Application Considerations for the DHP Methodology

George G. Lendaris[1], Thaddeus Shannon[2]

1) Professor of Systems Science and Electrical Engineering
2) Graduate Student, Systems Science Ph.D. Program
Portland State University
PO Box 751
Portland, Oregon 97207

IJCNN’98, Anchorage

SESSION on “Intelligent” Control

Generally, the human designer of the control system has lots of work to do before turning (remainder of) the design task over to our “intelligent” (aka: Neural Networks?) controller.

We here ASSUME that many design decisions have already been made:

e.g.

- need for reinforcement learning approach (due to absence of certain otherwise needed information)
- representation of plant (state?) variables/ measurables
- criterion function (re. "desired" qualities)
- basic architecture of controller neural network

Context (re. Reinforcement Learning):

<u>Adaptive Critic Design (ACD)</u>

A methodology for adaptively designing an (approximately) optimal controller for a given plant according to a stated criterion.

We use a NN as the controller [actionNN], and another NN [criticNN] to update (assist in the design of) the controller.

"Plant" is represented via state vector R(t).

Context:

The ACD method entails the user defining a (primary) utility function $U(t)$ for the specific application, and then maximizing a new utility function (Bellman Eqn.):

$$J(t) = \sum_{k=0}^{\infty} \gamma^k U(t+k)$$

[We note: $J(t) = U(t) + \gamma J(t+1)$

Context: Family of Adaptive Critic Designs

The criticNN approximates either $J(t)$ or gradient of $J(t)$ wrt state vector $R(t)$ $[\nabla J(R)]$

❖Heuristic Dynamic Programming (HDP)

CriticNN approximates $J(t)$

❖Dual Heuristic Programming (DHP)

CriticNN approximates $\nabla J(R) \equiv \lambda$

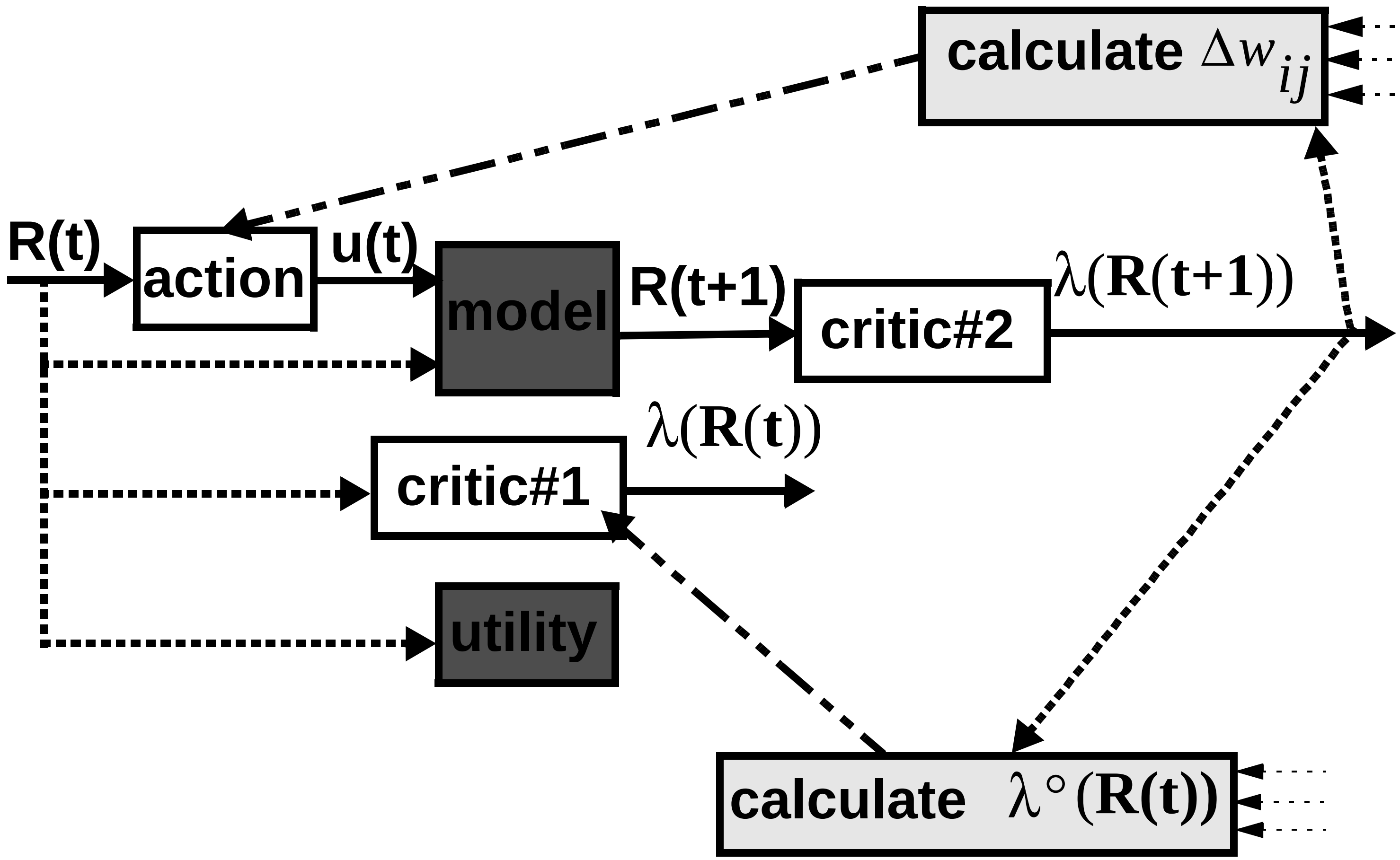
❖Generalized Dual HeuristicProgramming(GDHP)

