Second-order backpropagation algorithms for a stagewise-partitioned separable Hessian matrix

Eiji Mizutani
Department of Computer Science
Tsing Hua University
Hsinchu, 300 Taiwan
E-mail: eiji@wayne.cs.nthu.edu.tw

Stuart E. Dreyfus
Industrial Engineering & Operations Research
University of California, Berkeley
Berkeley, CA 94720 USA
E-mail: dreyfus@ieor.berkeley.edu

James W. Demmel
Mathematics & Computer Science
University of California, Berkeley
Berkeley, CA 94720 USA
E-mail: demmel@cs.berkeley.edu

***Abstract*— Recent advances in computer technology allow the implementation of some important methods that were assigned lower priority in the past due to their computational burdens. Second-order backpropagation (BP) is such a method that computes the exact Hessian matrix of a given objective function. We describe two algorithms for feed-forward neural-network (NN) learning with emphasis on how to organize Hessian elements into a so-called stagewise-partitioned block-arrow matrix form: (1) stagewise BP, an extension of the discrete-time optimal-control stagewise Newton of Dreyfus 1966; and (2) nodewise BP, based on direct implementation of the chain rule for differentiation attributable to Bishop 1992. The former, a more systematic and cost-efficient implementation in both memory and operation, progresses in the same layer-by-layer (i.e., stagewise) fashion as the widely-employed first-order BP computes the gradient vector. We also show intriguing separable structures of each block in the partitioned Hessian, disclosing the rank of blocks.**

I. INTRODUCTION

In multi-stage *optimal control* problems, second-order optimization procedures (see [8] and references therein) proceed in a *stagewise* manner since N, the number of stages, is often very large. Naturally, those methods can be employed for optimizing multi-stage feed-forward neural networks: In this paper, we focus on an N-layered multilayer perceptron (MLP), which gives rise to an N-stage decision making problem. At each stage s, we assume there are P_s $(s=1,\cdots,N)$ states (or nodes) and n_s $(s=1,\cdots,N-1)$ decision parameters (or weights), denoted by an n_s-vector $\boldsymbol{\theta}^{s,s+1}$ (between layers s and s+1). No decisions are to be made at terminal stage N (or layer N); hence, the N-1 decision stages in total. To compute the gradient vector for optimization purposes, we employ the "first-order" backpropagation (BP) process [5], [6], [7], which consists of two major procedures: *forward pass* and *backward pass* [see later Eq. (2)]. A forward-pass situation in MLP-learning, where the node outputs in layer $s-1$ (denoted by $\mathbf{y}^{s-1}$) affect the node outputs in the next layer s (i.e., $\mathbf{y}^s$) via connection parameters (denoted by $\boldsymbol{\theta}^{s-1,s}$ between those two layers), can be interpreted as a situation in optimal control where state $\mathbf{y}^{s-1}$ at stage s–1 is moved to state $\mathbf{y}^s$ at the next stage s by decisions $\boldsymbol{\theta}^{s-1,s}$. In the backward pass, *sensitivities of the objective function E with respect to states* (i.e., **node sensitivities**) are propagated from one stage back to another while computing gradients and Hessian elements. However, MLPs exhibit a great deal of structure, which turns out to be a very special case in optimal control; for instance, the "after-node" outputs (or *states*) are evaluated individually at each stage as $y_j^s = f_j^s(x_j^s)$, where $f(.)$ denotes a differentiable state-transition function of nonlinear dynamics, and x_j^s, the "before-node" *net input* to node j at layer s, depends on only a *subset* of all decisions taken at stage $s-1$. In spite of this distinction and others, using a vector of states as a basic ingredient allows us to adopt analogous formulas available in the optimal control theory (see [8]). The key concept behind the theory resides in **stagewise** implementation; in fact, first-order BP is essentially a simplified *stagewise* optimal-control gradient formula developed in early 1960s [6]. We first review the important "stagewise concept" of first-order BP, and then advance to stagewise second-order BP with particular emphasis on our organization of Hessian elements into a *stagewise-partitioned block-arrow Hessian matrix* form.

II. STAGEWISE FIRST-ORDER BACKPROPAGATION

The backward pass in MLP-learning starts evaluating the so-called terminal **after-node sensitivities** (also known as **costates** or **multipliers** in optimal control) $\xi_k^N \equiv \frac{\partial E}{\partial y_k^N}$ (defined as partial derivatives of an objective function E with respect to y_k^N, the output of node k at layer N) for $k=1,...,P_N$, yielding a P_N-vector $\boldsymbol{\xi}^N$. Then, at each node k, the *after-node sensitivity* is transformed into the **before-node sensitivity** (called **delta** in ref. [5]; see pages 325–326) $\delta_k^N \equiv \frac{\partial E}{\partial x_k^N}$ (defined as partial derivatives of E with respect to x_k^N, the before-node "net input" to node k) by multiplying by node-function derivatives as $\delta_k^N = f_k^{N\prime}(x_k^N)\xi_k^N$. The well-known *stagewise first-order BP* (i.e., **generalized delta rule**; see Eq.(14), p.326 in [5]) for intermediate stage s $(s=2,\cdots,N-1)$ can be written out with $\boldsymbol{\delta}$ or $\boldsymbol{\xi}$ as the recurrence relation below

$$\left\{\begin{array}{l} \underbrace{\boldsymbol{\delta}^s}_{P_s\times 1} \stackrel{\text{def}}{=} \underbrace{\frac{\partial E}{\partial \mathbf{x}^s}}_{P_s\times 1} = \underbrace{\mathbf{N}^{s,s+1^T}}_{P_s\times P_{s+1}} \underbrace{\boldsymbol{\delta}^{s+1}}_{P_{s+1}\times 1} = \left[\frac{\partial \mathbf{x}^{s+1}}{\partial \mathbf{x}^s}\right]^T \boldsymbol{\delta}^{s+1}, \\ \underbrace{\boldsymbol{\xi}^s}_{P_s\times 1} \stackrel{\text{def}}{=} \underbrace{\frac{\partial E}{\partial \mathbf{y}^s}}_{P_s\times 1} = \underbrace{\mathbf{W}^{s,s+1^T}}_{P_s\times P_{s+1}} \underbrace{\boldsymbol{\xi}^{s+1}}_{P_{s+1}\times 1} = \left[\frac{\partial \mathbf{y}^{s+1}}{\partial \mathbf{y}^s}\right]^T \boldsymbol{\xi}^{s+1}, \end{array}\right. \quad (1)$$

where E is a given certain objective function (to be minimized), and two P_{s+1}-by-P_s matrices, $\mathbf{N}^{s,s+1}$ and $\mathbf{W}^{s,s+1}$, are defined as $\mathbf{N}^{s,s+1} \stackrel{\text{def}}{=} \left[\frac{\partial \mathbf{x}^{s+1}}{\partial \mathbf{x}^s}\right]$ and $\mathbf{W}^{s,s+1} \stackrel{\text{def}}{=} \left[\frac{\partial \mathbf{y}^{s+1}}{\partial \mathbf{y}^s}\right]$.

These two are called before-node and after-node sensitivity transition matrices, respectively, for they *translate* the node-sensitivity vector $\boldsymbol{\delta}$ by $\mathbf{N}$ (and $\boldsymbol{\xi}$ by $\mathbf{W}$) from one stage to another; e.g., we can readily verify $\boldsymbol{\delta}^{s-1} = \mathbf{N}^{s-1,s^T} \mathbf{N}^{s,s+1^T} \boldsymbol{\delta}^{s+1} = \mathbf{N}^{s-1,s+1^T} \boldsymbol{\delta}^{s+1}$. Note that those two forms of sensitivity vectors become identical when node functions $f(.)$ are *linear identity* functions usually employed only at terminal layer N in MLP-learning.

