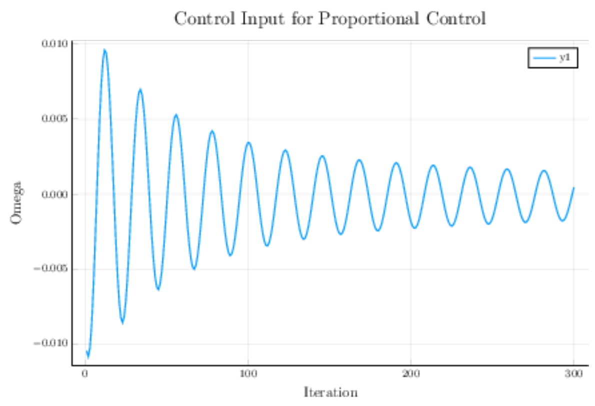

Figure 1: Control policy for the training data. The control policy comes from a proportional controller measuring the y-velocity at a point in the wake

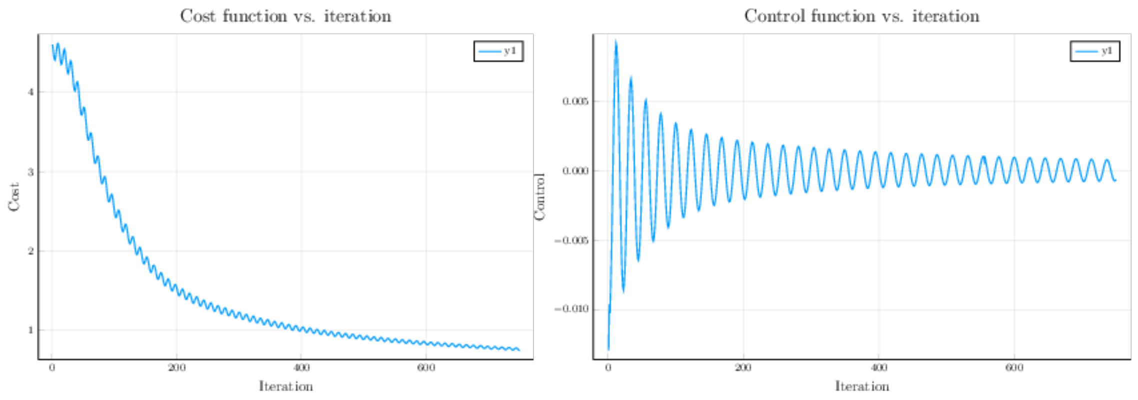

Figure 2: Control policy from DMDc dynamics with MPC controller.

What works

I've have shown that DMDc works for suppressing vortex shedding in a very narrow set of circumstances (see fig 2). Namely, when the dynamics and control matrices (A and B) are computed from snapshots taken from the proportional control of the same system. In this case, the DMDc + MPC algorithm creates essentially the same control policy.
I think this can be regarded as a minor success since the DMDc policy was entirely determined from MPC on the dynamical model that DMDc had constructed. But, as will be shown in the next section, this dynamical model is appears to be fragile and doesn't work in most other circumstances.

What Doesn't work

When A and B are found from training data that deviates even slightly from the proportional control data, MPC doesn't seem to work at all. I have tracked this back to the fact that the predictive capabilities of DMDc seem to fail outside of the specific circumstances of the training data.
In order to evaluate the predictive capabilities of DMDc, I did the following: From a sample MPC run, I copied the dynamics matrices, A and B, the projection matrix U, and all of the simulation data from the run. Then for each iteration of the simulation run, the following 16 timesteps are predicted using the dynamical system. If the t^{th} frame of the simulation is flattened into a state vector x_t, and the control input is u_t, then

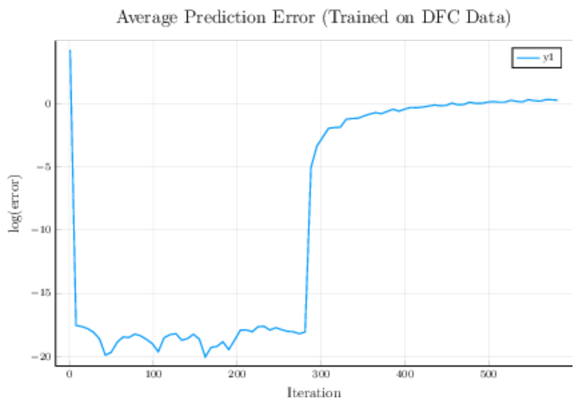

Figure 3: The prediction error of the DMDc model trained from the proportional control data, predicting the proportional control data

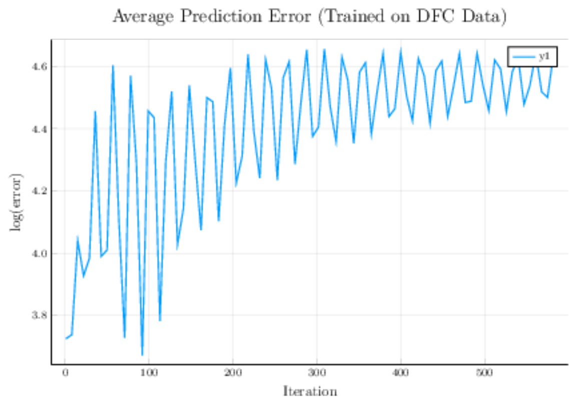

Figure 4: The prediction error of the DMDc model trained from a deep flow control data set, predicting the proportional control data

we predict via

$$\begin{aligned}\tilde{x}_t &= U^T x_t \\ \tilde{x}_{t+1}^{\text{pred}} &= A\tilde{x}_t + Bu_t \\ x_{t+1}^{\text{pred}} &= U\tilde{x}_{t+1}^{\text{pred}}\end{aligned}$$

Then at each predicted timestep, the error in the prediction can be found via the norm $||x_t^{\text{pred}} - x_t||$, where x_t is the exact state, found from the simulation.

Things I've tried

- Used many different types of input training data and checked the predictive power on many other sets of data
- tried online computation of the A and B matrices, for each timestep, or every interval of 20 timesteps. Due to the sensitivy of DMDs capbility of predicting future dynamics, the constant recomputing of these matrices seems to cause the algorithm to become unstable and fail