Direct Inverse Control
&
Internal Model Control

Modelling and Control of Dynamic Systems
17 Nov 2008

Jürgen Jänes and Andres Tiko

Talk Outline

* Introduction to Control
* Direct Inverse Control
  * General Training
  * Specialized Training

---

  * Demo
* Internal Model Control

Introduction

* Book so far: System identification
  * "How does the system behave?"
* Book from now on: System control
  * "How do I make the system do what I want?"

Taxonomy of Control Problems

* Regulation problem: keep the output of the system at a constant level
  * e.g. room temperature, inverted pendulum
* Servo problem: make the output follow a desired trajectory
  * e.g. brick doing 8-shapes on a tiltable plane

Stability

* in practice, nonlinear systems need to be stable for successful control
* altough transfer functions don't apply for nonlinear systems, "transfer function zeroes" are extended to nonlinear systems
  * details: Section 3.7 (Nov 24th)
* on-line training/adaptive control: approach with a grain of NaCl

Control system architectures

* direct control system: neural network is the controller
  + simple implementation
  - parameter change -> retrain network
* indirect control system: use a neural network as a (cheap!) system model
  + faster controller "tuning cycle"
  - more complicated to implement

Benchmark system

* "spring-mass-damper system with a hardening spring"

$$\ddot{y}(t) + \dot{y}(t) + y(t) + y^3(t) = u(t)$$

* open-loop stable
* inverse is unstable

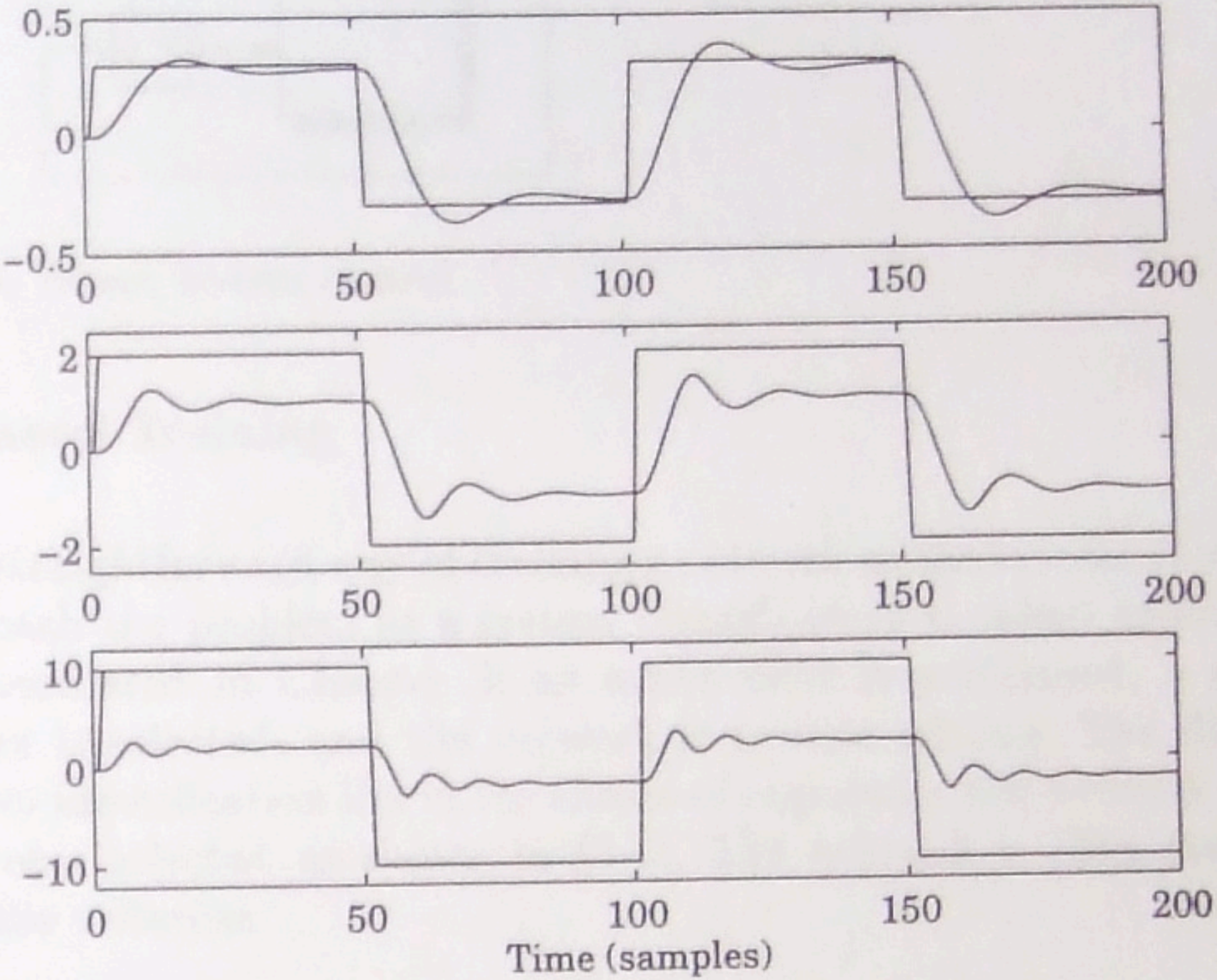


Figure 3.1. Open-loop simulation of benchmark system with three different square waves applied as input.