The forward and backward passes in first-order stagewise BP for the standard MLP-learning can be summarized below:

$$\begin{array}{ll} \underline{\text{Forward pass:}} & \underline{\text{Backward pass:}} \\ \begin{cases} \underbrace{x_j^{s+1}}_{\text{scalar}} = \mathbf{y}_+^{s^T} \underbrace{\boldsymbol{\theta}_{\cdot,j}^{s,s+1}}_{(1+P_s)\times 1} \\ \quad \Updownarrow \ (\text{for } \ j=1,\dots,P_{s+1}) \\ \underbrace{\mathbf{x}^{s+1}}_{P_{s+1}\times 1} = \underbrace{\boldsymbol{\Theta}^{s,s+1}}_{P_{s+1}\times(1+P_s)} \underbrace{\mathbf{y}_+^{s}}_{(1+P_s)\times 1}, \end{cases} & \begin{cases} \underbrace{\xi_i^{s}}_{\text{scalar}} = \boldsymbol{\delta}^{s+1^T} \underbrace{\boldsymbol{\theta}_{i,\cdot}^{s,s+1}}_{P_{s+1}\times 1} \\ \quad \Updownarrow \ (\text{for } \ i=1,\dots,P_{s}) \\ \underbrace{\boldsymbol{\xi}^{s}}_{P_s\times 1} = \underbrace{\boldsymbol{\Theta}_{\text{void}}^{s,s+1^T}}_{P_s\times P_{s+1}} \underbrace{\boldsymbol{\delta}^{s+1}}_{P_{s+1}\times 1}. \end{cases} \end{array} \tag{2}$$

Here, $\mathbf{y}_+^s$ (with subscript $+$ on $\mathbf{y}^s$) includes a scalar *constant output* y_0^s of a *bias node* (denoted by node 0) at layer s leading to $\mathbf{y}_+^{s^T} = [y_0^s, \mathbf{y}^{s^T}]$, a $(1+P_s)$-vector of outputs at layer s; $\boldsymbol{\theta}_{i,\cdot}^{s,s+1}$ is a P_{s+1}-vector of the parameters linked to node i at layer s; $\boldsymbol{\theta}_{\cdot,j}^{s,s+1}$ is a $(1+P_s)$-vector of the parameters linked to node j at layer $s+1$ (including a threshold parameter linked to bias node 0 at layer s); $\boldsymbol{\Theta}^{s,s+1}$ in forward pass, a P_{s+1}-by-$(1+P_s)$ matrix of parameters between layers s and s+1, includes the P_{s+1} threshold parameters (i.e., the P_{s+1}-vector $\boldsymbol{\theta}_{0,\cdot}^{s,s+1}$) linked to bias node 0 at layer s in the first column, whereas $\boldsymbol{\Theta}_{\text{void}}^{s,s+1}$ in backward pass *excludes* the threshold parameters. Note that a matrix can always be reshaped into a vector for our convenience; for instance, $\boldsymbol{\Theta}^{s,s+1}$ can be reshaped to $\boldsymbol{\theta}^{s,s+1}$, an n_s-vector [$n_s \equiv (1+P_s)P_{s+1}$] of parameters, as shown next:

$$\begin{aligned} \underbrace{\boldsymbol{\Theta}^{s,s+1}}_{P_{s+1}\times(1+P_s)} &= \left[\underbrace{\boldsymbol{\theta}_{0,\cdot}^{s,s+1}}_{P_{s+1}\times 1} \,\middle|\, \underbrace{\boldsymbol{\Theta}_{\text{void}}^{s,s+1}}_{P_{s+1}\times P_s} \right] \\ &\Updownarrow \ \textit{Reshape} \\ \underbrace{\boldsymbol{\theta}^{s,s+1^T}}_{1\times n_s} &= \left[\theta_{0,1}^{s,s+1}, \theta_{1,1}^{s,s+1}, \dots, \theta_{P_s,1}^{s,s+1} \middle| \cdots \middle| \theta_{0,P_{s+1}}^{s,s+1}, \theta_{1,P_{s+1}}^{s,s+1}, \dots, \theta_{P_s,P_{s+1}}^{s,s+1} \right], \end{aligned} \tag{3}$$

where scalar $\theta_{i,k}^{s,s+1}$ denotes a parameter between node i at layer s and node k at layer $s+1$. At each stage s, the n_s-length gradient vector associated with $\boldsymbol{\theta}^{s,s+1}$ can be written as

$$\mathbf{g}^{s,s+1} = \left[\frac{\partial \mathbf{y}^{s+1}}{\partial \boldsymbol{\theta}^{s,s+1}} \right]^T \boldsymbol{\xi}^{s+1} = \left[\frac{\partial \mathbf{x}^{s+1}}{\partial \boldsymbol{\theta}^{s,s+1}} \right]^T \boldsymbol{\delta}^{s+1}, \tag{4}$$

where the transposed matrices are *sparse* in a *block-diagonal* form; for instance, see $\left[\frac{\partial \mathbf{y}^4}{\partial \boldsymbol{\theta}^{3,4}} \right]$ later in Eqs. (25) and (26). Yet, the stagewise computation by first-order BP can be viewed in such a way that the gradients are efficiently computed (without forming such *sparse* block-diagonal matrices explicitly) by the *outer product* $\boldsymbol{\delta}^{s+1} \mathbf{y}_+^{s^T}$, which produces a P_{s+1}-by-$(1+P_s)$ matrix $\mathbf{G}^{s,s+1}$ of gradients [7] associated with the same-sized matrix $\boldsymbol{\Theta}^{s,s+1}$ of parameters; here, column i of $\mathbf{G}^{s,s+1}$ is given as a P_{s+1}-vector $\mathbf{g}_{i,\cdot}^{s,s+1}$ for $\boldsymbol{\theta}_{i,\cdot}^{s,s+1}$. Again, the resulting gradient matrix $\mathbf{G}^{s,s+1}$ can be reshaped to an n_s-length gradient vector $\mathbf{g}^{s,s+1}$ in the same manner as shown in Eq. (3). Furthermore, the before-node sensitivity vector $\boldsymbol{\delta}^{s+1}$ (used to get $\mathbf{g}^{s,s+1}$ in the outer-product operation) is *backpropagated* by $\boldsymbol{\xi}^s = \boldsymbol{\Theta}_{\text{void}}^{s,s+1^T} \boldsymbol{\delta}^{s+1}$, as shown in Eq. (2)(right), rather than by Eq. (1) due to $\mathbf{N}^{s,s+1} = \boldsymbol{\Theta}_{\text{void}}^{s,s+1} \left[\frac{\partial \mathbf{y}^s}{\partial \mathbf{x}^s} \right]$; that is, stagewise "first-order" BP forms neither $\mathbf{N}$ nor $\mathbf{W}$ explicitly.

Such matrices as $\mathbf{N}^{s,s+1}$ for adjacent layers also play an important role as a vehicle for bucket-brigading second-order information [see later Eqs (10) and (11)] necessary to obtain the Hessian matrix $\mathbf{H}$. Stagewise second-order BP computes one block after another in the stagewise-partitioned $\mathbf{H}$ without forming $\mathbf{N}^{s,s+1}$ explicitly in the same way as stagewise first-order BP, which we shall describe next.

III. The Stagewise-Partitioned Hessian Matrix

Given an N-layered MLP, let the total number of parameters be denoted by $n = \sum_{s=1}^{N-1} n_s$, and let each n_s-by-n_t $\mathbf{H}^{s,t}$ block include Hessian elements with respect to pairs of one parameter at stage s [in the space of n_s parameters ($\boldsymbol{\theta}^{s,s+1}$ between layers s and s+1)] and another parameter at stage t [in the space of n_t parameters ($\boldsymbol{\theta}^{t,t+1}$ between layers t and t+1)]. Then, the n-by-n symmetric Hessian matrix $\mathbf{H}$ of a certain objective function E can be represented as a partitioned form among N layers (i.e., $N-1$ *decision stages*) in such a stagewise format as shown next:

$$\begin{array}{c} \longleftarrow \quad \longleftarrow \quad \longleftarrow \quad \longleftarrow \ \cdots \ \longleftarrow \text{Forward} \\ \downarrow \quad \textbf{Stage } N\text{-1} \quad \textbf{Stage } N\text{-2} \quad \textbf{Stage } N\text{-3} \quad \cdots \quad \textbf{Stage 1} \\ \text{Backward} \ \longrightarrow \quad \longrightarrow \quad \longrightarrow \ \cdots \ \longrightarrow \end{array}$$

$$\underbrace{\mathbf{H}}_{n\times n} = \left[\begin{array}{c|c|c|c|c} \mathbf{H}^{N\text{-}1,N\text{-}1} & \mathbf{H}^{N\text{-}2,N\text{-}1^T} & \mathbf{H}^{N\text{-}3,N\text{-}1^T} & \cdots & \mathbf{H}^{1,N\text{-}1^T} \\ \hline \mathbf{H}^{N\text{-}2,N\text{-}1} & \mathbf{H}^{N\text{-}2,N\text{-}2} & \mathbf{H}^{N\text{-}3,N\text{-}2^T} & \cdots & \mathbf{H}^{1,N\text{-}2^T} \\ \hline \mathbf{H}^{N\text{-}3,N\text{-}1} & \mathbf{H}^{N\text{-}3,N\text{-}2} & \mathbf{H}^{N\text{-}3,N\text{-}3} & \cdots & \mathbf{H}^{1,N\text{-}3^T} \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ \hline \mathbf{H}^{1,N\text{-}1} & \mathbf{H}^{1,N\text{-}2} & \mathbf{H}^{1,N\text{-}3} & \cdots & \mathbf{H}^{1,1} \end{array} \right]. \tag{5}$$

By symmetry, we need to form only the lower (or upper) triangular part of $\mathbf{H}$; totally $\frac{N(N-1)}{2}$ blocks including n_s-by-n_t rectangular "off-diagonal" blocks $\mathbf{H}^{s,t}$ $(1 \le s \le t \le N-1)$ as well as $N-1$ symmetric n_s-by-n_s square "diagonal" blocks $\mathbf{H}^{s,s}$ $(1 \le s \le N-1)$, of which we need only the lower half.

