymbol{U}\boldsymbol{Z}'\boldsymbol{V}^\top$, where $\boldsymbol{U} \in \mathbb{R}^{m\times r}$, $\boldsymbol{V} \in \mathbb{R}^{n\times r}$, and $\boldsymbol{Z}' \in \mathbb{R}^{r\times r}$. If $\boldsymbol{Z}$ and $\boldsymbol{Z}'$ are Gaussian random matrices, and $\boldsymbol{U}$ and $\boldsymbol{V}$ are column-orthogonal matrices, then we have:

$$\mathbb{E}[\|\boldsymbol{Z}\|_F] = \sqrt{\frac{m\times n}{r^2}}\mathbb{E}\left[\|\hat{\boldsymbol{Z}}\|_F\right]. \tag{20}$$

Define $\mu = \sqrt{\frac{m\times n}{r^2}}$. Let MeZO's learning rate be η and perturbation scale be ε. There are two equivalent approaches to obtain the perturbation for SubZero. The first approach involves multiplying the random low-dimensional perturbation matrix by μ, with SubZero adopting MeZO's hyperparameters directly: $\eta' = \eta$ and $\varepsilon' = \varepsilon$. The second approach keeps the random low-dimensional perturbation matrix fixed and sets SubZero's learning rate and perturbation scale as follows:

$$\eta' = \eta\mu^2, \varepsilon' = \varepsilon\mu.$$

We argue that norm alignment is crucial for SubZero, as changing the rank r affects the norm of the gradient estimate, complicating the fine-tuning of the associated learning rate.

S-MeZO [39], a new ZO method, aims to improve MeZO's performance and convergence speed. However, its source code and detailed layer-wise hyperparameter configurations have not been released. Yang et al. [60] reproduce S-MeZO using a fixed sparsity ratio for each layer, selected based on the best overall result shown in Fig. 6 of their paper. So we perform S-MeZO with this non-official implementation code available at https://github.com/yifanycc/AdaZeta.

8.3. Datasets

Following [40], we use SuperGLUE [57] for OPT experiments, including BoolQ [10], CB [13], COPA [47], MultiRC [29], ReCoRD [63], RTE [4, 5, 11, 21], WiC [44], and WSC [34]. We also utilize SST-2 [50] and two question answering (QA) datasets, SQuAD [45] and DROP [18]. For each task, we randomly sampled 1000 examples for training, 500 for validation, and 1000 for testing.

For LLama2-7B and Mistral-7B, we use CB [13] in the full-parameter tuning and three PEFT schemes. For OPT-1.3B, we utilize SST-2 [50] in the full-parameter tuning and three PEFT schemes.

For RoBERTa-large, we consider classification datasets: SST-2 [50], SST-5 [50], MNLI [59], and SNLI [6]. Following [40], the test set has 1000 examples for fast iteration, while we have 512 examples per class for both training and validation.

8.4. Hyperparameters

Using a larger batch size can consistently reduce the variance in ZO optimization, thus enhancing fine-tuning performance [20, 40, 60]. However, this increase in batch size also raises the time for forward passes and significantly elevates memory usage. We focus on developing ZO methods that minimize variance and improve performance with small batch sizes, with a default

Table 17. The hyperparameter search grids for OPT-13B. For each task, we run 20K steps for ZO methods (MeZO, S-MeZO, and SubZero) and SGD. We record the best model checkpoint based on the validation loss every 500 training steps.

Experiment	Hyperparameter	Value
MeZO(FT)	batch size	16
	learning rate	{1e-7, 2e-7, 5e-7, 1e-6}
	ε	1e-3
MeZO(LoRA)	batch size	16
	learning rate	{1.5e-5, 3e-5, 5e-5}
	ε	1e-3
S-MeZO(FT)	batch size	16
	learning rate	{1e-6, 5e-6}
	ε	1e-3
	sparse rate	0.75
S-MeZO(LoRA)	batch size	16
	learning rate	{5e-5, 1e-4, 1e-3}
	ε	1e-3
	Sparse rate	0.75
SubZero(FT)	batch size	16
	learning rate	{1e-7, 2e-7, 5e-7, 1e-6}
	ε	1e-3
	rank	{32, 64, 128, 256 }
	subspace change frequency	{500, 1000, 2000}
SubZero(LoRA)	batch size	16
	learning rate	{1.5e-5, 3e-5, 5e-5}
	ε	1e-3
	rank	{4, 8, 16}
	subspace change frequency	{500, 1000, 2000}
SGD(FT)	batch size	16
	learning rate	{1e-4, 1e-3, 5e-3}

setting of 16. In some SGD experiments, like on MultiRC and SQuAD, the batch size is reduced to 8 due to limited GPU resources.

Consistent with previous studies [39, 40, 60, 65], we employ SGD without momentum by default to maintain memory efficiency. SGD utilizes the linear learning rate schedule, while all ZO methods with SGD apply a constant learning rate schedule, with weight decay set to 0.

For the projection matrix generation experiments summarized in Table 1, we perform full-parameter fine-tuning on the SST-2 dataset using three models: the OPT-1.3B model with a rank of 24 and a subspace update frequency of 1000, the LLaMA2-7B model with a rank of 24 and a subspace update frequency of 1000 and the OPT-13B model with a rank of 128 and a subspace update frequency of 1000. All other hyperparameters, including learning rate and perturbation scale, remain consistent across experiments. While the decomposition of the weight and Gaussian random matrices is relatively straightforward, the activation matrix relies on the AdaBK algorithm [61]. Specifically, we compute the second-order statistics of activations (see line 7 of Algorithm 1 in the paper of AdaBK) and employ an adaptive damping strategy based on the maximum singular value to improve the condition number, followed by QR decomposition. For dimensionality alignment, the second-order statistics of output and input features are decomposed to derive $\mathbf{U}$ and $\mathbf{V}$, respectively. Finally, for the decomposition of the historical zeroth-order gradient, its matrix from the previous iteration is readily obtained and subsequently QR-decomposed, facilitated by our random seed technique and the saved historical projection matrix.

For RoBERTa, we run Adam for 1K steps and ZO methods for 100K steps. In the rest experiments, we run Adam for 5 epochs and SGD and ZO methods for 20K steps.

We follow previous work to set the hyperparameters in PEFT schemes [40, 65]. For LoRA, the rank is set to 8 and α is set to 16. For prefix tuning, the length of prefix tokens is set to 5, and we initialize these tunable representations by randomly sampling tokens from the vocabulary and then passing them through the LLM to get their keys and values at different attention

layers. For prompt tuning, the length of prompt virtual tokens is set to 10, and the prompt tokens are initialized with actual token values from the model's embedding.

We present the hyperparameter search grids in Tables 17 and 18 to assist with result reproduction. For OPT-1.3B, we utilize the same hyperparameter settings as in Table 18. For RoBERTa-large, we use a learning rate of {1e-6, 5e-6} and ε=1e-3 for MeZO and SubZero, with a batch size of 64. The rank for SubZero is set to {8, 16, 24}, and subspace change frequency is adjusted to {1000, 2000}.

For the ablation study, we evaluate the effectiveness of the orthogonal projection matrix using the OPT-13B model in full-parameter tuning scheme on the RTE and WSC datasets, and the results are presented in Table 5. The hyperparameter settings are consistent with those in Table 3, and further details are available in Table 17. The subspace dimensionality remains fixed across all experiments. It is noteworthy that both orthogonal and non-orthogonal projection matrices can utilize the same learning rate and perturbation scale. This is because the overall perturbation matrix is scaled by a factor of $\frac{1}{r}$, following a similar norm alignment strategy as detailed in Eqn. (20). We also perform ablation studies on the rank and subspace update frequency for SubZero, with results shown in Table 7. Full-parameter tuning scheme is conducted on the RTE dataset using the OPT-13B model, with specific experimental settings outlined in Table 17. All experiments employ the same learning rate and perturbation scale, enabled by the norm alignment technique described in Eqn. (20).

8.5. Prompt Templates

For autoregressive LLMs, we have three task types: classification, multiple-choice, and question answering. We adopt the prompt templates for various tasks in [40], which are summarized in Table 19. For masked LLMs, we also adopt the prompt templates in [40] and present them in Table 20.

