Neural Network Training as an Optimal Control Problem — An Augmented Lagrangian Approach —

Brecht Evens[1] Puya Latafat[1] Andreas Themelis[2] Johan Suykens[1] Panagiotis Patrinos[1]

***Abstract*— Training of neural networks amounts to nonconvex optimization problems that are typically solved by using backpropagation and (variants of) stochastic gradient descent. In this work, we propose an alternative approach by viewing the training task as a nonlinear optimal control problem. Under this lens, backpropagation amounts to the sequential approach (single shooting) to optimal control, where the states variables have been eliminated. It is well known that single shooting may lead to ill-conditioning, and for this reason the simultaneous approach (multiple shooting) is typically preferred. Motivated by this hypothesis, an augmented Lagrangian algorithm is developed that only requires an approximate solution to the Lagrangian subproblems up to a user-defined accuracy. By applying this framework to the training of neural networks, it is shown that the inner Lagrangian subproblems are amenable to be solved using Gauss-Newton iterations. To fully exploit the structure of neural networks, the resulting linear least-squares problems are addressed by employing an approach based on forward dynamic programming. Finally, the effectiveness of our method is showcased on regression datasets.**

I. Introduction

Feedforward deep neural networks (DNNs) are a prominent model for supervised learning, having a lot of success in various fields. The primary objective of this work is to devise a novel method for training DNNs with smooth activation functions; this task can be formally stated as follows.

Main problem. *Given pairs* $\{(a^{(\ell)}, b^{(\ell)}) \in \mathbb{R}^{d_0} \times \mathbb{R}^{d_{N+1}}\}_{\ell\in[m]}$, *continuously differentiable functions* $\{\Phi_j : \mathbb{R}^{d_j} \to \mathbb{R}^{d_j}\}_{j\in[N+1]}$ *(operating in an element-wise fashion), and* $\mu_w > 0$, *find* $\{W_j \in \mathbb{R}^{d_j \times d_{j-1}}\}_{j\in[N+1]}$ *solutions to*

$$\underset{W_1,\dots,W_{N+1}}{\text{minimize}} \ \frac{1}{2m}\sum_{\ell=1}^{m} \|\Phi_{N+1}(W_{N+1}x_N^{(\ell)}) - b^{(\ell)}\|^2 + \frac{\mu_w}{2}\sum_{j=1}^{N+1}\|W_j\|_F^2$$

$$\text{where } x_0^{(\ell)} := a^{(\ell)}, \quad \ell \in [m], \tag{1a}$$

$$x_j^{(\ell)} := \Phi_j(W_j x_{j-1}^{(\ell)}), \quad j \in [N],\ \ell \in [m]. \tag{1b}$$

Here, $(a^{(\ell)}, b^{(\ell)})$ are (given) *training pairs*, $N \in \mathbb{N}$ is the number of *layers* of the network, each one having d_i many *neurons/nodes* and with Φ_i being the corresponding *activation function*, and μ_w is a regularization parameter for the *weights* W_i commonly used to avoid overfitting [15]. These optimization problems are typically solved using backpropagation [19] along with (variants of) stochastic gradient descent, due to their simplicity and effectiveness. However, these optimization methods suffer from various issues related to the challenging, highly nonconvex nature of the training task. First and foremost, due to the prominence of local minima and saddle points, trained DNN models tend to generalize poorly to test data. To alleviate this issue, various regularization methods have been introduced such as weight decay [15], batch normalization [13], and dropout [20], typically reducing the overfitting of the training data. More fundamentally, gradient-based methods are known to suffer from the vanishing gradient phenomenon [12], where the gradients in the output layers of DNNs decrease exponentially with the number of layers. Although recent studies have shown that piecewise affine activation functions such as ReLU, leaky ReLU [17], and Maxout unit [8] reduce the vanishing gradient problem by making the problem more sparse, the issue nevertheless persists especially in very deep networks.

To address these issues, in recent years a host of auxiliary variable methods have been introduced where the network structure is represented by equality constraints and the space of learning parameters is extended. By lifting the number of variables, these methods decompose the training task into a series of local subproblems which can be solved deterministically, typically using block coordinate descent (BCD) [4, 10, 24] or the alternating direction method of multipliers (ADMM) [21, 23, 25]. BCD and ADMM have been successful for this task due to their ability to convert the equality constrained optimization problems into unconstrained problems, which can then be solved more efficiently than their constrained counterparts. By increasing the dimension of the training problem, auxiliary variable methods can alleviate some of the issues from which classical gradient-based methods suffer. Most notably, it is observed that the vanishing gradient issue is alleviated as the auxiliary variables circumvent long-term dependencies between the network weights during training [25]. On the other hand, the increased dimensionality naturally makes the training task more challenging than when using classical gradient-based approaches.

The difference between traditional methods and auxiliary variable methods can be related to concepts from optimal control by viewing the training task as a nonlinear optimal control problem. Under this lens, auxiliary variable methods amount to the simultaneous approach (multiple shooting),

This work was supported by the Research Foundation Flanders (FWO) research projects G0A0920N, G086518N, G086318N, and PhD grant 1196820N; Research Council KU Leuven C1 project No. C14/18/068; Fonds de la Recherche Scientifique – FNRS and the Fonds Wetenschappelijk Onderzoek – Vlaanderen under EOS project no 30468160 (SeLMA); and the Japan Society for the Promotion of Science (JSPS) KAKENHI grant JP21K17710. Johan Suykens and Panagiotis Patrinos are affiliated to Leuven.AI - KU Leuven institute for AI, B-3000, Leuven, Belgium.

[1]KU Leuven, Department of Electrical Engineering ESAT-STADIUS – Kasteelpark Arenberg 10, bus 2446, B-3001 Leuven, Belgium {brecht.evens,puya.latafat,johan.suykens,panos.patrinos}@kuleuven.be

[2]Kyushu University, Faculty of Information Science and Electrical Engineering (ISEE) – 744 Motooka, Nishi-ku, Fukuoka 819-0395, Japan andreas.themelis@ees.kyushu-u.ac.jp

whereas backpropagation amounts to the sequential approach (single shooting), where the state variables are eliminated [16]. As it is well known that single shooting may lead to ill-conditioning of the optimization problem, it can be expected that multiple shooting methods can provide major advantages in the learning process of DNNs.

Motivated by this hypothesis, we develop a training methodology for neural networks based on an augmented Lagrangian framework that only requires finding approximate stationary points of the Lagrangian subproblems up to a user-defined accuracy. To fully exploit the structure of feed-forward neural networks, we additionally provide a computationally efficient approach to solve the inner subproblems based on forward dynamic programming. The overall approach leads to an efficient and provably convergent methodology for solving the highly nonconvex optimization problems emerging in the neural network training task.

A. Contributions

The contribution of this paper is twofold:

1) We introduce a novel augmented Lagrangian framework (ALM) for solving general nonconvex and nonsmooth equality constrained optimization problems. The framework is inspired by and extends [9, Alg. 1] by waiving smoothness assumptions and introducing a less conservative penalty update rule, yet preserving convergence to an approximate KKT point.

2) We apply this framework to the training of DNNs, which we address from an optimal control perspective. The resulting optimization problem's structure has a twofold benefit: first, the inner Lagrangian subproblems are amenable to be addressed with fast methods such as Gauss-Newton (GN); in turn, forward dynamic programming (FDP) can conveniently be employed to efficiently solve the resulting linear least-squares problems.

To reflect the modularity and the contribution of each component, the three procedures (outer ALM, inner GN, and FDP) are outlined in three standalone algorithms, each addressing a dedicated general problem.

B. Organization

The paper is organized as follows. The notation is introduced in the next subsection. An optimal control reformulation for the NN problem is presented in Section II. In Section III a novel augmented Lagrangian method (ALM) is proposed for general equality constrained nonlinear programs. The ALM method is specialized for the training of neural networks with smooth activation functions in Section IV, where a procedure based on the Gauss-Newton method and forward dynamic programming is proposed. The proofs of all the results are deferred to the appendix. Finally, numerical simulations showcasing the effectiveness of our proposed methodology on regression datasets are discussed in Section V.