A. Stagewise second-order backpropagation

Stagewise second-order BP computes the entire Hessian matrix by one forward pass followed by backward processes *per training datum* in a stagewise block-by-block fashion. The Hessian blocks are computed from stage N-1 in a stagewise manner in the order of

$$\begin{array}{ccccccc} \underline{\text{Stage } N\text{-1}} & & \underline{\text{Stage } N\text{-2}} & & \underline{\text{Stage } N\text{-3}} & & \underline{\text{Stage 1}} \\ \mathbf{H}^{N-1,N-1} & \Rightarrow & \begin{cases} \mathbf{H}^{N-2,N-2} \\ \mathbf{H}^{N-2,N-1} \end{cases} & \Rightarrow & \begin{cases} \mathbf{H}^{N-3,N-3} \\ \mathbf{H}^{N-3,N-2} \\ \mathbf{H}^{N-3,N-1} \end{cases} & \Rightarrow \dots \Rightarrow & \begin{cases} \mathbf{H}^{1,1} \\ \mathbf{H}^{1,2} \\ \vdots \\ \mathbf{H}^{1,N-1}. \end{cases} \end{array}$$

In what follows, we describe algorithmic details step by step:

Algorithm: *Stagewise second-order BP* (per training datum).

(*Step* 0) Do forward pass from stage 1 to N to obtain node outputs, and evaluate the objective function value E.

(*Step* 1) At terminal stage N, compute $\boldsymbol{\xi}^N = \left[\frac{\partial E}{\partial \mathbf{y}^N}\right]$, the P_N-length *after-node sensitivity vector* (defined in Sec. II), and

$$\begin{aligned}\underbrace{\mathbf{Z}^N}_{P_N\times P_N} &= \left[\frac{\partial^2 E}{\partial \mathbf{x}^N \partial \mathbf{x}^N}\right] = \left[\frac{\partial \boldsymbol{\delta}^N}{\partial \mathbf{x}^N}\right] \\ &= \underbrace{\left[\frac{\partial \mathbf{y}^N}{\partial \mathbf{x}^N}\right]^T}_{P_N\times P_N} \underbrace{\left[\frac{\partial^2 E}{\partial \mathbf{y}^N \partial \mathbf{y}^N}\right]}_{P_N\times P_N} \underbrace{\left[\frac{\partial \mathbf{y}^N}{\partial \mathbf{x}^N}\right]}_{P_N\times P_N} + \underbrace{\left\langle \left[\frac{\partial^2 \mathbf{y}^N}{\partial \mathbf{x}^N \partial \mathbf{x}^N}\right], \boldsymbol{\xi}^N \right\rangle}_{P_N\times P_N}.\end{aligned} \tag{6}$$

The (i,j)-element of the last *symmetric* matrix is obtainable from the following special $\langle .,. \rangle$-operation (set $s\!=\!N$ below):

$$\left\langle \left[\frac{\partial^2 \mathbf{y}^s}{\partial \mathbf{x}^s \partial \mathbf{x}^s}\right], \boldsymbol{\xi}^s \right\rangle_{ij} \stackrel{\text{def}}{=} \sum_{k=1}^{P_s}\sum_{j=1}^{P_s}\sum_{i=1}^{P_s} \xi_k^s \left[\frac{\partial^2 y_k^s}{\partial x_i^s \partial x_j^s}\right], \tag{7}$$

which is just a diagonal matrix in standard MLP-learning.

• Repeat the following *Steps* 2 to 6, starting at stage $s\!=\!N\!-\!1$:
(*Step* 2) Obtain the diagonal Hessian block at stage s by

$$\underbrace{\mathbf{H}^{s,s}}_{n_s\times n_s} = \underbrace{\left[\frac{\partial \mathbf{x}^{s+1}}{\partial \boldsymbol{\theta}^{s,s+1}}\right]^T}_{n_s\times P_{s+1}} \underbrace{\mathbf{Z}^{s+1}}_{P_{s+1}\times P_{s+1}} \underbrace{\left[\frac{\partial \mathbf{x}^{s+1}}{\partial \boldsymbol{\theta}^{s,s+1}}\right]}_{P_{s+1}\times n_s}. \tag{8}$$

(*Step* 3) Only when $2 \leq N\!-\!s$ holds, obtain $(N\!-\!s\!-\!1)$ off-diagonal Hessian blocks by

$$\underbrace{\mathbf{H}^{s,s+t}}_{n_s\times n_{s+t}} = \underbrace{\left[\frac{\partial \mathbf{x}^{s+1}}{\partial \boldsymbol{\theta}^{s,s+1}}\right]^T}_{n_s\times P_{s+1}} \underbrace{\mathbf{F}^{s+1,s+t}}_{P_{s+1}\times n_{s+t}} \quad \text{for} \;\; t=1,\ldots,N\!-\!s\!-\!1, \tag{9}$$

where $\mathbf{F}^{s+1,s+t}$ is not needed initially when $s\!=\!N\!-\!1$; hence, defined later in Eq. (11).

• If $s\!=\!1$, then **terminate**; otherwise continue:

(*Step* 4) When $2 \leq N\!-\!s$ holds, update previously-computed rectangular matrices $\mathbf{F}^{s+1,u}$ for the next stage by:

$$\underbrace{\mathbf{F}^{s,u}_{\text{(new)}}}_{P_s\times n_u} \leftarrow \underbrace{\mathbf{N}^{s,s+1^T}}_{P_s\times P_{s+1}} \underbrace{\mathbf{F}^{s+1,u}_{\text{(old)}}}_{P_{s+1}\times n_u} \quad \text{for} \;\; u=s\!+\!1,\ldots,N\!-\!1. \tag{10}$$

(*Step* 5) Compute a new P_s-by-n_s rectangular matrix $\mathbf{F}^{s,s}$ at the current stage s by

$$\underbrace{\mathbf{F}^{s,s}}_{P_s\times n_s} = \underbrace{\left[\frac{\partial \mathbf{y}^s}{\partial \mathbf{x}^s}\right]^T}_{P_s\times P_s} \left\{ \underbrace{\boldsymbol{\Theta}^{s,s+1^T}_{\text{void}}}_{P_s\times P_{s+1}} \underbrace{\mathbf{Z}^{s+1}}_{P_{s+1}\times P_{s+1}} \underbrace{\left[\frac{\partial \mathbf{x}^{s+1}}{\partial \boldsymbol{\theta}^{s,s+1}}\right]}_{P_{s+1}\times n_s} + \underbrace{\left\langle \left[\frac{\partial \boldsymbol{\theta}^{s,s+1}_{\text{void}}}{\partial \boldsymbol{\theta}^{s,s+1}}\right], \boldsymbol{\delta}^{s+1} \right\rangle}_{P_s\times n_s} \right\}. \tag{11}$$

Here, $\boldsymbol{\theta}^{s,s+1}$ has its length $n_s\!=\!(1+P_s)P_{s+1}$ including P_{s+1} threshold parameters $\boldsymbol{\theta}^{s,s+1}_{0,.}$ linked to node 0 at layer s, whereas $\boldsymbol{\theta}^{s,s+1}_{\text{void}}$ has its length P_sP_{s+1} excluding the thresholds; these two vectors can be reshaped to $\boldsymbol{\Theta}^{s,s+1}$ and $\boldsymbol{\Theta}^{s,s+1}_{\text{void}}$, respectively, as shown in Eq. (3). The (i,j)-element of the last P_s-by-n_s rectangular matrix is obtainable from the following particular $\langle .,. \rangle$-operation [compare Eq. (7)]:

$$\left\langle \left[\frac{\partial \boldsymbol{\theta}^{s,s+1}_{\text{void}}}{\partial \boldsymbol{\theta}^{s,s+1}}\right], \boldsymbol{\delta}^{s+1} \right\rangle_{ij} \stackrel{\text{def}}{=} \sum_{k=1}^{P_{s+1}}\sum_{i=1}^{P_s}\sum_{l=0}^{P_s} \delta_k^{s+1} \left[\frac{\partial \theta^{s,s+1}_{i,k}}{\partial \theta^{s,s+1}_{l,k}}\right], \tag{12}$$

where index j is subject to $j=(1+P_s)(k-1)+l+1$.

