Optimal Control of a Double Inverted Pendulum on a Cart

Technical Report · December 2004

CITATIONS
90

READS
7,122

1 author:

Alexander Bogdanov
Inertial Sense Inc., United States
29 PUBLICATIONS **478** CITATIONS

SEE PROFILE

Optimal Control of a Double Inverted Pendulum on a Cart

Alexander Bogdanov *
Department of Computer Science & Electrical Engineering,
OGI School of Science & Engineering, OHSU

Technical Report CSE-04-006
December 2004

In this report a number of algorithms for optimal control of a double inverted pendulum on a cart (DIPC) are investigated and compared. Modeling is based on Euler-Lagrange equations derived by specifying a Lagrangian, difference between kinetic and potential energy of the DIPC system. This results in a system of nonlinear differential equations consisting of three 2-nd order equations. This system of equations is then transformed into a usual form of six 1-st order ordinary differential equations (ODE) for control design purposes. Control of a DIPC poses a certain challenge, since unlike a robot, the system is underactuated: one controlling force per three degrees of freedom (DOF). In this report, problem of optimal control minimizing a quadratic cost functional is addressed. Several approaches are tested: linear quadratic regulator (LQR), state-dependent Riccati equation (SDRE), optimal neural network (NN) control, and combinations of the NN with the LQR and the SDRE. Simulations reveal superior performance of the SDRE over the LQR and improvements provided by the NN, which compensates for model inadequacies in the LQR. Limited capabilities of the NN to approximate functions over the wide range of arguments prevent it from significantly improving the SDRE performance, providing only marginal benefits at larger pendulum deflections.

1 Nomenclature

m	mass
l_i	distance from a pivot joint to the i-th pendulum link center of mass
L_i	length of an i-th pendulum link
θ_0	wheeled cart position
θ_1, θ_2	pendulum angles
I_i	moment of inertia of i-th pendulum link w.r.t. its center of mass
g	gravity constant
u	control force
T	kinetic energy
P	potential energy
L	Lagrangian
$\mathbf{w}$	neural network (NN) weights
N_h	number of neurons in the hidden layer
Δt	digital control sampling time

Subscripts $0, 1, 2$ used with the aforementioned parameters refer to the cart, first (bottom) pendulum and second (top) pendulum correspondingly.

2 Introduction

As a nonlinear underactuated plant, double inverted pendulum on a cart (DIPC) poses a challenging control problem. It seems to have been one of attractive tools for testing linear and nonlinear control laws [6, 17, 1]. Numerous papers use DIPC as a testbed. Cited ones are merely an example. Nearly all works on pendulum control concentrate on two problems: pendulums swing-up control design and stabilization of the inverted pendulums. In this report, optimal nonlinear stabilization problem is addressed: stabilize DIPC minimizing an accumulative cost functional quadratic in states and controls. For linear systems, this leads to linear feedback control, which is found by solving a Riccati equation [5], and thus referred to as linear quadratic regulator (LQR). DIPC, however, is a highly nonlinear system, and its linearization is far from adequate for control design purposes. Nonetheless, the LQR will be considered as a baseline controller, and other control designs will be tested and compared against it. To solve the nonlinear optimal control problem, we will employ several different approaches: a nonlinear extension to the LQR called SDRE; direct nonlinear optimization using a neural network (NN); and combinations of the direct NN optimization with the LQR and the SDRE.

For a baseline nonlinear control, we will utilize a technique that has shown considerable promise and involves manipulating the system dynamic equations into a pseudo-linear state-dependent coefficient (SDC) form, in which system matrices are given explicitly as a function of the current state. Treating the system matrices as constant, the approximate solution of the nonlinear state-dependent Riccati equation is obtained for the reformulated pseudo-linear dynamical system in discrete time steps. The solution is then used to calculate a feedback control law that is optimized around the system state estimated at each time step. This technique, referred to as *State-Dependent Riccati Equation* (SDRE) control, is thus an extension to the LQR as it solves the LQR problem at each time step.

As a next step, a direct optimization approach will be investigated. For this purpose, a NN will be used in the feedback loop, and a standard calculus of variations approach will be employed to adjust the NN parameters (weights) to optimize the cost functional over a wide

*Senior Research Associate, alexb@cse.ogi.edu

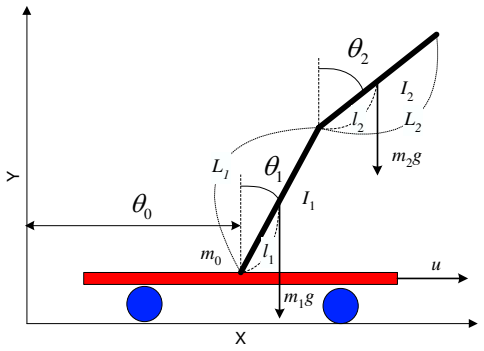

Figure 1: Double inverted pendulum on a cart

range of initial DIPC states. As a universal function approximator, the NN is thus trained to implement a nonlinear optimal controller.

As the last step, two combinations of feedback NN control with LQR and SDRE will be designed. Approximation capabilities of a NN are limited by its size and the fact that optimization in the space of its weights is non-convex. Thus, only local minimums are usually found, and the solution is at most suboptimal. The problem is more severe with wider ranges of NN inputs and outputs. To address this, a combination of the NN with a conventional feedback suboptimal control is designed to simplify the NN input-output mapping and therefore reduce training complexity. If the conventional feedback control were optimal, the optimal NN output would trivially be zero. Simple logic says that a suboptimal conventional control will simplify the NN mapping to some extent: instead of generating optimal controls, the NN is trained to produce *corrections* to the controls generated by the conventional suboptimal controller. For example, the LQR provides near-optimal control in the vicinity of the equilibrium, since nonlinear and linearized DIPC dynamics are close in the equilibrium. Thus the NN will only have to correct the LQR output when the linearization accuracy diminishes. The next sections will discuss the above concepts in details and illustrate them with simulations.

3 Modeling

The DIPC system is graphically depicted in Fig. 1. To derive its equations of motion, one of the possible ways is to use Lagrange equations:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = Q \qquad (1)$$

where $L = T - P$ is a Lagrangian, Q is a vector of generalized forces (or moments) acting in the direction of generalized coordinates θ and not accounted for in formulation of kinetic energy T and potential energy P.

Kinetic and potential energies of the system are given by the sum of energies of its individual components (a wheeled cart and two pendulums):

$$T = T_0 + T_1 + T_2$$

$$P = P_0 + P_1 + P_2$$

where

$$\begin{aligned}
T_0 &= \frac{1}{2} m_0 \dot{\theta}_0^2 \\
T_1 &= \frac{1}{2} m_1 \Big[\big(\dot{\theta}_0 + l_1 \dot{\theta}_1 \cos\theta_1\big)^2 + \big(l_1 \dot{\theta}_1 \sin\theta_1\big)^2 \Big] \\
&+ \frac{1}{2} I_1 \dot{\theta}_1^2 \\
&= \frac{1}{2} m_1 \dot{\theta}_0^2 + \frac{1}{2}\big(m_1 l_1^2 + I_1\big)\dot{\theta}_1^2 + m_1 l_1 \dot{\theta}_0 \dot{\theta}_1 \cos\theta_1 \\
T_2 &= \frac{1}{2} m_2 \Big[\big(\dot{\theta}_0 + L_1 \dot{\theta}_1 \cos\theta_1 + l_2 \dot{\theta}_2 \cos\theta_2\big)^2 \\
&+ \big(L_1 \dot{\theta}_1 \sin\theta_1 + l_2 \dot{\theta}_2 \sin\theta_2\big)^2 \Big] + \frac{1}{2} I_2 \dot{\theta}_2^2 \\
&= \frac{1}{2} m_2 \dot{\theta}_0^2 + \frac{1}{2} m_2 L_1^2 \dot{\theta}_1^2 + \frac{1}{2}\big(m_2 l_2^2 + I_2\big)\dot{\theta}_2^2 \\
&+ m_2 L_1 \dot{\theta}_0 \dot{\theta}_1 \cos\theta_1 + m_2 l_2 \dot{\theta}_0 \dot{\theta}_2 \cos\theta_2 \\
&+ m_2 L_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2) \\
P_0 &= 0 \\
P_1 &= m_1 g l_1 \cos\theta_1 \\
P_2 &= m_2 g \big(L_1 \cos\theta_1 + l_2 \cos\theta_2\big)
\end{aligned}$$

Thus the Lagrangian of the system is given by

$$\begin{aligned}
L &= \frac{1}{2}\big(m_0 + m_1 + m_2\big)\dot{\theta}_0^2 \\
&+ \frac{1}{2}\big(m_1 l_1^2 + m_2 L_1^2 + I_1\big)\dot{\theta}_1^2 + \frac{1}{2}\big(m_2 l_2^2 + I_2\big)\dot{\theta}_2^2 \\
&+ \big(m_1 l_1 + m_2 L_1\big)\cos(\theta_1)\dot{\theta}_0\dot{\theta}_1 \\
&+ m_2 l_2 \cos(\theta_2)\dot{\theta}_0\dot{\theta}_2 + m_2 L_1 l_2 \cos(\theta_1 - \theta_2)\dot{\theta}_1\dot{\theta}_2 \\
&- \big(m_1 l_1 + m_2 L_1\big) g \cos\theta_1 - m_2 l_2 g \cos\theta_2
\end{aligned}$$

Differentiating the Lagrangian by $\dot{\theta}$ and θ yields Lagrange equation (1) as

$$\begin{aligned}
\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_0}\right) - \frac{\partial L}{\partial \theta_0} &= u \\
\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_1}\right) - \frac{\partial L}{\partial \theta_1} &= 0 \\
\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_2}\right) - \frac{\partial L}{\partial \theta_2} &= 0
\end{aligned}$$

