Multilevel Objective-Function-Free Optimization with an Application to Neural Networks Training

Gratton, Serge; Kopanicakova, Alena; TOINT, Philippe

Publication date:
2023

Document Version
Early version, also known as pre-print

Multilevel Objective-Function-Free Optimization with an Application to Neural Networks Training

Serge Gratton*, Alena Kopaničáková†, Philippe L. Toint‡

3 II 2023

Abstract

A class of multi-level algorithms for unconstrained nonlinear optimization is presented which does not require the evaluation of the objective function. The class contains the momentum-less AdaGrad method as a particular (single-level) instance. The choice of avoiding the evaluation of the objective function is intended to make the algorithms of the class less sensitive to noise, while the multi-level feature aims at reducing their computational cost. The evaluation complexity of these algorithms is analyzed and their behaviour in the presence of noise is then illustrated in the context of training deep neural networks for supervised learning applications.

Keywords: nonlinear optimization, multilevel methods, objective-function-free optimization (OFFO), complexity, neural networks, deep learning.

1 Introduction

In many cases, optimization problems involving a large number of variables do exhibit some kind of structure, be it sparsity of derivatives [3, 5, 17, 37, 48, 49], specific invariance properties [10, 19, 21, 28, 40] or implicit spectral properties [4, 20, 26, 27, 36, 39, 41], to cite three cases of interest. If the problem arises from the discretization of an underlying infinite-dimensional setting, it has long been known that considering different discretizations of the same problem using different mesh sizes and carefully using them in what is called a *multi-level* or *multigrid* algorithm can bring substantial computational benefits. When this is the case, the remarkable numerical performance is typically obtained by exploiting the natural hierarchy between these discretizations to successively eliminate the various frequency components of the error (or residual) [6] while, at the same time, using the fact that evaluations of functions and derivatives are typically cheaper for coarse discretizations than for fine ones. Multigrid methods are now a well-researched area of numerical analysis and are viewed as a crucial tool for the solution of linear and nonlinear systems resulting from the solution of elliptic partial-differential equations. Similar ideas have also made their way in nonlinear optimization, where both the MG-Opt [39, 41] and RMTR [26] multi-level frameworks have been designed to exploit the same properties, often very successfully (see [38, 25], for instance).

Another approach of optimization for large problems has recently been explored extensively, promoting the use of very simple *first-order methods* (see, among many others, [14, 47, 33, 45, 52, 24]). These methods have a very low computational cost per iterations, but typically require a (sometimes very) large number of them. Their popularity relies on several facts. The first is that

*Université de Toulouse, INP, IRIT, Toulouse, France. Work partially supported by 3IA Artificial and Natural Intelligence Toulouse Institute (ANITI), French "Investing for the Future - PIA3" program under the Grant agreement ANR-19-PI3A-0004. Email: serge.gratton@enseeiht.fr.

†Division of Applied Mathematics, Brown University, Providence, USA. Email: alena_kopanicakova@brown.edu.

‡Namur Center for Complex Systems (naXys), University of Namur, Namur, Belgium. Email: philippe.toint@unamur.be.

they can be shown to be convergent with a global rate which is sometimes comparable to that of more complicated methods. The second is that simplicity is achieved by avoiding the computation of the objective-function values and, most commonly, of other derivatives than gradients (hence their name). This in turn has made them very robust in the presence of noise on the function and its derivatives [22], an important feature when the problem is so large that these quantities can only be realistically estimated (typically by sampling) rather than calculated exactly. The context in which optimization is performed with computing function values is sometimes denoted by OFFO (Objective-Function-Free Optimization). A very large number of first-order OFFO methods have been investigated, but, for the purpose of this paper, we will focus on AdaGrad [14], one of the best-known provably-convergent members of this class.

The purpose of this paper is to demonstrate that it is *possible, theoretically sound, and practically efficient to combine multi-level and OFFO algorithms*. We achieve these objectives by presenting our contributions in three steps.

- We first describe a novel class of multi-level OFFO algorithms (of which AdaGrad can be viewed as a single-level realization) (Section 2).
- We then analyze the global rate of convergence of algorithms in this class, showing results matching the state of the art (Sections 3 and 4).
- We finally illustrate the use and advantages of the proposed methods in the context of noisy optimization problems resulting from the training of deep neural nets (DNNs) for supervised learning applications (Section 5).

A brief conclusion is then proposed in Section 6.

Notation. The symbol $\|\cdot\|$ stands for the standard Euclidean norm. If x is a vector, $|x|$ is the vector whose j-th component is $|x_j|$. The singular values of the matrix M are denoted by $\sigma_i[M]$.

2 The class of multilevel OFFO algorithms

We now present an idealized multilevel OFFO framework merging ideas from [23] and [26] and its analysis. The problem we consider is a structured version of smooth unconstrained optimization, that is

$$\min_{x\in\mathbb{R}^n} f(x) \tag{1}$$

for a twice continuously differentiable function f from $\mathbb{R}^n$ into $\mathbb{R}$. The problem is structured in that we assume that we know a collection of functions which provide a "hierarchical" set of approximations of the objective function f. As indicated above, this is typically the case when considering a function of a continuous problem's discretization (the hierarchy being then given by varying the discretization mesh) or when the objective function involves a graph whose description may vary in its level of detail. More specifically, we assume that we know a collection of functions $\{f_\ell\}_{\ell=1}^r$ such that each f_ℓ is a twice-continuously differentiable function from $\mathbb{R}^{n_\ell}$ to $\mathbb{R}$, the connection with our original problem being that $n_r = n$ and $f_r(x) = f(x)$ for all $x \in \mathbb{R}^n$. We also assume that, for each $\ell = 2,\ldots,r$, f_ℓ is "more costly" to evaluate/minimize than $f_{\ell-1}$. This may be because f_ℓ has more variables than $f_{\ell-1}$ (as would typically be the case if the f_ℓ represent increasingly finer discretizations of the same infinite-dimensional objective), or because the structure (in terms of partial separability, sparsity or eigenstructure) of f_ℓ is more complex than that of $f_{\ell-1}$, or for any other reason. To fix terminology, we will refer to a particular ℓ as a *level*. However, for $f_{\ell-1}$ to be useful at all in minimizing f_ℓ, there should be some relation between the variables of these two functions. We thus assume that, for each $\ell = 2,\ldots,r$, there exist a full-rank linear operator R_ℓ from $\mathbb{R}^{n_\ell}$ into $\mathbb{R}^{n_{\ell-1}}$ (the restriction) and another full-rank operator P_ℓ from $\mathbb{R}^{n_{\ell-1}}$ into $\mathbb{R}^{n_\ell}$ (the prolongation) such that

$$\omega P_\ell = R_\ell^T \tag{2}$$

for some known constant* $\omega > 0$. In the context of multigrid algorithms, P_ℓ and R_ℓ are interpreted as restriction and prolongation between a fine and a coarse grid (see [6, 41, 26, 25], for instance).

Before going into further details, we establish an important convention on indices. Since we will have to identify, sometimes simultaneously, a level, an iteration of our algorithm and a vector's component, we associate the index ℓ with levels, i with iterations and j with components. For instance, $x_{\ell,i,j}$ stands for the j-th component of the vector x at iteration i within level ℓ. The iteration index i will be reset to zero each time a level is entered†.

Because our proposal is to extend the ASTR1 objective-function-free framework of [23] to the multilevel context, we now review the main concepts of this algorithm and establish some notation. As the TR in the name suggests, ASTR1 is a trust-region optimization algorithm. This class of algorithms is well-known to be both theoretically sound (see [11] for an in-depth presentation and [51] for a more recent survey) and practically very efficient. As in all trust-region methods, the next iterate at level ℓ is found by minimizing a model of a level-dependent objective function h_ℓ within a region where the model is trusted. In our multilevel framework, this model can be either the (potentially quadratic) Taylor-like

$$m_{\ell,i}(s) = g_{\ell,i}^T s + \tfrac{1}{2} s^T B_{\ell,i} s, \tag{3}$$

where $g_{\ell,i} \stackrel{\rm def}{=} \nabla_x^1 h_\ell(x_{\ell,i})$ and $B_{\ell,i}$ is a bounded Hessian approximation, or the lower level model defined by

$$h_{\ell-1}(x_{\ell-1,0} + s_{\ell-1}) \stackrel{\rm def}{=} f_{\ell-1}(x_{\ell-1,0} + s_{\ell-1}) + v_{\ell-1}^T s_{\ell-1}, \tag{4}$$

where

$$v_{\ell-1} = R_\ell g_{\ell,i} - \nabla f_{\ell-1}(x_{\ell-1,0}). \tag{5}$$

By convention, we set $v_r = 0$, so that, for all s_r,

$$h_r(x_{r,0} + s_r) = f_r(x_{r,0} + s_r) = f(x_{r,0} + s_r) \ \text{ and } \ g_{r,k} = \nabla_x^1 h_r(x_{r,k}) = \nabla_x^1 f(x_{r,k}). \tag{6}$$

The model h_ℓ therefore corresponds to a modification of f_ℓ by a linear term that enforces the "linear coherence" relation

$$g_{\ell-1,0} = \nabla_x^1 h_{\ell-1}(x_{\ell-1,0}) = R_\ell g_{\ell,i}. \tag{7}$$

This first-order modification (4) is commonly used in multigrid applications in the context of the full approximation scheme [6], but also in other contexts [16, 41, 1, 38, 26, 34]. We call it "linear coherence" because it crucially ensures that the first-order behaviours of h_ℓ and $h_{\ell-1}$ are coherent in a neighbourhood of $x_{\ell,i}$ and $x_{\ell-1,0}$, respectively. To see this, one checks that, if s_ℓ and $s_{\ell-1}$ satisfy $s_\ell = P_\ell s_{\ell-1}$, then, using (2) and (7),

$$g_{\ell,i}^T s_\ell = g_{\ell,i}^T P_\ell s_{i-1} = \frac{1}{\omega} R_\ell g_{\ell,i}^T s_{\ell-1} = \frac{1}{\omega} g_{\ell-1,0}^T s_{\ell-1}. \tag{8}$$

Once the model is defined/chosen, the typical iteration of a trust-region method proceeds by minimizing it in a ball centered at the current iterate, whose radius is adaptively computed by the algorithm, depending on past performance. This ball can be defined in different norms, but we will focus here on a scaled version of the "infinity norm" where the absolute value of each vector component is measured individually. In the ASTR1 context, the trust-region radius is computed using the size of the current gradient and a component-wise strictly positive vector of *weights*, which we do not fully define now, but which will be specified later in our analysis. Since our multilevel algorithm is recursive, it is also necessary to force termination at a given level when the Euclidean norm of the prolongation of the overall step to the previous level becomes too large, that is when the inequality

$$\|P_{\ell+1}(x_{\ell,i} - x_{\ell,0})\| \leq \delta_\ell \tag{9}$$

*For simplicity, we choose to make ω independent of ℓ, which can always be achieved by scaling.

†We are well aware that this creates some ambiguities, since a sequence of indices ℓ, i can occur more than once if level ℓ ($\ell < r$) is used more than once, implying the existence of more than one starting iterate at this level. This ambiguity is resolved by the context.

fails, where $\delta_\ell \geq 0$ is a bound on norm of the step at level $\ell+1$ if $\ell < r$ or $+\infty$ otherwise.

In order to specify the algorithm, we finally define, for a vector of weights w_ℓ of size n_ℓ, the diagonal matrices

$$D(\{w_\ell\}) \stackrel{\rm def}{=} \mathrm{diag}\left(\frac{1}{w_{\ell,1}}, \ldots, \frac{1}{w_{\ell,n_\ell}}\right). \tag{10}$$

We will assume that, for $j \in \{1, \ldots, n_\ell\}$, there exists a constant $\varsigma_j \in (0, 1]$, such that $w_{\ell,j} \geq \varsigma_j$ for each $\ell \in \{1, \ldots, r\}$. The Multilevel Objective-Function-Free Trust-Region (MOFFTR) algorithm is then specified on the following page.

Solving the original problem (1) is then obtained by calling

$$\mathsf{MOFFTR}(\, r,\, f,\, x_{r,0},\, \epsilon_r,\, i_r^{(\max)},\, +\infty,\, \varsigma\,), \tag{11}$$

where $i_r^{(\max)}$ is the maximum number of top level iterations and ς denotes the vector of weights' lower bounds. In order to fix terminology, we say that iterations at which the step is computed by Step 4 of the algorithm are *Taylor iterations*, while iterations at which the step results from the recursive call (17) are called *recursive iterations*.

Some comments are useful at this stage to further explain and motivate the details of the algorithm.

1. Note that the iterations at any level are terminated in Step 1 if a level-dependent accuracy threshold ϵ_ℓ is achieved or if a level-dependent maximum number of iterations $i_\ell^{(\max)}$ is reached. Also note that (12) enforces (9).

2. The componentwise trust-region radius is defined in (13). Observe that this choice prevents nonzero components of the step whenever the corresponding component of the gradient is zero. Observe also that (13) avoids large steps which would cause (12) to fail for $i+1$. Indeed, if $\ell < r$, (13) and (19) imply that

   $$\|P_{\ell+1}(x_{\ell,i+1} - x_{\ell,i})\| = \|P_{\ell+1} s_{\ell,i}\| \leq 2\delta_\ell.$$

   Thus any $x_{\ell,i+1}$ which would violate this inequality would not satisfy (12) (for $i+1$) since then

   $$\|P_{\ell+1}(x_{\ell,i+1} - x_{\ell,0})\| \geq \|P_{\ell+1}(x_{\ell,i+1} - x_{\ell,i})\| - \|P_{\ell+1}(x_{\ell,i} - x_{\ell,0})\| > 2\delta_\ell - \delta_\ell = \delta_\ell,$$

   where we used (12) (for i) to derive the last inequality.

   The choice of model (end of Step 2) is not formally determined and left to the user. In a typical pattern, known in the multigrid literature as a "V-cycle", the tasks to perform at a given level ℓ is as follows. A set of standard Taylor iterations is first performed. Then, if $\ell > 1$ (that is the current level is not the lowest one) and significant progress is likely on level $\ell-1$ (in the sense of (16) failing), one then recursively calls the algorithm at level $\ell-1$. A second set of Taylor iterations is then performed at level ℓ. The "V" shape suggested by the name results from the recursive application of this pattern at all levels. While it is customary to specify the number of Taylor iterations in both sets (the "pre-smoothing" and "post-smoothing" in multigrid parlance), this is not required in MOFFTR. Indeed the algorithm allows for a wide variety of iteration patterns, fixed or adaptive.

3. We next review the mechanism of Step 3 and start by noting that $\delta_{\ell,i}$ is the Euclidean norm of the step that would be allowed at iteration (ℓ, i), had this iteration been been a Taylor one (see (13)). We then select a set of weights $w_{\ell-1,0}$ to be used at the lower level. Beyond being bounded below by their respective ς_j, these weights have to satisfy two further conditions. The first, (14), is expressed in a very generic way for now and ensures that these weights

Algorithm 2.1: $\boxed{x_+ = \mathsf{MOFFTR}(\ell, h_\ell, x_{\ell,0}, \epsilon_\ell, i_\ell^{(\max)}, \delta_\ell, w_{\ell,0})}$

Step 0: Initialization. The constants $\kappa_{\rm R} \in (0,1)$, $\alpha \geq 1$, $\tau \in (0,1]$, $\kappa_{\rm B} \geq 1$ and $\varsigma_j \in (0,1]$ ($j \in \{1,\ldots,n_\ell\}$) are given. Set $i = 0$.