8.6. Proofs

In practice, SubZero employs smaller and layer-specific low-rank perturbation matrices instead of a large model-scale projection matrix. However, it is more convenient to prove SubZero's properties using a model-scale projection. Fortunately, the following lemma shows that the low-rank perturbation matrix for each layer can be represented as a layer-scale projection matrix, which is column orthogonal.

Lemma 1. *Let* $\tilde{\boldsymbol{Z}} = \boldsymbol{U}\boldsymbol{Z}\boldsymbol{V}^\top$, *where* $\boldsymbol{U} \in \mathbb{R}^{m\times r}, \boldsymbol{Z} \in \mathbb{R}^{r\times r}, \boldsymbol{V} \in \mathbb{R}^{n\times r}$, *and* $\boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{I}_r$. *Then we have* $\mathrm{vec}(\tilde{\boldsymbol{Z}}) = \boldsymbol{P}\mathrm{vec}(\boldsymbol{Z})$ *and* $\boldsymbol{P}^\top\boldsymbol{P} = \boldsymbol{I}_{r^2}$, *where* $\boldsymbol{P} = \boldsymbol{V} \otimes \boldsymbol{U}$.

Proof. Since $\mathrm{vec}(\boldsymbol{U}\boldsymbol{Z}\boldsymbol{V}^\top) = (\boldsymbol{V} \otimes \boldsymbol{U})\mathrm{vec}(\boldsymbol{Z})$, we only need to show $(\boldsymbol{V} \otimes \boldsymbol{U})^\top(\boldsymbol{V} \otimes \boldsymbol{U}) = \boldsymbol{I}_{r^2}$. In fact

$$(\boldsymbol{V} \otimes \boldsymbol{U})^\top(\boldsymbol{V} \otimes \boldsymbol{U}) = (\boldsymbol{V}^\top \otimes \boldsymbol{U}^\top)(\boldsymbol{V} \otimes \boldsymbol{U}) = (\boldsymbol{V}^\top\boldsymbol{V}) \otimes (\boldsymbol{U}^\top\boldsymbol{U}) = \boldsymbol{I}_r \otimes \boldsymbol{I}_r = \boldsymbol{I}_{r^2}.$$

The proof is completed. □

We can also demonstrate that the low-rank perturbation matrices across all layers can be represented as a model-scale projection matrix. We first give the following lemma.

Lemma 2. *Let a block diagonal matrix* $\boldsymbol{P} = \mathrm{bdiag}(\boldsymbol{P}_1, \boldsymbol{P}_2, \cdots, \boldsymbol{P}_l)$ *and* $\tilde{\boldsymbol{z}}_i = \boldsymbol{P}_i\boldsymbol{z}_i$, *where* $\boldsymbol{P}_i^\top\boldsymbol{P}_i = \boldsymbol{I}_{r^2}$ *and* $i = 1, 2, \ldots, l$. *Then we have* $\tilde{\boldsymbol{z}} = \boldsymbol{P}\boldsymbol{z}$, *where* $\tilde{\boldsymbol{z}} = [\tilde{\boldsymbol{z}}_1^\top, \ldots, \tilde{\boldsymbol{z}}_l^\top]^\top$, $\boldsymbol{z} = [\boldsymbol{z}_1^\top, \ldots, \boldsymbol{z}_l^\top]^\top$ *and* $\boldsymbol{P}^\top\boldsymbol{P} = \boldsymbol{I}_{lr^2}$.

Proof. It is easy to check that $\tilde{\boldsymbol{z}} = \boldsymbol{P}\boldsymbol{z}$. Besides, we have

$$\boldsymbol{P}^\top\boldsymbol{P} = \mathrm{bdiag}(\boldsymbol{P}_1^\top, \ldots, \boldsymbol{P}_l^\top)\mathrm{bdiag}(\boldsymbol{P}_1, \ldots, \boldsymbol{P}_l) = \mathrm{bdiag}(\boldsymbol{P}_1^\top\boldsymbol{P}_1, \ldots, \boldsymbol{P}_l^\top\boldsymbol{P}_l) = \boldsymbol{I}_{lr^2}.$$

The proof is completed. □

We may define $\boldsymbol{P} = \mathrm{bdiag}(\boldsymbol{V}_1 \otimes \boldsymbol{U}_1, \boldsymbol{V}_2 \otimes \boldsymbol{U}_2, \cdots, \boldsymbol{V}_l \otimes \boldsymbol{U}_l)$ that satisfies $\boldsymbol{P}^\top\boldsymbol{P} = \boldsymbol{I}$, $\boldsymbol{z} = [\mathrm{vec}(\boldsymbol{Z}_1)^\top, \mathrm{vec}(\boldsymbol{Z}_2)^\top, \ldots, \mathrm{vec}(\boldsymbol{Z}_l)^\top]^\top$, and $\tilde{\boldsymbol{z}} = [\mathrm{vec}(\tilde{\boldsymbol{Z}}_1)^\top, \mathrm{vec}(\tilde{\boldsymbol{Z}}_2)^\top, \ldots, \mathrm{vec}(\tilde{\boldsymbol{Z}}_l)^\top]^\top$. Then according to Lemma 2, the perturbation vector of SubZero is $\tilde{\boldsymbol{z}} = \boldsymbol{P}\boldsymbol{z}$, which is similar as existing random subspace methods in Eqn. (4), but with SubZero's projection matrix being block diagonal and column orthogonal.

To prove Theorem 1 and Theorem 2, we first introduce some definitions and lemmas about Gaussian distribution.

Table 18. The hyperparameter search grids for LLama2-7B and Mistral-7B. For each task, we run 20K steps for ZO methods (MeZO and SubZero) and SGD. We record the best model checkpoint based on the validation loss every 500 training steps.

Experiment	Hyperparameter	Value
MeZO(FT)	batch size	16
	learning rate	{1e-7, 5e-7, 1e-6}
	ε	1e-3
MeZO(LoRA)	batch size	16
	learning rate	{1e-6, 5e-6, 1e-5, 3e-5}
	ε	1e-3
MeZO(Prefix)	batch size	16
	learning rate	{1e-3, 5e-3, 1e-2}
	ε	1e-1
MeZO(Prompt)	batch size	16
	learning rate	{1e-3, 5e-3, 1e-2}
	ε	1e-2
SubZero(FT)	batch size	16
	learning rate	{1e-7, 5e-7, 1e-6}
	ε	1e-3
	rank	{24, 48}
	subspace change frequency	1000
SubZero(LoRA)	batch size	16
	learning rate	{1e-6, 5e-6, 1e-5, 3e-5}
	ε	1e-3
	rank	{4, 8}
	subspace change frequency	1000
SubZero(Prefix)	batch size	16
	learning rate	{1e-3, 5e-3, 1e-2}
	ε	1e-1
	rank	{4, 8}
	subspace change frequency	1000
SubZero(Prompt)	batch size	16
	learning rate	{1e-3, 5e-3, 1e-2}
	ε	1e-2
	rank	{16, 24}
	subspace change frequency	1000
SGD(FT)	batch size	16
	learning rate	{1e-5, 1e-4, 1e-3, 5e-3}

Defination 1. *We say $\boldsymbol{z}$ is a standard n-dimensional Gaussian vector (denote by $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n)$), if its probability density function $p(\boldsymbol{z}) = \frac{1}{\kappa} e^{-\frac{1}{2}\|\boldsymbol{z}\|^2}$, where $\kappa > 0$ satisfies $\int_{\mathbb{R}^n} \frac{1}{\kappa} e^{-\frac{1}{2}\|\boldsymbol{z}\|^2} d\boldsymbol{z} = 1$.*

Defination 2. *Let $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n)$. We say x is a chi-square random variable with degrees of freedom n (denote by $x \sim \chi^2(n)$), if $x = \|\boldsymbol{z}\|^2$.*

Lemma 3. *Let $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n)$. For any orthogonal $(n \times n)$-matrix $\boldsymbol{Q}$ and continuous function f, we have $\mathbb{E}_{\boldsymbol{z}}[f(\boldsymbol{z})] = \mathbb{E}_{\boldsymbol{z}}[f(\boldsymbol{Q}\boldsymbol{z})]$.*

Lemma 4. *If $x \sim \chi^2(n)$, then we have*

$$\mathbb{E}_x[x] = n, \quad \mathrm{Var}_x[x] = 2n.$$

Lemma 5. *[41] Let $f \in C_{L_2}^{2,2}(\mathbb{R}^n)$. Then for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, we have*

$$|f(\boldsymbol{y}) - f(\boldsymbol{x}) - \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle - \frac{1}{2} \langle \nabla^2 f(\boldsymbol{x})(\boldsymbol{y} - \boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle| \leq \frac{L_2}{6} \|\boldsymbol{y} - \boldsymbol{x}\|^3.$$

Table 19. The prompt templates used in the OPT-1.3B, OPT-13B, LLama2-7B, and Mistral-7B experiments.

