

# **NEW DEVELOPMENTS IN PREDICTIVE CONTROL FOR NONLINEAR SYSTEMS**

M. J. Grimble, A. Ordys, A. Dutka, P. Majecki

University of Strathclyde

Glasgow Scotland, U.K

# Introduction

- Model Predictive Control (MPC) is one of the most popular advanced control techniques
- The MPC algorithms are well established for linear systems
- Recent developments extended this methodology to the Non-linear systems control
- Techniques developed at the University of Strathclyde are presented

Linear Quadratic Gaussian Predictive  
Control  
LQGPC

# Preview Control

The algorithm was first presented by Tomizuka M. and D.E. Whitney. The algorithm uses the LQG approach to optimisation and a stochastic model for the reference signal beyond the preview horizon.

State space model:

$$\begin{aligned} \mathbf{x}(t+1) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{G}\mathbf{w}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{v}(t) \end{aligned}$$

Performance index:

$$J(t) = \mathbb{E} \left\{ \sum_{j=0}^M \left[ (\mathbf{y}(t+j+1) - \mathbf{r}(t+j+1))^T (\mathbf{y}(t+j+1) - \mathbf{r}(t+j+1)) + \lambda \cdot \mathbf{u}(t+j)^T \mathbf{u}(t+j) \right] \right\}$$

Reference generator:

$$\mathbf{R}_N(t+1) = \Theta_N \mathbf{R}_N(t) + \eta_N \xi(t+N)$$

# Preview Control

$$\begin{bmatrix} x(t+1) \\ x_{R,N}(t+1) \end{bmatrix} = \chi(t+1) = \underbrace{\begin{bmatrix} A & O \\ O & \Theta_N \end{bmatrix}}_A \chi(t) + \underbrace{\begin{bmatrix} B \\ O \end{bmatrix}}_B u(t) + \underbrace{\begin{bmatrix} G & O \\ O & \mu_N \end{bmatrix}}_G \begin{bmatrix} w(t) \\ \xi(t+N) \end{bmatrix}$$
$$\begin{bmatrix} y(t) \\ R_N(t) \end{bmatrix} = \underbrace{\begin{bmatrix} C & O \\ O & I \end{bmatrix}}_C \begin{bmatrix} x(t) \\ x_{R,N}(t) \end{bmatrix} + \begin{bmatrix} I \\ O \end{bmatrix} v(t)$$

**A standard (infinite horizon) LQG approach can now be used.**

# Linear Quadratic Gaussian Predictive Control LQGPC

The standard predictive control performance index:

$$J_t = (Y(t) - R(t))^T \Lambda_e (Y(t) - R(t)) + (U(t))^T \Lambda_u (U(t))$$

Starting with the state-space model:

$$\begin{aligned}x(t+1) &= Ax(t) + Bu(t) + Gw(t) \\ y(t) &= Dx(t) + v(t)\end{aligned}$$

re-formulate it to be compatible with the above index:

state equation:

$$x(t+1) = Ax(t) + \beta U(t) + \Gamma W(t)$$

output equation:

$$Y(t) = \Phi_N A x(t) + S_N U(t) + \tilde{S}_N W(t) + V(t)$$

The control vector  $U(t)$  contains all control actions within the horizon  $N$  into the future

# Linear Quadratic Gaussian Predictive Control

## LQGPC

Define the LQG-type performance index (finite or infinite) as a sum of “predictive” performance indices:

$$J_{DPC} = E \left\{ \frac{1}{T+1} \sum_{l=t}^{t+T} J_{GPC}(l) \right\}$$

$$J_{DPC} = E \left\{ \lim_{T \rightarrow \infty} \frac{1}{T+1} \sum_{l=t}^{t+T} J_{GPC}(l) \right\}$$

Substituting the criterion from the previous slide obtains:

$$J_t = E \left\{ \frac{1}{T+1} \sum_{l=t}^T \left[ (Y(l) - R(l))^T \Lambda_e (Y(l) - R(l)) + (U(l))^T \Lambda_u (U(l)) \right] \right\}$$

The solution can be obtained through Dynamic Programming with two Riccati equations involved.

The “reference generator” is used in a similar way as in Preview Control

**Solution of Quadratic Gaussian  
problem for nonlinear systems  
and  
Non-Linear Quadratic Gaussian  
Predictive Control**

# NLQG and NLQGPC

- NLQG - an extension of SDRE method
- SDRE method – State Dependent Riccati Equations
- The NLQGPC algorithm: predictive extension of SDRE

# System representation

- The model:

$$x_{t+1} = f(x_t) + g(x_t)u_t + G\xi_t$$

$$y_t = h(x_t)$$

- Linear State Dependent form of the model:

$$x_{t+1} = A(x_t)x_t + B(x_t)u_t + G\xi_t$$

$$y_t = C(x_t)x_t$$

- Assumption on the parameterisation of the model:

$$\forall_{x,u} \left( A(x_t), B(x_t) \right) \quad \text{is} \quad \text{controllable}$$

# System representation

- Simplified notation:

$$A_t = A(x_t), B_t = B(x_t), C_t = C(x_t)$$

- The Prediction of the trajectory:

$$\bar{u}_t, \bar{u}_{t+1}, \dots, \bar{u}_{t+N-1} \rightarrow \bar{x}_{t+1}, \bar{x}_{t+2}, \dots, \bar{x}_{t+N}$$

- Assumption on the trajectory after the prediction horizon.

# The NLQG algorithm

- Estimate (or measure) the state  $x(t)$
- Use previous feedback gain  $K(t-1)$  to calculate prediction of current control  $\underline{u}(t)$

$$\underline{u}(t) = -K(t-1)x(t)$$

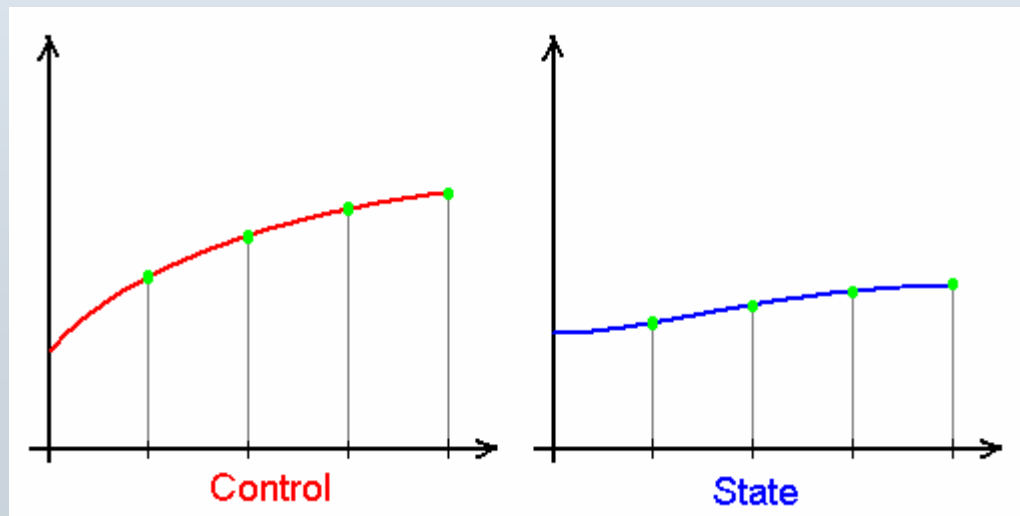
- Use current control prediction  $\underline{u}(t)$  and the model re-calculated at time instant  $t$  (with the state  $\underline{x}(t)$ ) to obtain future state prediction  $\underline{x}(t+1)$ .

# The NLQG algorithm

- The state prediction  $\underline{x}(t+1)$  together with the state feedback gain  $K(t-1)$  from previous iteration of the algorithm is used for a calculation of the future control prediction  $\underline{u}(t+1)$ .

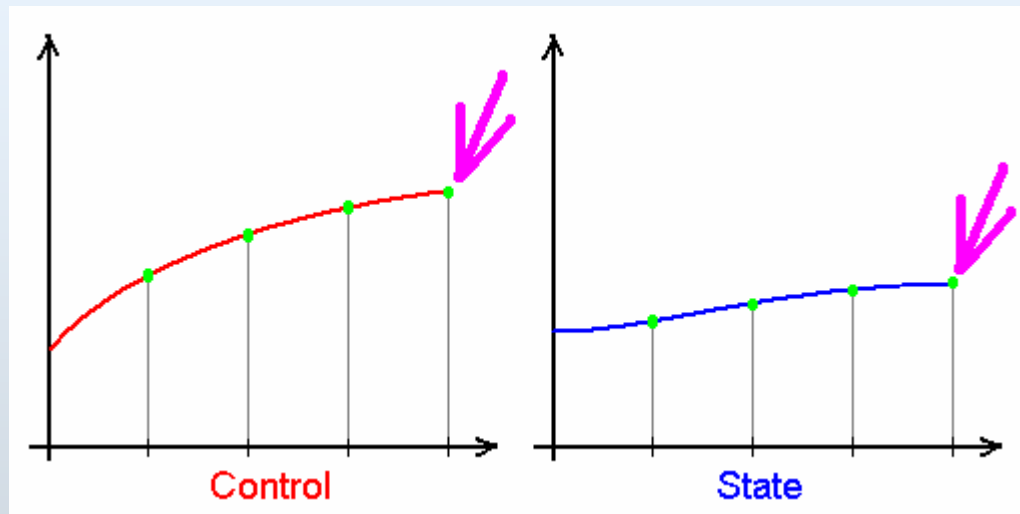
$$\underline{u}(t) = -K(t-1)x(t)$$

- The model once again is re-calculated using future state prediction, stored and sequence is repeated  $n$  times.



# The NLQG algorithm

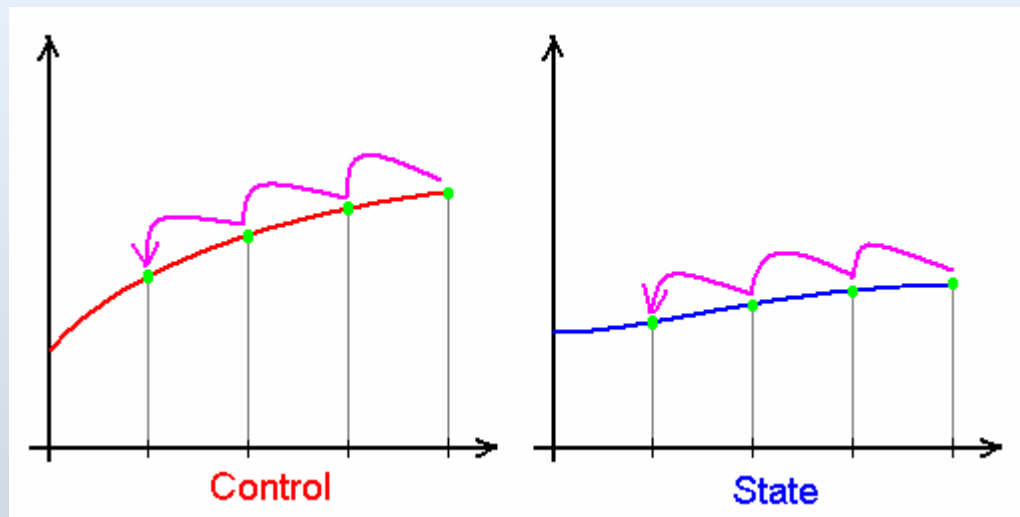
- Use the model prediction for time instant  $t+n$  and solve Algebraic Riccati Equation.