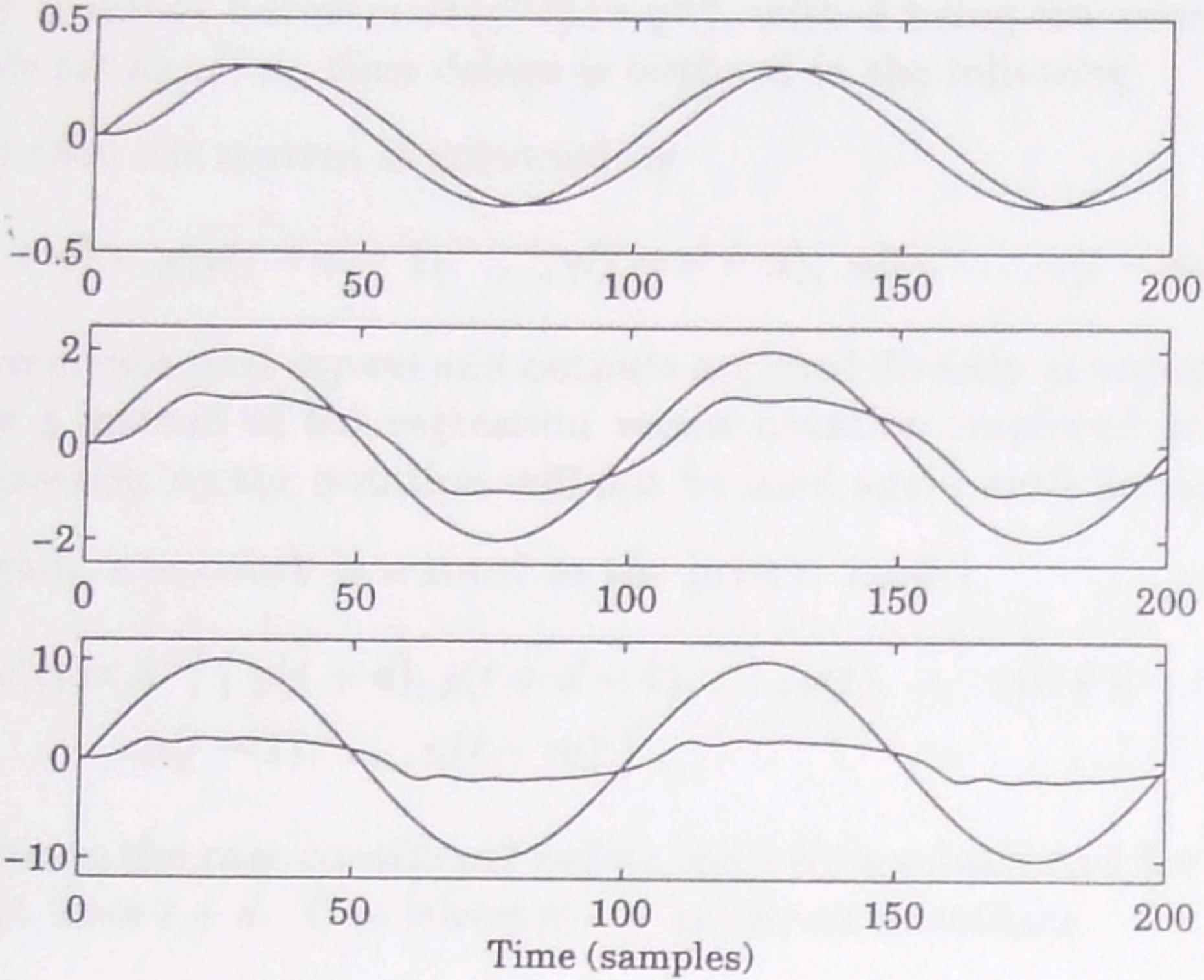


Figure 3.2. Open-loop simulation of benchmark system with three different sinusoidals applied as input.