C. Notation

We use $[N]$ to denote the set of indices $\{1,\dots,N\}$. We denote by $\mathbb{R}^n$ the standard n-dimensional Euclidean space with inner product $\langle\cdot,\cdot\rangle$ and induced norm $\|\cdot\|$. The set of extended real numbers is defined as $\overline{\mathbb{R}} \coloneqq \mathbb{R}\cup\{\infty\}$, and we say that an extended-real valued function $f:\mathbb{R}^n\to\overline{\mathbb{R}}$ is proper if $\operatorname{dom} f \coloneqq \{x\in\mathbb{R}^n \mid f(x)<\infty\}$ is nonempty. The set of real n-by-m matrices is denoted by $\mathbb{R}^{n\times m}$. Given $A\in\mathbb{R}^{n\times m}$, $\|A\|_F$ is its Frobenius norm and $\operatorname{vec}(A)\in\mathbb{R}^{nm}$ is the vector obtained by stacking the columns of A on top of one another. The sets of symmetric, symmetric positive semi-definite and symmetric positive definite n-by-n matrices are denoted by $\mathbb{S}^n$, $\mathbb{S}^n_+$ and $\mathbb{S}^n_{++}$, respectively. For $V\in\mathbb{S}^n_{++}$ we define the scalar product $\langle x,y\rangle_V = \langle x,Vy\rangle$ and the induced norm $\|x\|_V = \sqrt{\langle x,x\rangle_V}$. The n-by-n identity matrix is denoted by I_n, or simply I when no ambiguity occurs. The vector of all zeros with dimension n and the n-by-m matrix of all zeros are denoted by 0_n and $0_{n\times m}$, respectively. The matrix Kronecker product is denoted by $\otimes$. The Jacobian of a differentiable function $F:\mathbb{R}^n\to\mathbb{R}^m$, is denoted by $\mathrm{J}F:\mathbb{R}^n\to\mathbb{R}^{m\times n}$; $\mathrm{J}_x F$ is a short-hand notation for the partial derivative $\frac{\partial F}{\partial x}$.

II. An optimal control reformulation

In the traditional approach, (1a) and (1b) are absorbed into the cost, thus forming an unconstrained minimization which is then solved by employing a stochastic (sub)gradient-type method. Here we take an alternative approach by viewing the minimization as an optimal control problem with N stages. To this end, (1) represents the dynamics of the problem and may compactly be written as

$$X_j = \Phi_j(W_jX_{j-1}),\quad j\in[N], \tag{2}$$

where $X_j\in\mathbb{R}^{d_j\times m}$ is a matrix whose i-th column is the vector $x_j^{(i)}$, for $i\in[m]$. By similarly letting $A\in\mathbb{R}^{d_0\times m}$ and $Y\in\mathbb{R}^{d_{N+1}\times m}$ denote the input and output matrices (constructed using vectors $a^{(i)}$, $b^{(i)}$), the following compact reformulation of (1) is obtained

$$\begin{aligned}\underset{(W_i)_{i\in[N+1]},(X_i)_{i\in[N]}}{\text{minimize}}\quad & \tfrac{1}{2m}\|\Phi_{N+1}(W_{N+1}X_N)-Y\|_F^2+\tfrac{\mu_w}{2}\textstyle\sum_{i=1}^{N+1}\|W_i\|_F^2\\ \text{subject to}\quad & X_0 = A\\ & X_{j+1}=\Phi_j(W_jX_{j-1}),\quad j\in[N].\end{aligned} \tag{3}$$

A. Vectorized form

For simplicity of exposition and computational convenience, we condense the optimization variables W_i and X_i into a single long vector $\boldsymbol{z}=(\boldsymbol{w},\boldsymbol{x})$ with

$$\boldsymbol{w}=(\boldsymbol{w}_1,\dots,\boldsymbol{w}_{N+1})\quad\text{and}\quad\boldsymbol{x}=(\boldsymbol{x}_1,\dots,\boldsymbol{x}_N)$$

where, letting $w_{i,j}\in\mathbb{R}^{d_{i-1}}$ denote the j-th row of W_i and $x_i^{(j)}$ the j-th column of X_i,

$\boldsymbol{w}_i=(w_{i,1},\dots,w_{i,d_i})\in\mathbb{R}^{d_id_{i-1}}$ and $\boldsymbol{x}_i=(x_i^{(1)},\dots,x_i^{(m)})\in\mathbb{R}^{md_i}$. In the vectorized notation, the cost function and the nonlinear constraints in (3) may be represented by f and $F(z)=0$ with

$$\begin{aligned}f(\boldsymbol{z}) &= \tfrac{1}{2m}\|H_{N+1}(\boldsymbol{w}_{N+1},\boldsymbol{x}_N)-\boldsymbol{y}\|^2+\tfrac{\mu_w}{2}\|\boldsymbol{w}\|^2,\\ F(\boldsymbol{z}) &= \big(\boldsymbol{x}_1-H_1(\boldsymbol{w}_1,\boldsymbol{x}_0),\dots,\boldsymbol{x}_{N+1}-H_{N+1}(\boldsymbol{w}_{N+1},\boldsymbol{x}_N)\big),\end{aligned}$$

where $\boldsymbol{y}=\operatorname{vec}(Y)$ and

$$H_j(\boldsymbol{w}_j,\boldsymbol{x}_{j-1})=(\Phi_j(W_jx_{j-1}^{(1)}),\dots,\Phi_j(W_jx_{j-1}^{(m)})). \tag{4}$$

III. The outer ALM algorithm

With vectorized notation being adopted and as long as the activation functions Φ_j are locally Lipschitz, the minimization in (3) falls into the following general setting.

Problem I (General ALM framework)**.** *For a proper, lower semicontinuous, lower bounded $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ and a locally Lipschitz $F : \mathbb{R}^n \to \mathbb{R}^p$ such that $\{z \in \operatorname{dom} f \mid F(z) = 0\} \neq \emptyset$,*

$$\text{minimize}_{z\in\mathbb{R}^n} f(z) \quad \text{subject to } F(z) = 0. \tag{5}$$

This section proposes a conceptual algorithm for addressing Problem I, conceptual in the sense that, at this stage, no hint is given as to how the inner subproblems it involves can be solved. The algorithm will be concretized in the subsequent Section IV, where an implementable procedure for addressing these inner steps is detailed. The chosen method for the inner subproblems will ultimately require some additional structure and differentiability assumptions, which are nevertheless not needed for the (outer) ALM scheme presented in this section. For the sake of generality of the discussion and to well pinpoint where each requirement is invoked, the convergence proof of the outer scheme is given in this broader setting.

Equality constrained minimization problems as (5) are amenable to be addressed by means of augmented Lagrangian methods. For $\beta > 0$, we denote the corresponding β-augmented Lagrangian as

$$\begin{aligned}\mathscr{L}_\beta(z, \lambda) &:= f(z) + \langle \lambda, F(z)\rangle + \tfrac{\beta}{2}\|F(z)\|^2 \\ &= f(z) + \tfrac{\beta}{2}\|F(z) + \lambda/\beta\|^2 - \tfrac{1}{2\beta}\|\lambda\|^2,\end{aligned} \tag{6}$$

and we say that (z, λ) is an ε-KKT pair if

$$\|\nabla_z \mathscr{L}(z, \lambda)\|_\infty \le \varepsilon, \quad \text{and} \tag{7a}$$

$$\|F(z)\|_\infty \le \varepsilon, \tag{7b}$$

where $\mathscr{L} := \mathscr{L}_0$ is the (non-augmented) Lagrangian, and ∇_z denotes the gradient with respect to z or, in case of lack of differentiability, any vector in the subdifferential $\partial_z \mathscr{L}(z, \lambda)$.