- The solution at time instant  $t+n$  is obtained:  $P(t+n, \infty)$

# The NLQG algorithm

- Use as a boundary condition  $P(t+n) = P(t+n, \infty)$  for iterations of the Riccati Difference Equation and use appropriate prediction of the model throughout iterations of Riccati Equation.



- Use  $P(k+1)$  to calculate the feedback control gain and calculate the current control.

# The NLQG algorithm

- Use  $P(k+1)$  to calculate the feedback control gain and calculate the current control.

$$u(t) = -K(t)x(t)$$

$$K(t) = \text{function}(A(t), B(t), P(t+1))$$

- Calculated current control is used for the plant input signal manipulation.

# The NLQG cost function

- The following cost function is minimised:

$$J_t = E \left\{ \lim_{T_h \rightarrow \infty} \frac{1}{2T_h} \left( \sum_{k=t}^{t+T_h} \left\{ \begin{aligned} &x^T(k) Q_c(k) x(k) + u^T(k) R_c u(k) \\ &+ u^T(k) M_c^T x(k) + x^T(k) M_c u(k) \end{aligned} \right\} \right) \right\}$$

- The cost function may be split in two parts:

$$J_t = E \left\{ \lim_{T_h \rightarrow \infty} \frac{1}{2T_h} \left( J_t^{finite} + J_t^{infinite} \right) \right\}$$

# The NLQG cost function

- First part is an infinite cost function:

$$J_t^{infinite} = \sum_{k=t+n}^{t+T_h} \left\{ \begin{array}{l} x^T(k) Q_c(t+n_p) x(k) + u^T(k) R_c u(k) \\ + u^T(k) M_c^T x(k) + x^T(k) M_c u(k) \end{array} \right\}$$

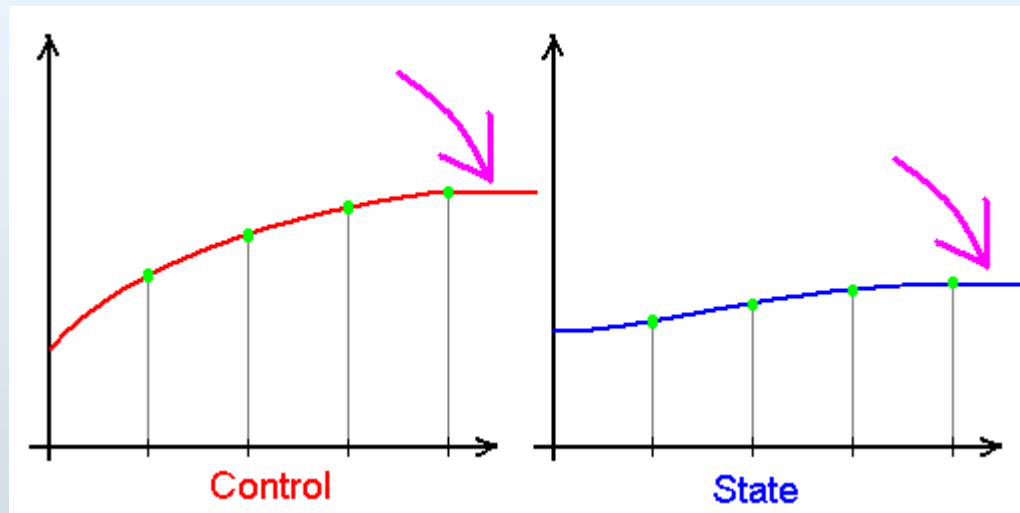
- and this part is minimised by Algebraic Riccati Equation.
- Second part is a finite cost function

$$J_t^{finite} = \sum_{k=t}^{t+n-1} \left\{ \begin{array}{l} x^T(k) Q_c(k) x(k) + u^T(k) R_c u(k) \\ + u^T(k) M_c^T x(k) + x^T(k) M_c u(k) \end{array} \right\}$$

- And is minimised by Difference Riccati Equation with border condition given by the solution of ARE.

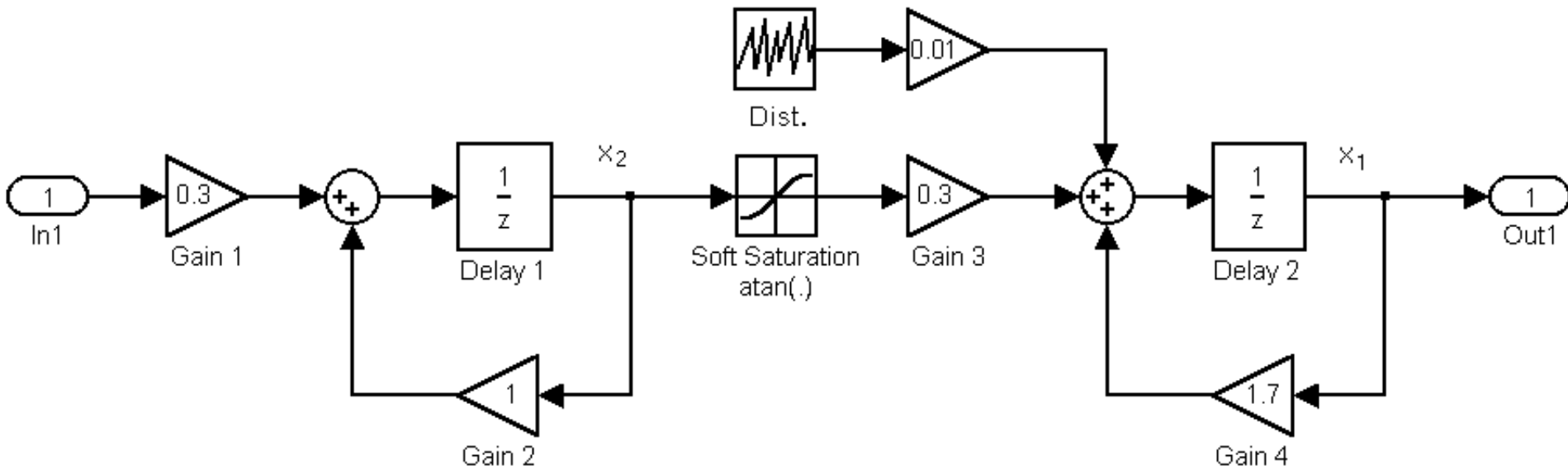
# Remarks

- To obtain accurate results it should be assumed that system will remain time invariant after  $t+n$  time instant:



- Therefore if real behaviour of the system is closer to the assumption results are more accurate

# Example



# Example

- Plant model

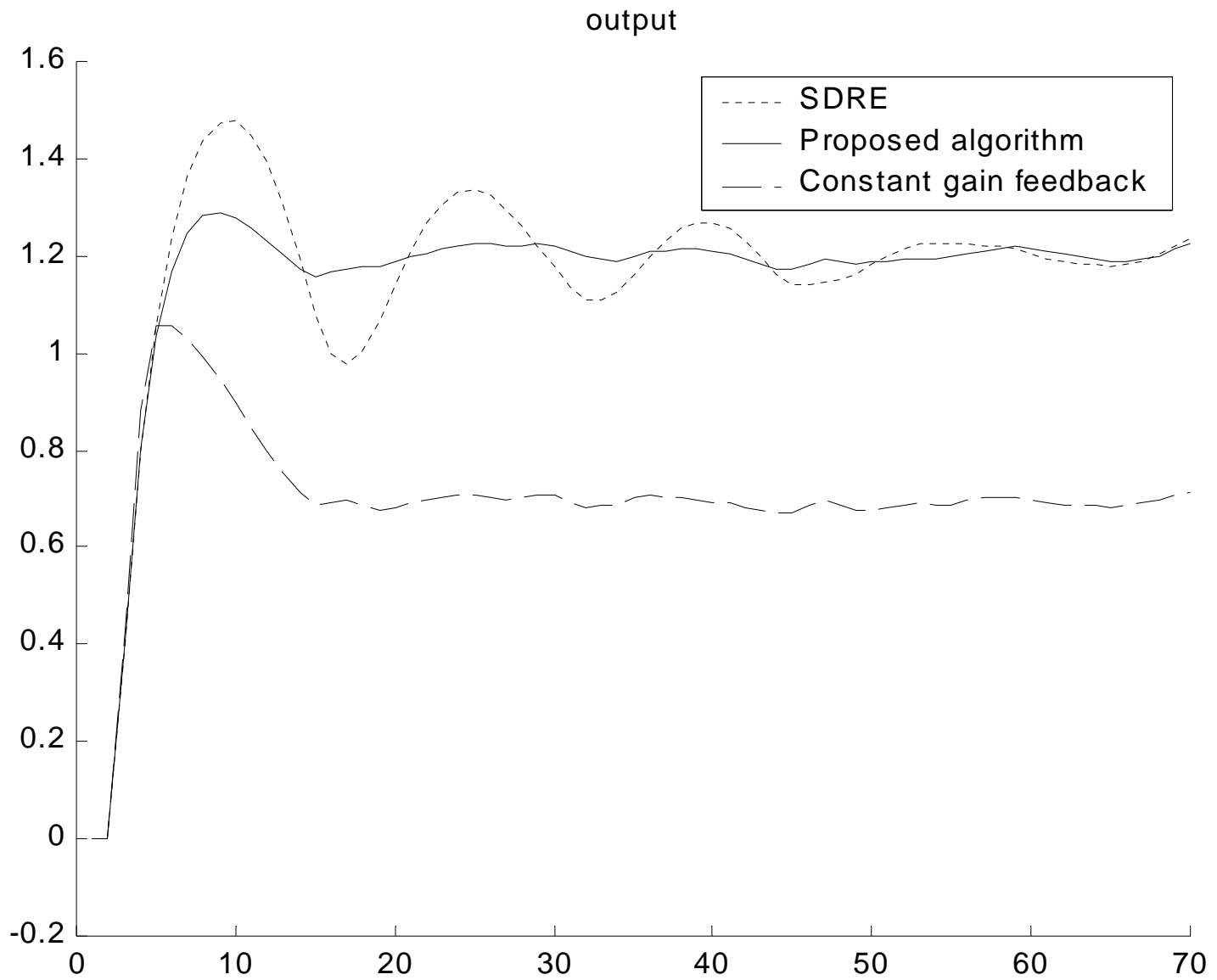
$$x_p(t+1) = \begin{bmatrix} 1.7 & \frac{\operatorname{atan}(x_{p,2}(t))}{x_{p,2}(t)} \\ 0 & 1 \end{bmatrix} x_p(t) + \begin{bmatrix} 0 \\ 0.3 \end{bmatrix} u(t) + \begin{bmatrix} 0.01 \\ 0 \end{bmatrix} \xi_p(t)$$
$$y_h(t) = y(t) = [1 \quad 0] x_p(t)$$

