2005 American Control Conference
June 8-10, 2005. Portland, OR, USA

Stagewise Newton, differential dynamic programming, and neighboring optimum control for neural-network learning

Eiji Mizutani and Stuart E. Dreyfus

Abstract— **The theory of optimal control is applied to multi-stage (i.e., multiple-layered) neural-network (NN) learning for developing efficient second-order algorithms, expressed in NN notation. In particular, we compare differential dynamic programming, neighboring optimum control, and stagewise Newton methods. Understanding their strengths and weaknesses would prove useful in pursuit of an effective intermediate step between the steepest descent and the Newton directions, arising in supervised NN-learning as well as reinforcement learning with function approximators.**

I. Introduction

We consider a discrete N-stage *optimal-control problem*: The state of a discrete dynamic system at stage s ($s = 1, \ldots, N$) is described by an m_s-length *state vector* $\mathbf{y}^s$; a transition of the system to the next state $\mathbf{y}^{s+1}$ (at stage $s+1$) is determined by our choice of an n_s-length *control vector* $\boldsymbol{\theta}^s$ through the following relation (with a *fixed* initial state $\mathbf{y}_{\text{init}}$):

$$\begin{array}{ll} & \mathbf{y}^{s+1} = \mathbf{f}^{s+1}(\mathbf{y}^s, \boldsymbol{\theta}^s), \ (\text{with } \mathbf{y}^1 \equiv \mathbf{y}_{\text{init}}) \\ \Longleftrightarrow & y_k^{s+1} = f_k^{s+1}(\mathbf{y}^s, \boldsymbol{\theta}^s); \ \ k = 1, \ldots, m_{s+1}, \end{array} \tag{1}$$

where $f(.)$ denotes some nonlinear transition function. The performance index (i.e., the objective function) to be minimized is given by $J = \sum_{s=1}^{N-1} L^s(\mathbf{y}^s, \boldsymbol{\theta}^s) + E(\mathbf{y}^N)$, where $L^s(.)$ is the cost at stage s and $E(.)$ the terminal cost. For our convenience, we define a scalar-valued *discrete* **Hamiltonian** function H^s below:

$$H^s(\mathbf{y}^s, \boldsymbol{\theta}^s, \boldsymbol{\lambda}^{s+1}) = L^s(\mathbf{y}^s, \boldsymbol{\theta}^s) + \boldsymbol{\lambda}^{s+1^T} \mathbf{f}^{s+1}(\mathbf{y}^s, \boldsymbol{\theta}^s), \tag{2}$$

where $\boldsymbol{\lambda}^s$ is an m_s-vector *Lagrange multiplier* (or *costate*) sequence for adjoining the equality constraints in Eq. (1) to J, which leads to the following Lagrangian function $\widetilde{J}$:

$$\widetilde{J} = \sum_{s=1}^{N-1} \left\{ L^s(\mathbf{y}^s, \boldsymbol{\theta}^s) + \boldsymbol{\lambda}^{s+1^T} \left[\mathbf{f}^{s+1}(\mathbf{y}^s, \boldsymbol{\theta}^s) - \mathbf{y}^{s+1} \right] \right\} + E(\mathbf{y}^N). \tag{3}$$

The first-order necessary conditions for obtaining an optimal control $\boldsymbol{\theta}^{s*}$ (that achieves a stationary value of J) are given by the *discrete Euler-Lagrange* (EL) equations below:

$$\begin{array}{l} \text{(a) Adjoint equations : } \ \boldsymbol{\lambda}^s = \frac{\partial H^s}{\partial \mathbf{y}^s} = \frac{\partial L^s}{\partial \mathbf{y}^s} + \left[\frac{\partial \mathbf{f}^{s+1}}{\partial \mathbf{y}^s} \right]^T \boldsymbol{\lambda}^{s+1}, \\ \text{(b) Optimality conditions : } \ \mathbf{0} = \frac{\partial H^s}{\partial \boldsymbol{\theta}^s} = \frac{\partial L^s}{\partial \boldsymbol{\theta}^s} + \left[\frac{\partial \mathbf{f}^{s+1}}{\partial \boldsymbol{\theta}^s} \right]^T \boldsymbol{\lambda}^{s+1} = \frac{\partial \widetilde{J}}{\partial \boldsymbol{\theta}^s}, \\ \text{(c) System equations : } \ \mathbf{y}^{s+1} = \mathbf{f}^{s+1}(\mathbf{y}^s, \boldsymbol{\theta}^s), \end{array} \tag{4}$$

E. Mizutani is with Computer Science, Tsing Hua University, Hsinchu 300, TAIWAN. eiji@wayne.cs.nthu.edu.tw

S. Dreyfus is with Faculty of Industrial Engineering and Operations Research, University of California, Berkeley, Berkeley, CA 94720, USA. dreyfus@ieor.berkeley.edu

with the two-point boundary conditions below for these three difference equations in Eq. (4):

$$\begin{array}{l} \text{(a) Initial state conditions : } \ \mathbf{y}^1 = \mathbf{y}_{\text{init}} \text{ (given)}, \\ \text{(b) Adjoint boundary conditions : } \ \boldsymbol{\lambda}^N = \frac{\partial E}{\partial \mathbf{y}^N}. \end{array} \tag{5}$$

This is a nonlinear *two-point boundary value problem* for three unknowns: $\mathbf{y}$, $\boldsymbol{\theta}$, and $\boldsymbol{\lambda}$. One may solve the posed problem in a stagewise manner by exploiting its sequential structure (e.g., sweep methods; see Section II-C) rather than solve it as one large set of nonlinear simultaneous algebraic equations; for more details, refer to any standard textbook on the subject (e.g., [3]).

Another line of attack is to use **dynamic programming** (**DP**) [1]; we define the *optimal value function* $V^s(\mathbf{y}^s)$ as the "minimum value of the performance index, starting at state $\mathbf{y}^s$ at stage s (i.e., **minimum cost-to-go**):" $V^s = \underset{\boldsymbol{\theta}^s, \ldots, \boldsymbol{\theta}^{N-1}}{\text{minimum}} \sum_{t=s}^{N-1} L^t(\mathbf{y}^t, \boldsymbol{\theta}^t) + E(\mathbf{y}^N)$. By Bellman's *principle of optimality*, any point on the optimal path can be an initial point for the remaining path; so, we obtain the following DP recurrence relation:

$$V^s(\mathbf{y}^s) = \min_{\boldsymbol{\theta}^s} \left[L^s(\mathbf{y}^s, \boldsymbol{\theta}^s) + V^{s+1}(\mathbf{y}^{s+1}) \right] = \min_{\boldsymbol{\theta}^s} Q^s(\mathbf{y}^s, \boldsymbol{\theta}^s), \tag{6}$$

where $V^s(.)$ is a function of the current state $\mathbf{y}^s$ alone, and the scalar-valued Q-function is defined as $Q^s(\mathbf{y}^s, \boldsymbol{\theta}^s) = L^s(\mathbf{y}^s, \boldsymbol{\theta}^s) + V^{s+1}(\mathbf{y}^{s+1})$. Our objective is: Given the initial condition $\mathbf{y}^1$, determine the control sequence $\{\boldsymbol{\theta}^s; s = 1, \cdots, N-1\}$ that attains V^1. In control engineering terminology, DP yields the *optimal nonlinear feedback control* (see Chap.VIII in [1]); yet, the DP algorithm becomes *infeasible* when m_s (the number of states at each stage s) is large due to the *curse of dimensionality*; i.e., the exponential growth of algorithmic complexity with respect to m_s. In the literature of neural networks, such an infeasible DP algorithm was described by Saratchandran [23] to optimize *multilayer perceptrons* (MLPs).

II. Stagewise Second-Order Algorithms in Optimal Control

To alleviate the curse of dimensionality, one may use the (first-order) gradient algorithms: In optimal control, Kelley and Bryson developed such a method based on a general form of **backpropagation** (**BP**), known as *adjoint equations* (4-a); see [16], [6] and references therein. Here, the state space is limited to a vicinity of a given nominal (or initial) state trajectory; so, the posed DP problem reduces to a *subproblem*, in which we may deal with the *second-order* approximation of the performance index or the DP value function [see Eq. (8)] as well as system equations.

