1

- $$x_{t+1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0{,}1 & -0{,}79 & 1{,}78 \end{bmatrix} x_t + \begin{bmatrix} 1 \\ 0 \\ 0{,}1 \end{bmatrix} u_t, \quad y_t = [0\ 0\ 1]\, x_t,$$

$$x_0 = [0\ 0\ 1]^T$$

$$f(y,u) = \sum_{t=0}^{N-1} (y_{t+1}^2 + r u_t^2), \quad r = 1,\ N = 30$$

ⓐ $$\det(A - \lambda I) = \lambda(\lambda(\lambda - 1{,}78) + 0{,}79)$$
$$= \lambda(\lambda^2 - 1{,}78\lambda + 0{,}79)$$
$$= \lambda(\lambda - 0{,}844)(\lambda - 0{,}936)$$

Since all eigenvalues are inside the unit circle, the system is stable.

ⓑ $x_t \in \mathbb{R}^3$, $u_t \in \mathbb{R}$.

$$f(z) = \frac{1}{2} \sum_{t=0}^{N-1} (x_{t+1}^T Q x_{t+1} + u_t^T R u_t)$$

$$Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 2 \end{bmatrix}, \quad R = 2$$

ⓒ Since Q and R are positive semidefinite, the problem is convex.

(d) $\min_{z} f(z) = \frac{1}{2} z^T G z$, s.t. $A_{eq} z = b_{eq}$.

$$z = [x_1^T \cdots x_N^T \; u_0^T \cdots u_{N-1}^T]^T$$

$$\Rightarrow G = \begin{bmatrix} Q & & & & & & \\ & Q & & & & & \\ & & \ddots & & & & \\ & & & Q & & & \\ & & & & R & & \\ & & & & & R & \\ & & & & & & \ddots \\ & & & & & & & R \end{bmatrix}$$