(*Step* 6) Compute a P_s-by-P_s matrix $\mathbf{Z}^s$ by

$$\underbrace{\mathbf{Z}^s}_{P_s\times P_s} = \underbrace{\mathbf{N}^{s,s+1^T}}_{P_s\times P_{s+1}} \underbrace{\mathbf{Z}^{s+1}}_{P_{s+1}\times P_{s+1}} \underbrace{\mathbf{N}^{s,s+1}}_{P_{s+1}\times P_s} + \underbrace{\left\langle \left[\frac{\partial^2 \mathbf{y}^s}{\partial \mathbf{x}^s \partial \mathbf{x}^s}\right], \boldsymbol{\xi}^s \right\rangle}_{P_s\times P_s}, \tag{13}$$

where the last matrix is obtainable from the $\langle .,. \rangle$-operation defined in Eq. (7).

• Go back to *Step* 2 by setting $s\!=\!s\!-\!1$. ⋄(*End of Algorithm*)⋄

Remarks: The $\langle .,. \rangle$-operation defined in Eq. (12) yields a matrix of only first derivatives below:

$$\underbrace{\left\langle \left[\frac{\partial \boldsymbol{\theta}^{s,s+1}_{\text{void}}}{\partial \boldsymbol{\theta}^{s,s+1}}\right], \boldsymbol{\delta}^{s+1} \right\rangle}_{P_s\times n_s} = \Bigg[\underbrace{\delta_1^{s+1}\Big(\mathbf{0}\Big|\underbrace{\mathbf{I}}_{P_s\times P_s}\Big)}_{(1+P_s)\text{ columns}} \Bigg| \cdots \Bigg| \underbrace{\delta_k^{s+1}\Big(\mathbf{0}\Big|\underbrace{\mathbf{I}}_{P_s\times P_s}\Big)}_{(1+P_s)\text{ columns}} \Bigg| \cdots \Bigg| \underbrace{\delta_{P_{s+1}}^{s+1}\Big(\mathbf{0}\Big|\underbrace{\mathbf{I}}_{P_s\times P_s}\Big)}_{(1+P_s)\text{ columns}} \Bigg].$$

Here, the n_s-column space of the resulting P_s-by-n_s matrix has totally P_{s+1} partitions, each of which consists of $(1\!+\!P_s)$ columns since $n_s\!=\!(1\!+\!P_s)P_{s+1}$, and each partition has a P_s-vector of zeros, denoted by **0**, in the first column. The posed sparsity is tied to particular applications like MLP-learning.

B. Nodewise second-order backpropagation

In the NN literature, the best-known second-order BP is probably Bishop's method [1], [3], where for every node individually one must run a forward pass to the terminal output layer followed by a backward pass back to the node to get information necessary for Hessian elements; here, that node is one of the variables differentiated with repect to [for seeking *node sensitivity* in Eq. (1)]. This is what we call *nodewise BP*, a **nodewise** implementation of the *chain rule for differentiation*, which yields a Hessian element below with respect to two parameters $\theta^{s-1,s}_{i,j}$ and $\theta^{u-1,u}_{k,l}$ for $1 < s \leq u \leq N$ using $n^{s,u-1}_{j,k} \equiv \frac{\partial x_k^{u-1}}{\partial x_j^s}$ and $z^{s,u}_{j,l} \equiv \frac{\partial \delta_l^u}{\partial x_j^s}$ [cf. Eqs. (6) and (13)]:

$$\begin{aligned}\frac{\partial^2 E}{\partial \theta^{s-1,s}_{i,j}\partial \theta^{u-1,u}_{k,l}} &= \frac{\partial x_j^s}{\partial \theta^{s-1,s}_{i,j}} \frac{\partial}{\partial x_j^s}\left(\frac{\partial E}{\partial \theta^{u-1,u}_{k,l}}\right) \\ &= y_i^{s-1}\left[\frac{\partial (y_k^{u-1}\delta_l^u)}{\partial x_j^s}\right] \\ &= y_i^{s-1}\left[\frac{\partial y_k^{u-1}}{\partial x_j^s}\delta_l^u + y_k^{u-1}\frac{\partial \delta_l^u}{\partial x_j^s}\right] \\ &= y_i^{s-1}\left[\frac{\partial x_k^{u-1}}{\partial x_j^s}\frac{\partial y_k^{u-1}}{\partial x_k^{u-1}}\delta_l^u + y_k^{u-1}\frac{\partial \delta_l^u}{\partial x_j^s}\right] \\ &= y_i^{s-1}\left[n^{s,u-1}_{j,k} f_k^{u-1\prime}(x_k^{u-1})\delta_l^u + y_k^{u-1} z^{s,u}_{j,l}\right].\end{aligned} \tag{14}$$

This is Eq.(4.71), p. 155 in [2] rewritten with *stages* introduced and denoted by superscripts, and $n^{s,u-1}_{j,k}$ is the (k,j)-element of P_{u-1}-by-P_s matrix $\mathbf{N}^{s,u-1}$ in Eq. (1). The basic idea of Bishop's nodewise BP is as follows: Compute all the necessary quantities: δ_l^u by stagewise first-order BP, $n^{s,u-1}_{j,k}$ by forward pass, and $z^{s,u}_{j,l}$ by backward pass in advance; then, use Eq. (14) to evaluate Hessian elements. Unfortunately, this nodewise implementation of chain rules (14) does not exploit *stagewise structure* unlike first-order BP (see Section II); in addition, it has no implication about how to organize Hessian elements into a stagewise-partitioned "block-arrow" Hessian matrix [see Eq. (5) and Fig. 1]: To this end, it would be of much greater value to rewrite the *nodewise* algorithm posed by Bishop (outlined on p. 157 in [2]) in matrix form below.

Algorithm: *Nodewise second-order BP* (per training datum).

(*Step* 0) Do forward pass from stage 1 to stage N.

(*Step* 1) Initialize $\mathbf{N}^{s,s}=\mathbf{I}$ (identity matrix) and $\mathbf{N}^{u,s}=\mathbf{0}$ (matrix of zeros) for $1<s<u\leq N$ (see pages 155 & 156 in [2] for this particular initialization), and then do *forward pass* to obtain a P_t-by-P_s non-zero dense matrix $\mathbf{N}^{s,t}$ (for $s<t; s=2,\cdots,N-1$) by the following computation:

$$n_{j,l}^{s,t}=\sum_{k=1}^{P_{t-1}} f_k^{t-1\prime}(x_k^{t-1})\theta_{k,l}^{t-1,t} n_{j,k}^{s,t-1} \Longleftrightarrow \underbrace{\mathbf{N}^{s,t}}_{P_t\times P_s}=\underbrace{\mathbf{N}^{t-1,t}}_{P_t\times P_{t-1}}\underbrace{\mathbf{N}^{s,t-1}}_{P_{t-1}\times P_s}. \quad (15)$$

(*Step* 2) At terminal stage N, compute $\boldsymbol{\delta}^N=\left[\frac{\partial E}{\partial \mathbf{x}^N}\right]$, the P_N-length before-node sensitivity vector, and matrix $\mathbf{Z}^N$ [defined in Eq. (6)], and then obtain the following for $2\leq s\leq N$:

$$z_{j,l}^{s,N}=\sum_{m=1}^{P_N} n_{j,m}^{s,N}\left(\frac{\partial^2 E}{\partial x_l^N \partial x_m^N}\right) \Longleftrightarrow \underbrace{\mathbf{Z}^{s,N}}_{P_s\times P_N}=\underbrace{\mathbf{N}^{s,N^T}}_{P_s\times P_N}\underbrace{\mathbf{Z}^{N}}_{P_N\times P_N}. \quad (16)$$