Step 1: Termination test. If $\ell < r$ and

$$\|P_{\ell+1}(x_{\ell,i} - x_{\ell,0})\| > \delta_\ell \tag{12}$$

return with $x_+ = x_{\ell,i-1}$. Otherwise, compute $g_{\ell,i} \stackrel{\rm def}{=} \nabla_x^1 h_\ell(x_{\ell,i})$. If $\|g_{\ell,i}\| \leq \epsilon_\ell$ or $i = i_\ell^{(\max)}$, return with $x_+ = x_{\ell,i}$.

Step 2: Define the trust-region. Set

$$\widehat{\Delta}_{\ell,i} = D(\{w_{\ell,i}\})|g_{\ell,i}| \;\;\text{and}\;\; \Delta_{\ell,i} = \begin{cases} \widehat{\Delta}_{\ell,i} & \text{if} \quad \ell = r, \\ \min\left[\dfrac{2\delta_\ell}{\|P_{\ell+1}\|\,\|\widehat{\Delta}_{\ell,i}\|}, 1\right] \widehat{\Delta}_{\ell,i} & \text{if} \quad \ell < r. \end{cases} \tag{13}$$

If $i > 0$, define $w_{\ell,i} \in \mathbb{R}^{n_\ell}$ such that $w_{\ell,i,j} \geq \varsigma_j$ for $j \in \{1,\ldots,n_\ell\}$. If a Taylor step is required at iteration i, go to Step 4.

Step 3: Recursive step. Select $w_{\ell-1,0} \in \mathbb{R}^{n_{\ell-1}}$ such that $w_{\ell-1,0,j} \geq \varsigma_j$ for $j \in \{1,\ldots,n_{\ell-1}\}$,

$$\boxed{\text{“the lower-level weights are large enough”}} \tag{14}$$

and

$$\|D(\{w_{\ell-1,0}\})\,|R_\ell g_{\ell,i}|\| \leq \frac{\alpha\|\Delta_{\ell,i}\|}{\|P_\ell\|}. \tag{15}$$

If either $\ell = 1$ or

$$\sum_{j=1}^{n_\ell - 1} \frac{[R_\ell g_{\ell,i}]_j^2}{w_{\ell-1,0,j}} < \kappa_{\rm R} \sum_{j=1}^{n_\ell} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}} \tag{16}$$

then go to Step 4. Otherwise (i.e. if $\ell > 1$ and (16) fails), compute

$$s_{\ell,i} = P_\ell\Big[\mathsf{MOFFTR}(\ell-1, h_{\ell-1}, R_\ell x_{\ell,i}, \epsilon_{\ell-1}, i_{\ell-1}^{(\max)}, \alpha\|\Delta_{\ell,i}\|, w_{\ell-1,0}) - R_\ell x_{\ell,i}\Big], \tag{17}$$

where $h_{\ell-1}$ is given by (4).

Step 4: Taylor step. Select a symmetric Hessian approximation $B_{\ell,i}$ such that

$$\|B_{\ell,i}\| \leq \kappa_{\rm B}. \tag{18}$$

Compute a step $s_{\ell,i}$ such that

$$|s_{\ell,i,j}| \leq \Delta_{\ell,i,j} \quad (j \in \{1,\ldots,n_\ell\}), \tag{19}$$

and

$$g_{\ell,i}^T s_{\ell,i} + \tfrac{1}{2} s_{\ell,i}^T B_{\ell,i} s_{\ell,i} \leq \tau\left(g_{\ell,i}^T s_{\ell,i}^Q + \tfrac{1}{2}(s_{\ell,i}^Q)^T B_{\ell,i} s_{\ell,i}^Q\right), \tag{20}$$

where

$$s_{\ell,i,j}^L = -\text{sign}(g_{\ell,i,j})\Delta_{\ell,i,j} \quad (j \in \{1,\ldots,n_\ell\}), \tag{21}$$

$$s_{\ell,i}^Q = \gamma_{\ell,i} s_{\ell,i}^L, \quad \text{with} \quad \gamma_{\ell,i} = \begin{cases} \min\left[1, \dfrac{|g_{\ell,i}^T s_{\ell,i}^L|}{(s_{\ell,i}^L)^T B_{\ell,i} s_{\ell,i}^L}\right] & \text{if} \quad (s_{\ell,i}^L)^T B_{\ell,i} s_{\ell,i}^L > 0, \\ 1 & \text{otherwise.} \end{cases} \tag{22}$$

Step 5: Update. Set

$$x_{\ell,i+1} = x_{\ell,i} + s_{\ell,i}, \tag{23}$$

Increment i by one and return to Step 1.

cannot be small if the weights at level ℓ are large. How this is achieved will depend on the specific choice of weights, as we will see below. The second is the seemingly obscure condition (15), which simply ensures that the global bound on the Euclidean norm on the total step at level $\ell-1$ is large enough to allow at least one iteration at the lower level. It states that the Euclidean length of the prolongation P_ℓ of the first lower level step is at most some multiple $\alpha \geq 1$ of the Euclidean length of the (hypothetical) step at level ℓ. Condition (16) then compares the decrease in a linear approximation of $h_{\ell-1}$ at $R_\ell x_{\ell,i}$ with that of the linear approximation of h_ℓ at $x_{\ell,i}$. If the former is less than a fraction $\kappa_{\rm R}$ of the latter, this suggests that "significant progress at the lower level is unlikely", and we then resort to continue minimization at the current level. If significant progress is likely, we then choose to minimize $h_{\ell-1}$ at the lower level (recursive iteration) using the weights $w_{\ell-1,0}$, starting from $R_\ell x_{\ell,i}$ and within a Euclidean ball of radius $\alpha\|\Delta_{\ell,i}\|$.

We observe that (15) is quite easy to satisfy. Indeed, one readily checks that it is guaranteed if

$$w_{\ell-1,0,j} \geq \max\left[\varsigma_j, \frac{\|P_\ell\|\,|R_\ell g_{\ell,i}|_j}{\alpha\|\Delta_{\ell,i}\|}\right]. \tag{24}$$

4. The step at Taylor iterations is computed in Step 4 using a technique borrowed from the ASTR1 algorithm [23]. The reader familiar with trust-region theory will recognize in $s^Q_{\ell,i}$ a variant the "Cauchy point" obtained by minimizing the quadratic model on the intersection of the negative gradient's span and the trust-region (see [11, Section 6.3.2]), while $s^L_{\ell,i}$ is minimizer of the simpler linear model $g^T_{\ell,i}s$ within the trust-region.

5. Neither the objective function f_r or its approximations $\{f_\ell\}_{\ell=1}^{r-1}$ are ever evaluated and the optimization method combining them is therefore truly "Objective-Function-Free".

3 Convergence Analysis

Our convergence analysis is based on the following standard assumptions.

AS.1: For each $\ell \in \{1,\ldots,r\}$, the function f_ℓ is continuously differentiable.

AS.2: For each $\ell \in \{1,\ldots,r\}$, the gradient $\nabla^1_x f_\ell(x)$ is Lipschitz continuous with Lipschitz constant $L \geq 0$, that is

$$\|\nabla^1_x f_\ell(x) - \nabla^1_x f_\ell(y)\| \leq L\|x-y\|$$

for all $x, y \in \mathbb{R}^n$ and all $\ell \in \{1,\ldots,r\}$.

AS.3: There exists a constant $f_{\rm low}$ such that, for all x, $f(x) \geq f_{\rm low}$.

In what follows, we use the notation

$$\Gamma_0 \stackrel{\rm def}{=} f(x_0) - f_{\rm low} \tag{25}$$

for the gap between the objective function at the starting point and its lower bound. Note that there is no assumption that the gradients of the f_ℓ remain bounded.

Before considering more specific choices for the weights, we first derive a fundamental property of the Taylor steps and strengthen [23, Lemma 2.1] quantifying the "linear decrease" (that is the decrease in the simple linear model of the objective) for Taylor iterations.

Lemma 3.1 Suppose that AS.1 and AS.2 hold. Consider a Taylor iteration i at level ℓ. Then

$$g_{\ell,i}^T s_{\ell,i} \leq -\frac{\tau \varsigma_{\min}}{2\kappa_{\rm B}} \sum_{j=1}^{n_\ell} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}} + \frac{\kappa_{\rm B}}{2} \|\Delta_{\ell,i}\|^2 \tag{26}$$

where $\varsigma_{\min} \stackrel{\rm def}{=} \min_{j\in\{1,\ldots,n_\ell\}} \varsigma_j \in (0,1]$.

Proof. First note that, because of (21) and the definition of $w_{\ell,i,j}$,

$$|g_{\ell,i}^T s_{\ell,i}^L| = \sum_{j=1}^{n_\ell} \frac{w_{\ell,i,j} g_{\ell,i,j}^2}{w_{\ell,i,j}^2} \geq \sum_{j=1}^{n_\ell} \frac{\varsigma_j g_{\ell,i,j}^2}{w_{\ell,i,j}^2} \geq \varsigma_{\min} \|s_{\ell,i}^L\|^2. \tag{27}$$

Suppose now that $(s_{\ell,i}^L)^T B_{\ell,i} s_{\ell,i}^L > 0$ and $\gamma_{\ell,i} < 1$. Then, in view of (20), (22), (27) and (18),

$$g_{\ell,i}^T s_{\ell,i}^Q + \tfrac{1}{2}(s_{\ell,i}^Q)^T B_{\ell,i} s_{\ell,i}^Q = \gamma_{\ell,i} g_{\ell,i}^T s_{\ell,i}^L + \tfrac{1}{2}\gamma_{\ell,i}^2 (s_{\ell,i}^L)^T B_{\ell,i} s_{\ell,i}^L = -\frac{(g_{\ell,i}^T s_{\ell,i}^L)^2}{2(s_{\ell,i}^L)^T B_{\ell,i} s_{\ell,i}^L} \leq -\frac{\varsigma_{\min} |g_{\ell,i}^T s_{\ell,i}^L|}{2\kappa_{\rm B}}.$$

Combining this inequality with (22) then gives that

$$g_{\ell,i}^T s_{\ell,i}^Q + \tfrac{1}{2}(s_{\ell,i}^Q)^T B_{\ell,i} s_{\ell,i}^Q \leq -\frac{\varsigma_{\min}}{2\kappa_{\rm B}} \sum_{j=1}^{n_\ell} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}}. \tag{28}$$

Alternatively, suppose that $(s_{\ell,i}^L)^T B_{\ell,i} s_{\ell,i}^L \leq 0$ or $\gamma_{\ell,i} = 1$. Then, using (22), (21) and bounds $\kappa_{\rm B} \geq 1$ and $\varsigma_{\min} \leq 1$,

$$g_{\ell,i}^T s_{\ell,i}^Q + \tfrac{1}{2}(s_{\ell,i}^Q)^T B_{\ell,i} s_{\ell,i}^Q = g_{\ell,i}^T s_{\ell,i}^L + \tfrac{1}{2}(s_{\ell,i}^L)^T B_{\ell,i} s_{\ell,i}^L \leq \tfrac{1}{2} g_{\ell,i}^T s_{\ell,i}^L \leq -\frac{\varsigma_{\min}}{2\kappa_{\rm B}} \sum_{j=1}^{n_\ell} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}}. \tag{29}$$

We thus obtain from (28), (29) and (20) that

$$g_{\ell,i,}^T s_{\ell,i} + \tfrac{1}{2} s_{\ell,i}^T B_{\ell,i} s_{\ell,i} \leq -\frac{\tau \varsigma_{\min}}{2\kappa_{\rm B}} \sum_{j=1}^{n_\ell} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}}.$$

As a consequence, we deduce from (28), (29), (20) and (18) that

$$g_{\ell,i}^T s_{\ell,i} \leq -\tau \varsigma_{\min} \sum_{j=1}^{n_\ell} \frac{g_{\ell,i,j}^2}{2\kappa_{\rm B} w_{\ell,i,j}} + \tfrac{1}{2} |s_{\ell,i}^T B_{\ell,i} s_{\ell,i}| \leq -\tau \varsigma_{\min} \sum_{j=1}^{n_\ell} \frac{g_{\ell,i,j}^2}{2\kappa_{\rm B} w_{\ell,i,j}} + \frac{\kappa_{\rm B}}{2} \sum_{j=1}^{n_\ell} s_{\ell,i,j}^2,$$

and (26) results from (19). □

We also prove the following easy lemma.

Lemma 3.2 Consider iteration (ℓ, i) in the course of the MOFFTR algorithm. Then

$$\|s_{\ell,i}\| \leq \alpha \|D(\{w_{\ell,i}\}) |g_{\ell,i}|\|. \tag{30}$$

Proof. If iteration (ℓ, i) is a Taylor iteration, (30) results from (13), (19) and the bound $\alpha \geq 1$. If it is a recursive iteration, we have, from (17) and (19), that

$$\|s_{\ell,i}\| = \|P_\ell(x_{\ell-1,i} - x_{\ell-1,0})\| \leq \delta_{\ell-1} = \alpha \|D(\{w_{\ell,i}\}) |g_{\ell,i}|\|,$$

yielding (30). □

We now consider what can happen at a recursive iteration.

Lemma 3.3 Consider an recursive iteration (ℓ, i). Then

$$\|\Delta_{\ell-1,0}\| \leq \frac{\alpha}{\|P_\ell\|}\|\Delta_{\ell,i}\|, \tag{31}$$

the iterate $x_{\ell-1,1}$ is accepted in Step 1 of the algorithm and at least one iteration is completed at level $\ell-1$.