Largely inspired by [9, Alg. 1], Algorithm 1 hinges on the upper boundedness of the augmented Lagrangian along the iterates (see, e.g. [3, Ex. 4.12]) ensured by the initialization at a feasible point z^0. Additionally to reducing the assumptions to the general setting of Problem I, we also introduced a less conservative update rule for the penalty parameter β, thus waiving the need for its unbounded growth which instead happens in [9] (unless the dual variable converges to 0).

Theorem 1. *Applied to Problem I, Algorithm 1 terminates yielding an ε-KKT pair for (5).*

Note that the result can cope with rather general functions f and F, not necessarily derived from formulations as in (3). The optimal control structure will instead be exploited in the following Section IV where an iterative method for addressing the inner problems at step 1.2 will be given.

Remark 2. When f is lower bounded, then so is $\mathscr{L}_\beta(\,\cdot\,, \lambda)$ for any λ, thus ensuring the existence of ε-stationary points as required in step 1.2 for any $\varepsilon > 0$. In the setting of the optimal control problem (3), not only is this condition trivially satisfied, but a feasible starting point z^0 can be obtained at virtually no cost by initializing the weights W_j^0 and unrolling the dynamics to generate the *state* variables X_j^0. □

Algorithm 1 ALM for Problem I

Require Initial feasible point $z^0 \in \operatorname{dom} f$ s.t. $F(z^0) = 0$, multiplier λ^0, and penalty $\beta_0 > 0$
Parameters $0 < \gamma < 1 < \alpha, \xi$ and tolerance $\varepsilon > 0$

For $k = 0, 1, 2 \ldots$

1.1: Set $\hat{z}^k = z^k$ if $\mathscr{L}_{\beta_k}(z^k, \lambda^k) \le \overline{f(z^0)}^{\mathscr{L}_{\beta_k}(z^0,\lambda^k)}$, or $\hat{z}^k = z^0$ otherwise

1.2: Starting at $\hat{z}^k$, apply a descent method to compute an ε-stationary point z^{k+1} of

$$\text{minimize}\, \mathscr{L}_{\beta_k}(\,\cdot\,, \lambda^k), \tag{8}$$

i.e., a point z^{k+1} such that

$$\|\nabla_z \mathscr{L}_{\beta_k}(z^{k+1}, \lambda^k)\|_\infty \le \varepsilon \tag{9}$$

1.3: Set $\lambda^{k+1} = \lambda^k + \beta_k F(z^{k+1})$

1.4: **if** $\|F(z^{k+1})\|_\infty \le \varepsilon$ **then**
Return ε-KKT pair (z^{k+1}, λ^{k+1})

1.5: Set $\beta_{k+1} = \beta_k$ if $\|F(z^{k+1})\|_\infty \le \gamma\|F(z^k)\|_\infty$,
or $\beta_{k+1} = \max\{\xi\beta_k, \beta_0(k+1)^\alpha\}$ otherwise.

IV. The Lagrangian subproblem via Gauss-Newton iterations

In this section we present a procedure for solving the inner minimization (8) in the setting of NNs with continuously differentiable activation functions. With the notational conventions of Section II-A, for a fixed multiplier $\lambda = (\lambda_1, \ldots, \lambda_N)$ (λ_j being the one associated with the j-th dynamics) the Lagrangian subproblem associated with (3) is cast as follows.

Problem II (Lagrangian subproblem)**.** *Given smooth functions $\{H_j\}_{j\in[N+1]}$, vectors λ, y of suitable sizes, and $\beta, \mu_w > 0$,*

$$\begin{aligned}\underset{z=(w,x)}{\text{minimize}}\, \mathscr{L}_\beta(z, \lambda) := \tfrac{1}{2m}\|H_{N+1}(w_{N+1}, x_N) - y\|^2 + \tfrac{\mu_w}{2}\|w\|^2 \\ -\tfrac{1}{2\beta}\|\lambda\|^2 + \tfrac{\beta}{2}\textstyle\sum_{j=1}^N \|x_j - H_j(w_j, x_{j-1}) + \lambda_j/\beta\|^2.\end{aligned}$$

This smooth unconstrained least-squares problem is amenable to be solved by the Gauss-Newton (GN) method, which amounts to iteratively solving minimizations obtained after linearizing functions H_j around the last iterates, and then applying a standard line search to guarantee convergence. In the next subsection, we derive explicit expressions of the Jacobian matrices involved in the linearization.

A. Gauss-Newton linearization and update direction

Let $D^j : \mathbb{R}^{d_{j-1}} \to \mathbb{R}^{d_j \times d_j}$ be given by

$$D^j(v) := \operatorname{diag}(\Phi_j'(\langle w_{j,1}, v\rangle), \ldots, \Phi_j'(\langle w_{j,d_j}, v\rangle)),$$

(recall that Φ_j operates element-wise) and define

$$\mathscr{D}^j := \operatorname{blkdiag}(D^j(x_{j-1}^{(1)}), \ldots, D^j(x_{j-1}^{(m)})).$$

The Jacobians $\mathrm{J}_{x_{j-1}} H_j \in \mathbb{R}^{md_j \times md_{j-1}}$ and $\mathrm{J}_{w_j} H_j \in \mathbb{R}^{md_j \times d_j d_{j-1}}$ are then given by

$$\mathrm{J}_{x_{j-1}} H_j(w_j, x_{j-1}) = \mathscr{D}^j(\mathrm{I}_m \otimes W_j), \quad \text{and}$$

Algorithm 2 Gauss-Newton procedure for Problem II

REQUIRE Initial point $z^0 = (w^0, x^0)$ and $0 < \eta_1, \eta_2 < 1$

For $l = 0, 1, 2 \ldots$

2.1: [UPDATE DIRECTION] set $p^l = \bar{z}^l - z^l$, where $\bar{z}^l = (\bar{w}^l, \bar{x}^l)$ solves Problem III with $\mathcal{A}, \mathcal{B}, c$ as in (10)

2.2: [LINE SEARCH] set $z^{l+1} = z^l + \tau_l p^l$, where τ_l is the largest number in $\{1, \eta_1, \eta_1^2, \ldots\}$ such that

$$\mathcal{L}_\beta(z^l + \tau_l p^l, \lambda) \le \mathcal{L}_\beta(z^l, \lambda) - 2\eta_2 \tau_l \mathcal{G}_{\mathcal{A},\mathcal{B},0}(p^l)$$

with $\mathcal{L}_\beta$ and $\mathcal{G}_{\mathcal{A},\mathcal{B},c}$ as in Problems II and III

2.3: IF $\|\nabla_z \mathcal{L}_\beta(z^{l+1}, \lambda)\|_\infty \le \varepsilon$ THEN

RETURN $z^{l+1} = (w^{l+1}, x^{l+1})$

$$\mathrm{J}_{w_j} H_j(w_j, x_{j-1}) = \mathcal{D}^j(\mathrm{I}_{d_j} \otimes x_{j-1}^{(1)}, \ldots, \mathrm{I}_{d_j} \otimes x_{j-1}^{(m)})^\top.$$

If $z^l = (w^l, x^l)$ is the l-th iterate of a GN algorithm, denoting

$$A_{j+1} = \begin{cases} 0_{md_1 \times md_0} & \text{if } j = 0 \\ \mathrm{J}_{x_j} H_{j+1}(w_{j+1}^l, x_j^l) & \text{if } j \in [N] \end{cases}$$
$$B_j = \mathrm{J}_{w_j} H_j(w_j^l, x_{j-1}^l), \quad j \in [N+1] \tag{10}$$
$$c_j = \begin{cases} H_j(w_j^l, x_{j-1}^l) - A_j x_{j-1}^l - B_j w_j^l - \frac{1}{\beta}\lambda_j & \text{if } j \in [N] \\ H_j(w_j^l, x_{j-1}^l) - A_j x_{j-1}^l - B_j w_j^l - y & \text{if } j = N+1, \end{cases}$$

the linearized minimization yielding the l-th GN update direction reduces to the following problem.

