Benefits (and Costs) of Model Predictive Control

James B. Rawlings
Department of Chemical and Biological Engineering
University of Wisconsin-Madison
Madison, WI 53706

June 20, 2013

Introduction

Control with PID controllers usually suffices for single-input single-output (SISO) systems with "normal" dynamics (low order, minimum phase) and without active constraints. In the following sections, we violate each of these conditions in turn, and show disadvantages of PID control and advantages of using model predictive control (MPC).

Difficult dynamics

As we have already seen in the course, nonminimum phase systems (time delays, right half-plane zeros), and high order systems limit the gain that can be used in simple feedback controllers without causing instability or excessive control action.[1] Predictive control does not suffer from this disadvantage because the model-based forecasting enables the controller to account for the inherent limitations in the process response.

Consider the following first order system with time delay shown in Figure 1.

$$g(s) = \frac{k}{\tau s + 1} e^{-\theta s}, \qquad k = 1, \tau = 1, \theta = 5$$

The large time delay compared to time constant severely limits the achievable closed-loop performance of this system.[2] Consider a unit step change in setpoint, y_{sp}, at $t = 0$. What would we expect an "ideal" controller to do in this situation. The physically intuitive answer is to take immediately the control action required to drive the system *without delay* to the desired setpoint and then ignore the large tracking error while

[1] Recall the Bode stability criterion for proportional control of a system with time delay.

[2] This example is for illustration. Recall you may want to redesign this process to remove this large delay, rather than force the controller to contend with it.

waiting for the system's time delay to pass; then the output should rise quickly to the setpoint. The predictive controller has this inherent time delay compensation because the forecast enables it to see that it should wait for the time delay to pass before taking further control action. As we see in Figure 2, the predictive controller behaves precisely as we expect an ideal controller to behave. It takes large control action early to achieve the setpoint and then does nothing further until the time delay passes. Without a model, the PID controller is blind to the delay and must take control action based on current tracking error and integral of tracking error,

$$u(t) = k_c\left(e(t) + \frac{1}{\tau_I}\int_0^t e(t')dt'\right), \qquad e = y_{\text{sp}} - y$$

Even if we choose k_c and τ_I carefully, there is only so much that we can expect. Figure 3 shows the response of a reasonably well tuned PID controller. Notice that the control action does not have the same structure as the MPC controller. Control action starts small and increases as the integral of the tracking error increases. After the time delay passes, the system response is still sluggish and does not reach setpoint until $t = 10$. A natural reaction of the control engineer might be to tune the PID controller more aggressively, i.e. increase the gain, in order to speed up the closed-loop response. Consider doubling the gain, k_c, and doubling τ_I so the integral term remains the same size. The results are shown in Figure 4. Notice that this retuning has not solved the problem. Can you find any values of the PID tuning parameters that gives performance similar to the MPC controller shown in Figure 2.

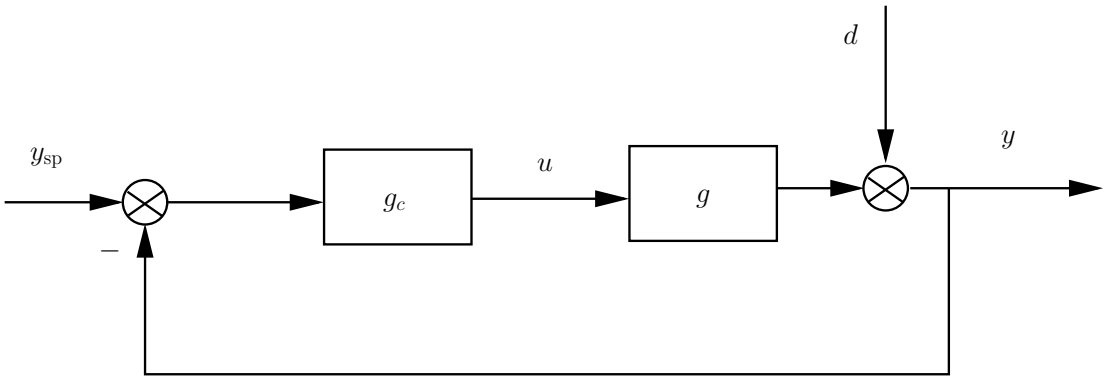

Figure 1: Feedback control system with output disturbance, d, and setpoint, y_{sp}.

