18.065 Final Project: Application of Basis Pursuit in a Control Systems Set-Up

Christoph Schultheiss and Akhil Chandra Panchumarthi
MIT EECS

In our project we wanted to explore the power of a compressed sensing matrix method, namely basis pursuit. In order to see its performance in a situation that could be of a value in the real world, we created a simulated control systems set-up, where basis pursuit might turn out to be useful.

1 Introduction

1.1 Compressed Sensing and Basis Pursuit

Imagine you have an incomplete representation of a vector y, call it y_s. The problem is to recover the full vector y with just this knowledge. This is possible using basis pursuit assuming that y can be sparsely represented in another basis. For simplicity let y be represented in the standard basis, let A be the matrix containing basis vectors with respect to which y is sparse, and let x be the corresponding coordinates, so x shall be sparse. We then have $Ax = y$. Let A_s denote the rows of A corresponding to y_s. It still holds $A_s x = y_s$ but this system of equations is not generally uniquely solvable. That is where basis pursuit comes in to play, it asks for the following optimization [4]:

$$\text{Minimize } ||x||_1 \text{ subject to } A_s x = y_s$$

The idea behind this is that we would like a sparse solution x and thus we ask for the best convex approximation to sparsity using the ℓ^1 ball similar to LASSO. Under certain conditions this is guaranteed to find the right vector x [1], though in practice it often even works when these guarantees are not fulfilled yet.
Having found the right vector x one can easily obtain the full y using $Ax = y$.
A typical situation to use basis pursuit would be a periodic signal, which can be sparsely represented in its Fourier basis, but for which only a limited amount of measurements in the time domain is available.

1.2 Inverted Pendulum

Most of the ideas presented in this section are inspired by [3].
The control system which we applied our idea to is an inverted pendulum; a rigid rod that rotates around a pivot. This is shown in figure 1.

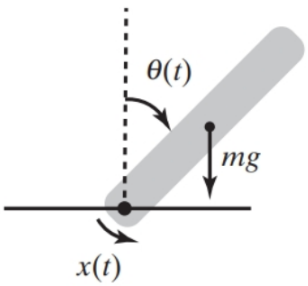


Figure 1: Inverted Pendulum [3]

The goal is then to stabilize this rod in its upright position, thus in an unstable equilibrium. Assume the rod to be of mass m, having moment of inertia I and let its center of mass be distance l away from the pivot. Let θ be the angular displacement with respect to the upright equilibrium, where θ goes from $-\pi$ to π, and $\dot{\theta}$ its corresponding derivative. To make the task harder, further assume there is no friction. Then, the physics of the system are governed by the following equation $\ddot{\theta} = \frac{mgl}{I} \sin(\theta)$. In order to control this, a correction torque x is applied changing the equation to $\ddot{\theta} = \frac{mgl}{I} \sin(\theta) - \frac{x}{I}$. Lastly, to get a better understanding of the system's behavior one can use that around the equilibrium $\sin(\theta) \approx \theta$ and thus in this regime $\ddot{\theta} \approx \frac{mgl}{I}\theta - \frac{x}{I}$. This now enables us to write the whole system's dynamics as a matrix equation.

$$\begin{pmatrix} \dot{\theta} \\ \ddot{\theta} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ \frac{mgl}{I} & 0 \end{pmatrix} \begin{pmatrix} \theta \\ \dot{\theta} \end{pmatrix} + \begin{pmatrix} 0 \\ -\frac{1}{I} \end{pmatrix} x$$

To simplify notation let this be $\dot{q} = Aq + bx$, where q will be referred to as the state in the following.