Task	Type	Prompt
SST-2	cls.	<text> It was terrible/great
RTE	cls.	<premise> Does this mean that "<hypothesis>" is true? Yes or No? Yes or No
CB	cls.	Does this mean that "<hypothesis>" is true? Yes or No? Yes/No/Maybe
BoolQ	cls.	<passage> <question>? Yes/No
WSC	cls.	<text> In the previous sentence, does the pronoun "<span2>" refer to <span1>? Yes or No? Yes/No
WIC	cls.	Does the word "<word>" have the same meaning in these two sentences? Yes, No? <sentence1> <sentence2> Yes/No
MultiRC	cls.	<paragraph> Question: <question> I found this answer "<answer". Is that correct? Yes or No? Yes/No
COPA	mch.	<premise> so/because <candidate>
ReCoRD	mch.	<passage> <query>.replace("@placeholder", <candidate>)
SQuAD	QA	Title: <title> Context: <context> Question: <question> Answer:
DROP	QA	Passage: <context> Question: <question> Answer:

Table 20. The prompt templates used in RoBERTa-large experiments. C is the number of classification categories.

Task	C	Type	Prompt
SST-2	2	sentiment cls.	<sentence1> It was great/terrible
SST-5	5	sentiment cls.	<sentence1> It was great/good/okay/bad/terrible
MNLI	3	NLI	<sentence1> ? Yes/Maybe/No , <sentence2>
SNLI	3	NLI	<sentence1> ? Yes/Maybe/No , <sentence2>

Lemma 6. *[41] Let $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n)$. For $0 \leq t \leq 2$, we have*

$$\mathbb{E}_{\boldsymbol{z}}[\|\boldsymbol{z}\|^t] \leq n^{t/2}.$$

For $t \geq 2$, we have

$$n^{t/2} \leq \mathbb{E}_{\boldsymbol{z}}[\|\boldsymbol{z}\|^t] \leq (n+t)^{t/2}.$$

Lemma 7. *Let $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n)$. For all $\boldsymbol{y} \in \mathbb{R}^n$, we have*

$$\mathbb{E}_{\boldsymbol{z}}[\|\langle \boldsymbol{y}, \boldsymbol{z}\rangle \boldsymbol{z}\|^2] = (n+2)\|\boldsymbol{y}\|^2.$$

Proof. Note that for any orthogonal $(n \times n)$-matrix $\boldsymbol{Q}$, we have

$$\|\langle \boldsymbol{y}, \boldsymbol{Q}\boldsymbol{z}\rangle \boldsymbol{Q}\boldsymbol{z}\|^2 = \|\langle \boldsymbol{Q}^\top \boldsymbol{y}, \boldsymbol{z}\rangle \boldsymbol{z}\|^2, \quad \|\boldsymbol{Q}^\top \boldsymbol{y}\| = \|\boldsymbol{y}\|.$$

In accordance with Lemma 3, we can set $\boldsymbol{y} = [1, 0, \ldots, 0]^{\mathsf{T}}$, and only need to prove $\mathbb{E}_{\boldsymbol{z}}[\|\langle \boldsymbol{y}, \boldsymbol{z}\rangle \boldsymbol{z}\|^2] = n + 2$. Equipped with Lemma 4, we get

$$\mathbb{E}_{\boldsymbol{z}}[\|\langle \boldsymbol{y}, \boldsymbol{z}\rangle \boldsymbol{z}\|^2] = \mathbb{E}_{\boldsymbol{z}}\left[\sum_{i=1}^{n} \boldsymbol{z}_1^2 \boldsymbol{z}_i^2\right] = \sum_{i=1}^{n} \mathbb{E}_{\boldsymbol{z}}[\boldsymbol{z}_1^2 \boldsymbol{z}_i^2] = \mathbb{E}_{\boldsymbol{z}_1}[\boldsymbol{z}_1^4] + \mathbb{E}_{\boldsymbol{z}_1}[\boldsymbol{z}_1^2] \sum_{i=2}^{n} \mathbb{E}_{\boldsymbol{z}}[\boldsymbol{z}_i^2] = n + 2.$$

The proof is completed. □

Theorem 1. *For the gradient estimation in Eqn. (8), the following two properties hold.*
a) *By using gradient estimation in (8), our estimated gradient $\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})$ is equivalent to*

$$\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z}) = \frac{f(\boldsymbol{x} + \varepsilon \boldsymbol{P}\boldsymbol{z}) - f(\boldsymbol{x} - \varepsilon \boldsymbol{P}\boldsymbol{z})}{2\varepsilon} \boldsymbol{P}\boldsymbol{z}, \tag{13}$$

where $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_q)$, $\varepsilon > 0$, $\boldsymbol{P} \in \mathbb{R}^{d \times q}$ satisfies $\boldsymbol{P}^{\mathsf{T}}\boldsymbol{P} = \boldsymbol{I}_q$ with $d = \sum_{i=1}^{l} m_i n_i$ and $q = lr^2$.
b) *Let $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_q)$, and $f \in C_{L_2}^{2,2}(\mathbb{R}^d)$. Then we have*

$$\Phi(\boldsymbol{x}) = \|\mathbb{E}_{\boldsymbol{z}}[\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})] - \boldsymbol{P}\boldsymbol{P}^{\mathsf{T}} \nabla f(\boldsymbol{x})\|_2 \leq \frac{\varepsilon^2}{6} L_2 (q + 4)^2.$$

Proof. **a)** Evidently, the conclusion is established based on Lemma 1 and Lemma 2.

b) Let $a_{\boldsymbol{z}}(\tau) = f(\boldsymbol{x} + \tau \boldsymbol{z}) - f(\boldsymbol{x}) - \tau \langle \nabla f(\boldsymbol{x}), \boldsymbol{z}\rangle - \frac{\tau^2}{2} \langle \nabla^2 f(\boldsymbol{x}) \boldsymbol{z}, \boldsymbol{z}\rangle$. Lemma 5 implies that

$$|a_{\boldsymbol{z}}(\pm\varepsilon)| \leq \frac{\varepsilon^3}{6} L_2 \|\boldsymbol{z}\|^3.$$

Note that

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{z}}[\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})] - \boldsymbol{P}\boldsymbol{P}^{\mathsf{T}} \nabla f(\boldsymbol{x}) \\
&= \frac{\boldsymbol{P}}{2\kappa\varepsilon} \int_{\mathbb{R}^q} [f(\boldsymbol{x} + \varepsilon \boldsymbol{P}\boldsymbol{z}) - f(\boldsymbol{x} - \varepsilon \boldsymbol{P}\boldsymbol{z}) - 2\varepsilon \langle \nabla f(\boldsymbol{z}), \boldsymbol{P}\boldsymbol{z}\rangle] \boldsymbol{z} e^{-\frac{1}{2}\|\boldsymbol{z}\|^2} d\boldsymbol{z}.
\end{aligned}$$