Or explicitly,

$$\begin{aligned}
u &= \Big(\sum m_i\Big)\ddot{\theta}_0 + \big(m_1 l_1 + m_2 L_1\big)\cos(\theta_1)\ddot{\theta}_1 \\
&+ m_2 l_2 \cos(\theta_2)\ddot{\theta}_2 - \big(m_1 l_1 + m_2 L_1\big)\sin(\theta_1)\dot{\theta}_1^2 \\
&- m_2 l_2 \sin(\theta_2)\dot{\theta}_2^2 \\
0 &= \big(m_1 l_1 + m_2 L_1\big)\cos(\theta_1)\ddot{\theta}_0 \\
&+ \big(m_1 l_1^2 + m_2 L_1^2 + I_1\big)\ddot{\theta}_1 \\
&+ m_2 L_1 l_2 \cos(\theta_1 - \theta_2)\ddot{\theta}_2 \\
&+ m_2 L_1 l_2 \sin(\theta_1 - \theta_2)\dot{\theta}_2^2 \\
&- \big(m_1 l_1 + m_2 L_1\big) g \sin\theta_1 \\
0 &= m_2 l_2 \cos(\theta_2)\ddot{\theta}_0 + m_2 L_1 l_2 \cos(\theta_1 - \theta_2)\ddot{\theta}_1 \\
&+ \big(m_2 l_2^2 + I_2\big)\ddot{\theta}_2 - m_2 L_1 l_2 \sin(\theta_1 - \theta_2)\dot{\theta}_1^2 \\
&- m_2 l_2 g \sin\theta_2
\end{aligned}$$

Lagrange equations for the DIPC system can be written in a more compact matrix form:

$$\mathbf{D}(\theta)\ddot{\theta} + \mathbf{C}(\theta,\dot{\theta})\dot{\theta} + \mathbf{G}(\theta) = \mathbf{H}u \tag{2}$$

where

$$\mathbf{D}(\theta) = \begin{pmatrix} d_1 & d_2\cos\theta_1 & d_3\cos\theta_2 \\ d_2\cos\theta_1 & d_4 & d_5\cos(\theta_1-\theta_2) \\ d_3\cos\theta_2 & d_5\cos(\theta_1-\theta_2) & d_6 \end{pmatrix} \tag{3}$$

$$\mathbf{C}(\theta,\dot{\theta}) = \begin{pmatrix} 0 & -d_2\sin(\theta_1)\dot{\theta}_1 & -d_3\sin(\theta_2)\dot{\theta}_2 \\ 0 & 0 & d_5\sin(\theta_1-\theta_2)\dot{\theta}_2 \\ 0 & -d_5\sin(\theta_1-\theta_2)\dot{\theta}_1 & 0 \end{pmatrix} \tag{4}$$

$$\mathbf{G}(\theta) = \begin{pmatrix} 0 \\ -f_1\sin\theta_1 \\ -f_2\sin\theta_2 \end{pmatrix} \tag{5}$$

$$\mathbf{H} = (1\ 0\ 0)^T$$

Assuming that centers of mass of the pendulums are in the geometrical center of the links, which are solid rods, we have: $l_i = L_i/2$, $I_i = m_i L_i^2/12$. Then for the elements of matrices $\mathbf{D}(\theta)$, $\mathbf{C}(\theta,\dot{\theta})$, and $\mathbf{G}(\theta)$ we get:

$$\begin{aligned}
d_1 &= m_0 + m_1 + m_2 \\
d_2 &= m_1 l_1 + m_2 L1 = (\frac{1}{2}m_1 + m_2)L_1 \\
d_3 &= m_2 l_2 = \frac{1}{2}m_2 L_2 \\
d_4 &= m_1 l_1^2 + m_2 L_1^2 + I_1 = (\frac{1}{3}m_1 + m_2)L_1^2 \\
d_5 &= m_2 L_1 l_2 = \frac{1}{2}m_2 L_1 L_2 \\
d_6 &= m_2 l_2^2 + I_2 = \frac{1}{3}m_2 L_2^2 \\
f_1 &= (m_1 l_1 + m_2 L_1)g = (\frac{1}{2}m_1 + m_2)L_1 g \\
f_2 &= m_2 l_2 g = \frac{1}{2}m_2 L_2 g
\end{aligned}$$

Note that matrix $\mathbf{D}(\theta)$ is symmetric and nonsingular.

4 Control

To design a control law, Lagrange equations of motion (2) are reformulated into a 6-th order system of ordinary differential equations. To do this, a state vector $\mathbf{x} \in R^6$ is introduced:

$$\mathbf{x} = (\theta\ \dot{\theta})^T$$

Then dropping dependencies of the system matrices on the generalized coordinates and their derivatives, the system dynamic equations appear as:

$$\dot{\mathbf{x}} = \begin{pmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & -\mathbf{D}^{-1}\mathbf{C} \end{pmatrix}\mathbf{x} + \begin{pmatrix} \mathbf{0} \\ -\mathbf{D}^{-1}\mathbf{G} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{D}^{-1}\mathbf{H} \end{pmatrix} u \tag{6}$$

In this report, optimal nonlinear stabilization control design is addressed: stabilize the DIPC minimizing an accumulative cost functional quadratic in states and controls. The general problem of designing an optimal control law involves minimizing a cost function

$$J_t = \sum_{k=t}^{t_{final}} L_k(\mathbf{x}_k, \mathbf{u}_k), \tag{7}$$

which represents an accumulated cost of the sequence of states $\mathbf{x}_k$ and controls $\mathbf{u}_k$ from the current discrete time t to the final time t_{final}. For *regulation* problems $t_{final} = \infty$. Optimization is done with respect to the control sequence subject to constraints of the system dynamics (6). In our case,

$$L_k(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{x}_k^T \mathbf{Q}\mathbf{x}_k + \mathbf{u}_k^T \mathbf{R}\mathbf{u}_k \tag{8}$$

corresponds to the standard *Linear Quadratic* cost. For *linear* systems, this leads to linear state-feedback control, LQR, designed in the next subsection. For nonlinear systems the optimal control problem generally requires a numerical solution, which can be computationally prohibitive. An analytical approximation to the nonlinear optimal control solution is utilized in subsection on SDRE control, which represents a nonlinear extension to the LQR and yields superior results. Neural network (NN) capabilities for function approximation are employed to approximate the nonlinear control solution in subsection on NN control. And combinations of the NN with LQR and SDRE are investigated in the subsection following the NN control.

4.1 Linear Quadratic Regulator

The linear quadratic regulator yields an optimal solution to the control problem (7)–(8) when system dynamics are linear. Since DIPC is nonlinear, as described by (6), it can be linearized to derive an approximate linear solution to the optimal control problem. Linearization of (6) around $\mathbf{x} = \mathbf{0}$ yields:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u \tag{9}$$

where

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{D}(\mathbf{0})^{-1}\frac{\partial \mathbf{G}(\mathbf{0})}{\partial \theta} & \mathbf{0} \end{pmatrix} \tag{10}$$

$$\mathbf{B} = \begin{pmatrix} \mathbf{0} \\ \mathbf{D}(\mathbf{0})^{-1}\mathbf{H} \end{pmatrix} \tag{11}$$

and the continuous LQR solution is obtained then by:

$$u = -\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P_c}\mathbf{x} \equiv -\mathbf{K_c}\mathbf{x} \tag{12}$$

where $\mathbf{P_c}$ is a steady-state solution of the differential Riccati equation. To implement computerized digital control, dynamic equations (9) are approximately discretized as $\mathbf{\Phi} \approx e^{\mathbf{A}\Delta t}$, $\mathbf{\Gamma} \approx \mathbf{B}\Delta t$, and digital LQR control is then given by

$$u_k = -\mathbf{R}^{-1}\mathbf{\Gamma}^T\mathbf{P}\mathbf{x}_k \equiv -\mathbf{K}\mathbf{x}_k \tag{13}$$

where $\mathbf{P}$ is the steady state solution of the difference Riccati equation, obtained by solving the discrete-time algebraic Riccati equation

$$\mathbf{\Phi}^T[\mathbf{P} - \mathbf{P}\mathbf{\Gamma}(\mathbf{R} + \mathbf{\Gamma}^T\mathbf{P}\mathbf{\Gamma})^{-1}\mathbf{\Gamma}^T\mathbf{P}]\mathbf{\Phi} - \mathbf{P} + \mathbf{Q} = \mathbf{0} \tag{14}$$

where $\mathbf{Q} \in R^{6\times 6}$ and $\mathbf{R} \in R$ are positive definite state and control cost matrices. Since linearization (9)–(11) accurately represents the DIPC system (6) in the equilibrium, the LQR control (12) or (13) will be a locally near-optimal stabilizing control.