$$x_{t+1} = A x_t + b u_t$$

$$x_1 = A x_0 + b u_0 \iff x_1 - b u_0 = A x_0$$

$$x_2 - A x_1 - b u_1 = 0$$

$$\vdots$$

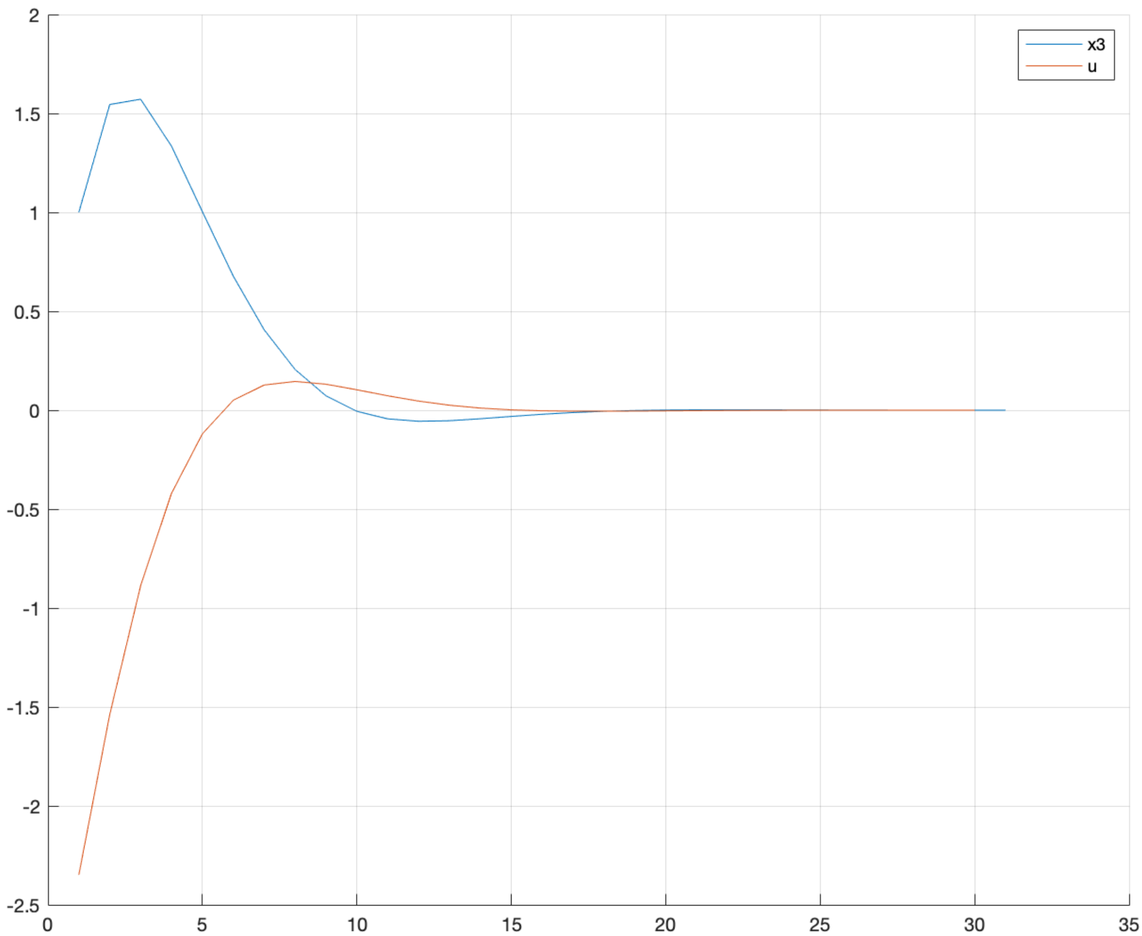
$$x_N - A x_{N-1} - b u_{N-1} = 0$$

Which in matrix notation can be expressed as:

$$3N \left\{ \underbrace{\begin{bmatrix} I & 0 & \cdots & 0 \\ -A & I & \cdots & \vdots \\ 0 & -A & \ddots & \\ \vdots & & & 0 \\ 0 & \cdots & -A & I \end{bmatrix}}_{3N} \underbrace{\begin{bmatrix} -b & 0 & \cdots & 0 \\ 0 & -b & & \vdots \\ \vdots & & \ddots & \\ 0 & \cdots & & -b \end{bmatrix}}_{N} \right. \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \\ u_0 \\ \vdots \\ u_{N-1} \end{bmatrix} = \left. \begin{bmatrix} A x_0 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix} \right\} 3N$$

KKT:

$$\begin{bmatrix} G & -A_{eq}^T \\ A_{eq} & 0 \end{bmatrix} \cdot \begin{bmatrix} z^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} 0 \\ b_{eq} \end{bmatrix}$$



x3
u
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
5
10
15
20
25
30
35

There is no feedback, so open-loop.
By continously solving the optimalization problem with the current measurement/estimate as x_0, we can close the loop. This obviously makes the system more robust, as it will reevaluate what the optimal trajectory is when noise, disturbance and model inaccuracies inevitably will make our optimal course not so optimal in the real world.