Therefore, in accordance with Lemma 6, we have

$$\begin{aligned}
&\|\mathbb{E}_{\boldsymbol{z}}[\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})] - \boldsymbol{P}\boldsymbol{P}^{\mathsf{T}} \nabla f(\boldsymbol{x})\| \\
&\leq \frac{1}{2\kappa\varepsilon} \int_{\mathbb{R}^q} |f(\boldsymbol{x} + \varepsilon \boldsymbol{P}\boldsymbol{z}) - f(\boldsymbol{x} - \varepsilon \boldsymbol{P}\boldsymbol{z}) - 2\varepsilon \langle \nabla f(\boldsymbol{z}), \boldsymbol{P}\boldsymbol{z}\rangle| \|\boldsymbol{z}\| e^{-\frac{1}{2}\|\boldsymbol{z}\|^2} d\boldsymbol{z} \\
&= \frac{1}{2\kappa\varepsilon} \int_{\mathbb{R}^q} |a_{\boldsymbol{P}\boldsymbol{z}}(\varepsilon) - a_{\boldsymbol{P}\boldsymbol{z}}(-\varepsilon)| \|\boldsymbol{z}\| e^{-\frac{1}{2}\|\boldsymbol{z}\|^2} d\boldsymbol{z} \\
&\leq \frac{\varepsilon^2 L_2}{6\kappa} \int_{\mathbb{R}^q} \|\boldsymbol{z}\|^4 e^{-\frac{1}{2}\|\boldsymbol{z}\|^2} d\boldsymbol{z} \leq \frac{\varepsilon^2}{6} L_2 (q + 4)^2.
\end{aligned}$$

The proof is completed. □

Theorem 2. *Let $f(\boldsymbol{x}) = \boldsymbol{x}^{\mathsf{T}} \boldsymbol{H} \boldsymbol{x}$ and $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_q)$, where $\boldsymbol{H} \in \mathbb{R}^{d \times d}$ is positive definite. We have*

$$\mathbb{E}_{\boldsymbol{z}}[\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})] = \boldsymbol{P}\boldsymbol{P}^{\mathsf{T}} \nabla f(\boldsymbol{x}), \tag{14}$$

$$\mathbb{E}_{\boldsymbol{z}}[\|\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})\|^2] = (q + 2)\|\boldsymbol{P}^{\mathsf{T}} \nabla f(\boldsymbol{x})\|^2, \tag{15}$$

$$\mathbb{E}_{\boldsymbol{z}}\left[\frac{\langle \nabla f(\boldsymbol{x}), \hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})\rangle^2}{\|\boldsymbol{P}^{\mathsf{T}} \nabla f(\boldsymbol{x})\|^2 \|\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})\|^2}\right] = \frac{1}{q}. \tag{16}$$

Proof. It is easy to check that $\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z}) = \boldsymbol{P}\langle \boldsymbol{P}^{\mathsf{T}} \nabla f(\boldsymbol{x}), \boldsymbol{z}\rangle \boldsymbol{z}$. Thus we have $\mathbb{E}_{\boldsymbol{z}}[\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})] = \boldsymbol{P}\boldsymbol{P}^{\mathsf{T}} \nabla f(\boldsymbol{x})$. Combined with Lemma 7, we get $\mathbb{E}_{\boldsymbol{z}}[\|\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})\|^2] = (q + 2)\|\boldsymbol{P}^{\mathsf{T}} \nabla f(\boldsymbol{x})\|^2$. Note that for any orthogonal $(q \times q)$-matrix $\boldsymbol{Q}$, we have

$$\mathbb{E}_{\boldsymbol{z}}\left[\frac{\langle\nabla f(\boldsymbol{x}),\hat{g}_\varepsilon(\boldsymbol{x},\boldsymbol{P},\boldsymbol{z})\rangle^2}{\|\boldsymbol{P}^\mathsf{T}\nabla f(\boldsymbol{x})\|^2\|\hat{g}_\varepsilon(\boldsymbol{x},\boldsymbol{P},\boldsymbol{z})\|^2}\right]=\mathbb{E}_{\boldsymbol{z}}\left[\frac{\langle\boldsymbol{P}^\mathsf{T}\nabla f(\boldsymbol{x}),\boldsymbol{z}\rangle^2}{\|\boldsymbol{P}^\mathsf{T}\nabla f(\boldsymbol{x})\|^2\|\boldsymbol{z}\|^2}\right]$$
$$=\mathbb{E}_{\boldsymbol{z}}\left[\frac{\langle\boldsymbol{P}^\mathsf{T}\nabla f(\boldsymbol{x}),\boldsymbol{Q}\boldsymbol{z}\rangle^2}{\|\boldsymbol{P}^\mathsf{T}\nabla f(\boldsymbol{x})\|^2\|\boldsymbol{Q}\boldsymbol{z}\|^2}\right]$$
$$=\mathbb{E}_{\boldsymbol{z}}\left[\frac{\langle\boldsymbol{Q}^\mathsf{T}\boldsymbol{P}^\mathsf{T}\nabla f(\boldsymbol{x}),\boldsymbol{z}\rangle^2}{\|\boldsymbol{Q}^\mathsf{T}\boldsymbol{P}^\mathsf{T}\nabla f(\boldsymbol{x})\|^2\|\boldsymbol{z}\|^2}\right].$$

In accordance with Lemma 3, we can set $\boldsymbol{P}^\mathsf{T}\nabla f(\boldsymbol{x})=[1,0,\ldots,0]^\mathsf{T}$. Thus we have

$$\mathbb{E}_{\boldsymbol{z}}\left[\frac{\langle\nabla f(\boldsymbol{x}),\hat{g}_\varepsilon(\boldsymbol{x},\boldsymbol{P},\boldsymbol{z})\rangle^2}{\|\boldsymbol{P}^\mathsf{T}\nabla f(\boldsymbol{x})\|^2\|\hat{g}_\varepsilon(\boldsymbol{x},\boldsymbol{P},\boldsymbol{z})\|^2}\right]=\mathbb{E}_{\boldsymbol{z}}\left[\frac{\boldsymbol{z}_1^2}{\|\boldsymbol{z}\|^2}\right]=\frac{1}{q}.$$

The proof is completed. □

To illustrate the convergence of Subzero with SGD, our analysis is divided into two main segments. We first investigate the convergence behavior of SubZero solution process while keeping the projection matrix $\boldsymbol{P}$ constant. Next, we evaluate the effects of the lazy updates to $\boldsymbol{P}$. Based on these evaluations, we establish the global convergence of Subzero. Without loss of generality, we concentrate on the scenario where the number of layers is 1.

First, when the subspace $\boldsymbol{P}$ is fixed, the original problem of SubZero can be reformulated as an optimization problem within the subspace. Define $h(\boldsymbol{y})=f(\boldsymbol{x}+\boldsymbol{P}\boldsymbol{y})$, $h_\varepsilon(\boldsymbol{y})=\mathbb{E}_{\boldsymbol{z}}[h(\boldsymbol{y}+\varepsilon\boldsymbol{z})]$, and $g_\varepsilon(\boldsymbol{y})=\frac{h(\boldsymbol{y}+\varepsilon\boldsymbol{z})-f(\boldsymbol{y})}{\varepsilon}\boldsymbol{z}$. According to Lemma 8, if f is first L_1-smooth, then h is also first L_1-smooth.

Lemma 8. *Let $h(\boldsymbol{y})=f(\boldsymbol{x}+\boldsymbol{P}\boldsymbol{y})$, where $f\in C_{L_1}^{1,1}(\mathbb{R}^d)$, and $\boldsymbol{P}^\mathsf{T}\boldsymbol{P}=\boldsymbol{I}$, then we have $h\in C_{L_1}^{1,1}(\mathbb{R}^q)$.*

Proof. The following proves that if f is first L_1-smooth, then h is also first L_1-smooth. For any $\boldsymbol{y}_1\in\mathbb{R}^q$ and $\boldsymbol{y}_2\in\mathbb{R}^q$, we have

$$\begin{aligned}\|\nabla h(\boldsymbol{y}_1)-\nabla h(\boldsymbol{y}_2)\|&=\left\|\boldsymbol{P}^\mathsf{T}\nabla(f(\boldsymbol{x}+\boldsymbol{P}\boldsymbol{y}_1)-\boldsymbol{P}^\mathsf{T}\nabla(f(\boldsymbol{x}+\boldsymbol{P}\boldsymbol{y}_2)\right\|\\&\le\left\|\boldsymbol{P}^\mathsf{T}\right\|\|\nabla(f(\boldsymbol{x}+\boldsymbol{P}\boldsymbol{y}_1)-\nabla(f(\boldsymbol{x}+\boldsymbol{P}\boldsymbol{y}_2)\|\\&\le L_1\|\boldsymbol{P}(\boldsymbol{y}_1-\boldsymbol{y}_2)\|\\&=L_1\|\boldsymbol{y}_1-\boldsymbol{y}_2\|.\end{aligned}$$

The proof is completed. □

Now, we can analyze the convergence of SubZero when fixing the subspace.