CriticNN approximates $J(t)$ and $\nabla J(R)$

❖Action Dependent ....

CriticNN also inputs $u(t)$ and outputs$\nabla J(u)$

Computing Schema for discussing Strategies



Weights in actionNN are updated with objective of maximizing J(t):

$$\Delta w_{ij}(t) = lcoef \bullet \frac{\partial}{\partial w_{ij}(t)} J(t)$$

where

$$\frac{\partial}{\partial w_{ij}(t)} J(t) = \sum_{k=1}^{u} \frac{\partial}{\partial u_k(t)} J(t) \bullet \frac{\partial}{\partial w_{ij}(t)} u_k(t)$$

and

$$\frac{\partial}{\partial u_k(t)} J(t) = \frac{\partial}{\partial u_k(t)} U(t) + \frac{\partial}{\partial u_k(t)} J(t+1)$$

and

$$\frac{\partial}{\partial u_k(t)} J(t+1) = \sum_{s=1}^{n} \frac{\partial}{\partial R_s(t+1)} J(t+1) \bullet \frac{\partial}{\partial u_k(t)} R_s(t+1)$$

call this term $\lambda(t+1)$ **(to be output of critic)**

CriticNN output is λ.

For training criticNN, “desired output” is λ°.

(cf. Eqn. (6) in paper)

Paraphrase of Eqn. (6) [cf. Eqn. (7) in paper]:

$$\lambda_s^\circ(t) = [\sim\text{Utility}] + \sum_{j=1}^{a} ([\sim\text{Utility}] \bullet [\sim\text{Action}])$$

$$+ \sum_{k=1}^{n} ([\sim\text{Critic(t+1)}] \bullet [\sim\text{Plant}])$$

$$+ \sum_{k=1}^{n} \left\{ \sum_{j=1}^{a} ([\sim\text{Critic(t+1)}] \bullet [\sim\text{Plant}] \bullet [\sim\text{Action}]) \right\}$$

Strategies to solve Eqn. (6) [and (7)]