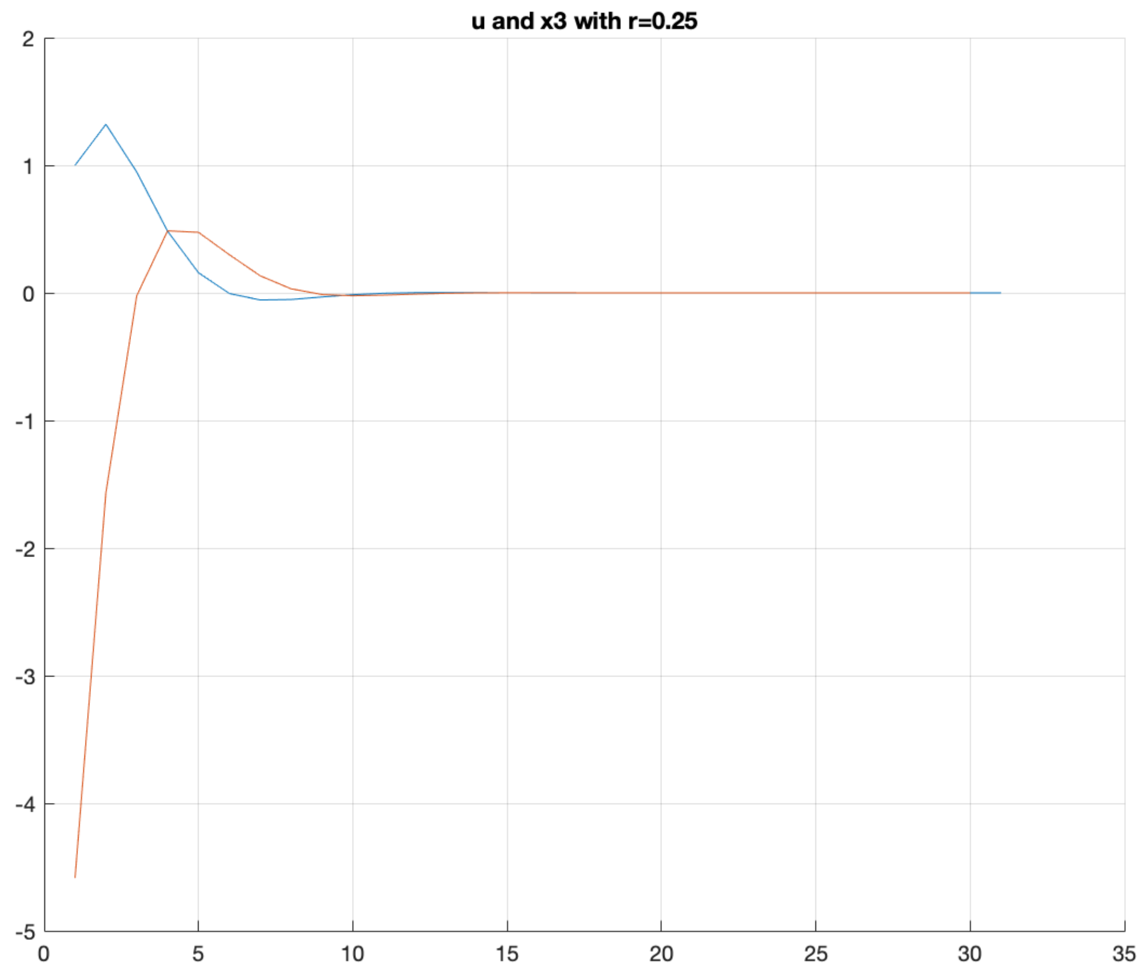
(e) As we change the weight of the input, we see that the controller gets more / less aggressive / conservative.

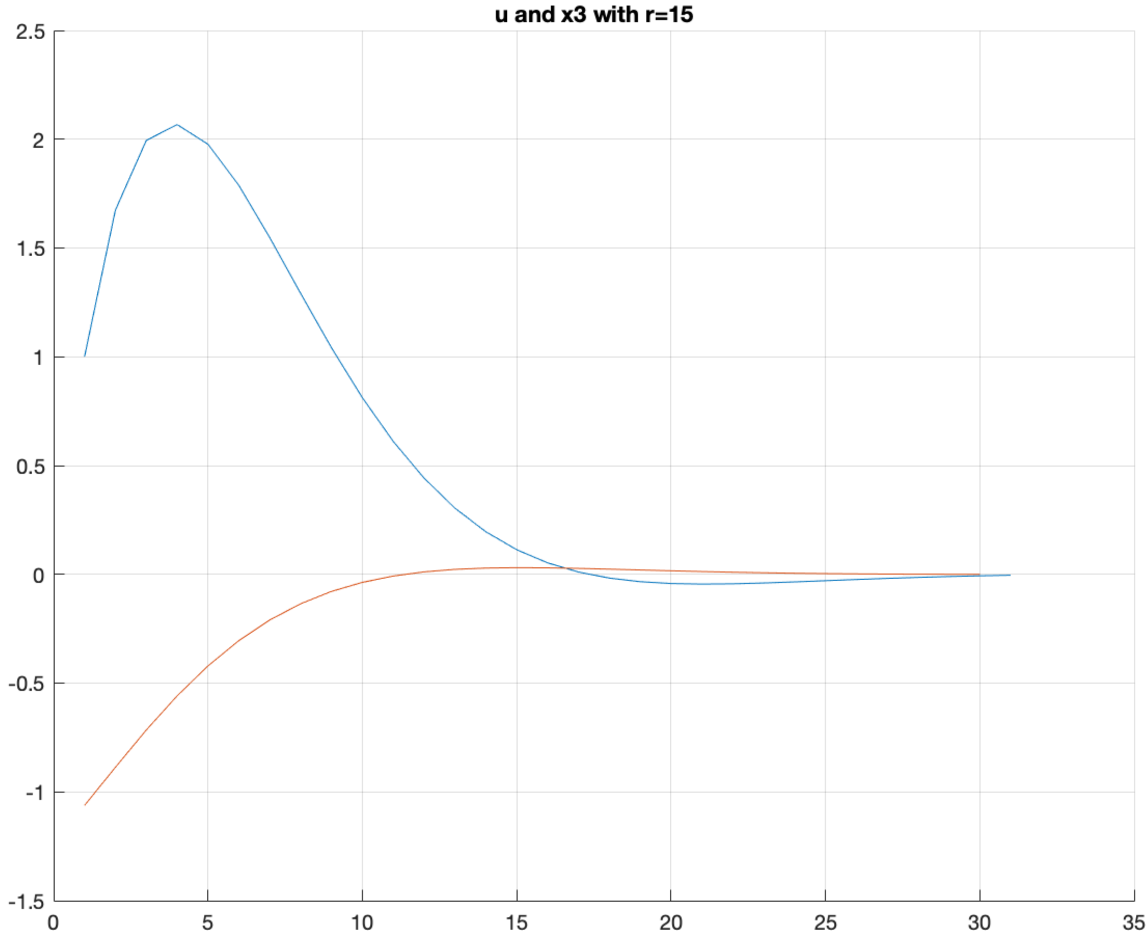
(f) $-1 \leq u_t \leq 1, \; t \in [0, N-1]$