A has distinct eigenvalues ($\lambda_1 = \sqrt{\frac{mgl}{I}}$, $\lambda_2 = -\sqrt{\frac{mgl}{I}}$) and is thus diagonalizable as $A = V\Lambda V^{-1}$. Plugging this into the equation and multiplying from the left by V^{-1} it becomes $V^{-1}\dot{q} = \Lambda V^{-1}q + V^{-1}bx$. By defining $r = V^{-1}q$ and $\beta = V^{-1}b$ to get $\dot{r} = \Lambda r + \beta x$ we have decoupled analytically solvable equations in the eigenbasis, $\dot{r_i} = \lambda_i r_i + \beta_i x$ for $i = 1, 2$.
$\forall\, t_2 > t_1$ it then holds

$$r_i(t_2) = r_i(t_1)\, e^{\lambda_i (t_2 - t_1)} + \int_{t_1}^{t_2} e^{\lambda_i (t_2 - \tau)} \beta_i x(\tau)\, d\tau$$

1.2.1 Eigenvalue Shaping

Without any control the linearized system would behave as $q(t) = c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t}$, where c_1 and c_2 are determined from the initial conditions. This is problematic since $\lambda_1 > 0$ corresponds to the unstable equilibrium. A very basic control idea is then to set $x(t) = g^T q(t)$ for some suitable g^T. Now the linearized equations become $\dot{q} = \left(A + bg^T\right) q$ and $q(t)$ evolves according to the eigenvalues of $\left(A + bg^T\right)$ instead, which we are able to make negative using a suitable g^T. This is the essence of eigenvalue shaping.

1.3 Real World Simulation

In order to test our ideas we created a simulated environment of the inverted pendulum presented in section 1.2. The simulation is done using an explicit 4th order Runge-Kutta method of the

non-linear evolution equations, which at every time-step has complete information about the real state q, control torque x and possible disturbances. Each time-step is 0.001s. In order to distinguish this from the internal state estimate simulation described in section 1.4, we denote this simulation of the evolution of the true state q as the real world simulation.

1.4 Our Control

Our control is based on the idea presented in section 1.2.1 though instead of using the real vector $q(t)$, which would require measurements, we create an estimate $\hat{q}(t)$ using the analytic solution of the linearized model. Thus our input moment becomes $x(t) = g^T \hat{q}(t)$ and it is held constant at this value until the next update. The torque update frequency is defined such that the time needed for calculating the next $\hat{q}(t)$, the internal simulation, is shorter than the interval it is supposed to represent in the real world simulation.
Preliminary tests using this control without any disturbances showed that it is able to correct an initial offset towards the equilibrium, though due to slight differences between the real q and its linearized estimate $\hat{q}$ it does not bring the real angle and angular frequency to exactly zero. The control would not be applied anymore when the state estimate is exactly zero, but the real angle is nonzero. The system equations revert back to the case with no eigenvalue shaping. Therefore the states will eventually diverge according to $\lambda_1 > 0$.
Having seen this we slightly modified our control approach: With a certain frequency we take measurements of the real state q and take this a new reference point for future control and state estimate $\hat{q}$ until the next measurement. With this modification we were able to stabilize the system.

2 Basis Pursuit in our Control Set-Up

To apply basis pursuit we assumed that besides from our control torque there is an additional input torque. This can be thought of as either disturbances or as an adversary. Further these disturbances are constraint to being periodic and sparsely representable in the Fourier domain. Our goal is then to find the signal's Fourier representation as fast as possible, such that we afterwards know the disturbance for all times and can correct for it.
More precisely the disturbances were constraint to have a period of $100s$ and 1000 possible Fourier coefficients. Thus a measurement is taken every 0.1 seconds until the outcome of basis pursuit does not change anymore. Since after the first few measurements basis pursuit might not yet find the correct Fourier representation, the created signal estimate might generally be complex. In order to avoid this, the classic DFT matrix was replaced by its real equivalent. The real DFT $N \times N$ matrix, call it F, for even N then has the following structure (name the rows and columns 0 to $N-1$):

$$F_{i0} = 1; F_{ij} = \cos\left(2\pi \tfrac{ij}{N}\right) \text{ if } 1 \leq j \leq \tfrac{N}{2}; F_{ij} = \sin\left(2\pi \tfrac{i(j-N/2)}{N}\right) \text{ if } \tfrac{N}{2} < j \leq N-1$$

This matrix still consists of orthonormal columns. Since one transform direction uses basis pursuit and the other direction actually wants to estimate a signal for all times and not just the measurement instances, FFT and inverse FFT could not be applied anyways, thus this change to a real DFT matrix does not come with any additional computational cost. In order to actually apply basis pursuit, we used the algorithm provided by [2]. In order to create disturbances, nonzero Fourier indices were chosen at random according to the sparsity level and a random amplitude was matched to every coefficient, which then defined the disturbance for all times.

