MPC Workshop

- There exists a MPC Toolbox available through Mathworks
  - I do not use this!
- Thus far you have written all of your own MPC code
- This tutorial "workshop" is based on MATLAB code that I have written
  - It is definitely NOT commercially quality code
- Feel free to modify my code to fit your needes

B. Wayne Bequette

MPC Workshop

- Primary MPC files
  - QSSmpcNLPlant.m (QP, State Space MPC)
    - isim = 1 (DMC), isim = 2 (KF-based MPC)
    - iqp = 1 (unconstrained), iqp = 2 (QP solution)
  - KmatQSS.m (generates several matrices)
  - qSSmpccalc.m (calculates control moves)
  - planteqns.m (set of plant odes, deviation form – *can be nonlinear*)
- Driver files
  - r_FOmpc.m (first-order example)
    - FOodedev.m (planteqns = 'FOodedev')
  - r_VDV.m (linear van de vuuse example: Module 5)
  - r_quadlin.m (linear quadruple tank, Chapter 14)

First-order Example

- First-order Transfer Function

$$y(s) = g_p(s)u(s) + g_d(s)l(s)$$
$$= \frac{1}{2s+1}u(s) + \frac{1}{2s+1}l(s)$$

- State Space Plant (and model)

$$\dot{x} = -\frac{1}{2}x + \frac{1}{2}u + \frac{1}{2}l$$
$$y = x$$

Driving Programs: r_FOmpc.m

$$\dot{x} = -\frac{1}{2}x + \frac{1}{2}u + \frac{1}{2}d$$

$$y = x$$

```
% r_FOmpc.m
% 20 July 2011 - B.W. Bequette
%    First-order problem
% Illustrates the use of State Space MPC
% The plant equations are continuous ODES provided in FOodedev.m
% The plant equations are in deviation variable form
% The controller model is discretized
%
% ---------- model parameters ---------------------------------
%       (continuous time, first-order problem)
  am = [-1/2]         % single state (time constant = 2)
  bm = [1/2 1/2]      % first column manip, 2nd column disturbance
  cm = [1]
  dm = [0 0]
%
  ninputs = size(bm,2);     % includes disturbance input
  noutputs = size(cm,1);
  nstates = size(am,1);     % number of model states
  sysc_mod = ss(am,bm,cm,dm);
%
```

$$x_{k+1} = \Phi x_k + \Gamma u_k + \Gamma^d d_k$$

$$y_k = C x_k$$

```
% discretize the model
  delt = 0.2;               % sample time = 0.2
  sysd_mod = c2d(sysc_mod,delt);
  [phi_mod,gamma_stuff,cd_mod,dd_stuff] = ssdata(sysd_mod)
%
  gamma_mod  = gamma_stuff(:,1);   % first input is manipulated
  gammad_mod = gamma_stuff(:,2);   % second input is a disturbance

% ------------ plant parameters ------------------------------
  planteqns = 'FOodedev'
  cplant = [1];   % the only state is also the output
  nstates_p = 1; % one plant state
  parvec(1) = 2; % the parameter is a time constant with value = 2
```

planteqns = 'FOodedev' → Continuous plant

Controller Parameters

```
% ---------------- controller parameters -------------------
  p = 10;        % prediction horizon
  m = 1;         % control horizon
  ny = 1;        % number of measured outputs
  nu = 1;        % number of manipulated inputs
  nd = 1;        % number of actual disturbances
  nd_est = 1;    % number of estimated disturbances (used by KF)
  weightu = [0]; % weighting matrix for control action
  weighty = [1]; % weighting matrix for outputs
% -----------------constraints ----------------------------
  umin = [-1000];
  umax = [1000];
  dumin = [-1000];
  dumax = [1000];
% ------ Kalman Filter matrices ---------------------------
  Q  = 100*eye(nd_est,nd_est);  % state covariance
  R  = eye(ny);             % measurement noise covariance
%
```