- Reference model

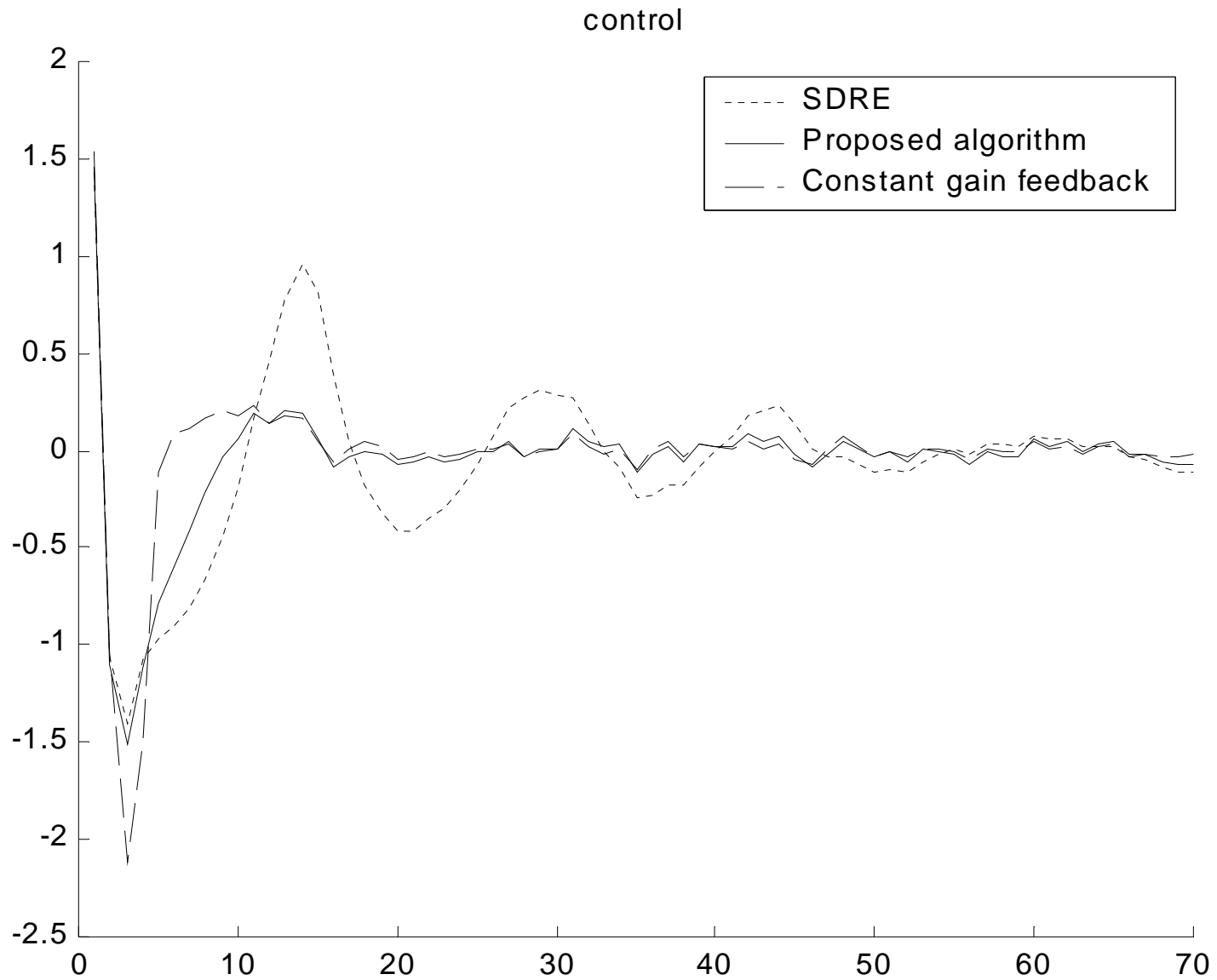
$$x_r(t+1) = [1] x_r(t) + [0] \xi_r(t)$$

$$r_h(t) = [1] x_r(t)$$

# Results



# Results



# The NLQGPC control law derivation

- Re-written state equation:

$$x_{t+1} = A_t x_t + \beta_t U_{t,N} + G \xi_t$$

where

$$\beta_t = [B_t, 0_1, 0_2, \dots, 0_N]$$

- State-space model with prediction-output equation

$$x_{t+1} = A_t x_t + \beta_t U_{t,N} + G \xi_t$$

$$Y_{t+1,N} = \Phi_{t,N} A_t x_t + S_{t,N} U_{t,N} + G_{t,N} \bar{\Xi}_{t,N-1}$$

Note: that state equation is identical to the state equation of the controlled system.

# The NLQGPC control law derivation

- Reference signal model:

$$X_{t+1}^R = A^R X_t^R + G^R \xi_t^R$$

$$R_{t+1,N} = C^R X_t^R$$

- Augmented system :

$$\chi_{t+1} = \Theta_t \chi_t + \Omega_t U_{t,N} + \Gamma \xi_t^A$$

$$\Psi_{t+1,N} = Y_t \chi_t + S_{t,N} U_{t,N} + G_{t,N} \Xi_{t,N-1}$$

with

$$\chi_t = \begin{bmatrix} x_t \\ X_t^R \end{bmatrix}, \Theta_t = \begin{bmatrix} A_t & 0 \\ 0 & A^R \end{bmatrix}, \xi_t^A = \begin{bmatrix} \xi_t \\ \xi_t^R \end{bmatrix}, \Omega_t = \begin{bmatrix} \beta_t \\ 0 \end{bmatrix}, \Gamma = \begin{bmatrix} G & 0 \\ 0 & G^R \end{bmatrix},$$

$$\Psi_{t+1,N} = Y_{t+1,N} - R_{t+1,N}, Y_t = \begin{bmatrix} \Phi_{t,N} A_t & -C^R \end{bmatrix}$$

# The NLQGPC control law derivation

- The cost function:

$$J_t = E \left\{ \lim_{T_h \rightarrow \infty} \left[ \frac{1}{2T_h} \sum_{k=t}^{t+T_h} \left( \sum_{i=1}^N \left\{ (y_{k+i} - r_{k+i})^T \Lambda_E^i (y_{k+i} - r_{k+i}) \right\} + \sum_{i=0}^N \left\{ u_{k+i}^T \Lambda_U^i u_{k+i} \right\} \right) \right] \right\}$$

or:

$$J_t = E \left\{ \lim_{T_h \rightarrow \infty} \left[ \frac{1}{2T_h} \sum_{k=t}^{t+T_h} \left( \begin{aligned} &\chi_k^T \cdot \Upsilon_k^T \Lambda_E \Upsilon_k \cdot \chi_k + U_{k,N}^T \cdot S_{k,N}^T \Lambda_E \Upsilon_k \cdot \chi_k + \\ &\chi_k^T \cdot \Upsilon_k^T \Lambda_E S_{k,N} \cdot U_{k,N} + U_{k,N}^T \cdot S_{k,N}^T \Lambda_E S_{k,N} \cdot U_{k,N} + \\ &+ U_{k,N}^T \cdot \Lambda_U \cdot U_{k,N} \end{aligned} \right) \right] \right\} + J_0$$

Introduce notation:

$$Q_k = \Upsilon_k^T \Lambda_E \Upsilon_k, \quad M_k = \Upsilon_k^T \Lambda_E S_{k,N}, \quad R_k = S_{k,N}^T \Lambda_E S_{k,N} + \Lambda_U$$

Final form:

$$J_t = E \left\{ \lim_{T_h \rightarrow \infty} \left[ \frac{1}{2T_h} \sum_{k=t}^{t+T_h} \left( \begin{bmatrix} \chi_k^T & U_{k,N}^T \end{bmatrix} \begin{bmatrix} Q_k & M_k \\ M_k^T & R_k \end{bmatrix} \begin{bmatrix} \chi_k \\ U_{k,N} \end{bmatrix} \right) \right] \right\} + J_0$$

# NLQGPC control law derivation - *Algorithm 1*

The control law minimising the cost function:

$$U_{t,N} = -\left(\Omega_t^T P_{t,\infty} \Omega_t + R_t\right)^{-1} \left(\Omega_t^T P_{t,\infty} \Theta_t + M_t^T\right) \chi_t$$

where  $P_{t,\infty}$  : solution of Algebraic Riccati Equation

$$P_{t,\infty} = Q_t + \Theta_t^T P_{t,\infty} \Theta_t - \left(M_t + \Theta_t^T P_{t,\infty} \Omega_t\right) \left(R_t + \Omega_t^T P_{t,\infty} \Omega_t\right)^{-1} \left(M_t^T + \Omega_t^T P_{t,\infty} \Theta_t\right)$$

This Algebraic Riccati Equation contains state dependent matrices  $Q_t, M_t, R_t$  calculated at time  $t$ , which contain the prediction of future system behaviour

# NLQGPC control law derivation - *Algorithm 2*

- A more accurate solution of the minimisation problem:

$$J_t = E \left\{ \lim_{T_h \rightarrow \infty} \left[ \frac{1}{2T_h} \left( \sum_{k=t}^{t+N-1} \left( \begin{bmatrix} \chi_k^T & U_{k,N}^T \end{bmatrix} \begin{bmatrix} Q_k & M_k \\ M_k^T & R_k \end{bmatrix} \begin{bmatrix} \chi_k \\ U_{k,N} \end{bmatrix} \right) + \sum_{k=t+N}^{t+N+T_h} \left( \begin{bmatrix} \chi_k^T & U_{k,N}^T \end{bmatrix} \begin{bmatrix} Q_k & M_k \\ M_k^T & R_k \end{bmatrix} \begin{bmatrix} \chi_k \\ U_{k,N} \end{bmatrix} \right) \right] \right\} + J_0$$

- Cost function split in two parts
  - finite horizon
  - infinite horizon.

# NLQGPC control law derivation - *Algorithm 2*

- Difference Riccati Equation :

$$P_k = Q_k + \Theta_k^T P_{k+1} \Theta_k - \left( M_k + \Theta_k^T P_{k+1} \Omega_k \right) \left( R_k + \Omega_k^T P_{k+1} \Omega_k \right)^{-1} \left( M_k^T + \Omega_k^T P_{k+1} \Theta_k \right)$$

boundary condition  $P_{t+N} = P_{t+N,\infty}$

iterated backwards for  $k = t + N - 1, t + N - 1, \dots, t + 1$

- The control vector minimising cost function:

$$U_{t,N} = - \left( \Omega_t^T P_{t+1} \Omega_t + R_t \right)^{-1} \left( \Omega_t^T P_{t+1} \Theta_t + M_t^T \right) \chi_t$$

# Example

- The model is given by the following non-linear state space equations

$$\zeta_{t+1}^1 = \zeta_t^1 - 0.3 \cdot \sin(\zeta_t^1) + \zeta_t^2 + g \xi_t^1$$

$$\zeta_{t+1}^2 = \zeta_t^2 - 0.3 \cdot (\zeta_t^1)^3 + \nu_t + g \xi_t^2$$

$$y_t = \zeta_t$$

- Next the model is re-arranged into Linear State Dependent form:

$$\zeta_{t+1} = \begin{bmatrix} 1 - \frac{0.3 \cdot \sin(\zeta_t^1)}{\zeta_t^1} & 1 \\ 0 & 1 - 0.3 \cdot (\zeta_t^1)^2 \end{bmatrix} \zeta_t + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \nu_t + \begin{bmatrix} g & 0 \\ 0 & g \end{bmatrix} \xi_t$$
$$y_t = [1 \quad 0] \zeta_t + [0] \nu_t,$$