Problem III (GN direction)**.** *Given $\mathcal{A} = (A_1, \ldots, A_{N+1})$, $\mathcal{B} = (B_1, \ldots, B_{N+1})$ and $c = (c_1, \ldots, c_{N+1})$ with matrices A_j, B_j and vectors c_j of suitable sizes, and given scalars $\beta, \mu_w > 0$,*

$$\underset{z=(w,x)}{\text{minimize}}\ \mathcal{G}_{\mathcal{A},\mathcal{B},c}(z) \quad \textit{where,}$$

denoting $(\delta_j, \rho_j) = (1, \beta)$ for $j \le N$ and $(0, \frac{1}{m})$ otherwise,

$$\mathcal{G}_{\mathcal{A},\mathcal{B},c}(z) := \textstyle\sum_{j=1}^{N+1} \left(\frac{\rho_j}{2} \|\delta_j x_j - A_j x_{j-1} - B_j w_j - c_j\|^2 + \frac{\mu_w}{2} \|w_j\|^2 \right).$$

B. The Gauss-Newton algorithm

The structure of Problem III emphasizes how variables are weakly coupled, a phenomenon that owes to the stagewise structure of the optimal control problem (3). As a result, in spite of the large scale, Problem III admits a closed-form solution that is efficiently retrievable with a forward dynamic programming (FDP) approach detailed in the following Section IV-C. This routine may then be invoked by the GN method, synopsized in Algorithm 2, when computing the update directions at step 2.1.

In the next lemma we show that the GN method yields an ε-stationary solution for the original Lagrangian subproblem.

Lemma 3. *The limit points of the sequence $(z^l)_{l \in \mathbb{N}}$ generated by Algorithm 2 are stationary for Problem II. In particular, if $(z^l)_{l \in \mathbb{N}}$ remains bounded (e.g., when $\mathcal{L}_\beta(\cdot, \lambda)$ is coercive), then Algorithm 2 yields an ε-stationary solution.*

C. Forward dynamic programming

In this subsection we propose a recursive procedure for solving Problem III with given matrices $A_j \in \mathbb{R}^{r_j \times r_{j-1}}$, $B_j \in \mathbb{R}^{r_j \times s_j}$, and vectors $c_j \in \mathbb{R}^{r_j}$, $j \in [N+1]$, thus providing an efficient routine for step 2.1 of Algorithm 2. Inspired by

Algorithm 3 Recursive solution to Problem III with FDP

REQUIRE Initial state $x_0 \in \mathbb{R}^{r_0}$

set $M_1 = \frac{1}{\rho_1}\mathrm{I} + \frac{1}{\mu_u} B_1 B_1^\top$, $S_1 = \mathrm{I}_{r_0}$, $q_0 = x_0$

3.1: [FORWARD RECURSION] For $j = 1, \ldots, N$

(a) solve the linear system $M_j \tilde{c}_j = c_j$,

(b) $\tilde{q}_j = S_j q_{j-1}$ ▷ as described in (18)

(c) $q_j = \rho_j G_j A_j \tilde{q}_j + \tilde{c}_j$

(d) $M_{j+1} = \frac{1}{\rho_{j+1}}\mathrm{I} + \frac{1}{\mu_u} B_{j+1} B_{j+1}^\top + A_{j+1} M_j A_{j+1}^\top$

3.2: [BACKWARD RECURSION]

(a) $\tilde{x}_{N+1} = -S_{N+1}(A_{N+1}^\top G_{N+1} c_{N+1})$

$\tilde{q}_{N+1} = S_{N+1} q_N$ ▷ as described in (18)

(b) $x_N = \tilde{q}_{N+1} + \rho_{N+1} \tilde{x}_{N+1}$

$w_{N+1} = -E_{N+1}(A_{N+1} x_N + c_{N+1})$

(c) For $j = N, \ldots, 2$:

$\tilde{x}_j = S_j(A_j^\top G_j(x_j - c_j))$ ▷ as described in (18)

$x_{j-1} = \tilde{q}_j + \rho_j \tilde{x}_j$

$w_j = E_j(x_j - A_j x_{j-1} - c_j)$

(d) $u_1 = E_1(x_1 - A_1 x_0 - c_1)$

RETURN $z = (w, x)$ with $w = (w_1, \ldots, w_{N+1})$, $x = (x_1, \ldots, x_N)$

the idea of forward dynamic programming, the minimization may be split into a series of simpler subproblems that are solved in a recursive manner:

$$V_1^\star(x_1) = \min_{w_1} \left\{ \tfrac{\rho_1}{2} \|x_1 - A_1 x_0 - B_1 w_1 - c_1\|^2 + \tfrac{\mu_w}{2} \|w_1\|^2 \right\} \tag{11}$$

$$V_j^\star(x_j) = \min_{x_{j-1}, w_j} \Big\{ V_{j-1}^\star(x_{j-1}) + \tfrac{\rho_j}{2} \|x_j - A_j x_{j-1} - B_j w_j - c_j\|^2 + \tfrac{\mu_w}{2} \|w_j\|^2 \Big\}, \quad j = 2, \ldots, N \tag{12}$$

$$V_{N+1}^\star = \min_{x_N, w_{N+1}} \Big\{ V_N^\star(x_N) + \tfrac{\rho_{N+1}}{2} \|A_{N+1} x_N + B_{N+1} w_{N+1} + c_{N+1}\|^2 + \tfrac{\mu_w}{2} \|w_{N+1}\|^2 \Big\}. \tag{13}$$

Each stage consists of the minimization of the sum of the cost at the current stage and the optimal cost from the previous stage. The cost at the final stage $V_{N+1}^\star$ is equal to the optimal cost for Problem III. In order to obtain closed form solutions for each of the above minimizations, let $E_j \in \mathbb{R}^{s_j \times r_j}$ and $G_j, M_j, S_j \in \mathbb{R}^{r_j \times r_j}$, $j \in [N+1]$, be defined as

$$E_j = \left(\tfrac{\mu_w}{\rho_j}\mathrm{I} + B_j^\top B_j \right)^{-1} B_j^\top, \tag{14}$$
$$G_j = \mathrm{I} - B_j E_j, \tag{15}$$
$$M_j = \begin{cases} \frac{1}{\rho_1}\mathrm{I} + \frac{1}{\mu_w} B_1 B_1^\top & \text{if } j = 1 \\ \frac{1}{\rho_j}\mathrm{I} + \frac{1}{\mu_w} B_j B_j^\top + A_j M_{j-1} A_j^\top & \text{if } j > 1 \end{cases} \tag{16}$$
$$S_j = \begin{cases} \mathrm{I}_{r_0} & \text{if } j = 1 \\ M_{j-1} - M_{j-1} A_j^\top M_j^{-1} A_j M_j & \text{if } j > 1. \end{cases} \tag{17}$$

Note that matrices S_j need not be computed explicitly. Instead, given a vector $v \in \mathbb{R}^{r_j}$, $S_j v$ is computed as follows:

$$\begin{cases} (i) \text{ solve the linear system } M_j \bar{v} = A_j(M_{j-1} v) \\ (ii) \text{ set } S_j v = M_{j-1}(v - A_j^\top \bar{v}). \end{cases} \tag{18}$$

The FDP procedure is presented in Algorithm 3. Other than matrix-vector products, the algorithm requires solving linear systems several times, which may be performed by

computing the *Cholesky factorization* of M_j and $\frac{\mu_w}{\rho_j}\mathrm{I} + B_j^\top B_j$ once, thus resulting in operations involving simple forward and backward substitution steps that substantially reduce the computational overhead.

Remark 4 (Positive definiteness)**.** Since $\rho_j, \mu_w > 0$, it holds that $G_j, M_j \in \mathbb{S}^{r_j}_{++}$ for any j. Furthermore, using the Woodbury matrix identity and (16), the following alternative expression for S_{j+1} is obtained

$$S_{j+1} = (M_j^{-1} + \rho_{j+1} A_{j+1}^\top G_{j+1} A_{j+1})^{-1}, \tag{19}$$

establishing that also $S_{j+1} \in \mathbb{S}^{r_j}_{++}$. □

The optimality of the solution obtained by the FDP procedure is established in the next lemma.