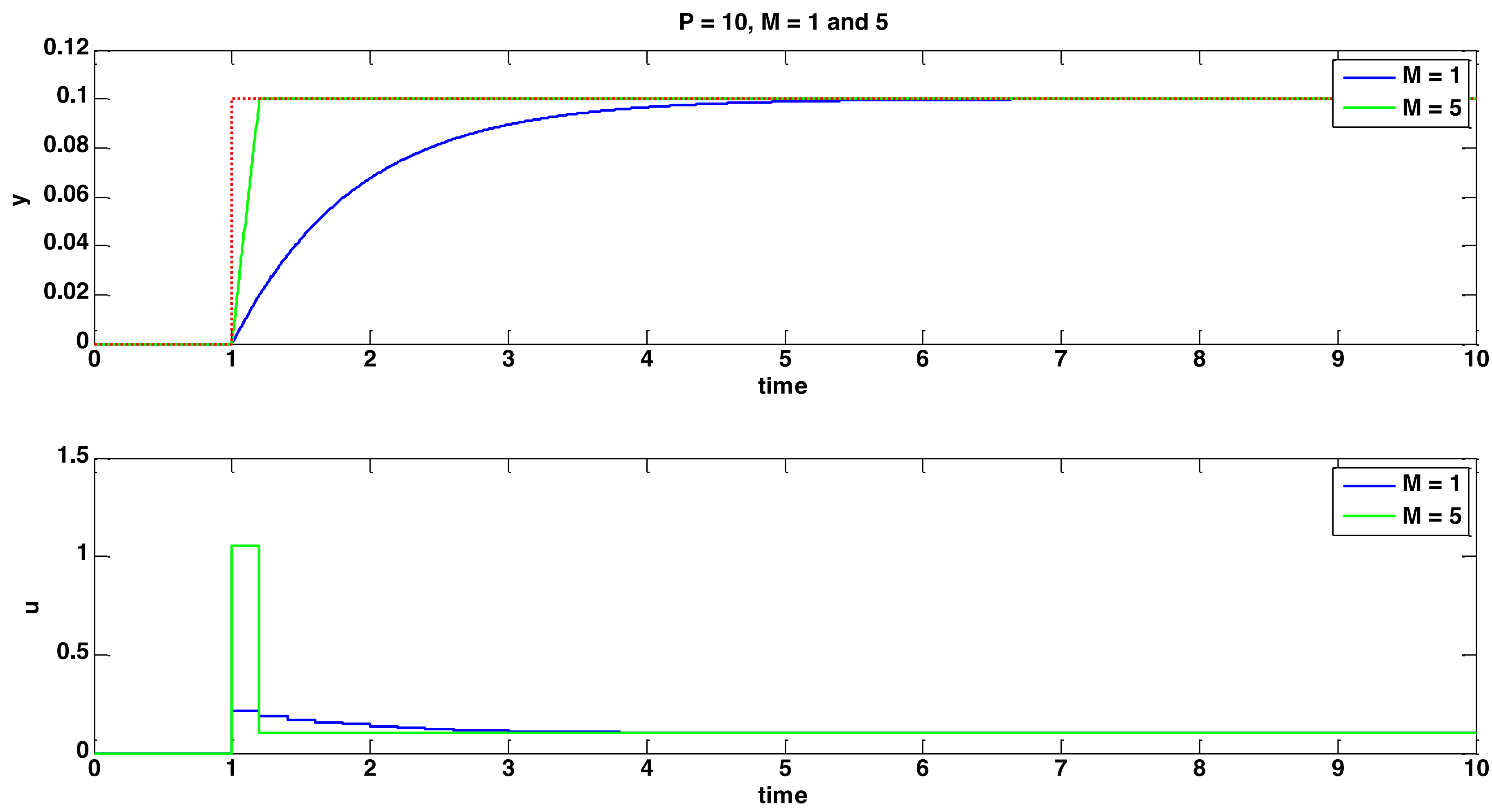
Simulation Parameters

```
% -------- simulation parameters ----------------------------
% first, setpoint change only (no disturbance)
  ysp = [0.1];    % setpoint change (from 0 vector); dimension ny
  timesp = 1;   % time of setpoint change
  dist = [0];   % magnitude of input disturbance; dimension nd
  timedis = 6;  % time of input disturbance
  tfinal = 10;
%
  isim = 1; % additive output disturbance
  iqp = 1;  % unconstrained solution
% noisemag = zeros(ny,1); % no noise
  noisemag = 0.0;      % no measurement noise
```