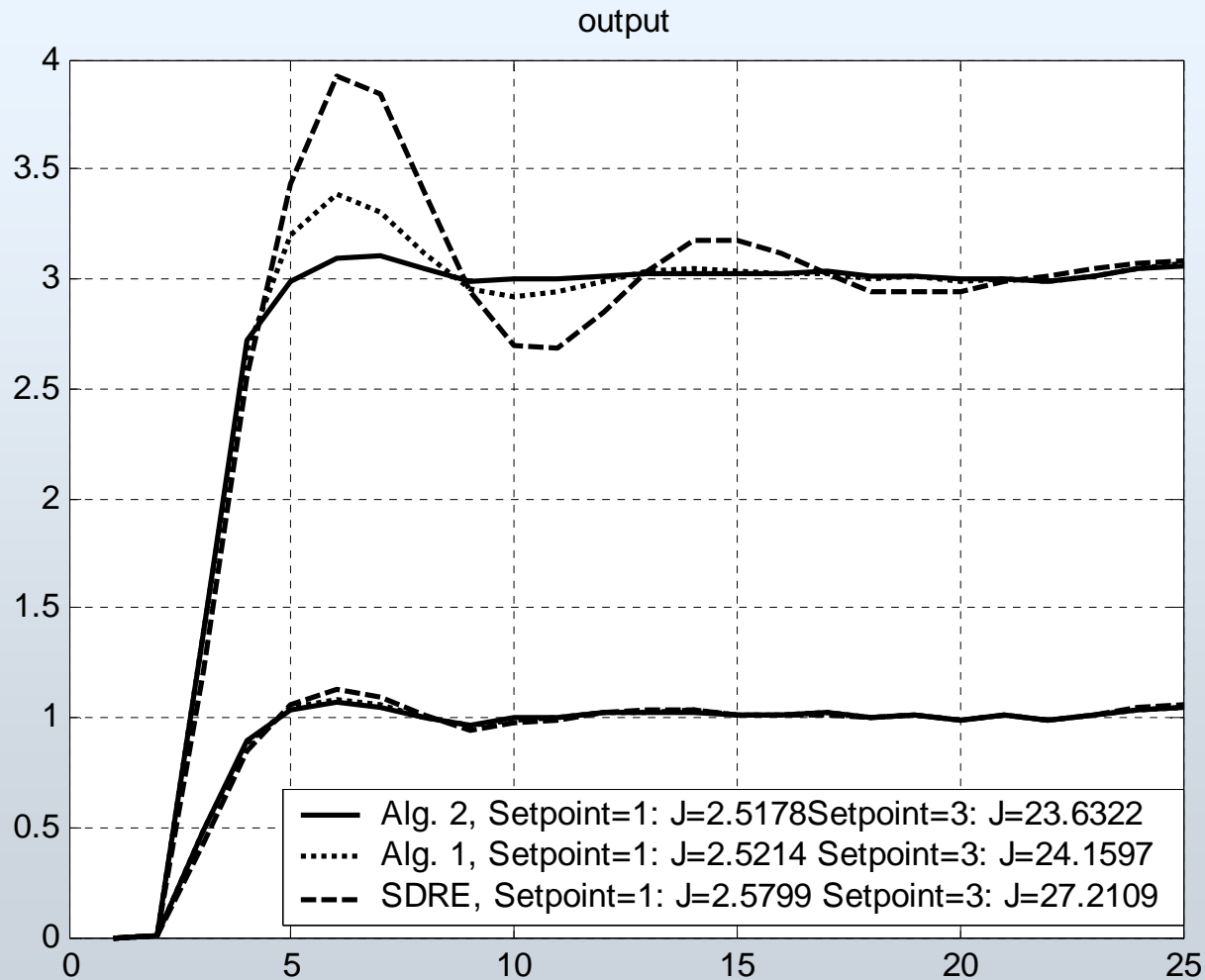
# Example

- The system is controllable since

$$\forall_{\zeta} \text{rank} \left( \begin{bmatrix} 0 & 1 \\ 1 & 1 - 0.3 \cdot \left( \zeta_t^{(2)} \right)^2 \end{bmatrix} \right) = 2$$

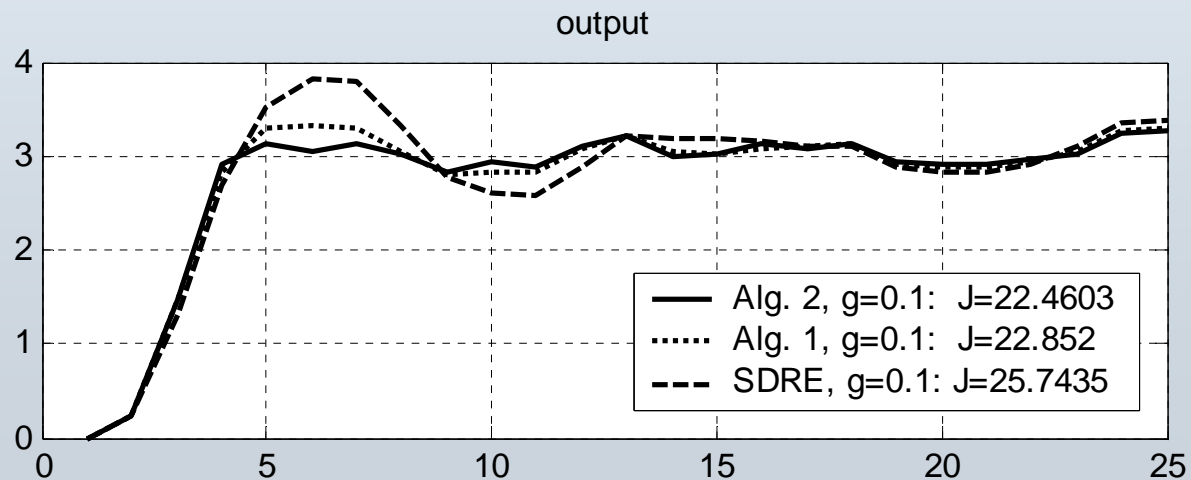
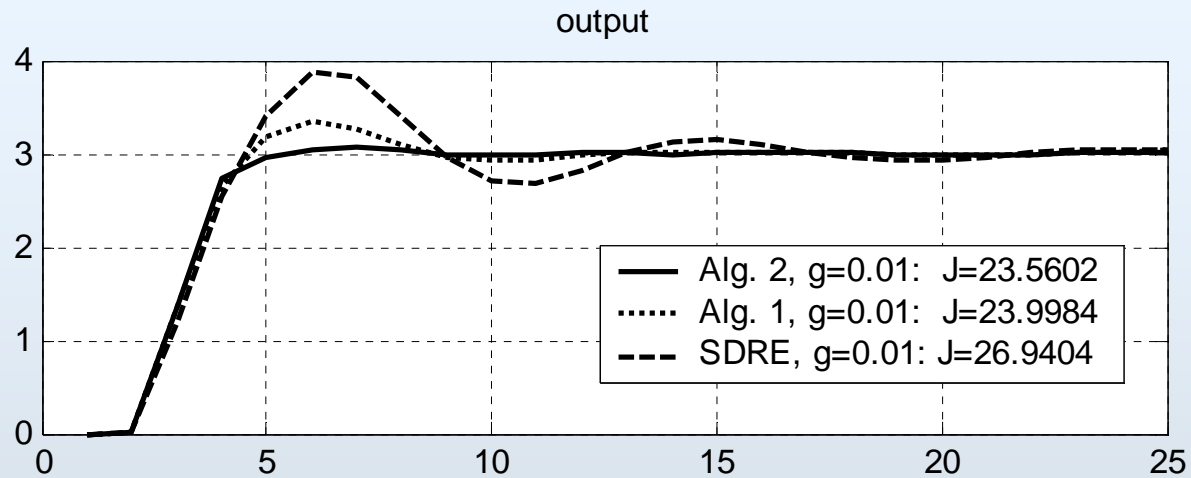
# Example

- The step response for two NLQGPC algorithms is compared with SDRE with  $g=0.01$  (noise level)



# Example

- Now compare noise rejection (for two levels of process noise)



# Advantages & Disadvantages of NLQGPC

- Advantages:

1. Controls based on solutions to the NLQGPC have been shown to offer high performance.
2. Less computational burden than other non-linear predictive control techniques.

- Disadvantages:

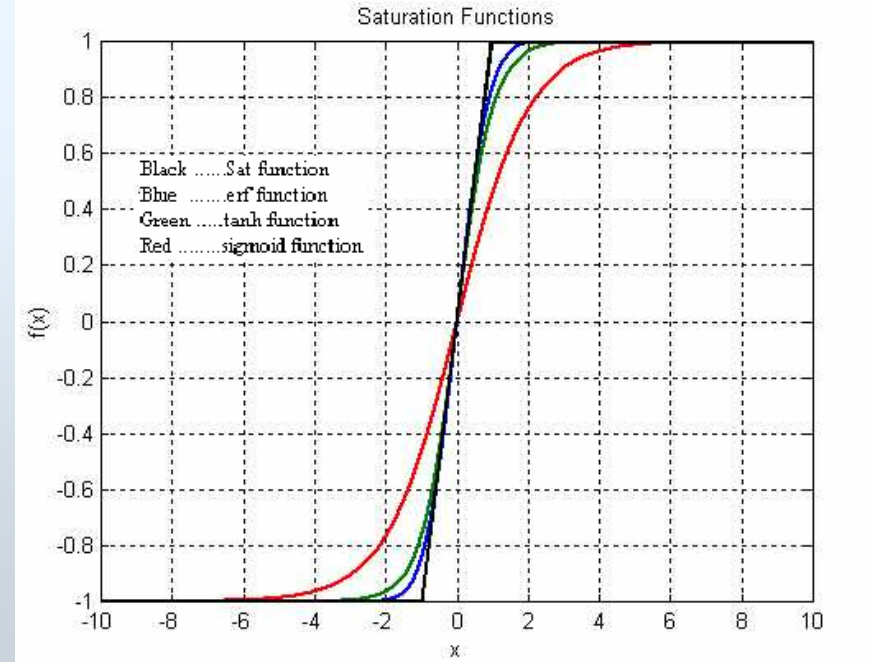
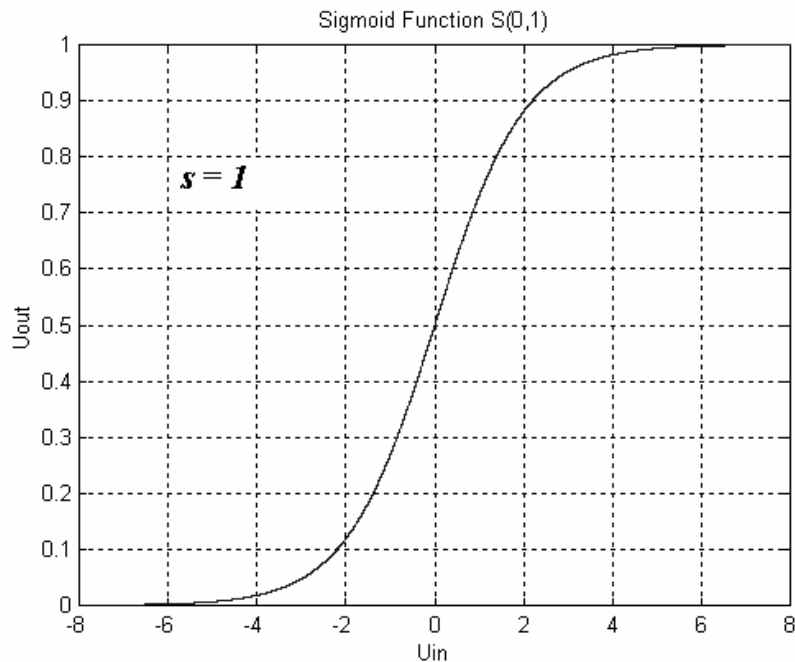
1. Since NLQGPC utilizes the Riccati equation, it is an unconstrained predictive control technique.
2. Like SDRE, NLQGPC doesn't guarantee closed-loop global stability.

# Dealing with constraints in NLQGPC

The input constraints can be approximated by means of smooth limiting functions, and then included into the dynamics of the plant in a state-dependent state-space form.

$$0 \leq u \leq \alpha$$

$$-\alpha \leq u \leq \alpha$$



# Improving stability via “Satisficing”

**Satisficing** is based on a point-wise cost/benefit comparison of an action.

The benefits are given by the “**Selectability**” function  $P_s(u,x)$ , while the costs are given by the “**Rejectability**” function  $P_r(u,x)$ .