Lemma 5. *Suppose that $\mu_w > 0$. Then, $z = (w, x)$ generated by Algorithm 3 is the unique minimizer of Problem III.*

V. Numerical experiments

A. Design of numerical experiments

We will generate training (and test) pairs $\{(a^{(\ell)}, b^{(\ell)})\}_{\ell \in [m]}$ for a three-layer neural network under the regression setting, analogous to the approach in [6], as follows:

$$b^{(\ell)} = W_3 \Phi(W_2 \Phi(W_1 a^{(\ell)})) + \delta,$$

where $a^{(\ell)} \sim \mathcal{N}(\mu, \Sigma)$ and $\delta \sim \delta_0 \mathcal{N}(0, 1)$. The mean $\mu \in \mathbb{R}^{d_0}$ and an additional random matrix $\Sigma_0 \in \mathbb{R}^{d_0 \times d_0}$ are generated by a normal distribution with standard deviation 0.2, and the covariance Σ is set to be $\Sigma_0^\top \Sigma_0$. The three-layer network consists of $N = 2$ hidden layers with respectively 20 and 5 neurons. As activation function the softplus function is used, i.e. $\Phi(x) := \ln(1 + \exp(x))$, a smooth approximation to the ReLU activation function which is often used in deep learning and known for its faster convergence. The weights W_i of the neural network are initialized according to Kaiming [11], which is a weight initialization procedure suitable for networks consisting of softplus activation functions, and we obtain a feasible starting point z^0 by applying (2) recursively. All networks in this section are trained with regularization parameter $\mu_w = 0.1$. The following parameters for Algorithm 1 are used:

$$\lambda^0 = 0, \qquad \beta_0 = 0.001 f(z^0), \qquad \gamma = 0.5,$$
$$\alpha = 2, \qquad \varepsilon = 10^{-3}, \qquad \xi = 2.$$

Furthermore, to prevent solving the inner problems (8) up to an unnecessarily high tolerance ε in the first iterations, Eq. (9) is relaxed as follows:

$$\|\nabla_z \mathcal{L}_{\beta_k}(z^{k+1}, \lambda^k)\|_\infty \leq \varepsilon_k := \max(\bar{\varepsilon}, 0.5\varepsilon_{k-1}) \tag{20}$$

with $\varepsilon_0 = 10^{-1}$ and $\bar{\varepsilon} = 10^{-2}$. Finally, the following parameters for the line search in Algorithm 2 are used:

$$\eta_1 = 0.8, \qquad \eta_2 = 0.1.$$

The ALM framework and corresponding Gauss-Newton procedure are implemented using the SciPy sparse matrix library [22] in Python. The CHOLMOD library [5] is used to factorize $(\frac{\mu_w}{\rho_j} I + B_j^\top B_j)$ and M_j. All experiments are conducted on an HP EliteBook 845 G7 with a 1.7GHz AMD Ryzen 7 PRO 4750U processor and 32 GB RAM.

B. Numerical results and discussion

The left-hand side of Table I shows the numerical results for training the previously introduced feedforward neural networks with varying input dimension d_0 and noise level δ_0 (averaged over 15 simulations) using our proposed ALM method, which yields an $\bar{\varepsilon}$-KKT pair after only a couple of ALM iterations (as (7a) is satisfied for $\bar{\varepsilon}$ instead of ε).

All experiments are performed with a fixed sample size $m = 250$ for the training and test datasets. We should remark that the current implementation does not scale well with the sample size m both in terms of memory usage and computation time, as the matrices M_j in the FDP procedure become increasingly large. For this reason, our method would greatly benefit from a mini-batch implementation where the training set is split into smaller batches to compute the inner GN steps. This is considered as future work.

The typical performance of the ALM algorithm with low tolerance ($\varepsilon = \bar{\varepsilon} = 10^{-7}$) is visualized in Fig. 1 for a simulation with $d_0 = 15$ and $\delta_0 = 20\%$. In the earlier GN iterations mainly the loss is reduced, while in the final iterations the feasibility is recovered as the penalty parameter increases in the outer ALM iterations. For this reason, it makes sense to terminate our algorithm at higher tolerances, as in neural network training we are mainly interested in reducing the loss.

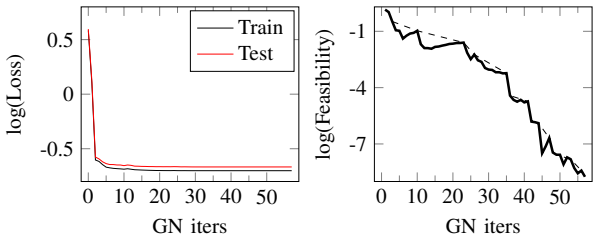

Fig. 1. Typical performance of the ALM algorithm. (Left) Training and test loss. (Right) Feasibility. The dashed line connects the points of the outer ALM iterations.

C. Comparison with first-order methods

We compare our method with two commonly used first-order methods for stochastic optimization, namely Adam [14] and stochastic gradient descent (SGD). We use the default implementations of these algorithms provided by the Keras library using the TensorFlow [1] backend with batch size 10, MSE loss function and ℓ_2 regularization with parameter μ_w.

The right portion of Table I shows the numerical results for training the three-layer network using Adam and SGD (averaged over 15 simulations) for 1000 epochs. No early stopping or other monitoring callbacks are used, minimizing the computation time per epoch. SGD is typically susceptible to stagnate at suboptimal points where it ceases to make significant progress, which explains its higher training MSE compared to Adam. When comparing with Adam and SGD it can be seen that our method tends to converge towards very good local optima, surpassing the performance

TABLE I

NUMERICAL RESULTS FOR TRAINING A THREE-LAYER NETWORK WITH VARYING INPUT DIMENSION AND NOISE LEVEL USING ALM, ADAM AND SGD.

		ALM							**Adam**			**SGD**		
d_0	δ_0	Training MSE	Test MSE	$\mathscr{L}_{\beta_k}$ evals	$\nabla_x \mathscr{L}_{\beta_k}$ evals	ALM iters	GN iters	Time (m:ss)	Training MSE	Test MSE	Time (m:ss)	Training MSE	Test MSE	Time (m:ss)
5	10%	5.37e-2	5.04e-2	22	19	6	13	0:09	5.36e-2	5.05e-2	0:14	5.47e-2	5.12e-2	0:13
5	20%	6.93e-2	6.62e-2	22	19	6	13	0:09	6.92e-2	6.63e-2	0:15	7.03e-2	6.68e-2	0:12
10	10%	6.52e-2	6.47e-2	32	25	6	19	0:14	6.50e-2	6.44e-2	0:15	6.56e-2	6.48e-2	0:12
10	20%	7.95e-2	8.08e-2	35	27	6	20	0:15	7.93e-2	8.07e-2	0:14	8.04e-2	8.15e-2	0:13
15	10%	7.36e-2	7.98e-2	40	29	6	22	0:17	7.34e-2	7.96e-2	0:15	7.63e-2	8.35e-2	0:12
15	20%	8.76e-2	9.49e-2	41	30	6	23	0:17	8.74e-2	9.47e-2	0:15	9.07e-2	9.93e-2	0:12

of SGD and occasionally even finding a better local minimum than Adam. Furthermore, the computation time of our methodology for training the introduced networks is reasonably similar to the ones of Adam and SGD. Overall, these results are encouraging as our method is expected to greatly benefit from a mini-batch implementation, further reducing the computation time and increasing scalability.

VI. CONCLUSIONS

In this paper a novel procedure for the training of neural networks was introduced that leverages an optimal control view and relies on three main components. First, a novel augmented Lagrangian method is presented for general nonsmooth nonconvex equality constrained problems, which attains an ε-KKT solution. Second, when applied to the DNN problem we propose to solve the Lagrangian subproblems by employing Gauss-Newton iterations resulting in a series of linear least-squares problems. Third, owing to the stagewise structure in the optimal control formulation, we solve the linear least-squares GN problems through a simple recursive procedure based on forward dynamic programming. We observed encouraging results in comparison to fast first-order solvers such as Adam which are often used in a heuristic manner without theoretical guarantees. In the current implementation, our method is not competitive when using large numbers of training data. Future research directions include extending our scheme to mini-batch settings to tackle this issue. It is also interesting to extend the framework to allow for nonsmooth activations functions.