Proof. Because of (13), (15) and the calling sequence of MOFFTR, we have that

$$\|\Delta_{\ell-1,0}\| \leq \|\widehat{\Delta}_{\ell-1,0}\| = \|D(\{w_{\ell-1,0}\})|g_{\ell-1,0}|\| = \|D(\{w_{\ell-1,0}\})|R_\ell g_{\ell,i}|\| \leq \frac{\alpha}{\|P_\ell\|}\|\Delta_{\ell,i}\|$$

and hence, using (19), that

$$\|P_\ell(x_{\ell-1,1} - x_{\ell-1,0})\| \leq \|P_\ell\|\,\|x_{\ell-1,1} - x_{\ell-1,0}\| \leq \|P_\ell\|\,\|\Delta_{\ell-1,0}\| \leq \alpha\|\Delta_{\ell,i}\| = \delta_{\ell-1}.$$

Thus (12) fails at iteration $(\ell-1,1)$, the iterate $x_{\ell-1,1}$ is thus accepted and the desired conclusion follows. □

Lemma 3.4 Consider an recursive iteration (ℓ, i) and suppose that $i_{\ell-1} \geq 1$ iterations of the algorithm have been completed at level $\ell-1$. Then

$$\left| g_{\ell,i}^T s_{\ell,i} - \frac{1}{\omega}\sum_{k=0}^{i_{\ell-1}-1} g_{\ell-1,k}^T s_{\ell-1,k} \right| \leq \frac{2i_{\ell-1}^{(\max)} L\delta_{\ell-1}^2}{\omega\sigma_{\min}[P_\ell]^2}. \tag{32}$$

Proof. Using (17) and (2) (see also (8)), we deduce that

$$g_{\ell,i}^T s_{\ell,i} = g_{\ell,i}^T \sum_{k=0}^{i_{\ell-1}-1} P_\ell s_{\ell-1,k} = \frac{1}{\omega}\sum_{k=0}^{i_{\ell-1}-1} g_{\ell,i}^T R_\ell^T s_{\ell-1,k} = \frac{1}{\omega}\sum_{k=0}^{i_{\ell-1}-1} g_{\ell-1,0}^T s_{\ell-1,k}. \tag{33}$$

Now

$$g_{\ell-1,0}^T s_{\ell-1,k} = g_{\ell-1,k}^T s_{\ell-1,k} + (g_{\ell-1,0} - g_{\ell-1,k})^T s_{\ell-1,k} \stackrel{\rm def}{=} g_{\ell-1,k}^T s_{\ell-1,k} + \nu_{\ell-1,k}, \tag{34}$$

where, using the Cauchy-Schwarz inequality, (4), AS.2 and (9),

$$\begin{aligned} |\nu_{\ell-1,k}| &\leq \|g_{\ell-1,0} - g_{\ell-1,k}\|\,\|s_{\ell-1,k}\| \\ &= \|\nabla_x^1 f_\ell(x_{\ell-1,0}) - \nabla_x^1 f_\ell(x_{\ell-1,k})\|\,\|s_{\ell-1,k}\| \\ &\leq L\|x_{\ell-1,k} - x_{\ell-1,0}\|\,\|s_{\ell-1,k}\| \\ &\leq \frac{L}{\sigma_{\min}[P_\ell]^2}\|P_\ell(x_{\ell-1,k} - x_{\ell-1,0})\|\,\|P_\ell s_{\ell-1,k}\| \\ &\leq \frac{2L\delta_{\ell-1}^2}{\sigma_{\min}[P_\ell]^2}. \end{aligned} \tag{35}$$

Combining (33), (34), (35) and the bound $i_{\ell-1} \leq i_{\ell-1}^{(\max)}$ then yields (32). □

This crucial lemma allows us to quantify what can be said of the "linear decrease" at recursive iterations, and, as a consequence of Lemma 3.1, at all iterations of the MOFFTR algorithm.

Lemma 3.5 Suppose that AS.1 and AS.2 hold. Then, for all $\ell \in \{1, \ldots, r\}$ and all $i \geq 0$,

$$g_{\ell,i}^T s_{\ell,i} \leq -\beta_{1,r} \sum_{j=1}^{n} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}} + \beta_{2,r} \sum_{j=1}^{n} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}^2} \tag{36}$$

for some constants $\beta_{1,r} > 0$ and $\beta_{2,r} > 0$ independent of ℓ and i.

Proof. We first apply Lemma 3.4 to deduce that (32) holds. We also apply Lemma 3.3 to conclude that (31) holds and that $i_{\ell-1} \geq 1$. Suppose first that iteration (ℓ, i) is a recursive iteration and that ℓ is one plus the index of lowest level reached by the call to MOFFTR in (17). Then each iteration of the MOFFTR algorithm at level $\ell - 1$ is a Taylor iteration and inequality (26) in Lemma 3.1 applies. Thus, using (32) and (13), we derive that

$$\begin{aligned} g_{\ell,i}^T s_{\ell,i} &= -\frac{\tau \varsigma_{\min}}{2\kappa_{\rm B}\omega} \sum_{k=0}^{i_{\ell-1}-1} \sum_{j=1}^{n_{\ell-1}} \frac{g_{\ell-1,k,j}^2}{w_{\ell-1,k,j}} + \frac{\kappa_{\rm B}}{2\omega} \sum_{k=0}^{i_{\ell-1}-1} \|\Delta_{\ell-1,k}\|^2 + \frac{2 i_{\ell-1}^{(\max)} L \delta_{\ell-1}^2}{\omega \sigma_{\min}[P_\ell]^2} \\ &\leq -\frac{\tau \varsigma_{\min}}{2\kappa_{\rm B}\omega} \sum_{k=0}^{i_{\ell-1}-1} \sum_{j=1}^{n_{\ell-1}} \frac{g_{\ell-1,k,j}^2}{w_{\ell-1,k,j}} + \frac{\kappa_{\rm B}}{2\omega} \sum_{k=0}^{i_{\ell-1}-1} \frac{4\delta_{\ell-1}^2}{\sigma_{\min}[P_\ell]^2} + \frac{2 i_{\ell-1}^{(\max)} L \delta_{\ell-1}^2}{\omega \sigma_{\min}[P_\ell]^2} \\ &= -\frac{\tau \varsigma_{\min}}{2\kappa_{\rm B}\omega} \sum_{k=0}^{i_{\ell-1}-1} \sum_{j=1}^{n_{\ell-1}} \frac{g_{\ell-1,k,j}^2}{w_{\ell-1,k,j}} + \frac{2 i_{\ell-1}^{(\max)} (\kappa_{\rm B} + L)}{\omega \sigma_{\min}[P_\ell]^2} \delta_{\ell-1}^2. \end{aligned} \tag{37}$$

Taking now into account the fact that $i_{\ell-1} \geq 1$, ignoring now the terms for $k \in \{1, \ldots, i_{\ell-1}-1\}$ in the first sum of the right-hand side, using the definition of δ_ℓ from the call (17), the failure of (16) and (13), we obtain that

$$g_{\ell,i}^T s_{\ell,i} \leq -\frac{\tau \varsigma_{\min}}{2\kappa_{\rm B}\omega} \sum_{j=1}^{n_{\ell-1}} \frac{g_{\ell-1,0,j}^2}{w_{\ell-1,0,j}} + \frac{2 i_{\ell-1}^{(\max)} (\kappa_{\rm B} + L)}{\omega \sigma_{\min}[P_\ell]^2} \alpha^2 \|\Delta_{\ell,i}\|^2 \tag{38}$$

$$\leq -\frac{\tau \varsigma_{\min} \kappa_{\rm R}}{2\kappa_{\rm B}\omega} \sum_{j=1}^{n_\ell} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}} + \frac{2\alpha^2 i_{\ell-1}^{(\max)} (\kappa_{\rm B} + L)}{\omega \sigma_{\min}[P_\ell]^2} \|\widehat{\Delta}_{\ell,i}\|^2$$

$$\leq -\frac{\tau \varsigma_{\min} \kappa_{\rm R}}{2\kappa_{\rm B}\omega} \sum_{j=1}^{n_\ell} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}} + \frac{2\alpha^2 i_{\ell-1}^{(\max)} (\kappa_{\rm B} + L)}{\omega \sigma_{\min}[P_\ell]^2} \sum_{j=1}^{n_\ell} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}^2}. \tag{39}$$

Alternatively, if iteration (ℓ, i) is a Taylor iteration, (26) and (13) give that

$$g_{\ell,i}^T s_{\ell,i} \leq -\frac{\tau \varsigma_{\min} \kappa_{\rm R}}{2\kappa_{\rm B}} \sum_{j=1}^{n_\ell} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}} + \kappa_{\rm B} \sum_{j=1}^{n_\ell} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}^2}. \tag{40}$$

Combining (39) and (40), we obtain that, for all iterations at level ℓ (recursive and Taylor),

$$\begin{aligned} g_{\ell,i}^T s_{\ell,i} \leq & -\frac{\kappa_{\rm R}}{\max[\omega, 1]} \left[\frac{\tau \varsigma_{\min}}{2\kappa_{\rm B}} \right] \sum_{j=1}^{n_\ell} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}} \\ & + \max\left\{ [\kappa_{\rm B}], \frac{2\alpha^2 i_{\ell-1}^{(\max)}}{\max[\omega, 1]\, \sigma_{\min}[P_\ell]^2} \Big([\kappa_{\rm B}] + L \Big) \right\} \sum_{j=1}^{n_\ell} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}^2}. \end{aligned}$$

Note that the terms in square brackets correspond to the bound (40) with ℓ replaced by $\ell-1$ (because level $\ell-1$ contains Taylor iterations only). We may then recursively define

$$\beta_{1,1} \stackrel{\rm def}{=} \frac{\tau \varsigma_{\min}}{2\kappa_{\rm B}} \quad \text{and} \quad \beta_{1,\ell+1} \stackrel{\rm def}{=} \frac{\kappa_{\rm R}}{\max[\omega,1]}\,\beta_{1,\ell}, \tag{41}$$

$$\beta_{2,1} \stackrel{\rm def}{=} \kappa_{\rm B} \quad \text{and} \quad \beta_{2,\ell+1} \stackrel{\rm def}{=} \max\left\{\beta_{2,\ell}, \frac{2\alpha^2 i_{\ell-1}^{(\max)}}{\max[\omega,1]\,\sigma_{\min}[P_\ell]^2}\Big(\beta_{2,\ell}+L\Big)\right\} \tag{42}$$

for $\ell \in \{1,\ldots,r\}$ and obtain the desired conclusion. □

We may now deduce a central bound on the decrease of the objective function.

Lemma 3.6 Suppose that AS.1 and AS.2 hold. Then, for $\ell \in \{1,\ldots,r\}$,

$$h_\ell(x_{\ell,i}) - h_\ell(x_{\ell,i+1}) \geq \sum_{j=1}^{n_\ell} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}}\left[\beta_{1,r} - \frac{\beta_{2,r}+\frac{1}{2}\alpha^2 L}{w_{\ell,i,j}}\right]. \tag{43}$$

Proof. Successively using AS.1, AS.2, (30), (13) and (36), we obtain that

$$\begin{aligned}
h_\ell(x_{\ell,i+1}) &\leq h_\ell(x_{\ell,i}) + g_{\ell,i}^T s_{\ell,i} + \tfrac{1}{2}L\|s_{\ell,i}\|^2 \\
&\leq h_\ell(x_{\ell,i}) + g_{\ell,i}^T s_{\ell,i} + \tfrac{1}{2}L\alpha^2\|D(\{w_{\ell,i}\})|g_{\ell,i}|\|^2 \\
&= h_\ell(x_{\ell,i}) + g_{\ell,i}^T s_{r,i} + \tfrac{1}{2}\alpha^2 L\sum_{j=1}^{n_\ell}\frac{g_{\ell,i,j}^2}{w_{\ell,i,j}^2} \\
&\leq h_\ell(x_{\ell,i}) - \beta_{1,r}\sum_{j=1}^{n_\ell}\frac{g_{\ell,i,j}^2}{w_{\ell,i,j}} + \big(\beta_{2,r}+\tfrac{1}{2}\alpha^2 L\big)\sum_{j=1}^{n_\ell}\frac{g_{\ell,i,j}^2}{w_{\ell,i,j}^2}
\end{aligned} \tag{44}$$

giving (43). □

3.1 Divergent Weights

For our approach to be coherent and practical, we now have to specify how the weights are chosen, and also make condition (14) more explicit. We start by considering "divergent weights" defined as follows. We assume, in this section, that the weights $w_{i,k}$ are chosen such that, for some power parameter $0 < \nu \leq \mu < 1$, all $i \in \{1,\ldots,n\}$ and some constants $\varsigma_i \in (0,1]$,

$$\max[\varsigma_i, v_{i,j}]\,(i+1)^\nu \leq w_{r,i,j} \leq \max[\varsigma_i, v_{i,j}]\,(i+1)^\mu \quad (j \geq 0), \tag{45}$$

where, for each i, the $v_{i,j}$ are such that

$$v_{i+1,j} > v_{i,j} \quad \text{implies that} \quad v_{i+1,j} \leq |g_{r,i+1,j}| \tag{46}$$

and

$$v_{i,j} \geq |g_{r,i,j}|/a(i) \tag{47}$$

for some positive function $a(i)$ only depending on i. Using weights of the form

$$v_{i,j} = \max_{t\in\{0,\ldots,i\}} |g_{r,t,j}| \tag{48}$$

has resulted in good numerical performance when applied to noisy examples in the single-level case (see [23]). This particular choice, referred to as the MAXGI update rule, satisfies (46) and (47) (with $a(i) = 1$). The associated condition (14) is now specified as the requirement that

$$\min_{j\in\{1,\ldots,n_{\ell-1}\}} w_{\ell-1,0,j} \geq \min_{j\in\{1,\ldots,n_\ell\}} w_{\ell,i,j}. \tag{49}$$

Taking (24) into account, we see that the definition

$$w_{\ell-1,0,j} = \max\left[\varsigma_j, \frac{\|P_\ell\|\,|R_\ell g_{\ell,i}|_j}{\alpha\|\Delta_{\ell,i}\|}, \min_{j\in\{1,\ldots,n_\ell\}} w_{\ell,i,j}\right]$$

implies both (15) and (49). Note that we only need (45)-(47) for level r, the necessary growth of the weights for lower levels being guaranteed by (49).

Lemma 3.5 and (47) may be used to immediately deduce a lower bound on the change in the objective function's value obtained at each iteration at level r.

Lemma 3.7 Suppose that AS.1 and AS.2 hold. Then

$$f(x_i) - f(x_{i+1}) = h_r(x_{r,i}) - h_r(x_{r,i+1}) \geq -n(\beta_{2,r} + \tfrac{1}{2}\alpha^2 L)a(i)^2 \tag{50}$$

for all $i \geq 0$.

Proof. Ignoring negative terms in (43) with $\ell = r$ and using (45) and (47), we deduce that

$$\begin{aligned} h_r(x_{r,i+1}) &\leq h_r(x_{r,i}) + \big(\beta_{2,r} + \tfrac{1}{2}\alpha^2 L\big) \sum_{j=1}^{n} \frac{g_{r,i,j}^2}{w_{r,i,j}^2} \\ &\leq h_r(x_{r,i}) + \big(\beta_{2,r} + \tfrac{1}{2}\alpha^2 L\big) \sum_{j=1}^{n} \frac{g_{r,i,j}^2}{\max[\varsigma, v_{i,j}]^2 (j+1)^\nu} \\ &\leq h_r(x_{r,i}) + \big(\beta_{2,r} + \tfrac{1}{2}\alpha^2 L\big) \sum_{j=1}^{n} \frac{g_{r,i,j}^2}{v_{i,j}^2 (j+1)^\nu} \end{aligned}$$

and (50) follows from (6). □

We are now ready to state our main result for the MOFFTR algorithm using (45)-(47) and (49).

Theorem 3.8 Suppose that AS.1–AS.3 hold and that the MOFFTR algorithm is applied to problem (1) in a call of the form (11), where the weights $w_{\ell,i,j}$ are chosen according to (45)-(47) and the condition (14) is instantiated as (49). Then, for any $\vartheta \in (0, \beta_{1,r})$, there exists a subsequence $\{i_t\} \subseteq \{i\}_{i_\vartheta}^{\infty}$ such that

$$\min_{k\in\{i_\varsigma+1,\ldots,i_t\}} \|\nabla_x^1 f(x_{r,k})\|^2 = \min_{k\in\{i_\varsigma+1,\ldots,i_t\}} \|g_{r,k}\|^2 \leq \kappa_\diamond \frac{(i_t+1)^\mu}{i_t - i_\vartheta} \leq \frac{2\kappa_\diamond(i_\vartheta+1)}{i_t^{1-\mu}} \tag{51}$$

where

$$i_\vartheta \stackrel{\rm def}{=} \left(\frac{\beta_{2,r} + \frac{1}{2}\alpha^2 L}{\varsigma_{\min}(\beta_{1,r} - \vartheta)}\right)^{\frac{1}{\nu}} - 1, \qquad i_\varsigma \stackrel{\rm def}{=} \left(\frac{2(i_\vartheta+1)\kappa_\diamond}{\varsigma_{\min}}\right)^{\frac{1}{1-\mu}} \tag{52}$$

and

$$\kappa_\diamond \stackrel{\rm def}{=} \frac{2}{\vartheta}\left[f(x_0) - f_{\rm low} + n(\beta_{2,r} + \tfrac{1}{2}\alpha^2 L)\sum_{k=0}^{i_\vartheta} a(k)^2\right].$$

Proof. See Appendix A. □

Some comments on this result are in order.

