ACS-6110/336/6336
Rapid Control Prototyping
Modelling, Simulation and Control Laboratory

Payam Soulatiantork, Ben Taylor

January 2019

1 Introduction

The aims of these laboratory exercises are as follows:

aims

1. To model a real-world multi-physics system, leading to the construction of a plausible simulation model.
2. To exploit the simulation model in order to enable the rapid prototyping of a stabilising feedback controller.
3. To assess and refine the performance of the resulting closed-loop system to the stage where a real-time controller can be deployed on the actual system hardware.

Objectives

At the end of these exercises, the student will be able to:

1. Operate a real-world system using high-level design software.
2. Develop and analyse models of physical systems from first principles.
3. Devise methods to identify the parameters required to model a real-world system,
4. Design, evaluate and implement a stabilising MIMO feedback controller.
5. Use MATLAB and SIMULINK to design, analyse and verify controller synthesis and performance.
6. Implement and evaluate the performance of the designed controller upon the actual system.

Note

MATLAB possesses well-written and extensive documentation that explains how to use all of its inbuilt functions via the `help` and `doc` commands. For example, to obtain a brief description of the usage of the `lqr` command, simply type `help lqr` in the command window. For more extensive description, type `doc lqr`, instead. If you are unsure of what command to use for a particular task then make use of the search facility in the documentation. Remember, this documentation exists to help you, and should be your first point of call before seeking help from the teaching assistants.

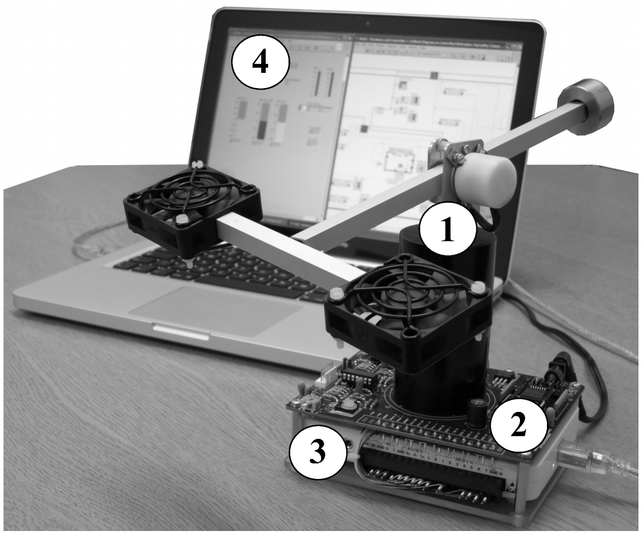

Figure 1: System set-up, showing (1) the 3-DOF helicopter, (2) signal conditioning and power electronics, (3) NI myDAQ interface, (4) control PC.

The System

The system under consideration is a three-degree-of-freedom helicopter shown in Figure 1. Note that this system has two control inputs (from the myDAQ) and two measured outputs (elevation and travel angles). Recall from your lecture notes that the control objectives are (a) to stabilise the helicopter in order to maintain steady hover with the pitch and elevation axes perpendicular to the gravitational vector, and (b) to achieve reference tracking of step commands about the elevation and travel axes. Much of the modelling has been completed and can be found in your lecture notes. You will need to refer to your lecture notes throughout these lab exercises.

The Laboratory

SIMULINK block diagrams

In order to assist you in constructing a SIMULINK model of the helicopter for the purposes of simulating a closed-loop controller, you are provided with three SIMULINK block diagrams with filenames `PitchSys.mdl`, `OpenLoopSys.mdl` and `ClosedLoopSys.mdl`.

The first SIMULINK model, `PitchSys.mdl` models the pitch axis dynamics. You will modify this block diagram for inclusion into the second SIMULINK model, `OpenLoopSys.mdl`, that models the continuous time helicopter dynamics. This will then be inserted into the final model, `ClosedLoopSys.mdl`, that models the entire feedback control system, including the discrete-time dynamics arising from the use of digital control.