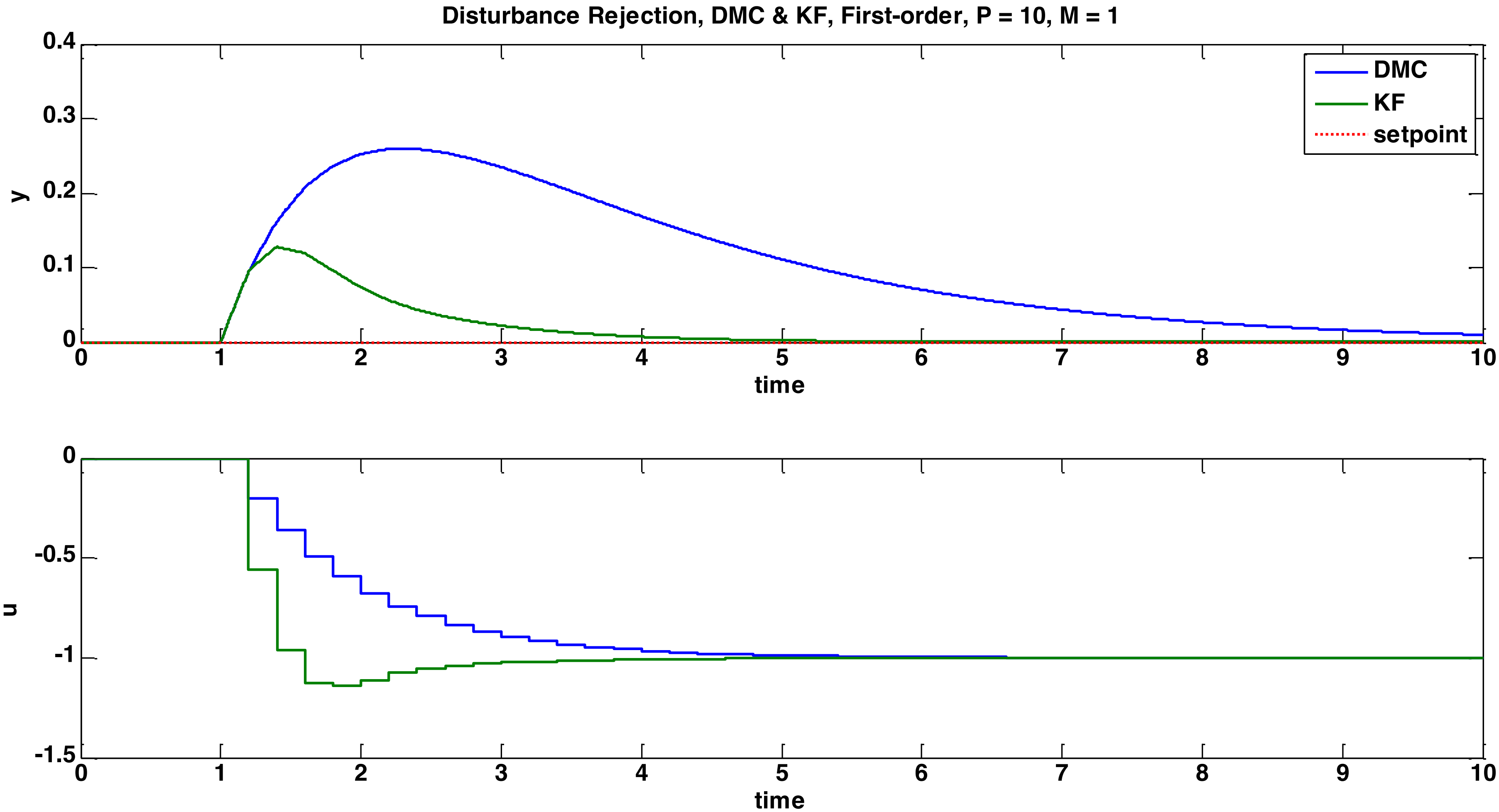
Plant Equations file

```
function xdot = FOodedev(t,x,flag,parvec,u,d)
%
% b.w. bequette - 20 July 2011
% First-order problem
% revised for explicit disturbance and deviation variable form
%
 time_constant = parvec(1);
%
 dxdt(1) = -(1/time_constant)*x(1) + u(1)/time_constant + d(1)/time_constant;
 xdot = dxdt(1);
```

Simulation Results, P = 10
Comparison of M = 1 and M = 5



Disturbance Results, P = 10, M = 1
Comparison of KF and DMC

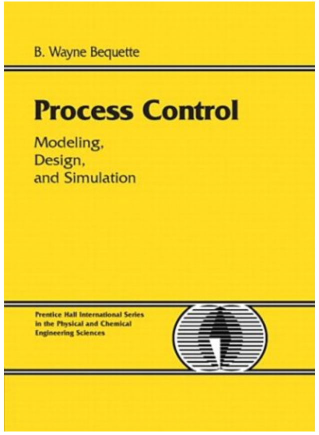


Linear Van de Vuuse Reactor (inverse response)

$$\dot{x} = \begin{bmatrix} -2.4048 & 0 \\ 0.833 & -2.2381 \end{bmatrix} x + \begin{bmatrix} 7 \\ -1.117 \end{bmatrix} u + \begin{bmatrix} 7 \\ -1.117 \end{bmatrix} l$$