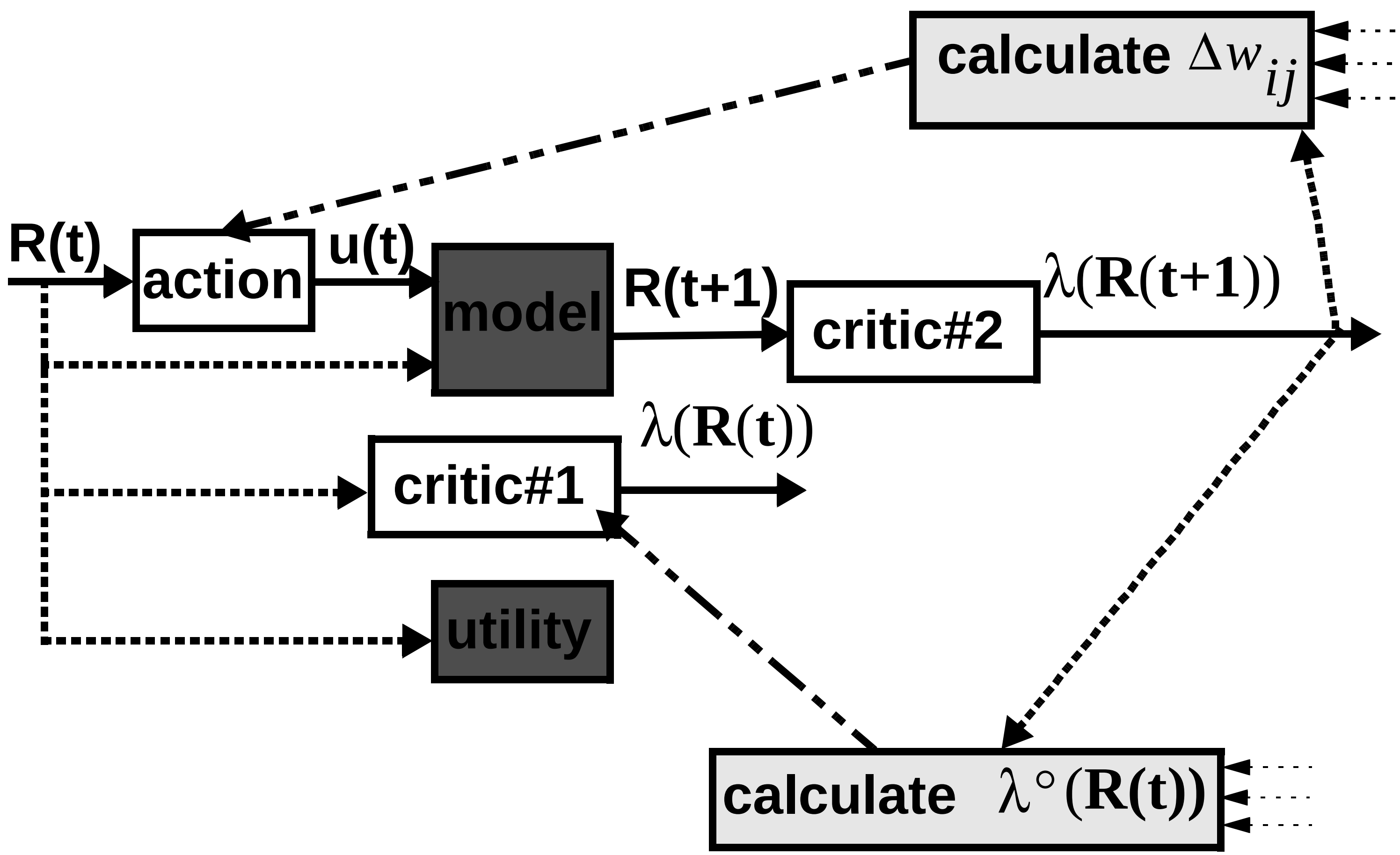
Strategy 1. Straight application of the equation.

Strategy 2. Basic 2-stage process [“flip/flop”].
[e.g., Santiago/Werbos, Prokhorov/Wunsch]
During stage 1, train criticNN, not actionNN;
During stage 2, train actionNN, not criticNN.