MATLAB m-files

The m-file that you will be developing in this laboratory is named `heli.m`. This contains the numeric values of the model parameters used by the SIMULINK blocks above, as well as commands to design a control system. Over the course of the laboratory, you will be adding to your commands, so you should make sure that you save this file at regular intervals.

Exercise Completion

This lab sheet contain a series of exercises that are designed to guide you through the process of rapid control prototyping for the helicopter hardware. As you progress through this assignment, you will be instructed to request a teaching assistant verifies your completion of the exercises. You will need to complete these exercises *and have them verified* before the final day of the course.

Before You Start

Before you begin the laboratory you will need to follow these instructions:

1. Firstly, go to the `Matlab` folder contained within the course folder in MOLE. Then, copy the files `PitchSys.mdl`, `OpenLoopSys.mdl`, `ClosedLoopSys.mdl` and `heli.m` from the `Matlab` folder to a directory in your user area. For example, you could create a folder in your `U`-drive called `u:\lab` and store the files in there.

2. Change MATLAB's working directory to the directory where you have stored the lab files. For example, to change to the `u:\lab` folder in MATLAB type `cd u:\lab`

3. Before using any of the SIMULINK models, you will need to run the `heli.m` file. In order to edit this m-file, simply type `edit heli` at the command line in MATLAB.

Calibration

No two helicopters share the exact same dynamics. Before using your helicopter, you will need to calibrate the dynamics about its pitch and elevations axes.

- To calibrate the pitch axis, begin by holding the elevation arm horizontally and then gently perturb the helicopter about the pitch axis. Most of the helicopters have already been balanced, but if your helicopter consistently settles to one side (by more than approximately 10° from horizontal) then you will need to rebalance it. Balance the helicopter by gently attaching metal nuts, available from the TAs, onto the white polycarbonate screws that hold the fan guards.

  It is also good practice to calibrate the pitch axis sensor at the beginning of each session, since this is known to be sensitive to temperature variations. This is particularly relevant to ACS336/6336 students, since the module runs from February until May, over which time ambient room temperatures typically increase.

- To calibrate the elevation axis, you will need to take your helicopter over to the weighing scales. If there is a queue for this then proceed immediately to Exercises 2 and 3.

  Zero the weighing scales and then gently lower the elevation arm until the pitch beam is resting on the scales. The aim is to obtain a reading on the scales of 4.0 g.

  If the helicopter is too heavy, then begin unscrewing the cylindrical counterweight at the rear of the elevation beam. This moves the helicopter's centre of mass backwards, thus reducing the effective weight of the helicopter.

  If the helicopter is too light, then begin tightening the counterweight at the rear of the elevation beam. Do not over tighten this, or you will damage the threaded insert. If the counterweight is fully screwed in and the helicopter is still too light, then begin attaching metal nuts to the polycarbonate screws that hold the fan guards in place. Attaching nuts in a symmetrical fashion about the pitch axis will ensure that this does not interfere with the balance of the pitch beam.

Exercise 1: Modelling the Pitch Axis Dynamics

The aim of this exercise is to implement the rigid body equations governing the pitch axis dynamics (given in your lecture notes) in SIMULINK, and to identify the parameters that dictate these dynamics.

- To begin with, open the `heli.m` file ready for editing and uncomment the lines for Exercise 1.

- Next, open the SIMULINK model `PitchSys.mdl` and double-click on the block named `Pitch Dynamics`. This block has been partially completed for you, but you will need to complete it by connecting appropriate blocks from the SIMULINK library (`View>Library Browser`) in order to model the pitch axis equation given in lectures.

  When specifying the parameters of these blocks, it is advisable to do so algebraically, rather than numerically. For instance, by double-clicking on the `Double Integrator` block you will notice that the upper and lower limits of the pitch-axis motion are specified by the parameters `pitch_lim_u` and `pitch_lim_l`, respectively, and that the numeric values of these parameters are defined in the `heli.m` file.