4.2 State-Dependent Riccati Equation Control

An approximate nonlinear analytical solution of the optimal control problem (7)–(8) subject to (6) is given by a technique referred to as the state-dependent Riccati equation (SDRE) control. The SDRE approach [3] involves manipulating the dynamic equations

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$$

into a pseudo-linear state-dependent coefficient (SDC) form in which system matrices are explicit functions of the current state:

$$\dot{\mathbf{x}} = \mathbf{A}(\mathbf{x})\mathbf{x} + \mathbf{B}(\mathbf{x})\mathbf{u} \tag{15}$$

A standard LQR problem (Riccati equation) can then be solved at each time step to design the state feedback control law on-line. For digital implementation, (15) is approximately discretized at each time step into

$$\mathbf{x}_{k+1} = \mathbf{\Phi}(\mathbf{x}_k)\mathbf{x}_k + \mathbf{\Gamma}(\mathbf{x}_k)\mathbf{u}_k \tag{16}$$

And the SDRE regulator is then specified similar to the discrete LQR (compare with (13)) as

$$\mathbf{u}_k = -\mathbf{R}^{-1}\mathbf{\Gamma}^T(\mathbf{x}_k)\mathbf{P}(\mathbf{x}_k)\mathbf{x}_k \equiv -\mathbf{K}(\mathbf{x}_k)\mathbf{x}_k \tag{17}$$

where $\mathbf{P}(\mathbf{x}_k)$ is the steady state solution of the difference Riccati equation, obtained by solving the discrete-time algebraic Riccati equation (14) using state-dependent matrices $\mathbf{\Phi}(\mathbf{x}_k)$ and $\mathbf{\Gamma}(\mathbf{x}_k)$, which are treated as being constant at each time step. Thus the approach is sometimes considered as a nonlinear extension to the LQR.

Proposition 1. *Dynamic equations of a double inverted pendulum on a cart are presentable in SDC form (15).*

Proof. From the derived dynamic equations for the DIPC system (6) it is clear that the required SDC form (15) can be obtained if vector $\mathbf{G}(\theta)$ is presentable in the SDC form: $\mathbf{G}(\theta) = \mathbf{G}_{sd}(\theta)\theta$. Let us construct $\mathbf{G}_{sd}(\theta)$ as

$$\mathbf{G}_{sd}(\theta) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -f_1\frac{\sin\theta_1}{\theta_1} & 0 \\ 0 & 0 & -f_2\frac{\sin\theta_2}{\theta_2} \end{pmatrix}$$

Elements of constructed $\mathbf{G}_{sd}(\theta)$ are bounded everywhere and $\mathbf{G}(\theta) = \mathbf{G}_{sd}(\theta)\theta$ as required. Thus the system dynamic equations can be presented in the SDC form as

$$\dot{\mathbf{x}} = \begin{pmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{D}^{-1}\mathbf{G}_{sd} & -\mathbf{D}^{-1}\mathbf{C} \end{pmatrix}\mathbf{x} + \begin{pmatrix} \mathbf{0} \\ \mathbf{D}^{-1}\mathbf{H} \end{pmatrix} u \tag{18}$$

□

Derived system equations (18) (compare with the linearized system (9)–(11)) are discretized at each time step into (16), and control is then computed as given by (17).

Remark. In the neighborhood of equilibrium $\mathbf{x} = \mathbf{0}$ system equations in the SDC form (18) turn into linearized equations (9) used in the LQR design. This can be checked either by direct computation or by noting that on one hand, linearization yields

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}u + \mathbf{O}(\mathbf{x})^2$$

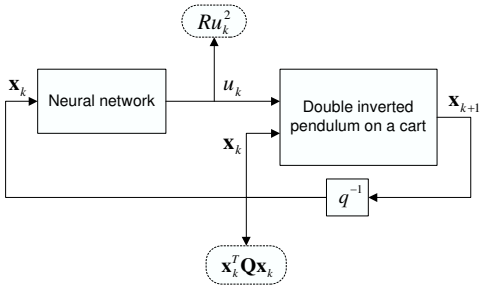

Figure 2: Neural network control diagram

and on the other hand, SDC form of the system dynamics is

$$\dot{\mathbf{x}} = \mathbf{A}(\mathbf{x})\mathbf{x} + \mathbf{B}(\mathbf{x})u$$

Since $\mathbf{O}(\mathbf{x})^2 \to \mathbf{0}$ when $\mathbf{x} \to \mathbf{0}$, then $\mathbf{A}(\mathbf{x}) \to \mathbf{A}$ and $\mathbf{B}(\mathbf{x}) \to \mathbf{B}$. Therefore, it is natural to expect that performance of the SDRE regulator will be very close to the LQR in the vicinity of the equilibrium, and the difference will show at larger pendulum deflections. This is exactly illustrated in the *Simulation Results* section.

4.3 Neural Network Learning Control

A neural network (NN) control is often popular in control of nonlinear systems due to universal function approximation capabilities of NNs. Neural networks with only one hidden layer and an arbitrarily large number of neurons represent nonlinear mappings, which can be used for approximation of any nonlinear function $f \in C(R^n, R^m)$ over a compact subset of R^n [7, 4, 12]. In this section, we utilize the function approximation properties of NNs to approximate a solution of the nonlinear optimal control problem (7)–(8) subject to system dynamics (6), and thus design an optimal NN regulator. The problem can be solved by directly implementing a feedback controller (see Figure 2) as:

$$\mathbf{u}_k = NN\left(\mathbf{x}_k, \mathbf{w}\right),$$

Next, an optimal set of weights $\mathbf{w}$ is computed to solve the optimization problem. To do this, a standard calculus of variations approach is taken: minimization of a functional subject to equality constraints $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$. Let λ_k be a vector of Lagrange multipliers in the augmented cost function

$$H = \sum_{k=t}^{t_{final}} \left\{ L(\mathbf{x}_k, \mathbf{u}_k) + \lambda_k^T \left(\mathbf{x}_{k+1} - \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)\right) \right\} \tag{19}$$

We can now derive the recurrent Euler-Lagrange equations by solving $\frac{\partial H}{\partial \mathbf{x}_k} = \mathbf{0}$ w.r.t. the Lagrange multipliers and then find the optimal set of NN weights $\mathbf{w}^\star$ by solving the optimality condition $\frac{\partial H}{\partial \mathbf{w}} = \mathbf{0}$ (numerically by means of gradient descent).

Dropping dependence of $L(\mathbf{x}_k, \mathbf{u}_k)$ and $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ on $\mathbf{x}_k$ and $\mathbf{u}_k$, and $NN(\mathbf{x}_k, \mathbf{w})$ on $\mathbf{x}_k$ and $\mathbf{w}$ for brevity, the

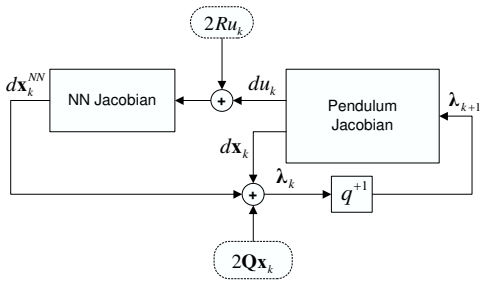

Figure 3: Adjoint system diagram

Euler-Lagrange equations are derived as

$$\begin{aligned}\lambda_k &= \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}_k} + \frac{\partial \mathbf{f}}{\partial \mathbf{u}_k}\frac{\partial NN}{\partial \mathbf{x}_k}\right)^T \lambda_{k+1} \\ &+ \left(\frac{\partial L}{\partial \mathbf{x}_k} + \frac{\partial L}{\partial \mathbf{u}_k}\frac{\partial NN}{\partial \mathbf{x}_k}\right)^T \end{aligned} \quad (20)$$

with $\lambda_{t_{final}}$ initialized as zero vector. For $L(\mathbf{x}_k, \mathbf{u}_k)$ given by (8), $\frac{\partial L}{\partial \mathbf{x}_k} = 2\mathbf{x}_k^T\mathbf{Q}$, and $\frac{\partial L}{\partial \mathbf{u}_k} = 2\mathbf{u}_k^T\mathbf{R}$. These equations correspond to an adjoint system shown graphically in Figure 3, with optimality condition

$$\frac{\partial H}{\partial \mathbf{w}} = \sum_{k=t}^{t_{final}} \left\{ \left(\lambda_{k+1}^T \frac{\partial \mathbf{f}}{\partial \mathbf{u}_k} + \frac{\partial L}{\partial \mathbf{u}_k}\right)\frac{\partial NN}{\partial \mathbf{w}} \right\} = \mathbf{0}. \quad (21)$$

The overall training procedure for the NN can now be summarized as follows:

1. Simulate the system forward in time for t_{final} time steps (Figure 2). Although, as we mentioned, $t_{final} = \infty$ for regulation problems, in practice it is set to a sufficiently large number (in our case $t_{final} = 500$).
2. Run the adjoint system backward in time to accumulate the Lagrange multipliers (20) (see Figure 3). System Jacobians are evaluated analytically or by perturbation.
3. Update the weights using gradient descent[1], $\Delta\mathbf{w} = -\gamma\left(\frac{\partial H}{\partial \mathbf{w}}\right)^T$.
4. Repeat until convergence or until an acceptable level of cost reduction is achieved.

Due to the nature of the training process (propagation of the Lagrange multipliers backwards through time), this training algorithm is referred to as Back Propagation Through Time (BPTT) [12, 16].