Lemma 9. *[41] Let $f\in C_{L_1}^{1,1}(\mathbb{R})$. Then, for any $\boldsymbol{x}\in\mathbb{R}$, we have*

$$E_{\boldsymbol{z}}[\|g_\varepsilon(\boldsymbol{x})\|^2]=E_{\boldsymbol{z}}\left[\|\frac{f(\boldsymbol{x}+\varepsilon\boldsymbol{z})-f(\boldsymbol{x})}{\varepsilon}\|^2\right]\le 4(n+4)\|\nabla f_\varepsilon(\boldsymbol{x})\|^2+3\varepsilon^2L_1^2(f)(n+4)^3,\tag{21}$$

and

$$\|\nabla f(\boldsymbol{x})\|^2\le 2\|\nabla f_\varepsilon(\boldsymbol{x})\|^2+\frac{\varepsilon^2}{2}L_1^2(f)(n+6)^3,\tag{22}$$

where $f_\varepsilon(\boldsymbol{x})=\mathbb{E}_{\boldsymbol{z}}[f(\boldsymbol{x}+\varepsilon\boldsymbol{z})]$.

Lemma 10. *Let $\boldsymbol{y}^*=\arg\min_{\boldsymbol{x}\in\mathbb{R}^q}h(\boldsymbol{y})$, where $h\in C_{L_1}^{1,1}(\mathbb{R}^q)$ and h is non-convex. Suppose $\mathcal{E}_k=(\boldsymbol{z}_0,\boldsymbol{z}_1,\cdots,\boldsymbol{z}_{k-1},\boldsymbol{z}_k)$, where $\boldsymbol{z}_k\sim\mathcal{N}(0,\boldsymbol{I}_q)$ and $\eta=\frac{1}{4(q+4)L_1}$. $\{\boldsymbol{y}_k\}_{k>0}$ is the sequence generated by Algorithm 3. Let $\phi_0=h(\boldsymbol{y}_0)$, and for $k\ge 1$, $\phi_k=\mathbb{E}_{\mathcal{E}_{k-1}}[h(\boldsymbol{y}_k)]$. For the $\boldsymbol{P}$ defined in (10), which is fixed, we have*

$$\phi_{k+1}-\phi_k\le-\frac{1}{4}\eta\mathbb{E}_{\mathcal{E}_k}\left[\|\nabla h(\boldsymbol{y}_k)\|^2\right]+\frac{\varepsilon^2(q+6)^3}{8}L_1^2+\frac{3\varepsilon^2(q+4)}{32}L_1\tag{23}$$

Proof. If a subspace $\boldsymbol{P} \in \mathbb{R}^{d\times q}$ is fixed, the optimization objective can be reformulated as

$$\min_{\boldsymbol{y}\in\mathbb{R}^q} h(\boldsymbol{y}) := f(\boldsymbol{x} + \boldsymbol{P}\boldsymbol{y}),$$

Let $\boldsymbol{y}_0$ be an initial point and $\{\eta_k\}_{k\geq 0}$ a sequence of positive real numbers. Consider the randomized gradient search algorithm $\mathcal{RG}_\varepsilon(\varepsilon > 0)$:

1) Generate $\boldsymbol{z}_k$ and the corresponding $g_\varepsilon(\boldsymbol{y}_k)$, where $\boldsymbol{z}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_q)$.

2) Update $\boldsymbol{y}_{k+1} = \boldsymbol{y}_k - \eta_k g_\varepsilon(\boldsymbol{y}_k)$.

We aim to estimate the evolution of the function h_ε after one iteration of this algorithm.

Given that h is L_1-Lipschitz continuous for the first derivative, and h_ε is L_ε-Lipschitz continuous for the first derivative (where $L_\varepsilon \leq L_1$)[41]. Thus, we have

$$h_\varepsilon(\boldsymbol{y}_{k+1}) \leq h_\varepsilon(\boldsymbol{y}_k) - \eta_k\langle\nabla h_\varepsilon(\boldsymbol{y}_k), g_\varepsilon(\boldsymbol{y}_k)\rangle + \frac{1}{2}\eta_k^2 L_\varepsilon\|g_\varepsilon(\boldsymbol{y}_k)\|^2.$$

Taking expectation with respect to $\boldsymbol{z}_k$, we obtain

$$\mathbb{E}_{\boldsymbol{z}_k}[h_\varepsilon(\boldsymbol{y}_{k+1})] \leq h_\varepsilon(\boldsymbol{y}_k) - \eta_k\|\nabla h_\varepsilon(\boldsymbol{y}_k)\|^2 + \frac{1}{2}\eta_k^2 L_\varepsilon\,\mathbb{E}_{\boldsymbol{z}_k}[\|g_\varepsilon(\boldsymbol{y}_k)\|^2].$$

Since $h \in C^{1,1}(\mathbb{R}^q)$, from Lemma 9, we have

$$\begin{aligned}\mathbb{E}_{\boldsymbol{z}_k}[h_\varepsilon(\boldsymbol{y}_{k+1})] \leq\ & h_\varepsilon(\boldsymbol{y}_k) - \eta_k\|\nabla h_\varepsilon(\boldsymbol{y}_k)\|^2 \\ & + \frac{1}{2}\eta_k^2 L_1\left(4(q+4)\|\nabla h_\varepsilon(\boldsymbol{y}_k)\|^2 + 3\varepsilon^2 L_1^2(q+4)^3\right).\end{aligned}$$

Setting $\eta_k = \hat{\eta} = \frac{1}{4(q+4)L_1}$, we get

$$\mathbb{E}_{\boldsymbol{z}_k}[h_\varepsilon(\boldsymbol{y}_{k+1})] \leq h_\varepsilon(\boldsymbol{y}_k) - \frac{1}{2}\hat{\eta}\|\nabla h_\varepsilon(\boldsymbol{y}_k)\|^2 + \frac{3\varepsilon^2}{32}L_1(q+4).$$

Taking the expectation with respect to $\mathcal{E}_k$, we get

$$\phi_{k+1} \leq \phi_k - \frac{1}{2}\hat{\eta}\mathbb{E}_{\mathcal{E}_k}[\|\nabla h_\varepsilon(\boldsymbol{y}_k)\|^2] + \frac{3\varepsilon^2(q+4)}{32}L_1,$$

From Lemma 9, we have $\mathbb{E}_{\mathcal{E}_k}[\|\nabla h(\boldsymbol{y}_k)\|^2] \leq 2\mathbb{E}_{\mathcal{E}_k}[\|\nabla h_\varepsilon(\boldsymbol{y}_k)\|^2] + \frac{\varepsilon^2(q+6)^3}{2}L_1^2$. Therefore,

$$\phi_{k+1} - \phi_k \leq -\frac{1}{4}\hat{\eta}\mathbb{E}_{\mathcal{E}_k}\left[\|\nabla h(\boldsymbol{y}_k)\|^2\right] + \frac{\varepsilon^2(q+6)^3}{8}L_1^2 + \frac{3\varepsilon^2(q+4)}{32}L_1. \tag{24}$$

The proof is completed. □

Next, we need to measure the randomness of our random subspace. From Lemma 14, if the projection matrix is obtained by Algorithm 1, we have $\mathbb{E}[\boldsymbol{P}\boldsymbol{P}^T] = \frac{q}{d}\boldsymbol{I}$, where q represents the dimension of the subspace, d represents the dimension of the origin space, and $\boldsymbol{P} = \boldsymbol{V} \otimes \boldsymbol{U}$ (see Lemma 1).