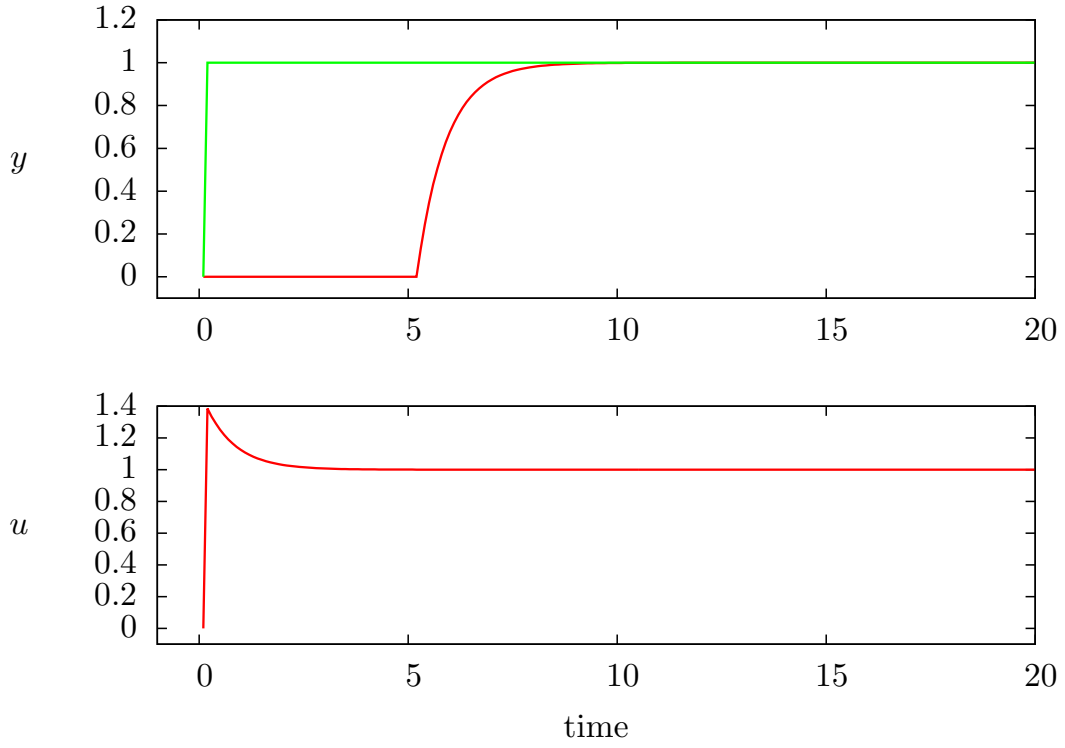

Figure 2: MPC applied to system with large delay, $g(s) = 1/(s+1)e^{-5s}$, $Q = 1, R = 1, S = 0$; unit step setpoint change at $t = 0$.

Constraints

Constraints are present in every physical process: valves saturate full open and full closed, safety considerations stipulate constraints on process temperatures and pressures, and the like. Even if the process is designed so that the constraints are not active at the designed steady state, market conditions and process operations change over time so that plants may be operated against constraints in some normal future operating condition. Model predictive control is particularly useful for controlling constrained systems, because the constraints can be added explicitly to the on-line optimization problem.

First consider a simple SISO system, first order without delay,

$$g(s) = \frac{k}{\tau s + 1}, \qquad k = 1, \tau = 1$$

As we know, a PID controller has no trouble controlling this simple system. Figure 5

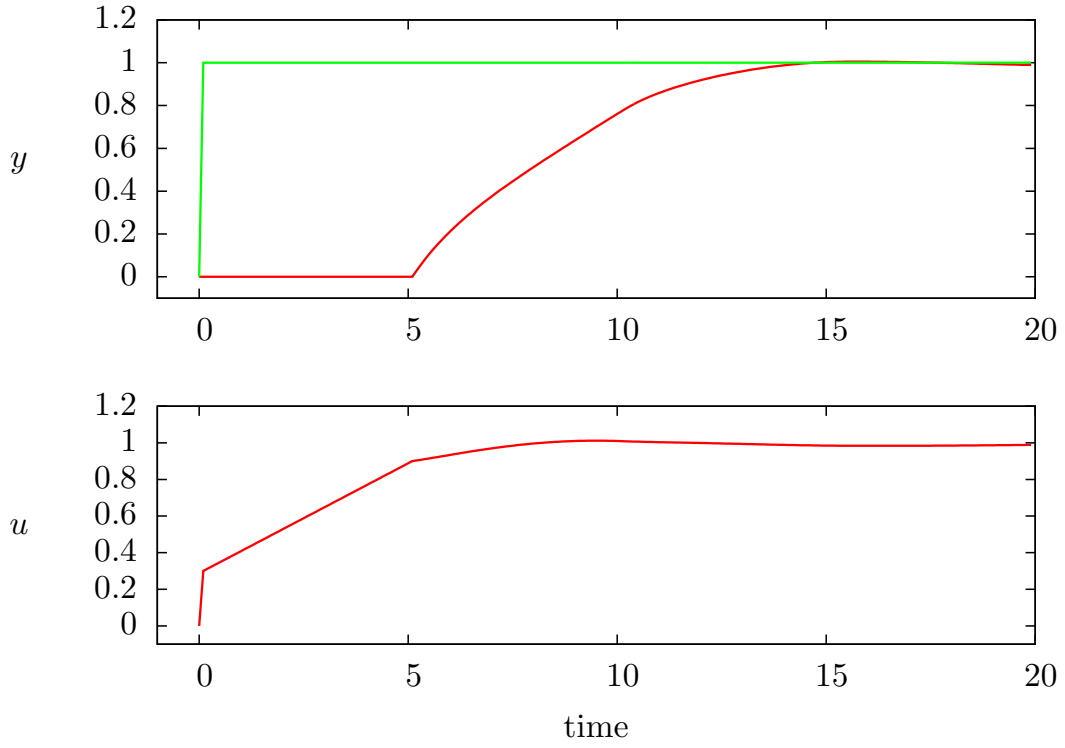

Figure 3: PID applied to system with large delay, $g(s) = 1/(s+1)e^{-5s}$, $k_c = 0.3, \tau_I = 2.5$; unit step setpoint change at $t = 0$.

shows the nice closed-loop response for a unit setpoint change. Next let's consider the response to a disturbance. Figure 6 shows the response to a pulse disturbance of magnitude 2 at the process output that lasts until $t = 10$. Again we see good disturbance rejection achieved by the PID controller. The controller quickly moves to $u = -2$ to offset the disturbance and bring y quickly back to zero, and then after $t = 10$ moves back to $u = 0$ after the disturbance goes to zero and again quickly returns y to zero.

Now let's assume that the control valve saturates full open at $u = 1$ and full closed at $u = -1$.

$$u_{\min} \le u(t) \le u_{\max}, \qquad u_{\min} = -1, \quad u_{\max} = 1, \tag{1}$$

Given the same pulse disturbance, now the controller saturates the valve at full closed and there is steady-state tracking error until $t = 10$ and the disturbance goes to zero as shown in Figure 7. That behavior is as expected; the disturbance is simply too large for *any* controller given the process constraints. But then a strange thing happens after $t = 10$. Notice the valve stays full closed until almost $t = 20$ even though the controller