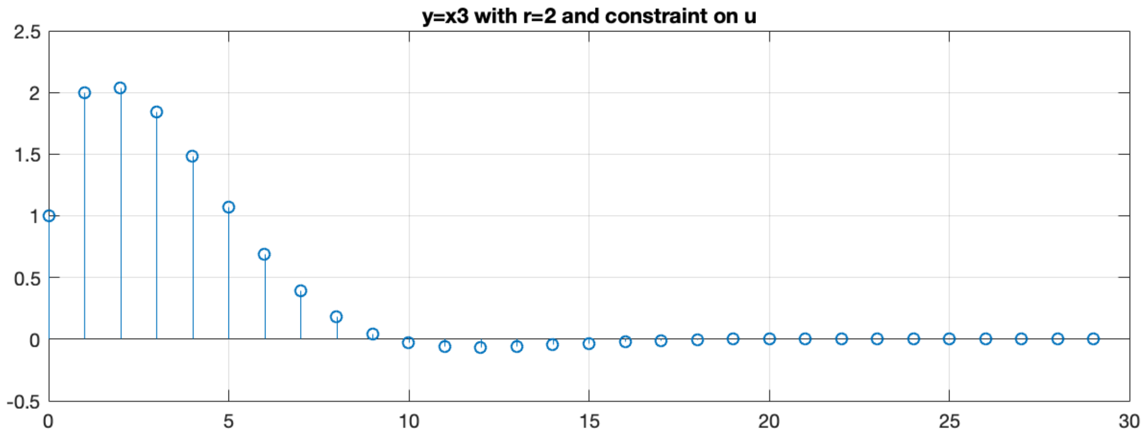
We incorporate this as a single constraint:

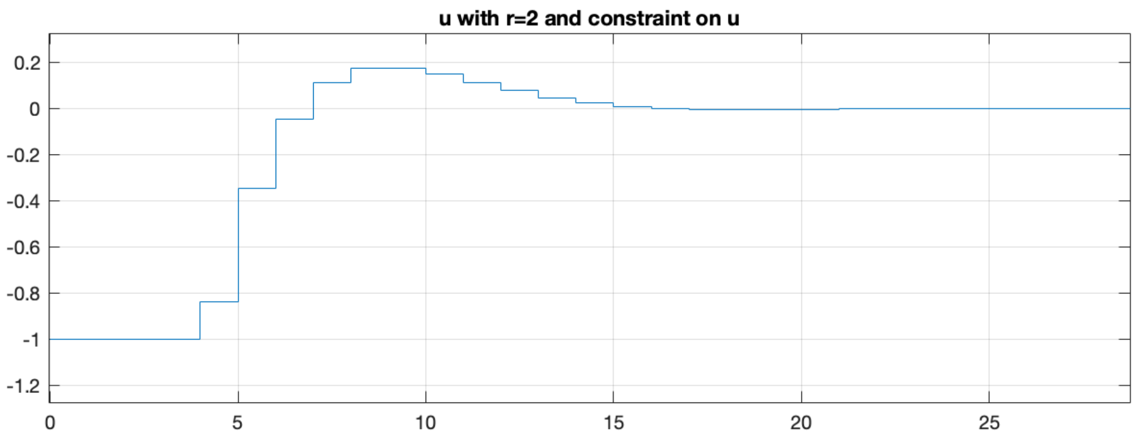
$u_t - 1 \leq 0, \; -1 - u_t \leq 0 \iff Az \leq b$