1. Theorem 3.8 provides useful information on the rate of convergence of the MOFFTR algorithm beyond iteration of index i_ς, which can computed *a priori*. Indeed i_ς only depends on μ, ν and problem's constants. If $\{i_t\} = \{i\}_{i_\varsigma}^{\infty}$, the complexity bound to reach an iteration satisfying the accuracy requirement $\|g_k\| \leq \epsilon$ is then
$$\mathcal{O}\left(\epsilon^{-\frac{2}{1-\mu}}\right) + i_\varsigma(\mu,\nu), \tag{53}$$
which, for small values of ν and μ, can be close to $\mathcal{O}(\epsilon^{-2})$ (albeit at the price of a larger i_ς). This rate is also achieved by some variants of the single-level ASTR1 algorithm (see [23, Theorem 4.1]. If $\{i_t\} \subset \{i\}_{i_\varsigma}^{\infty}$, one may have to wait for the next iteration in $\{i_t\}$ beyond (53) for the gradient bound to be achieved. Note that the rate of decay of the right-hand side of (51) depends on the index i_t (in the complete sequence) rather than on t (the subsequence index). Interestingly, it is possible to prove that $\{i_t\} = \{i\}_{i_\vartheta}^{\infty}$ if one assumes that the objective-function's gradients remain uniformly bounded (see [23, Theorem 4.1]).

2. As all worst-case bounds, the bound (51) is pessimistic. In this context it is especially the case because we have only considered the case where all iterations before i_ϑ generate an increase in the objective function which is as large as allowed by our assumptions. This is extremely unlikely in practice.

3. It is also possible to relax condition (49) by only requiring that the right-hand side is at least a fixed fraction of the left-hand side. The arguments are essentially unmodifed, but involve yet another constant which percolates though the proofs. We haven't included this possibility to avoid further notational burden.

4. It was proved in [23, Theorem 4.2] that the above complexity bound is sharp for a single-level. It is therefore also sharp for the multilevel case.

5. Note that the requirement (45) allows a variety of choices for the weights. The specific choice (48) will be explored from the numerical point of view in the next section.

3.2 AdaGrad-like weights

Instead of focusing on (45), we now consider a choice of weights inspired by the popular AdaGrad method, where the necessary growth in weight size is obtained by accumulating squared gradient components. Interestingly, this will allow us to prove a complexity result for the complete sequence of iterates (no subsequence is involved). More specifically, given $\varsigma \in (0,1]$ and $\mu \in (0,1)$, we define the weights for all $\ell \in \{1,\ldots,r\}$, all $i \geq 0$ and $j \in \{1,\ldots,n_\ell\}$ by
$$w_{\ell,i,j} \stackrel{\rm def}{=} \left(\varsigma + \sum_{i=0}^{j} g_{\ell,i,j}^2\right)^{\mu}. \tag{54}$$
The AdaGrad weights are recovered for $\mu = \frac{1}{2}$. The condition (14) is then specified as the requirement that
$$\|w_{\ell-1,0}\| \geq \|w_{\ell,i}\|. \tag{55}$$
Taking again (24) into account, we verify that the definition
$$w_{\ell-1,0,j} = \max\left[1, \frac{\|w_{\ell,i}\|}{\|\widehat{w}_{\ell-1,0}\|}\right] \widehat{w}_{\ell-1,0,j} \quad \text{where} \quad \widehat{w}_{\ell-1,0,j} = \max\left[\varsigma_j, \frac{\|P_\ell\|\,|R_\ell g_{\ell,i}|_j}{\alpha\|\Delta_{\ell,i}\|}\right] \tag{56}$$
is sufficient to ensure both (15) and (55).

We now state our complexity result for the variant of the MOFFTR algorithm using (54) and (55). This result parallels [23, Theorem 3.2] but uses the more complex multilevel version of the linear decrease given by Lemma 3.6.

Theorem 3.9 Suppose that AS.1–AS.3 hold and that the MOFFTR algorithm is applied to problem (1) in a call of the form (11), where the weights are chosen according to (54) and (14) is instantiated as (55). Then

$$\operatorname*{average}_{k\in\{0,\ldots,i\}} \|\nabla_x^1 f(x_{r,k})\|^2 = \operatorname*{average}_{k\in\{0,\ldots,i\}} \|g_{r,k}\|^2 \leq \frac{\kappa_*}{i+1}, \tag{57}$$

where

$$\kappa_* \stackrel{\rm def}{=} \begin{cases} \max\left[\varsigma, \left(\dfrac{4n(\beta_{2,r}+\frac{1}{2}L)}{\beta_{1,r}(1-2\mu)}\right)^{\frac{1}{\mu}}, \dfrac{1}{2}\left(\dfrac{(1-2\mu)\Gamma_0}{n(\beta_{2,r}+\frac{1}{2}L)}\right)^{\frac{1}{1-2\mu}}\right] & \text{if} \quad 0<\mu<\frac{1}{2}, \\ \max\left[\varsigma, \dfrac{1}{2}e^{\frac{2\Gamma_0}{n(\beta_{2,r}+\frac{1}{2}L)}}, \dfrac{\varsigma\psi^2}{2}\left|W_{-1}\left(-\dfrac{1}{\psi}\right)\right|^2\right] & \text{if} \quad \mu=\frac{1}{2}, \\ \max\left[\varsigma, \left[\dfrac{2^\mu}{\beta_{1,r}}\left(\Gamma_0+\dfrac{n(\beta_{2,r}+L)\varsigma^{1-2\mu}}{2\mu-1}\right)\right]^{\frac{1}{1-\mu}}\right] & \text{if} \quad \frac{1}{2}<\mu<1, \end{cases} \tag{58}$$

where $\beta_{1,r}$ and $\beta_{2,r}$ are the constants defined by (41) and (42), respectively,

$$\psi \stackrel{\rm def}{=} \frac{4\max[\frac{3}{2}\beta_{r,1}, n(\beta_{r,2}+\frac{1}{2}L)]}{\beta_{1,r}\sqrt{\varsigma}} \tag{59}$$

and W_{-1} is the second branch of the Lambert function [12].

Proof. See Appendix B. □

We conclude this analysis section with a few brief comments on Theorem 3.9.

1. It results from this theorem that, for any $\epsilon > 0$, at most $\mathcal{O}(\epsilon^{-2})$ iterations of the MOFFTR algorithm are needed to reduce $\|g_{r,k}\|$ below ϵ. This result is thus stronger than that of Theorem 3.8, unless $\{i_t\} = \{i\}_{i_\vartheta}^{\infty}$. It is also equivalent (in order) to that known for single-level trust-region algorithms (see [7, Theorem 2.3.7]).

2. An exact momentum-less version of AdaGrad is obtained by choosing $r = 1$ and $\mu = \frac{1}{2}$. Theorem 3.9 therefore provides a convergence analysis for both single- and multi-level versions of this method.

3. It was shown in [23, Theorem 3.4] that the global rate of convergence of the single-level ($r = 1$) algorithm using AdaGrad-like weights cannot be better than $\mathcal{O}(1/\sqrt{i})$. This is therefore also the case for $r \geq 1$.

4. The bound (57) may be refined (although not improved in order) if one is ready to assume that the gradients are uniformly bounded. We refer the reader to [23] for a proof in the single-level case.

5. As noted in this last reference, the bound involving the Lambert function can be replaced by a weaker but more explicit one by using the inequality

$$\left|W_{-1}(-e^{-x-1})\right| \leq 1 + \sqrt{2x} + x \quad \text{for} \quad w > 0 \tag{60}$$

[9, Theorem 1]. Remembering that, for γ_1 and γ_2 given by (83), $\log\left(\frac{\gamma_2}{\gamma_1}\right) \geq \log(3) > 1$ and setting $x = \log\left(\frac{\gamma_2}{\gamma_1}\right) - 1 > 0$ in (60) then yields that

$$\left|W_{-1}\left(-\frac{\gamma_1}{\gamma_2}\right)\right| \leq \log\left(\frac{\gamma_2}{\gamma_1}\right) + \sqrt{2\left(\log\left(\frac{\gamma_2}{\gamma_1}\right) - 1\right)}.$$

6. The strict monotonicity of the weights implied by (54) can also be relaxed to provide further algorithmic flexibility. In turns out that (54) may be replaced by

$$w_{\ell,i,j} \in [\chi\, v_{\ell,i,j}, v_{\ell,i,j}] \;\text{ with }\; v_{\ell,i,j} \stackrel{\rm def}{=} \left(\varsigma + \sum_{i=0}^{j} g_{\ell,i,j}^2\right)^{\mu}$$

for some (fixed) $\chi \in (0,1]$ without altering the nature of Theorem 3.9, in that only the constant κ_* is modified to explicitly involve χ. Again, see [23] for a proof in the single-level case.

4 Extensions

The above theory can be extended is a number of practically useful and/or theoretically interesting ways.

4.1 Iteration-dependent algorithmic elements

There is much flexibility in the implementation of the MOFFTR algorithm than the statement on page 5 suggests, in part because we have considered certain algorithm's parameters as constant. While this is an advantage in many circumstances, it may happen that performance can be enhanced on specific problems by allowing these parameters to vary in a fixed range. This is for instance the case for α, the factor by which the lower-level trust-region radius can exceed the upper-level one. This is also the case of the definition of f_ℓ which is never used explicitly in our analysis, or of factor 2 in the right-hand side of the second part of (13). Thus, it is fair to say that our analysis covers a whole class of possible algorithms.

4.2 Exploiting lower-level iterations

The multilevel theory we have presented so far is limited[‡] to exploiting the first iteration of each level (see the transition between (37) and (38)). This approach is fairly coarse in the sense that it does not give any indication on why performing more than a single iteration at a lower level can be beneficial for convergence. To improve our understanding, we need to say more about how $h_{\ell-1}$ provides an approximation to h_ℓ. At iteration (ℓ, i), $\Delta_{\ell,i}$ is meant to represent the radius of the ball around $x_{\ell,i}$ in which the first-order Taylor model approximates h_ℓ sufficiently well. If $f_{\ell-1}$ in turn approximates f_ℓ somehow, we expect the linear decreases in $h_{\ell-1}$ (that is the terms $\sum_{j=1}^{n_{\ell-1}} g_{\ell-1,k,j}^2/w_{\ell-1,k,j}$) to be consistent with the linear decrease at level ℓ (that is $\sum_{j=1}^{n_\ell} g_{\ell-1,k,j}^2/w_{\ell-1,k,j}$) within the ball whose prolongation is of radius $\Delta_{\ell,i}$. In what follows, we consider what can be said if one assumes (or imposes) that condition (16) fails for all iterations $(\ell-1,k)$ rather than just for $(\ell-1,0)$, that is if

$$\sum_{j=1}^{n_{\ell-1}} \frac{g_{\ell-1,k,j}^2}{w_{\ell-1,k,j}} \geq \kappa_{\rm R} \sum_{j=1}^{n_\ell} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}} \quad \text{for} \quad k \in \{0,\ldots,i_{\ell-1}\}. \tag{61}$$

[‡]We ignore in this discussion the fact that evaluating gradients at the lower level is typically significantly cheaper computationally than computing them at the upper level, an advantage sometimes crucial in practice.

(Compared to (16), we may need to use a smaller $\kappa_{\rm R}$.)

Returning to Lemma 3.5 with this strengthened assumption, we obtain the following result.

Lemma 4.1 Suppose that AS.1, AS.2 and (61) hold. Then, for all $\ell \in \{1, \ldots, r\}$ and all $i \geq 0$,

$$g_{\ell,i}^T s_{\ell,i} \leq -i_\ell^{(\rm low)} \beta_{1,r} \sum_{j=1}^{n} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}} + \beta_{2,r} \sum_{j=1}^{n} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}^2} \tag{62}$$

for some constants $\beta_{1,r} > 0$ and $\beta_{2,r} > 0$ independent of ℓ and i, and where $i_\ell^{(\rm low)}$ is the total number of Taylor iterations from iteration (ℓ, i) to (and excluding) iteration $(\ell, i+1)$.

Proof. We first follow the proof of Lemma 3.5 up to (37). Then, instead of ignoring terms to obtain (38), we keep them and use (61) and (13) to deduce that

$$\begin{aligned} g_{\ell,i}^T s_{\ell,i} &\leq -i_{\ell-1} \frac{\tau \varsigma_{\min}}{2\kappa_{\rm B}\omega} \sum_{j=1}^{n_{\ell-1}} \frac{g_{\ell-1,0,j}^2}{w_{\ell-1,0,j}} + \frac{2 i_{\ell-1}^{(\max)} (\kappa_{\rm B} + L)}{\omega \sigma_{\min}[P_\ell]^2} \alpha^2 \|\Delta_{\ell,i}\|^2 \\ &\leq -i_{\ell-1} \frac{\tau \varsigma_{\min} \kappa_{\rm R}}{2\kappa_{\rm B}\omega} \sum_{j=1}^{n_\ell} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}} + \frac{2\alpha^2 i_{\ell-1}^{(\max)} (\kappa_{\rm B} + L)}{\omega \sigma_{\min}[P_\ell]^2} \|\widehat{\Delta}_{\ell,i}\|^2 \\ &\leq -i_{\ell-1} \frac{\tau \varsigma_{\min} \kappa_{\rm R}}{2\kappa_{\rm B}\omega} \sum_{j=1}^{n_\ell} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}} + \frac{2\alpha^2 i_{\ell-1}^{(\max)} (\kappa_{\rm B} + L)}{\omega \sigma_{\min}[P_\ell]^2} \sum_{j=1}^{n_\ell} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}^2}, \end{aligned} \tag{63}$$

where, as in Lemma 3.5, we used the failure of (16) and (13) to obtain the last two inequalities. As in Lemma 3.5, (40) still holds if iteration (ℓ, i) is a Taylor iteration (in which case $i_{\ell-1} = 0$). Combining (63) and (40), we obtain that, for all iterations at level ℓ,

$$\begin{aligned} g_{\ell,i}^T s_{\ell,i} \leq & -\max[1, i_{\ell-1}] \frac{\kappa_{\rm R}}{\max[\omega, 1]} \left[\frac{\tau \varsigma_{\min}}{2\kappa_{\rm B}} \right] \sum_{j=1}^{n_\ell} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}} \\ & + \max \left\{ [\kappa_{\rm B}], \frac{2\alpha^2 i_{\ell-1}^{(\max)}}{\max[\omega, 1]\, \sigma_{\min}[P_\ell]^2} \Big([\kappa_{\rm B}] + L \Big) \right\} \sum_{j=1}^{n_\ell} \frac{g_{\ell,i,j}^2}{w_{\ell,i,j}^2}. \end{aligned}$$

We may then recursively define $\beta_{1,\ell}$ and $\beta_{1,\ell}$ using (41) and (42) and (62) finally follows from the definition of $i_\ell^{(\rm low)}$. □

Observe that (62) is the same as (36), except that $\beta_{1,r}$ is now multiplied by $i_\ell^{(\rm low)}$. This modification percolates through all proofs, resulting in improved[§] constants in (52) and (58). We finally note that requiring (61) can be viewed as one way to improve the balance between negative and positive terms in (62), but may not be the only one.