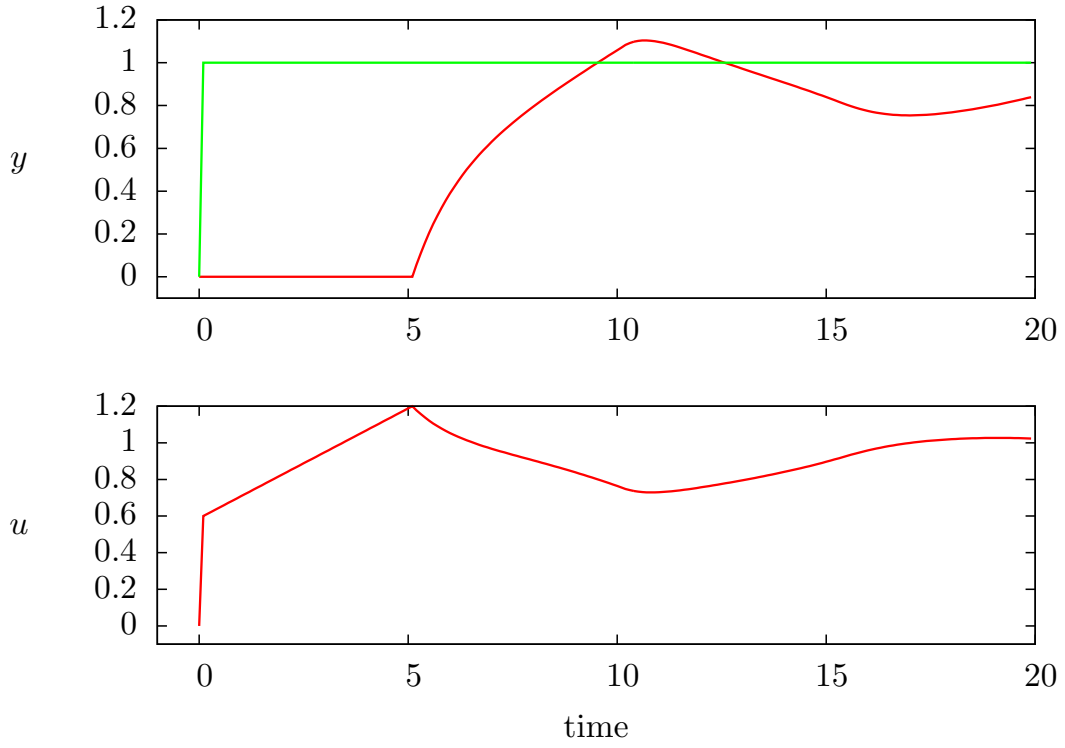

Figure 4: PID applied to system with large delay, $g(s) = 1/(s+1)e^{-5s}$, $k_c = 0.6, \tau_I = 5.0$; unit step setpoint change at $t = 0$.

could bring the process back to setpoint by just setting $u = 0$. This phenomenon is known as *integrator (or reset) windup*.[3] Can you explain why the integrator is the culprit in this windup phenomenon? We also know that we can easily prevent the windup problem in this SISO case. Can you explain how to modify this PID controller to avoid windup? But the anti-windup strategies are not clear in complex multi-variable problems when multiple active constraints are possible.

In contrast, consider the MPC controller behavior. with the same constraints as before, Equation 1. In this case, the MPC controller performance is shown in Figure 8. Notice that the MPC controller quickly moves the output back to setpoint at $t = 10$ after the disturbance goes to zero. There is no windup problem in an MPC controller. One does not need to design ad-hoc anti-windup strategies. The influence of constraints is forecast correctly and handled naturally by an MPC controller. That is one of the

[3]See (Ogunnaike and Ray, 1994, p. 585) for further discussion of windup.

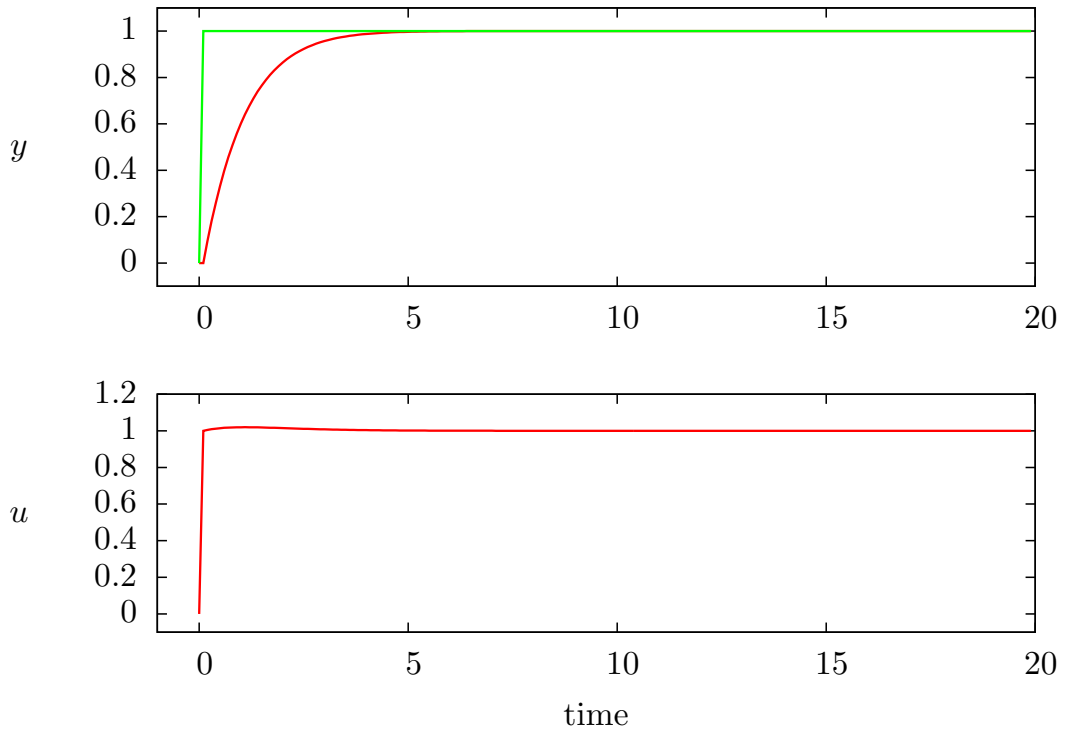

Figure 5: PID applied to unconstrained first order system, $g(s) = 1/(s+1), k_c = 1/k, \tau_I = \tau$. Setpoint tracking.

truly attractive features of MPC for industrial practitioners.