(*Step* 3) Compute $\boldsymbol{\delta}^s$ using first-order BP: $\xi_k^t=\sum_{l=1}\theta_{k,l}^{t,t+1}\delta_l^{t+1}$ in Eq.(2)(right) and obtain the next for $1<s\leq t<N$:

$$z_{j,k}^{s,t}=n_{j,k}^{s,t} f_k^{t\,\prime\prime}(x_k^t)\sum_{l=1}^{P_{t+1}}\theta_{k,l}^{t,t+1}\delta_l^{t+1} + f_k^{t\,\prime}(x_k^t)\sum_{l=1}^{P_{t+1}}\theta_{k,l}^{t,t+1} z_{j,l}^{s,t+1},$$

$$\Longleftrightarrow \underbrace{\mathbf{Z}^{s,t}}_{P_s\times P_t}=\underbrace{\mathbf{N}^{s,t^T}}_{P_s\times P_t}\underbrace{\left\langle\left[\frac{\partial^2 \mathbf{y}^t}{\partial \mathbf{x}^t \partial \mathbf{x}^t}\right],\boldsymbol{\xi}^t\right\rangle}_{P_t\times P_t}+\underbrace{\mathbf{Z}^{s,t+1}}_{P_s\times P_{t+1}}\underbrace{\mathbf{N}^{t,t+1}}_{P_{t+1}\times P_t}. \quad (17)$$

(*Step* 4) Evaluate the Hessian blocks by Eq. (14) in matrix form for $1<s\leq u\leq N$:

$$\underbrace{\mathbf{H}^{s-1,u-1}}_{n_{s-1}\times n_{u-1}}=\underbrace{\left[\frac{\partial \mathbf{x}^s}{\partial \boldsymbol{\theta}^{s-1,s}}\right]^T}_{n_{s-1}\times P_s}\left(\underbrace{\mathbf{N}^{s,u-1^T}}_{P_s\times P_{u-1}}\underbrace{\left[\frac{\partial \mathbf{y}^{u-1}}{\partial \mathbf{x}^{u-1}}\right]^T}_{P_{u-1}\times P_{u-1}}\underbrace{\left\langle\left[\frac{\partial \boldsymbol{\theta}_{\text{void}}^{u-1,u}}{\partial \boldsymbol{\theta}^{u-1,u}}\right],\boldsymbol{\delta}^u\right\rangle}_{P_{u-1}\times n_{u-1}} + \underbrace{\mathbf{Z}^{s,u}}_{P_s\times P_u}\underbrace{\left[\frac{\partial \mathbf{x}^u}{\partial \boldsymbol{\theta}^{u-1,u}}\right]}_{P_u\times n_{u-1}}\right), \quad (18)$$

where Eq. (12) is used for evaluating a $\langle.,.\rangle$-term. ⋄(*End*)⋄

Remarks: Eqs. (15), (16), and (17) correspond to Eqs.(4.75), (4.79), and (4.78), respectively, on pages 155 & 156 in ref. [2].

IV. Two hidden-layer MLP learning

In optimal control, N, the number of stages, is arbitrarily large. In MLP-learning, however, use of merely one or two hidden layers is by far the most popular at this stage. For this reason, we consider standard two-hidden-layer MLP-learning. This is a four-stage (N=4; three *decision stages* plus a terminal stage) problem, in which the total number of parameters (or decision variables) is given as: $n=n_3+n_2+n_1=P_4(1+P_3)+P_3(1+P_2)+P_2(1+P_1)$ (including *threshold parameters*). In this setting, we have a three-block by three-block stagewise **symmetric** Hessian matrix $\mathbf{H}$ in a nine-block partitioned form below as well as a three-block-partitioned gradient vector $\mathbf{g}$ defined in Eq. (4):

$$\mathbf{H}=\begin{bmatrix}\mathbf{H}^{3,3} & \mathbf{H}^{2,3^T} & \mathbf{H}^{1,3^T}\\ \mathbf{H}^{2,3} & \mathbf{H}^{2,2} & \mathbf{H}^{1,2^T}\\ \mathbf{H}^{1,3} & \mathbf{H}^{1,2} & \mathbf{H}^{1,1}\end{bmatrix},\quad \mathbf{g}=\begin{bmatrix}\mathbf{g}^{3,4}\\ \mathbf{g}^{2,3}\\ \mathbf{g}^{1,2}\end{bmatrix}. \quad (19)$$

Here, we need to form three off-diagonal blocks and only the lower (or upper) triangular part of three diagonal blocks; totally, six blocks $\mathbf{H}^{s,t}$ $(1\leq s\leq t\leq 3)$. Each block $\mathbf{H}^{s,t}$ includes Hessian elements with respect to pairs of one parameter at stage s and another at stage t.

A. Algorithmic behaviors

We describe how our version of nodewise second-order BP algorithm in Section III-B works:

(*Step* 1): By initialization, set $\mathbf{N}^{4,4}=\mathbf{I}$, $\mathbf{N}^{3,3}=\mathbf{I}$, $\mathbf{N}^{2,2}=\mathbf{I}$, $\mathbf{N}^{4,3}=\mathbf{0}$, $\mathbf{N}^{4,2}=\mathbf{0}$, and $\mathbf{N}^{3,2}=\mathbf{0}$. By forward pass in Eq. (15), get three dense blocks: $\mathbf{N}^{2,3}$, $\mathbf{N}^{3,4}$, and $\mathbf{N}^{2,4}=\mathbf{N}^{3,4}\mathbf{N}^{2,3}$.

(*Step* 2): Get $\mathbf{Z}^4$ by Eq. (6) and $\mathbf{Z}^{4,4}=\mathbf{N}^{4,4}\mathbf{Z}^4$ by Eq. (16); similarly, obtain $\mathbf{Z}^{3,4}$ and $\mathbf{Z}^{2,4}$ as well.

(*Step* 3): Use Eq. (17) to get $\mathbf{Z}^{3,3}$, $\mathbf{Z}^{2,3}$, and $\mathbf{Z}^{2,2}$; for instance, by $\mathbf{Z}^{3,3}=\mathbf{N}^{3,3^T}\left\langle\left[\frac{\partial^2 \mathbf{y}^3}{\partial \mathbf{x}^3 \partial \mathbf{x}^3}\right],\boldsymbol{\xi}^3\right\rangle+\mathbf{Z}^{3,4}\mathbf{N}^{3,4}$.

(*Step* 4): Use Eq. (18) [i.e., Eq. (14)] to obtain the desired six Hessian blocks.

All those nine $\mathbf{N}$ blocks can be pictured in an *augmented* "upper triangular" before-node sensitivity transition matrix $\widetilde{\mathbf{N}}$ defined below together with $\widetilde{\mathbf{x}}$, a $\widetilde{P}$-dimensional augmented vector, which consists of all the before-node net-inputs per datum at three layers except the first input layer $(N=1)$ because $\mathbf{x}^1$ is a *fixed* vector of given inputs; hence, $\widetilde{P}\equiv P_4+P_3+P_2$:

$$\underbrace{\widetilde{\mathbf{N}}}_{\widetilde{P}\times\widetilde{P}}\overset{\text{def}}{=}\left[\frac{\partial\widetilde{\mathbf{x}}}{\partial\widetilde{\mathbf{x}}}\right]=\left[\begin{array}{c|c|c}\underbrace{\mathbf{I}_{P_4\times P_4}}_{=\mathbf{N}^{4,4}} & \underbrace{\mathbf{N}^{3,4}}_{P_4\times P_3} & \underbrace{\mathbf{N}^{2,4}}_{P_4\times P_2}\\ \hline \underbrace{\mathbf{0}_{P_3\times P_4}}_{=\mathbf{N}^{4,3}} & \underbrace{\mathbf{I}_{P_3\times P_3}}_{=\mathbf{N}^{3,3}} & \underbrace{\mathbf{N}^{2,3}}_{P_3\times P_2}\\ \hline \underbrace{\mathbf{0}_{P_2\times P_4}}_{=\mathbf{N}^{4,2}} & \underbrace{\mathbf{0}_{P_2\times P_3}}_{=\mathbf{N}^{3,2}} & \underbrace{\mathbf{I}_{P_2\times P_2}}_{=\mathbf{N}^{2,2}}\end{array}\right];\quad \underbrace{\widetilde{\mathbf{x}}}_{\widetilde{P}\times 1}\overset{\text{def}}{=}\left[\begin{array}{c}\mathbf{x}^4\\ \hline \mathbf{x}^3\\ \hline \mathbf{x}^2\end{array}\right]. \quad (20)$$

Here, three diagonal identity blocks $\mathbf{I}$ correspond to $\mathbf{N}^{4,4}$, $\mathbf{N}^{3,3}$, and $\mathbf{N}^{2,2}$. At first glance, Bishop's nodewise BP relies on using $\widetilde{\mathbf{N}}$ explicitly, requiring $\mathbf{N}^{s,t}$ even for non-adjacent layers $(s+1<t)$ as well as identity blocks $\mathbf{N}^{s,s}$ and zero blocks. For adjacent blocks $\mathbf{N}^{s,s+1}$, Eq. (15) just implies *multiply by an identity matrix*; hence, no need to use it in reality. Likewise, at Step 2, $\mathbf{Z}^{4,4}=\mathbf{Z}^4$ due to $\mathbf{N}^{4,4}=\mathbf{I}$. Furthermore, in Eq. (18), $\mathbf{N}^{4,3}=\mathbf{0}$ and $\mathbf{N}^{3,2}=\mathbf{0}$ (matrices of zeros) are used when diagonal blocks $\mathbf{H}^{s,s}$ are evaluated (but $\mathbf{N}^{4,2}=\mathbf{0}$ is not needed at all). In this way, nodewise BP yields Hessian blocks by Eq. (18), a matrix form of Eq. (14), as long as $\widetilde{\mathbf{N}}$ in Eq. (20) is obtained correctly in advance by *forward* pass at Step 1 (according to pp.155–156 in [2]); yet, it is not very efficient to work on such zero entries and multiply by one.