4.3 Weak coherence

It is possible to relax somewhat the linear coherence requirement between high and low levels models. Examination of the above theory shows that it is only used in (33). If we were to assume that $\omega P_\ell^T = R_\ell + E_\ell$ instead of (2), then it is easy to verify that an error matrix E_ℓ satisfying

$$\|E_\ell g_{\ell,i}\| \leq \kappa_E \delta_{\ell-1} \tag{64}$$

[§]In particular, by offsetting the effect of the $i_{\ell-1}^{(\max)}$ constants in $\beta_{2,r}$.

for some fixed $\kappa_E \geq 0$ ensures that, for each t in (33),

$$g_{\ell,i}^T P_\ell s_{\ell-1,t} = \frac{1}{\omega}(R_\ell g_{\ell,i})^T s_{\ell-1,t} + \frac{\kappa_E \delta_{\ell-1}\|s_{\ell-1,t}\|}{\omega} \leq \frac{1}{\omega} g_{\ell-1,i}^T s_{\ell-1,t} + \frac{2\kappa_E}{\omega\sigma_{\min}[P_\ell]}\,\delta_{\ell-1}^2,$$

where we used (9) to derive the last inequality. This adds a term in $\mathcal{O}(\delta_{\ell-1}^2)$ in the right-hand side of (32), allowing the argument to be continued with different constants. The condition (64) is implementable because $\delta_{\ell-1}$ is known before R_ℓ or P_ℓ is used at iteration (ℓ, i) and may be quite lax in the early iterations where $\|g_{\ell,i}\|$ is still relatively large. That linear coherence often only needs to be preserved approximately is of particular relevance when the algorithm is applied to problems whose gradient is noisy. In that case, insisting on exact linear coherence would merely propagate the error in the gradient at the upper level to the lower level, which is clearly undesirable.

5 Numerical results

In this section, we illustrate the numerical performance of the proposed MOFFTR algorithms in the context of deep neural networks' training with a particular focus on supervised learning applications. Let $\mathcal{D} = \{(y_s, c_s)\}_{s=1}^{n_s}$ be a dataset of labeled data, where $y_s \in \mathbb{R}^{n_{in}}$ represents input features and $c_s \in \mathbb{R}^{n_{out}}$ denotes a desirable target. Our goal is to learn the parameters of DNNs, such that they can approximate c_s for a given y_s. In what follows, we exploit a continuous-in-depth approach to DNNs, the forward propagation of which can be interpreted as a discretization of a nonlinear ordinary differential equation (ODE) [44, 8, 50]. This approach allows us to construct a multilevel hierarchy and transfer operators required by the MOFFTR framework in a fairly natural way (see Section 5.1).

Using a continuous-in-depth approach, the supervised learning problem can be formulated as the following continuous optimal control problem [29]:

$$\begin{aligned} \min_{Q,q,\theta,W_T,b_T} \quad & \frac{1}{n_s}\sum_{s=1}^{n_s} \hbar(\mathcal{P}(W_T q_s(T) + b_T), c_s) + \int_0^T \mathcal{R}(\theta(t))\; dt + \mathcal{S}(W_T, b_T), \\ \text{subject to} \quad & \partial_t q_s(t) = \mathcal{F}(q_s(t), \theta(t)), \qquad \forall t \in (0,T), \\ & q_s(0) = Q y_s, \end{aligned} \tag{65}$$

where q denotes time-dependent states from $\mathbb{R}$ into $\mathbb{R}^{n_{fp}}$ and θ denotes the time-dependent control parameters from $\mathbb{R}$ into $\mathbb{R}^{n_c}$. The constraint in (65) continuously transforms an input feature y_s into final state $q_s(T)$, defined at the time T. This is achieved in two steps. Firstly, the input y_s is mapped into the dimension of the dynamical system as $q_s(0) = Qy_s$, where $Q \in \mathbb{R}^{n_{fp} \times n_{in}}$. Secondly, the nonlinear transformation of the features is performed using a "residual block" $\mathcal{F}$ from $\mathbb{R}^{n_{fp}} \times \mathbb{R}^{n_c}$ into $\mathbb{R}^{n_{fp}}$. We consider two types of such blocks: dense and convolutional. A dense residual block is defined as $\mathcal{F}(q_s(t), \theta(t)) := \sigma(W(t)q_s(t) + b(t))$, where $\theta(t) = (\mathrm{flat}(W(t)), \mathrm{flat}(b(t)))$, σ is an activation function from $\mathbb{R}^{n_{fp}}$ into $\mathbb{R}^{n_{fp}}$, $b(t) \in \mathbb{R}^{n_{fp}}$ is the "bias" and $W(t) \in \mathbb{R}^{n_{fp} \times n_{fp}}$ is a dense matrix. A convolutional residual block has the form $\mathcal{F}(q_s(t), \theta(t)) := \sigma(\mathrm{BN}(t, W(t)q_s(t) + b(t)))$, where $W(t)$ now stands for a sparse convolutional operator and BN denotes continuous-in-time batch-normalization [32, 44] from $\mathbb{R} \times \mathbb{R}^{n_{fp}}$ into $\mathbb{R}^{n_{fp}}$.

The objective function in (65) is defined such that the deviation, measured by the loss function $\hbar$ from $\mathbb{R}^{n_{out} \times n_{out}}$ into $\mathbb{R}$, between the desirable target c_s and predicted output $\hat{c}_s \in \mathbb{R}^{n_{out}}$ is minimized. Here, the predicted output is obtained as $\hat{c}_s := \mathcal{P}(W_T q_s(T) + b_T) \in \mathbb{R}^{n_{out}}$, where $\mathcal{P}$ denotes a hypothesis function from $\mathbb{R}^{n_{out}}$ into $\mathbb{R}^{n_{out}}$. The linear operators $W_T \in \mathbb{R}^{n_{out} \times n_{fp}}$, and $b_T \in \mathbb{R}^{n_{fp}}$ are used to perform an affine transformation of the extracted features $q_s(T)$, i.e., features obtained as an output of the dynamical system at time T. The regularizers $\mathcal{R}$ and $\mathcal{S}$ with parameters $\beta_1, \beta_2 > 0$ are defined as follows. A Tikhonov regularization is used to penalize the

magnitude of W_T and b_T, i.e., $\mathcal{S}(W_T, b_T) := \frac{\beta_1}{2}\|W_T\|_F^2 + \frac{\beta_1}{2}\|b_T\|^2$, where $\|\cdot\|_F$ denotes the Frobenius norm. For the time-dependent control parameters, we use $\mathcal{R}(\theta(t)) := \frac{\beta_1}{2}\|\theta(t)\|^2 + \frac{\beta_2}{2}\|\partial_t\theta(t)\|^2$, where the second term ensures that the parameters vary smoothly in time, see [29] for details.

To solve the problem (65) numerically, we follow the first-discretize-then-optimize approach. The discretization is performed using equidistant grid $0 = \tau_0 < \cdots < \tau_{K-1} = T$, consisting of K points. The states and controls are then approximated at a given time τ_k as $q_k \approx q(\tau_k)$, and $\theta_k \approx \theta(\tau_k)$, respectively. Note, each θ_k and q_k now corresponds to parameters and states associated with the k-th layer of the DNN. Our time-discretization uses the forward Euler scheme, which gives rise to the well-known ResNet architecture with identity skip connections [31]. Alternatively, one could employ more advanced, and perhaps more numerically stable, time integration schemes, see for example [29]. In the case of the explicit Euler scheme considered here, we ensure numerical stability by employing a sufficiently small time-step $\Delta_t = T/(K-1)$.

5.1 Implementation and algorithmic setup

Our implementation of ResNets is based on the deep-learning library Keras [15], while the MOFFTR framework is implemented using the library NumPy. We consider four different variants of first-order MOFFTR algorithms (i.e. $B_{\ell,i} = 0$ for all (ℓ, i)). The first variant employs divergent weights, specified by the MAXGI update rule given by (45) and(48) with $\mu = 0.1$. All other variants use AdaGrad-like weights, specified by (54) and (56), with $\mu \in \{0.1, 0.5, 0.9\}$. The selected update rules are used to update weights at all levels. The MOFFTR algorithms are implemented as a V-cycle with one pre-smoothing step and zero post-smoothing steps. For the ResNets with dense residual blocks, we perform 10 iterations on the lowest level, i.e., $i_1^{(\max)} = 10$ and employ $\kappa_R = 0.01$, and $\alpha = 5$. For the convolutional ResNets, we use $i_1^{(\max)} = 5$, $\kappa_R = 0.001$ and $\alpha = 25$. Parameters ν and ς are set as $\nu = 0.1$, and $\varsigma = 0.01$ for all numerical examples. Moreover, we take our discussion of Section 4.3 into account and do not impose the first-order coherence relation (7), as we apply the MOFFTR framework in stochastic settings where subsampling noise is present. In our realization of the MOFFTR algorithm, the subsampled derivatives are used at all (ℓ, i).

The hierarchy of objective functions $\{f\}_{\ell=1}^{r}$ required by the MOFFTR framework is obtained by discretizing the problem (65) with varying discretization parameter Δ_t. Each f_ℓ is then associated with a network of different depth. Unless stated otherwise, all numerical examples considered below take advantage of three levels, which we obtain using uniform refinement with a factor of two. The operators $\{P\}_{\ell=1}^{r-1}$ are constructed using piecewise linear interpolation in 1D (in time), see [18, 35] for details. Note that similar approaches for assembly of prolongation operators were also employed in the context of multilevel parameter initialization in [30, 13, 8]. We define the restriction operators $\{R\}_{\ell=1}^{r-1}$ from (2), choosing $\omega = \frac{1}{2}$ and $\omega = 1$ for networks with dense and convolutional residual blocks, respectively.

In order to assess the performance of the MOFFTR method, we provide a comparison with the single-level ASTR1 methods, which we obtain by calling the corresponding MOFFTR algorithm with $r = 1$. Our comparison also includes the baseline stochastic gradient (SGD) [46], ADAM [33] and AdaGrad[¶] methods. The learning rate of all methods is chosen by thorough hyper-parameter search, performed individually for each dataset, network, and batch size. More precisely, we consider learning rates from the set $\{0.0001, 0.00025, 0.0005, 0.00075, 0.001, 0.0025, 0.005, 0.0075, 0.01, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1.0\}$. The learning rate, which gave rise to the best generalization results (averaged over five independent runs), is then used in the presented numerical experiments. In the context of the MOFFTR method, the same learning rate is employed on all levels.

In what follows, the comparison between single and multilevel methods is performed by analyzing their dominant computational cost, i.e., that associated with gradient evaluations. Let C_r be a computational cost associated with an evaluation of the gradient on the uppermost level using a full dataset $\mathcal{D}$. Using the definition of C_r and taking advantage of the fact that the cost of the back-propagation scales linearly with the number of layers and the number of samples, we

[¶]AdaGrad is obtained by calling the MOFFTR algorithm with $r = 1$, weights given by (54) and $\mu = \frac{1}{2}$.

define the total computational cost C as follows:

$$C = \sum_{\ell=1}^{r} 2^{\ell-r} \sharp_\ell C_r, \tag{66}$$

where the scaling factor[‖] $2^{\ell-r}$ accounts for the difference between the cost associated with level ℓ and level r. The symbol $\sharp_\ell$ describes a number of gradient evaluations performed on a level ℓ using full dataset $\mathcal{D}$. For instance, if we evaluate gradient three times on level ℓ using n_b samples, then $\sharp_\ell = 3n_b/|\mathcal{D}|$.

All presented experiments were obtained using XC50 compute nodes (Intel Xeon E5-2690 v3 processor, NVIDIA Tesla P100 graphics card) of the Piz Daint supercomputer from the Swiss National Supercomputing Centre (CSCS).

5.2 Numerical examples

We investigate the convergence properties and the efficacy of the proposed MOFFTR algorithms using three numerical examples from the field of classification and regression.

5.2.1 Hyperspectral image segmentation using Indian Pines dataset

Our first example arises from soil segmentation using hyperspectral images provided by the Indian Pines dataset [2]. The input data was gathered by an Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) sensor and consist of 145×145 pixels with 200 spectral bands in the range from 400 to 2500 nm. Out of all pixels, only $10,249$ contain labeled data, which we split into $7,174$ for training and $3,075$ for validation. Each pixel is assigned to one of the 16 classes, representing the type of land cover, e.g., corn, soybean, etc. Segmentation is performed using ResNet with dense residual blocks of width 50, and the *ReLu* activation function σ. Parameters K, T, β_1, β_2 are set to $K = T = 3$ and $\beta_1 = \beta_2 = 10^{-3}$. Moreover, we employ the softmax hypothesis function together with the cross-entropy loss function, defined as $\hbar(\hat{c}_s, c_s) := c_s^T \log(\hat{c}_s)$. To train ResNet, we run variants of MOFFTR, ASTR1 and SGD methods without momentum. All methods are terminated as soon as $\text{acc}_{\text{train}} > 0.98$ or $\text{acc}_{\text{val}} > 0.98$. Termination also occurs as soon as $\sum_{i=1}^{15} (\text{acc}_{\text{train}})_e - (\text{acc}_{\text{train}})_{e-i} < 0.001$ or $\sum_{i=1}^{15} (\text{acc}_{\text{val}})_e - (\text{acc}_{\text{val}})_{e-i} < 0.001$, where $(\text{acc}_{\text{train/val}})_e$ is defined as the ratio between the number of correctly classified samples from the train/validation dataset and the total number of samples in the train/validation dataset for a given epoch e.

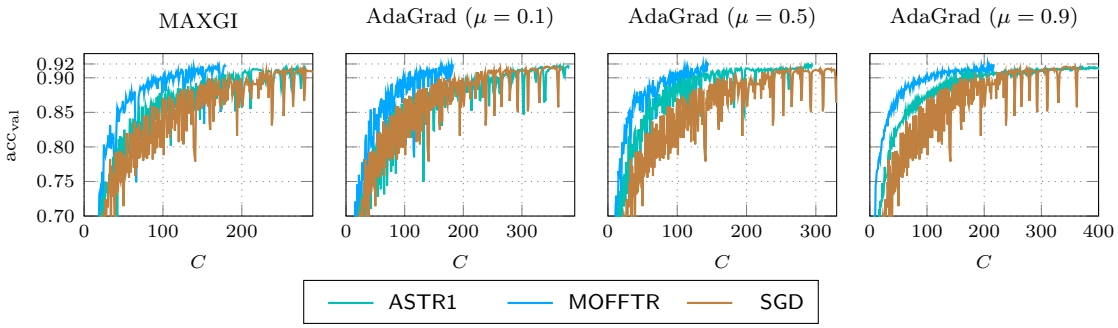

Figure 1: Indian Pines example: The validation accuracy as a function of the total computational cost C. The run with the highest validation accuracy is reported among 10 independent runs. The experiments were performed using a batch size of $1,024$.

Table 1 reports the total computational cost and validation accuracy required by all solution strategies, for increasing noise (i.e., decreasing batch size). As can be seen in this table, all

[‖] Uniform coarsening in 1D by a factor of two is assumed.