Multivariable interactions

The final advantage of MPC comes to light if we examine strongly interacting multi-input multi-output (MIMO) systems, which we know causes problems for multi-loop SISO PID controllers. Let's review the example we studied from Ogunnaike and Ray (1994, pp. 757–758). We have a two-input, two-output process with the following transfer function

$$G(s) = \begin{bmatrix} \frac{2}{10s+1} & \frac{2}{s+1} \\ \frac{1}{s+1} & \frac{-4}{10s+1} \end{bmatrix}$$

with RGA given by

$$\Lambda = \begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix}$$

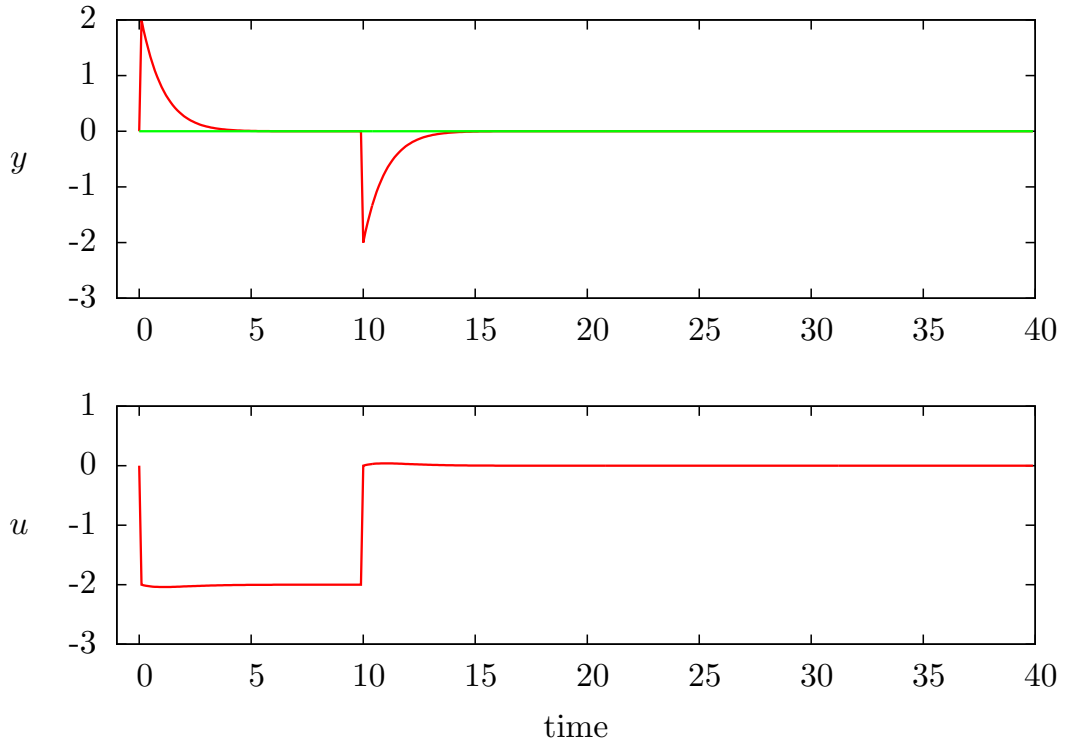

Figure 6: PID applied to unconstrained first order system, $g(s) = 1/(s+1), k_c = 1/k, \tau_I = \tau$. Disturbance rejection.

The RGA indicates that pairing u_1–y_1, u_2–y_2 is best considering the steady-state interactions. Figure 9 shows how two SISO PID controllers manage a unit setpoint change in the first output. A similar figure is shown in Ogunnaike and Ray with slightly different controller tuning parameters.

However, Ogunnaike and Ray also show that this pairing is not attractive from the dynamic perspective. The much slower u_1 input is driving y_1 ($\tau_{11} = 10$) when a ten times faster input, u_2, could have been used ($\tau_{12} = 1$). Similarly, the slow input, u_2, drives y_2 ($\tau_{22} = 10$) when the first input is ten times faster ($\tau_{21} = 1$). Dynamic considerations would favor switching to the pairing u_2–y_1, u_1–y_2. Figure 10 shows the results for the same setpoint change with two SISO PID controllers using this new pairing. Notice that, for this case, dynamics carry the day and the pairing based on speed of response exhibits better setpoint tracking than the pairing based on steady-state interactions.

These points are all clear for this example, but they will not be so clear if we start to move the two time constants closer to each other and the dynamic separation