As seen from Euler-Lagrange equations (20), back propagation of the Lagrange multipliers includes computation of the system and NN Jacobians. The DIPC Jacobians can be computed numerically by perturbation or derived analytically from the dynamic equations (6). Let us show the analytically derived Jacobians $\frac{\partial \mathbf{f}_c(\mathbf{x},u)}{\partial \mathbf{x}}$ and $\frac{\partial \mathbf{f}_c(\mathbf{x},u)}{\partial u}$, where $\mathbf{f}_c(\mathbf{x}, u)$ is a brief notation for the right-hand side of the *continuous* system (6), i.e. $\dot{\mathbf{x}} = \mathbf{f}_c(\mathbf{x}, u)$.

$$\frac{\partial \mathbf{f}_c(\mathbf{x},u)}{\partial \mathbf{x}} = \begin{pmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{D}^{-1}\mathbf{M} & -2\mathbf{D}^{-1}\mathbf{C} \end{pmatrix} \quad (22)$$

$$\frac{\partial \mathbf{f}_c(\mathbf{x},u)}{\partial u} = \begin{pmatrix} \mathbf{0} \\ \mathbf{D}^{-1}\mathbf{H} \end{pmatrix} \quad (23)$$

where matrix $\mathbf{M}(\theta, \dot{\theta}) = (\mathbf{M}_0\ \mathbf{M}_1\ \mathbf{M}_2)$, and each of its columns (note that $\frac{\partial \mathbf{D}^{-1}}{\partial \theta_i} = -\mathbf{D}^{-1}\frac{\partial \mathbf{D}}{\partial \theta_i}\mathbf{D}^{-1}$) is given by

$$\mathbf{M}_i = \frac{\partial \mathbf{C}}{\partial \theta_i}\dot{\theta} + \frac{\partial \mathbf{G}}{\partial \theta_i} - \frac{\partial \mathbf{D}}{\partial \theta_i}\mathbf{D}^{-1}(\mathbf{C}\dot{\theta} + \mathbf{G} - \mathbf{H}u) \quad (24)$$

Remark 1. Clearly, Jacobians (22)–(24) transform into linear system matrices (10)–(11) in equilibrium $\theta = \mathbf{0}$, $\dot{\theta} = \mathbf{0}$.

Remark 2. From (3)–(5) it follows that $\mathbf{M}_0 \equiv \mathbf{0}$.

Remark 3. Jacobians (22)–(24) are derived from the continuous system equations (6). BPTT requires computation of the discrete system Jacobians. Thus, to use the derived matrices in NN training, they should be discretized (e.g. as it was done for the LQR and SDRE).

Computation of the NN Jacobian is easy to perform given the nonlinear functions of individual neural elements are $y = \tanh(z)$. In this case, NN with N_0 inputs, single output u and a single hidden layer with N_h elements is described by

$$u = \sum_{i=1}^{N_h}\left[w_{2_i}\tanh\left(\sum_{j=1}^{N_0} w_{1_{i,j}}x_j + w_{1_i}^b\right)\right] + w_2^b$$

Or in a more compact form,

$$u = \mathbf{w}_2\mathbf{tanh}\left(\mathbf{W}_1\mathbf{x} + \mathbf{w}_1^b\right) + w_2^b$$

where $\mathbf{tanh}(\mathbf{z}) = (\tanh(z_1), \ldots, \tanh(z_{N_h}))^T$, $\mathbf{w}_2 \in R^{N_h}$ is a row-vector of weights in the NN output, $\mathbf{W}_1 \in R^{N_h \times N_0}$ is a matrix of weights of the hidden layer elements, $\mathbf{w}_1^b \in R^{N_h}$ is a vector of bias weights in the hidden layer and w_2^b is a bias weight in the NN output. Graphically NN is depicted in Figure 4.

Remark 4. In case of a NN with N_{out} outputs (MIMO control problems), $\mathbf{w}_2$ becomes a matrix $\mathbf{W}_2 \in R^{N_{out} \times N_h}$, and w_2^b becomes a vector $\mathbf{w}_2^b \in R^{N_{out}}$.

Noting that $\frac{\partial \tanh(z)}{\partial z} = 1 - \tanh^2(z)$, the NN Jacobian w.r.t. the NN input (state vector $\mathbf{x}$) is given by

$$\frac{\partial NN}{\partial \mathbf{x}} = \left(\mathbf{w}_2^T \odot \mathbf{dtanh}\left(\mathbf{W}_1\mathbf{x} + \mathbf{w}_1^b\right)\right)^T \mathbf{W}_1 \quad (25)$$

where operator $\odot$ is an element-by-element multiplication of two vectors, i.e. $\mathbf{x} \odot \mathbf{y} = (x_1y_1, \ldots, x_ny_n)^T$; and vector field $\mathbf{dtanh}(\mathbf{z}) = \left(1 - \tanh^2(z_1), \ldots, 1 - \tanh^2(z_{N_h})\right)^T$. Again, in MIMO case when the NN has N_{out} outputs, individual rows of $\mathbf{W}_2$ take place of row-vector $\mathbf{w}_2$ in (25).

[1] In practice we use an adaptive learning rate for each weight in the network using a procedure similar to delta-bar-delta [8]

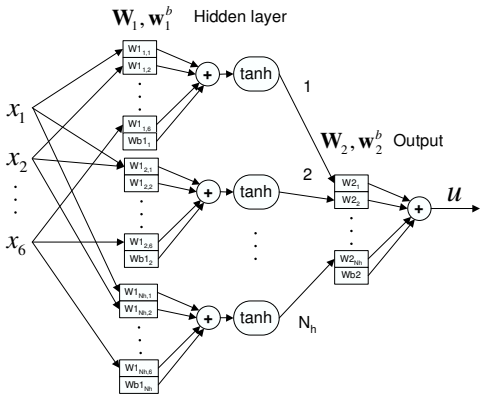

Figure 4: Neural network structure

The last essential part of the NN training algorithm is the NN Jacobian w.r.t. the network weights:

$$\begin{aligned}
\frac{\partial NN}{\partial \mathbf{W}_{1_i}} &= \left(\mathbf{w}_2^T \odot \mathbf{dtanh}\left(\mathbf{W}_1\mathbf{x}+\mathbf{w}_1^b\right)\right)^T x_i \\
\frac{\partial NN}{\partial \mathbf{w}_2} &= \mathbf{tanh}\left(\mathbf{W}_1\mathbf{x}+\mathbf{w}_1^b\right) \qquad (26)\\
\frac{\partial NN}{\partial \mathbf{w}_1^b} &= \left(\mathbf{w}_2^T \odot \mathbf{dtanh}\left(\mathbf{W}_1\mathbf{x}+\mathbf{w}_1^b\right)\right)^T \\
\frac{\partial NN}{\partial w_2^b} &= 1
\end{aligned}$$

where $\mathbf{W}_{1_i}$ is an i-th column of $\mathbf{W}_1$.

Now we have all the quantities to compute weight updates and control. Explicitly, from (21) and (26) the weight update recurrent law is given by

$$\begin{aligned}
\mathbf{W}_{1_{n+1}} &= \mathbf{W}_{1_n} - \gamma \sum_{k=t}^{t_{final}} \left\{ \left[\mathbf{W}_{2_n}^T \left(\lambda_{k+1}^T \frac{\partial \mathbf{f}}{\partial \mathbf{u}_k} + \frac{\partial L}{\partial \mathbf{u}_k} \right)^T \right] \right. \\
&\quad \left. \odot \ \mathbf{dtanh}\left(\mathbf{W}_{1_n}\mathbf{x}_k + \mathbf{w}_{1_n}^b\right)\mathbf{x}_k^T \right\} \\
\mathbf{W}_{2_{n+1}} &= \mathbf{W}_{2_n} - \gamma \sum_{k=t}^{t_{final}} \left\{ \left(\lambda_{k+1}^T \frac{\partial \mathbf{f}}{\partial \mathbf{u}_k} + \frac{\partial L}{\partial \mathbf{u}_k} \right)^T \right. \\
&\quad \left. \times \ \mathbf{tanh}^T\left(\mathbf{W}_{1_n}\mathbf{x}_k + \mathbf{w}_{1_n}^b\right) \right\} \qquad (27)\\
\mathbf{w}_{1_{n+1}}^b &= \mathbf{w}_{1_n}^b - \gamma \sum_{k=t}^{t_{final}} \left\{ \left[\mathbf{W}_{2_n}^T \left(\lambda_{k+1}^T \frac{\partial \mathbf{f}}{\partial \mathbf{u}_k} + \frac{\partial L}{\partial \mathbf{u}_k} \right)^T \right] \right. \\
&\quad \left. \odot \ \mathbf{dtanh}\left(\mathbf{W}_{1_n}\mathbf{x}_k + \mathbf{w}_{1_n}^b\right) \right\} \\
\mathbf{w}_{2_{n+1}}^b &= \mathbf{w}_{2_n}^b - \gamma \sum_{k=t}^{t_{final}} \left(\lambda_{k+1}^T \frac{\partial \mathbf{f}}{\partial \mathbf{u}_k} + \frac{\partial L}{\partial \mathbf{u}_k} \right)^T
\end{aligned}$$

In conclusion to this section, it should be mentioned that the number of elements in the NN hidden layer affects optimality of the control design. One one hand, the more elements, the better the theoretically achievable function approximation capabilities of the NN. On the other hand, too many elements make gradient descent in space of NN weights more prone to getting stuck in local minima (recall, this is a non-convex problem), thus prohibiting achievement of good approximation levels. Therefore a reasonable compromise is usually necessary.

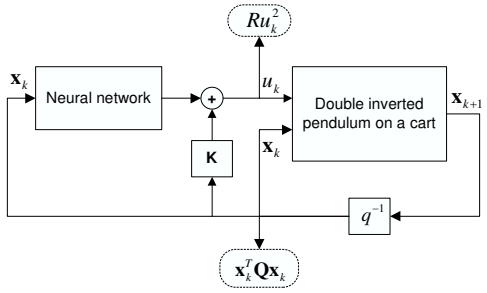

Figure 5: Neural network + LQR control diagram

4.4 Neural Network Control + LQR/SDRE

In the previous section a NN was optimized directly to stabilize the DIPC at minimum cost (7)–(8). To achieve this, the NN was trained to generate optimal controls over the wide range of the pendulums motion. As illustrated in *Simulation Results* section, NN approximation capabilities, limited by the number of neurons and numerical challenges of non-convex minimization, provided achievement of stabilization only within the range comparable to the LQR. One might ask whether it is possible to make the NN training more efficient and faster. Simple logic says that if the DIPC were stable and closer to optimal, the NN would not have much to learn since its optimal output would be closer to zero. In the trivial case, if the DIPC were optimally stabilized by an internal controller, the optimal NN output would be zero. Now let us recall that LQR provides optimal control in the vicinity of the equilibrium. Thus if we include LQR into the control scheme as shown in Figure 5, it is reasonable to expect better overall performance: the NN will be trained to generate only "corrections" to the LQR controls to provide optimality in the wide range of pendulums motion.