$$y = \begin{bmatrix} 0 & 1 \end{bmatrix} x$$

Module 5



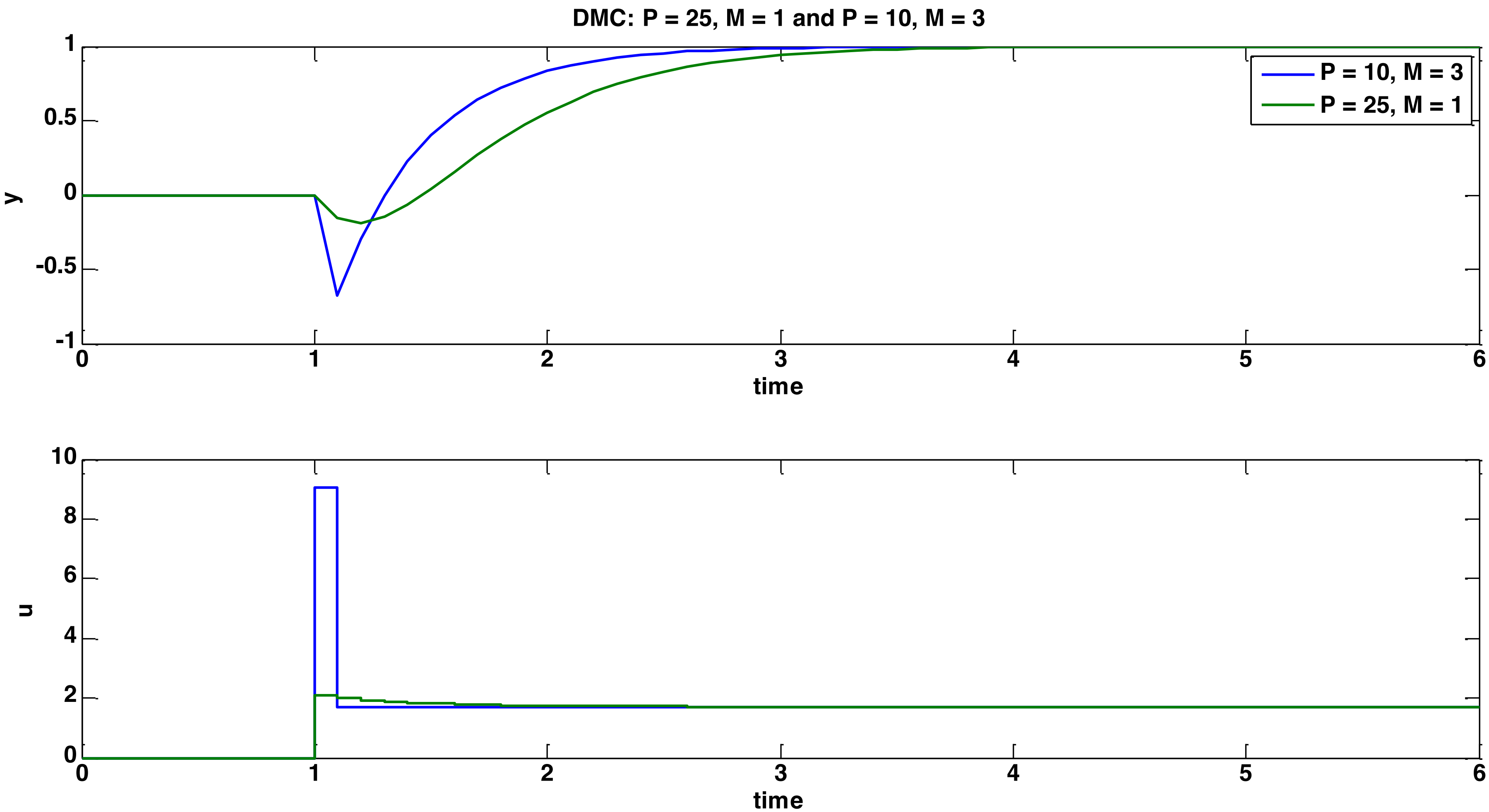
Linear Van de Vuuse Reactor (inverse response)

```
% r_VDV.m
% driving file for MPC simulations using the VDV reactor
% requires the following files:
%       QSSmpcNLPlant.m    main simulation file
%       qSSmpccalc.m       calculates the control move (solves
%                          either the unconsrained or constrained problem)
%       KmatQSS.m          generates matrices
%       linVDVode.m        linear continuous plant odes
% 22 July 2011 -- presentation at PASI Workshop
% 20 November 2011 -- revised with more comments for clarity and use
%                   in Fall 2011 MPC course
% linear van de vuuse reactor model (Module 5 from Bequette, 2003)
% run 1: p = 10, m = 3, wu = 0, unconstrained
% run 2: p = 25, m = 1, wu = 0, unconstrained
% run 3: p =  8, m = 1, wu = 0, unconstrained
% run 4: p =  7, m = 1, wu = 0, unconstrained (unstable)
% run 5: p =  7, m = 1, wu = 0, constrained (unstable, but attains a
%                              steady-state at the input constraint)
```