Direct Inverse Control (DIC)

DIC: Basic Idea

* Train a neural network as the inverse of a system, use this as the controller

* System is described by

$$y(t+1) = g[y(t), \ \ldots, y(t-n+1), \ u(t), \ \ldots, u(t-m)]$$

* We make the neural network learn

$$\hat{u}(t) = \hat{g}^{-1}[y(t+1), y(t), \ \ldots, y(t-n+1) \ , u(t), \ \ldots, u(t-m)]$$

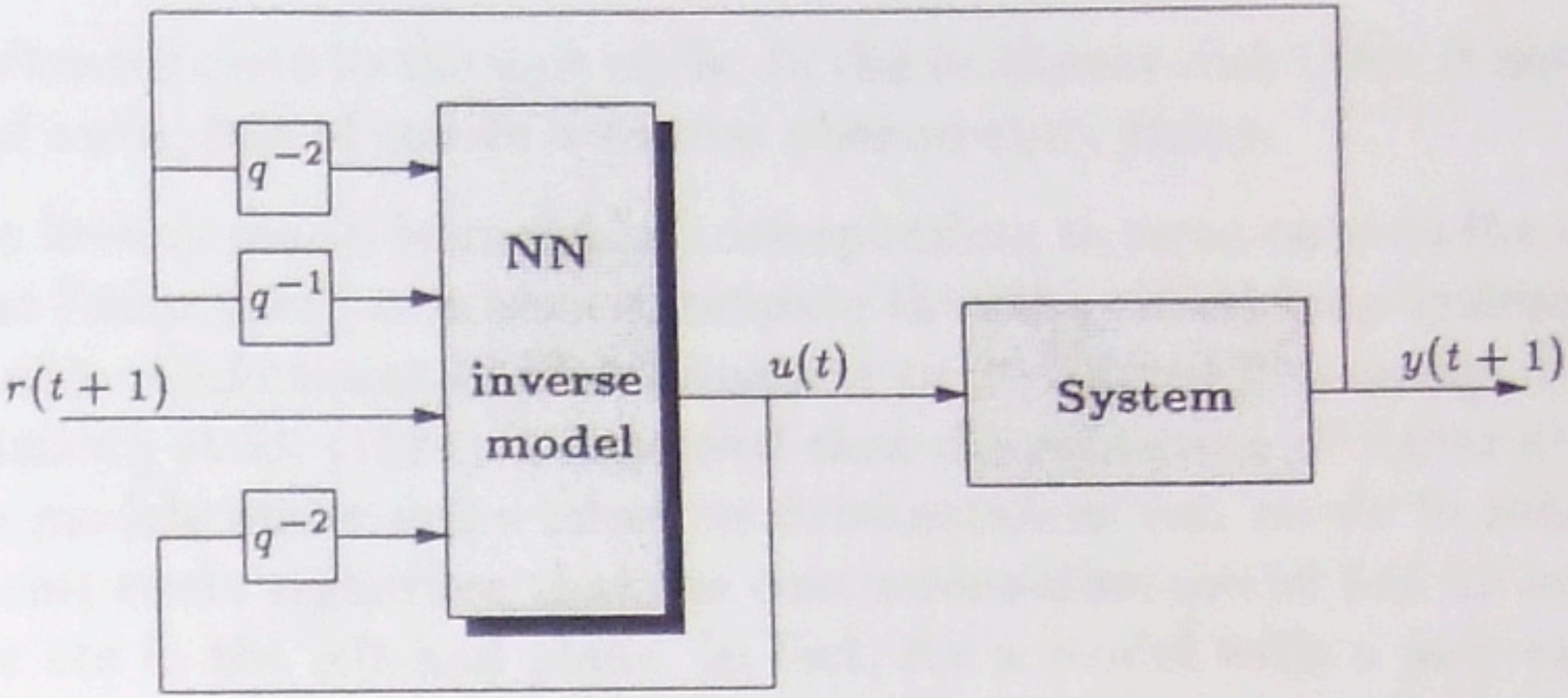


Figure 3.3. Direct inverse control.

DIC: time delays

* Assuming the model is governed by

$$y(t+d) = g[y(t+d-1),\ \ldots,y(t+d-n),\ u(t),\ \ldots,u(t-m)]$$

* We would like to train the network to learn

$$\hat{u}(t) = \hat{g}^{-1}\,[\,y(t+d), y(t+d-1),\ \ldots,y(t),\ \ldots,y(t+d-n)$$
$$u(t-1),\ \ldots,u(t-m)\,]\ .$$

* But we don't know $\{y(t+1),\ldots,y(t+d-1)\}$

DIC: time delays (2)

* Solution #1: this is a system identi-fication task!
* Solution #2: "incorporate" inverse model directly (assume d=2 -> y(t+1) missing)
  * Assuming y(t+1) can be precicted with

$$y(t+1) \simeq \hat{y}(t+1) = \hat{g}_1[y(t), \ \ldots, y(t+1-n), \\ u(t-1), \ \ldots, u(t-m-1)]$$

  * Train the network with (=add more past data)

$$\hat{u}(t) = \hat{g}^{-1}[y(t+2), y(t), \ \ldots, y(t+2-n), \ \ldots, y(t+1-n) \ , \\ u(t-1), \ \ldots, u(t-m), \ u(t-m-1)] \ .$$