Lemma 11. *Let matrix $\boldsymbol{A} = (\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_r) \in \mathbb{R}^{n\times r}$ be composed of column vectors $\boldsymbol{a}_k$ which are mutually independent and $\boldsymbol{a}_k \in \mathcal{N}(0, \boldsymbol{I}_n)$. Suppose Gram-Schmidt process $\boldsymbol{u}_k = \boldsymbol{a}_k - \sum_{s=1}^{k-1}\langle\boldsymbol{a}_k, \boldsymbol{e}_s\rangle\,\boldsymbol{e}_s$ and $\boldsymbol{e}_k = \frac{\boldsymbol{u}_k}{\|\boldsymbol{u}_k\|}$. $[\boldsymbol{a}_k]_i \leftrightarrow [\boldsymbol{a}_k]_j$ represents the exchange of the i-th element and the j-th element of $\boldsymbol{a}_k$, while all other elements remain unchanged. $[\boldsymbol{a}_k]_i = -1 \times [\boldsymbol{a}_k]_i$ signifies that only the i-th element of $\boldsymbol{a}_k$ is multiplied by -1, while all other elements remain unchanged. Suppose $f(\boldsymbol{A}, \boldsymbol{U}, \boldsymbol{E})$ be a function of the matrix $\boldsymbol{A}$, $\boldsymbol{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_r)$ and $\boldsymbol{E} = (\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_r)$, then*

(1) if $[\boldsymbol{a}_k]_i \leftrightarrow [\boldsymbol{a}_k]_j$ or $[\boldsymbol{a}_k]_i = -1 \times [\boldsymbol{a}_k]_i$, $\mathbb{E}[f]$ remain unchanged.

(2) if $[\boldsymbol{a}_k]_i \leftrightarrow [\boldsymbol{a}_k]_j \Rightarrow [\boldsymbol{u}_k]_i \leftrightarrow [\boldsymbol{u}_k]_j$ and $[\boldsymbol{e}_k]_i \leftrightarrow [\boldsymbol{e}_k]_j$.

(3) if $[\boldsymbol{a}_k]_i = -1 \times [\boldsymbol{a}_k]_i \Rightarrow [\boldsymbol{u}_k]_i = -1 \times [\boldsymbol{u}_k]_i$, $[\boldsymbol{e}_k]_i = -1 \times [\boldsymbol{e}_k]_i$, $[\boldsymbol{u}_k]_j = 1 \times [\boldsymbol{u}_k]_j$, *and* $[\boldsymbol{e}_k]_j = 1 \times [\boldsymbol{e}_k]_j$, *where* $i \neq j$.

(4) $\mathbb{E}\left[\frac{[\boldsymbol{u}_k]_i^2}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k \rangle}\right] = \frac{1}{n}$.

(5) $\mathbb{E}\left[\frac{[\boldsymbol{u}_k]_i [\boldsymbol{u}_k]_j}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k \rangle}\right] = 0$, *where* $i \neq j$.

Proof. According to real analysis, the matrix $\boldsymbol{A}$ is full rank almost everywhere under a Gaussian distribution, and both $\boldsymbol{u}_k$ and $\boldsymbol{e}_k$ are non-zero almost everywhere.

(1) Since $\boldsymbol{a}_k$ is independently and identically distributed, it obviously holds.

(2) For base case $k = 1$, it obviously holds. Assume the result holds for all $k = 1, 2, \cdots, k-1$, where $k \geq 2$, then $[\boldsymbol{a}_k]_i \leftrightarrow [\boldsymbol{a}_k]_j \Rightarrow [\boldsymbol{u}_k]_i = [\boldsymbol{a}_k]_i - \sum_{s=1}^{k-1} \langle \boldsymbol{a}_k, \boldsymbol{e}_s \rangle [\boldsymbol{e}_s]_i$, $[\boldsymbol{u}_k]_j = [\boldsymbol{a}_k]_j - \sum_{s=1}^{k-1} \langle \boldsymbol{a}_k, \boldsymbol{e}_s \rangle [\boldsymbol{e}_s]_j$, $[\boldsymbol{e}_k]_i = \frac{[\boldsymbol{u}_k]_i}{\|\boldsymbol{u}_k\|}$, and $[\boldsymbol{e}_k]_j = \frac{[\boldsymbol{u}_k]_j}{\|\boldsymbol{u}_k\|}$.

Thus, by strong induction, we have $[\boldsymbol{u}_k]_i \leftrightarrow [\boldsymbol{u}_k]_j$ and $[\boldsymbol{e}_k]_i \leftrightarrow [\boldsymbol{e}_k]_j$.

(3) For base case $k = 1$, it obviously holds. Assume the result holds for all $k = 1, 2, \cdots, k-1$, where $k \geq 2$, then

$$[\boldsymbol{a}_k]_i = -1 \times [\boldsymbol{a}_k]_i \Rightarrow \begin{cases} [\boldsymbol{u}_k]_i = [\boldsymbol{a}_k]_i \times (-1) - \sum_{s=1}^{k-1} \langle \boldsymbol{a}_k, \boldsymbol{e}_s \rangle [\boldsymbol{e}_s]_i \times (-1) = [\boldsymbol{u}_k]_i \times (-1) \\ [\boldsymbol{u}_k]_j = [\boldsymbol{u}_k]_j \times 1, i \neq j \end{cases}$$
$$\Rightarrow \begin{cases} [\boldsymbol{e}_k]_i \times (-1) = \frac{[\boldsymbol{u}_k]_i}{\|\boldsymbol{u}_k\|} \times (-1) \\ [\boldsymbol{e}_k]_j = [\boldsymbol{e}_k]_j \times 1, j \neq i \end{cases}$$

By strong induction, we have $[\boldsymbol{u}_k]_i = -1 \times [\boldsymbol{u}_k]_i$, $[\boldsymbol{e}_k]_i = -1 \times [\boldsymbol{e}_k]_i$, $[\boldsymbol{u}_k]_j = 1 \times [\boldsymbol{u}_k]_j$, and $[\boldsymbol{e}_k]_j = 1 \times [\boldsymbol{e}_k]_j$, where $i \neq j$. □

(4) Since $\left|\frac{[\boldsymbol{u}_k]_i^2}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k \rangle}\right| \leq 1$, $\mathbb{E}\left[\frac{[\boldsymbol{u}_k]_i^2}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k \rangle}\right]$ exists. $[\boldsymbol{a}_k]_i \leftrightarrow [\boldsymbol{a}_k]_j \Rightarrow \frac{[\boldsymbol{u}_k]_i^2}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k \rangle} \leftrightarrow \frac{[\boldsymbol{u}_k]_j^2}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k \rangle}$.

Thus, $\mathbb{E}\left[\frac{[\boldsymbol{u}_k]_i^2}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k \rangle}\right] \times n = \sum_{s=1}^{n} \mathbb{E}\left[\frac{[\boldsymbol{u}_k]_s^2}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k \rangle}\right] = \mathbb{E}\left[\frac{\langle \boldsymbol{u}_k, \boldsymbol{u}_k \rangle}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k \rangle}\right] = 1 \Rightarrow \mathbb{E}\left[\frac{[\boldsymbol{u}_k]_i^2}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k \rangle}\right] = \frac{1}{n}$.

(5) Since $\left|\frac{[\boldsymbol{u}_k]_i [\boldsymbol{u}_k]_j}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k \rangle}\right| \leq \left|\frac{[\boldsymbol{u}_k]_i^2 + [\boldsymbol{u}_k]_j^2}{2\langle \boldsymbol{u}_k, \boldsymbol{u}_k \rangle}\right| \leq 1$, $\mathbb{E}\left[\frac{[\boldsymbol{u}_k]_i [\boldsymbol{u}_k]_j}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k \rangle}\right]$ exists.

$[\boldsymbol{a}_k]_i = [\boldsymbol{a}_k]_i \times -1 \Rightarrow \mathbb{E}\left[\frac{[\boldsymbol{u}_k]_i [\boldsymbol{u}_k]_j}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k \rangle}\right] = \mathbb{E}\left[\frac{-[\boldsymbol{u}_k]_i [\boldsymbol{u}_k]_j}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k \rangle}\right] = 0$, where $i \neq j$.