REFERENCES

[1] M. Abadi, A. Agarwal, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467*, 2016.

[2] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2016.

[3] E. Birgin and J. Martínez. *Practical Augmented Lagrangian Methods for Constrained Optimization*. SIAM, 2014.

[4] M. Carreira-Perpinan and W. Wang. Distributed optimization of deeply nested systems. In *Artif. Intell. Stat.*, pages 10–19, 2014.

[5] Y. Chen, T. Davis, et al. Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM Trans Math Softw*, 35(3), oct 2008.

[6] Y. Cui, Z. He, and J. Pang. Multicomposite nonconvex optimization for training deep neural networks. *SIAM J. Optim.*, 30(2):1693–1723, 2020.

[7] F. Facchinei and J. Pang. *Finite-dimensional variational inequalities and complementarity problems*, volume II. Springer, 2003.

[8] I. Goodfellow, D. Warde-Farley, et al. Maxout networks. In *Int. Conf. Mach. Learn.*, pages 1319–1327, 2013.

[9] G. Grapiglia and Y. Yuan. On the complexity of an augmented Lagrangian method for nonconvex optimization. *IMA J. Numer. Anal.*, jul 2020.

[10] F. Gu, A. Askari, and L. El Ghaoui. Fenchel lifted networks: A Lagrange relaxation of neural network training. In *Int. Conf. Artif. Intell. Stat.*, pages 3362–3371, 2020.

[11] K. He, X. Zhang, et al. Deep residual learning for image recognition. In *2016 IEEE Conf. Comput. Vision Pattern Recogn. (CVPR)*, pages 770–778, Los Alamitos, CA, USA, jun 2016. IEEE Computer Society.

[12] S. Hochreiter, Y. Bengio, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[13] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. 32nd Int. Conf. Mach. Learn.*, volume 37 of *Proc. Mach. Learn. Res.*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[14] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[15] A. Krogh and J. Hertz. A simple weight decay can improve generalization. In *Proc. 4th Int. Conf. NIPS*, page 950–957. Morgan Kaufmann Publishers Inc., 1991.

[16] Y. LeCun. *A theoretical framework for back-propagation*. IEEE Computer Society Press, 1992.

[17] A. Maas, A. Hannun, and A. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc ICML*, volume 30, page 3, 2013.

[18] R. Rockafellar and R. Wets. *Variational Analysis*, volume 317. Springer, 2009.

[19] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[20] N. Srivastava, G. Hinton, et al. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014.

[21] G. Taylor, R. Burmeister, et al. Training neural networks without gradients: A scalable ADMM approach. In *Int. Conf. Mach. Learn.*, pages 2722–2731, 2016.

[22] P. Virtanen, R. Gommers, et al. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17:261–272, 2020.

[23] J. Wang, F. Yu, et al. ADMM for efficient deep learning with global convergence. In *Proc. 25th ACM SIGKDD Int. Conf. Knowledge Discov. Data Min.*, pages 111–119, 2019.

[24] Z. Zhang and M. Brand. On the convergence of block coordinate descent in training dnns with Tikhonov regularization. In *Adv. Neural Inf. Process. Syst.*, pages 1719–1728, 2017.

[25] Z. Zhang, Y. Chen, and V. Saligrama. Efficient training of very deep neural networks for supervised hashing. In *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, pages 1487–1495, 2016.

Appendix

Lemma A.1. *Suppose that $G : \mathbb{R}^n \to \mathbb{R}^p$ is locally Lipschitz around a point $\bar{z}$ at which $G(\bar{z}) = 0$. Then, $\phi(z) := \frac{1}{2}\|G(z)\|^2$ is strictly differentiable at $\bar{z}$ (in the sense of [18, Def. 9.17]) with null gradient.*

Proof. Let L be a Lipschitz constant for G in a neighborhood $\mathcal{U}$ of $\bar{z}$. Then, for $z, z' \in \mathcal{U}$ we have

$$\begin{aligned}\frac{\left|\phi(z) - \phi(z') - \langle 0, z - z'\rangle\right|}{\|z - z'\|} &= \frac{\left|\|G(z)\|^2 - \|G(z')\|^2\right|}{2\|z - z'\|} \\ &= \frac{\left|\|G(z) - G(z')\|^2 + 2\langle G(z'), G(z) - G(z')\rangle\right|}{2\|z - z'\|} \\ &\le \tfrac{L}{2}\|G(z) - G(z')\| + L\|G(z')\|\end{aligned}$$

which vanishes as $z, z' \to \bar{z}$, hence the claim. □

Lemma A.2. *Let $\mathcal{H}_i \in \mathbb{R}^{r_i \times r_i}$ be symmetric positive definite, $\mathcal{V}_i \in \mathbb{R}^{r_i \times p}$, and $\nu_i \in \mathbb{R}^{r_i}$, $i \in [N]$. If $\mathcal{U} := \sum_{i=1}^N \mathcal{V}_i^\top \mathcal{H}_i^{-1} \mathcal{V}_i$ is symmetric positive definite, then*

$$\textstyle\sum_{i=1}^N \|\mathcal{V}_i x - \nu_i\|^2_{\mathcal{H}_i^{-1}} = \|\mathcal{U}x - d\|^2_{\mathcal{U}^{-1}} - \|d\|^2_{\mathcal{U}^{-1}} + \sum_{i=1}^N \|\nu_i\|^2_{\mathcal{H}_i^{-1}},$$

where $d := \sum_{i=1}^N \mathcal{V}_i^\top \mathcal{H}_i^{-1} \nu_i$.

Proof. Let $q(x) = \sum_{i=1}^N \|\mathcal{V}_i x - \nu_i\|^2_{\mathcal{H}_i^{-1}}$. That $q(x) = \|x\|^2_{\mathcal{U}} + \sum_{i=1}^N \|\nu_i\|^2_{\mathcal{H}_i^{-1}} - 2\langle x, d\rangle$ is of immediate verification. Since q is quadratic, the Taylor expansion around its minimizer $x^\star = \mathcal{U}^{-1}d$ is given by $q(x) = q(x^\star) + \|x - x^\star\|^2_{\mathcal{U}}$. Substituting $x^\star$ results in the claimed form. □

Proof of Theorem 1. Owing to the update at step 1.3,

$$\mathcal{L}(z, \lambda^{k+1}) = \mathcal{L}_{\beta_k}(z, \lambda^k) + \tfrac{\beta_k}{2}\|F(z^{k+1})\|^2 - \tfrac{\beta_k}{2}\|F(z) - F(z^{k+1})\|^2.$$