- Once your block diagram is complete, enter arbitrary values for the spring and damping coefficients, `k_s` and `k_d` in the `heli.m` file, run this file from the MATLAB command line, simulate the response of the pitch beam dynamics in SIMULINK (`Simulation>Start`) and view the response by double-clicking on the scope. Note that the force inputs to the pitch dynamics are tied to zero, and so you are simulating the unforced response. The initial condition on the pitch angle and angular velocity are set in the `Double Integrator` block, and correspond to the pitch beam being released form rest at an angle specified by `pitch_lim_u`.

Most of the helicopters display a lightly damped response about their pitch axes when released from rest. Compare your simulated response to the actual response of your helicopter. These responses are unlikely to agree at the first attempt, so you will need to adjust the values of `k_s` and `k_d` in the `heli.m` file, and repeat the experiment with these new values. Do this until you obtain *qualitative* agreement between your simulated and actual responses.

Once you are satisfied with your response, call over a TA to tick-off this exercise.

Exercise 2: Modelling the Power Electronics

The aim of the following two exercises is to model the separate sub-systems that make up the overall helicopter continuous-time system, and connect them together to complete the open-loop model. We will begin with the power electronics.

- To begin with, open the `heli.m` file ready for editing and uncomment the lines for Exercise 2.
- Next, open the SIMULINK model `OpenLoopSys.mdl`, then open the block `Helicopter Continuous Time Dynamics` and double-click on the block named `Power Amplifier`. Complete this block diagram by considering the simplified schematic of the power amplifiers presented in the lectures. In the `heli.m` file, define the values of the amplifier gain `k_a` and upper/lower saturation limits, `amp_sat_u` and `amp_sat_l`, respectively.

Exercise 3: Modelling the Fan Dynamics.

Next we model the electro-mechanical actuation provided by the fans.

- To begin with, open the `heli.m` file ready for editing and uncomment the lines for Exercise 3.
- Next, open the SIMULINK model `OpenLoopSys.mdl`, followed by the block `Helicopter Continuous Time Dynamics` and double-click on the block named `Fan Assembly`. Complete this block diagram using appropriate blocks from the library browser. In the `heli.m` file, specify the voltage/thrust characteristic by defining the fan voltage and thrust data vectors, `V_ab` and `Fss_ab`, respectively, as well as the fan first order time constant `tau`.
  - ACS6110 students: Use values of `V_ab` and `Fss_ab` given to you in lecture handouts. To determine the time constant `tau`, you will need to refer to the fan step response characteristic in Appendix A. This shows a plot of fan force against time, together with a fitted first order response that gives a reasonable match to the raw data. From the fitted response, use the following equation to determine `tau` (τ):

$$t_{10\%-90\%} = 2.2\tau,$$

    where $t_{10\%-90\%}$ is the ten to ninety percent rise time of the first order response.
  - ACS336/6336 students: Firstly, use values of `V_ab` and `Fss_ab` obtained from your steady state `LabVIEW` fan module experiments. Secondly, write a short `Matlab` script to import and plot your fan step response data (force against time). Then use whatever method you like to identify a first order model that provides good agreement with this data. Justify your choice of model by plotting its step response on the same axes as your raw data (this should look similar to Figure 3 in the back of this handout, and you will be expected to show this to a TA at the end of the exercise). Enter the time constant `tau` of this model into `heli.m`.

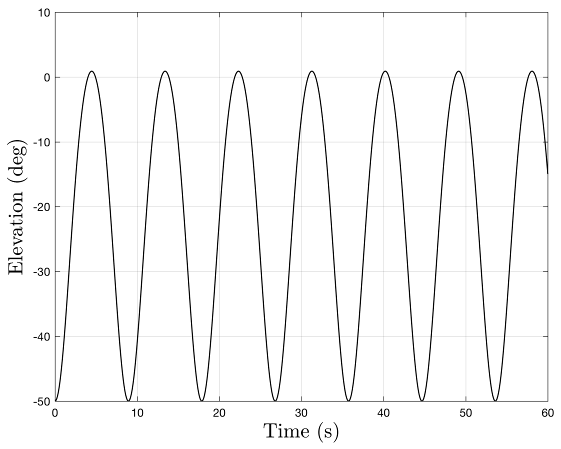

Figure 2: Typical open-loop elevation response of the simulated helicopter with both input voltages set to U_e.