Although in Figure 5 an LQR is shown, the SDRE controller can be used as well in place of the LQR. As demonstrated in the *Simulation Results* section, the SDRE control provides superior performance in terms of minimized cost and range of stability over the LQR. Thus, it is natural to expect a better performance of the control design shown in Figure 5 when the SDRE is used in place of the LQR. Since both the LQR and the SDRE have a lot in common (in fact, the SDRE is often called a nonlinear extension to the LQR), both cases will be discussed in this section, and when the differences between the two will call for clarification, it will be provided.

The problem solved in this section is nearly the same as in the previous section, therefore, almost all the formulas stay the same or have just a few changes.

Since control of the cart is a sum of the NN output and the LQR (SDRE),

$$u_k = NN\left(\mathbf{w}, \mathbf{x}_k\right) + \mathbf{K}\mathbf{x}_k$$

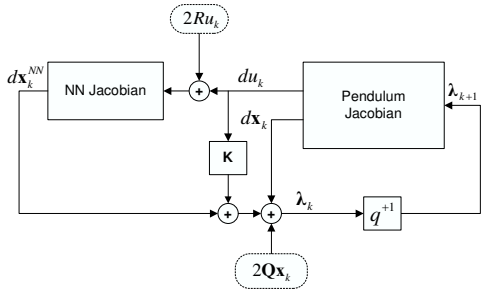

Figure 6: Adjoint NN + LQR system diagram

where in case of SDRE $\mathbf{K} \equiv \mathbf{K}(\mathbf{x}_k)$, minimization of the augmented cost function (19) by taking derivatives of H w.r.t. $\mathbf{x}_k$ and $\mathbf{w}$ yields slightly modified Euler-Lagrange equations:

$$\begin{aligned} \lambda_k &= \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}_k} + \frac{\partial \mathbf{f}}{\partial \mathbf{u}_k}\left(\frac{\partial NN}{\partial \mathbf{x}_k} + \mathbf{K}\right)\right]^T \lambda_{k+1} \\ &+ \left[\frac{\partial L}{\partial \mathbf{x}_k} + \frac{\partial L}{\partial \mathbf{u}_k}\left(\frac{\partial NN}{\partial \mathbf{x}_k} + \mathbf{K}\right)\right]^T \end{aligned} \quad (28)$$

Note that when SDRE is used instead of the LQR, $\frac{\partial(\mathbf{K}(\mathbf{x}_k)\mathbf{x}_k)}{\partial \mathbf{x}_k}$ should be used in place of $\mathbf{K}$ in the above equation. These equations correspond to an adjoint system shown graphically in Figure 6 (compare to Figure 3), with optimality condition (21)

The training procedure for the NN is the same as in the previous section, and all the system Jacobians and the NN Jacobian are computed in the same way. The only new item in this section is the SDRE Jacobian $\frac{\partial(\mathbf{K}(\mathbf{x}_k)\mathbf{x}_k)}{\partial \mathbf{x}_k}$ which appears in the Euler-Lagrange equations (28). This Jacobian is computed either numerically, which may be computationally expensive, or approximately as

$$\frac{\partial\left(\mathbf{K}(\mathbf{x}_k)\mathbf{x}_k\right)}{\partial \mathbf{x}_k} \approx \mathbf{K}\left(\mathbf{x}_k\right)$$

Experiments in the *Simulation Results* section illustrate the superior performance of such combination control over the pure NN or LQR control. For the SDRE, a noticeable cost reduction is achieved only near critical pendulum deflections (close to the boundaries of the SDRE recovery region).

Remark. The idea of using NNs to adjust outputs of a conventional controller to account for differences between the actual system and its model used in the conventional control design was also employed in a number of works [15, 14, 2, 11]. Calise, Rysdyk and Johnson [2, 11] used a NN controller to supplement an approximate feedback linearization control of a fixed wing aircraft and a helicopter. The NN provided additional controls to match the response of the vehicle to the reference model, compensating for the approximations in the autopilot design. Wan and Bogdanov [15, 14] designed a model predictive neural control (MPNC) for an autonomous helicopter, where a NN worked in pair with the SDRE autopilot and provided minimization of the quadratic cost function over a receding horizon. The SDRE control was designed for a simplified nonlinear model, and the NN made it possible to compensate for wind disturbances and higher-order terms unaccounted for in the SDRE design.

4.5 Receding Horizon Neural Network Control

Is it possible to further improve control scheme designed in the previous section? Recall, that the NN was trained numerous times over the wide range of the pendulum angles to minimize the cost functional (19). The complexity of the approach is a function of the final time t_{final}, which determines the length of a training epoch. Solution suggested in this section is to apply a *receding horizon* framework and train the NN in a *limited* range of the pendulum angles, along a trajectory starting in the current point and having relatively short duration (horizon) N. This is accomplished by rewriting the cost function (19) as

$$H = \sum_{k=t}^{t+N-1} \left\{ L(\mathbf{x}_k, \mathbf{u}_k) + \lambda_k^T \left(\mathbf{x}_{k+1} - \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)\right)\right\} + V(\mathbf{x}_{t+N})$$

where the last term $V(\mathbf{x}_{t+N})$ denotes the *cost-to-go* from time $t+N$ to time t_{final}. Since range of the NN inputs in this case is limited and the horizon length N is short, a shorter time is required to train the NN. However, only local minimization of function (19) is achieved: starting with a significantly different initial condition the NN will not provide cost minimization since it was not trained for it. This issue is addressed by periodically retraining the NN: after a period of time called an *update interval*, the NN is retrained taking the current state vector as initial. Update interval is usually significantly shorter than the horizon, and in classical model-predictive control (MPC) is often only one time step. This technique was applied in helicopter control problem [15, 14] and is referred to as *Model Predictive Neural Control* (MPNC).

In practice, the true value of $V(\mathbf{x}_{t+N})$ is unknown, and must be approximated. Most common is to simply set $V(\mathbf{x}_{t+N}) = 0$; however, this may lead to reduced stability and poor performance for short horizon lengths [9]. Alternatively, we may include a control Lyapunov function (CLF), which guarantees stability if the CLF is an upper bound on the cost-to-go, and results in a region of attraction for the MPC of at least that of the CLF [9]. The cost-to-go $V(\mathbf{x}_{t+N})$ can be approximated using the solution of the SDRE at time $t+N$,

$$V(\mathbf{x}_{t+N}) \approx \mathbf{x}_{t+N}^T \mathbf{P}(\mathbf{x}_{t+N})\mathbf{x}_{t+N}$$

This CLF provides the exact cost-to-go for regulation assuming a linear system at the horizon time. A similar formulation was used for nonlinear regulation by Cloutier et al [13].

All the equations from the previous section apply here as well. The Euler-Lagrange equations (20) are initialized in this case as

$$\lambda_{t+N} = \left(\frac{\partial V(\mathbf{x}_{t+N})}{\partial \mathbf{x}_{t+N}}\right)^T \approx \mathbf{P}(\mathbf{x}_{t+N})\mathbf{x}_{t+N}$$

where dependence of the SDRE solution $\mathbf{P}$ on state vector $\mathbf{x}_{t+N}$ was neglected.

Stability of the MPNC is closely related to that of the traditional MPC. Ideally, in the case of unconstrained optimization, stability is guaranteed provided $V(\mathbf{x}_{t+N})$ is a CLF and is an (incremental) upper bound on the cost-to-go [9]. In this case, the minimum region of attraction of the receding horizon optimal control is determined by the CLF used and horizon length. The guaranteed region of operation contains that of the CLF controller and may be made as large as desired by increasing the optimization horizon (restricted to the infinite horizon domain) [10]. In our case, the minimum region of attraction of the receding horizon MPNC is determined by the SDRE solution used as the CLF to approximate the terminal cost. In addition, we also restrict the controls to be of the form given by (4.4) and the optimization is performed with respect to the NN weights $\mathbf{w}$. In theory, the *universal mapping* capability of NNs implies that the stability guarantees are equivalent to that of the traditional MPC framework. However, in practice stability is affected by the chosen size of the NN (which affects the actual mapping capabilities), as well as the horizon length N and update interval length (how often NN is re-optimized). When the horizon is short, performance is more affected by the chosen CLF. On the other hand, when the horizon is long, performance is limited by the NN properties. An additional factor affecting stability is the specific algorithm used for numeric optimization. Gradient descent, which we use to minimize the cost function (4.5), is guaranteed to converge to only a local minimum (the cost function is not guaranteed convex with respect to the NN weights), and thus depends on the initial conditions. In addition, convergence is assured only if the learning rate is kept sufficiently small. To summarize these points, stability of the MPNC is guaranteed under certain restricted ideal conditions. In practice, the designer must select appropriate settings and perform sufficient experiments to assure stability and performance.