The last term on the right-hand side is continuously differentiable (with null gradient) at z^{k+1}, owing to Lemma A.1. It then follows from [18, Ex. 8.8(c)] that $\partial_z \mathcal{L}(z^{k+1}, \lambda^{k+1}) = \partial_z \mathcal{L}_{\beta_k}(z^{k+1}, \lambda^k)$, hence that the pair (z^k, λ^k) satisfies condition (7a) for every $k \ge 1$, by virtue of Eq. (9) in step 1.2. It remains to show that (7b) too is eventually satisfied. Notice that, by definition of $\hat{z}^k$ at step 1.1, $\mathcal{L}(z^{k+1}, \lambda^k) \le f(z^0)$ holds for every k, which combined with (6) yields

$$\begin{aligned}\tfrac{1}{2\beta_k}\|\lambda^{k+1}\|^2 = \tfrac{\beta_k}{2}\|F(z^{k+1}) + \lambda^k/\beta_k\|^2 &\le f(z^0) - f(z^{k+1}) + \tfrac{1}{2\beta_k}\|\lambda^k\|^2 \\ &\le c + \tfrac{1}{2\beta_k}\|\lambda^k\|^2,\end{aligned}$$

where $c := f(z^0) - \inf f$ is a constant. Since $\beta_{k+1} \ge \beta_k$, it holds that $\frac{1}{2\beta_{k+1}}\|\lambda^{k+1}\|^2 \le c + \frac{1}{2\beta_k}\|\lambda^k\|^2$, which leads to

$$\tfrac{1}{\beta_k}\|\lambda^k\|^2 \le \tfrac{1}{\beta_0}\|\lambda^0\|^2 + 2kc \tag{A.1a}$$

for every $k \in \mathbb{N}$. Moreover, since

$$\begin{aligned}\tfrac{1}{2}\|F(z^{k+1})\|^2 &\le \|F(z^{k+1}) + \lambda^k/\beta_k\|^2 + \|\lambda^k/\beta_k\|^2 \\ &= \tfrac{1}{\beta_k^2}[\|\lambda^{k+1}\|^2 + \|\lambda^k\|^2],\end{aligned} \tag{A.1b}$$

the β-update at step 1.5 implies that $\|F(z^{k+1})\|_\infty \to 0$ (Q-linearly) if β_k is asymptotically constant, hence the claim. Otherwise, the set $\mathcal{K} := \{k \in \mathbb{N} \mid \beta_k = \max\{\xi\beta_{k-1}, \beta_0 k^\alpha\}\}$ is infinite. Then, for $k \in \mathcal{K}$, combining (A.1) yields

$$\begin{aligned}\tfrac{1}{2}\|F(z^{k+1})\|^2 &\le \tfrac{\beta_{k+1}}{\beta_k^2}(\tfrac{1}{\beta_0}\|\lambda^0\|^2 + 2(k+1)c) + \tfrac{1}{\beta_k}(\tfrac{1}{\beta_0}\|\lambda^0\|^2 + 2kc) \\ &\le \max\left\{\tfrac{\xi}{\beta_k}, \tfrac{\beta_0(k+1)^\alpha}{\beta_k^2}\right\}(\tfrac{1}{\beta_0}\|\lambda^0\|^2 + 2(k+1)c) \\ &\quad + \tfrac{1}{\beta_k}(\tfrac{1}{\beta_0}\|\lambda^0\|^2 + 2kc) \\ (\beta_k \ge k^\alpha) \quad &\le \max\left\{\tfrac{\xi}{k^\alpha}, \tfrac{\beta_0(k+1)^\alpha}{k^{2\alpha}}\right\}(\tfrac{1}{\beta_0}\|\lambda^0\|^2 + 2(k+1)c) \\ &\quad + \tfrac{1}{k^\alpha}(\tfrac{1}{\beta_0}\|\lambda^0\|^2 + 2kc) \to 0\end{aligned}$$

as $\mathcal{K} \ni k \to \infty$, owing to the fact that $\alpha > 1$. The second inequality uses the fact that, regardless of whether $k+1 \in \mathcal{K}$ or not, $\beta_{k+1} \le \max\{\xi\beta_k, \beta_0(k+1)^\alpha\}$ holds (since $\xi > 1$). □

Proof of Lemma 3. Note that matrices A_j, B_j and vectors $\boldsymbol{c}_j$ in Algorithm 2 depend on the current iterate z^l. Here, we use superscript l to emphasize this dependence. The linear least-squares Problem III solved at step 2.1 may equivalently be written as

$$\underset{z=(\boldsymbol{w},\boldsymbol{x})}{\text{minimize}}\ \tfrac{1}{2}\|J(z^l)z - b(z^l)\|^2, \tag{A.2}$$

where

$$J(z^l) = \begin{pmatrix} -\bar{B}^l & \bar{A}^l \\ \sqrt{\mu_w}\mathrm{I} & \end{pmatrix}$$

with

$$\bar{A} = \begin{pmatrix} \sqrt{\rho_1}\mathrm{I} & & & & \\ -\sqrt{\rho_2}A_2^l & \sqrt{\rho_2}\mathrm{I} & & & \\ & \ddots & \ddots & & \\ & & -\sqrt{\rho_N}A_N^l & \sqrt{\rho_N}\mathrm{I} & \\ & & & -\sqrt{\rho_{N+1}}A_{N+1}^l & \end{pmatrix},$$

$$\bar{B}^l = \operatorname{blkdiag}(\sqrt{\rho_1}B_1^l, \ldots, \sqrt{\rho_{N+1}}B_{N+1}^l),$$

and $b(z^l) = (\sqrt{\rho_1}c_1, \ldots, \sqrt{\rho_{N+1}}c_{N+1}, 0_s)$.

In what follows we show that the eigenvalues of $J(z^l)^\top J(z^l)$ along a converging subsequence are bounded above and away from zero, and that step 2.2 is a restatement of the standard Armijo line search, at which point the claimed convergence results follow by standard arguments for gradient methods [2, 7]. For the latter, note that by the optimality conditions for (A.2), the solution $\bar{z}^l$ satisfies $J(z^l)^\top J(z^l)\bar{z}^l = J(z^l)^\top b(z^l)$. Moreover, since $\nabla_z \mathcal{L}_\beta(z^l, \lambda) = J(z^l)^\top J(z^l)z^l - J(z^l)^\top b(z^l)$, combining the two equalities yields $\left\langle \nabla_z \mathcal{L}_\beta(z^l, \lambda), \bar{z}^l - z^l \right\rangle = -\|J(z^l)(\bar{z}^l - z^l)\|^2 = -2\mathcal{G}_{\mathcal{A},\mathcal{B},0}(\boldsymbol{p}^l)$ establishing the claimed equivalence.

Suppose $(z^l)_{l\in K}$ is a subsequence converging to a limit point $z^\star$. Note that $J(z)^\top J(z)$ is nonsingular since $J(z)$ has full column rank for any z. Therefore, by continuity of $J(\cdot)$ and [7, Lem. 7.5.2] we have that $c_1\mathrm{I} \le J(z^l)^\top J(z^l) \le c_2\mathrm{I}$ for some $c_1, c_2 > 0$. The stationarity of $z^\star$ follows from [7, Prop. 8.3.7]. If $(z^l)_{l\in\mathbb{N}}$ remains bounded at least one converging subsequence exists establishing the claim. □

Proof of Lemma 5. First, note that by (14)

$$\begin{aligned}G_j^\top G_j + \tfrac{\mu_w}{\rho_j}E_j^\top E_j &= \mathrm{I} - B_jE_j - E_j^\top B_j^\top + E_j^\top\left(B_j^\top B_j + \tfrac{\mu_w}{\rho_j}\mathrm{I}\right)E_j \\ &= \mathrm{I} - B_jE_j = G_j.\end{aligned} \tag{A.3}$$

We proceed by induction to show that, for $j \in [N]$,

$$V_j^\star(\boldsymbol{x}_j) = \tfrac{1}{2}\|M_j^{-1}\boldsymbol{x}_j - \boldsymbol{q}_j\|^2_{M_j} + C_j, \tag{A.4}$$

where the term C_j does not depend on $\boldsymbol{x}_j, \boldsymbol{x}_{j+1}, \ldots, \boldsymbol{x}_N$. Here, we avoid deriving a recursion for C_j since it does not affect the computation of $\boldsymbol{x}_j$ and $\boldsymbol{w}_{j+1}$ in the next stages.

For the base case $j = 1$, by the first-order optimality condition for the minimization (11) the unique minimizer is computed as

$$\boldsymbol{w}_1^\star(x_1) = E_1(\boldsymbol{x}_1 - A_1\boldsymbol{x}_0 - \boldsymbol{c}_1).$$

After substitution, by using (A.3) and simple algebra we obtain

$$V_1^\star(\boldsymbol{x}_1) = \tfrac{1}{2}\|\boldsymbol{x}_1 - A_1\boldsymbol{x}_0 - \boldsymbol{c}_1\|^2_{\rho_1 G_1} = \tfrac{1}{2}\|M_1^{-1}\boldsymbol{x}_1 - \boldsymbol{q}_1\|^2_{M_1},$$

where $M_1 = \rho_1^{-1}G_1^{-1}$ and $q_1 = M_1^{-1}(A_1\boldsymbol{x}_0 + \boldsymbol{c}_1)$.