2.1 Updated Control

With the presence of disturbance naturally the control must be adjusted. Optimally the same total moment (the sum of disturbance and control torque) as in the case without disturbances would be applied, thus the control torque x should correct for the presence of disturbance. Denote the current estimated disturbance function by $\hat{d}(t)$, and let t_i and t_{i+1} be two successive instances, where the control torque is updated. In this interval the torque is then held at

$$x(t) = g^T \hat{q}(t_i) - \int_{t_i}^{t_{i+1}} \hat{d}(\tau)\, d\tau / (t_{i+1} - t_i),$$

thus it subtracts the estimated disturbance torque's mean over the interval.
For the state estimate $\hat{q}$ the equation from section 1.2 is adjusted to

$$r_i(t_2) = r_i(t_1)\, e^{\lambda_i(t_2 - t_1)} + \int_{t_1}^{t_2} e^{\lambda_i(t_2-\tau)} \beta_i \left(x(\tau) + \hat{d}(\tau) \right) d\tau$$

As long as basis pursuit has not converged yet, t_1 is chosen to be the instance of the last measurement of the true state. As soon as the disturbance estimate does not change further, i.e. basis pursuit has converged, t_1 can be simplified to be the instance of the last state estimate in order to increase computational efficiency.

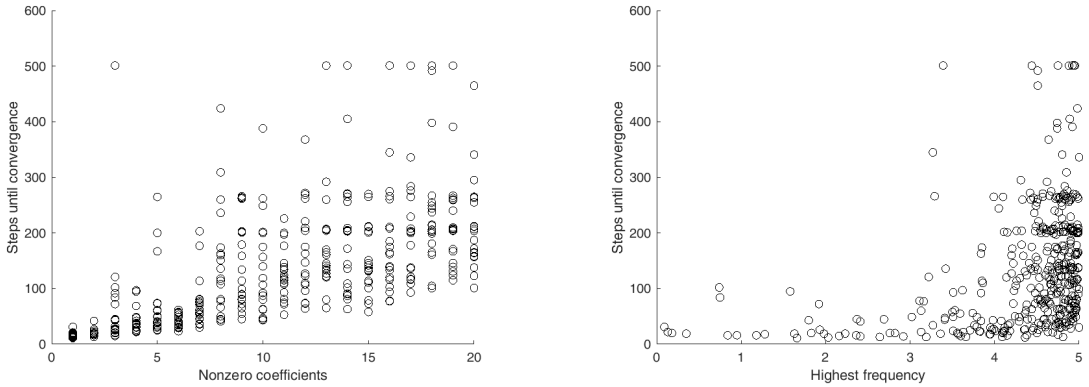
3 Results

3.1 General Analysis of Basis Pursuit

In a first step we wanted to see how well basis pursuit would generally perform with our real DFT matrix. For this signals were created the same way as in the control set-up of section 3.2 for different levels of sparsity. Namely the number of nonzero coefficients was varied from 1 to 20 and for every sparsity level 20 runs were done. In each run the number of needed measurements until basis pursuit has converged was noted. Here convergence is defined to be reached if the outcome of basis pursuit changes by less than a certain threshold after adding an additional measurement into the algorithm. Based on a couple of previous tests we also assumed higher frequencies would lead to basis pursuit taking longer to converge, therefore the highest randomly created frequency was also noted. Figure 2 shows the outcome of this analysis, a small amount of noise is added to every point in order to distinguish overlapping points. Notably the two explanatory features sparsity level and highest frequency are not uncorrelated, since the more frequencies are chosen the higher the maximum will generally be.
These plots at first glance confirmed our two assumptions. To get an even better understanding a linear regression was then fit to the data, thus an estimate of the form:

$$\text{convergence} = a + b*(\text{number of nonzero coefficients}) + c*(\text{highest frequency})$$