Strategy 3. Modified 1st stage of 2-stage process.
While train criticNN during stage 1, keep parameters constant in module that calculates critic’s desired output $\lambda^{\circ}(\mathbf{R})$. Then adjust weights all at once at end of stage 1.

Strategy 4. Single-stage process, using modifications introduced in Strategy 3.

Computing Schema for discussing Strategies



Recap re. criticNN:

Performs mapping: $\lambda(\mathbf{R})$

Desired output for training purposes: $\lambda^{\circ}(\mathbf{R})$

Solution (not known) of Bellman equation: λ^$(\mathbf{R})$

Learn process: $\lambda(\mathbf{R})$ **is to converge to** $\lambda^{\circ}(\mathbf{R})$;

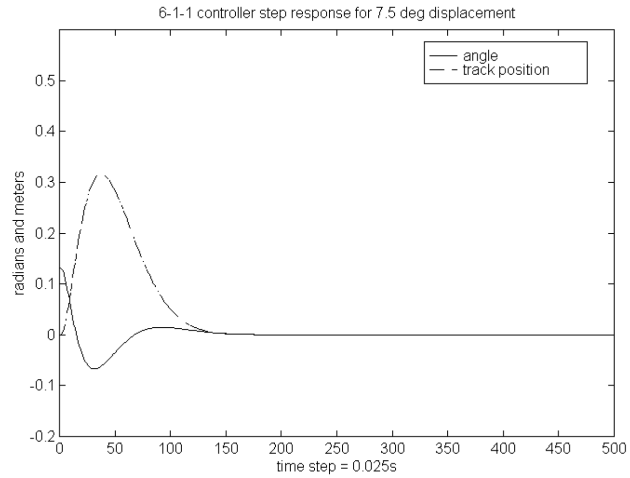
$\lambda^{\circ}(\mathbf{R})$ **is to converge to** λ^$(\mathbf{R})$.

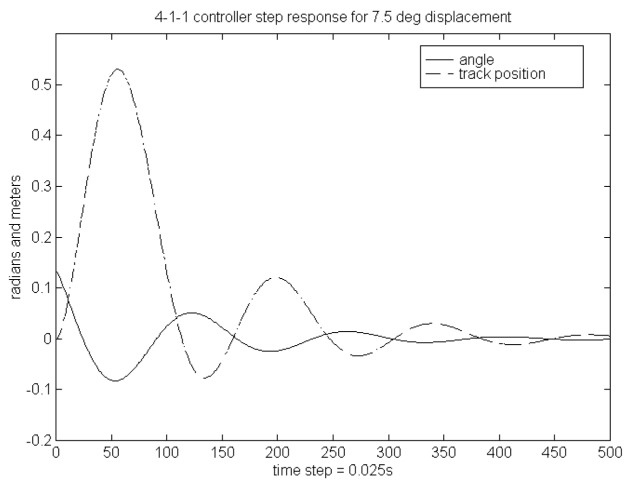
i.e., $\lambda(\mathbf{R})$ $\lambda^{\circ}(\mathbf{R})$ λ^$(\mathbf{R})$

[The better the criticNN "solves" the Bellman eqn., the better the actionNN will approximate an optimal controller.]

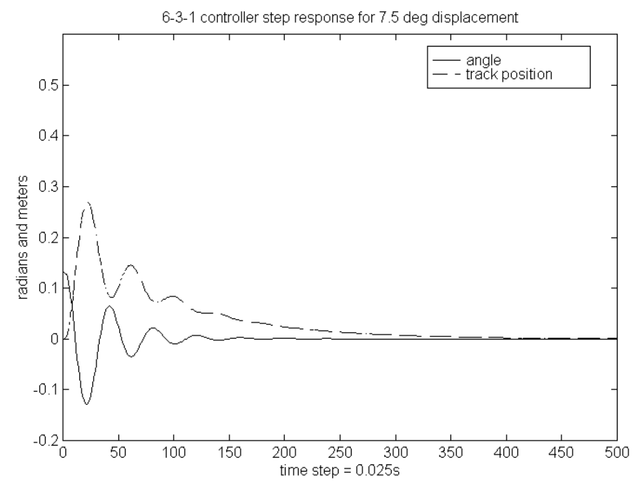
Step Responses to 7.5^{o} pole displacement