The “satisficing” set is those options for which selectability exceeds rejectability: i.e.,

$$S(x,b) = \{u : P_s(u,x) \geq bP_r(u,x)\}$$

# CLF-Based Satisficing Technique

The selectability criteria is defined to be:

$$P_s(u, x) = -V_x^T (f + gu)$$

The rejectability criteria is defined to be:

$$P_r(u, x) = l(x) + x^T R x$$

$b=0$ , therefore, the satisficing set  $S$ :

$$S(x, b) = \{u : -V_x^T (f + gu) \geq 0\}$$

## Augmenting NLQGPC with Satisficing

By projecting the NLQGPC controller point-wise onto the satisficing set, the good properties of the NLQGPC approach are combined with the analytical properties of satisficing.

# Example: Control of F-8 aircraft



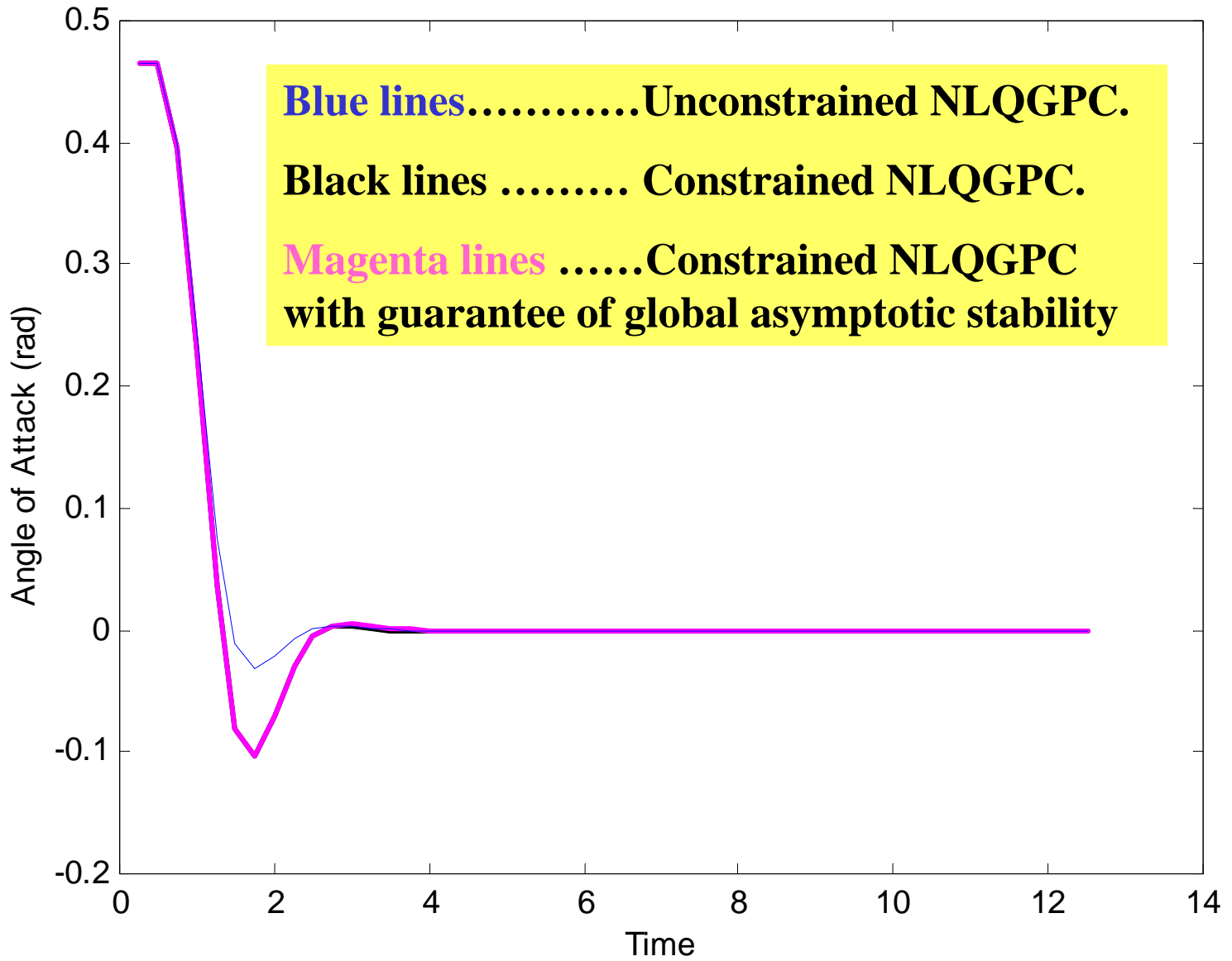
The non-linear dynamical model of the F-8 fighter aircraft:

$$\begin{aligned}\dot{x}_1 = & -0.877x_1 + x_3 - 0.088x_1x_3 + 0.47x_1^2 - 0.019x_2^2 \\ & - x_1^2x_3 + 3.846x_1^3 - 0.215u + 0.28x_1^2u + 0.47x_1u^2 + 0.63u^3,\end{aligned}$$

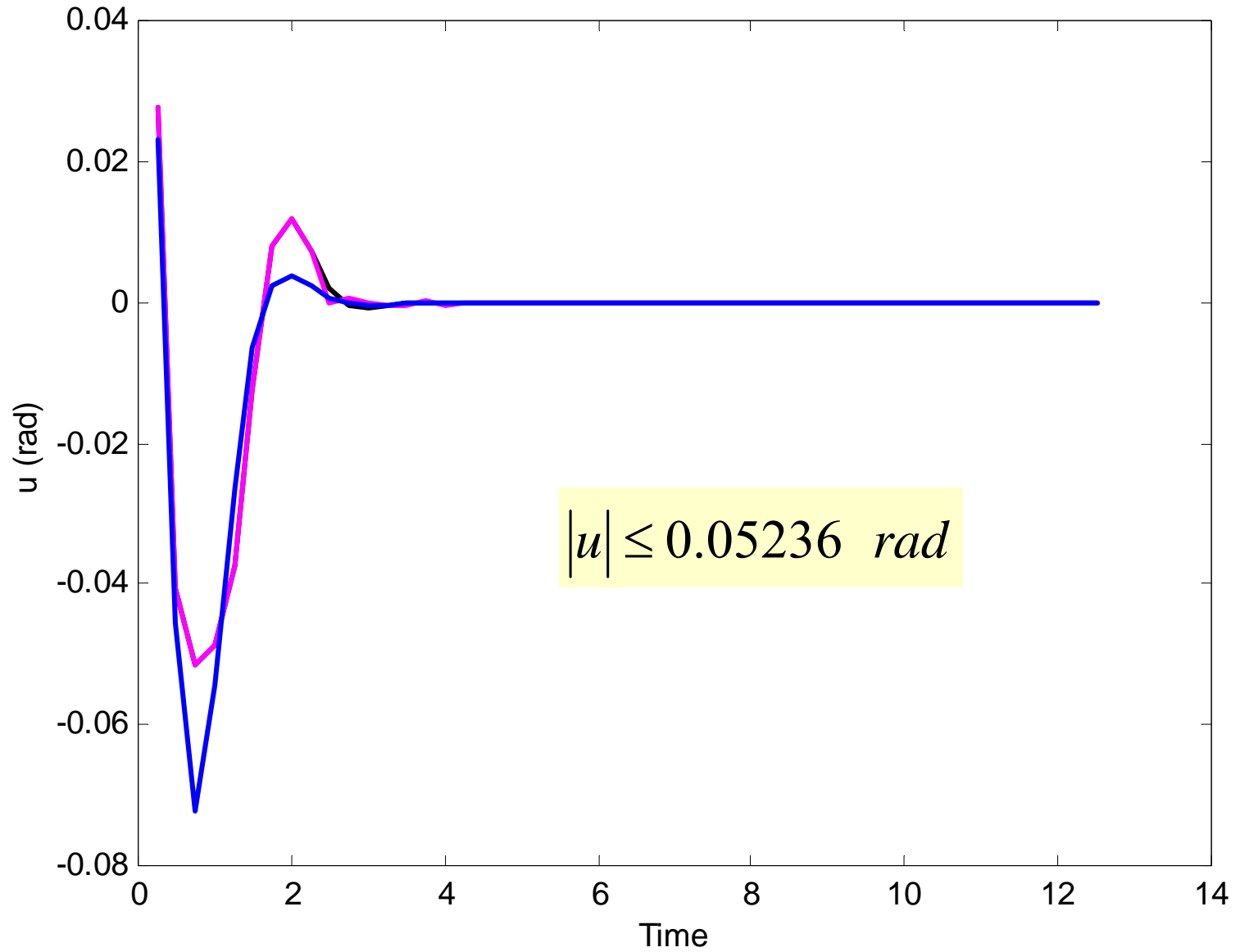
$$\dot{x}_2 = x_3,$$

$$\begin{aligned}\dot{x}_3 = & -4.208x_1 - 0.396x_3 - 0.47x_1^2 - 3.564x_1^3 \\ & - 20.967u + 6.265x_1^2u + 46x_1u^2 + 61.4u^3\end{aligned}$$

$$|u| \leq 0.05236 \text{ rad}$$



# Elevator Deflection



**NLQGPC**

```
graph TD; NLQGPC[NLQGPC] --> HP[High Performance]; NLQGPC --> DIC[Deals with input Constraints]; NLQGPC --> LCB[Low Computational Burden]; NLQGPC --> GRAS[Guarantee of Robustness & Asymptotic Stability];
```

**High  
Performance**

**Deals with input  
Constraints**

**Low  
Computational  
Burden**

**Guarantee of  
Robustness &  
Asymptotic  
Stability**

# Non-Linear Generalized Minimum Variance Control

# Contents

- Introduction
- Nonlinear GMV control problem and solution
- Relationship to the Smith Predictor
- Incorporating future information:  
Feedforward and Tracking
- Simulation example