In modern trust-region methods [5], [17], [18], the posed subproblem corresponds to the *trust-region subproblem*, wherein we aim at finding the best compromise between the steepest descent step and the Newton step. In optimal control, the second-order steps can be obtained by **differential dynamic programming** (**DDP**) [13], [12], [14] and **stagewise Newton** methods [22], [15], [8]. Furthermore, when the nominal solution (trajectory) is chosen to satisfy the *first-order necessary conditions* [i.e., EL-equations (4)], the methods are related to so-called **neighboring optimum control** (see [2]; Chaps. 6 & 7 [3]; Chap. 8 [4]), also known as **guidance schemes** [11], [7], wherein the associated subproblem is called Jacobi's *Accessory Minimum Problem* (AMP) for the second variation [see Eq. (21)].

A. Differential dynamic programming (DDP)

The following summary of DDP derivations and results is based on [13]. Given a *nominal* "non-optimal" control sequence $\boldsymbol{\theta}^s_{\text{now}}$ [that does not satisfy Eq. (4-b)], DDP seeks an *increment of control* $\delta\boldsymbol{\theta}^s$ for $\boldsymbol{\theta}^s_{\text{next}}=\boldsymbol{\theta}^s_{\text{now}}+\delta\boldsymbol{\theta}^s$ to improve the performance based on the second-order considerations. To this end, DDP considers ΔV^s, the **variation** of V^s, due to the variation of state vector $\delta\mathbf{y}^s$ resulting from the control increment $\delta\boldsymbol{\theta}^s$. The DP recurrence relation in Eq. (6) gives

$$\begin{aligned}\Delta V^s(\delta\mathbf{y}^s) &= \min_{\delta\boldsymbol{\theta}^s}\left[\Delta L^s(\delta\mathbf{y}^s,\delta\boldsymbol{\theta}^s)+\Delta V^{s+1}(\delta\mathbf{y}^{s+1})\right]\\ &= \min_{\delta\boldsymbol{\theta}^s}\Delta Q^s(\delta\mathbf{y}^s,\delta\boldsymbol{\theta}^s),\end{aligned} \tag{7}$$

where $\Delta V^s(.)$ is a function of the variation of the current state $\delta\mathbf{y}^s$ alone under the assumption that $\Delta V^s(.)$ is optimized with respect to $\delta\boldsymbol{\theta}^s,\cdots,\delta\boldsymbol{\theta}^{N-1}$. Applying the second-order truncated Taylor series expansion to the terms: $\Delta L^s(\delta\mathbf{y}^s,\delta\boldsymbol{\theta}^s)$, $\Delta V^{s+1}(\delta\mathbf{y}^{s+1})$, and $\delta y^{s+1}_k=\Delta f^{s+1}_k(\delta\mathbf{y}^s,\delta\boldsymbol{\theta}^s)$ yields the DP subproblem. Substituting them into Eq. (7) and then neglecting the terms of order higher than two in $\delta\mathbf{y}^s$ and $\delta\boldsymbol{\theta}^s$ yields:

$$\begin{aligned}\Delta V^s(\delta\mathbf{y}^s)=\min_{\delta\boldsymbol{\theta}^s}\Bigg\{&\begin{bmatrix}\left(\frac{\partial H^s}{\partial\mathbf{y}^s}\right)^T & \left(\frac{\partial H^s}{\partial\boldsymbol{\theta}^s}\right)^T\end{bmatrix}\begin{bmatrix}\delta\mathbf{y}^s\\ \\ \delta\boldsymbol{\theta}^s\end{bmatrix}\\ &+\tfrac{1}{2}\begin{bmatrix}\delta\mathbf{y}^{s^T} & \delta\boldsymbol{\theta}^{s^T}\end{bmatrix}\begin{bmatrix}\mathbf{A}^s & \mathbf{B}^s\\ \\ \mathbf{B}^{s^T} & \mathbf{C}^s\end{bmatrix}\begin{bmatrix}\delta\mathbf{y}^s\\ \\ \delta\boldsymbol{\theta}^s\end{bmatrix}\Bigg\}+v^{s+1},\end{aligned} \tag{8}$$

where H is the *Hamiltonian function* in Eq. (2) with $\boldsymbol{\lambda}^s$ replaced by $\widetilde{\boldsymbol{\lambda}}^s\overset{\text{def}}{=}\left(\frac{\partial V^s}{\partial\mathbf{y}^s}\right)$; this partial derivative of the value function with respect to the state plays the role of the Lagrange multiplier (see p. 195 in [1]). Also in Eq. (8), v^{s+1} is a scalar; $\left(\frac{\partial H^s}{\partial\mathbf{y}^s}\right)$ is an m_s-length column vector; $\left(\frac{\partial H^s}{\partial\boldsymbol{\theta}^s}\right)$ an n_s-vector; and three matrices $\mathbf{A}^s$, $\mathbf{B}^s$, and $\mathbf{C}^s$ are given below with an $m_{s+1}\times m_{s+1}$ matrix $\mathbf{E}^{s+1}\overset{\text{def}}{=}\left[\frac{\partial^2 V^{s+1}}{\partial\mathbf{y}^{s+1}\partial\mathbf{y}^{s+1}}\right]$:

$$\underbrace{\mathbf{A}^s}_{m_s\times m_s}=\left[\frac{\partial^2 Q^s}{\partial\mathbf{y}^s\partial\mathbf{y}^s}\right]=\left[\frac{\partial\mathbf{y}^{s+1}}{\partial\mathbf{y}^s}\right]^T\mathbf{E}^{s+1}\left[\frac{\partial\mathbf{y}^{s+1}}{\partial\mathbf{y}^s}\right]+\left[\frac{\partial^2 H^s}{\partial\mathbf{y}^s\partial\mathbf{y}^s}\right], \tag{9}$$

$$\underbrace{\mathbf{B}^s}_{m_s\times n_s}=\left[\frac{\partial^2 Q^s}{\partial\mathbf{y}^s\partial\boldsymbol{\theta}^s}\right]=\underbrace{\left[\frac{\partial\mathbf{y}^{s+1}}{\partial\mathbf{y}^s}\right]^T}_{m_s\times m_{s+1}}\mathbf{E}^{s+1}\underbrace{\left[\frac{\partial\mathbf{y}^{s+1}}{\partial\boldsymbol{\theta}^s}\right]}_{m_{s+1}\times n_s}+\underbrace{\left[\frac{\partial^2 H^s}{\partial\mathbf{y}^s\partial\boldsymbol{\theta}^s}\right]}_{m_s\times n_s}, \tag{10}$$

$$\underbrace{\mathbf{C}^s}_{n_s\times n_s}=\left[\frac{\partial^2 Q^s}{\partial\boldsymbol{\theta}^s\partial\boldsymbol{\theta}^s}\right]=\left[\frac{\partial\mathbf{y}^{s+1}}{\partial\boldsymbol{\theta}^s}\right]^T\mathbf{E}^{s+1}\left[\frac{\partial\mathbf{y}^{s+1}}{\partial\boldsymbol{\theta}^s}\right]+\left[\frac{\partial^2 H^s}{\partial\boldsymbol{\theta}^s\partial\boldsymbol{\theta}^s}\right]. \tag{11}$$

In Eq. (10), the last m_s-by-n_s matrix can be computed by

$$\left[\frac{\partial^2 H^s}{\partial y^s_i\partial\theta^s_j}\right]=\left[\frac{\partial^2 L^s}{\partial y^s_i\partial\theta^s_j}\right]+\sum_{k=1}^{m_{s+1}}\widetilde{\lambda}^{s+1}_k\left[\frac{\partial^2 f^{s+1}_k(\mathbf{y}^s,\boldsymbol{\theta}^s)}{\partial y^s_i\partial\theta^s_j}\right], \tag{12}$$

for $i=1,...,m_s;\ j=1,...,n_s$. Obvious modifications apply to obtaining the last matrix in Eqs. (9) and (11).