Method		Batch-size (batch-size/$\lvert\mathcal{D}\rvert$)							
		512 (6%)		1,024 (12%)		4,096 (50%)		8,100 (100%)	
		C	acc_{val}	C	acc_{val}	C	acc_{val}	C	acc_{val}
SGD		359	92.5%	369	91.7%	1,230	92.2%	3,093	92.1%
ASTR1	AdaGrad($\mu = 0.1$)	230	92.6%	379	91.8%	979	92.3%	2,748	92.4%
	AdaGrad($\mu = 0.5$)	143	92.4%	295	91.8%	853	92.3%	3,171	91.9%
	AdaGrad($\mu = 0.9$)	232	91.4%	404	91.6%	976	92.0%	2,626	91.7%
	MAXGI	195	92.4%	281	91.7%	884	92.2%	2,719	91.8%
MOFFTR	AdaGrad($\mu = 0.1$)	125	92.4%	183	91.7%	422	92.3%	1,683	92.3%
	AdaGrad($\mu = 0.5$)	113	92.1%	145	92.1%	412	92.2%	1,394	92.3%
	AdaGrad($\mu = 0.9$)	118	91.5%	217	91.9%	453	92.1%	1,760	91.9%
	MAXGI	96	92.3%	174	91.7%	334	91.9%	1,292	92.4%

Table 1: Indian Pines example: The total computational cost C and validation accuracy acc_{val} required to train ResNet. The best result in terms of validation accuracy is reported among 10 independent runs.

solution strategies achieve comparable validation accuracy for a given batch size. However, the computational cost of all variants of MOFFTR is smaller than that of their single-level counterparts and of the SGD method, see also Figure 1. Compared to ASTR1, the speedup factor fluctuates from 1.2 to 2.6. Compared to SGD method, the speedup is higher as it ranges from 1.7 to 3.7. Moreover, our results suggest that the speedup can be consistently observed even for small batch sizes. This empirically confirms that the proposed algorithmic framework retains enhanced convergence of the multilevel methods and at the same time is insensitive to the subsampling noise as a majority of OFFO methods.

5.2.2 Surrogate modelling of parametric neutron diffusion-reaction (NDR)

Our second example considers the construction of a surrogate model for a parametric neutron diffusion-reaction problem with spatially-varying coefficients and an external source. The goal is to construct a surrogate that can predict the average neutron flux for a given set of parameters. To this aim, we generate a dataset of $3,000$ samples, which we split into $2,600$ samples for training and 400 for testing. Following [43], the computational domain $\Omega = (0, 170)^2$ is heterogeneous and consists of four different material regions, denoted by $\Omega_1, \ldots, \Omega_4$ (see the left panel of Figure 2).

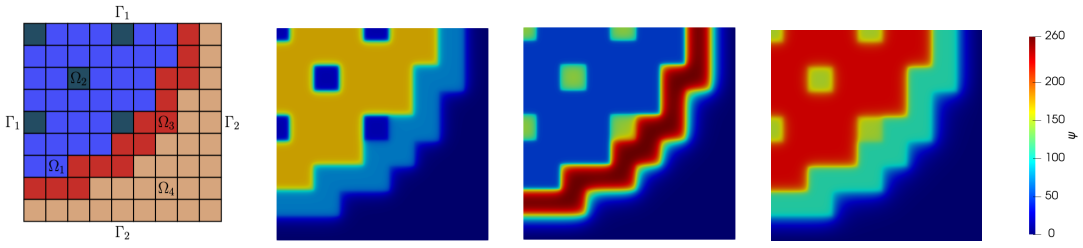

Figure 2: Left: Computational domain used for the creation of the NDR dataset. Each subdomain is illustrated by a different color. Middle-Right: Example of samples contained in the NDR dataset.

The strong form of the problem is given as

$$\begin{aligned} \nabla \cdot [D(x)\nabla\psi(x)] + \alpha(x)\nabla\psi(x) &= q(x), && \text{in } \Omega, \\ \psi(x) &= 0, && \text{on } \Gamma_1 := [0,170]\times\{1\}\cup\{0\}\times[0,170], \\ D(x)\nabla\psi(x)\cdot n(x) &= 0, && \text{on } \Gamma_2 := [0,170]\times\{0\}\cup\{1\}\times[0,170], \end{aligned} \tag{67}$$

where x denotes the spatial coordinates and ψ is the neutron flux from Ω to $\mathbb{R}$. Functions D, α, q are defined as $D(x) = \sum_{i=1}^{4} \mathbb{1}_{\Omega_i}(x) D_i$, $q(x) = \sum_{i=1}^{3} \mathbb{1}_{\Omega_i}(x) q_i$, and $\alpha(x) = \sum_{i=1}^{4} \mathbb{1}_{\Omega_i}(x) \alpha_i$, respectively ($\mathbb{1}_{\Omega_i}$ denotes the indicator function of the domain Ω). Problem (67) is then parametrized using 11 parameters, which we sample from a uniform distribution $\mathcal{U}(a, b)$, specified by lower (a) and upper (b) bounds. More precisely, diffusion coefficients $\{D_i\}_{i=1}^{3}$ are sampled from $\mathcal{U}(0.15, 0.6)$, while D_4 is sampled from $\mathcal{U}(0.2, 0.8)$. Reaction coefficients $\alpha_1, \ldots, \alpha_4$ take on values from $\mathcal{U}(0.0425, 0.17)$, $\mathcal{U}(0.065, 0.26)$, $\mathcal{U}(0.04, 0.16)$, $\mathcal{U}(0.005, 0.02)$, respectively. The values of sources $\{q_i\}_{i=1}^{3}$ are sampled from $\mathcal{U}(5, 20)$, while the value value of q_4 is set to 0. For each set of parameters/input features, we create a target $c_s = \int_\Omega \psi(x)\, dx / \int_\Omega dx$ by solving (67) numerically using the finite element method with a quadrilateral mesh (500 elements in both spatial dimensions).

To build the desired surrogate, we train ResNet with dense residual blocks of width 10, *tanh* activation function σ and parameters $T = K = 3$ and $\beta_1 = \beta_2 = 10^{-4}$. The identity hypothesis function and mean square loss functional are defined as $\hbar(\hat{c}_s, c_s) := \|c_s - \hat{c}_s\|_2^2$.

As common for regression problems, we train ResNet using variants of the MOFFTR, ASTR1, SGD and ADAM methods with momentum. The details of how to handle the momentum in the multilevel framework can be found in [35, Appendix A]. The training is performed for a fixed computational budget, i.e., all solution strategies terminate as soon as $C > C_{\max}$, where we set $C_{\max}$ to $2,000$.

Method		Batch-size (batch-size/$\lvert\mathcal{D}\rvert$)			
		256 (25%)		1,024 (100%)	
		$f_{\text{train}}(\times 10^3)$	$f_{\text{val}}(\times 10^3)$	$f_{\text{train}}(\times 10^3)$	$f_{\text{val}}(\times 10^3)$
SGD		0.61 ± 0.30	3.57 ± 1.44	1.87 ± 1.32	6.87 ± 1.09
ADAM		0.39 ± 0.23	5.77 ± 0.79	0.47 ± 0.50	5.35 ± 0.23
ASTR1	AdaGrad($\mu = 0.1$)	0.56 ± 0.13	5.41 ± 0.72	1.01 ± 0.37	5.72 ± 1.18
	AdaGrad($\mu = 0.5$)	0.47 ± 0.32	4.62 ± 2.71	0.79 ± 0.12	7.04 ± 4.93
	AdaGrad($\mu = 0.9$)	0.54 ± 0.17	7.43 ± 2.48	0.86 ± 0.10	4.93 ± 2.56
	MAXGI	0.53 ± 0.35	5.67 ± 1.23	1.04 ± 0.29	4.98 ± 1.28
MOFFTR	AdaGrad($\mu = 0.1$)	0.48 ± 0.29	0.89 ± 0.31	0.48 ± 0.73	1.02 ± 0.42
	AdaGrad($\mu = 0.5$)	0.39 ± 0.31	0.95 ± 0.29	0.63 ± 0.45	1.29 ± 0.53
	AdaGrad($\mu = 0.9$)	0.45 ± 0.25	1.06 ± 0.36	0.62 ± 0.41	1.36 ± 0.49
	MAXGI	0.51 ± 0.14	0.91 ± 0.30	0.73 ± 0.47	1.38 ± 0.45

Table 2: NDR example: The mean and standard deviation of the value of train (f_{train}) and validation (f_{val}) loss function achieved by training ResNet for $2,000$ epochs. The statistics is obtained from 10 independent runs.

We observe in Table 2 that all solution strategies, except SGD, achieve comparable values of the training loss (f_{train}). Interestingly, we also notice that all variants of the MOFFTR algorithm generalize better, i.e., reach a lower value of the validation loss (f_{val}) than all single-level methods. This phenomenon can be also observed in Figure 3. The results presented in Table 2 suggest that the MOFFTR methods preserve good generalization properties in the presence of noise, a property of particular interest in surrogate modeling and other scientific applications that require reliable solutions.

5.2.3 Image classification using the SVNH dataset

Our last numerical example is associated with an image classification using SVNH dataset [42]. Each image is represented by 32×32 pixels and contains overlapping digits from 0 to 9. This dataset consists of $99,289$ samples, from which $73,257$ are used for training and $26,032$ for testing purposes. We pre-process all samples by standardizing the images, so that pixel values lie in the range $[0, 1]$, and by subtracting the mean from each pixel. In addition, we use standard data augmentation techniques, in particular image rotation, horizontal and vertical shift, and horizontal flip. The image classification is performed using ResNet with convolutional residual blocks (32

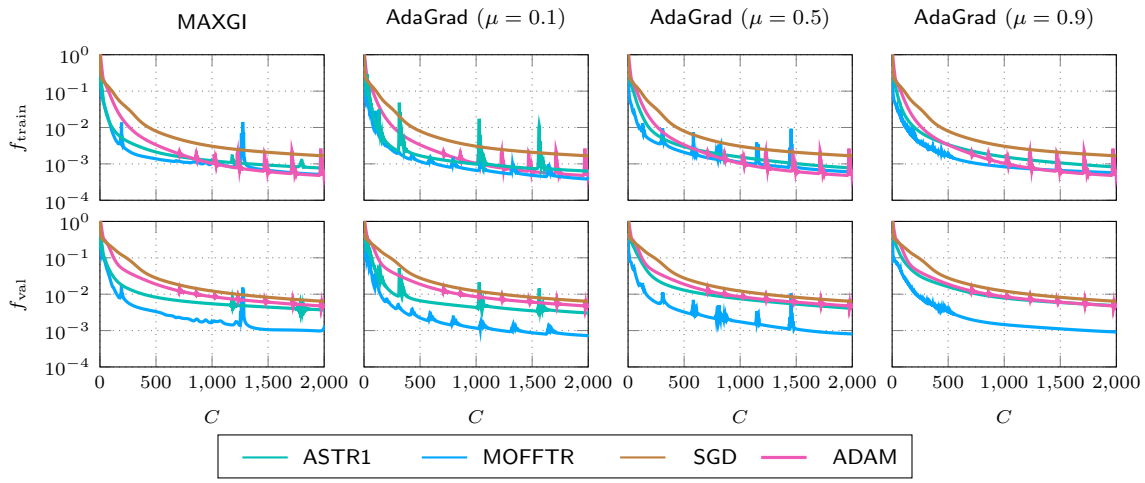

Figure 3: NDR example: The history of train (f_{train}) and validation (f_{val}) loss function obtained while training ResNet for $2,000$ epochs with batch size $1,024$ (full dataset). The result with lowest f_{val} among 10 independent runs is reported.

filters), *ReLu* activation function and parameters $T = 3$, $K = 5$, $\beta_1 = 10^{-3}$, $\beta_2 = 0.005$. Moreover, we use the softmax hypothesis function with the cross-entropy loss function.

The training of ResNets is performed using a batch size of 512 and the same stopping criterion as that used for the Indian Pines example in Section 5.2.1. For this experiment, we consider the MOFFTR, ASTR1 and SGD methods without momentum. Table 3 reports the computational cost and validation accuracy achieved by all methods. The study is performed with respect to an increasing number of refinement levels**.

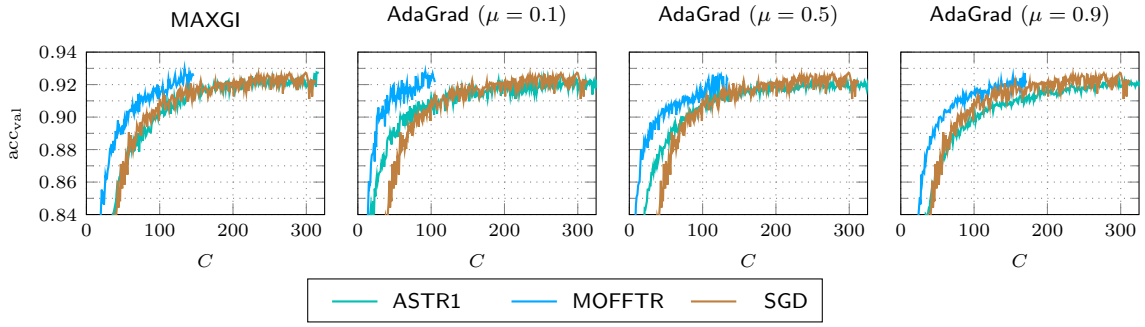

Figure 4: SVNH example: The validation accuracy as a function of total computational cost obtained while training ResNet with 17 residual block (three refinement levels). The run with the highest validation accuracy is selected among 10 independent runs.

These results suggest that increasing the network depth enhances its representation capacity, which in turn justifies a higher computational cost. We also notice that all variants of the MOFFTR method require lower computational cost than single-level solution strategies while achieving comparable accuracy. The obtained computational gains vary from the factor of 1.7 to 3.4. This can be observed independently of the refinement level, which indicates that the MOFFTR algorithm has a clear potential to speed up the training of large-scale networks, such as the ones used in real-life applications. We also note that, among all variants of the MOFFTR framework, the choice of the MAXGI weights yields higher validation accuracy than that obtained with the AdaGrad-like

**The number of levels utilized by the MOFFTR algorithms increases linearly with refinement level.

Method		Levels (residual blocks)					
		2 (9)		3 (17)		4 (33)	
		C	acc_{val}	C	acc_{val}	C	acc_{val}
SGD		280	92.4%	313	92.7%	336	92.9%
ASTR1	AdaGrad($\mu = 0.1$)	325	92.4%	361	92.6%	341	92.8%
	AdaGrad($\mu = 0.5$)	334	92.3%	333	92.4%	328	92.7%
	AdaGrad($\mu = 0.9$)	351	92.1%	358	92.4%	369	92.6%
	MAXGI	313	92.3%	317	92.7%	325	92.8%
MOFFTR	AdaGrad($\mu = 0.1$)	134	92.3%	105	92.7%	139	92.9%
	AdaGrad($\mu = 0.5$)	148	92.2%	135	92.6%	144	92.8%
	AdaGrad($\mu = 0.9$)	169	92.1%	171	92.6%	151	92.8%
	MAXGI	158	92.5%	146	92.9%	143	93.1%

Table 3: SVNH example: The total computational cost C and validation accuracy acc_{val}, which were required to train ResNet with increasing depth (refinement level). The best result in terms of validation accuracy is reported among 10 independent runs.

variants. We finally observe that, among all AdaGrad-like variants, the configuration with $\mu = 0.1$ achieves the highest validation accuracy as well as the lowest computational cost. The fact that a similar observation can be made also for the corresponding single-level ASTR1 methods suggests that exploring values of μ other than the traditional $\mu = 0.5$ might be beneficial in practice.