General Training

* Train the network using "brute force", i.e. minimize:

$$J(\theta, Z^N) = \frac{1}{2N} \sum_{t=1}^{N} [u(t) - \hat{u}(t|\theta)]^2$$

* can directly use any method from Section 2.4

Practical Considerations

* essentially, DIC produces dead-beat controllers
* -> feedback is used to achieve fast response time
* poor robustness, high sensitivity to noise and high frequency disturbances

Practical considerations 2

* it has been shown that a discretization of a continuous linear system can have zeros near the unit circle (=is unstable), regardles of how the zeros in the continuous system are placed
* similar behaviour expected in the nonlinear case

Practical considerations 3

* System is not one-to-one

$$g[y(t), \ \ldots y(t-n+1), u_1(t), \ \ldots, u(t-m)] =$$
$$g[y(t), \ \ldots y(t-n+1), u_2(t), \ \ldots u(t-m)]$$

* Might work if non-uniqueness is not reflected in the training set

Practical considerations 4

* identification for control: training data should be similar to testing data
  * but we don't know how the system responds to the NN controller
* solution: iterative training

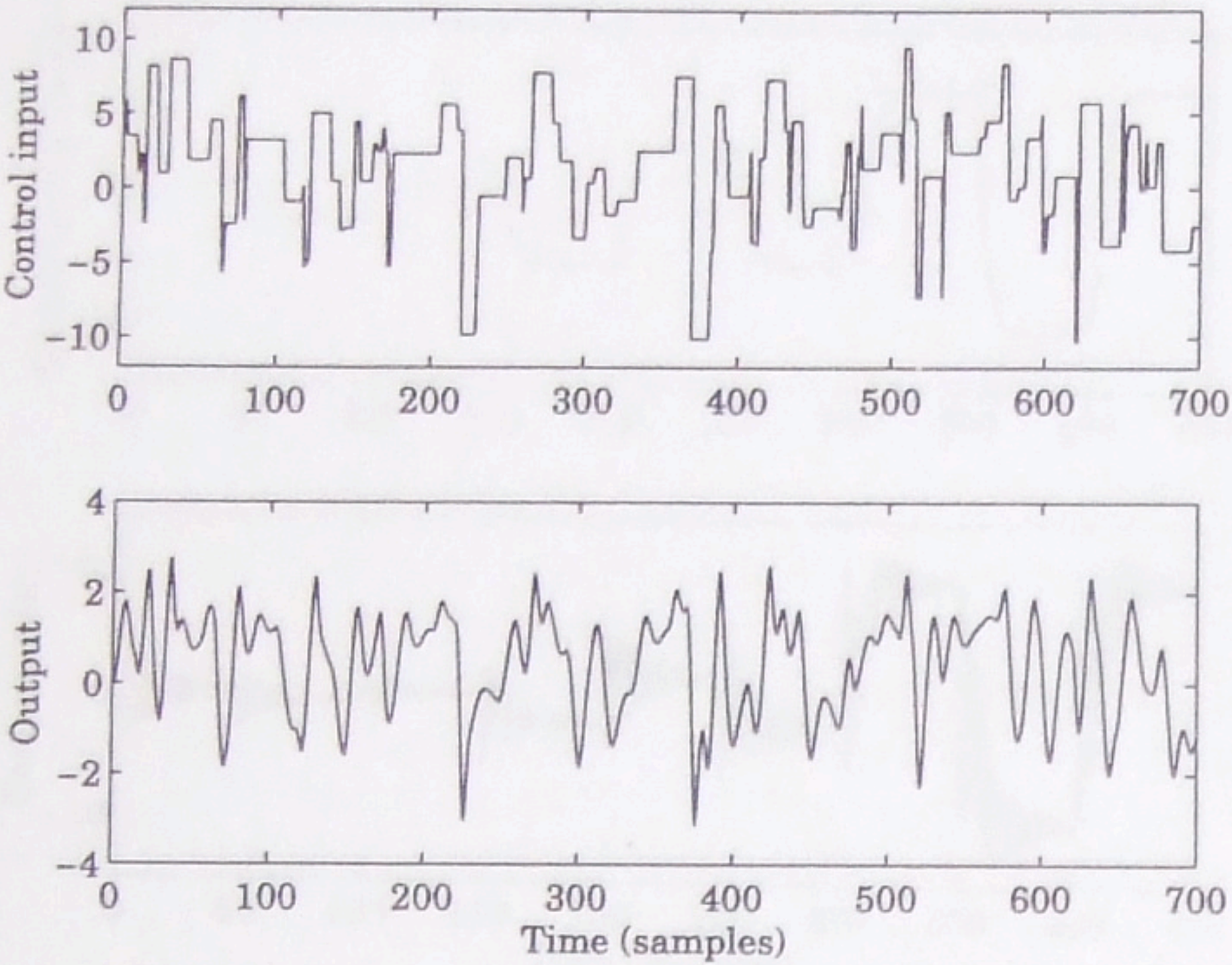


Figure 3.4. Training data set. Upper panel: control signal. Lower panel: output signal.