Now, from Eq. (8), an n_s-vector control increment, called the DDP step, at stage s is obtained as

$$\delta\boldsymbol{\theta}^s_{\text{DDP}}=\mathbf{u}^s_F-\mathbf{K}^s_{\text{DDP}}\delta\mathbf{y}^s,\ \text{ with }\begin{cases}\mathbf{u}^s_F=-\mathbf{C}^{s^{-1}}\left[\frac{\partial H^s}{\partial\boldsymbol{\theta}^s}\right],\\ \mathbf{K}^s_{\text{DDP}}=\mathbf{C}^{s^{-1}}\mathbf{B}^{s^T}.\end{cases} \tag{13}$$

Here, the resulting control can be interpreted as two distinct control actions:

- open-loop feedforward control $\mathbf{u}^s_F$ [that minimizes H^s with respect to $\boldsymbol{\theta}^s$ while holding $\mathbf{y}^s$ (and $\widetilde{\boldsymbol{\lambda}}^s$) fixed] to satisfy the optimality condition in Eq. (4-b); plus
- linear state feedback control with an m_s-by-m_s $\mathbf{K}^s_{\text{DDP}}$, a stage/time-varying feedback gain matrix [with variations of $\mathbf{y}$ (and thus $\widetilde{\boldsymbol{\lambda}}$) assumed to deviate the control from $\mathbf{u}^s_F$] to obtain a further improved control.

Substituting the obtained control increment $\delta\boldsymbol{\theta}^s_{\text{DDP}}$ in Eq. (13) into Eq. (8) and comparing to the truncated Taylor expansion of $\Delta V^s(\delta\mathbf{y}^s)=\left(\frac{\partial V^s}{\partial\mathbf{y}^s}\right)^T\delta\mathbf{y}^s+\frac{1}{2}\delta\mathbf{y}^{s^T}\mathbf{E}^s\delta\mathbf{y}^s+v^s$ yields:

$$\mathbf{E}^s\overset{\text{def}}{=}\left[\frac{\partial^2 V^s}{\partial\mathbf{y}^s\partial\mathbf{y}^s}\right]=\mathbf{A}^s-\mathbf{B}^s\mathbf{C}^{s^{-1}}\mathbf{B}^{s^T}=\mathbf{A}^s-\mathbf{B}^s\mathbf{K}^s_{\text{DDP}}, \tag{14}$$

$$\begin{aligned}\underbrace{\widetilde{\boldsymbol{\lambda}}^s}_{m_s\times 1}&\overset{\text{def}}{=}\left(\frac{\partial V^s}{\partial\mathbf{y}^s}\right)=\left(\frac{\partial H^s}{\partial\mathbf{y}^s}\right)-\mathbf{B}^s\mathbf{C}^{s^{-1}}\left(\frac{\partial H^s}{\partial\boldsymbol{\theta}^s}\right)\\ &=\left(\frac{\partial L^s}{\partial\mathbf{y}^s}\right)+\left[\frac{\partial\mathbf{y}^{s+1}}{\partial\mathbf{y}^s}\right]^T\widetilde{\boldsymbol{\lambda}}^{s+1}+\mathbf{B}^s\mathbf{u}^s_F,\end{aligned} \tag{15}$$

$$v^s=v^{s+1}-\tfrac{1}{2}\left(\frac{\partial H^s}{\partial\boldsymbol{\theta}^s}\right)^T\mathbf{C}^{s^{-1}}\left(\frac{\partial H^s}{\partial\boldsymbol{\theta}^s}\right). \tag{16}$$

Boundary conditions at terminal stage N are given by:

$$\underbrace{\widetilde{\boldsymbol{\lambda}}^N}_{m_N\times 1}=\left[\frac{\partial E}{\partial\mathbf{y}^N}\right];\quad\underbrace{\mathbf{E}^N}_{m_N\times m_N}=\left[\frac{\partial^2 E}{\partial\mathbf{y}^N\partial\mathbf{y}^N}\right]=\left[\frac{\partial\widetilde{\boldsymbol{\lambda}}^N}{\partial\mathbf{y}^N}\right];\quad v^N=0. \tag{17}$$

The scalar quantity v^1 gives a predicted reduction of the approximate performance index (or local quadratic model) resulting from the DDP-step in Eq.(13). In trust-region methods [5], [17], [18], the ratio of the predicted reduction to the actual one is used to control a trust-region radius.

B. Neighboring optimum control (NOC)

The following summary of NOC derivations and results is based on [7] (see also Sec. 3.8 in [10]). NOC considers perturbations on the optimal state trajectory, starting with a *nominal* (initially "optimal") control sequence $\boldsymbol{\theta}^{s*}$ (as $\boldsymbol{\theta}^s_{\text{now}}$); NOC seeks the optimal *control increment* $\delta\boldsymbol{\theta}^{s*}$ for $\boldsymbol{\theta}^s_{\text{next}}=\boldsymbol{\theta}^s_{\text{now}}+\delta\boldsymbol{\theta}^{s*}$ to improve the given a state increment $\delta\mathbf{y}^s$. NOC is derivable from Eq. (6): The optimality condition, Eq. (4-b), indicates

$$\mathbf{0}=\frac{\partial Q^s}{\partial\boldsymbol{\theta}^s}=\left(\frac{\partial L^s}{\partial\boldsymbol{\theta}^s}\right)+\left[\frac{\partial\mathbf{y}^{s+1}}{\partial\boldsymbol{\theta}^s}\right]^T\boldsymbol{\lambda}^{s+1}\ \left(\boldsymbol{\lambda}^s\overset{\text{def}}{=}\left[\frac{\partial V^s}{\partial\mathbf{y}^s}\right]\right), \tag{18}$$

which corresponds to Eq.(3.4), p. 367 in [7]. Since the optimal control $\boldsymbol{\theta}^{s*}$ may change due to perturbations in $\mathbf{y}^s$, the objective is to get an n_s-by-m_s *neighboring optimum feedback gain matrix* $\mathbf{K}^s \overset{\text{def}}{=} -\left[\frac{\partial \boldsymbol{\theta}^{s*}}{\partial \mathbf{y}^s}\right]$, *the rate of change in optimal control* $\boldsymbol{\theta}^{s*}$ *associated with a perturbation in the nominal value of* $\mathbf{y}^s$, for the following **linear feedback guidance rule** on the state deviation from nominal optimum state [compare the control in Eq. (13)]:

$$\delta\boldsymbol{\theta}^{s*} = \left[\frac{\partial \boldsymbol{\theta}^{s*}}{\partial \mathbf{y}^s}\right]\delta\mathbf{y}^s \iff \delta\boldsymbol{\theta}^{s*} = -\mathbf{K}^s\delta\mathbf{y}^s. \tag{19}$$

Take partial derivatives of Eq. (18) with respect to $\mathbf{y}^s$, obtaining

$$\begin{aligned}\mathbf{0} &= \underbrace{\frac{\partial}{\partial \mathbf{y}^s}\left(\frac{\partial Q^s}{\partial \boldsymbol{\theta}^s}\right)^T}_{m_s\times n_s} = \underbrace{\left[\frac{\partial \boldsymbol{\theta}^{s*}}{\partial \mathbf{y}^s}\right]^T}_{m_s\times n_s}\underbrace{\frac{\partial}{\partial \boldsymbol{\theta}^s}\left(\frac{\partial Q^s}{\partial \boldsymbol{\theta}^s}\right)^T}_{n_s\times n_s} + \underbrace{\left[\frac{\partial^2 Q^s}{\partial \mathbf{y}^s \partial \boldsymbol{\theta}^s}\right]}_{m_s\times n_s} \\ &= \underbrace{\left[\frac{\partial \boldsymbol{\theta}^{s*}}{\partial \mathbf{y}^s}\right]^T}_{m_s\times n_s}\underbrace{\mathbf{C}^s}_{n_s\times n_s} + \underbrace{\mathbf{B}^s}_{m_s\times n_s},\end{aligned}$$

which yields the desired guidance scheme in Eq. (19) with $\mathbf{K}^s = {\mathbf{C}^s}^{-1}{\mathbf{B}^s}^T$. To obtain recursions for $\boldsymbol{\lambda}^s$ on the nominal optimum path, take partial derivatives of $V^s = Q^s = L^s + V^{s+1}$ with respect to $\mathbf{y}^s$ and use Eq. (18), which yields **BP-formula** (4-a). To get recursions for $\left[\frac{\partial^2 V^s}{\partial \mathbf{y}^s \partial \mathbf{y}^s}\right]$, differentiate *BP-formula* (4-a) with respect to $\mathbf{y}^s$:

$$\begin{aligned}\left[\frac{\partial^2 V^s}{\partial \mathbf{y}^s \partial \mathbf{y}^s}\right] &= \mathbf{A}^s + \left[\frac{\partial \boldsymbol{\theta}^{s*}}{\partial \mathbf{y}^s}\right]^T \frac{\partial}{\partial \boldsymbol{\theta}^s}\left(\frac{\partial Q^s}{\partial \mathbf{y}^s}\right)^T \\ &= \mathbf{A}^s - {\mathbf{K}^s}^T{\mathbf{B}^s}^T = \mathbf{A}^s - \mathbf{B}^s{\mathbf{C}^s}^{-1}{\mathbf{B}^s}^T,\end{aligned} \tag{20}$$

which corresponds to Eq.(3.8), p. 368 in [7]; see also Eq. (14). The posed guidance control scheme could be employed for re-training an optimized NN model when a small change occurs in input data (or training data).