6 Conclusion

We have presented a class of multilevel algorithms for unconstrained minimization which do not require the computation of the objective function's value. We have also shown that the performance of algorithms of the class is competitive in the presence of noise-induced by subsampling in the context of deep neural network training. The convergence theory of two interesting subclasses has been analyzed and shown to match the state of the art. Our experiments also indicate that, although currently not covered by our theory, the benefits of the multilevel approach are preserved when momentum is added to the framework. The authors are well aware that only continued experimentation will reveal the true practical value of the present proposal, but they note that the first numerical tests are encouraging.

Acknowledgements

A. Kopaničáková gratefully acknowledges support of the Swiss National Science Foundation through the project "Multilevel training of DeepONets — multiscale and multiphysics applications" (grant no. 206745), and partial support of Platform for Advanced Scientific Computing (PASC) under the project EXATRAIN. Ph. L. Toint acknowledges the continued and friendly partial support of ANITI.

References

[1] N. M. Alexandrov and R. L. Lewis. An overview of first-order model management for engineering optimization. Optimization and Engineering, 2:413–430, 2001.

[2] M. F. Baumgardner, L. L. Biehl, and D. A. Landgrebe. 220 band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pine TeSite 3. Purdue University Research Repository, 10(7):991, 2015.

[3] A. Beck and N. Hallak. Optimization problems involving group sparsity terms. Mathematical Programming, 2018. online.

[4] A. Borzi and K. Kunisch. A globalisation strategy for the multigrid solution of elliptic optimal control problems. Optimization Methods and Software, 21(3):445–459, 2006.

[5] A. Bouaricha and R. B. Schnabel. Tensor methods for large sparse systems of nonlinear equations. Mathematical Programming, 82(3):377–412, 1998.

[6] W. L. Briggs, V. E. Henson, and S. F. McCormick. A Multigrid Tutorial. SIAM, Philadelphia, USA, 2nd edition, 2000.

[7] C. Cartis, N. I. M. Gould, and Ph. L. Toint. Evaluation complexity of algorithms for nonconvex optimization. Number 30 in MOS-SIAM Series on Optimization. SIAM, Philadelphia, USA, June 2022.

[8] B. Chang, L. Meng, E. Haber, F. Tung, and D. Begert. Multi-level residual networks from dynamical systems view. arXiv:1710.10348, 2017.

[9] I. Chatzigeorgiou. Bounds on the Lambert function and their application to the outage analysis of user cooperation. IEEE Communications Letters, 17(8):1505–1508, 2013.

[10] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. LANCELOT: a Fortran package for large-scale nonlinear optimization (Release A). Number 17 in Springer Series in Computational Mathematics. Springer Verlag, Heidelberg, Berlin, New York, 1992.

[11] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Trust-Region Methods. Number 1 in MOS-SIAM Optimization Series. SIAM, Philadelphia, USA, 2000.

[12] R. M. Corless, G. H. Gonnet, D. E. Hare, D. J. Jeffrey, and D. E. Knuth. On the Lambert W function. Advances in Computational Mathematics, 5:329—359, 1996.

[13] E. C. Cyr, S. Günther, and J. B. Schroder. Multilevel initialization for layer-parallel deep neural network training. International Journal of Computing and Visualization in Science and Engineering, 2021.

[14] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 12, July 2011.

[15] F. Chollet et al. Keras, 2015. `https://keras.io`.

[16] M. Fisher. Minimization algorithms for variational data assimilation. In Recent Developments in Numerical Methods for Atmospheric Modelling, pages 364–385, Reading, UK, 1998. European Center for Medium-Range Weather Forecasts.

[17] K. Fujisawa, M. Kojima, and K. Nakata. Exploiting sparsity in primal-dual interior-point methods for semidefinite programming. Mathematical Programming, Series B, 79(1–3):235–253, 1997.

[18] L. Gaedke-Merzhäuser, A. Kopaničáková, and R. Krause. Multilevel minimization for deep residual networks. In ESAIM. Proceedings and Surveys. 71:131-144, 2021.

[19] D. M. Gay. Automatically finding and exploiting partially separable structure in nonlinear programming problems. Technical report, Bell Laboratories, Murray Hill, New Jersey, USA, 1996.

[20] E. Gelman and J. Mandel. On multilevel iterative methods for optimization problems. Mathematical Programming, 48(1):1–17, 1990.

[21] D. Goldfarb and S. Wang. Partial-update Newton methods for unary, factorable and partially separable optimization. SIAM Journal on Optimization, 3(2):383–397, 1993.

[22] S. Gratton, S. Jerad, and Ph. L. Toint. Convergence properties of an objective-function-free optimization regularization algorithm, including an $\mathcal{O}(\epsilon^{-3/2})$ complexity bound. arXiv:2203.09947, 2022.

[23] S. Gratton, S. Jerad, and Ph. L. Toint. First-order objective-function-free optimization algorithms and their complexity. arXiv:2203.01757, 2022.

[24] S. Gratton, S. Jerad, and Ph. L. Toint. Parametric complexity analysis for a class of first-order Adagrad-like algorithms. arXiv:2203.01647, 2022.

[25] S. Gratton, M. Mouffe, A. Sartenaer, Ph. L. Toint, and D. Tomanos. Numerical experience with a recursive trust-region method for multilevel nonlinear bound-constrained optimization. Optimization Methods and Software, 25(3):359 – 386, 2010.

[26] S. Gratton, A. Sartenaer, and Ph. L. Toint. Recursive trust-region methods for multiscale nonlinear optimization. SIAM Journal on Optimization, 19(1):414–444, 2008.

[27] S. Gratton and Ph. L. Toint. Approximate invariant subspaces and quasi-Newton optimization methods. Optimization Methods and Software, 25(4):507–529, 2010.

[28] A. Griewank and Ph. L. Toint. On the unconstrained optimization of partially separable functions. In M. J. D. Powell, editor, Nonlinear Optimization 1981, pages 301–312, London, 1982. Academic Press.

[29] E. Haber and L. Ruthotto. Stable architectures for deep neural networks. Inverse Problems, 34(1):014004, 2017.

[30] E. Haber, L. Ruthotto, E. Holtham, and S.-H. Jun. Learning across scales—multiscale methods for convolution neural networks. In Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[31] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In European conference on computer vision, pages 630–645, Heidelberg, Berlin, New York, 2016. Springer Verlag.

[32] S. Ioffe and Ch. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167, 2015.

[33] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In Proceedings in the International Conference on Learning Representations (ICLR), 2015.

[34] A. Kopaničáková. On the use of hybrid coarse-level models in multilevel minimization methods. arXiv:2211.15078, 2022.

[35] A. Kopaničáková and R. Krause. Globally convergent multilevel training of deep residual networks. SIAM Journal on Scientific Computing, 0:S254–S280, 2022.

[36] R. Kornhuber. Adaptive monotone multigrid methods for some non-smooth optimization problems. In R. Glowinski, J. Périaux, Z. Shi, and O. Widlund, editors, Domain Decomposition Methods in Sciences and Engineering, pages 177–191. J. Wiley and Sons, Chichester, England, 1997.

[37] J. B. Lasserre. Convergent semidefinite relaxation in polynomial optimization with sparsity. Technical report, LAAS-CNRS, 7, avenue du Colonel Roche, 31077 Toulouse, France, November 2005.

[38] M. Lewis and S. G. Nash. Practical aspects of multiscale optimization methods for VLSICAD. In Jason Cong and Joseph R. Shinnerl, editors, Multiscale Optimization and VLSI/CAD, pages 265–291, Dordrecht, The Netherlands, 2002. Kluwer Academic Publishers.

[39] M. Lewis and S. G. Nash. Model problems for the multigrid optimization of systems governed by differential equations. SIAM Journal on Scientific Computing, 26(6):1811–1837, 2005.

[40] J. Mareček, P. Richtárik, and M. Takáč. Distributed block coordinate descent for minimizing partially separable functions. Technical report, Department of Mathematics and Statistics, University of Edinburgh, Edinburgh, Scotland, 2014.

[41] S. G. Nash. A multigrid approach to discretized optimization problems. Optimization Methods and Software, 14:99–116, 2000.

[42] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A.Y. Ng. Reading digits in natural images with unsupervised feature learning, 2011.

[43] Z. M. Prince and J. C. Ragusa. Parametric uncertainty quantification using proper generalized decomposition applied to neutron diffusion. International Journal for Numerical Methods in Engineering, 119(9):899–921, 2019.

[44] A. F. Queiruga, N. B. Erichson, D. Taylor, and M. W. Mahoney. Continuous-in-depth neural networks. arXiv:2008.02389, 2020.

[45] S. Reddi, S. Kale, and S. Kumar. On the convergence of Adam and beyond. In Proceedings in the International Conference on Learning Representations (ICLR), 2018.

[46] H. Robbins and S. Monro. A stochastic approximation method. The Annals of Mathematical Statistics, 22(3):400—407, 1951.

[47] T. Tieleman and G. Hinton. Lecture 6.5-RMSPROP. COURSERA: Neural Networks for Machine Learning, 2012.

[48] Ph. L. Toint. A note on sparsity exploiting quasi-Newton methods. Mathematical Programming, 21(2):172–181, 1981.

[49] Ph. L. Toint. Towards an efficient sparsity exploiting Newton method for minimization. In I. S. Duff, editor, Sparse Matrices and Their Uses, pages 57–88, London, 1981. Academic Press.

[50] E. Weinan. A proposal on machine learning via dynamical systems. Communications in Mathematics and Statistics, 5(1):1–11, 2017.

[51] Y. Yuan. Recent advances in trust region algorithms. Mathematical Programming, Series A, 151(1):249–281, 2015.

[52] D. Zhou, J. Chen, Y. Tang, Z. Yang, Y. Cao, and Q. Gu. On the convergence of adaptive gradient methods for nonconvex optimization. arXiv:2080.05671, 2020.

A Proof of Theorem 3.8

We first recall a useful technical result.

Lemma A.1 [23, Lemma 3.5] Consider and arbitrary $i \in \{1, \ldots, n\}$ and suppose that there exists a j_ς such that

$$\min\left[\frac{g_{i,j}^2}{\varsigma_i}, \frac{g_{i,j}^2}{v_{i,j}}\right] \leq \varsigma_i \quad \text{for} \quad j \geq j_\varsigma. \tag{68}$$

Then

$$\min\left[\frac{g_{i,j}^2}{\varsigma_i}, \frac{g_{i,j}^2}{v_{i,j}}\right] \geq \frac{g_{i,j}^2}{2\varsigma_i} \quad \text{for} \quad j \geq j_\varsigma. \tag{69}$$

Proof of Theorem 3.8 We obtain from (43) and (6) that, for $i \geq i_\vartheta$,

$$f(x_{r,i_\vartheta}) - f(x_{r,i+1}) \geq \frac{1}{2} \sum_{k=i_\vartheta+1}^{i} \sum_{j=1}^{n} \frac{g_{r,k,j}^2}{w_{r,k,j}} \left[\beta_{1,r} - \frac{\beta_{2,r} + \frac{1}{2}\alpha^2 L}{w_{\ell,k,j}}\right].$$

Moreover, (45) and (52) also ensure that, since $i \geq i_\vartheta$,

$$\beta_{1,r} - \frac{\beta_{2,r} + \frac{1}{2}\alpha^2 L}{w_{\ell,k,j}} \geq \vartheta$$

for all $j \in \{1, \ldots, n\}$, so that

$$\begin{aligned}
f(x_{r,i_\vartheta}+1) - f(x_{r,i+1}) &\geq \frac{\vartheta}{2} \sum_{k=i_\vartheta+1}^{i} \sum_{j=1}^{n} \frac{g_{r,k,j}^2}{w_{r,k,j}} \\
&\geq \frac{\vartheta}{2} \sum_{k=i_\vartheta+1}^{i} \sum_{j=1}^{n} \frac{g_{r,k,j}^2}{\max[\varsigma_j, v_{r,k}](k+1)^\mu} \\
&\geq \frac{\vartheta}{2(i+1)^\mu} \sum_{k=i_\vartheta+1}^{i} \sum_{j=1}^{n} \min\left[\frac{g_{r,k,j}^2}{\varsigma_j}, \frac{g_{r,k,j}^2}{v_{r,k}}\right] \\
&\geq \frac{\vartheta(i-i_\vartheta)}{2(i+1)^\mu} \min_{k\in\{i_\vartheta,\ldots,i\}} \left(\sum_{j=1}^{n} \min\left[\frac{g_{r,k,j}^2}{\varsigma_j}, \frac{g_{r,k,j}^2}{v_{r,k}}\right]\right).
\end{aligned}$$

But, using (50) in Lemma 3.7,

$$f(x_0) - f(x_{i_\vartheta+1}) = f(x_{r,0}) - f(x_{i_\vartheta+1}) \geq -n(\beta_{2,r} + \tfrac{1}{2}\alpha^2 L) \sum_{k=0}^{i_\vartheta} a(k)^2. \tag{70}$$

Combining these two last inequalities with AS.3 then gives that

$$\begin{aligned}
\min_{k\in\{i_\vartheta,\ldots,i\}} \left(\sum_{j=1}^{n} \min\left[\frac{g_{r,k,j}^2}{\varsigma_j}, \frac{g_{r,k,j}^2}{v_{r,k}}\right]\right) &\leq \frac{2(i+1)^\mu}{\vartheta(i-i_\vartheta)}\Big(f(x_{i_\vartheta+1}) - f(x_{i+1})\Big) \\
&= \frac{2(i+1)^\mu}{\vartheta(i-i_\vartheta)}\Big(f(x_0) - f(x_{i+1}) + f(x_{i_\vartheta+1}) - f(x_0)\Big) \\
&\leq \frac{2(i+1)^\mu}{\vartheta(i-i_\vartheta)}\Big(\Gamma_0 + n(\beta_{2,r} + \tfrac{1}{2}\alpha^2 L) \sum_{k=0}^{i_\vartheta} a(k)^2\Big)
\end{aligned}$$

from which we obtain that there exists a subsequence $\{i_t\} \subseteq \{i\}_{i_\vartheta}^{\infty}$ such that

$$\sum_{j=1}^{n} \min\left[\frac{g_{r,i_t,j}^2}{\varsigma_j}, \frac{g_{r,i_t,j}^2}{v_{r,i_t}}\right] \leq \frac{2(i_t+1)^\mu}{\vartheta(i_t-i_\vartheta)} \left[\Gamma_0 + n(\beta_{2,r} + \tfrac{1}{2}\alpha^2 L) \sum_{k=0}^{i_\vartheta} a(k)^2\right]. \tag{71}$$

Now,

$$\frac{(i_t+1)^\mu}{i_t - i_\vartheta} < \frac{2^\mu i_t^\mu}{i_t - i_\vartheta} < \frac{2i_t^\mu}{i_t - i_\vartheta} = \frac{2i_t^\mu i_t}{(i_t - i_\vartheta)i_t} = \frac{i_t}{i_t - i_\vartheta} \cdot \frac{2}{i_t^{1-\mu}} \leq \frac{2(i_\vartheta+1)}{i_t^{1-\mu}}, \tag{72}$$

where we used the facts that $\mu < 1$ and that $\frac{i_t}{i_t - i_\vartheta}$ is a decreasing function for $i_t \geq i_\vartheta + 1$. As a consequence, we obtain from (71) that

$$\sum_{j=1}^{n} \min\left[\frac{g_{r,i_t,j}^2}{\varsigma_j}, \frac{g_{r,i_t,j}^2}{v_{r,i_t}}\right] \leq \frac{4(i_\vartheta+1)}{\vartheta i_t^{1-\mu}} \left[\Gamma_0 + n(\beta_{2,r} + \tfrac{1}{2}\alpha^2 L)\sum_{k=0}^{i_\vartheta} a(k)^2\right].$$

If we now define

$$i_\varsigma \stackrel{\rm def}{=} \left(\frac{4(i_\vartheta+1)\left[\Gamma_0 + \frac{1}{2}n(\beta_{2,r} + \frac{1}{2}\alpha^2 L)\sum_{k=0}^{i_\vartheta} a(k)^2\right]}{\vartheta \varsigma_{\min}}\right)^{\frac{1}{1-\mu}},$$

we see that, for all $i_t \geq i_\varsigma$,

$$\min\left[\frac{g_{r,i_t,j}^2}{\varsigma_j}, \frac{g_{r,i_t,j}^2}{v_{r,i_t}}\right] \leq \varsigma_{\min}.$$

We then apply Lemma A.1 to deduce from (71) that, for all $i_t \geq i_\varsigma$,

$$\sum_{j=1}^{n} \frac{g_{r,i_t,j}^2}{2\varsigma_j} \leq \frac{2(i_t+1)^\mu}{\vartheta(i_t - i_\vartheta)} \left[\Gamma_0 + n(\beta_{2,r} + \tfrac{1}{2}\alpha^2 L)\sum_{k=0}^{i_\vartheta} a(k)^2\right]$$

and therefore, because $\varsigma_j \leq 1$, that

$$\|g_{r,i_t}\|^2 \leq \frac{(i_t+1)^\mu}{i_t - i_\vartheta}\left(\frac{2}{\vartheta}\right)\left[\Gamma_0 + n(\beta_{2,r} + \tfrac{1}{2}\alpha^2 L)\sum_{k=0}^{i_\vartheta} a(k)^2\right]$$

for all $i_t \geq i_\varsigma$. This then gives (51), the last inequality following from (72) and (6). □

B Proof of Theorem 3.9

Again, we first recall a useful technical lemma.