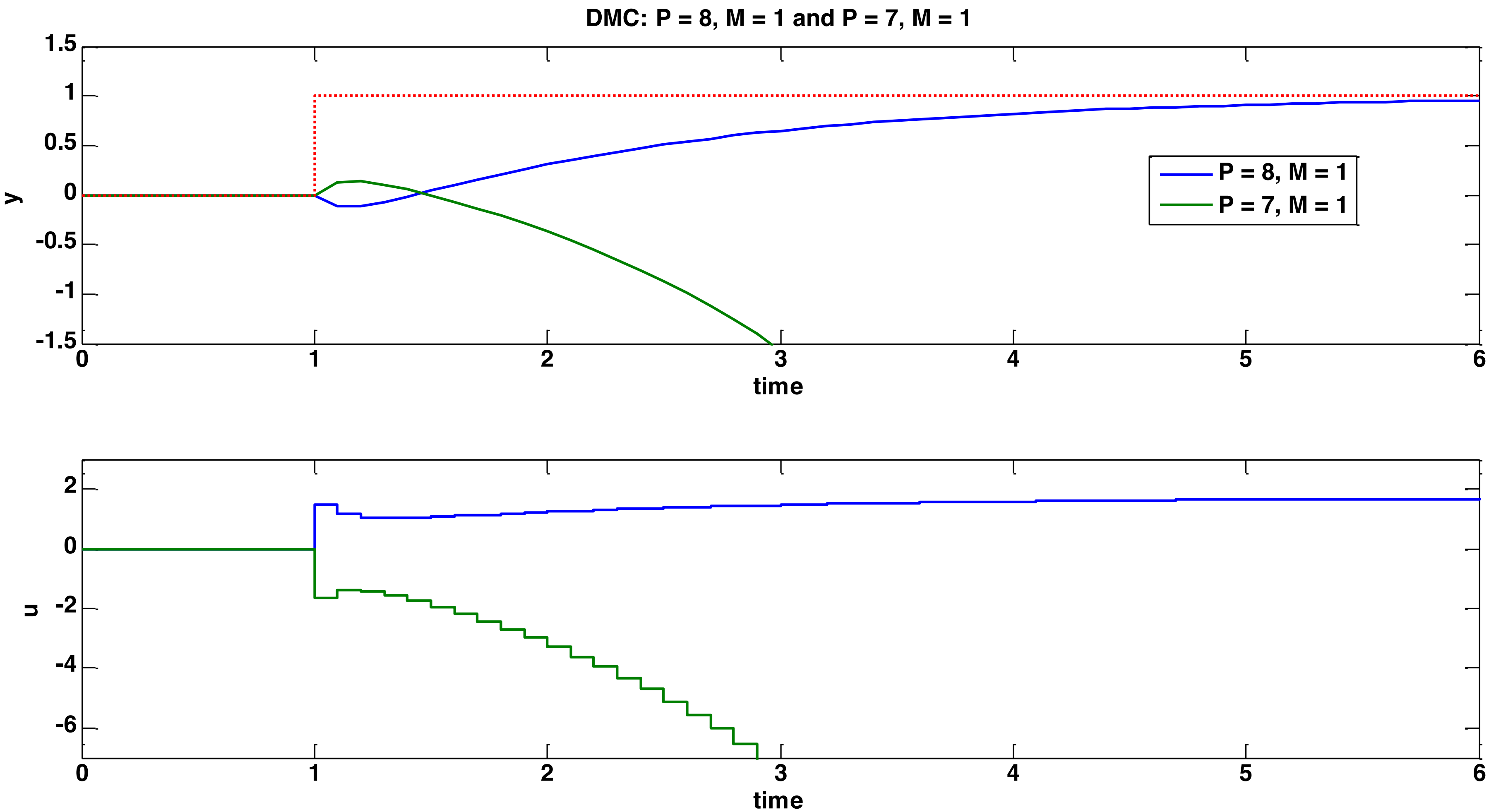
Linear Van de Vuuse (plant ODEs)

```
function xdot = linVDVode(t,x,flag,parvec,u,d)
%
% b.w. bequette - 22 July 2011
% linear VDV reactor model
  a = [-2.4048 0;0.8333 -2.2381];
  b = [7 7; -1.117 -1.117]; % 1 manip input, 1 dist
%
  xdot = a*x + b*[u(1);d(1)];
```