The output of this model is as follows: $a = 13.7$, $b = 11.5$, $c = -6.70 * 10^{-3}$.
Thus the natural assumption, that more nonzero coefficients would slow down convergence was confirmed, having also a highly significant p-value of $5 * 10^{-36}$. Though, when already taking sparsity into account the actual frequency seems to have barely any effect, for this data set even a negative, and the corresponding p-value is also non-significant (0.9). This assumption was therefore disproved.
Looking at the data, naturally, this model is not sufficient to explain the inherent behavior, but still gives some reasonable estimate of the different effects.
Looking at figure 2a we also saw that up until around five nonzero coefficients basis pursuit should converge reasonably fast for most randomly created signals, and therefore control should be possible in these cases.



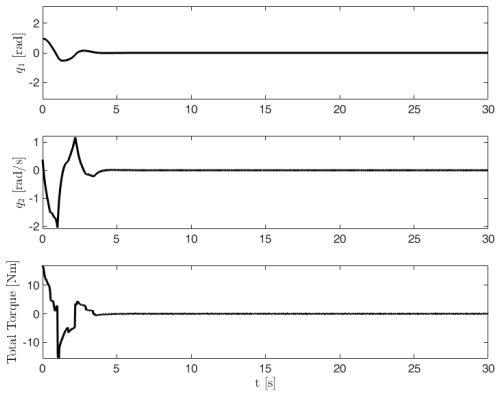
(a) Effect of sparsity level

(b) Effect of highest frequency

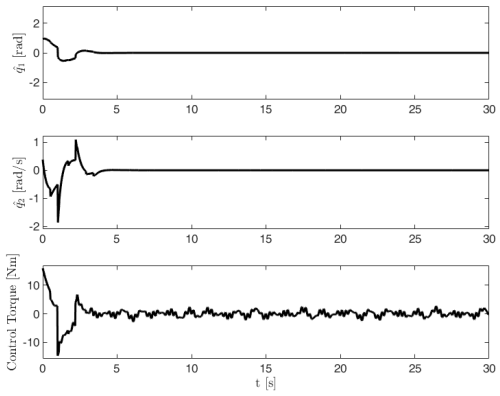
Figure 2: Convergence behavior of basis pursuit

3.2 Controlling with Basis Pursuit

The following results were obtained setting $m = 1\ kg$, $l = 1\ m$, $g = 9.81\ \frac{m}{s^2}$ and accordingly, assuming the rod to be infinitely thin, $I = \frac{4}{3}\ m * l^2 = \frac{4}{3}\ kg * m^2$ (note that in section 1.2 l was defined as the distance from the pivot to the center of mass, i.e. half the length of the rod). This leads to $\lambda_1 = \sqrt{\frac{mgl}{I}} = 2.7125\ \frac{1}{s}$, $\lambda_2 = -\sqrt{\frac{mgl}{I}} = -2.7125\ \frac{1}{s}$. The feedback vector g^T was designed such that $\tilde{\lambda}_1 = -\frac{1}{2}\lambda_1$ and $\tilde{\lambda}_2 = \lambda_2$. The number of nonzero Fourier coefficients was set to 5, each having an amplitude chosen uniformly in $[-1, 1]$. The first real state measurement is taken after $1s$ and afterwards measurements are taken equidistantly every $1.2s$. All test runs correspond to 30 seconds in the real world simulation. The initial angular velocity is randomly chosen in $[-0.4, 0.4]$ and the initial angle is used as a parameter of analysis. Both initial states are assumed to be known to the control system.
In this set-up stabilization of the pendulum was reliably possible up until an initial angle of $\pi/3$. By reliably possible we mean, that within our test runs we did not find any randomly created initialization for which the systems became unstable in the sense that the pendulum swung around strongly for a long time or even started doing loops.
In most cases, using an initial angle of $\pi/3$, stabilization was reached after around $5s$, thus after only having seen around 5% of the disturbances' temporal structure. This shows how powerful basis pursuit is in this set-up, since a usual matrix inversion technique could only have created a complete estimate of the disturbances after having seen a full period of $100s$, and thus complete stabilization could only happen afterwards.
Figure 3 shows the outcome of one such simulation run.
These plots reveal several things. Firstly, looking at the initial transient phase one sees that the real state and the estimated state diverge quite a bit. This confirms the observation in section 1.4 that control without any state measurements would not be possible. Visually the estimate of the angular velocity seems to diverge more from the reality than the estimate of the angle. This is approved by the corresponding correlation coefficients between truth and estimate in the first five seconds, being 0.9551 for the angle and 0.9039 for the angular velocity. Starting from around $t = 5s$ true states q as well as estimated states $\hat{q}$ stay close to 0 which denotes the mentioned stabilization. Lastly, one also sees that the total applied torque, i.e. the sum of control and disturbances, also stays very close to 0, so our control is able to counterbalance the disturbances using the estimate created by basis pursuit.