C. Sweep methods for NOC with discrete Riccati equations

What follows in this and the next subsections is largely attributable to ref. [4]. A discrete version of the AMP subproblem for neighboring optimum control is given by

$$\begin{aligned}\min_{\delta\boldsymbol{\theta}^s} \Delta J \approx \tfrac{1}{2}\sum_{s=1}^{N-1}\begin{bmatrix}\delta\mathbf{y}^{s^T} & \delta\boldsymbol{\theta}^{s^T}\end{bmatrix}\begin{bmatrix}\frac{\partial^2 H^{s+1}}{\partial \mathbf{y}^s \partial \mathbf{y}^s} & \frac{\partial^2 H^{s+1}}{\partial \mathbf{y}^s \partial \boldsymbol{\theta}^s} \\ \frac{\partial^2 H^{s+1}}{\partial \boldsymbol{\theta}^s \partial \mathbf{y}^s} & \frac{\partial^2 H^{s+1}}{\partial \boldsymbol{\theta}^s \partial \boldsymbol{\theta}^s}\end{bmatrix}\begin{bmatrix}\delta\mathbf{y}^s \\ \delta\boldsymbol{\theta}^s\end{bmatrix} \\ + \tfrac{1}{2}\delta\mathbf{y}^{N^T}\left[\frac{\partial^2 E}{\partial \mathbf{y}^N \partial \mathbf{y}^N}\right]\delta\mathbf{y}^N,\end{aligned} \tag{21}$$

subject to

$$\delta\mathbf{y}^{s+1} = \left[\frac{\partial \mathbf{f}^{s+1}}{\partial \mathbf{y}^s}\right]\delta\mathbf{y}^s + \left[\frac{\partial \mathbf{f}^{s+1}}{\partial \boldsymbol{\theta}^s}\right]\delta\boldsymbol{\theta}^s, \quad \delta\mathbf{y}^1 = \delta\mathbf{y}_{\text{init}} = \mathbf{0}. \tag{22}$$

From Eq. (4), the associated discrete EL-equations (with multiplier $\delta\boldsymbol{\lambda}^s$) become

$$\begin{bmatrix}\delta\boldsymbol{\lambda}^s \\ \mathbf{0} \\ \delta\mathbf{y}^{s+1}\end{bmatrix} = \begin{pmatrix}\left[\frac{\partial^2 H^s}{\partial \mathbf{y}^s \partial \mathbf{y}^s}\right] & \left[\frac{\partial^2 H^s}{\partial \mathbf{y}^s \partial \boldsymbol{\theta}^s}\right] & \left[\frac{\partial \mathbf{y}^{s+1}}{\partial \mathbf{y}^s}\right]^T \\ \left[\frac{\partial^2 H^s}{\partial \boldsymbol{\theta}^s \partial \mathbf{y}^s}\right] & \left[\frac{\partial^2 H^s}{\partial \boldsymbol{\theta}^s \partial \boldsymbol{\theta}^s}\right] & \left[\frac{\partial \mathbf{y}^{s+1}}{\partial \boldsymbol{\theta}^s}\right]^T \\ \left[\frac{\partial \mathbf{y}^{s+1}}{\partial \mathbf{y}^s}\right] & \left[\frac{\partial \mathbf{y}^{s+1}}{\partial \boldsymbol{\theta}^s}\right] & \mathbf{0}\end{pmatrix}\begin{bmatrix}\delta\mathbf{y}^s \\ \delta\boldsymbol{\theta}^s \\ \delta\boldsymbol{\lambda}^{s+1}\end{bmatrix}, \tag{23}$$

which is a two-point boundary value problem for $\delta\mathbf{y}^s$, $\delta\boldsymbol{\lambda}^s$, and $\delta\boldsymbol{\theta}^s$. To solve this, use a sweep method with a linear homogeneous relation between the costate and state vectors:

$$\delta\boldsymbol{\lambda}^s = \mathbf{E}^s\delta\mathbf{y}^s \quad (\text{with an } m_s\text{-by-}m_s \text{ matrix } \mathbf{E}^s). \tag{24}$$

To obtain a recurrence relation, use Eq. (22) to write $\delta\boldsymbol{\lambda}^{s+1} = \mathbf{E}^{s+1}\delta\mathbf{y}^{s+1} = \mathbf{E}^{s+1}\left\{\left[\frac{\partial \mathbf{f}^{s+1}}{\partial \mathbf{y}^s}\right]\delta\mathbf{y}^s + \left[\frac{\partial \mathbf{f}^{s+1}}{\partial \boldsymbol{\theta}^s}\right]\delta\boldsymbol{\theta}^s\right\}$, and then substitute it in Eq. (23), yielding

$$\begin{aligned}\begin{bmatrix}\delta\boldsymbol{\lambda}^s \\ \mathbf{0}\end{bmatrix} = \left\{\begin{pmatrix}\left[\frac{\partial^2 H^s}{\partial \mathbf{y}^s \partial \mathbf{y}^s}\right] & \left[\frac{\partial^2 H^s}{\partial \mathbf{y}^s \partial \boldsymbol{\theta}^s}\right] \\ \left[\frac{\partial^2 H^s}{\partial \boldsymbol{\theta}^s \partial \mathbf{y}^s}\right] & \left[\frac{\partial^2 H^s}{\partial \boldsymbol{\theta}^s \partial \boldsymbol{\theta}^s}\right]\end{pmatrix}\right. \\ \left. + \begin{pmatrix}\left[\frac{\partial \mathbf{y}^{s+1}}{\partial \mathbf{y}^s}\right]^T \\ \left[\frac{\partial \mathbf{y}^{s+1}}{\partial \boldsymbol{\theta}^s}\right]^T\end{pmatrix}\mathbf{E}^{s+1}\begin{pmatrix}\left[\frac{\partial \mathbf{y}^{s+1}}{\partial \mathbf{y}^s}\right] & \left[\frac{\partial \mathbf{y}^{s+1}}{\partial \boldsymbol{\theta}^s}\right]\end{pmatrix}\right\}\begin{bmatrix}\delta\mathbf{y}^s \\ \delta\boldsymbol{\theta}^s\end{bmatrix},\end{aligned}$$

which can be written in the compact form below:

$$\begin{aligned}&\begin{bmatrix}\delta\boldsymbol{\lambda}^s \\ \mathbf{0}\end{bmatrix} = \begin{bmatrix}\mathbf{A}^s & \mathbf{B}^s \\ {\mathbf{B}^s}^T & \mathbf{C}^s\end{bmatrix}\begin{bmatrix}\delta\mathbf{y}^s \\ \delta\boldsymbol{\theta}^s\end{bmatrix} \\ \iff &\begin{cases}\delta\boldsymbol{\theta}^s = -{\mathbf{C}^s}^{-1}{\mathbf{B}^s}^T\delta\mathbf{y}^s = -\mathbf{K}^s\delta\mathbf{y}^s, \\ \delta\boldsymbol{\lambda}^s = \left[\mathbf{A}^s - \mathbf{B}^s{\mathbf{C}^s}^{-1}{\mathbf{B}^s}^T\right]\delta\mathbf{y}^s = [\mathbf{A}^s - \mathbf{B}^s\mathbf{K}^s]\delta\mathbf{y}^s.\end{cases}\end{aligned} \tag{25}$$

Here, $\delta\boldsymbol{\theta}^s$ is the *guidance control* in Eq. (19), and $\mathbf{A}^s$, $\mathbf{B}^s$, and $\mathbf{C}^s$ are defined in Eqs. (9), (10), and (11), respectively, but $\boldsymbol{\lambda}^s$ for Eq. (12) is updated by *BP-formula* (4-a) [rather than Eq. (15)], as described in Sec. II-B. Comparing the last equation in Eq.(25) with Eq. (24) yields the same recurrence relation as Eqs. (14) and (20): $\mathbf{E}^s = \mathbf{A}^s - \mathbf{B}^s{\mathbf{C}^s}^{-1}{\mathbf{B}^s}^T$, which is called the **discrete Riccati equation** in this context (e.g., see Chap. 8 in [4]). The Riccati matrix $\mathbf{E}^s$ is the "*Schur complement matrix* of block $\mathbf{C}^s$ in matrix $\begin{bmatrix}\mathbf{A}^s & \mathbf{B}^s \\ {\mathbf{B}^s}^T & \mathbf{C}^s\end{bmatrix}$ associated with the $\delta\mathbf{y}^s$ variable," resulting from *block Gaussian elimination* (i.e., *block Cholesky* in this context due to symmetry; see Section III).