- To complete this exercise, connect the separate subsystems, `Power Amplifier`, `Fan Assembly` and `Rigid Body Dynamics` together. You will also need to copy and paste your `Pitch Dynamics` block from Exercise 1 into the `Rigid Body Dynamics` block, and connect it in an appropriate manner. Lastly, double click on the `Double Integrator` in the `Pitch Dynamics` block, and set `Initial condition x:` to zero.

  Once completed, ask a TA to tick this and the previous exercise off.

Exercise 4: Obtaining the Equilibrium Control Voltage

The ultimate control objective is to stabilise the helicopter about steady, level flight. In order to achieve this, the control signals from the myDAQ will be the sum of two parts; a constant voltage that represents the majority of the effort required to achieve this state, and a smaller, time varying signal to compensate for disturbances. In this exercise we will define the constant (or equilibrium) control voltage.

- To begin with, open the `heli.m` file ready for editing and uncomment the lines for Exercise 4.
- Next, open the SIMULINK model `OpenLoopSys.mdl`, followed by the block `Helicopter Continuous Time Dynamics`, followed by `Rigid Body Dynamics` and double-click on the block named `Elevation Dynamics`. All the library functions have been provided for you, but you will need to connect them together in the correct fashion. In order to do this you will need to refer to the governing equation for the elevation dynamics provided in your lecture handouts.
- Define a small value for the equilibrium control voltage, say `U_e=2`. Modify your SIMULINK block diagram `OpenLoopSys.mdl` so that the input to both power amplifiers is `U_e`. Simulate the open-loop response of the helicopter and study the response about the elevation axis. You should notice that the helicopter has failed to gain any elevation. Increase `U_e` until the value where any subsequent increase causes the helicopter to saturate at its upper elevation limit. At this value of `U_e`, the simulated helicopter should display an oscillatory response about the elevation axis, similar to that shown in Figure 2. Make a note of this voltage. Note that this is not quite how `U_e` is defined in the notes. Owing to the dynamics of the model it is not possible to find a value of `U_e` that makes the simulated helicopter achieve a stable, hovering position, but this value will be close enough for you to assume they are the same.

- Take a small piece of sticky tape and, holding the pitch-beam level, stick the tape over the front of the elevation beam so as to prevent the pitch-beam from rotating (ask a TA if you're not sure what to do). Now, using your LABVIEW open-loop controller, determine the minimum constant voltage from the myDAQ required to achieve level hover. Set the value of `U_e` in `heli.m` to this value, and ask a TA to tick this exercise off for you. This value of `U_e` is unlikely to be the same as found in the previous step, and you are encouraged to think of the reasons for this.

Exercise 5: Defining a Linear Control Model

The aim of this exercise is to define the remaining parameters required to specify a linear, time invariant state-space model of the helicopter. This model will subsequently be used as the basis for synthesising a stabilising feedback controller.

- To begin with, open the `heli.m` file ready for editing and uncomment the lines for Exercise 5.
- Next, obtain a linear approximation to the steady-state fan voltage-thrust relationships by specifying values for the coefficients α and β in the equation:

  $$F_{\mathrm{ss}_{a,b}}(t) \approx \alpha V_{a,b}(t) + \beta.$$

  Use the supplied plotting commands to compare your approximation to the raw data. Adjust α and β until your are satisfied with your approximation. Note, it is not possible to obtain an accurate approximation over the entire range of fan voltages, but it is important that the approximation be accurate around the fan voltage corresponding to that of the operating point ($V_a = k_a U_e$ in this case).
- You have now specified all the parameters required to construct a state-space model of the form:

  $$\dot{x}(t) = Ax(t) + Bu(t), \quad y(t) = Cx(t) + Du(t),$$

  where the state, control and measurement vectors $x(t)$, $u(t)$ and $y(t)$, respectively, have all been defined in lectures. In `heli.m`, define the system matrices A, B, C and D and get this exercise checked-off by a TA.