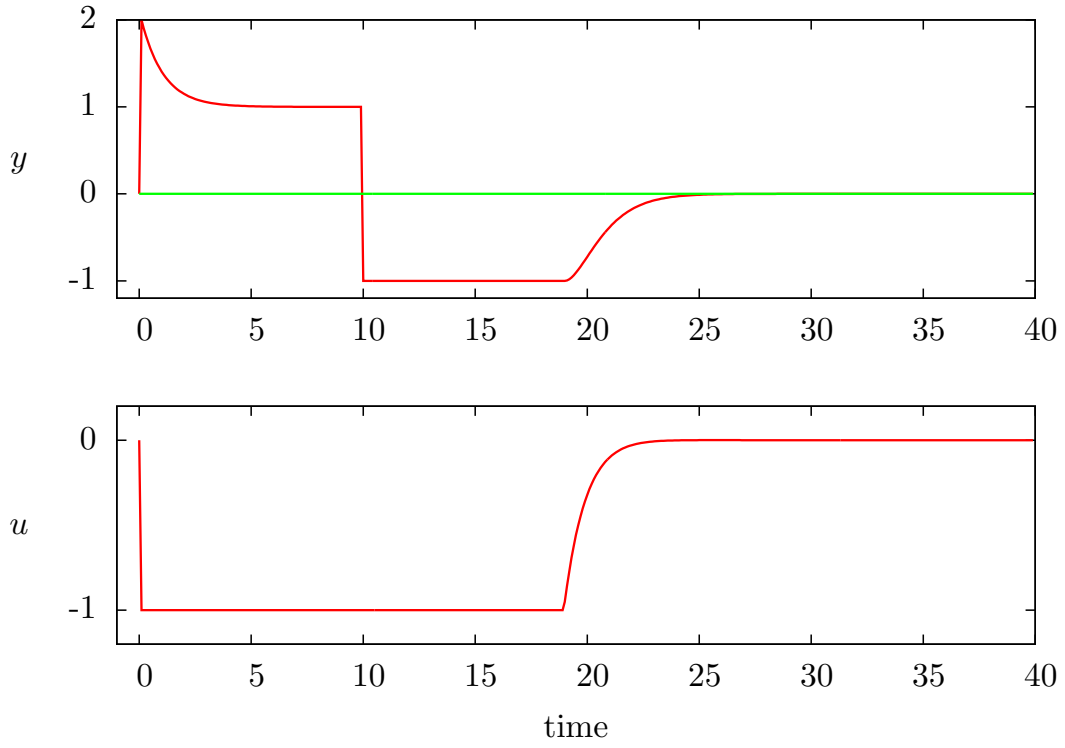

Figure 7: PID applied to *constrained* first order system, $g(s) = 1/(s+1), k_c = 1/k, \tau_I = \tau, -1 \leq u \leq 1$. Disturbance rejection; integrator windup after $t = 10$.

of time scales is not so large. Let's explore what happens if we refuse to become involved in choosing pairings and allow a multi-variable MPC controller to use its model to sort things out automatically. Figure 11 shows the results for an MPC controller given the same setpoint change. Notice that the performance is essentially deadbeat (i.e. immediate) in moving output y_1 to its new setpoint while not disturbing output y_2. In other words, the MPC controller uses its model to forecast exactly what input trajectories in both inputs are required to track exactly the setpoint in both outputs. These inputs are also shown in Figure 11. Interactions are not an issue because the model forecast accounts for them. Performance is as good or better than the best performance we can achieve using PID with the best tuning constants and the best pairing. No significant tuning effort for the MPC controller is required to achieve this good nominal performance. Figure 11, for example, uses $Q = I, S = 0$ for deadbeat response. As we know, we will have to detune this controller (increase S to use less input) if we want it to accommodate sensor noise, and mismatch between the plant and

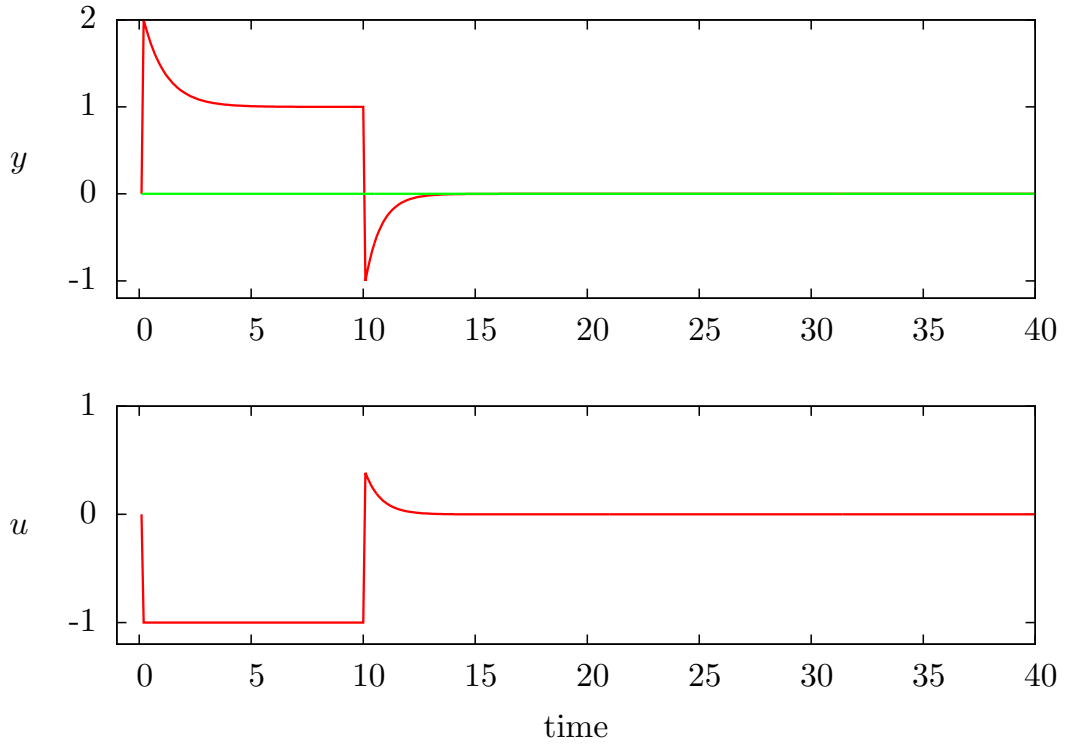

Figure 8: MPC applied to *constrained* first order system, $g(s) = 1/(s+1), Q = 1, R = 1, S = 0, N = 10, -1 \leq u \leq 1$. Disturbance rejection; no integrator windup after $t = 10$.

the model.

The effect of increasing the S penalty is shown in Figure 12 The S penalty affects the optimization tradeoff between the output tracking error and the manipulation of the input. Notice the output response has slowed down and is no longer deadbeat as in Figure 11, and that the manipulated input is moving less aggressively in Figure 12 compared to Figure 11.

And MPC offers yet more flexibility to shape this tradeoff between output tracking error and manipulated input. Let us now assume that the inputs are scaled so that valve saturation occurs at $u = \pm 1$. We see that the second input in Figure 12 violates this constraint. A PID controller would clip this signal and saturate the valve. But the MPC controller can be told about these constraints and they can be treated naturally by the on-line optimization. The results are shown in Figure 13. As we see the output tracking error is not seriously affected while the MPC controller deduces what changes to make in the input trajectory to account for the valve saturation.