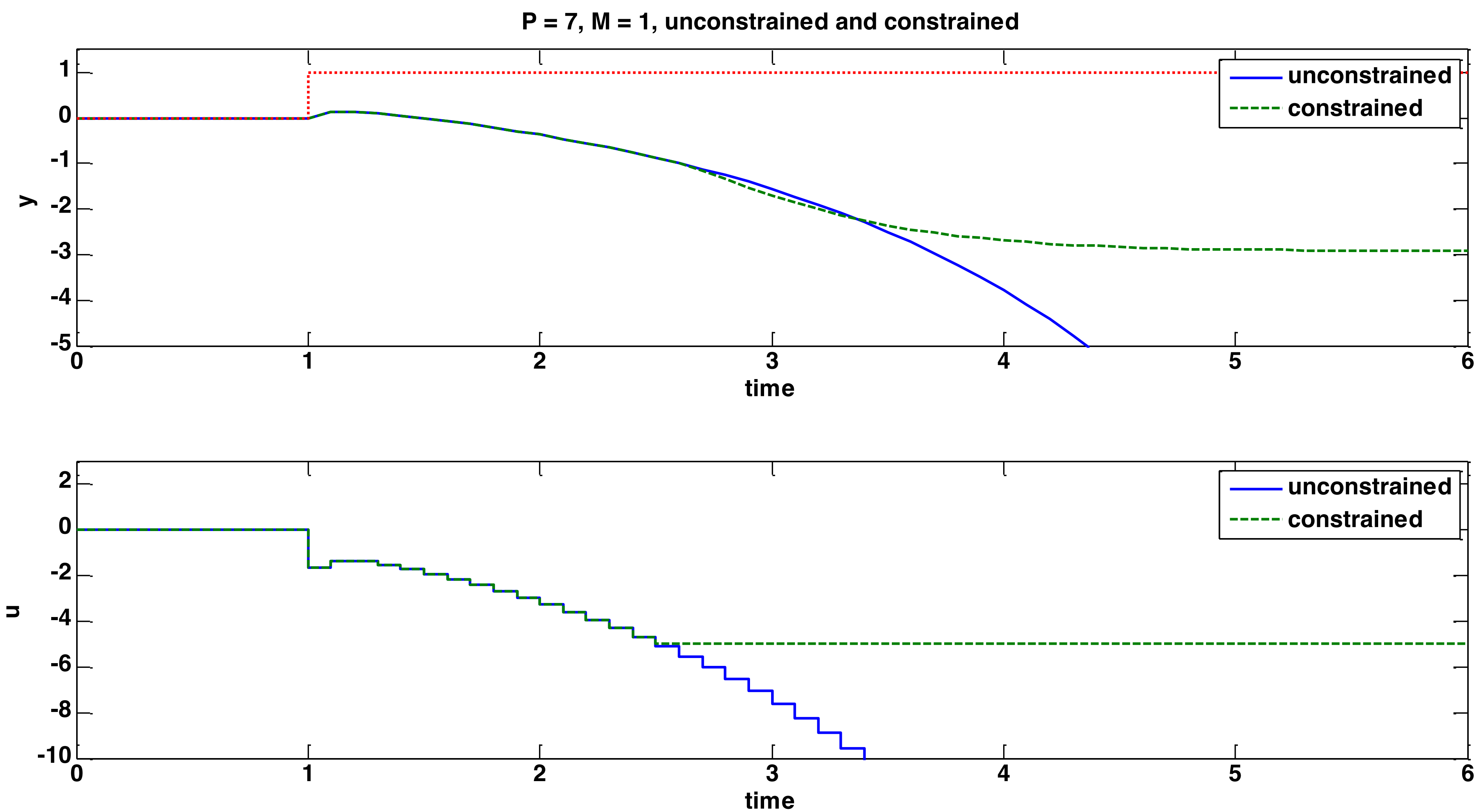
r_VDV
Comparison of (P=10,M=3) with (P=25,M=1)