# Introduction

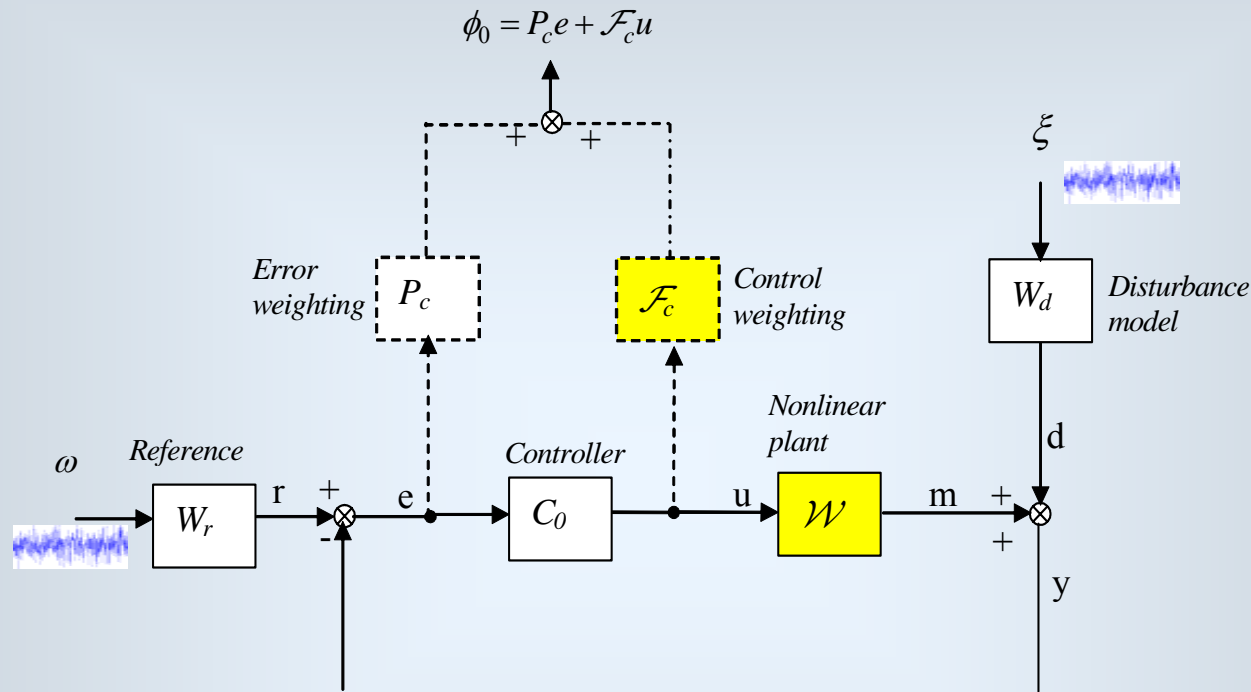
- 1969: Åström introduces *Minimum Variance* (MV) controller assuming linear minimum phase plant. Successful applications in pulp and paper industry.
- 1970s: Clarke and Hastings-James modify the MV control law by adding a control costing term. This is termed a *Generalized Minimum Variance* (GMV) control law and is the basis for their later self-tuning controller.
- The GMV control law has similar characteristics to LQG design in some cases and is much simpler to implement
- However, when the control weighting tends to zero the control law reverts to the initial algorithm of Åström, which is unstable for non-minimum phase processes.

# Introduction

Aim: *introduce a GMV controller for nonlinear, multivariable, possibly time-varying processes*

- The structure of the system is defined so that a simple solution is obtained.
- When the system is linear the results revert to those for the linear GMV controller.
- There is some loss of generality in assuming the reference and disturbance models are represented by linear subsystems.
- However, *plant model can be in a very general nonlinear operator form, which might involve state-space, transfer operators, neural networks or even nonlinear function look-up tables.*

# Nonlinear system description



**Nonlinear plant model:**

$$(\mathcal{W}u)(t) = z^{-k} (\mathcal{W}_k u)(t)$$

**Linear disturbance model:**

$$W_d = A_f^{-1} C_d$$

**Linear reference model:**

$$W_r = A_f^{-1} E_r$$

# Plant model

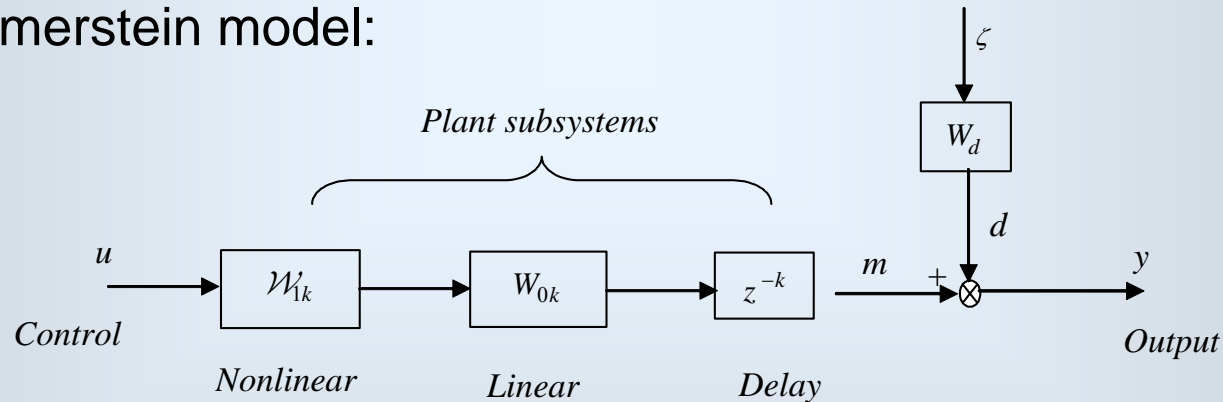
Nonlinear plant model can be given in a very general form, e.g.:

- state-space formulation
- neural network / neuro-fuzzy model
- look-up table
- Fortran/C code



$$f(u, y) = 0$$

It can include both linear and nonlinear components, e.g. Hammerstein model:



Just need to obtain the output to given input signal

# NGMV problem formulation

To minimize: variance of the “generalized” output  $\phi_0(t)$ :

$$J_{NGMV} = E[\phi_0^2(t)]$$

with  $\phi_0(t) = P_c e(t) + (\mathcal{F}_c u)(t)$

$P_c = P_{cn} P_{cd}^{-1}$  - linear error weighting

$(\mathcal{F}_c u)(t) = z^{-k} (\mathcal{F}_{ck} u)(t)$  - control weighting (possibly nonlinear)

- Control weighting assumed invertible and potentially nonlinear to compensate for plant nonlinearities in appropriate cases
- The weighting selection is restricted by closed-loop stability

# NGMV problem solution

The approach also similar:

$$\begin{aligned}\phi_0(t) &= P_c(-z^{-k}(\mathcal{W}_k u)(t) + Y_f \varepsilon(t)) + (\mathcal{F}_c u)(t) \\ &= z^{-k}(\mathcal{F}_{ck} - P_c \mathcal{W}_k)u(t) + P_c Y_f \varepsilon(t)\end{aligned}$$

$$\phi_0(t) = F \varepsilon(t) + [(\mathcal{F}_{ck} - P_c \mathcal{W}_k)u(t-k) + R \varepsilon(t-k)]$$

statistically independent

$\varepsilon(t)$  – white noise (sequence of independent random variables)

$$P_c Y_f = F + z^{-k} R$$

Diophantine equation

$$Y_f Y_f^* = W_d W_d^* + W_r W_r^*$$

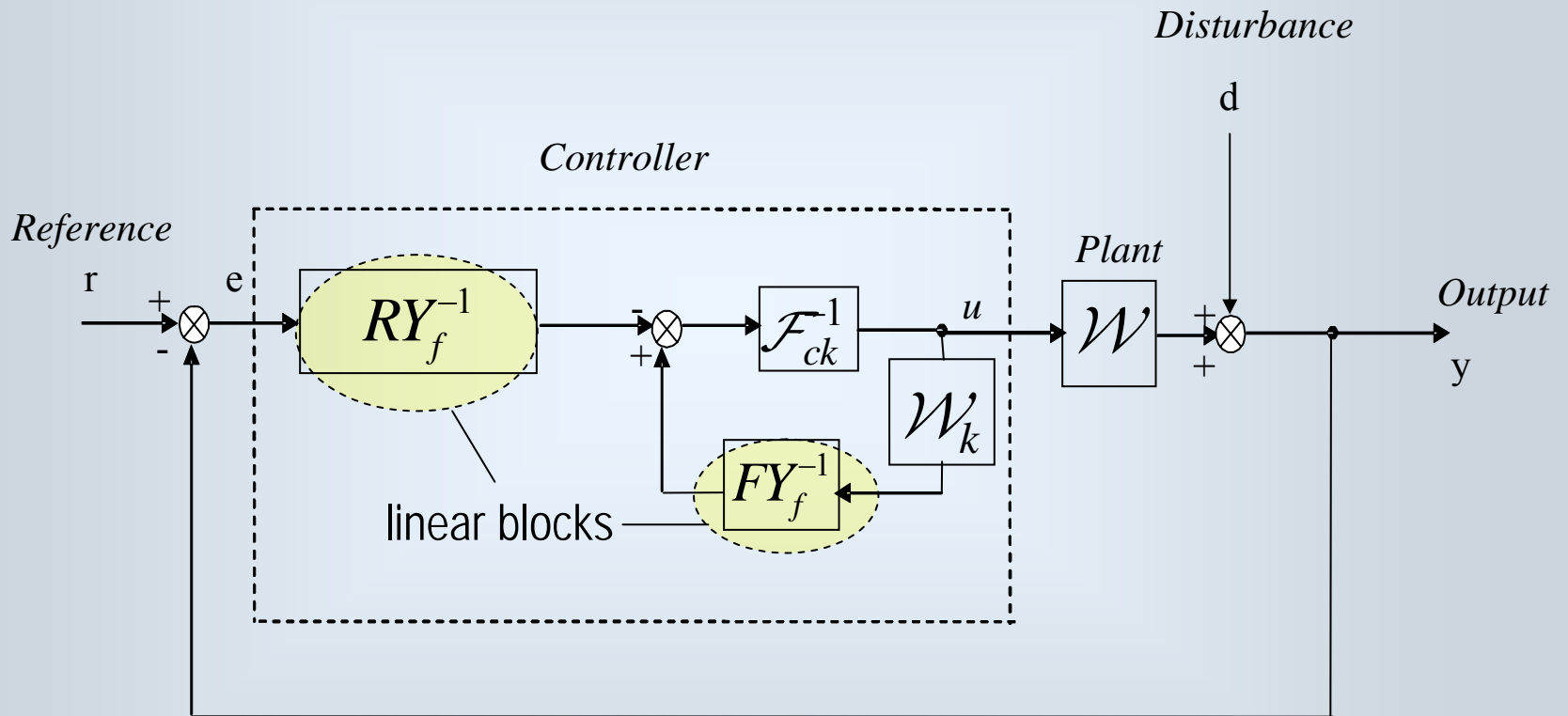
Spectral factorization

Optimal control:  $u^{NGMV}(t) = -(\mathcal{F}_{ck} - P_c \mathcal{W}_k)^{-1} R \varepsilon(t)$

stable causal nonlinear operator inverse

# Controller implementation

$$u^{NGMV}(t) = -[(\mathcal{F}_{ck} - FY_f^{-1}\mathcal{W}_k)^{-1}RY_f^{-1}e](t)$$

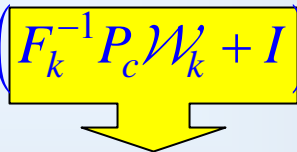


# Existence of a Stable Operator Inverse

- Necessary condition for optimality:

operator  $(P_c \mathcal{W}_k - \mathcal{F}_{ck})$  must have a stable inverse

- For linear systems: the operator must be strictly minimum-phase.
- To show this is satisfied for a very wide class of systems consider the case where  $\mathcal{F}_{ck}$  is linear and negative so that  $\mathcal{F}_{ck} = -F_k$ . Then obtain:

$$(P_c \mathcal{W}_k + F_k)u = F_k \left( F_k^{-1} P_c \mathcal{W}_k + I \right) u$$


return-difference operator for a feedback system with  
A delay-free plant and controller

$$K_c = F_k^{-1} P_c.$$

# PID-based initial design

Consider the delay-free plant  $\mathcal{W}_k$  and assume a PID controller  $K_{PID}$  exists to stabilize the closed-loop system.