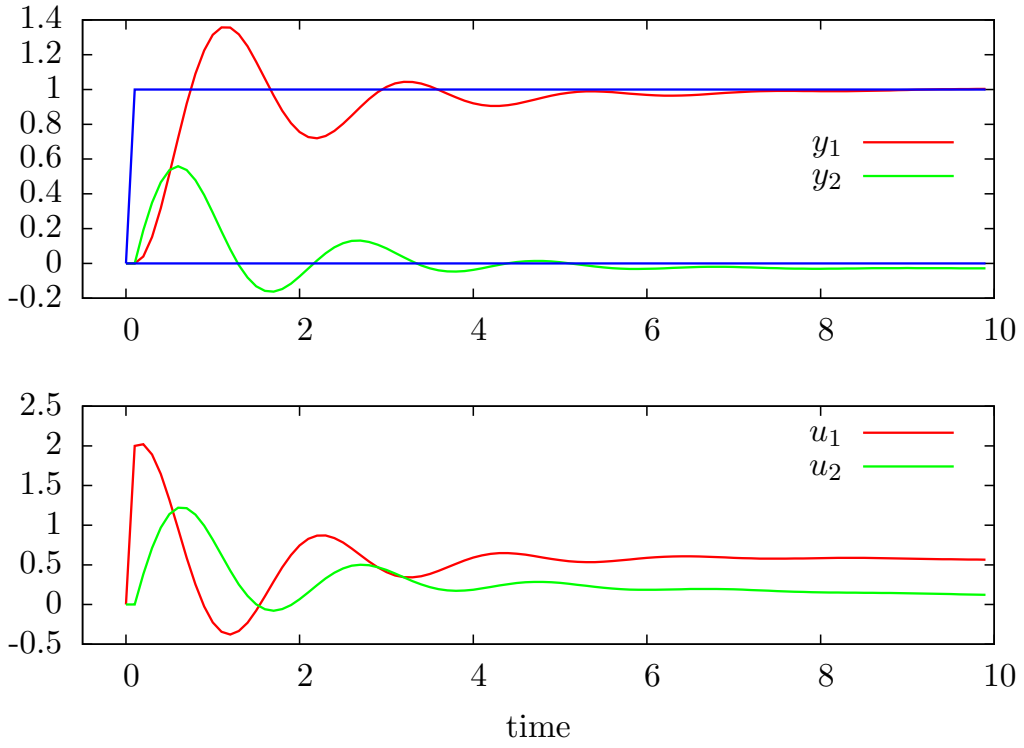

Figure 9: PID control with steady-state pairing (u_1–y_1, u_2–y_2), $k_{c1} = 2, \tau_{I1} = 2, k_{c2} = -2, \tau_{I2} = 3$; setpoint change in y_1.

Another practically motivated method for shaping the tradeoff between tracking and control action is to constrain rather than penalize the rate of change of the input. Practitioners or operators may be comfortable with some rates at which the valves may be moved between each controller execution, but may not know how to translate that speed into an appropriate S penalty as in Figure 13 without extensive simulation. In these situations constraining rather than (or in addition to) penalizing the rate of change of the input is simpler to implement. Assume the initial portion of the input manipulation in Figure 13 is unacceptable to the process operators, and they would be more comfortable if u changed not more than 0.1 in each sample,

$$-0.1 \leq \Delta u \leq 0.1$$

Adding this constraint to the MPC problem produces the results in Figure 14. As we can see in Figure 14, the input now slows down significantly in the early portion of the setpoint change. The rate of change constraint becomes binding in both inputs. The

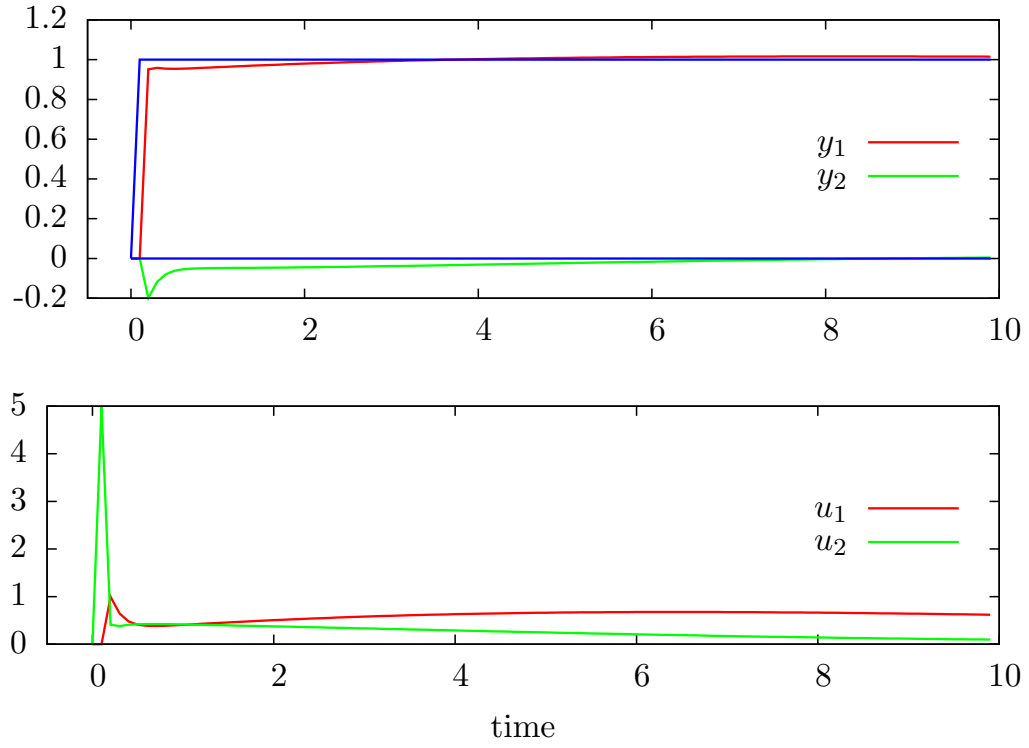

Figure 10: PID control with dynamic pairing (u_2–y_1, u_1–y_2), $k_{c1} = 5, \tau_{I1} = 2, k_{c2} = 5, \tau_{I2} = 3$; setpoint change in y_1.

slower input manipulation in turn affects the speed of response in the output. This kind of simulation allows one to see directly what penalty must be paid in the speed of output tracking if one wishes to impose this rate constraint on the input.