$$
2N \left\{ \left[\begin{array}{c|c} \overbrace{\qquad 0 \qquad}^{3N} & \overbrace{\begin{matrix} 1 & 0 & \cdots & \\ -1 & 0 & & \\ 0 & 1 & 0 & \\ 0 & -1 & 0 & \ddots \\ \vdots & & & 1 \\ & & & -1 \end{matrix}}^{N} \end{array} \right] \right. \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \\ u_0 \\ \vdots \\ u_{N-1} \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}
$$


u and x3 with r=0.25
2
1
0
-1
-2
-3
-4
-5
0
5
10
15
20
25
30
35

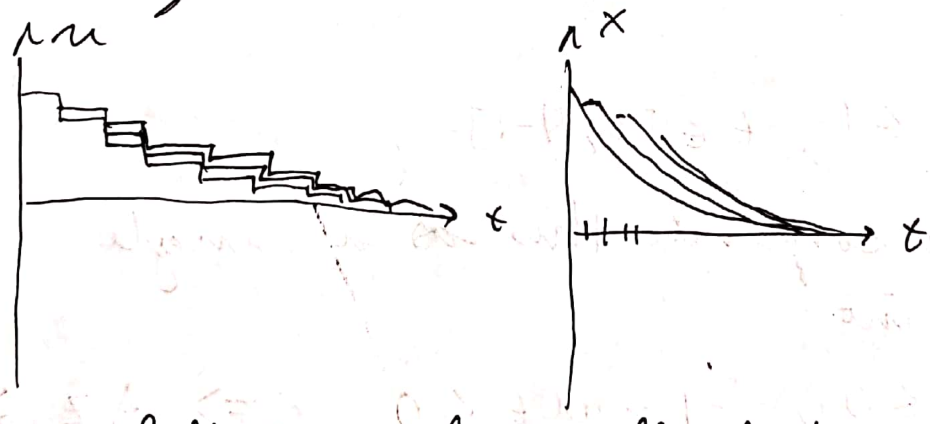

u and x3 with r=15
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
5
10
15
20
25
30
35


y=x3 with r=2 and constraint on u
2.5
2
1.5
1
0.5
0
-0.5
0
5
10
15
20
25
30

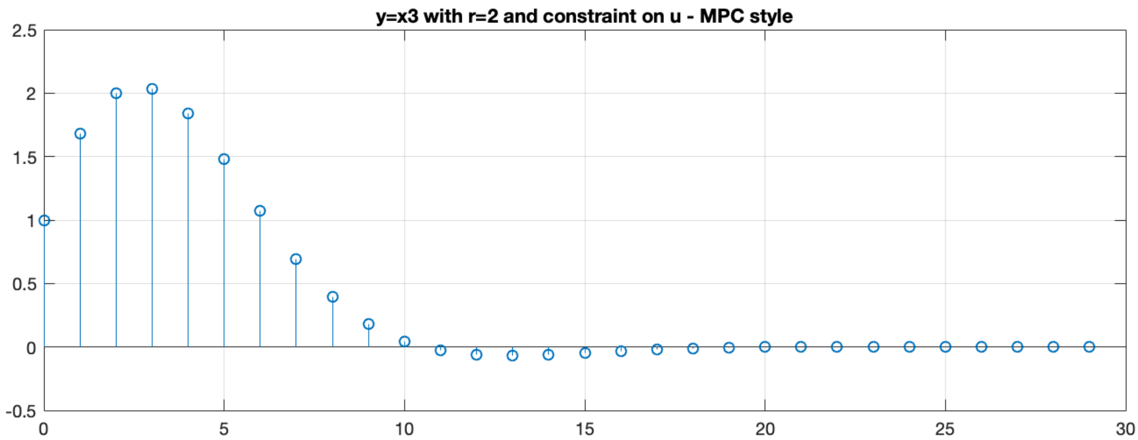

u with r=2 and constraint on u
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
0
5
10
15
20
25