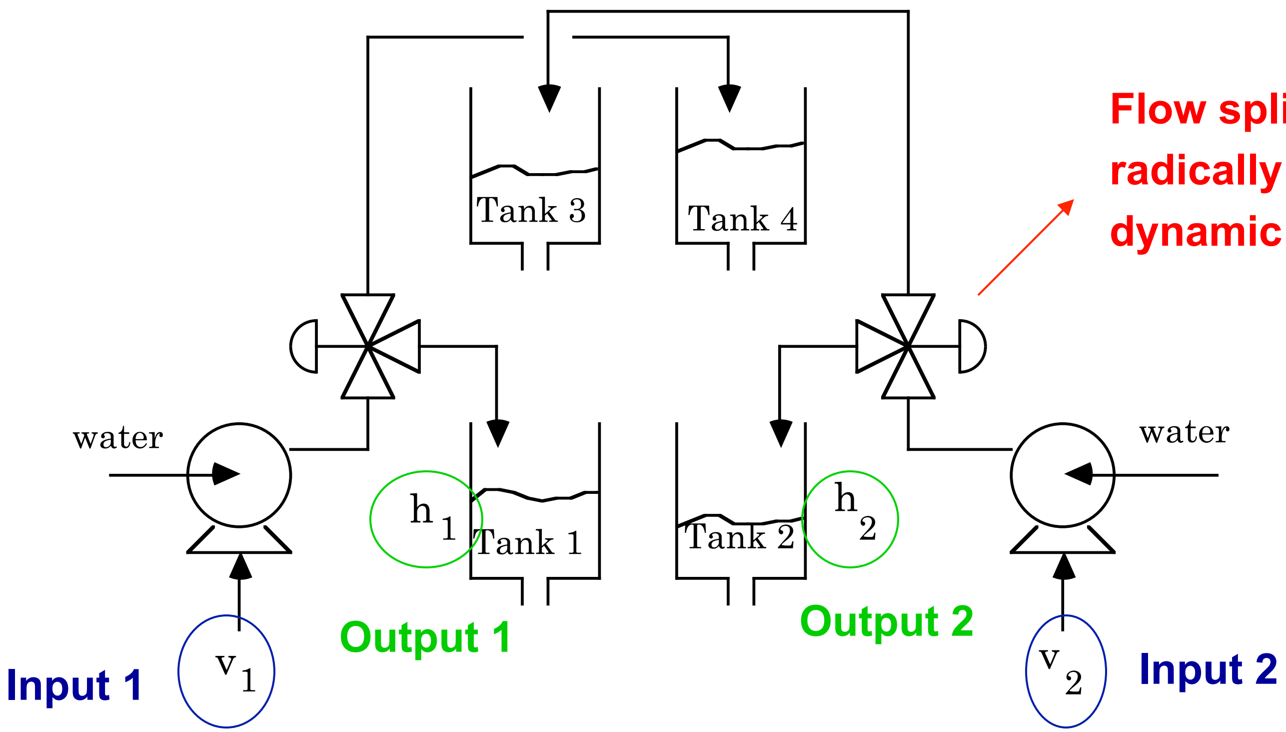
r_VDV
Comparison of (P=8,M=1) with (P=7,M=1)



r_VDV
Constrained vs. Unconstrained (P=7,M=1)

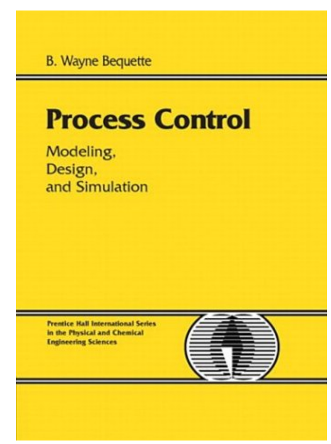


Quadruple Tank Problem: r_quadlin



Flow splits can radically change dynamic behavior

Chapter 14



Quadruple Tank Problem

MP Operating Point

$$G_1(s) = \begin{bmatrix} \dfrac{2.6}{62s+1} & \dfrac{1.5}{(23s+1)(62s+1)} \\ \dfrac{1.4}{(30s+1)(90s+1)} & \dfrac{2.8}{(90s+1)} \end{bmatrix}$$

NMP Operating Point

$$G_2(s) = \begin{bmatrix} \dfrac{1.5}{63s+1} & \dfrac{2.5}{(39s+1)(63s+1)} \\ \dfrac{2.5}{(56s+1)(91s+1)} & \dfrac{1.6}{(91s+1)} \end{bmatrix}$$

Quadruple Tank – plant ODEs

```
function xdot = odequadtanklin(t,x,flag,parvec,u,d)
%
% b.w. bequette - 18 Nov 09
% linear quadruple tank problem
% if parvec(1) == 1, then operating point 1
% disturbances are perturbations to manipulated inputs
%
% parameters (continuous time, state space)
%
 A1 = 28;    A3 = A1; % cross-sectional area
 A2 = 32;    A4 = A2; % cross-sectional area
 a1 = 0.071; a3 = a1;
 a2 = 0.057; a4 = a2;
 beta = 0.5;
 grav = 981;
 if parvec(1) == 1;
% parameters for operating point 1 (minimum phase)
   k1 = 3.33;  k2 = 3.35;
   gamma1 = 0.7;  gamma2 = 0.6;
   h1s = 12.4; h2s = 12.7;
   h3s = 1.8;  h4s = 1.4;
   v1s = 3;    v2s = 3;
```