D. Discrete-time stagewise Newton Methods

In the literature, the earliest reference of *discrete-time stagewise Newton* method is often cited as Pantoja 1984 [21] or 1988 [22], but we have recently recognized that an LQ-subproblem approach developed by Dreyfus 1966 [8] is a discrete-time stagewise Newton method. A different (but equivalent) LQ-subproblem approach was also described by Dunn & Bertsekas later in [9] independently; see [20].

Stagewise Newton begins with a given nominal "non-optimal" path, as DDP does in Section II-A. The linear homogeneous relation between the costate and state vectors in Eq. (24) is replaced by the inhomogeneous one:

$$\delta\boldsymbol{\lambda}^s = \mathbf{E}^s\delta\mathbf{y}^s + \mathbf{h}^s \quad (\text{with an } m_s\text{-by-}m_s \text{ matrix } \mathbf{E}^s) \tag{26}$$

because the optimality condition, Eq. (4-b), is not satisfied; hence, the left-hand-side zero vector $\mathbf{0}$ in Eq. (23) is replaced by $\delta\left(\frac{\partial H^s}{\partial \boldsymbol{\theta}^s}\right)$. *Stagewise Newton* takes the next two

actions simultaneously to deal with non-zero $\delta\left(\frac{\partial H^s}{\partial \boldsymbol{\theta}^s}\right)$:

- feedforward control to make $\left(\frac{\partial H^s}{\partial \boldsymbol{\theta}^s}\right)+\delta\left(\frac{\partial H^s}{\partial \boldsymbol{\theta}^s}\right)=\mathbf{0}$ for a nominal "optimum" path, and
- linear state feedback control [as the neighboring optimum guidance control in Eq. (19)].

Hence, the stagewise Newton step is $\delta\boldsymbol{\theta}^s_{\rm N}=\mathbf{u}^s_F-\mathbf{K}^s\delta\mathbf{y}^s$, where the open-loop feedforward control $\mathbf{u}^s_F$ is given with a gradient vector $\mathbf{g}^s\equiv\left(\frac{\partial H^s}{\partial \boldsymbol{\theta}^s}\right)=-\delta\left(\frac{\partial H^s}{\partial \boldsymbol{\theta}^s}\right)$ (see Chap. 8 in [4]) as

$$\begin{aligned}\mathbf{u}^s_F &= -\mathbf{C}^{s^{-1}}\left[\frac{\partial \mathbf{y}^{s+1}}{\partial \boldsymbol{\theta}^s}\right]^T\mathbf{h}^{s+1}-\mathbf{C}^{s^{-1}}\mathbf{g}^s \\ &= -\mathbf{C}^{s^{-1}}\left[\frac{\partial \mathbf{y}^{s+1}}{\partial \boldsymbol{\theta}^s}\right]^T\left(\mathbf{h}^{s+1}+\boldsymbol{\lambda}^{s+1}\right)-\mathbf{C}^{s^{-1}}\left(\frac{\partial L^s}{\partial \boldsymbol{\theta}^s}\right).\end{aligned} \tag{27}$$

One gets a recurrence relation of $\mathbf{h}^s$, following the same procedures for Eq. (25): Write Eq. (26) as

$$\begin{aligned}\delta\boldsymbol{\lambda}^{s+1} &= \mathbf{E}^{s+1}\delta\mathbf{y}^{s+1}+\mathbf{h}^{s+1} \\ &= \mathbf{E}^{s+1}\left\{\left[\frac{\partial \mathbf{f}^{s+1}}{\partial \mathbf{y}^s}\right]\delta\mathbf{y}^s+\left[\frac{\partial \mathbf{f}^{s+1}}{\partial \boldsymbol{\theta}^s}\right]\delta\boldsymbol{\theta}^s\right\}+\mathbf{h}^{s+1},\end{aligned}$$

where Eq. (22) is used, and then substitute it in Eq. (23), where $\delta\left(\frac{\partial H^s}{\partial \boldsymbol{\theta}^s}\right)=-\left(\frac{\partial H^s}{\partial \boldsymbol{\theta}^s}\right)$ used on the left-hand side:

$$\begin{aligned}\mathbf{h}^s &= \left\{\left[\frac{\partial \mathbf{y}^{s+1}}{\partial \mathbf{y}^s}\right]-\left[\frac{\partial \mathbf{y}^{s+1}}{\partial \boldsymbol{\theta}^s}\right]\mathbf{C}^{s^{-1}}\mathbf{B}^{s^T}\right\}^T\mathbf{h}^{s+1}-\mathbf{B}^s\mathbf{C}^{s^{-1}}\left(\frac{\partial H^s}{\partial \boldsymbol{\theta}^s}\right) \\ &= \left[\frac{\partial \mathbf{y}^{s+1}}{\partial \mathbf{y}^s}\right]^T\mathbf{h}^{s+1}+\mathbf{B}^s\mathbf{u}^s_F.\end{aligned} \tag{28}$$

In summary, this version of stagewise Newton (see pages 325-326 in [4]) employs Eq. (28) for $\mathbf{h}^s$ as well as $\mathbf{E}^s=\mathbf{A}^s-\mathbf{B}^s\mathbf{C}^{s^{-1}}\mathbf{B}^{s^T}=\mathbf{A}^s-\mathbf{B}^s\mathbf{K}^s$, and the boundary conditions are $\mathbf{h}^N=\mathbf{0}$ plus Eq. (17). Notice here that $\boldsymbol{\lambda}^s$ is sequenced backward by BP-formula (4-a) separately from $\mathbf{h}^s$, which is backpropagated by Eq. (28).

A slightly different version of stagewise Newton [8], [21], [22], [9] defines a sensitivity vector $\boldsymbol{\zeta}^s\equiv\left(\frac{\partial V^s}{\partial \mathbf{y}^s}\right)=\mathbf{h}^s+\boldsymbol{\lambda}^s$, where V^s is a non-optimal value function (of course, $\mathbf{h}^s=\mathbf{0}$ on the optimal path); then, stagewise Newton uses $\boldsymbol{\zeta}^{s+1}$ in Eq. (27) and $\boldsymbol{\zeta}^s$ (and $\boldsymbol{\zeta}^{s+1}$) in place of $\widetilde{\boldsymbol{\lambda}}^s$ (and $\widetilde{\boldsymbol{\lambda}}^{s+1}$) in Eq. (15) as well as BP-formula (4-a) for $\boldsymbol{\lambda}^s$ in Eq. (12); hence, Eq. (28) is not needed. Starting with the boundary condition in Eq. (17) plus $\boldsymbol{\zeta}^N=\boldsymbol{\lambda}^N$ at terminal stage N, sequence the discrete Riccati equation backward (to propagate second derivatives) together with *BP-formula* (4-a) down to the first stage 1. Then, the posed two-point boundary problem reduces to an initial value problem. In the next section, we apply this algorithm with $\boldsymbol{\zeta}^s$ to NN-learning.