Arguing by induction, suppose that (A.4) holds for some j such that $1 \le j \le N-1$. Let $\varphi(\boldsymbol{x}_j, \boldsymbol{w}_{j+1})$ denote the argument being minimized in (12). From direct computation

$$\nabla^2\varphi(\boldsymbol{x}_j, \boldsymbol{w}_{j+1}) = \begin{pmatrix} M_j^{-1} + \rho_{j+1}A_{j+1}^\top A_{j+1} & \rho_{j+1}A_{j+1}^\top B_{j+1} \\ \rho_{j+1}B_{j+1}^\top A_{j+1} & \mu_w \mathrm{I} + \rho_{j+1}B_{j+1}^\top B_{j+1} \end{pmatrix}.$$

Since $M_j \in \mathbb{S}^{r_j}_{++}$, by forming its Schur complement and using (14) it follows that the Hessian is symmetric positive definite if and only if so is $M_j^{-1} + \rho_{j+1}A_{j+1}^\top G_{j+1}A_{j+1}$, which holds true. Hence, the subproblems have unique solutions. By the first-order optimality condition for (12), the solution pair $(\boldsymbol{x}_j^\star, \boldsymbol{w}_{j+1}^\star)$ satisfies

$$0 = M_j^{-1}\boldsymbol{x}_j^\star - \boldsymbol{q}_j - \rho_{j+1}A_{j+1}^\top\Big(\boldsymbol{x}_{j+1} - A_{j+1}\boldsymbol{x}_j^\star - B_{j+1}\boldsymbol{w}_{j+1}^\star - \boldsymbol{c}_{j+1}\Big)$$

and

$$0 = \mu_w\boldsymbol{w}_{j+1}^\star - \rho_{j+1}B_{j+1}^\top\Big(\boldsymbol{x}_{j+1} - A_{j+1}\boldsymbol{x}_j^\star - B_{j+1}\boldsymbol{w}_{j+1}^\star - \boldsymbol{c}_{j+1}\Big).$$

The latter reads $\boldsymbol{w}_{j+1}^\star = E_{j+1}(\boldsymbol{x}_{j+1} - A_{j+1}\boldsymbol{x}_j^\star - \boldsymbol{c}_{j+1})$. After substituting $\boldsymbol{w}_{j+1}^\star$ into the former, using (14) and (A.3) we obtain

$$\boldsymbol{x}_j^\star = S_{j+1}\boldsymbol{q}_j + P_j(\boldsymbol{x}_{j+1} - \boldsymbol{c}_{j+1}),$$

where $P_k = \rho_{j+1}S_{j+1}A_{j+1}^\top G_{j+1}$ and S_{j+1} is as in (17). Substituting the minimizer pair $(\boldsymbol{x}_j^\star, \boldsymbol{w}_{j+1}^\star)$ back in (12) and using (A.3) yields

$$\begin{aligned} V_{j+1}^\star(\boldsymbol{x}_{j+1}) &= C_j + \tfrac{1}{2}\|M_j^{-1}\boldsymbol{x}_j^\star - \boldsymbol{q}_j\|^2_{M_j} \\ &\quad + \tfrac{1}{2}\|\boldsymbol{x}_{j+1} - A_{j+1}\boldsymbol{x}_j^\star - \boldsymbol{c}_{j+1}\|^2_{\rho_{j+1}G_{j+1}} \\ \text{(subs. } \boldsymbol{x}_j^\star) &= C_j + \tfrac{1}{2}\|\mathscr{V}_1\boldsymbol{x}_{j+1} - \nu_1\|^2_{\mathscr{H}_1^{-1}} \\ &\quad + \tfrac{1}{2}\|\mathscr{V}_2\boldsymbol{x}_{j+1} - \nu_2\|^2_{\mathscr{H}_2^{-1}} \end{aligned}$$

where $\mathscr{V}_1 = M_j^{-1}P_j$, $\mathscr{H}_1 = M_j^{-1}$, $\mathscr{V}_2 = \big(\mathrm{I} - A_{j+1}P_j\big)$, $\mathscr{H}_2 = \rho_{j+1}^{-1}G_{j+1}^{-1}$,

$$\nu_1 = \big(\mathrm{I} - M_j^{-1}S_{j+1}\big)\boldsymbol{q}_j + M_j^{-1}P_j\boldsymbol{c}_{j+1}, \text{ and}$$
$$\nu_2 = A_{j+1}S_{j+1}\boldsymbol{q}_j + (\mathrm{I} - A_{j+1}P_j)\boldsymbol{c}_{j+1}.$$

On the other hand, we have that

$$\begin{aligned} \mathscr{U} &= \sum_{i=1}^{2}\mathscr{V}_i^\top\mathscr{H}_i^{-1}\mathscr{V}_i \\ &= P_j^\top M_j^{-1}P_j + \rho_{j+1}\big(\mathrm{I} - A_{j+1}P_j\big)^\top G_{j+1}\big(\mathrm{I} - A_{j+1}P_j\big) \quad \text{(A.5)} \\ &= P_j^\top S_{j+1}^{-1}P_j + \rho_{j+1}(\mathrm{I} - P_j^\top A_{j+1}^\top)G_{j+1} - \rho_{j+1}G_{j+1}A_{j+1}P_j \\ &= \rho_{j+1}G_{j+1} - \rho_{j+1}^2 G_{j+1}A_{j+1}S_{j+1}A_{j+1}^\top G_{j+1} \\ &= M_{j+1}^{-1}, \end{aligned}$$

where (19) was used in the second equality, and the Woodbury matrix identity in the last one. Therefore, we may apply Lemma A.2 to obtain

$$V_{j+1}^\star(\boldsymbol{x}_{j+1}) = \tfrac{1}{2}\|M_{j+1}^{-1}\boldsymbol{x}_{j+1} - \boldsymbol{q}_{j+1}\|^2_{M_{j+1}}$$
$$- \tfrac{1}{2}\|\boldsymbol{q}_{j+1}\|^2_{M_{j+1}} + \tfrac{1}{2}\textstyle\sum_{i=1}^{2}\|\nu_i\|^2_{\mathscr{H}_i^{-1}} + C_j, \quad \text{(A.6)}$$

with

$$\boldsymbol{q}_{j+1} = P_j^\top\nu_1 + \rho_{j+1}\big(\mathrm{I} - A_{j+1}P_j\big)^\top G_{j+1}\nu_2 \quad \text{(A.7)}$$
$$= M_{j+1}^{-1}\boldsymbol{c}_{j+1} + \rho_{j+1}G_{j+1}A_{j+1}\boldsymbol{q}_j, \quad \text{(A.8)}$$

where we used (19) and the alternative expression for M_{j+1} in (A.5). The last three terms in (A.6) are absorbed into C_{j+1} completing the induction argument.

It remains to solve (13). Arguing as before, from the first-order optimality condition the solution pair $(\boldsymbol{x}_N^\star, \boldsymbol{w}_{N+1}^\star)$ must satisfy

$$0 = M_N^{-1}\boldsymbol{x}_N^\star - \boldsymbol{q}_N + \rho_{N+1}A_{N+1}^\top\Big(A_{N+1}\boldsymbol{x}_N^\star + B_{N+1}\boldsymbol{w}_{N+1}^\star + \boldsymbol{c}_{N+1}\Big)$$

and

$$0 = \mu_w\boldsymbol{w}_{N+1}^\star + \rho_{N+1}B_{N+1}^\top\Big(A_{N+1}\boldsymbol{x}_N^\star + B_{N+1}\boldsymbol{w}_{N+1}^\star + \boldsymbol{c}_{N+1}\Big).$$

The former equality is equivalent to the one given in step 3.2. After substituting $\boldsymbol{w}_{N+1}^\star$ back into the latter and using (A.3), the update for $\boldsymbol{x}_N^\star$ is obtained. □