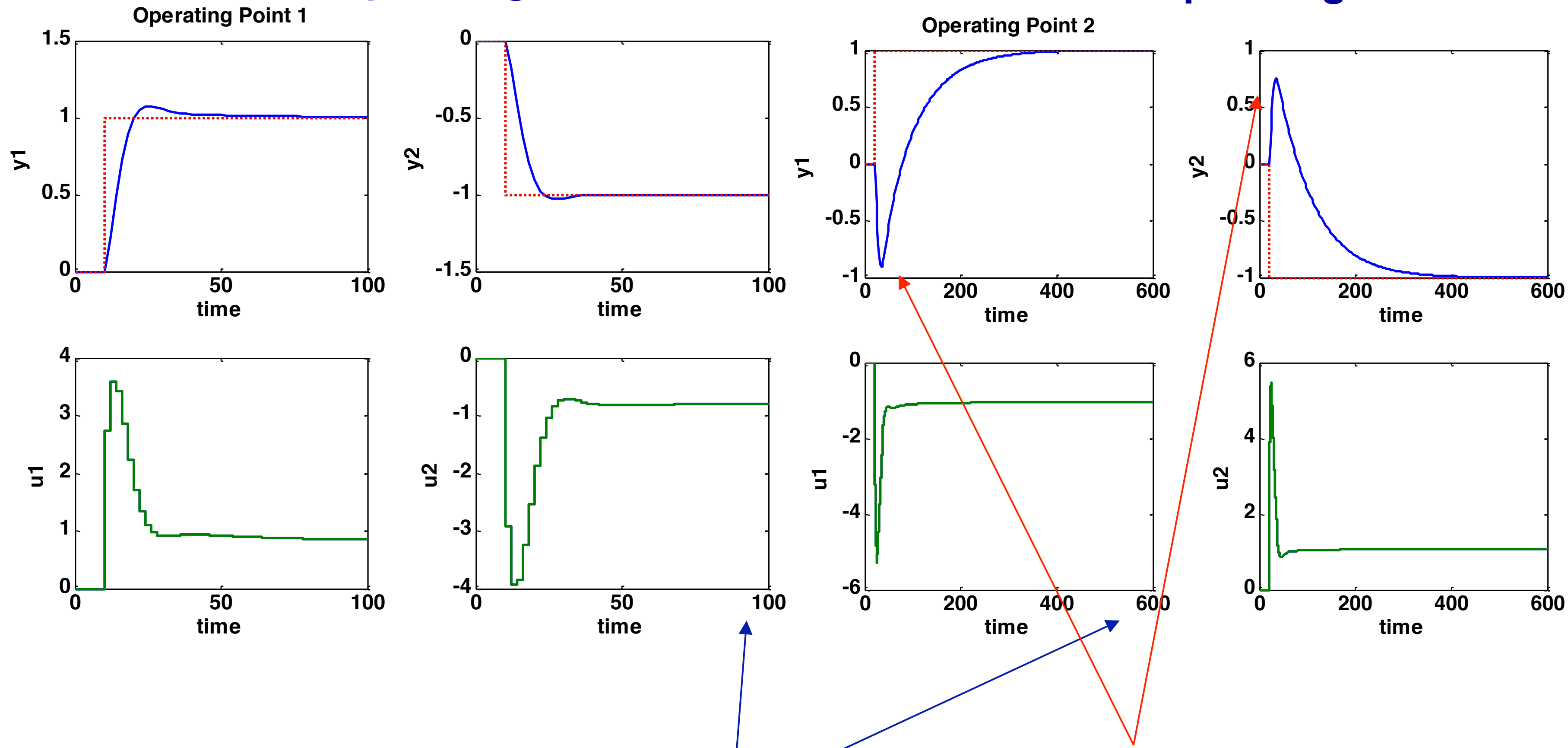
```
 else
% operating point 2 (nonminimum phase) -- these are the first
   simulations
  k1 = 3.14;  k2 = 3.29;
  gamma1 = 0.43;  gamma2 = 0.34;
  h1s = 12.6; h2s = 13.0;
  h3s = 4.8;  h4s = 4.9;
  v1s = 3.15;    v2s = 3.15;
 end
% time constants
 tau1 = (A1/a1)*sqrt(2*h1s/grav);
 tau2 = (A2/a2)*sqrt(2*h2s/grav);
 tau3 = (A3/a3)*sqrt(2*h3s/grav);
 tau4 = (A4/a4)*sqrt(2*h4s/grav);
% continuous state space model
 aplant = [-1/tau1,0,A3/(A1*tau3),0;0,-1/tau2,0,A4/(A2*tau4);0,0,-1/
   tau3,0;0,0,0,-1/tau4];
 bplant = [gamma1*k1/A1,0;0,gamma2*k2/A2;0,(1-gamma2)*k2/A3;
   (1-gamma1)*k1/A4,0];
%  the states are the tank heights, in deviation variables
%
 xdot = aplant*x + bplant*u + bplant*d;
```

r_quadlin
Comparison of NMP and MP Operating Points

Setpoint Responses

MP Operating Point

NMP Operating Point



Note difference in time scales

“Wrong way” behavior