Exercise 6: Full state feedback control design

The aim of this exercise is to design a reference tracking controller to command the helicopter to desired angles about the elevation and travel axes. As this system is MIMO and coupled, we will design an LQG controller, and to achieve offset-free tracking, we will also use integral control via the state-augmentation method. To achieve this, we augment the existing state vector with two extra states that define the integral of the difference between the measured elevation (travel) angle and the reference elevation (travel) angle, and design a LQR controller for this augmented system.

- To begin with, open the `heli.m` file ready for editing and uncomment the lines for Exercise 6.
- Copy your completed `Helicopter ContinuousTime Dynamics` block from `OpenLoopSys.mdl`, paste it into `ClosedLoopSys.mdl` and connect it appropriately to the `MyDAQ Discrete Time Dynamics` block. Open the latter and take a few minutes to familiarise yourself with the structure of the controller contained within the `DT Controller` block. Notice the separation structure featuring a Kalman Filter that feeds state estimates into a state feedback controller.
- Your first objective is to design the state-feedback and integral control gains. This can be achieved in a single step by designing a state feedback controller upon an augmented state vector, $x_{\mathrm{aug}}(t)$, defined as follows:

  $$x_{\mathrm{aug}}^T(t) := \begin{bmatrix} \epsilon(t) & \psi(t) & \theta(t) & \dot{\epsilon}(t) & \dot{\psi}(t) & \dot{\theta}(t) & \int(\epsilon_{\mathrm{ref}}(t) - \epsilon(t))dt & \int(\theta_{\mathrm{ref}}(t) - \theta(t))dt \end{bmatrix}^T.$$

The augmented state feedback controller you will design will be a LQR controller, which, if you recall, is synthesised by finding the control input that minimises the cost function J, where:

$$J := \min_{u(\cdot)} \int_0^\infty x_{\text{aug}}^T(t) Q_x x_{\text{aug}}(t) + u^T(t) Q_u u(t)\, dt$$

In `heli.m` you will need to specify the augmented state penalty matrix $Q_x \in \mathbb{R}^{8\times 8}$ and control penalty matrix $Q_u \in \mathbb{R}^{2\times 2}$.

To simplify the design process, you are advised to define Q_x and Q_u to be diagonal matrices. Each diagonal term will then penalise nonzero values of a particular state, or control input. For example, making the $(1,1)$ element of Q_x large will place a large penalty on elevation angle deviations from the horizontal and hence can be thought of as a *proportional* control term on the elevation angle. Similarly, and for example, making the $(5,5)$ element of Q_x large will place a large penalty on the rate of change of the pitch angle, and hence can be thought of as a *derivative* control term on the pitch angle.

- Your second objective is to design the Kalman Filter. To do this, you will need to specify the process noise covariance matrix $R_w \in \mathbb{R}^{6\times 6}$ and measurement noise covariance matrix $R_v \in \mathbb{R}^{3\times 3}$. Again, your are advised to make these diagonal matrices, with the selection of diagonal weights based on sound engineering judgement. Poor choices for these matrices may lead to problems upon applying the controller on the actual hardware, such as creating large offsets between the estimated and measured angles. Once you have specified these matrices, run `heli.m` and observe the closed-loop responses by simulating `ClosedLoopSys.mdl`.

  The aim is to obtain a controller that provides smooth offset-free tracking to the elevation and travel reference signals specified in the `Reference Inputs` block within `DT Controller`. You should also observe the control signals $U_{a,b}$ to ensure they rarely, if ever, saturate. Specific guidelines are provided in the oral-assessment handout.

  Once you are satisfied with the simulated response from your controller, show this to a TA. You are now ready to implement your controller upon the actual helicopter.

Exercise 7: Controller implementation

Uncomment the final few lines of `heli.m`. These lines will save your controller parameters as text files that can be read by a LabVIEW script that will implement your controller. In order to implement your controller in LabVIEW, you will need to follow the instructions in the handout 'Closed-Loop Helicopter Controller' that can be found in the Lab Exercises folder in MOLE.