The foregoing discussions lead to a conclusion that the DDP step $\delta\boldsymbol{\theta}^s_{\rm DDP}$ in Eq. (13) is not the Newton step $\delta\boldsymbol{\theta}^s_{\rm N}$ above (e.g., see [22]); that is, observe the following three facts in the DDP procedure: First, $\widetilde{\boldsymbol{\lambda}}^s$ itself is updated & backpropagated by Eq. (15) (*without* using BP-formula (4-a); see p.95 in [13]; p.189 in [12]), and employed all the way for computing $\left(\frac{\partial H^s}{\partial \boldsymbol{\theta}^s}\right)$, $\mathbf{A}^s$, $\mathbf{B}^s$, and $\mathbf{C}^s$. Second, $\mathbf{g}^s$ is never computed exactly $\left[\mathbf{g}^s\neq\left(\frac{\partial H^s}{\partial \boldsymbol{\theta}^s}\right)\right]$. Third, $\mathbf{K}^s_{\rm DDP}$ in Eq. (13) and $\mathbf{K}^s$ above differ because Eq. (12) for matrices of second derivatives makes a difference. Also, for this reason, Eq.(3.118) on p.70 in [10] yields the DDP step, although it is called the Newton-Raphson method.

III. Application to Multilayer Perceptron (MLP) Learning

The aforementioned optimal-control algorithms are applicable to an N-stage multilayer perceptron (MLP) with $N-2$ hidden layers and P_s nodes at stage s ($s=1,\cdots,N$; the first input layer is stage 1). Here, an n_s-length decision vector $\boldsymbol{\theta}^s$ ($s=1,\cdots,N-1$) corresponds to an n_s-vector of the parameters [$n_s=(1+P_s)P_{s+1}$ including thresholds] between layers s and $s+1$, and an m_s-length state vector $\mathbf{y}^s$ at stage s to an m_s-vector of node outputs at layer s, where $m_s=P_s$ in *on-line learning* and $m_s=P_sD$ in *batch learning* when D training data involved. Using two-hidden-layer MLP-learning ($N=4$), we compare two methods: *stagewise Newton* and *standard Newton with Cholesky factorization*. Both seek the same Newton step $\delta\boldsymbol{\theta}_{\rm N}$ (subscript N denotes Newton) by solving differently the **Newton formula**: $\mathbf{H}\,\delta\boldsymbol{\theta}_{\rm N}=-\mathbf{g}$, where both the gradient vector $\mathbf{g}$ and the Hessian matrix $\mathbf{H}$ are stagewise-partitioned, as shown below

$$\mathbf{H}=\begin{bmatrix}\mathbf{H}^{3,3} & \mathbf{H}^{2,3^T} & \mathbf{H}^{1,3^T}\\ \mathbf{H}^{2,3} & \mathbf{H}^{2,2} & \mathbf{H}^{1,2^T}\\ \mathbf{H}^{1,3} & \mathbf{H}^{1,2} & \mathbf{H}^{1,1}\end{bmatrix},\quad \delta\boldsymbol{\theta}_{\rm N}=\begin{bmatrix}\delta\boldsymbol{\theta}^3_{\rm N}\\ \delta\boldsymbol{\theta}^2_{\rm N}\\ \delta\boldsymbol{\theta}^1_{\rm N}\end{bmatrix},\quad \mathbf{g}=\begin{bmatrix}\mathbf{g}^3\\ \mathbf{g}^2\\ \mathbf{g}^1\end{bmatrix}. \tag{29}$$

A. Discrete-time stagewise Newton step

First, we show the step obtainable from the stagewise Newton method $\delta\boldsymbol{\theta}^s_{\rm N}=\mathbf{u}^s_F-\mathbf{K}^s\delta\mathbf{y}^s$ below by three passes: (1) forward to get node outputs; (2) backward to get matrices $\mathbf{A}^s$, $\mathbf{B}^s$, $\mathbf{C}^s$, and $\mathbf{K}^s$ with Eq. (27) (using $\boldsymbol{\zeta}^s$) to get $\mathbf{u}^s_F$; and (3) forward with Eq. (22) to get $\delta\mathbf{y}^s$ and $\delta\boldsymbol{\theta}^s_{\rm N}$:

$$\begin{cases}\delta\boldsymbol{\theta}^1_{\rm N}=-\mathbf{C}^{1^{-1}}\left[\frac{\partial \mathbf{y}^2}{\partial \boldsymbol{\theta}^1}\right]^T\boldsymbol{\zeta}^2; \quad (\boldsymbol{\zeta}^s\equiv\mathbf{h}^s+\boldsymbol{\lambda}^s)\\ \delta\boldsymbol{\theta}^2_{\rm N}=-\mathbf{C}^{2^{-1}}\left[\frac{\partial \mathbf{y}^3}{\partial \boldsymbol{\theta}^2}\right]^T\boldsymbol{\zeta}^3-\mathbf{K}^2\delta\mathbf{y}^2;\\ \delta\boldsymbol{\theta}^3_{\rm N}=-\mathbf{C}^{3^{-1}}\left[\frac{\partial \mathbf{y}^4}{\partial \boldsymbol{\theta}^3}\right]^T\boldsymbol{\zeta}^4-\mathbf{K}^3\delta\mathbf{y}^3,\end{cases} \tag{30}$$

where we have

$$\begin{cases}\delta\mathbf{y}^2=\left[\frac{\partial \mathbf{y}^2}{\partial \boldsymbol{\theta}^1}\right]\delta\boldsymbol{\theta}^1_{\rm N};\\ \delta\mathbf{y}^3=\left[\frac{\partial \mathbf{y}^3}{\partial \boldsymbol{\theta}^2}\right]\delta\boldsymbol{\theta}^2_{\rm N}+\left[\frac{\partial \mathbf{y}^3}{\partial \mathbf{y}^2}\right]\delta\mathbf{y}^2;\\ \boldsymbol{\zeta}^4=\boldsymbol{\lambda}^4.\end{cases} \tag{31}$$

After a little algebra, the stagewise Newton step obtained here can be rewritten in terms of the original Hessian blocks and the gradient vector in Eq. (29), as shown next:

$$\begin{cases}\delta\boldsymbol{\theta}^1_{\rm N}=-\left[\widehat{\mathbf{H}}^{1,1}-\widehat{\mathbf{H}}^{1,2}\widehat{\mathbf{H}}^{2,2^{-1}}\widehat{\mathbf{H}}^{1,2^T}\right]^{-1}\widehat{\mathbf{g}}^1;\\ \delta\boldsymbol{\theta}^2_{\rm N}=-\widehat{\mathbf{H}}^{2,2^{-1}}\widehat{\mathbf{g}}^2-\widehat{\mathbf{H}}^{2,2^{-1}}\widehat{\mathbf{H}}^{1,2^T}\delta\boldsymbol{\theta}^1_{\rm N};\\ \delta\boldsymbol{\theta}^3_{\rm N}=-\mathbf{H}^{3,3^{-1}}\mathbf{g}^3-\mathbf{H}^{3,3^{-1}}\mathbf{H}^{1,3^T}\delta\boldsymbol{\theta}^1_{\rm N}-\mathbf{H}^{3,3^{-1}}\mathbf{H}^{2,3^T}\delta\boldsymbol{\theta}^2_{\rm N},\end{cases} \tag{32}$$

where hatted terms are given by:

$$\begin{cases}\widehat{\mathbf{g}}^2=\mathbf{g}^2-\mathbf{H}^{2,3}\mathbf{H}^{3,3^{-1}}\mathbf{g}^3,\\ \widehat{\mathbf{g}}^1=\mathbf{g}^1-\mathbf{H}^{1,3}\mathbf{H}^{3,3^{-1}}\mathbf{g}^3-\widehat{\mathbf{H}}^{1,2}\widehat{\mathbf{H}}^{2,2^{-1}}\widehat{\mathbf{g}}^2,\\ \widehat{\mathbf{H}}^{1,1}=\mathbf{H}^{1,1}-\mathbf{H}^{1,3}\mathbf{H}^{3,3^{-1}}\mathbf{H}^{1,3^T},\\ \widehat{\mathbf{H}}^{1,2}=\mathbf{H}^{1,2}-\mathbf{H}^{1,3}\mathbf{H}^{3,3^{-1}}\mathbf{H}^{2,3^T},\\ \widehat{\mathbf{H}}^{2,2}=\mathbf{H}^{2,2}-\mathbf{H}^{2,3}\mathbf{H}^{3,3^{-1}}\mathbf{H}^{2,3^T},\end{cases}$$

because $\mathbf{C}^1$, $\mathbf{C}^2$, and $\mathbf{C}^3$ in Eq. (30) can be expressed as

$$\begin{cases} \mathbf{C}^3 = \mathbf{H}^{3,3}; \\ \mathbf{C}^2 = \widehat{\mathbf{H}}^{2,2} = \mathbf{H}^{2,2} - \mathbf{H}^{2,3}\mathbf{H}^{3,3^{-1}}\mathbf{H}^{2,3^T}; \\ \mathbf{C}^1 = \widehat{\mathbf{H}}^{1,1} - \widehat{\mathbf{H}}^{1,2}\widehat{\mathbf{H}}^{2,2^{-1}}\widehat{\mathbf{H}}^{1,2^T}. \end{cases}$$

Clearly, when the off-diagonal Hessian blocks $\mathbf{H}^{s,t}$ $(s \neq t)$ are all ignored, then $\delta\boldsymbol{\theta}_{\text{N}}^3$ in Eq. (32) reduces to the open-loop feedforward control $-\mathbf{H}^{3,3^{-1}}\mathbf{g}^3$ alone with no state feedback guidance control.