(a) Real Angle and Angular Frequency and Total Torque



(b) Estimated Angle and Angular Frequency and Control Torque

Figure 3: Outcome of a test run

In order to assess how powerful basis pursuit actually is we compared it to just taking the pseudo-inverse, which is equivalent to

$$\text{Minimize } ||x||_2 \text{ subject to } A_s x = y_s \text{ [4]}$$

So it is the "ℓ^2-version" of basis pursuit. As expected this worked clearly worse than basis pursuit, since it is unlikely to find a sparse solution and therefore not really approaches the true solution. Furthermore, this missing sparsity makes the whole problem become unfeasible in real time, since every state estimate and torque update includes an integral over loads of nonzero sinusoidals. Figure 4 shows the outcome using this approach using the same initial condition as in figures 3a and 3b. Notably the total applied torque never actually stays at 0 and therefore also the state fluctuates more strongly around its equilibrium. Thus, as expected ℓ^1 minimization strongly outperforms ℓ^2 minimization in this sparsity set-up.

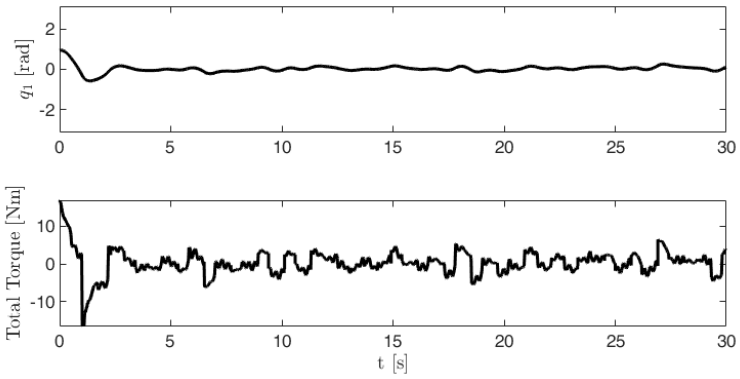


Figure 4: Real Angle and Total Torque using ℓ^2 minimization

One problem we saw in this approach is that the control problem is somewhat easy since most possible disturbance terms are periodic with zero mean and do not push the pendulum in the

same direction for a longer time period. Thus we did a further test where one of the nonzero Fourier coefficients was forced to correspond to the constant term. Then something remarkable happened: basis pursuit was able to figure this coefficient of the constant term much faster than the other coefficients convergence. Therefore stable control is still possible in this changed set-up. Even more we could increase this constant term to having a value of $2Nm$, thus being at least twice as large as any other disturbance part, and stable control was still reliably possible.

3.3 Effect of Noise

Measurements in any real world system are susceptible to noise. Our inverted pendulum simulation, when considered under ideal conditions, isn't, however, it is necessary to test its performance under the influence of noise. There are different ways how noise could influence our control system, however, we shall focus on one important occurrence. This is noise generating errors in the measurements of the clean disturbance signal by our system. Noise is a huge problem to basis pursuit for obvious reasons. Basis pursuit attempts to obtain the exact form of the disturbance signal. In short, the faulty measurements that basis pursuit operates on makes it assume that they are of the underlying signal. It would then try to find the exact disturbance signal that matches them. In a noise-free scenario, this works perfectly, as each measurement is an accurate representation of our disturbance signal. In the case of measurement errors, however, the measurements are of an aperiodic signal and basis pursuit never converges. The construction that we used has our system taking a measurement of the disturbance every $0.1s$. Keeping this in mind, since our simulation runs in 30 seconds, there will be 300 measurements. We assumed basis pursuit would never converge in these 300 steps when subject to noise. Our test trials confirm this.