Lemma 12. *Let* $\boldsymbol{A} \in \mathbb{R}^{n \times r}$ *be a matrix with independent standard normal entries, i.e., each element of* $\boldsymbol{A}$ *is an i.i.d.* $\mathcal{N}(0, 1)$ *random variable. Suppose* $\boldsymbol{A}$ *undergoes QR decomposition via the Gram-Schmidt process to yield a column-orthogonal matrix* $\boldsymbol{Q} \in \mathbb{R}^{n \times r}$ *with orthonormal columns* $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_r$ *and an upper triangular matrix* $\boldsymbol{R} \in \mathbb{R}^{r \times r}$. *Then, for each* $k = 1, 2, \ldots, r$, *the expected value of the outer product of the* k*-th orthonormal column vector* $\boldsymbol{e}_k$ *of* $\boldsymbol{Q}$ *is given by:*

$$\mathbb{E}[\boldsymbol{e}_k \boldsymbol{e}_k^T] = \frac{1}{n} \boldsymbol{I},$$

where $\boldsymbol{I}$ *is the* $n \times n$ *identity matrix.*

Proof. By the Gram-Schmidt process, we have $\boldsymbol{e}_k = \frac{\boldsymbol{u}_k}{\|\boldsymbol{u}_k\|}$, where $\boldsymbol{u}_k = \boldsymbol{a}_k - \sum_{s=1}^{k-1} \langle \boldsymbol{a}_k, \boldsymbol{e}_s \rangle \boldsymbol{e}_s$. Thus, $\boldsymbol{e}_k \boldsymbol{e}_k^T = \frac{\boldsymbol{u}_k \boldsymbol{u}_k^T}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k \rangle}$.

The (i, j)-th entry of $\mathbb{E}[\boldsymbol{e}_k \boldsymbol{e}_k^T]$ can be written as:

$$\mathbb{E}[[\boldsymbol{e}_k \boldsymbol{e}_k^T]_{ij}] = \mathbb{E}\left[\frac{[\boldsymbol{u}_k]_i [\boldsymbol{u}_k]_j}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k \rangle}\right].$$

For diagonal entries ($i = j$): When $i = j$, from Lemma 11(4), we have:

$$\mathbb{E}[[\boldsymbol{e}_k \boldsymbol{e}_k^T]_{ii}] = \mathbb{E}\left[\frac{[\boldsymbol{u}_k]_i^2}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k \rangle}\right] = \frac{1}{n}.$$

For off-diagonal entries ($i \neq j$): When $i \neq j$, from Lemma 11(5), we have:

$$\mathbb{E}[[\boldsymbol{e}_k \boldsymbol{e}_k^T]_{ij}] = \mathbb{E}\left[\frac{[\boldsymbol{u}_k]_i [\boldsymbol{u}_k]_j}{\langle \boldsymbol{u}_k, \boldsymbol{u}_k \rangle}\right] = 0.$$

Combining these two cases, we conclude that $\mathbb{E}[\boldsymbol{e}_k \boldsymbol{e}_k^T]$ is a diagonal matrix with all diagonal entries equal to $\frac{1}{n}$. Thus,

$$\mathbb{E}[\boldsymbol{e}_k \boldsymbol{e}_k^T] = \frac{1}{n}\boldsymbol{I},$$

where $\boldsymbol{I}$ is the $n \times n$ identity matrix. The proof is completed. □

Lemma 13. *Let $\boldsymbol{A} \in \mathbb{R}^{n \times r}$ be a matrix with independent standard normal entries, i.e., each element of $\boldsymbol{A}$ is an i.i.d. $\mathcal{N}(0, 1)$ random variable. Suppose $\boldsymbol{A}$ undergoes QR decomposition to yield an orthogonal matrix $\boldsymbol{Q} \in \mathbb{R}^{n \times r}$ with orthonormal columns and an upper triangular matrix $\boldsymbol{R} \in \mathbb{R}^{r \times r}$. Then, the expected value of the outer product of the matrix $\boldsymbol{Q}$ with itself is given by:*

$$\mathbb{E}[\boldsymbol{Q}\boldsymbol{Q}^T] = \frac{r}{n}\boldsymbol{I}$$

where $\boldsymbol{I}$ is the $n \times n$ identity matrix.

Proof. The QR decomposition of $\boldsymbol{A}$ is given by $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}$, where $\boldsymbol{Q}$ is an orthogonal matrix with columns $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_r$ and $\boldsymbol{R}$ is an upper triangular matrix. Since $\boldsymbol{Q}$ is orthogonal, $\boldsymbol{Q}\boldsymbol{Q}^T = \boldsymbol{I}_r$, where $\boldsymbol{I}_r$ is the $r \times r$ identity matrix. We aim to compute $\mathbb{E}[\boldsymbol{Q}\boldsymbol{Q}^T]$. By linearity of expectation and the fact that the columns of $\boldsymbol{Q}$ are orthonormal, we have:

$$\mathbb{E}[\boldsymbol{Q}\boldsymbol{Q}^T] = \mathbb{E}\left[\sum_{k=1}^{r} \boldsymbol{e}_k \boldsymbol{e}_k^T\right] = \sum_{k=1}^{r} \mathbb{E}[\boldsymbol{e}_k \boldsymbol{e}_k^T].$$

From Lemma 12, we know that $\mathbb{E}[\boldsymbol{e}_k \boldsymbol{e}_k^T] = \frac{1}{n}\boldsymbol{I}$ for each k. Therefore:

$$\mathbb{E}[\boldsymbol{Q}\boldsymbol{Q}^T] = \sum_{k=1}^{r} \frac{1}{n}\boldsymbol{I} = \frac{r}{n}\boldsymbol{I}.$$

The proof is completed. □

Lemma 14. *Let $\boldsymbol{A}_1 \in \mathbb{R}^{m \times r}$ and $\boldsymbol{A}_2 \in \mathbb{R}^{n \times r}$ be matrices with independent standard normal entries, i.e., each element of $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ is an i.i.d. $\mathcal{N}(0, 1)$ random variable. Suppose $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ undergo QR decomposition to yield orthogonal matrices $\boldsymbol{Q}_1 \in \mathbb{R}^{m \times r}$ and $\boldsymbol{Q}_2 \in \mathbb{R}^{n \times r}$ with orthonormal columns, respectively. Define $\boldsymbol{P} = \boldsymbol{Q}_2 \otimes \boldsymbol{Q}_1$, where $\otimes$ denotes the Kronecker product. Then, the expected value of the outer product of the matrix $\boldsymbol{P}$ with itself is given by:*

$$\mathbb{E}[\boldsymbol{P}\boldsymbol{P}^T] = \frac{r^2}{mn}\boldsymbol{I},$$

where $\boldsymbol{I}$ is the $mn \times mn$ identity matrix.

Proof. The Kronecker product $\boldsymbol{P} = \boldsymbol{Q}_2 \otimes \boldsymbol{Q}_1$ results in a matrix $\boldsymbol{P} \in \mathbb{R}^{mn \times r^2}$. From Lemma 13, we have $\boldsymbol{Q}_1\boldsymbol{Q}_1^T = \frac{r}{m}\boldsymbol{I}$ and $\boldsymbol{Q}_2\boldsymbol{Q}_2^T = \frac{r}{n}\boldsymbol{I}$. We aim to compute $\mathbb{E}[\boldsymbol{P}\boldsymbol{P}^T]$. Using the properties of the Kronecker product, we have:

$$\mathbb{E}[\boldsymbol{P}\boldsymbol{P}^T] = \mathbb{E}[(\boldsymbol{Q}_2 \otimes \boldsymbol{Q}_1)(\boldsymbol{Q}_2^T \otimes \boldsymbol{Q}_1^T)] = \mathbb{E}[(\boldsymbol{Q}_2\boldsymbol{Q}_2^T)] \otimes \mathbb{E}[(\boldsymbol{Q}_1\boldsymbol{Q}_1^T)] = \frac{r^2}{mn}\boldsymbol{I} \otimes \boldsymbol{I} = \frac{r^2}{mn}\boldsymbol{I}$$

The proof is completed. □

Now we can assess the impact of the lazy updates to $\boldsymbol{P}$.