4.6 Neural Network Adaptive Control

What if pendulum parameters are unknown? In this case the system Jacobians (22)–(23) can not be computed directly. A solution then is to use an additional NN, called "*emulator*", to emulate pendulum dynamics (see Figure (7)–(8)). This NN-emulator is trained to recreate pendulum outputs, given control input and current state vector. The regular error back-propagation (BP) algorithm is used if the NN-emulator is connected as in Figure (7), and the BPTT is employed in case depicted in Figure (8). The NN-regulator training procedure is the same as in subsection on the NN learning control, but instead of the system Jacobians (22)–(23), the NN-emulator Jacobians computed as (25) are used (see Figure (9)).

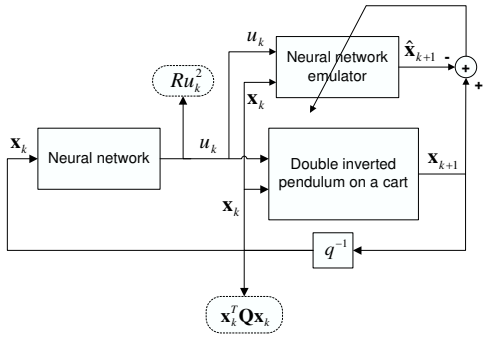

Figure 7: Neural network adaptive control diagram

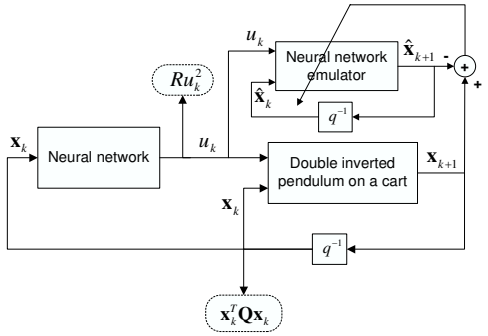

Figure 8: Neural network adaptive control diagram

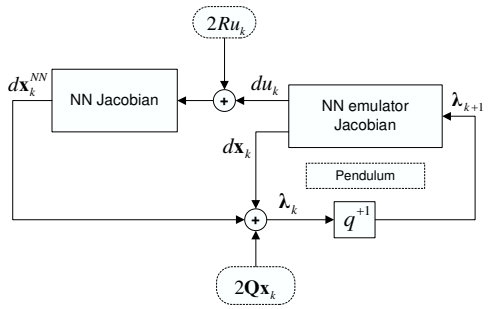

Figure 9: Adjoint NN adaptive system diagram

5 Simulation results

To evaluate the control performance, two sets of simulations were conducted. In the first set, the LQR, SDRE, NN and NN+LQR/SDRE control schemes were tested to stabilize the DIPC when both pendulums were initially deflected from their vertical position. System parameters used in the simulations are presented in Table 1.

Slight deflection (10 degrees in the opposite directions or 20 degrees in the same direction) results in a very similar performance of all control approaches as it was predicted (see Fig. 10–11).

Larger deflections (15 degrees in the opposite directions or 30 degrees in the same direction) reveal differences between the LQR and the SDRE: the latter one provides lesser cost (7)–(8) and keeps pendulum velocities lower (see Fig. 12–13 and Table 2). The NN-based controllers behave similar to the SDRE.

Note that the LQR could not recover the pendulums starting from 19 deg. initial deflection in the opposite directions or 36 deg. deflection in the same direction. Critical cases for the LQR (pendulums starting at 18 deg. deflections in the opposite directions from the upward position and 35 deg. in the same direction) are presented in Fig. 14–15. The NN control provides consistently better cost than the LQR at these larger deflec-

Table 1: Simulation parameters

Parameter	Value
m_0	1.5 kg
m_1	0.5 kg
m_2	0.75 kg
L_1	0.5 m
L_2	0.75 m
Δt	0.02 s
$\mathbf{Q}$	diag(5 50 50 20 700 700)
$\mathbf{R}$	1
N_h	40
t_{final}	500

Table 2: Control performance (cost) comparison

Control	Deflection of pendulums, deg. Opposite direction			
	10	15	18	28
LQR	115.2	328.4	705.0	n/r
SDRE	112.9	277.3	437.5	3655.8
NN	114.1	323.4	n/r	n/r
NN+LQR	113.3	275.7	448.3	n/r
NN+SDRE	112.5	276.3	436.6	2753.7
	Same direction			
	20	30	35	67
LQR	36.7	108.3	325.3	n/r
SDRE	36.0	84.0	118.0	5250.5
NN	36.9	85.7	144.8	n/r
NN+LQR	37.3	86.2	136.4	n/r
NN+SDRE	36.0	84.0	118.0	n/r

Table 3: Regions of pendulums recovery

Control	Recovery region, deg.	
	Max. opposite dir.	Max. same dir.
LQR	18	35
SDRE	28	67
NN	15	38
NN+LQR	21	40
NN+SDRE	28	62

tions, but it can't stabilize the system beyond 15 (38) deg. pendulum deflections in the opposite (same) directions due to limited approximation capabilities of the NN.

In the second set of experiments, regions of pendulum recovery (*i.e.* maximum initial pendulum deflections from which a control law can bring the DIPC back to the equilibrium) were evaluated. The results are shown in Table 3. Cases of critical initial pendulums positions for SDRE (28 deg. deflections the in opposite directions or 67 deg. in the same direction) are presented for the reference in Fig. 16–17. Note that no limits were imposed on the control force magnitude in all simulations. The purpose was to compare LQR and SDRE without introducing model changes which are not directly accounted for in control design. It should be mentioned however, that limited control authority and ways of taking it into account in the SDRE design were investigated earlier [3] but are left currently beyond our scope.

6 Conclusions

This report demonstrated potential advantage of the SDRE technique over the LQR design in nonlinear optimal control problems with an underactuated plant. Region of pendulum recovery for SDRE appeared to be 55 to 91 percent larger than in case of the LQR control.

Direct optimization via using neural networks yields results superior to the LQR, but the recovery region is about the same as in the LQR case. This happens due to limited approximation capabilities of the NN and nonconvex numerical optimization challenges.

Combination of the NN control with the LQR (or with the SDRE) provides larger recovery regions and better overall performance. In this case the NN learns to generate corrections to the LQR (SDRE) control to compensate for suboptimality of the LQR (SDRE).

To enhance this report, it would be valuable to investigate taking limited control authority into account in all control designs.

References

[1] R. W. Brockett and H. Li. A light weight rotary double pendulum: maximizing the domain of attraction. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, Maui, Hawaii, December 2003.

[2] A. Calise and R. Rysdyk. Nonlinear adaptive flight control using neural networks. *IEEE Control Systems Magazine*, 18(6), December 1998.

[3] J. R. Cloutier, C. N. D'Souza, and C. P. Mracek. Nonlinear regulation and nonlinear H-infinity control via the state-dependent Riccati equation technique: Part1, theory. In *Proceedings of the International Conference on Nonlinear Problems in Aviation and Aerospace*, Daytona Beach, FL, May 1996.

[4] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4), 1989.

[5] G. F. Franklin, J. D. Powell, and A. Emami-Naeini. *Feedback control of dynamic systems.* Addison-Wesley, 2 edition, 1991.

[6] K. Furuta, T. Okutani, and H. Sone. Computer control of a double inverted pendulum. *Computer and Electrical Engineering*, 5:67–84, 1978.

[7] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward neural networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[8] R. A. Jacobs. Increasing rates of convergence through learning rate adaptation. *Neural Networks*, 1(4):295–307, 1988.

[9] A. Jadbabaie, J. Yu, and J. Hauser. Stabilizing receding horizon control of nonlinear systems: a control Lyapunov function approach. In *Proceedings of American Control Conference*, 1999.

[10] A. Jadbabaie, J. Yu, and J. Hauser. Unconstrained receding horizon control of nonlinear systems. In *Proceedings of IEEE Conference on Decision and Control*, 1999.

[11] E. Johnson, A. Calise, R. Rysdyk, and H. El-Shirbiny. Feedback linearization with neural network augmentation applied to x-33 attitude control. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, August 2000.

[12] W. T. Miller, R. S. Sutton, and P. J. Werbos. *Neural networks for control*. MIT Press, Cambridge, MA, 1990.

[13] M. Sznaizer, J. Cloutier, R. Hull, D. Jacques, and C. Mracek. Receding horizon control Lyapunov function approach to suboptimal regulation of nonlinear systems. *Journal of Guidance, Control and Dynamics*, 23(3):399–405, May-June 2000.

[14] E. Wan, A. Bogdanov, R. Kieburtz, A. Baptista, M. Carlsson, Y. Zhang, and M. Zulauf. Model predictive neural control for aggressive helicopter maneuvers. In T. Samad and G. Balas, editors, *Software Enabled Control: Information Technologies for Dynamical Systems*, chapter 10, pages 175–200. IEEE Press, John Wiley & Sons, 2003.

[15] E. A. Wan and A. A. Bogdanov. Model predictive neural control with applications to a 6 DOF helicopter model. In *Proceedings of IEEE American Control Conference*, Arlington, VA, June 2001.

[16] P. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of IEEE, special issue on neural networks*, 2:1550–1560, October 1990.

[17] W. Zhong and H. Rock. Energy and passivity based control of the double inverted pendulum on a cart. In *Proceedings of the IEEE international conference on control applications*, Mexico City, Mexico, September 2001.