Then a starting point for the weighting choice that will ensure the operator

$(P_c \mathcal{W}_k + F_k)$  is stably invertible is

$$P_c = K_{PID}, \quad F_k = 1$$

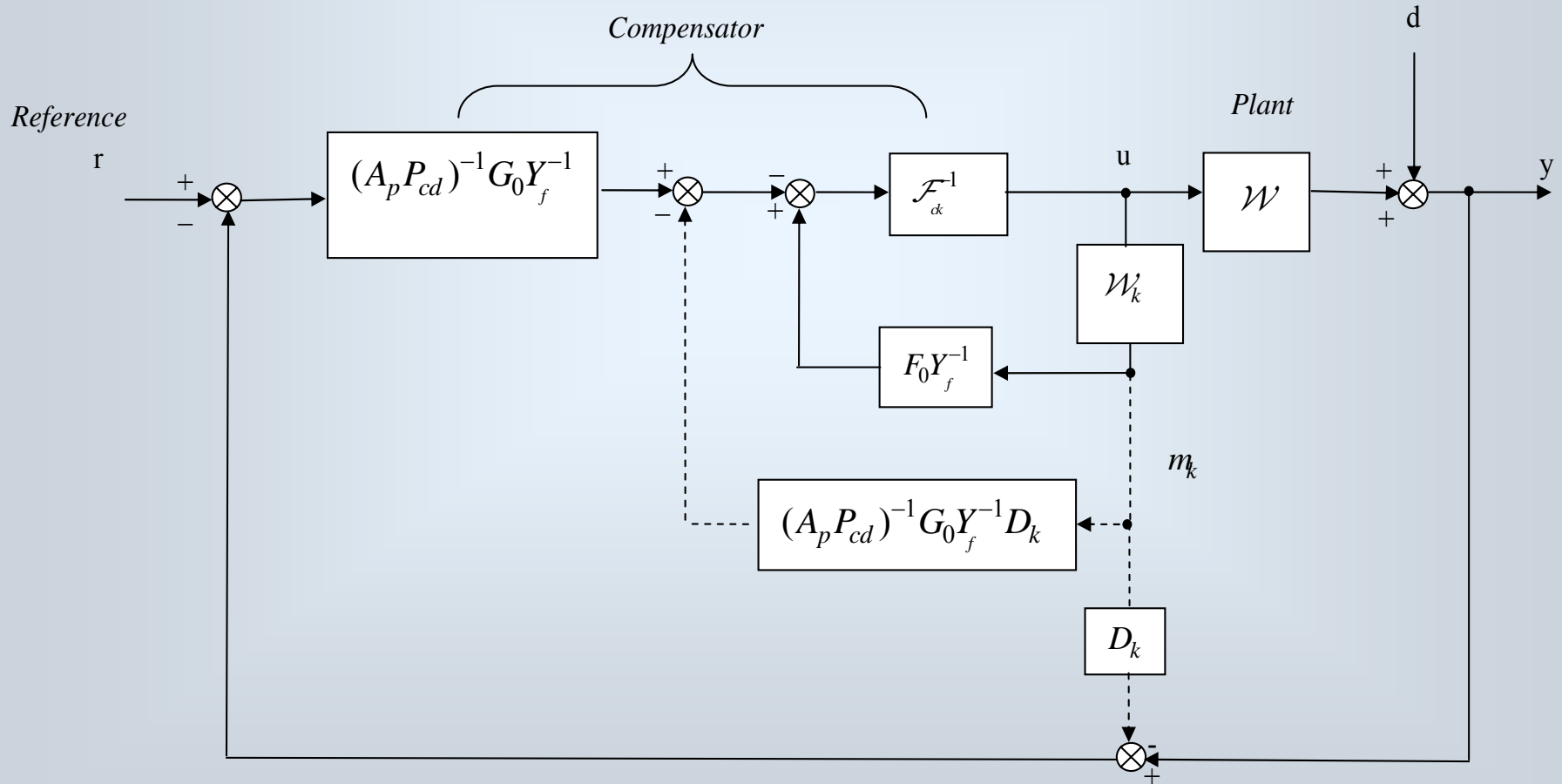
To demonstrate this selection reasonable consider scalar case and let controller

$$K_c = k_0 + \frac{k_1}{1-z^{-1}} + k_2(1-z^{-1}) = \frac{(k_0 + k_1 + k_2) - (k_0 + 2k_2)z^{-1} + k_2z^{-2}}{1-z^{-1}}$$

Assume the PID gains are positive numbers, with small derivative gain. Then simple to confirm if  $F_k = 1$  the  $P_{cn}$  term is minimum phase and has real zeros.

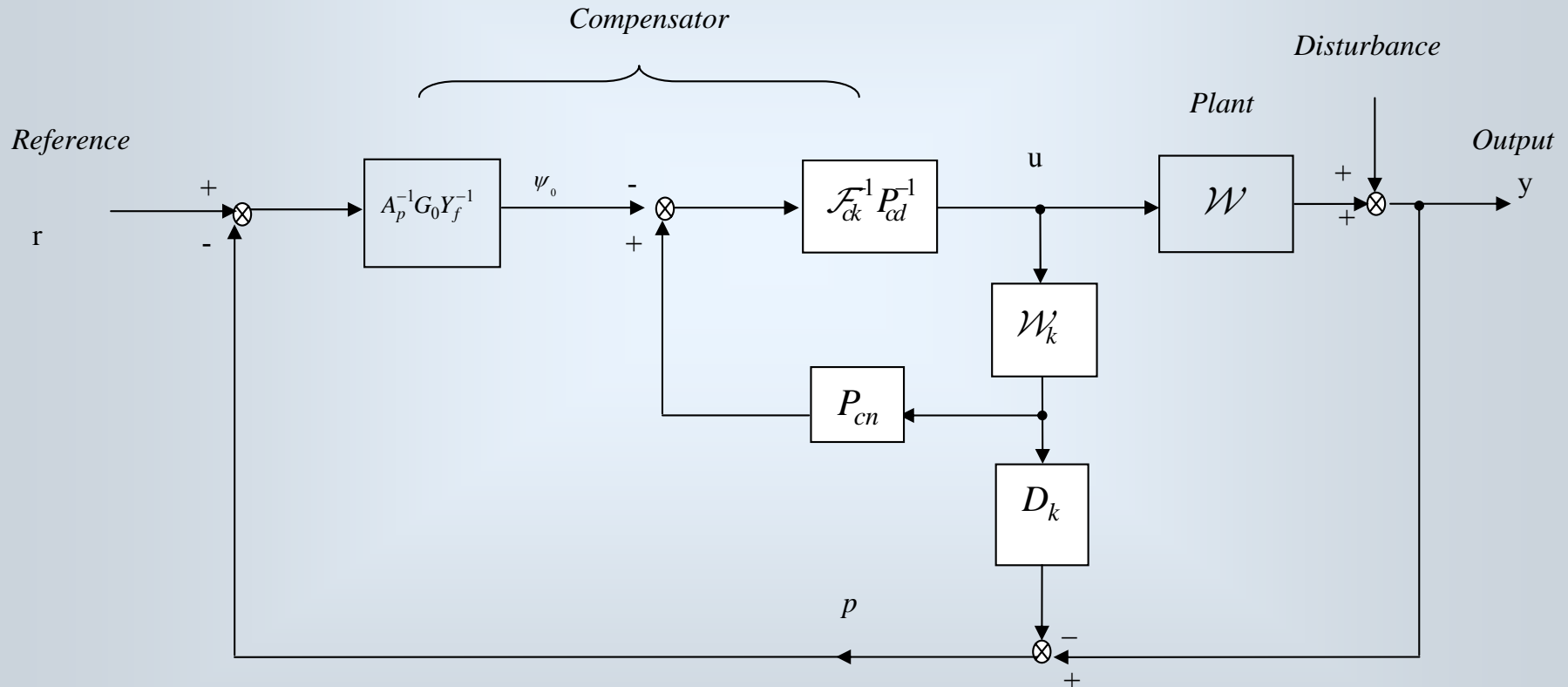
# Relationship to the Smith Predictor

The optimal controller can be expressed in a similar form to that of a Smith Predictor. This provides a new nonlinear version of the Smith Predictor.



# Smith Predictor form of NGMV controller

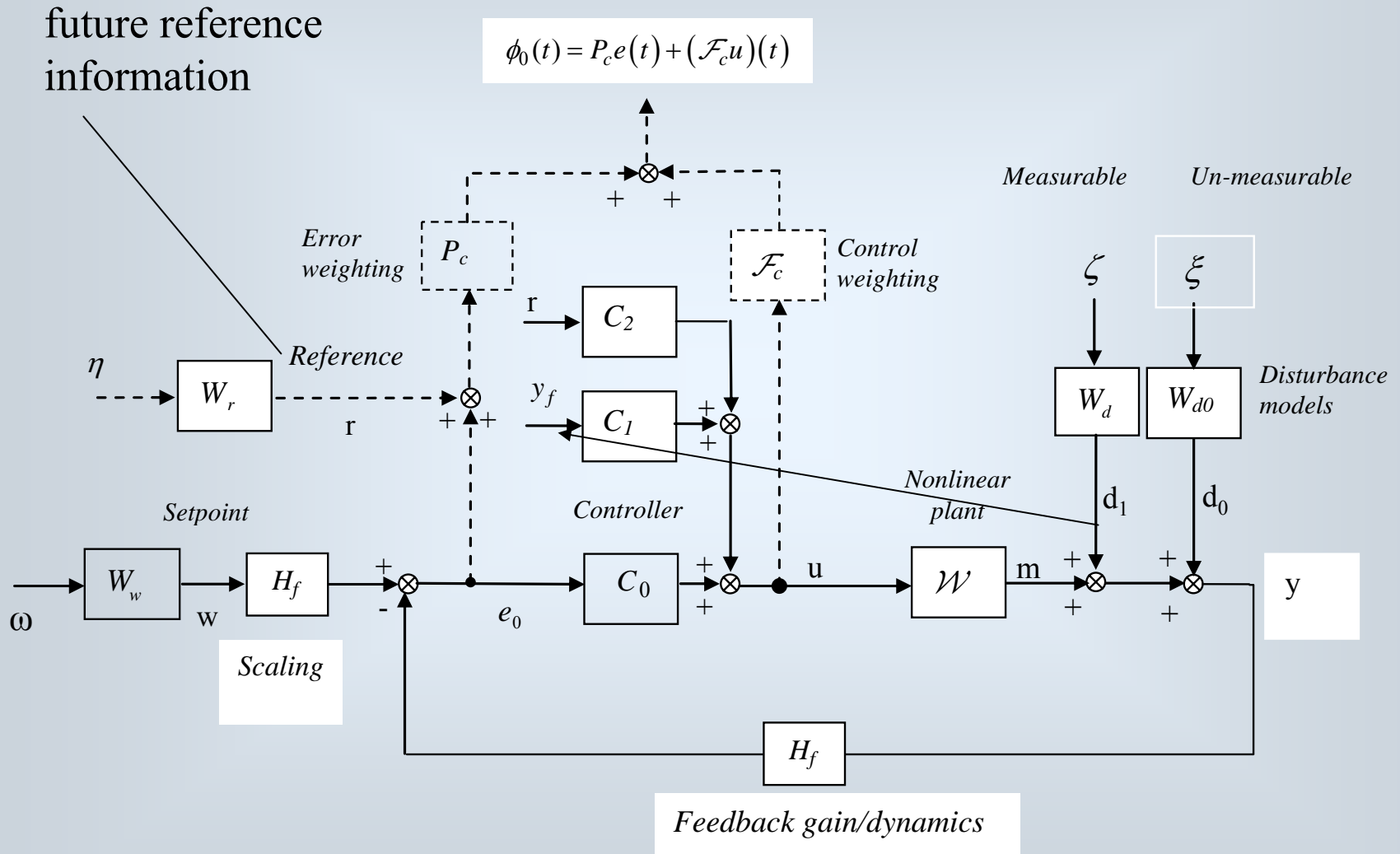
The system may be redrawn and the compensator rearranged as shown below. This structure is essential if  $P_c$  includes an integrator.