Theorem 3. *Let $f \in C_{L_1}^{1,1}(\mathbb{R}^d)$ be a non-convex function bounded below by f^*. Suppose $\mathcal{E}_k = (\boldsymbol{z}_0, \boldsymbol{z}_1, \cdots, \boldsymbol{z}_k)$ with $\boldsymbol{z}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_q)$, and let $\mathcal{P}_j = (\boldsymbol{P}_0, \boldsymbol{P}_1, \cdots, \boldsymbol{P}_j)$, where $\boldsymbol{P}_j$ is defined in (10) with a fixed update frequency F. Then, the sequence $\{\boldsymbol{x}_k\}_{k>0}$ generated by Algorithm 3 satisfies:*

$$\frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}_{\mathcal{E}_k,\mathcal{P}_{\lfloor k/F \rfloor}}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right] \leq \epsilon$$

with $T = \mathcal{O}\left(\frac{d}{\epsilon}\right)$ if the perturbation scale satisfies $\varepsilon \leq \mathcal{O}\left(\frac{\epsilon^{1/2}}{q^{3/2}d^{1/2}L_1^{3/2}}\right)$. Here $T = KF$, where K denotes the total number of subspace updates.

Proof. Suppose $\mathcal{P}_j = (\boldsymbol{P}_0, \boldsymbol{P}_1, \cdots, \boldsymbol{P}_j)$, where $\boldsymbol{P}_j$ is the sequence generated by Eqn. (10) and $j \leq K$. In accordance with Lemma 8 and Lemma 10, if the subspace is fixed, we can transform the original problem $f \in C_{L_1}^{1,1}(\mathbb{R}^d)$ into $h \in C_{L_1}^{1,1}(\mathbb{R}^q)$ through transformation $h(\boldsymbol{y}) = f(\boldsymbol{x} + \boldsymbol{P}\boldsymbol{y})$. Consider the update rule:

$$\boldsymbol{y}_{j,0} = 0, h_j(\boldsymbol{y}) = f(\boldsymbol{x}_{jF} + \boldsymbol{P}_j\boldsymbol{y}), \forall j \in 0, 1, \cdots, K-1 \tag{25}$$

$$\boldsymbol{y}_{j,k} = \boldsymbol{y}_{j,k-1} - \eta\widehat{\nabla}h_j(\boldsymbol{y}_{j,k-1}), \forall k \in 0, 1, \cdots, F \tag{26}$$

$$\boldsymbol{x}_{jF+k} = \boldsymbol{x}_{jF} + \boldsymbol{P}_j\boldsymbol{y}_k, \tag{27}$$

In the j-th subspace, the projection matrix $\boldsymbol{P}_j$ remains constant, hence we can accumulate the changes of ϕ within the current subspace. Using Lemma 10, we have

$$\phi_{(j+1)F} - \phi_{jF} \leq -\frac{1}{4}\hat{\eta}\sum_{i=0}^{K-1}\mathbb{E}_{\mathcal{E}_{jF+i}}\left[\|\nabla h_j(\boldsymbol{y}_{j,i})\|^2\right] + \frac{\varepsilon^2(q+6)^3}{8}KL_1^2 + \frac{3\varepsilon^2(q+4)}{32}KL_1 \tag{28}$$

$$\leq -\frac{1}{4}\hat{\eta}\mathbb{E}_{\mathcal{E}_{jF}}\left[\|\nabla h_j(\boldsymbol{y}_{j,0})\|^2\right] + \frac{\varepsilon^2(q+6)^3}{8}KL_1^2 + \frac{3\varepsilon^2(q+4)}{32}KL_1. \tag{29}$$

Additionally, we note that $\nabla h_j(\boldsymbol{y}_{j,0}) = (\boldsymbol{P}_j)^\mathsf{T}\nabla f(\boldsymbol{x}_{jF})$. Taking expectations over the overall historical projection matrix $\mathcal{P}_j$, and noting Lemma 14, $\mathbb{E}[\boldsymbol{P}_j(\boldsymbol{P}_j)^\mathsf{T}] = \frac{q}{d}\boldsymbol{I}$, with $\boldsymbol{P}_j$ independent of $\boldsymbol{x}_{jF}$, we get

$$\mathbb{E}_{\mathcal{P}_{j+1}}[\phi_{(j+1)F}] - \mathbb{E}_{\mathcal{P}_j}[\phi_{jF}] \leq -\frac{1}{4}\hat{\eta}\mathbb{E}_{\mathcal{E}_{jF},\mathcal{P}_j}\left[\|(\boldsymbol{P}_j)^\mathsf{T}\nabla f(\boldsymbol{x}_{jF})\|^2\right] + \frac{\varepsilon^2(q+6)^3}{8}KL_1^2 + \frac{3\varepsilon^2(q+4)}{32}KL_1 \tag{30}$$

$$= -\frac{q}{4d}\hat{\eta}\mathbb{E}_{\mathcal{E}_{jF},\mathcal{P}_j}\left[\|\nabla f(\boldsymbol{x}_{jF})\|^2\right] + \frac{\varepsilon^2(q+6)^3}{8}KL_1^2 + \frac{3\varepsilon^2(q+4)}{32}KL_1. \tag{31}$$

Assuming $f(\boldsymbol{x}) \geq f^*$ holds for all $\boldsymbol{x} \in \mathbb{R}^d$, and letting $T = KF$, summing the inequality yields

$$\mathbb{E}_{\mathcal{P}_{K-1}}[\phi_T] \leq \mathbb{E}_{\mathcal{P}_0}[\phi_0] - \frac{q}{4d}\hat{\eta}\sum_{j=0}^{K-1}\mathbb{E}_{\mathcal{E}_{jF},\mathcal{P}_j}\left[\|\nabla f(\boldsymbol{x}_{jF})\|^2\right] + T\frac{\varepsilon^2(q+6)^3}{8}L_1^2 + T\frac{3\varepsilon^2(q+4)}{32}L_1. \tag{32}$$

Since $\mathbb{E}_{\mathcal{P}_{K-1}}[\phi_T] \geq f^*$, we have:

$$f^* \leq \mathbb{E}_{\mathcal{P}_0}[\phi_0] - \frac{q}{4d}\hat{\eta}\sum_{j=0}^{K-1}\mathbb{E}_{\mathcal{E}_{jF},\mathcal{P}_j}\left[\|\nabla f(\boldsymbol{x}_{jF})\|^2\right] + T\frac{\varepsilon^2(q+6)^3}{8}L_1^2 + T\frac{3\varepsilon^2(q+4)}{32}L_1. \tag{33}$$

Rearranging the inequality, we get

$$\frac{q}{4d}\hat{\eta}\sum_{j=0}^{K-1}\mathbb{E}_{\mathcal{E}_{jF},\mathcal{P}_j}\left[\|\nabla f(\boldsymbol{x}_{jF})\|^2\right] \leq \mathbb{E}_{\mathcal{P}_0}[\phi_0] - f^* + T\frac{\varepsilon^2(q+6)^3}{8}L_1^2 + T\frac{3\varepsilon^2(q+4)}{32}L_1. \tag{34}$$

Substituting $\hat{\eta} = \frac{1}{4(q+4)L_1}$, we obtain:

$$\frac{q}{16d(q+4)L_1}\sum_{j=0}^{K-1}\mathbb{E}_{\mathcal{E}_{jF},\mathcal{P}_j}\left[\|\nabla f(\boldsymbol{x}_{jF})\|^2\right] \leq \mathbb{E}_{\mathcal{P}_0}[\phi_0] - f^* + T\frac{\varepsilon^2(q+6)^3}{8}L_1^2 + T\frac{3\varepsilon^2(q+4)}{32}L_1. \tag{35}$$

Thus, we have

$$\frac{1}{T}\sum_{k=0}^{T-1}\mathbb{E}_{\mathcal{E}_k,\mathcal{P}_{\lfloor k/F\rfloor}}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right] \leq \frac{16(q+4)dL_1(\mathbb{E}_{\mathcal{P}_0}[\phi_0]-f^*)}{qT} + \frac{2\varepsilon^2(q+6)^3(q+4)d}{q}L_1^3 + \frac{3\varepsilon^2(q+4)^2d}{2q}L_1^2. \tag{36}$$

To ensure $\sum_{k=0}^{T-1}\mathbb{E}_{\mathcal{E}_k,\mathcal{P}_{\lfloor k/F\rfloor}}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right] \leq \epsilon$, we can choose

$$\varepsilon \leq \mathcal{O}\left(\frac{\epsilon^{1/2}}{q^{3/2}d^{1/2}L_1^{3/2}}\right).$$

Then, the upper bound for the expected number of steps is $\mathcal{O}\left(\frac{d}{\epsilon}\right)$. The proof is completed. □