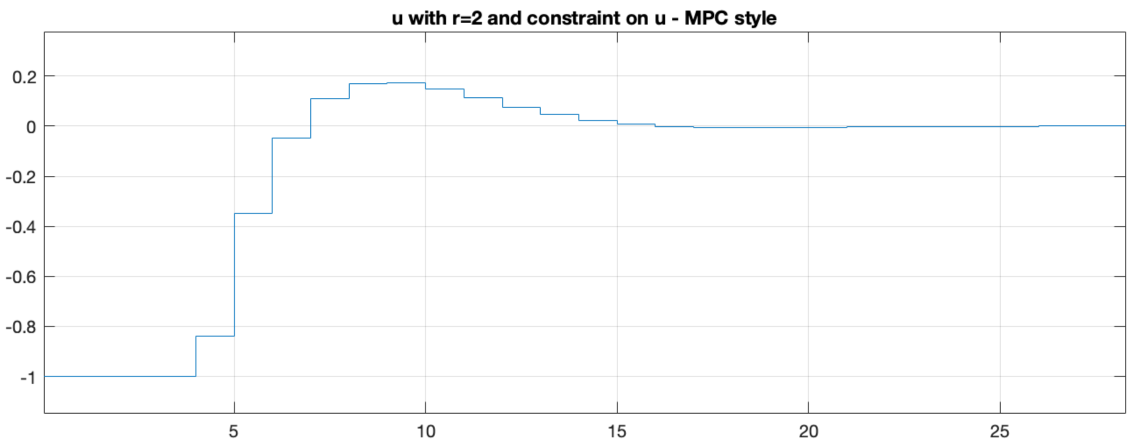
Now the algorithm uses 5 iterations instead of 1. This is because an EQP is solved by simply solving a linear system, as we saw in 1d. Now the algorithm uses the active set method, which has to converge on the optimal solution.

2 a) MPC takes the open loop optimalization problem, and solves it continously at a given time interval with X_0 as a measured/estimated current state of the system.