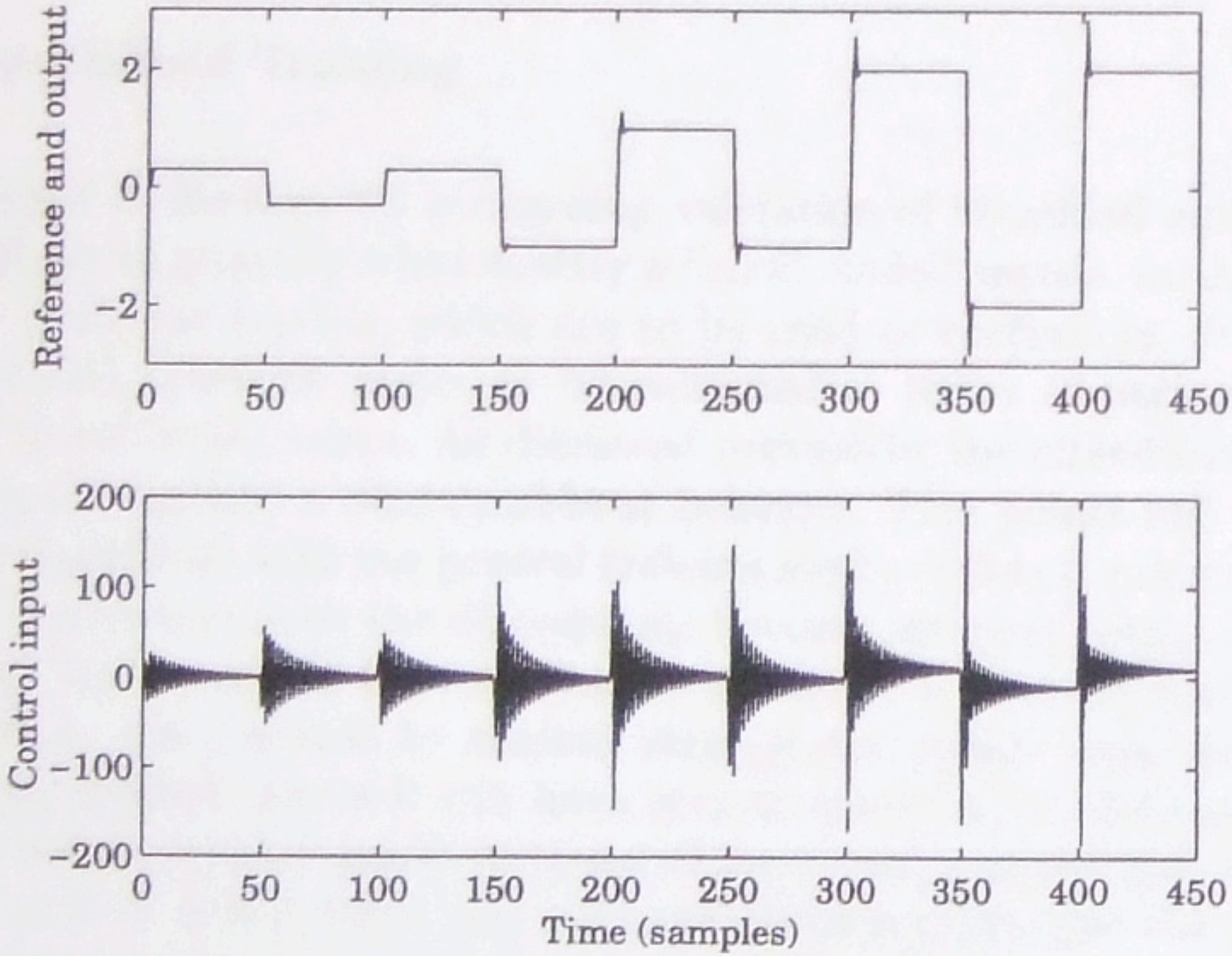


Figure 3.5. Direct inverse control of the benchmark system. Upper panel: reference and output signal. Lower panel: control signal.

Benchmarking

* Linearized+discretized system has a zero at z=-0.9354 (near the unit circle)
* -> large+oscillating control signal
* Solution: low-pass filtering of reference