To fully understand the effect of noise, throughout the rest of the experiment, noise is taken from a normal distribution with standard deviation σ, which can be used as a parameter of analysis.

We analyze part of our results using the signal to noise ratio (SNR), which gives a good understanding how clean our measurements are. Here, SNR is defined to be the value in decibels of the ratio of the magnitude of the true signal and the noise(measurement error).

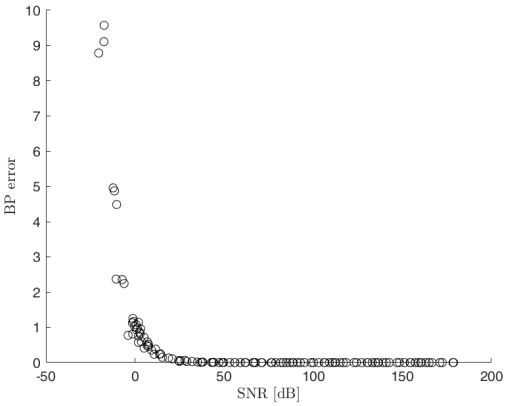
$$\text{SNR} = 20log_{10}\left(\frac{||signal||}{||noise||}\right)$$

Naturally SNR is expected to be negatively correlated with σ. Finite nonzero values of σ affect basis pursuit's convergence to the true signal. The corresponding error of basis pursuit (BP error) is defined by the norm of the difference between the actual signal representation and the basis pursuit estimated signal representation in the Fourier domain after having taken these 300 measurements. The plot is shown in figure 5a.

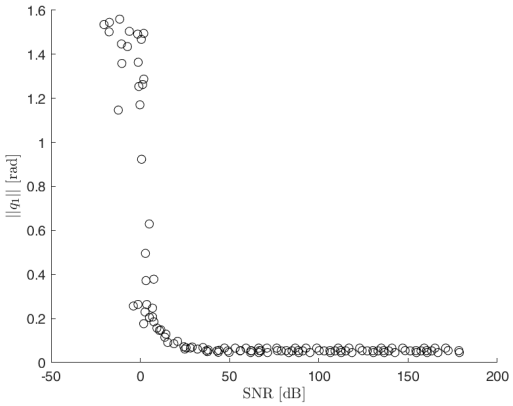
Both plots in figure 5 represent 120 points, calculated over 40 different values of σ ranging from 10^{-4} to 8, with each value being sampled three times. Figure 5a, shows a sharp change at SNR value 0, which corresponds to equal magnitudes of the signal and noise. Negative SNR values express that the noise dominates the signal. It should be noted that at $\sigma = 0.1$, the SNR value is about 20 dB which is where BP error starts to be reasonably small.

Our focus now shifts to how noise affects our inverted pendulum simulation. Besides the noise, the simulation was set up exactly the same as it was set up in section 3.2, with an initial angle of $\pi/3$. Figure 5b shows the plot between SNR and $||q_1||$. $||q_1||$ is calculated by taking ℓ^2 norm of the q_1 vector at all time-steps of the real world simulation. In figure 5b we see a sharp decrease in $||q_1||$ at SNR $= 0$, similar to figure 5a. However, $||q_1||$ stabilizes at a nonzero value at higher SNR values, which is roughly equal to $||q_1||$ in the noise free case coming from the initial transient behavior, as it was seen in figure 3.

We then looked at sample simulations of the inverted pendulum for different values of σ.