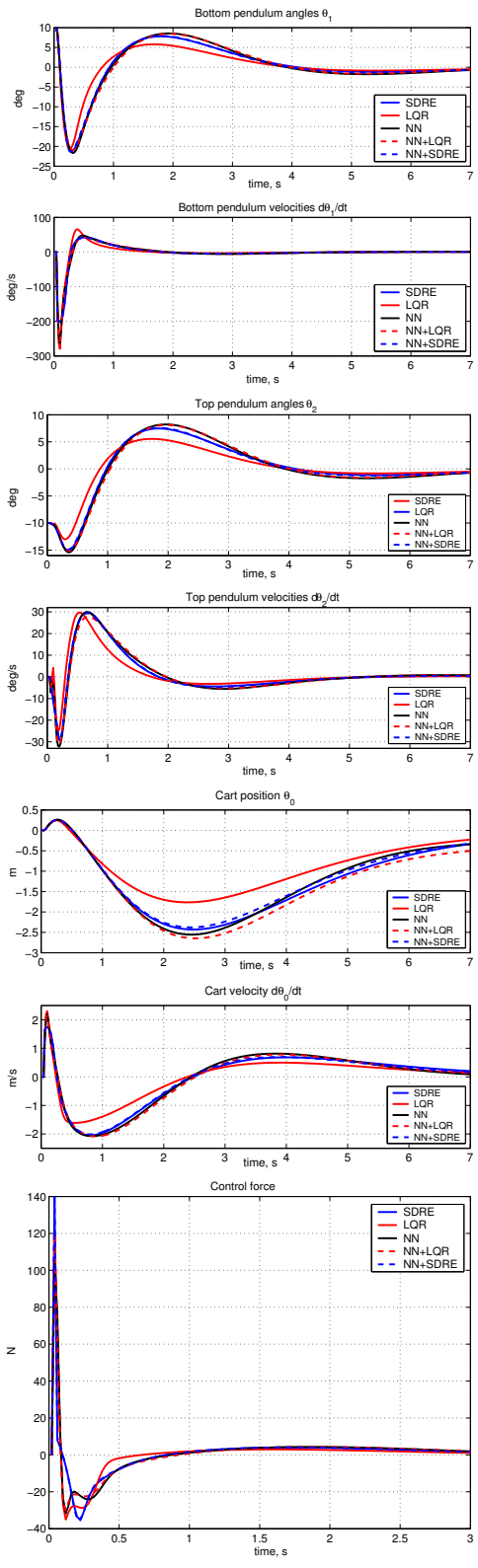

Figure 10: 10 deg. opposite direction

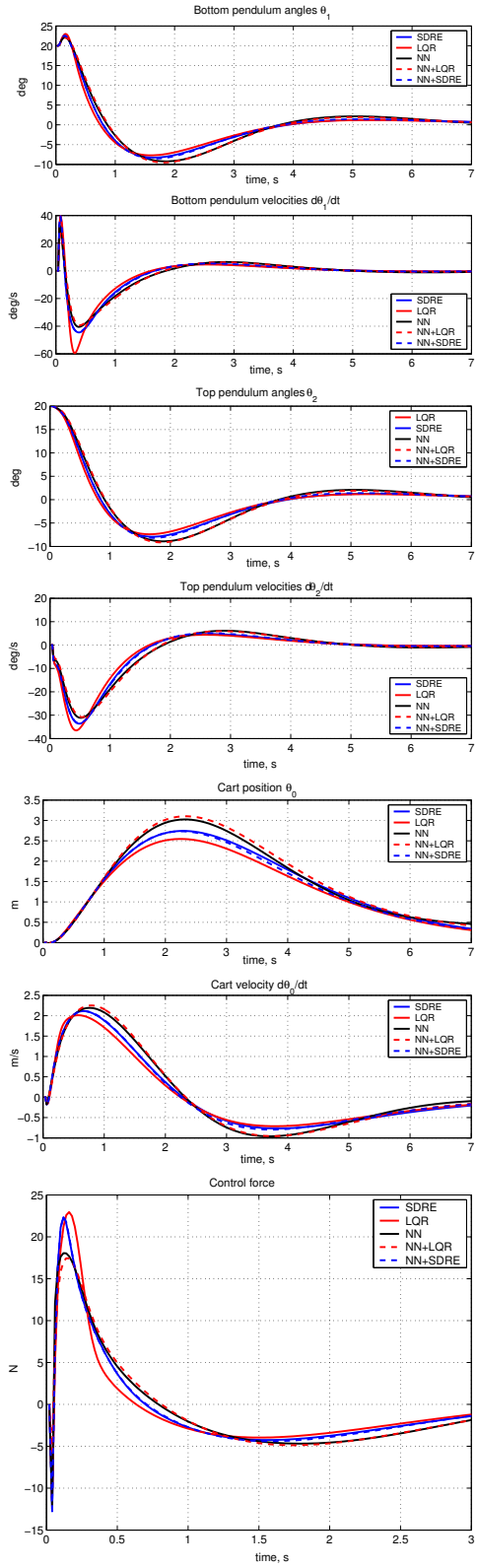

Figure 11: 20 deg. same direction

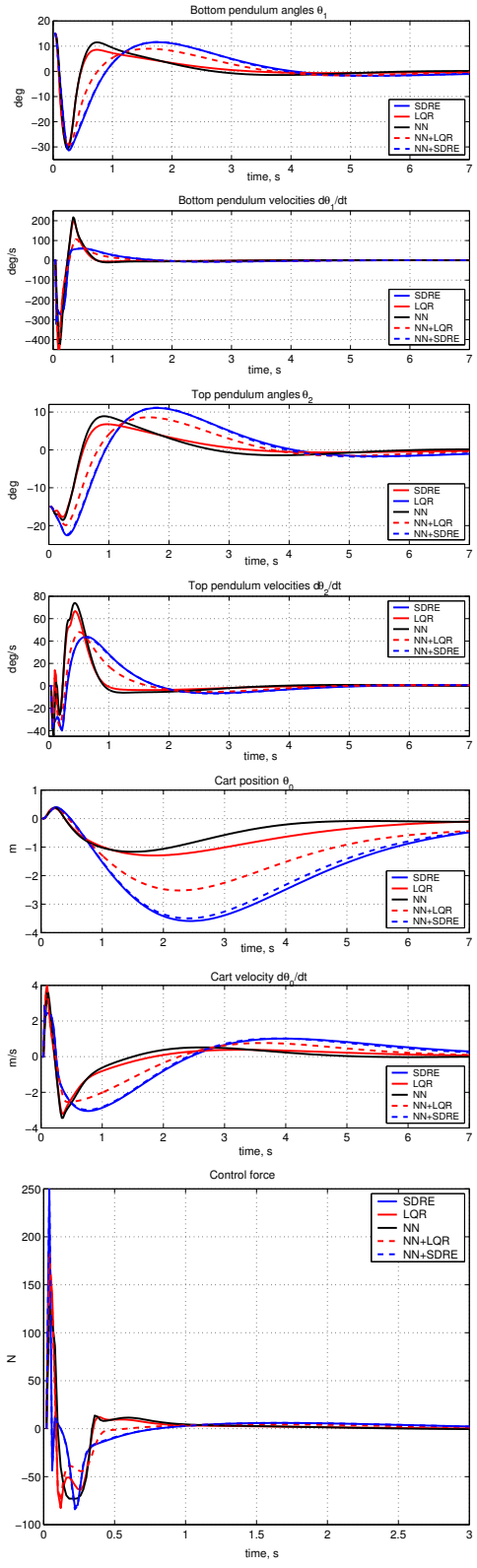

Figure 12: 15 deg. opposite direction

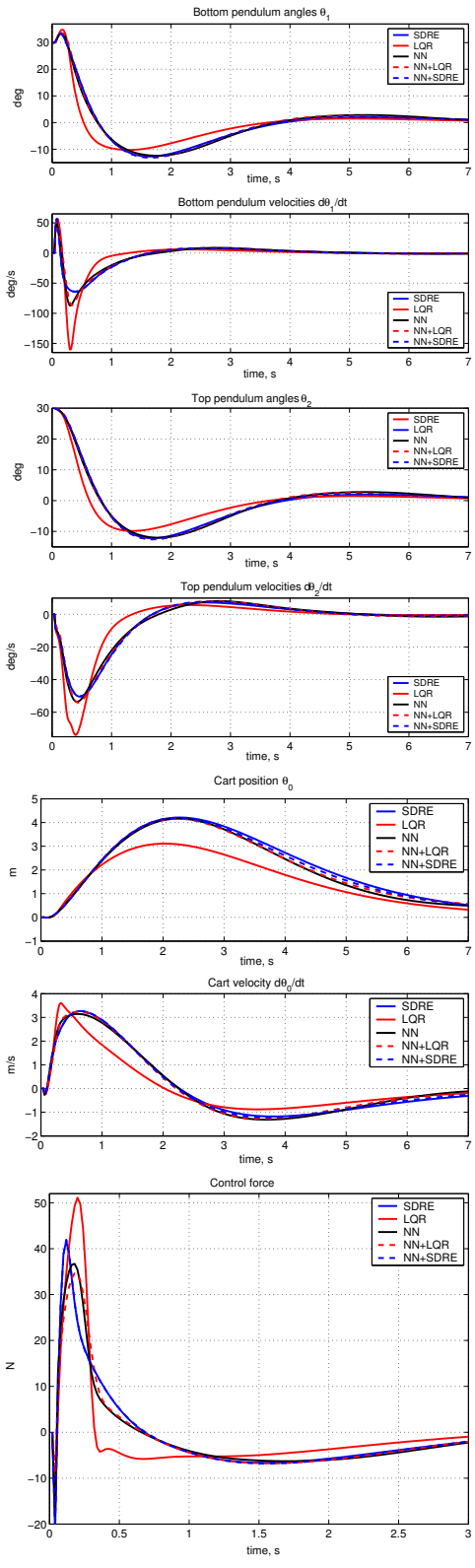

Figure 13: 30 deg. same direction