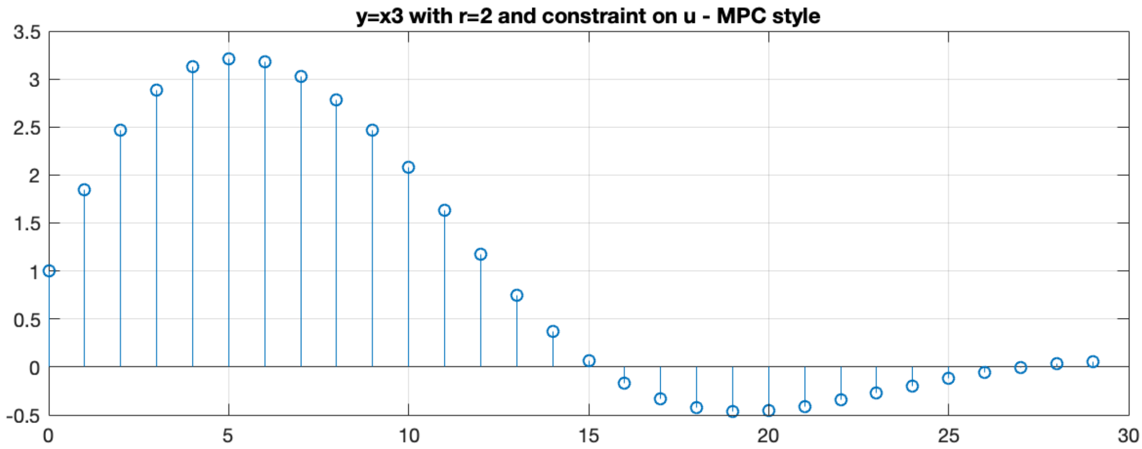


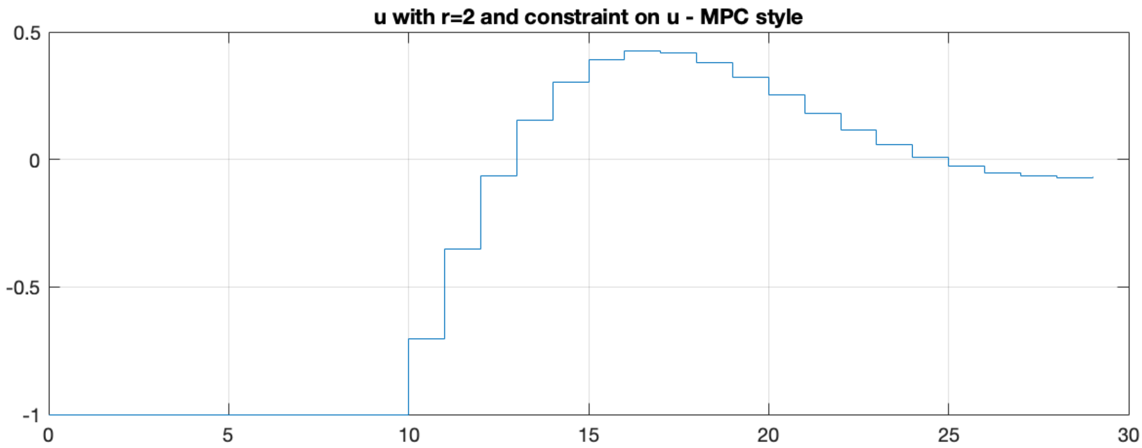
Hopefully my bad illustration somewhat shows that the optimal input sequence is reevaluated at each timestep as X will not behave exactly as we predict.

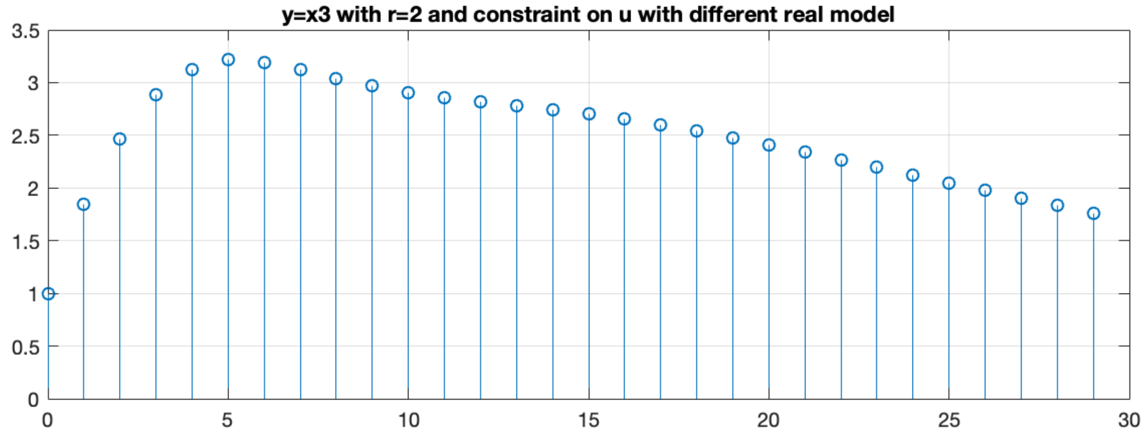


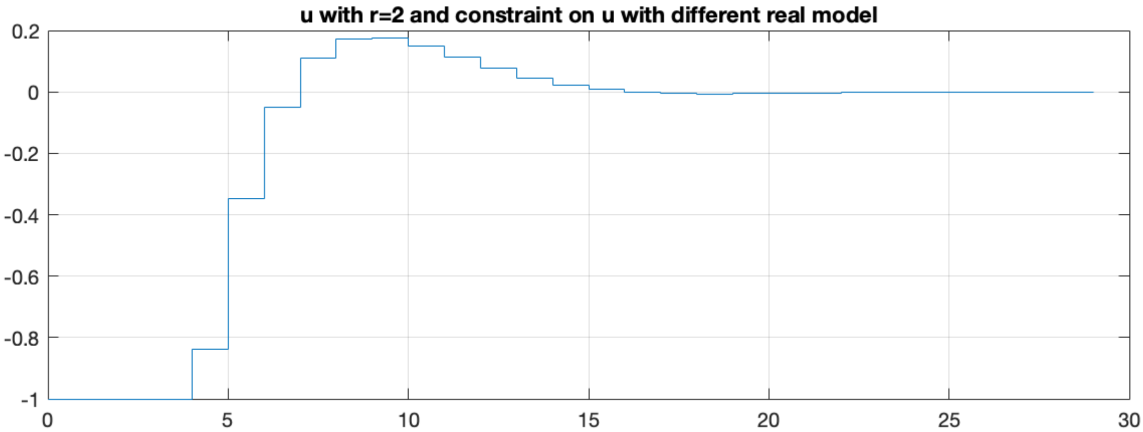


When the model is perfect there is not much difference between MPC and open loop optimization.