On the other hand, stagewise second-order BP evaluates $\mathbf{N}^{s,s+1}$ *implicitly* only for adjacent layers during the *backward* process (not by forward pass) essentially in the same manner as stagewise first-order BP does with no $\mathbf{N}^{s,s+1}$ blocks required explicitly, and thus avoids operating on such zeros and ones [for Eq. (20)]. For *off-diagonal* Hessian blocks $\mathbf{H}^{s,u}$ $(s < u)$, the parenthesized terms in Eq. (18) become the rectangular matrix $\mathbf{F}^{s,u-1}$ in Eq. (11). That is, stagewise BP splits Eq. (18) into Eqs. (8) and (9) by exploitation of the stagewise MLP structure.

B. Separable Hessian Structures

We next show the Hessian-block structures to be **separable** into several portions. Among the six distinct blocks in Eq. (19), due to space limitation we display below three Hessian blocks: two diagonal blocks and one off-diagonal block alone.

$$\underbrace{\mathbf{H}^{3,3}}_{n_3\times n_3} = \underbrace{\left[\frac{\partial \mathbf{X}^4}{\partial \boldsymbol{\theta}^{3,4}}\right]^T}_{n_3\times P_4} \underbrace{\left[\frac{\partial \mathbf{y}^4}{\partial \mathbf{x}^4}\right]^T}_{P_4\times P_4} \underbrace{\left[\frac{\partial^2 E}{\partial \mathbf{y}^4 \partial \mathbf{y}^4}\right]}_{P_4\times P_4} \underbrace{\left[\frac{\partial \mathbf{y}^4}{\partial \mathbf{x}^4}\right]}_{P_4\times P_4} \underbrace{\left[\frac{\partial \mathbf{X}^4}{\partial \boldsymbol{\theta}^{3,4}}\right]}_{P_4\times n_3} + \underbrace{\left[\frac{\partial \mathbf{X}^4}{\partial \boldsymbol{\theta}^{3,4}}\right]^T}_{n_3\times P_4} \underbrace{\left\langle\left[\frac{\partial^2 \mathbf{y}^4}{\partial \mathbf{x}^4 \partial \mathbf{x}^4}\right], \boldsymbol{\xi}^4\right\rangle}_{P_4\times P_4} \underbrace{\left[\frac{\partial \mathbf{X}^4}{\partial \boldsymbol{\theta}^{3,4}}\right]}_{P_4\times n_3}; \tag{21}$$

$$\begin{aligned}\underbrace{\mathbf{H}^{1,1}}_{n_1\times n_1} &= \underbrace{\left[\frac{\partial \mathbf{X}^2}{\partial \boldsymbol{\theta}^{1,2}}\right]^T}_{n_1\times P_2} \underbrace{\mathbf{N}^{2,3^T}}_{P_2\times P_3} \underbrace{\mathbf{N}^{3,4^T}}_{P_3\times P_4} \underbrace{\left[\frac{\partial \mathbf{y}^4}{\partial \mathbf{x}^4}\right]^T}_{P_4\times P_4} \underbrace{\left[\frac{\partial^2 E}{\partial \mathbf{y}^4 \partial \mathbf{y}^4}\right]}_{P_4\times P_4} \underbrace{\left[\frac{\partial \mathbf{y}^4}{\partial \mathbf{x}^4}\right]}_{P_4\times P_4} \underbrace{\mathbf{N}^{3,4}}_{P_4\times P_3} \underbrace{\mathbf{N}^{2,3}}_{P_3\times P_2} \underbrace{\left[\frac{\partial \mathbf{X}^2}{\partial \boldsymbol{\theta}^{1,2}}\right]}_{P_2\times n_1} \\ &+ \underbrace{\left[\frac{\partial \mathbf{X}^2}{\partial \boldsymbol{\theta}^{1,2}}\right]^T}_{n_1\times P_2} \underbrace{\mathbf{N}^{2,3^T}}_{P_2\times P_3} \underbrace{\mathbf{N}^{3,4^T}}_{P_3\times P_4} \underbrace{\left\langle\left[\frac{\partial^2 \mathbf{y}^4}{\partial \mathbf{x}^4 \partial \mathbf{x}^4}\right], \boldsymbol{\xi}^4\right\rangle}_{P_4\times P_4} \underbrace{\mathbf{N}^{3,4}}_{P_4\times P_3} \underbrace{\mathbf{N}^{2,3}}_{P_3\times P_2} \underbrace{\left[\frac{\partial \mathbf{X}^2}{\partial \boldsymbol{\theta}^{1,2}}\right]}_{P_2\times n_1} \\ &+ \underbrace{\left[\frac{\partial \mathbf{X}^2}{\partial \boldsymbol{\theta}^{1,2}}\right]^T}_{n_1\times P_2} \underbrace{\mathbf{N}^{2,3^T}}_{P_2\times P_3} \underbrace{\left\langle\left[\frac{\partial^2 \mathbf{y}^3}{\partial \mathbf{x}^3 \partial \mathbf{x}^3}\right], \boldsymbol{\xi}^3\right\rangle}_{P_3\times P_3} \underbrace{\mathbf{N}^{2,3}}_{P_3\times P_2} \underbrace{\left[\frac{\partial \mathbf{X}^2}{\partial \boldsymbol{\theta}^{1,2}}\right]}_{P_2\times n_1} \\ &+ \underbrace{\left[\frac{\partial \mathbf{X}^2}{\partial \boldsymbol{\theta}^{1,2}}\right]^T}_{n_1\times P_2} \underbrace{\left\langle\left[\frac{\partial^2 \mathbf{y}^2}{\partial \mathbf{x}^2 \partial \mathbf{x}^2}\right], \boldsymbol{\xi}^2\right\rangle}_{P_2\times P_2} \underbrace{\left[\frac{\partial \mathbf{X}^2}{\partial \boldsymbol{\theta}^{1,2}}\right]}_{P_2\times n_1};\end{aligned} \tag{22}$$