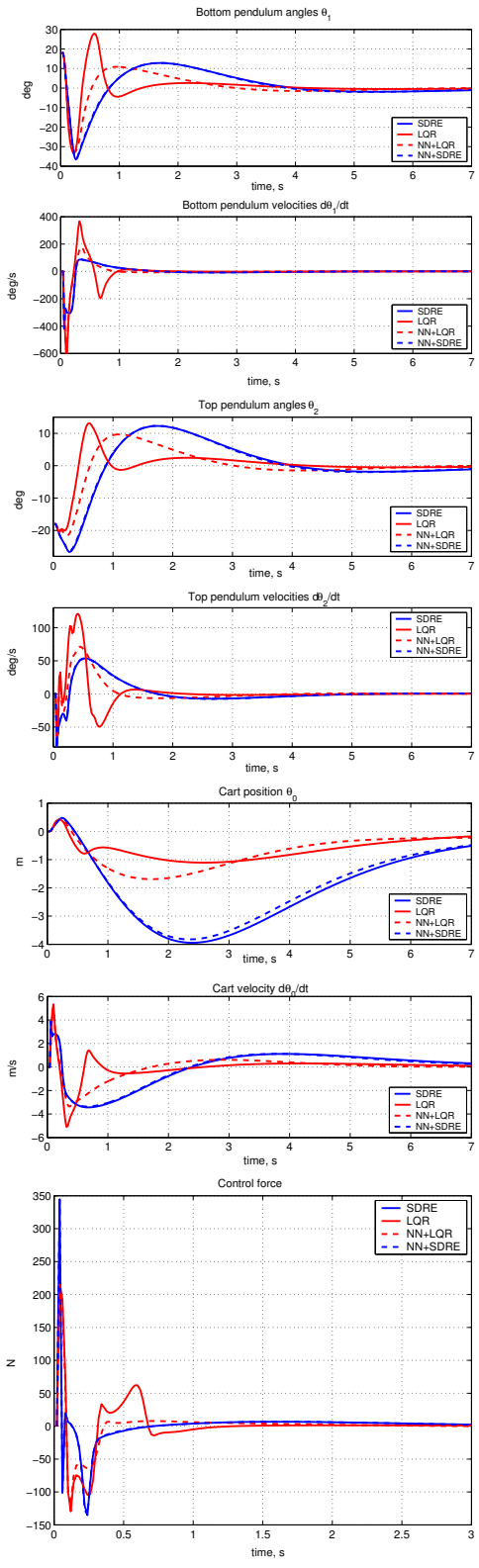

Figure 14: 18 deg. opposite direction

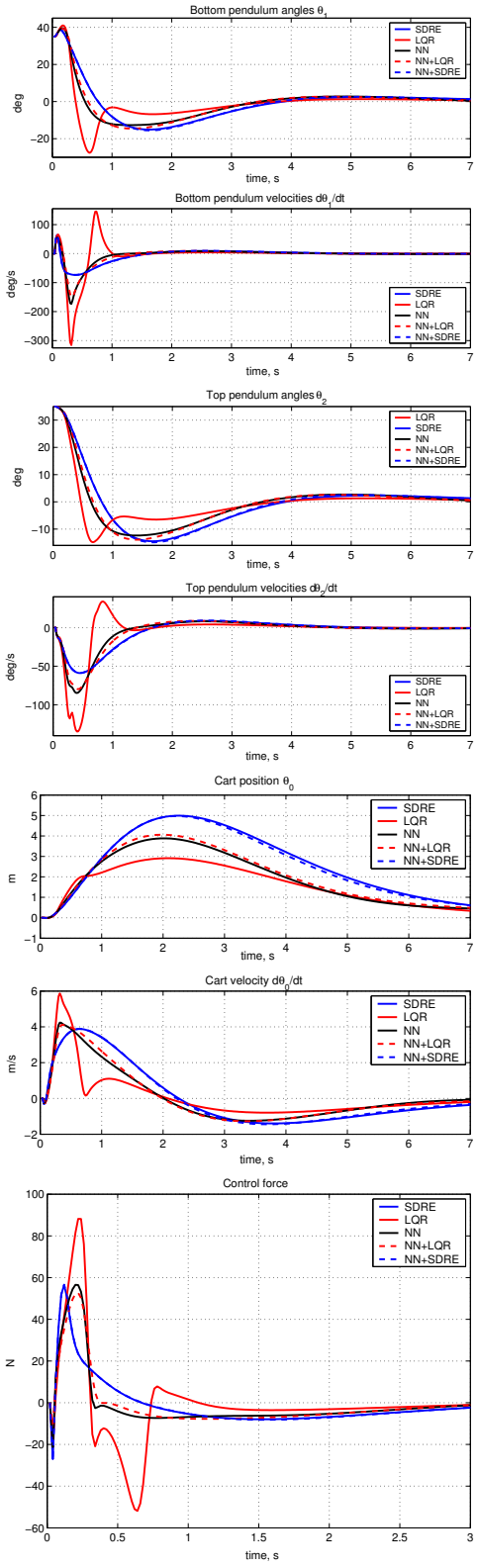

Figure 15: 35 deg. same direction

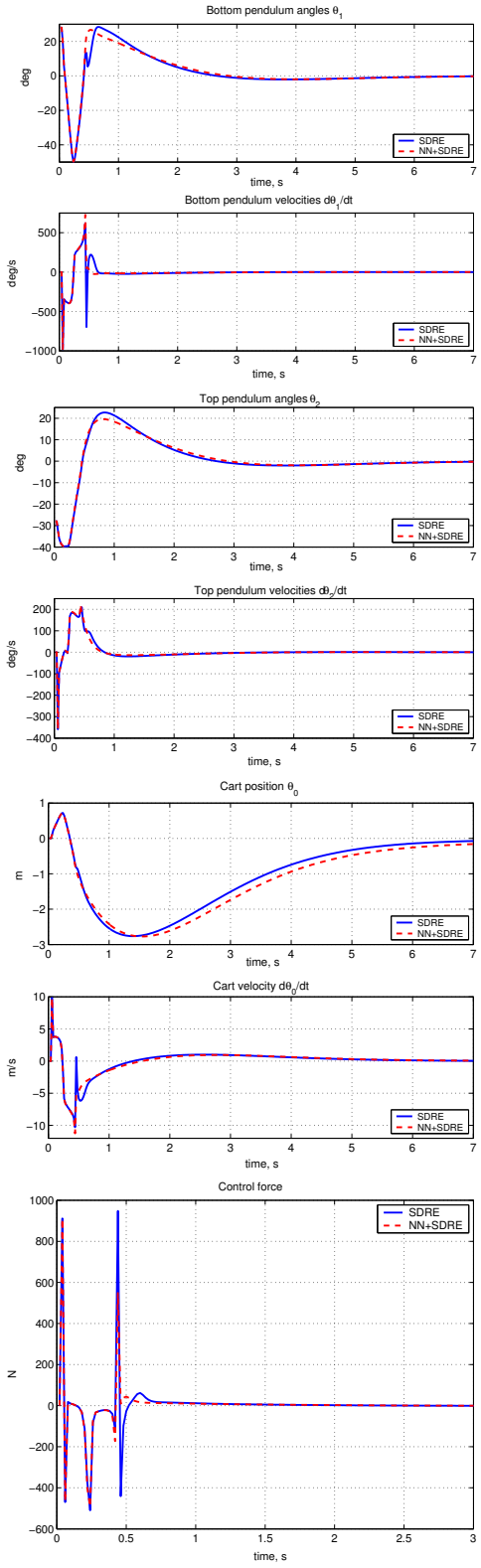

Figure 16: 28 deg. opposite direction, SDRE vs. NN+SDRE

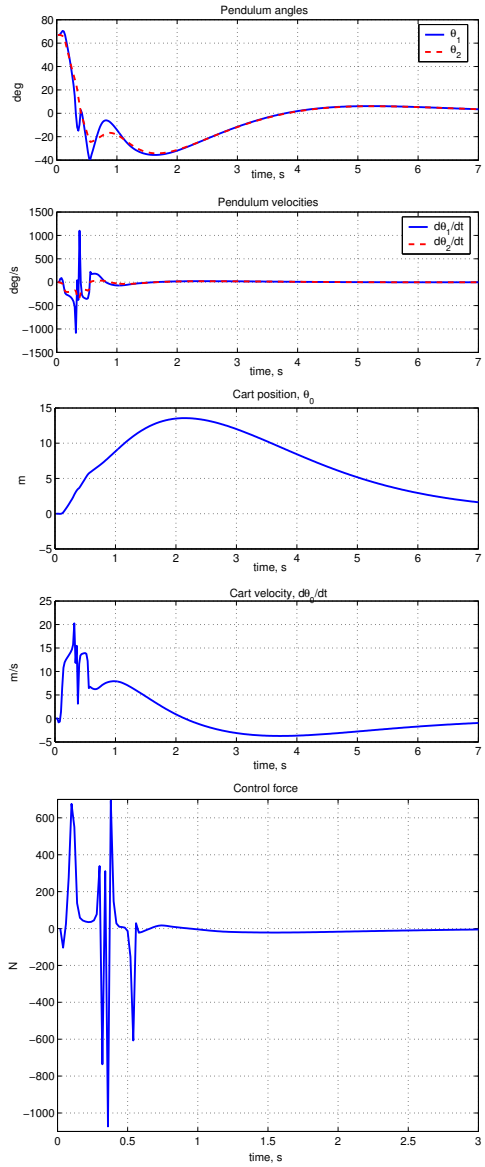

Figure 17: 67 deg. same direction, SDRE