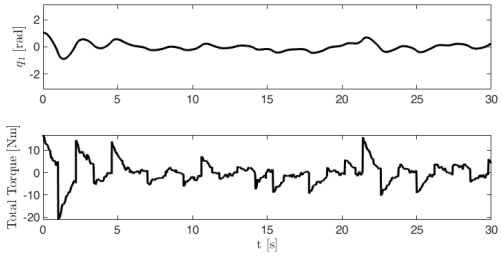
(a) SNR vs BP error



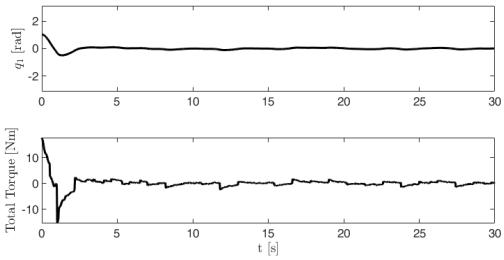
(b) SNR vs $||q_1||$

Figure 5: Outcome of testing with various σ values

Based on the observations from figure 5, we have selected σ values ranging from 10^{-3} to 1. Of the test trials conducted, it is around the value of $\sigma = 0.5$ that the pendulum never completes a full rotation. Rare instances of destabilization at around 25 seconds can though still be noted. Higher values led to unstable behavior in almost every case. In all cases that the pendulum does destabilize, it goes of into a series of rapid rotations. At around $\sigma = 0.25$ one can finally say that we are tending towards our goal, to stabilize q_1 to be as close as possible to zero. For values of σ smaller than 0.1, q_1 stabilizes to 0 relatively quickly within five seconds, almost exactly like figure 3 in section 3.2.



(a) Test run with $\sigma = 0.5$



(b) Test run with $\sigma = 0.1$

Figure 6: Test runs with added noise

The two key values, $\sigma = 0.5$ and $\sigma = 0.1$ are shown side by side in comparison in figure 6. The totally applied torque (sum of control torque and disturbance) in the case of $\sigma = 0.1$ is much more stable around 0, when compared to $\sigma = 0.5$. We can see this same relationship with q_1. The error of the simulation's approximation of the states q_1 and q_2 versus the actual values also reduces with the reduction in σ.
In conclusion, noise naturally decreases the performance of basis pursuit. However, if noise is reasonably small, in the sense that the energy of true underlying signal is at least 10^2 times the energy of the noise (i.e. SNR $\geq$ 20 dB), stabilization of the pendulum still works pretty well.

4 Outlook

One of our initial main goals was actually to be able to do control without any state measurements. Though, as explained in section 1.4, this was quickly discouraged by preliminary tests. Playing around with the system ourselves we then tried to reduce the measurement frequency as much as possible, subject to stable control still being possible. In the age of machine learning this could be nicely recast into a reinforcement learning system. Briefly explained one wants to train a system, that, given the actual measurement, the actual disturbance estimate and a certainty level of this estimate as input features, it outputs a maximal time-step one can keep controlling without taking any further real state measurements. To train such a system one could run a lot of simulations of feature/time-step pairs and create for each pair a reward consisting of both the time-step and the divergence of the real state. Having collected enough data one could train a neural net, which for any given feature/time-step pair estimates this reward, and finally at control-time one uses the argmax over possible time-steps as output. In order to not go beyond the scope of this project we refrained from exploring this idea in more details, though we think this might be a very interesting reinforcement learning problem.
As seen in section 3.3, measurement errors (noise) affect the performance of our control system. Denoised basis pursuit is one way of overcoming this problem. The idea is that unlike basis pursuit, denoised basis pursuit brings about some relaxation to the constraint $A_s x = y_s$, by allowing for an error, i.e. $||A_s x - y_s|| \leq \epsilon$ in order to decrease $||x||_1$ and thus potentially increase sparsity. However preliminary tests using denoised basis pursuit in a set-up similar to the one in figure 5a did not really improve over just using regular basis pursuit, thus this idea was not further pursued.

References

[1] Helmut Boelcskei. “Compressive Sensing”. In: *Lecture Notes for Mathematics of Information* (2018).

[2] Emmanuel Candes and Justin Romberg. *l1-magic : Recovery of Sparse Signals via Convex Programming*. 2005.

[3] Alan V. Oppenheim and George C. Verghese. *Signals Systems and Inference*. Pearson, 2015.

[4] Gilbert Strang. *Linear Algebra and Learning from Data*. Wellesley-Cambridge Press, 2018.