$$H_m(q^{-1}) = \frac{0.09}{1 - 1.4q^{-1} + 0.49q^{-2}}$$

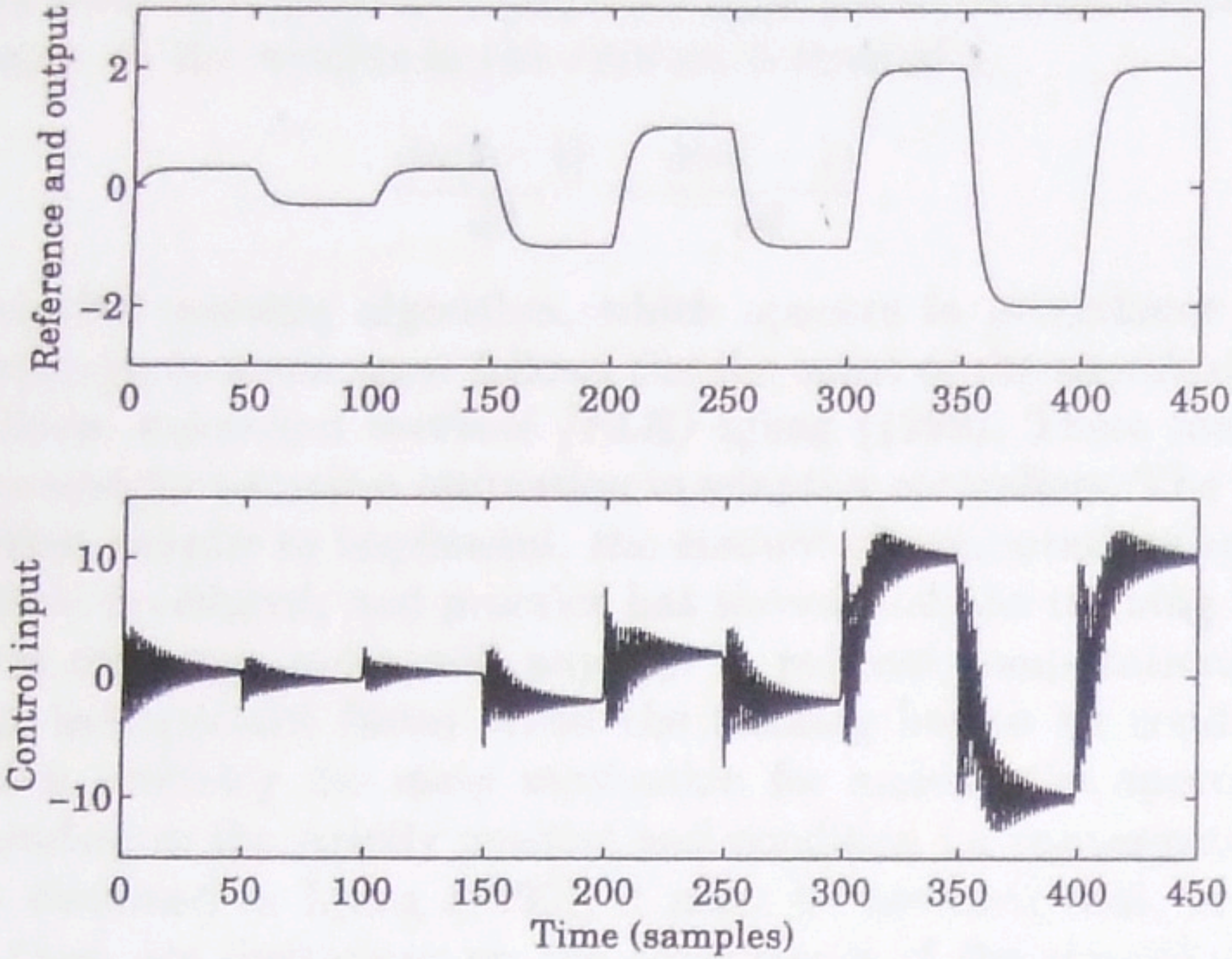


Figure 3.6. Direct inverse control after a low-pass filtering of the reference trajectory. Upper panel: reference and output signal. Lower panel: control signal.

Specialized Training

* General Training: minimises error between network output and "true" control input

$$J(\theta, Z^N) = \frac{1}{2N} \sum_{t=1}^{N} [u(t) - \hat{u}(t|\theta)]^2$$

* We would like to minimise error between system output and reference signal

$$J(\theta, Z^N) = \frac{1}{2N} \sum_{t=1}^{N} [r(t) - y(t)]^2$$

"Deriving a training scheme based on this criterion is not completely straightforward. A few approximations are required to make implementation possible."

Our  for this Proof

recall from Analysis - The Chain Rule:

$$\frac{df(x_1,\ldots,x_n)}{dt} = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}\frac{dx_i}{dt}$$

Given that $x_1=x_1(t)$, ..., $x_n=x_n(t)$.