B. Standard Newton step by block Cholesky factorization

Next, we show the standard Newton method: We first need to form explicitly $\mathbf{H}$ and $\mathbf{g}$ both in such a stagewise-partitioned format as shown in Eq. (29), where for $\mathbf{H}$, we need to form three off-diagonal blocks and only the lower half 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 (see [19]). We then perform block-Cholesky factorization on it; that is, $\mathbf{H} = \mathbf{L}\mathbf{L}^T$, where $\mathbf{L}$ denotes the lower-triangular Cholesky factor below:

$$\mathbf{L} = \begin{bmatrix} \mathbf{U}_3^T & & \\ \mathbf{X} & \mathbf{U}_2^T & \\ \mathbf{Y} & \mathbf{Z} & \mathbf{U}_1^T \end{bmatrix}.$$

After that, the Newton step $\delta\boldsymbol{\theta}_{\text{N}}$ can be obtained by solving the two triangular systems: $\mathbf{L}\mathbf{p} = -\mathbf{g}$ and $\mathbf{p} = \mathbf{L}^T\delta\boldsymbol{\theta}_{\text{N}}$. This algorithm can be summarized below:

Algorithm: *A standard Newton method with block-Cholesky factorization*.

(0) Compute $\mathbf{g}$ and $\mathbf{H}$ in Eq. (29) (e.g., see [19]) to form the Newton formula;
(1) Perform Cholesky on the first diagonal block: $\mathbf{H}^{3,3} = \mathbf{U}_3^T\mathbf{U}_3$;
(2) Solve a triangular system (no need to invert $\mathbf{U}_3$): $\mathbf{X} = \mathbf{H}^{2,3}\mathbf{U}_3^{-1}$;
(3) Solve a triangular system (without inverting $\mathbf{U}_3$): $\mathbf{Y} = \mathbf{H}^{1,3}\mathbf{U}_3^{-1}$;
(4) Compute a Schur complement matrix $\mathbf{V}_2$ (using $\mathbf{X}$) by: $\mathbf{V}_2 = \mathbf{H}^{2,2} - \mathbf{X}\mathbf{X}^T$;
(5) Perform Cholesky on $\mathbf{V}_2 = \mathbf{U}_2^T\mathbf{U}_2$;
(6) Solve a triangular system (without inverting $\mathbf{U}_2$): $\mathbf{Z} = \left[\mathbf{H}^{1,2} - \mathbf{Y}\mathbf{X}^T\right]\mathbf{U}_2^{-1}$;
(7) Compute a Schur complement matrix $\mathbf{V}_1$ (using $\mathbf{Y}$ and $\mathbf{Z}$) by: $\mathbf{V}_1 = \mathbf{H}^{1,1} - \mathbf{Y}\mathbf{Y}^T - \mathbf{Z}\mathbf{Z}^T$;
(8) Perform Cholesky on $\mathbf{V}_1 = \mathbf{U}_1^T\mathbf{U}_1$;
(9) Solve a lower-triangular system $\mathbf{L}\mathbf{p} = -\mathbf{g}$ by forward substitution;
(10) Solve an upper-triangular system $\mathbf{L}^T\delta\boldsymbol{\theta}_{\text{N}} = \mathbf{p}$ by back substitution.

The desired *full* Newton step $\delta\boldsymbol{\theta}_{\text{N}}$ [see Eq. (29)] is given below as the solution to the two triangular systems:

$$\begin{cases} \delta\boldsymbol{\theta}_{\text{N}}^1 = \mathbf{U}_1^{-1}\mathbf{p}_1; \\ \delta\boldsymbol{\theta}_{\text{N}}^2 = \mathbf{U}_2^{-1}\left(\mathbf{p}_2 - \mathbf{Z}^T\delta\boldsymbol{\theta}_{\text{N}}^1\right); \\ \delta\boldsymbol{\theta}_{\text{N}}^3 = \mathbf{U}_3^{-1}\left(\mathbf{p}_3 - \mathbf{X}^T\delta\boldsymbol{\theta}_{\text{N}}^2 - \mathbf{Y}^T\delta\boldsymbol{\theta}_{\text{N}}^1\right); \end{cases} \tag{33}$$

where we have

$$\begin{cases} \mathbf{p}_1 = -\mathbf{U}_1^{-T}\left[\mathbf{g}^{1,2} - \mathbf{Y}\mathbf{U}_3^{-T}\mathbf{g}^{3,4} - \mathbf{Z}\mathbf{U}_2^{-T}\widehat{\mathbf{g}}^{2,3}\right]; \\ \mathbf{p}_2 = -\mathbf{U}_2^{-T}\left(\mathbf{g}^{2,3} - \mathbf{X}\mathbf{U}_3^{-T}\mathbf{g}^{3,4}\right) = -\mathbf{U}_2^{-T}\widehat{\mathbf{g}}^{2,3}; \\ \mathbf{p}_3 = -\mathbf{U}_3^{-T}\mathbf{g}^{3,4}. \end{cases}$$

It is easy to verify that this resulting Newton step is equivalent to the stagewise Newton step in Eq. (30) [i.e., (32)], but obtained differently. Stagewise Newton computes exactly the first diagonal block $\mathbf{H}^{3,3}$ at Stage 3 [see $\mathbf{H}$ in Eq. (29)]. At later subsequent stages, however, it calculates no Hessian blocks explicitly; instead, it computes $\mathbf{C}^s$, which is identical to the Schur complement matrix $\mathbf{V}_s$, because, in this example, we have

$$\begin{aligned} \mathbf{V}_2 &= \mathbf{H}^{2,2} - \mathbf{X}\mathbf{X}^T = \mathbf{H}^{2,2} - \left(\mathbf{H}^{2,3}\mathbf{U}_3^{-1}\right)\left(\mathbf{H}^{2,3}\mathbf{U}_3^{-1}\right)^T \\ &= \mathbf{H}^{2,2} - \mathbf{H}^{2,3}\mathbf{H}^{3,3^{-1}}\mathbf{H}^{2,3^T} = \mathbf{C}^2, \\ \mathbf{V}_1 &= \mathbf{H}^{1,1} - \mathbf{Y}\mathbf{Y}^T - \mathbf{Z}\mathbf{Z}^T = \widehat{\mathbf{H}}^{1,1} - \widehat{\mathbf{H}}^{1,2}\,\mathbf{V}_2^{-1}\,\widehat{\mathbf{H}}^{1,2^T} = \mathbf{C}^1. \end{aligned}$$

Note that the overall cost to get $\delta\boldsymbol{\theta}_{\text{N}}$ with the posed block Cholesky that factors an n_s-by-n_s block $\mathbf{H}^{s,s}$ per stage is essentially the same as that with standard Cholesky that factors one large n-by-n $\mathbf{H}$ [17]; hence, $O(N^3)$ due to linear-equation solving in the parameter (i.e., decision) space alone. By contrast, stagewise Newton computes the Schur complement matrices $\mathbf{C}^s$ in the parameter space, but *propagates the second-order information through the state space*; therefore, it works in $O(N)$ to get the same Newton step $\delta\boldsymbol{\theta}_{\text{N}}$ without forming the Newton formula explicitly.

IV. Discussion

Optimal-control problems can be viewed as NN-learning problems involving only a *single training datum* $(D = 1)$; yet, the associated Hessian $\mathbf{H}$ can be "full-rank" because

- the number of decision variables at each stage is not larger than that of states;
- stage costs generate extra residuals (on top of the usual terminal residuals).