6-1-1 Controller





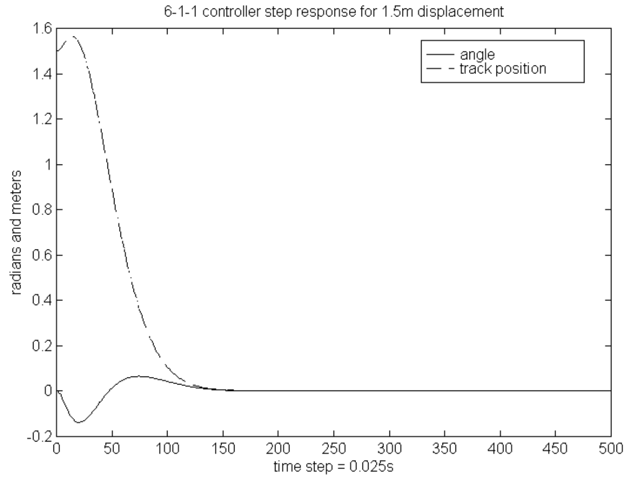
4-1-1 Controller

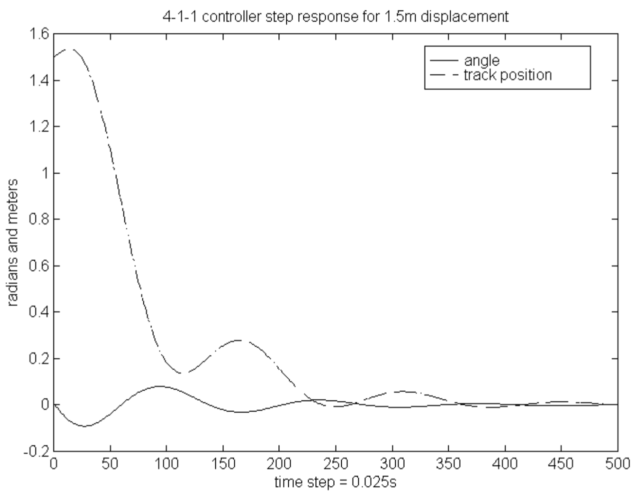


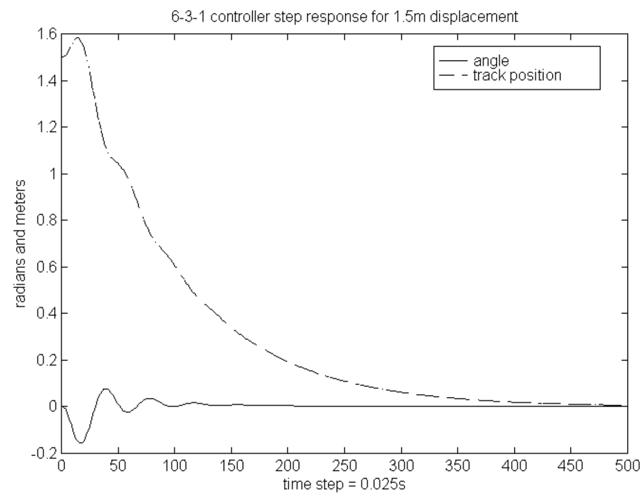
6-3-1 Controller

Step Responses to 1.5m cart displacement

6-1-1 Controller







4-1-1 Controller

6-3-1 Controller

Step Responses of 6-1-1 Controller, 1m pole

[Trained w/ max ± 10^o]

[No explicit X training]

Trained: 7.5^{o}displ.

Tested: 38^{o}displ.

Tested: -6.6m displ.