Implementation

* specialized training can be implemented by slightly modifying algorithms from Section 2.4
* details in book

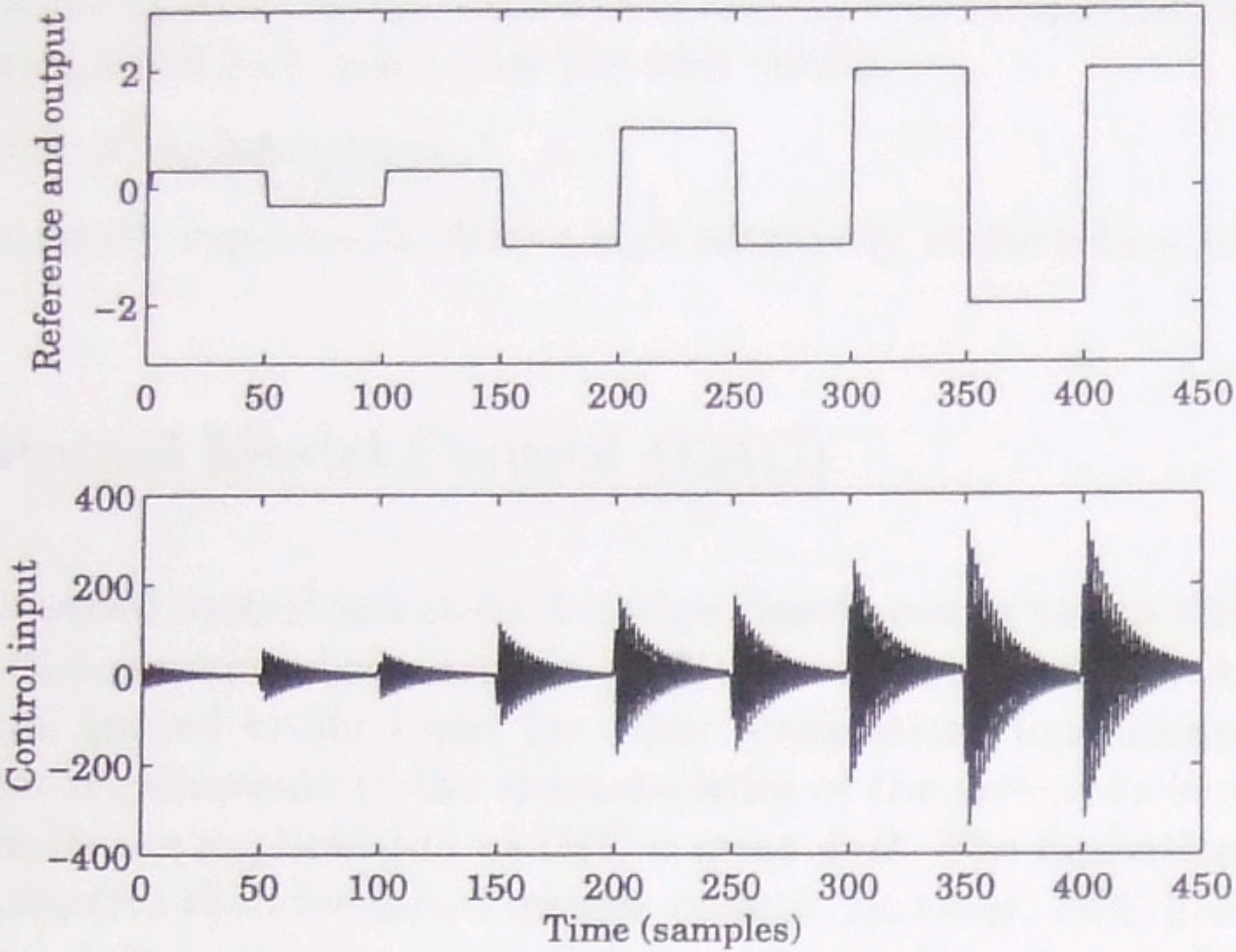


Figure 3.8. Direct inverse control with an inverse model obtained with specialized training. Upper panel: reference and output signal. Lower panel: control signal.

Summary

* Generalized training. Off-line, minimize RMS between experimentally determined control signal and predicted control signal
* Spezialized training. On-line, minimize RMS between reference signal and system output

DIC recommended approch

1. Generate data set from experiment
2. Generate forward model
3. Initialize controller with general training
4. Specialized training on system model (off-line)
5. Specialized training on real system (on-line)
6. (Profit!)

DIC: the good

+ intuitive & simple to implement
+ controller can be optimized for specific trajectory with specialized training
+ in theory, should work on time-varying systems

DIC: the bad

- does not work for systems with unstable inverse
- problems when system is not one-to-one
- problems with inverse models not well-damped
- lack of tuning options (parameter change -> retrain network)
- high sensitivity to disturbance & noise