$$\begin{aligned}\underbrace{\mathbf{H}^{1,2}}_{n_1\times n_2} &= \underbrace{\left[\frac{\partial \mathbf{X}^2}{\partial \boldsymbol{\theta}^{1,2}}\right]^T}_{n_1\times P_2} \underbrace{\mathbf{N}^{2,3^T}}_{P_2\times P_3} \underbrace{\mathbf{N}^{3,4^T}}_{P_3\times P_4} \underbrace{\left[\frac{\partial \mathbf{y}^4}{\partial \mathbf{x}^4}\right]^T}_{P_4\times P_4} \underbrace{\left[\frac{\partial^2 E}{\partial \mathbf{y}^4 \partial \mathbf{y}^4}\right]}_{P_4\times P_4} \underbrace{\left[\frac{\partial \mathbf{y}^4}{\partial \mathbf{x}^4}\right]}_{P_4\times P_4} \underbrace{\mathbf{N}^{3,4}}_{P_4\times P_3} \underbrace{\left[\frac{\partial \mathbf{X}^3}{\partial \boldsymbol{\theta}^{2,3}}\right]}_{P_3\times n_2} \\ &+ \underbrace{\left[\frac{\partial \mathbf{X}^2}{\partial \boldsymbol{\theta}^{1,2}}\right]^T}_{n_1\times P_2} \underbrace{\mathbf{N}^{2,3^T}}_{P_2\times P_3} \underbrace{\mathbf{N}^{3,4^T}}_{P_3\times P_4} \underbrace{\left\langle\left[\frac{\partial^2 \mathbf{y}^4}{\partial \mathbf{x}^4 \partial \mathbf{x}^4}\right], \boldsymbol{\xi}^4\right\rangle}_{P_4\times P_4} \underbrace{\mathbf{N}^{3,4}}_{P_4\times P_3} \underbrace{\left[\frac{\partial \mathbf{X}^3}{\partial \boldsymbol{\theta}^{2,3}}\right]}_{P_3\times n_2} \\ &+ \underbrace{\left[\frac{\partial \mathbf{X}^2}{\partial \boldsymbol{\theta}^{1,2}}\right]^T}_{n_1\times P_2} \underbrace{\mathbf{N}^{2,3^T}}_{P_2\times P_3} \underbrace{\left\langle\left[\frac{\partial^2 \mathbf{y}^3}{\partial \mathbf{x}^3 \partial \mathbf{x}^3}\right], \boldsymbol{\xi}^3\right\rangle}_{P_3\times P_3} \underbrace{\left[\frac{\partial \mathbf{X}^3}{\partial \boldsymbol{\theta}^{2,3}}\right]}_{P_3\times n_2} \\ &+ \underbrace{\left[\frac{\partial \mathbf{X}^2}{\partial \boldsymbol{\theta}^{1,2}}\right]^T}_{n_1\times P_2} \underbrace{\left[\frac{\partial \mathbf{y}^2}{\partial \mathbf{X}^2}\right]^T}_{P_2\times P_2} \underbrace{\left\langle\left[\frac{\partial \boldsymbol{\theta}^{2,3}_{\text{void}}}{\partial \boldsymbol{\theta}^{2,3}}\right], \boldsymbol{\delta}^3\right\rangle}_{P_2\times n_2}.\end{aligned} \tag{23}$$

In ordinary MLP-learning with multiple terminal outputs $(P_4 > 1)$, only $C_A (\equiv 1 + P_3 = \frac{n_3}{P_4})$ parameters, a subset of the n_3 *terminal parameters* $\boldsymbol{\theta}^{3,4}$, contribute to each terminal output: At some node k at layer 4, for instance, only C_A parameters $\boldsymbol{\theta}^{3,4}_{.,k}$ influence output y^4_k, whereas the other $(n_3 - C_A)$ parameters $\boldsymbol{\theta}^{3,4}_{.,j}$ $(k \neq j;\ j = 1, ..., P_4)$ have no effect on it. Therefore, the first diagonal Hessian block $\mathbf{H}^{3,3}$ in Eq. (21) [i.e., $\mathbf{H}^{N\text{-}1,N\text{-}1}$ placed at the upper-left corner in Eq. (5)] always becomes *block-diagonal* (with P_4 sub-blocks $\times$) below:

$$\underbrace{\mathbf{H}^{3,3}}_{n_3\times n_3} = \begin{bmatrix} \times_1 & & & \\ & \times_2 & & \\ & & \ddots & \\ & & & \times_{P_4} \end{bmatrix}, \tag{24}$$

where $\times_k$ denotes a C_A-by-C_A dense symmetric sub-block. In consequence, the entire Hessian matrix becomes a block-arrow form; see later the front panel in Fig. 1. In addition, if *linear identity* node functions are employed at the terminal layer (hence, $\mathbf{y}^4 = \mathbf{x}^4$), then all the diagonal sub-blocks in Eq. (24) become identical; so, need to store only half of one sub-block due to symmetry. In such a case, it is clear from our *separable representations* of the Hessian blocks that the second term on the right-hand side of Eqs (21), (22), and (23) will disappear because $\left[\frac{\partial^2 \mathbf{y}^4}{\partial \mathbf{x}^4 \partial \mathbf{x}^4}\right]$ reduces to a matrix of zeros.

Furthermore, the last term in $\mathbf{H}^{1,2}$ is a *sparse* matrix of only first derivatives due to Eq. (12); in the next section, we shall explain this finding in nonlinear least squares learning.

C. Neural Networks Nonlinear Least Squares Learning

When our objective function E is the sum over all the d training data of squared residuals, we have $E(\boldsymbol{\theta}) = \frac{1}{2}\mathbf{r}^T\mathbf{r}$, where $\mathbf{r} \equiv \mathbf{y}^4(\boldsymbol{\theta}) - \mathbf{t}$; in words, an m-vector $\mathbf{r}$ of *residuals* is the difference between an m-vector $\mathbf{t}$ of the desired outputs and an m-vector $\mathbf{y}^4$ of the terminal outputs of a two hidden-layer MLP (with $N=4$), and $m \equiv P_4 d$ $(P_4 > 1$, or multiple terminal outputs in general). The gradient vector of E is given by $\mathbf{g} = \mathbf{J}^T\mathbf{r}$; here, $\mathbf{J}$ denotes the m-by-n Jacobian matrix $\mathbf{J}$ of the residual vector $\mathbf{r}$, which is $\mathbf{J}$ of $\mathbf{y}^4$ because $\mathbf{t}$ is independent of $\boldsymbol{\theta}$ by assumption. As shown in Eqs. (19)(right) and (4), $\mathbf{g}$ is stagewise-partitioned as: $\mathbf{g}^{s,s+1} = \left[\frac{\partial \mathbf{y}^{s+1}}{\partial \boldsymbol{\theta}^{s,s+1}}\right]^T \boldsymbol{\xi}^{s+1}$ for $s = 1, ..., 3$, where $\boldsymbol{\xi}^4 = \mathbf{r}$. Likewise, $\mathbf{J}$ can be given in *stagewise column-partitioned form* below in Eq. (25), or equivalently in *block-angular form* below in Eq. (26) [with $n_B \equiv n - n_3 = n_1 + n_2$]:

$$\underbrace{\mathbf{J}}_{m\times n} = \Bigg[\underbrace{\frac{\partial \mathbf{y}^4}{\partial \boldsymbol{\theta}^{3,4}}}_{m\times n_3} \ \underbrace{\Bigg|\ \underbrace{\frac{\partial \mathbf{y}^4}{\partial \boldsymbol{\theta}^{2,3}}}_{m\times n_2} \ \Bigg|\ \underbrace{\frac{\partial \mathbf{y}^4}{\partial \boldsymbol{\theta}^{1,2}}}_{m\times n_1}}_{\text{dense } n_B \text{ columns}} \Bigg] \tag{25}$$

$$= \left[\begin{array}{cccc|c} \mathbf{A}_1 & & & & \mathbf{B}_1 \\ & \mathbf{A}_2 & & & \mathbf{B}_2 \\ & & \ddots & & \vdots \\ & & & \mathbf{A}_{P_4} & \mathbf{B}_{P_4} \end{array}\right] \quad (\text{dense } n_B \text{ columns}) \tag{26}$$

where $\mathbf{A}_k$ is d-by-C_A $(k = 1, ..., P_4)$ and $\mathbf{B}_k$ d-by-n_B. The *block-angular* form is due to the same reason as Eq. (24); i.e., only C_A parameters affect each terminal residual. Since $\mathbf{J}$ has the block-angular form in Eq. (26), its cross-product matrix $\mathbf{J}^T\mathbf{J}$ has a so-called *block-arrow* form due to its appearance, as illustrated in Fig. 1, where $\mathbf{H} = \mathbf{J}^T\mathbf{J}$ and $\mathbf{H}^{3,3}$ in Eqs (21) and (24) consists of P_4 diagonal blocks $\mathbf{A}_k^T\mathbf{A}_k$ for $k = 1, ..., P_4$. If the terminal node functions are the linear