Lemma B.1 [23, Lemma 3.1] Let $\{a_k\}_{k\geq 0}$ be a non-negative sequence, $\alpha > 0$, $\xi > 0$ and define, for each $k \geq 0$, $b_k = \sum_{t=0}^{k} a_t$. Then

$$\sum_{k=0}^{i} \frac{a_k}{(\xi + b_k)^\alpha} \leq \begin{cases} \frac{1}{(1-\alpha)}((\xi + b_i)^{1-\alpha} - \xi^{1-\alpha}) & \text{if} \quad \alpha \neq 1, \\ \log\left(\frac{\xi + b_i}{\xi}\right) & \text{if} \quad \alpha = 1. \end{cases} \tag{73}$$

Proof of Theorem 3.9 We first note that (54) implies that

$$\varsigma^\mu \leq \max_{j\in\{1,\ldots,n\}} w_{r,i,j} \leq \left(\varsigma + \sum_{k=0}^{i} \|g_{r,k}\|^2\right)^\mu \tag{74}$$

for all $i \geq 0$. We also deduce from (6) and (43) in Lemma 3.6 with $\ell = r$ that, for $i \geq 0$,

$$f(x_{i+1}) \leq f(x_0) - \beta_{1,r}\sum_{k=0}^{i} \frac{\|g_{r,k}\|^2}{\max\limits_{j\in\{1,\ldots,n\}} w_{r,k,j}} + (\beta_{2,r} + \tfrac{1}{2}L)\sum_{j=1}^{n}\sum_{k=0}^{i} \frac{g_{r,k,j}^2}{w_{r,k,j}^2}. \tag{75}$$

Case (i). Suppose first that $\mu \in (0, \frac{1}{2})$. For each $j \in \{1, \ldots, n\}$, we then apply (73) in Lemma B.1 with $a_k = g_{r,k,j}^2$, $\xi = \varsigma$ and $\alpha = 2\mu < 1$, and obtain from (13) and (54) that,

$$\sum_{k=0}^{i} \frac{g_{r,k,j}^2}{w_{r,k,j}^2} \leq \frac{1}{1-2\mu} \left[\left(\varsigma + \sum_{k=0}^{i} g_{r,k,j}^2 \right)^{1-2\mu} - \varsigma^{1-2\mu} \right] \leq \frac{1}{1-2\mu} \left(\varsigma + \sum_{k=0}^{i} g_{r,k,j}^2 \right)^{1-2\mu}. \tag{76}$$

Now substituting this bound in (75) and using AS.3 gives that

$$\beta_{1,r} \sum_{k=0}^{i} \frac{\|g_{r,k}\|^2}{\max\limits_{j \in \{1,\ldots,n\}} w_{r,k,j}} \leq \Gamma_0 + \frac{n(\beta_{2,r} + \frac{1}{2}L)}{1-2\mu} \left(\varsigma + \sum_{k=0}^{i} \|g_{r,k}\|^2 \right)^{1-2\mu}. \tag{77}$$

Suppose now that

$$\sum_{k=0}^{i} \|g_{r,k}\|^2 \geq \max \left\{ \varsigma, \frac{1}{2} \left[\frac{(1-2\mu)\Gamma_0}{n(\beta_{2,r} + \frac{1}{2}L)} \right]^{\frac{1}{1-2\mu}} \right\}, \tag{78}$$

implying

$$\varsigma + \sum_{k=0}^{i} \|g_{r,k}\|^2 \leq 2 \sum_{k=0}^{i} \|g_{r,k}\|^2 \ \text{ and } \ \Gamma_0 \leq \frac{n(\beta_{2,r} + \frac{1}{2}L)}{1-2\mu} \left(2 \sum_{k=0}^{i} \|g_{r,k}\|^2 \right)^{1-2\mu}.$$

Then, using (77) and (74),

$$\frac{\beta_{1,r}}{2^\mu \left[\sum_{k=0}^{i} \|g_{r,k}\|^2 \right]^\mu} \sum_{k=0}^{i} \|g_{r,k}\|^2 \leq \frac{2^{2(1-\mu)} n(\beta_{2,r} + \frac{1}{2}L)}{1-2\mu} \left(\sum_{k=0}^{i} \|g_{r,k}\|^2 \right)^{1-2\mu}.$$

Solving this inequality for $\sum_{k=0}^{i} \|g_{r,k}\|^2$ and using the fact that $2^{2(1-\mu)} < 4$ gives that

$$\sum_{k=0}^{i} \|g_{r,k}\|^2 < \left[\frac{4n(\beta_{2,r} + \frac{1}{2}L)}{\beta_{1,r}(1-2\mu)} \right]^{\frac{1}{\mu}}$$

and therefore

$$\operatorname*{average}_{k \in \{0,\ldots,i\}} \|g_{r,k}\|^2 < \left[\frac{4n(\beta_{2,r} + \frac{1}{2}L)}{\beta_{1,r}(1-2\mu)} \right]^{\frac{1}{\mu}} \cdot \frac{1}{i+1}. \tag{79}$$

Alternatively, if (78) fails, then

$$\operatorname*{average}_{k \in \{0,\ldots,i\}} \|g_{r,k}\|^2 < \max \left\{ \varsigma, \frac{1}{2} \left[\frac{(1-2\mu)\Gamma_0}{n(\beta_{2,r} + \frac{1}{2}L)} \right]^{\frac{1}{1-2\mu}} \right\} \cdot \frac{1}{i+1}. \tag{80}$$

Combining (79), (80) and (6) gives (57) for $0 < \mu < \frac{1}{2}$.

Case (ii). Let us now consider the case where $\mu = \frac{1}{2}$. For each $j \in \{1, \ldots, n\}$, we apply (73) in Lemma B.1 with $a_k = g_{r,k,j}^2$, $\xi = \varsigma$ and $\alpha = 2\mu = 1$ and obtain that,

$$\sum_{j=1}^{n} \sum_{k=0}^{i} \frac{g_{r,k,j}^2}{w_{r,k,j}^2} \leq \sum_{j=1}^{n} \log \left(\frac{1}{\varsigma} \left(\varsigma + \sum_{k=0}^{i} g_{r,k,j}^2 \right) \right) \leq n \log \left(1 + \frac{1}{\varsigma} \sum_{k=0}^{i} \|g_{r,k}\|^2 \right)$$

and substituting this bound in (75) then gives that

$$\beta_{1,r} \sum_{k=0}^{i} \frac{\|g_{r,k}\|^2}{\max\limits_{j \in \{1,\ldots,n\}} w_{r,k,j}} \leq \Gamma_0 + \tfrac{1}{2} n(\beta_{2,r} + \tfrac{1}{2}L) \log \left(1 + \frac{1}{\varsigma} \sum_{k=0}^{i} \|g_{r,k}\|^2 \right).$$

Suppose now that

$$\sum_{k=0}^{i} \|g_{r,k}\|^2 \geq \max\left[\varsigma, e^{\frac{2\Gamma_0}{n(\beta_{2,r}+\frac{1}{2}L)}}\right], \tag{81}$$

implying that

$$\varsigma + \sum_{k=0}^{i} \|g_{r,k}\|^2 \leq 2\sum_{k=0}^{i} \|g_{r,k}\|^2 \;\text{ and }\; \Gamma_0 \leq \tfrac{1}{2}n(\beta_{r,2}+\tfrac{1}{2}L)\log\left(\frac{2}{\varsigma}\sum_{k=0}^{i} \|g_{r,k}\|^2\right).$$

Using (74) for $\mu = \frac{1}{2}$, we obtain then that

$$\frac{\beta_{1,r}}{\sqrt{2}\sqrt{\sum_{k=0}^{i} \|g_{r,k}\|^2}} \sum_{k=0}^{i} \|g_{r,k}\|^2 \leq n(\beta_{r,k}+\tfrac{1}{2}L)$$

and thus that

$$\beta_{r,1}\sqrt{\sum_{k=0}^{i} \|g_{r,k}\|^2} \leq 2\sqrt{2}\max[3\beta_{r,1}, n(\beta_{r,2}+\tfrac{1}{2}L)]\log\left(\sqrt{\frac{2}{\varsigma}\sum_{k=0}^{i} \|g_{r,k}\|^2}\right). \tag{82}$$

Now define

$$\gamma_1 \stackrel{\rm def}{=} \beta_{r,1}\sqrt{\frac{\varsigma}{2}}, \quad \gamma_2 \stackrel{\rm def}{=} 2\sqrt{2}\max[\tfrac{3}{2}\beta_{r,1}, n(\beta_{r,2}+\tfrac{1}{2}L)] \quad \text{and} \quad u \stackrel{\rm def}{=} \sqrt{\frac{2}{\varsigma}\sum_{k=0}^{i} \|g_{r,k}\|^2} \tag{83}$$

and observe that $\gamma_2 > 3\gamma_1$ by construction, because $\varsigma \leq 1$. The inequality (82) can then be rewritten as

$$\gamma_1 u \leq \gamma_2 \log(u). \tag{84}$$

Let us denote by $\psi(u) \stackrel{\rm def}{=} \gamma_1 u - \gamma_2 \log(u)$. Since $\gamma_2 > 3\gamma_1$, the equation $\psi(u) = 0$ admits two roots $u_1 \leq u_2$ and (84) holds for $u \in [u_1, u_2]$. The definition of u_2 then gives that

$$\log(u_2) - \frac{\gamma_1}{\gamma_2}u_2 = 0.$$

Setting $z = -\frac{\gamma_1}{\gamma_2}u_2$, we obtain that

$$ze^z = -\frac{\gamma_1}{\gamma_2}.$$

Thus $z = W_{-1}(-\frac{\gamma_1}{\gamma_2}) < 0$, where W_{-1} is the second branch of the Lambert function defined over $[-\frac{1}{e}, 0)$. As $-\frac{\gamma_1}{\gamma_2} \geq -\frac{1}{3}$, z is well defined and thus

$$u_2 = -\frac{\gamma_2}{\gamma_1}\,z = -\frac{\gamma_2}{\gamma_1}\,W_{-1}\left(-\frac{\gamma_1}{\gamma_2}\right) > 0 = -\psi\,W_{-1}\left(-\frac{1}{\psi}\right) > 0,$$

where $\psi = \frac{\gamma_2}{\gamma_1}$ is given by (59). As a consequence, we deduce from (84) and (83) that

$$\sum_{k=0}^{i} \|g_{r,k}\|^2 = \frac{\varsigma}{2}\,u_2^2 = \frac{\varsigma\psi^2}{2}\,\left|W_{-1}\left(-\frac{1}{\psi}\right)\right|^2$$

and

$$\underset{k\in\{0,\ldots,i\}}{\text{average}} \|g_{r,k}\|^2 \leq \frac{\varsigma\psi^2}{2}\,\left|W_{-1}\left(-\frac{1}{\psi}\right)\right|^2 \cdot \frac{1}{i+1}. \tag{85}$$

If (81) does not hold, we have that

$$\underset{k\in\{0,\ldots,i\}}{\text{average}} \|g_{r,k}\|^2 < \max\left[\varsigma, e^{\frac{2\Gamma_0}{n(\beta_{2,r}+\frac{1}{2}L)}}\right] \cdot \frac{1}{i+1}. \tag{86}$$

Combining (85), (86) and (6) gives (57) for $\mu = \frac{1}{2}$.

Case (iii). Finally, suppose that $\frac{1}{2} < \mu < 1$. Once more, we apply (73) in Lemma B.1 for each $j \in \{1, \ldots, n\}$ with $a_k = g_{r,k,j}^2$, $\xi = \varsigma$ and $\alpha = 2\mu > 1$ and obtain that, for $j \in \{1, \ldots, n\}$,

$$\sum_{k=0}^{i} \frac{g_{r,k,j}^2}{w_{r,k,j}^2} \leq \frac{1}{1-2\mu} \left(\left(\varsigma + \sum_{k=0}^{i} g_{r,k,j}^2 \right)^{1-2\mu} - \varsigma^{1-2\mu} \right) \leq \frac{\varsigma^{1-2\mu}}{2\mu - 1}. \tag{87}$$

Substituting the bound (87) in (75) and using (74) and AS.3 gives that

$$\beta_{r,1} \sum_{k=0}^{i} \frac{1}{(\varsigma + \sum_{t=0}^{i} \|g_{r,t}\|^2)^{\mu}} \|g_{r,k}\|^2 \leq \Gamma_0 + \frac{n(\beta_{2,r} + \frac{1}{2}L)\varsigma^{1-2\mu}}{2\mu - 1}.$$

If we now suppose that

$$\sum_{k=0}^{i} \|g_{r,k}\|^2 \geq \varsigma, \tag{88}$$

then

$$\underset{k \in \{0,\ldots,i\}}{\text{average}} \|g_{r,k}\|^2 \leq \left[\frac{2^{\mu}}{\beta_{r,1}} \left(\Gamma_0 + \frac{n(\beta_{2,r} + \frac{1}{2}L)\varsigma^{1-2\mu}}{2\mu - 1} \right) \right]^{\frac{1}{1-\mu}} \cdot \frac{1}{i+1}. \tag{89}$$

If (88) does not hold, we derive that

$$\underset{k \in \{0,\ldots,i\}}{\text{average}} \|g_{r,k}\|^2 \leq \frac{\varsigma}{(i+1)}. \tag{90}$$

Thus, (89), (90) and (6) finally imply (57) for $\frac{1}{2} < \mu < 1$. □