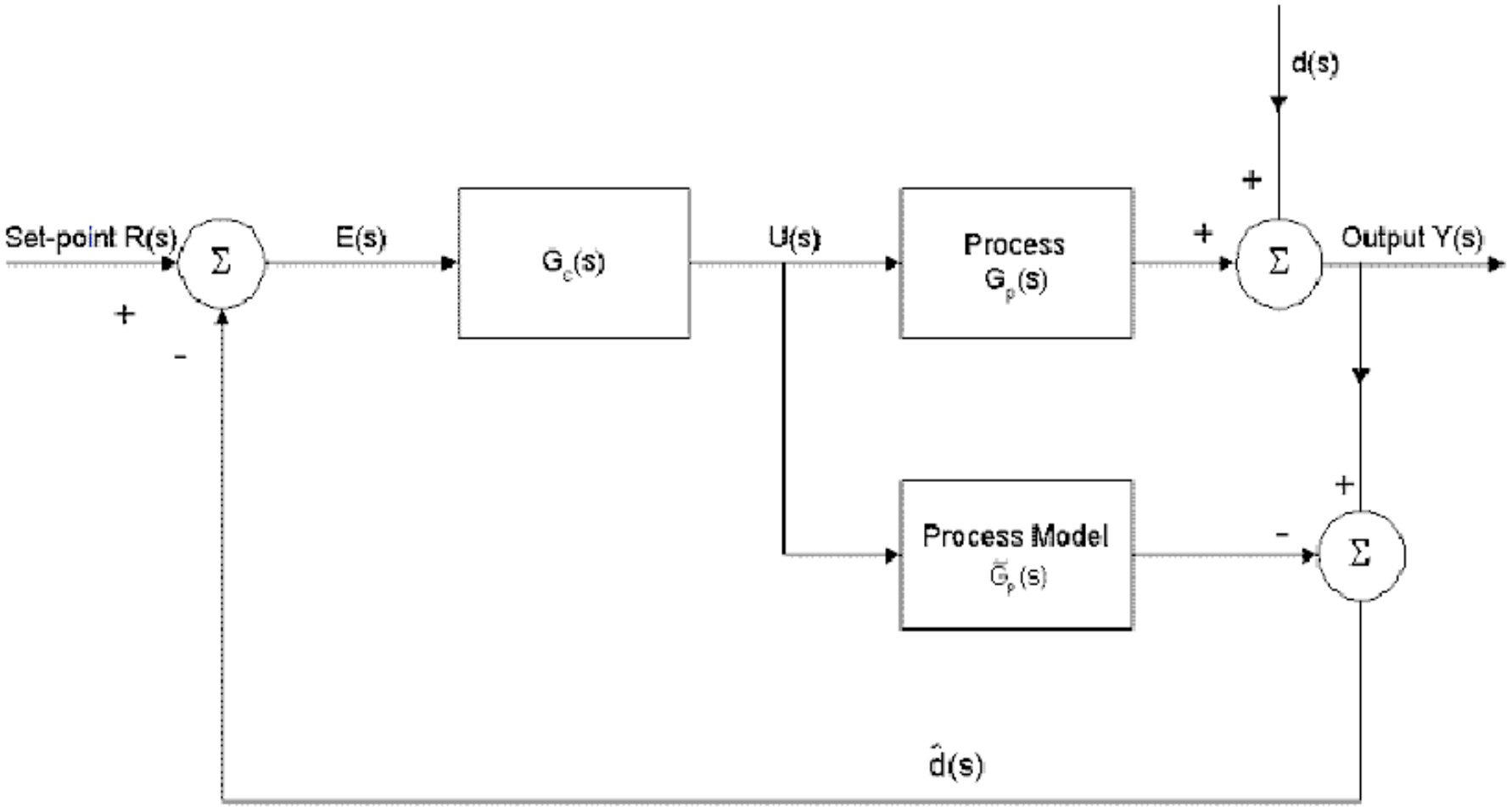
End of Part 1.

Internal Model Control (IMC)

- A design closely connected to direct inverse control
- Restrictive requirements to the characteristics of the system
- Some nice features – compensation for constant disturbances

IMC

- Requires a forward model as well as an inverse model of the system
- Feedback consists of the error between system output and model output – zero for a perfect model


d(s)
+
Set-point R(s)
Σ
E(s)
$G_c(s)$
U(s)
Process
$G_p(s)$
+
Σ
Output Y(s)
+
-
+
Process Model
$\tilde{G}_p(s)$
-
Σ
$\hat{d}(s)$

IMC

$$y(t) = v(t) + \frac{q^{-d} FCP}{1 + q^{-d} FC(P - M)} [r(t) - v(t)]$$

$$u(t) = \frac{FC}{1 + q^{-d} FC(P - M)} [r(t) - v(t)]$$

Stability

- For global stability of the closed loop system the system to be controlled and the inverse model should both be stable
- Requirement of open-loop stability severely limits the class of systems that can be controlled with IMC

The Ideal World

- Under idealized conditions: M = P and C = P^{-1}

$$u(t) = \frac{F}{P}[r(t) - v(t)]$$

$$y(t) = q^{-d} F\, r(t) + (1 - q^{-d} F)\, v(t)$$

Demands to the Filter

- F is the only design parameter, therefore it is difficult to impose constraints on the control signal
- Must be stable and have unity steady-state gain to ensure tracking of the reference
- For disturbance rejection ability, it should hold that $(1-q^{-d}F)\, v(t) = 0$
- F = 1 is a good choice

IMC is Special...

- Offset-free response for systems affected by constant disturbance
- Requirement that the system is open-loop stable
- Difficult to ensure that the inverse model is trained on a realistic data set