Conclusion

In this handout we have seen that MPC offers significant advantages compared to PID control in the following situations

1. Difficult dynamics: time delays, right half plane zeros, high order systems.
2. Tightly constrained systems: valve saturation, rate of change constraints, avoidance of integrator windup.
3. Strongly interacting multi-variable systems. No need for pairing, and true multi-

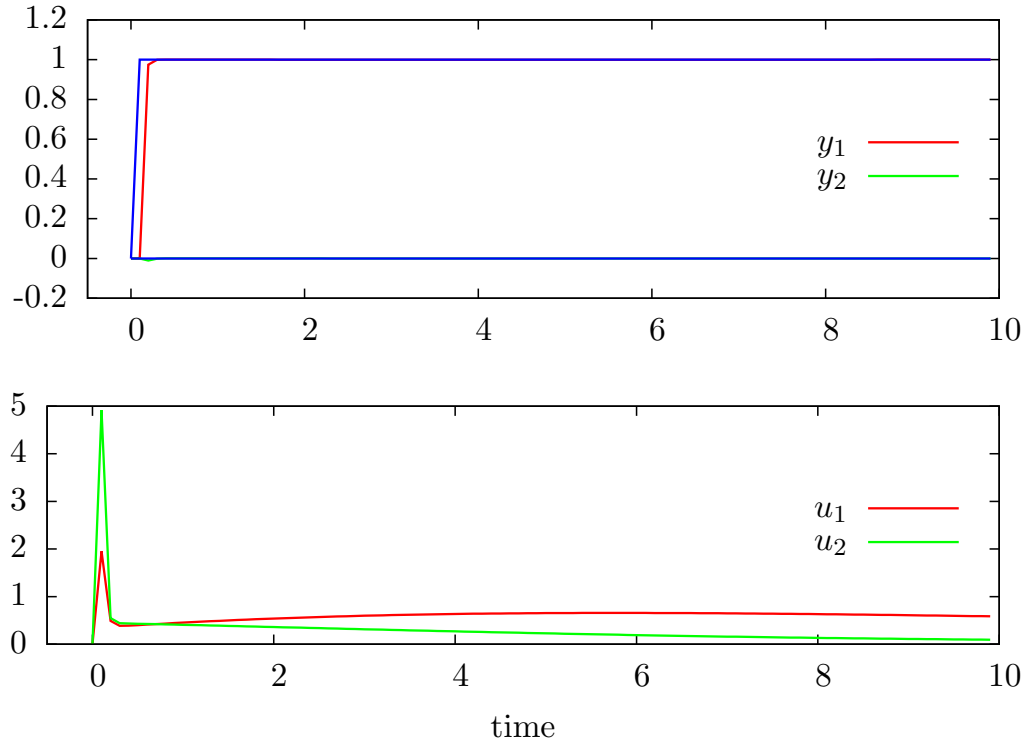

Figure 11: Multi-variable MPC control, $Q_y = I, S = 0$. Setpoint change.

variable control.

I would be remiss if I left you with the impression that MPC solves all control problems. The biggest disadvantage of MPC, or any model-based control system, is the significant effort required to obtain a process model and tune the controller to handle a variety of plant conditions without human intervention and constant re-tuning and maintenance. Given the successful implementation of MPC in the process industries, however, we can conclude that this modeling effort can pay off in improved profitability, product quality and safety. The decision about when to implement MPC versus simple, well-tuned PID controllers requires an assessment of the achievable process improvements versus the time and expense of the modeling effort.

Finally, nothing restricts the ideas of MPC to *linear* models. Practitioners are now pursuing the implementation of MPC based on nonlinear, fundamental chemical process models. The on-line optimization becomes significantly more complex, but today's computational capabilities are already up to the task of implementation for reasonably

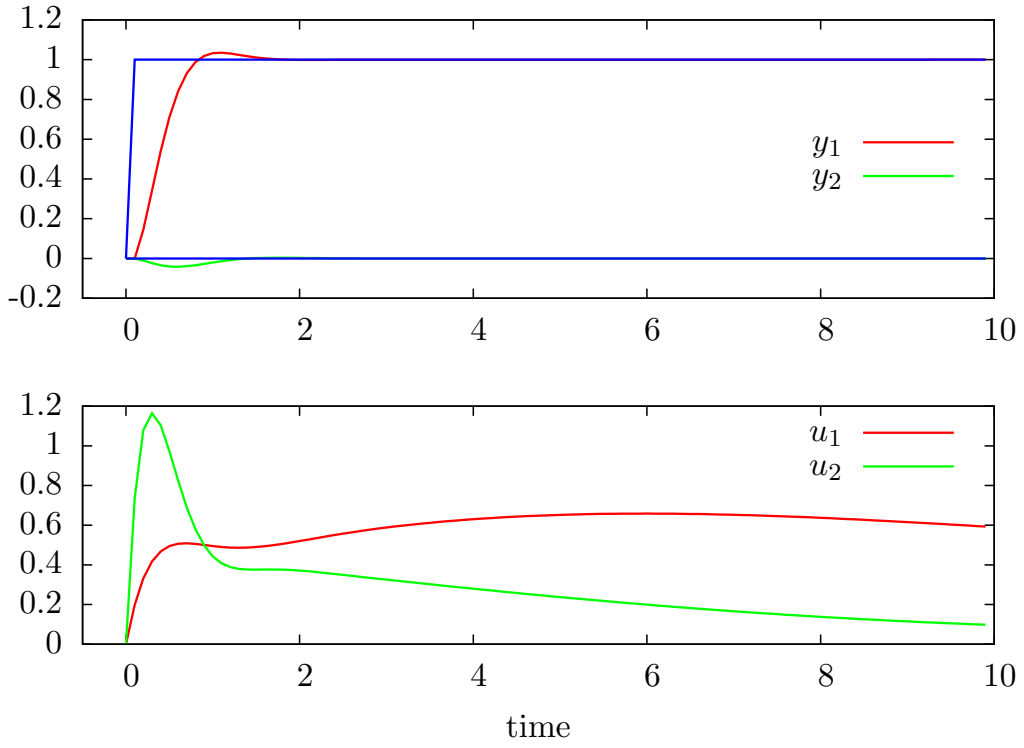

Figure 12: Multi-variable infinite horizon MPC control, $Q_y = I, S = I$. Setpoint change.

sized process units. The bottlenecks at this point are, again, obtaining nonlinear process models from first principles and operating data, and algorithms for reliably solving the nonlinear MPC optimization.