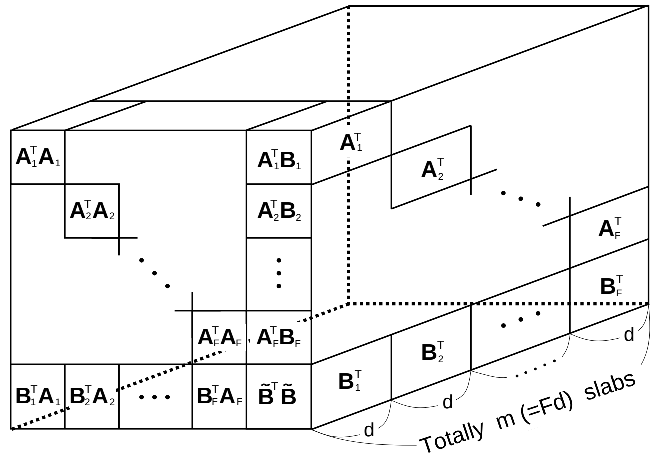

Fig. 1. The front square panel shows the block-arrow Gauss-Newton Hessian matrix $\mathbf{J}^T\mathbf{J}$ obtainable from the sum of $m(\equiv Fd)$ slabs over all the d training data with a multiple $F(\equiv P_N)$-output multilayer-perceptron (MLP) model; here, the lower-right block of the Hessian is: $\widetilde{\mathbf{B}}^T\widetilde{\mathbf{B}} \equiv \sum_{k=1}^{F} \mathbf{B}_k^T\mathbf{B}_k$, and the right-front rectangular panel depicts the transposed block-angular residual Jacobian matrix $\mathbf{J}^T$ [in Eq.(26)]. The ith slab ($i=1,...,m$) consists of four rank-one blocks: $\mathbf{A}_k^T\mathbf{A}_k$, $\mathbf{A}_k^T\mathbf{B}_k$, $\mathbf{B}_k^T\mathbf{A}_k$, and $\mathbf{B}_k^T\mathbf{B}_k$, resulting from the kth residual $r_{k,p}$ computed at node k ($k=1,...,F$) at terminal layer on datum p ($p=1,...,d$); hence, the relation $i=(k-1)d+p$. In standard MLP-learning, the full Hessian $\mathbf{H}$ (e.g., $\mathbf{J}^T\mathbf{J}+\mathbf{S}$) also has the same block-arrow form because $\mathbf{H}^{N\text{-}1,N\text{-}1}$ in Eq.(5) is block-diagonal;e.g., see Eq.(24).

identity function, then all the diagonal blocks $\mathbf{A}_k$ become identical; so do $\mathbf{A}_k^T\mathbf{A}_k$, as described after Eq. (24).

Since $E(\boldsymbol{\theta}) = \frac{1}{2}\mathbf{r}^T\mathbf{r}$, matrix $\left[\frac{\partial^2 E}{\partial \mathbf{y}^4 \partial \mathbf{y}^4}\right]$ in Eq. (6) reduces to the identity matrix $\mathbf{I}$; therefore, the *full* Hessian can be given as $\mathbf{H} = \mathbf{J}^T\mathbf{J} + \mathbf{S}$, where $\mathbf{J}^T\mathbf{J}$ is a matrix of *only first derivatives* (called the *Gauss-Newton Hessian* in Fig. 1), the first term on the right hand side of Eqs. (21) to (23), and $\mathbf{S}$ is a matrix of second derivatives, the rest of right-hand-side terms in those equations. Intriguingly, in *off-diagonal* Hessian blocks $\mathbf{H}^{s,t} = \left[\mathbf{J}^T\mathbf{J}\right]^{s,t} + \mathbf{S}^{s,t}$ $(s<t)$, we can further pull $\mathbf{T}^{s,t}$, a *sparse matrix of only first derivatives*, out of $\mathbf{S}^{s,t}$ as $\mathbf{H}^{s,t} = \left(\left[\mathbf{J}^T\mathbf{J}\right]^{s,t} + \mathbf{T}^{s,t}\right) + \left(\mathbf{S}^{s,t} - \mathbf{T}^{s,t}\right)$, where we have

$$\mathbf{T}^{s,t} = \left[\frac{\partial \mathbf{y}^t}{\partial \boldsymbol{\theta}^{s,s+1}}\right]^T \left\langle \left[\frac{\partial \boldsymbol{\theta}^{t,t+1}_{\text{void}}}{\partial \boldsymbol{\theta}^{t,t+1}}\right], \boldsymbol{\delta}^{t+1} \right\rangle. \tag{27}$$

For instance, $\mathbf{T}^{1,2}$ is the last term of $\mathbf{H}^{1,2}$ [see Eq. (23)], obtainable from Eq. (12).

V. Conclusion and Future Directions

Given a general objective function arising in multi-stage NN-learning, we have described in matrix form both stagewise second-order BP and our version of nodewise second-order BP with a particular emphasis on how to organize Hessian elements into the stagewise-partitioned "block-arrow" Hessian matrix $\mathbf{H}$ (*with its arrow-head pointing downwards to the right*; see pp. 83–90 in [4]), as illustrated in Fig. 1, so as to exploit inevitable *sparsity* [9] when $P_N > 1$ (i.e., multiple terminal outputs). In more elaborate MLP-learning, one may introduce direct connections between the first input and the terminal layers; this increases C_A, the diagonal sub-block size in $\mathbf{H}^{N\text{-}1,N\text{-}1}$ [see Eq. (24)], leading to a very nice block-arrow form. On the other hand, such nice sparsity may disappear when *weight-sharing* and *weight-pruning* are applied (as usual in optimal control [8]) so that all the *terminal parameters* $\boldsymbol{\theta}^{N\text{-}1,N}$ are shared among the terminal states $\mathbf{y}^N$. In this way, MLP-learning exhibits a great deal of structure.

For the parameter optimization, we recommend *trust-region* globalization, which works even if $\mathbf{H}$ is *indefinite* [10], [9]. In large-scale problems, where $\mathbf{H}$ may not be needed explicitly, we could use sparse Hessian matrix-vector multiply (e.g., [11]) to construct *Krylov subspaces* for optimization purposes, but it is still worth exploiting sparsity of $\mathbf{H}$ for pre-conditioning [10]. In this context, it is not recommendable to compute (or approximate) the *inverse* matrix of (sparse) block-arrow $\mathbf{H}$ (see Fig. 1) because it always becomes *dense*.

Our matrix-based algorithms revealed that blocks in the stagewise-partitioned $\mathbf{H}$ are separable into several distinct portions, and disclosed that sparse matrices of only first derivatives [see Eq. (27)] can be further identified. Furthermore, by inspection of the common matrix terms in block [e.g., see Eqs. (21) to (23)], we see that the Hessian part computed on each datum at stage s, which consists of blocks $\mathbf{H}^{s,t}$ $(1 \leq s \leq t \leq N-1)$, is at most *rank* P_{s+1}, where P_{s+1} denotes the number of nodes at layer $s+1$. We plan to report in another opportunity more on those findings as well as the practical implementation issues of stagewise second-order BP, for which the matrix recursive formulas may allow us to take advantage of level-3 BLAS (Basic Linear Algebra Subprograms; see http://www.netlib.org/blas/).

References

[1] Christopher M. Bishop. "Exact calculation of the Hessian matrix for the multilayer perceptron." In *Neural Computation*. pages 494–501, Vol.4, No. 4, 1992.

[2] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford Press, 1995.

[3] Wray L. Buntine and Andreas S. Weigend. "Computing Second Derivatives in Feed-Forward Networks: A Review." In *IEEE Trans. on Neural Networks*, pp. 480–488, Vol.5, No. 3, 1994.

[4] James W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning internal representations by error propagation." In *Parallel distributed processing: explorations in the microstructure of cognition*, pp. 318–362, Vol. 1, MIT press, Cambridge, MA., 1986.

[6] E. Mizutani, S.E. Dreyfus, and K. Nishio. "On derivation of MLP backpropagation from the Kelley-Bryson optimal-control gradient formula and its application." In *Proc. of the IEEE International Joint Conference on Neural Networks*, Vol.2, pages 167–172, Como ITALY, July 2000.

[7] Eiji Mizutani and Stuart E. Dreyfus. "On complexity analysis of supervised MLP-learning for algorithmic comparisons." In *Proceedings of the INNS-IEEE International Joint Conference on Neural Networks*. Vol. 1, pages 347–352, Washington D.C., July, 2001.

[8] Eiji Mizutani and Stuart E. Dreyfus. "Stagewise Newton, differential dynamic programming, and neighboring optimum control for neural-network learning." To appear in *Proc. of the 2005 American Control Conference, Portland OR, June 2005*. (Available at http://www.ieor.berkeley.edu/People/Faculty/dreyfus-pubs/ACC05.pdf)

[9] Eiji Mizutani and James Demmel. "On structure-exploiting trust-region regularized nonlinear least squares algorithms for neural-network learning." In *Neural Networks*, Elsevier Science, Vol. 16, pp. 745-753, 2003.

[10] Eiji Mizutani and James W. Demmel. "Iterative scaled trust-region learning in Krylov subspaces via Pearlmutter's implicit sparse Hessian-vector multiply." In *Advances in Neural Information Processing Systems (NIPS)*, pp. 209–216, Vol. 16, MIT Press, 2004.

[11] Barak A. Pearlmutter. "Fast exact multiplication by the Hessian." In *Neural Computation*, pp. 147–160, Vol. 6, No. 1, 1994.