In MLP-learning, those two situations correspond to the two "uncommon" cases below:

- using *weight-sharing* and *weight-pruning* combined;
- presenting desired outputs at hidden nodes (i.e., *hidden-node teaching*; see [16]).

To alleviate rank-deficiency of $\mathbf{H}$, one can

- add some (positive) diagonal matrix to $\mathbf{C}^s$ per stage in stagewise Newton or DDP;
- prepare a training data set (yielding multiple terminal residual sets).

The first idea is related to the Levenberg-Marquardt method in nonlinear least squares sense [5], [17], [18], and the second is ubiquitous in supervised MLP-learning. If **on-line second-order learning** is adopted, then one must use the first idea for MLP-learning: In the nonlinear least squares, for instance, if we omit matrices of second derivatives [e.g., Eq. (12)], then the method reduces to a so-called *incremental Gauss-Newton method* [or stagewise version of *natural gradient learning* (under certain assumptions)];

here, the posed on-line stagewise implementation could be done by updating Cholesky factor of approximate $\mathbf{H}$ (e.g., with a MATLAB command cholupdate) on each datum.

Batch-mode MLP-learning (with D data) conceptually corresponds to optimal-control problems, in which a posed MLP model is copied and concatenated D times; so, there are totally $m_s = P_s D$ states at each stage s, while the number of decision parameters remains the same as n_s per stage because all the D MLPs (i.e., D copies of our MLP model) share the same parameter set (i.e., weight-sharing). In this setting, all the D data go through all those juxtaposed D MLPs (MLP d; $d = 1, \cdots, D$) simultaneously, as if a single state vector (including all the D data) goes to one large system. Therefore, stagewise Newton must deal with very large matrices (e.g., an m_s-by-m_s $\mathbf{A}^s$) since $m_s = P_s D$ can be arbitrarily large and $N \ll D$ holds typically in MLP-learning; hence, stagewise Newton progresses in $O(D^3)$. This situation can be illustrated with a classification benchmark called the *letter recognition problem* (available at the UCI machine learning repository), which involves 16 inputs (features), 26 outputs (alphabets), and 16,000 training data (i.e., D=16,000). This problem can be attacked with a 16-70-50-26 MLP [18]; that is, $P_1 = 16; P_2 = 70; P_3 = 50; P_4 = 26$; n_1=1,190; n_2=3,550; and n_3=1,326. Hence, $m_s = P_s D$ become very large. In addition, the node outputs of any single MLP d (for datum d) have nothing to do with those of the other MLPs i $(\neq d)$ because there is no connection between adjacent MLPs. Consequently, in MLP batch-learning, stagewise Newton may not work more efficiently than standard Newton that forms and then solves the *Newton formula* in $O(D)$, where stagewise second-order BP [19] processes one datum after another to obtain $\mathbf{H}$; hence, $m_s = P_s$ regardless of D.

V. CONCLUSION

We have described a class of second-order optimal control methods for NN-learning, and applied stagewise Newton to MLP-learning. The following results are highlighted:

(1) The Newton step consists of open-loop feedforward control plus state feedback guidance control;
(2) No use of off-diagonal Hessian blocks results in open-loop feedforward control alone;
(3) Differentiating *BP-formula* (4-a) with respect to the state yields a discrete Riccati equation;
(4) *Stagewise* implementation leads to *on-line* second-order learning (or approximate Newton methods);
(5) In batch-learning, stagewise Newton may not be more efficient than standard Newton (with stagewise second-order BP [19]).

The posed second-order optimal-control methods can be applied to *temporal-difference reinforcement learning* [24] with differentiable NN function approximators. In any context, we recommend **trust-region** frameworks [5], [17], [18] to pursue a good *compromise* between the steepest descent step (by Kelley-Bryson gradient method [16]) and a second-order step (e.g., by stagewise Newton). Further analysis available in control engineering could prove useful in developing more elaborate NN-learning algorithms.

REFERENCES

[1] Richard E. Bellman and Stuart E. Dreyfus. *Applied Dynamic Programming*, Princeton University Press, 1962.
[2] J. V. Breakwell, J. L. Speyer, and A. E. Bryson. "Optimization and Control of Nonlinear Systems Using the Second Variation." *SIAM Journal, Control*, Ser. A, Vol. 1, No. 2, pp. 193–223, 1963.
[3] A. E. Bryson and Y.-C. Ho. *Applied Optimal Control*. Blaisdell, Waltham, MA, 1969; *Revised:* Hemisphere, Washington, D.C., 1975.
[4] Arthur E. Bryson, Jr. *Dynamic Optimization*. Addison-Wesley, 1999.
[5] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust-Region Methods.* SIAM, 2000.
[6] Y. LeCun. "A Theoretical Framework for Back-Propagation." In *Proc. of the 1988 Connectionist Models Summer School*, pp. 21–28, 1988.
[7] Stuart E. Dreyfus and Jarrell R. Elliott. "An Optimal Linear Feedback Guidance Scheme." *Journal of Mathematical Analysis and Applications*, Vol. 8, No. 3, pp. 364–386, 1964.
[8] Stuart E. Dreyfus. "The Numerical Solution of Non-Linear Optimal Control Problems." In *Numerical Solutions of Nonlinear Differential Equations: Proceedings of an Advanced Symposium*, Ed. by D. Greenspan, pp. 97–113, John Wiley & Sons, Inc., 1966.
[9] J. Dunn and D. P. Bertsekas. "Efficient dynamic programming implementations of Newton's method for unconstrained optimal control problems." *Journal of Optimization Theory and Applications*, Vol. 63, pp. 23–38, 1989.
[10] Peter Dyer and Stephen R. McReynolds. *The Computation and Theory of Optimal Control*, Vol. 65 in Mathematics in Science and Engineering, Academic press, New York, 1970.
[11] Henry J. Kelley. "Guidance Theory and Extremal Fields." *IRE Trans on Automatic Control*, pp. 75–82, AC-7, 1962.
[12] David H. Jacobson. "Second-Order and Second-Variation Methods for Determining Optimal Control." *Int'l Journal of Control*, Vol. 7, No. 2, pp. 175–196, 1968.
[13] David Mayne. "A Second Order Gradient Method for Determining Optimal Trajectories of Non-Linear Discrete Time Systems." *Int'l Journal of Control*, Vol. 3, No. 1, pp. 85–95, 1966.
[14] Stephen R. McReynolds. "The Successive Sweep Method and Dynamic Programming." In *Journal of Mathematical Analysis and Applications*, Vol. 19, pp. 565–598, 1967.
[15] S. K. Mitter. "Successive approximation methods for the solution of optimal control problems." *Automatica*, Vol. 3, pp. 135–149, 1966.
[16] 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 Int'l Joint Conf. on Neural Networks*, Vol.2, pp. 167–172, Como ITALY, July 2000.
[17] Eiji Mizutani and James W. Demmel. "On structure-exploiting trust-region regularized nonlinear least squares algorithms for neural-network learning," In *Journal of Neural Networks*, Elsevier Science, Vol. 16, pp. 745-753, 2003.
[18] 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 16 (NIPS 2003)*, MIT Press, Vol.16, pp. 209-216, 2004.
[19] E. Mizutani, S. Dreyfus, and J. Demmel. "Second-order back-propagation algorithms for a stagewise-partitioned separable Hessian matrix." *Submitted* to the Int'l Joint Conf. on Neural Networks (IJCNN 2005), Montreal Quebec, CANADA, July 31 - August 4, 2005.
[20] Eiji Mizutani and Stuart E. Dreyfus. "LQ-subproblem approaches toward discrete-time stagewise Newton algorithms." *Working manuscript*, 2005.
[21] J. F. A. De O. Pantoja. "Algorithms for Constrained Optimization Problems." PhD thesis. Imperial College of Science and Technology, University of London, 1984.
[22] J. F. A. De O. Pantoja. "Differential dynamic programming and Newton's method." In *International Journal of Control,* Vol. 47, No.5, pp. 1539-1553, 1988.
[23] P. Saratchandran. "Dynamic programming approach to optimal weight selection in multilayer neural networks," *IEEE Transactions on Neural Networks*, Vol. 2, No. 4, pp. 465-467, July 1991.
[24] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.