References

B. A. Ogunnaike and W. H. Ray. *Process Dynamics, Modeling, and Control.* Oxford University Press, New York, 1994.

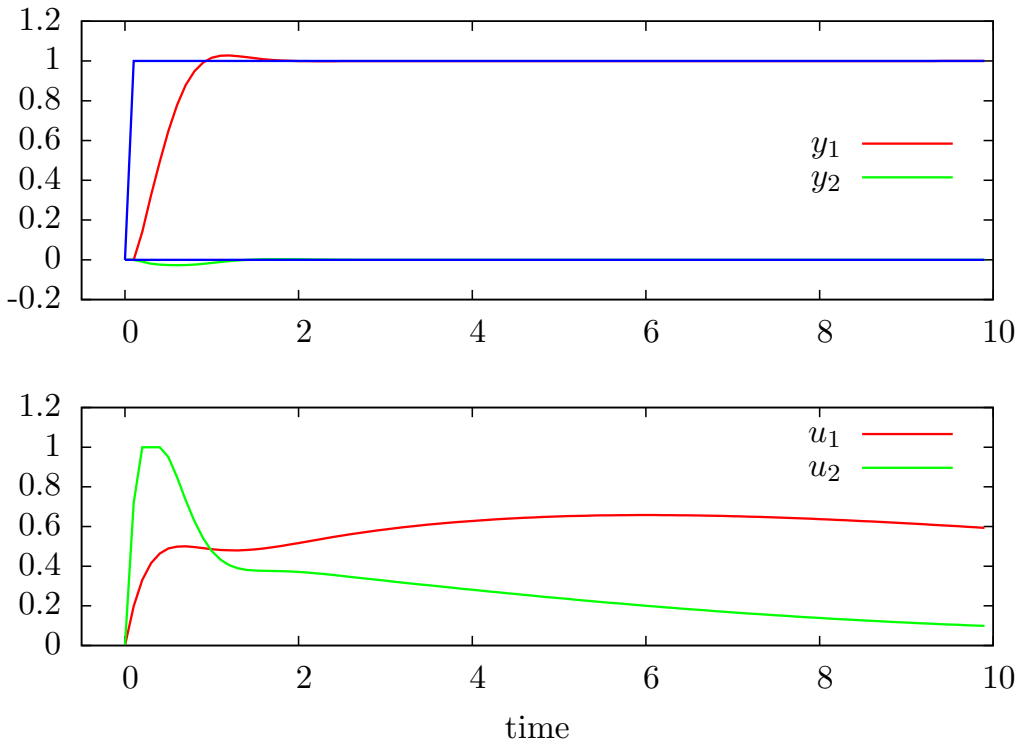

Figure 13: Multi-variable infinite horizon MPC control with input constraints, $Q_y = I, S = I, u_{\min} = -1, u_{\max} = 1$. Setpoint change.

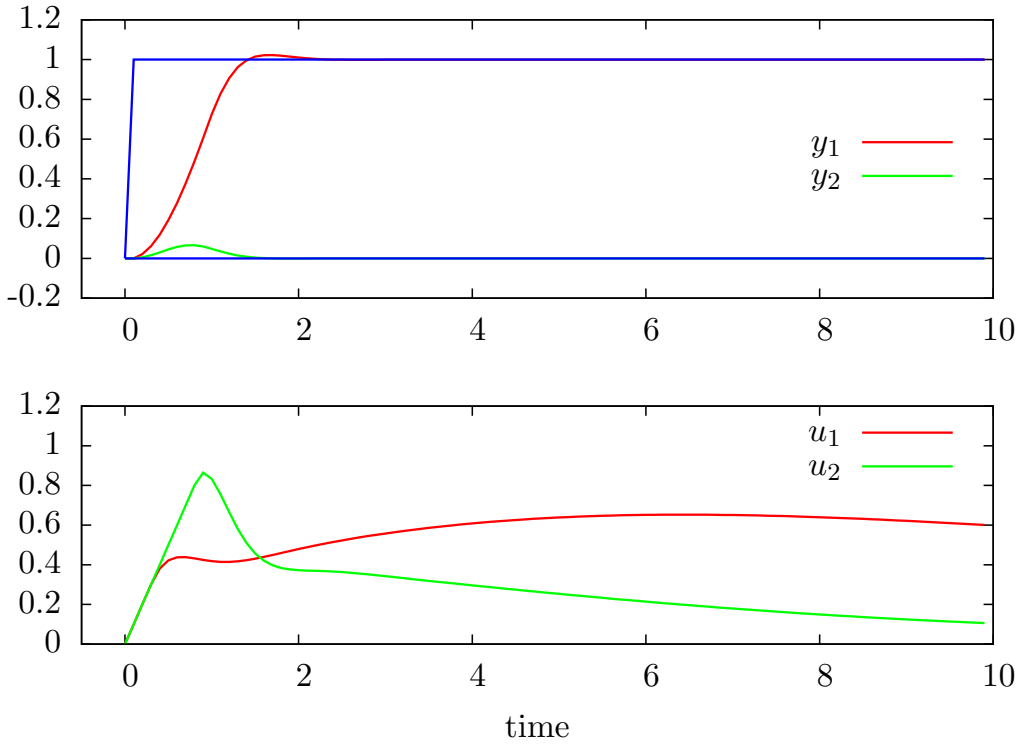

Figure 14: Multi-variable infinite horizon MPC control with rate of change constraints, $Q_y = I, S = I, \Delta u_{\max} = 0.1$. Setpoint change.