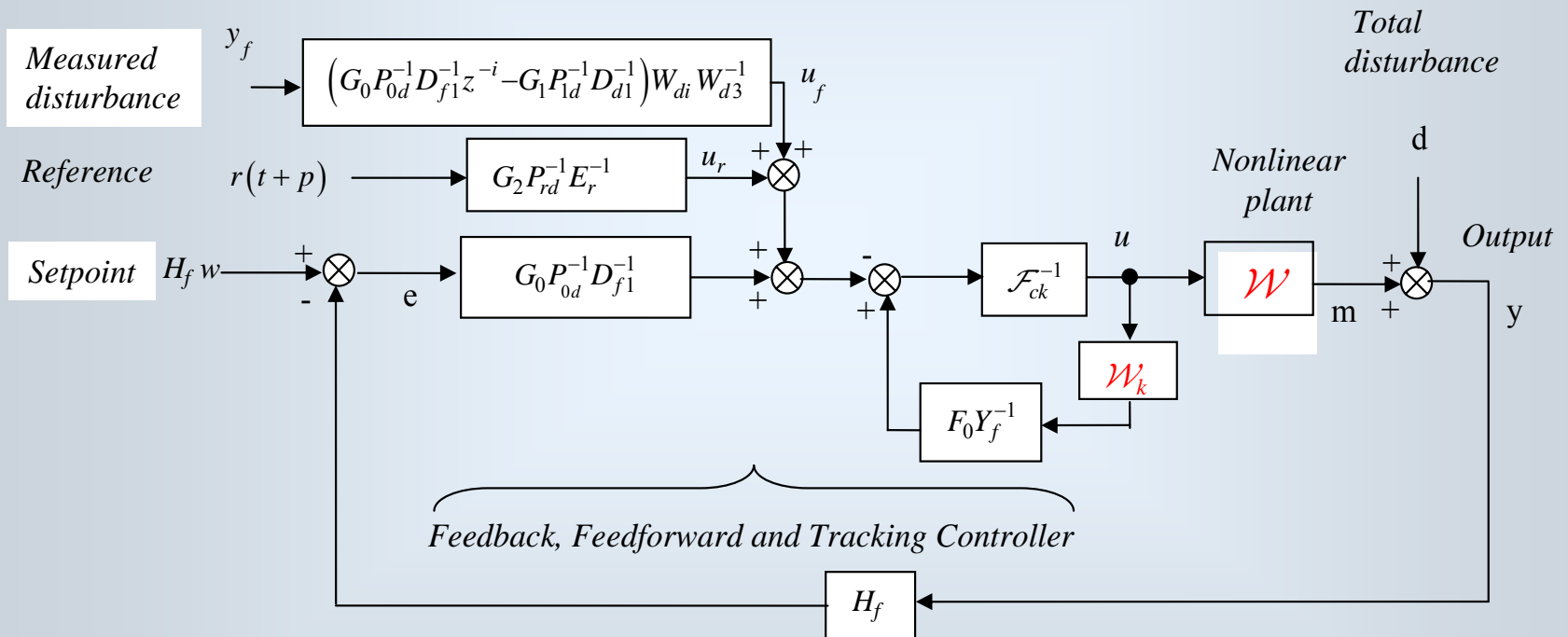
# Comments

- This last structure is intuitively reasonable. With no plant-model mismatch, the control is not due to feedback but involves an open-loop stable compensator.
- The nonlinear inner-loop has weightings  $\mathcal{F}_{ck}^{-1}P_c$  acting like an inner-loop controller. If weightings are chosen to be of usual form this will represent a filtered PID controller.
- Such a choice of weightings is only a starting point, since stability is easier to achieve. However, control weighting can have additional lead term and high frequency characteristics of optimal controller will then have more realistic roll off.
- **Stability:** Under the given assumptions the resulting *Smith* system is stable. This follows because the plant is stable, the inner-loop is stable and there are only stable terms in the input block.

# Feedback + Feedforward + Tracking Control



# Feedback, Tracking and Feedforward Control Signal Generation Control Modules



# Simulated example

2-by-2 model given in the non-linear state space form:

$$x_1(t+1) = \frac{x_2(t)}{1+x_1^2(t)} + u_1(t)$$

$$x_2(t+1) = 0.9x_2(t)e^{-x_1^2(t)} + u_2(t)$$

$$y(t) = x(t)$$

Both outputs are followed by a transport delay of  $k = 6$  samples so the time-delay matrix:

$$D_k = \begin{bmatrix} z^{-6} & 0 \\ 0 & z^{-6} \end{bmatrix}$$

Models:

$$W_r = \begin{bmatrix} \frac{1}{1-z^{-1}} & 0 \\ 0 & \frac{1}{1-z^{-1}} \end{bmatrix}$$

reference

$$W_d = z^{-6} \begin{bmatrix} \frac{1}{1-z^{-1}} & 0 \\ 0 & \frac{1}{1-z^{-1}} \end{bmatrix}$$

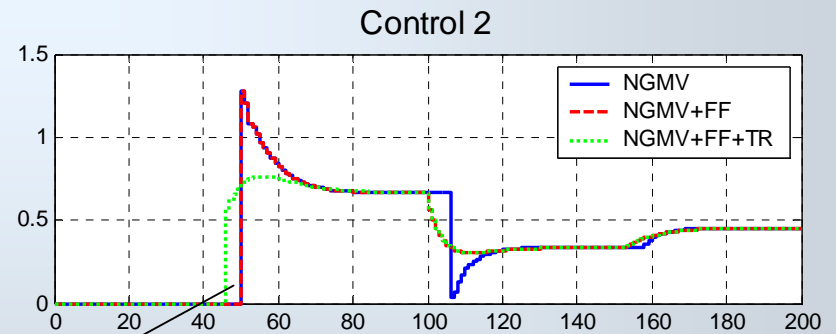
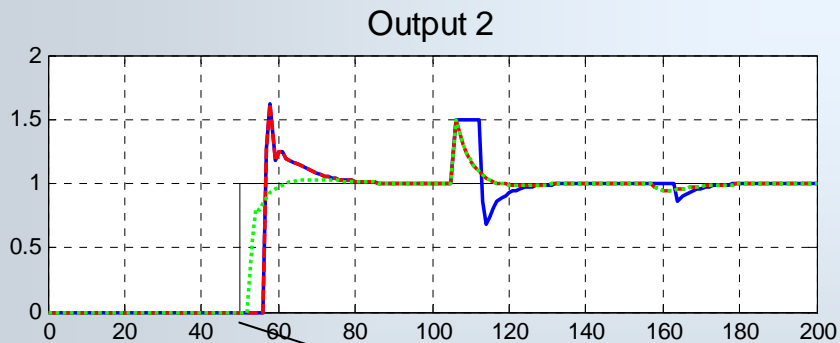
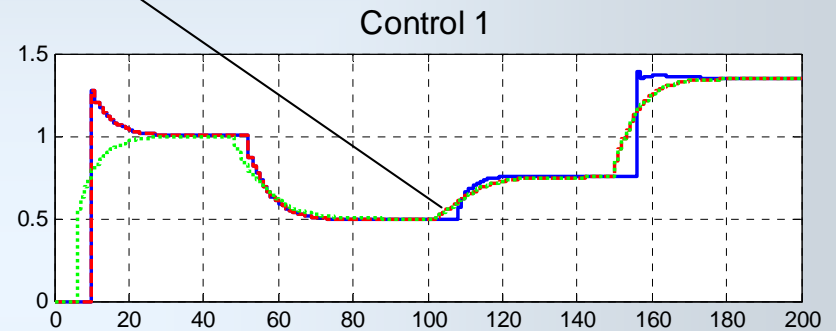
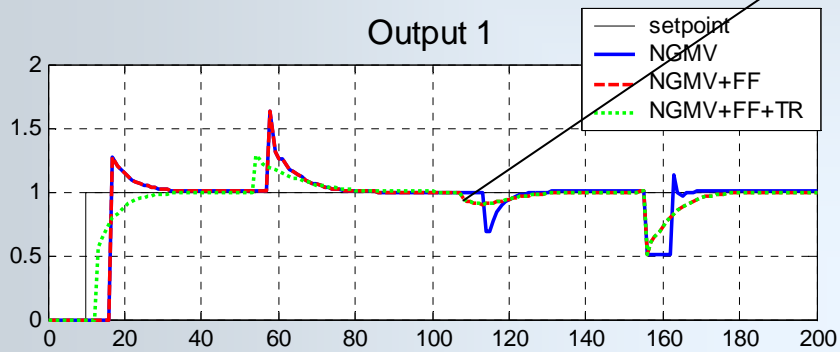
measurable  
disturbance

$$W_{d0} = \begin{bmatrix} \frac{0.1}{1-0.5z^{-1}} & 0 \\ 0 & \frac{0.1}{1-0.5z^{-1}} \end{bmatrix}$$

unmeasurable  
disturbance

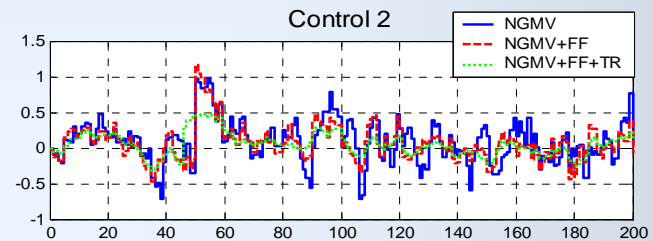
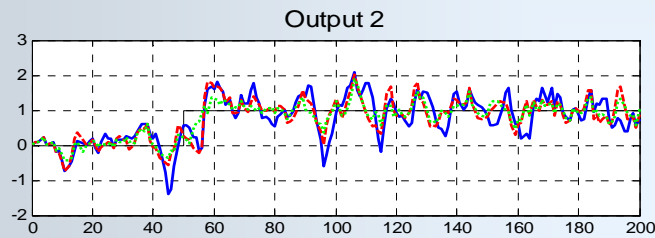
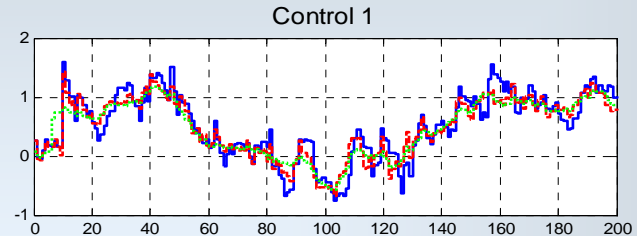
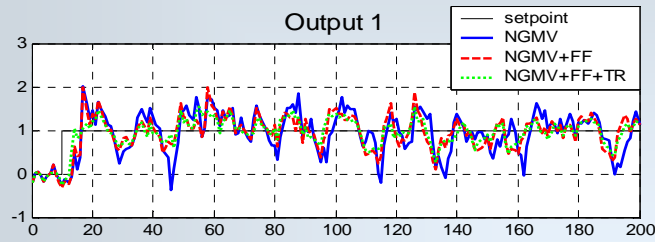
# Transient responses

Feedforward action



Future reference information incorporated

# Stochastic performance



Controller	Var[e]	Var[u]	Var[ $\phi_0$ ]
NGMV FB	0.61	8.43	2.45
NGMV FB+FF	0.51	8.41	0.82
NGMV FB+FF+TR	0.48	8.37	0.27

# Concluding Remarks

- State dependent models gives a useful structure for NL predictive controllers
- NGMV has much potential for development with multi - step predictive control an obvious development.
- Key to success in NL control is to show works and practical on real processes - why NGMV seems great potential.