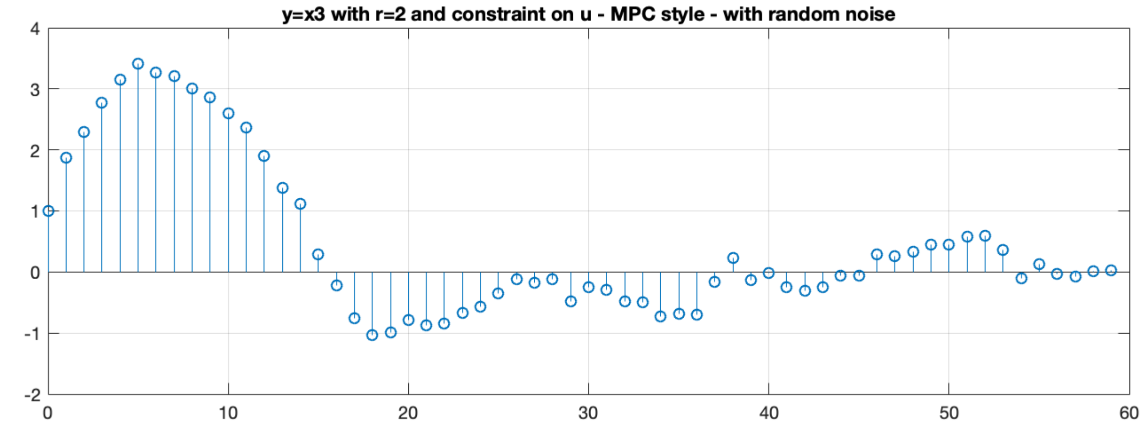

y=x3 with r=2 and constraint on u - MPC style
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
0
5
10
15
20
25
30

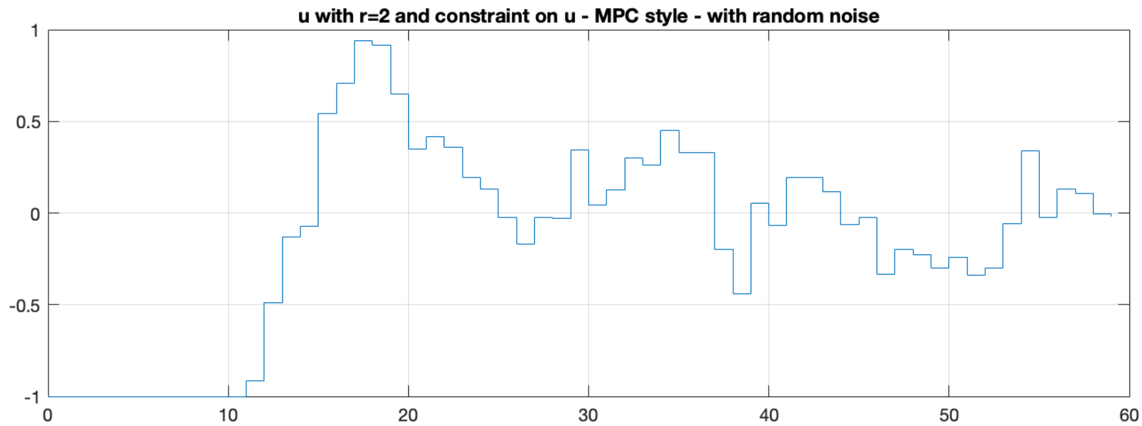

u with r=2 and constraint on u - MPC style
0.5
0
-0.5
-1
0
5
10
15
20
25
30





We observe that the MPC controller is a lot worse when the model is not completely accurate, but it still manages to get to zero ish within the time horizon. If we didn't have feedback on the other hand, as in task 1, we see that the controller is terrible.
This is clear experimental evidence that using feedback and MPC is a lot more robust. We cannot realistically assume that the model is correct, even with small deviations the open loop controller sucks.





Here is the MPC with a wrong model and gaussian disturbance on the state, fun stuff