As a word of caution, don't be surprised if your controller performs poorly at the first attempt, even though the simulated responses suggest otherwise. Your controller is synthesised from a model, and that model is based on a number of assumptions. It is likely that you will need to refine your control design several times before you obtain a satisfactory response. Once you are satisfied with the actual response from your controller, show this to a TA. Don't spend too long tuning the controller with a view towards achieving marginal improvements in performance. As long as the helicopter takes off and achieves stable hover, and can demonstrate reference tracking, then you can get this final exercise ticked-off.

Lastly, the helicopter is a challenging system to control, and the key lies in keeping the pitch angles small, which in turn means not being too demanding about tracking reference travel angles. Good luck.

Optional Additional Exercises for ACS336/6336 students

For those of you who have finished the above exercises in a timely fashion and want more ACS336/6336 goodness in your lives, then read on. The following optional exercises carry no extra marks, but provide some suggestions for more deeply exploring the dynamics of the helicopter, and ultimately improving the performance of the control system. You are free to attempt any or all of the following:

- Fan dynamics: Although we modelled the first order fan dynamics in our simulation model, we did not include these in our state-space model. This meant that our controllers had no knowledge of the time lag between voltage inputs to the fans and thrust output. This can lead to quite a noisy response in the controller outputs (as shown in the lecture notes). However, it is straightforward to include the the fan dynamics in the state-space model by defining two additional states (one for each fan) that model the first order response. In order to do this, you will need to convert your fan transfer functions into first order state-space models (e.g. using the `tf2ss` command in MATLAB), and augmenting these into your existing state-space model.
- Reference filtering: The existing step elevation and travel reference signals enter the control loop without any filtering. This can give rise to sudden and large changes in the controller outputs. For instance, the control signals typically saturate at take-off when a sudden large increase in elevation is requested, whilst travel commands of greater than 180° typically destabilise the helicopter. You may wish to experiment with passing the reference signals through a low-pass filter (e.g. a first order transfer function) in order to smooth any sudden demands.
- When the helicopter takes off, the pitch axis will always pitch to one side, and will do so even if both fans are identical. You are encouraged to think about why this is. It is possible to model this phenomenon through a re-derivation of the Euler Lagrange equations, taking the rotational inertia of the fans into account. The resulting dynamics of this phenomenon are linear and can therefore be incorporated easily into the existing state-space model. The resulting controller will thus attenuate the initial pitching activity.
- The elevation axis is rigidly fixed to the shaft of the potentiometer that measures the elevation angle. Potentiometers possess a mechanical resistance to rotary motion, which results in the helicopter falling less freely under the influence of gravity. How might you modify the governing equation for motion about the elevation axis to include this effect, and how might you modify your simulation model accordingly? If the dynamics of this resistive torque are linear, then they can easily be incorporated into the existing state-space model, and so the resulting closed-loop responses between the simulation model and the hardware will be in closer agreement.
- Referring to the governing equations for the rigid body mechanics, note that the forcing from the fans appears either as the sum $F_\mathrm{a} + F_\mathrm{b}$ (elevation and travel axes) or as the difference $F_\mathrm{a} - F_\mathrm{b}$ (pitch axis). By redefining the inputs as $U_+ := U_\mathrm{a} + U_\mathrm{b}$ and $U_- := U_\mathrm{a} - U_\mathrm{b}$ then we can significantly reduce the dynamic coupling between inputs and outputs. This then justifies the use of SISO control techniques such as PID controllers, specifically, PID controllers for control about the elevation and travel axes, and a PD controller for control of the pitch axis. However, note that the use of derivative control still necessitates the use of a state observer to provide the angular derivatives. The challenge is to replace the state feedback controller with a separate SISO controller for each axis. This may seem contrived, but for industrial applications the use of SISO control is greatly preferred, owing to its conceptual simplicity.
- The closed-loop responses included in your lab reports only show the simulated responses. It would be interesting to plot these on the same graphs as your actual closed-loop responses for comparison. In order to do this, you will need to write extra LABVIEW and MATLAB programmes to firstly, save the actual closed-loop response data to file, and secondly, import this data into MATLAB for plotting.

A Fan step response data and fitted first order model

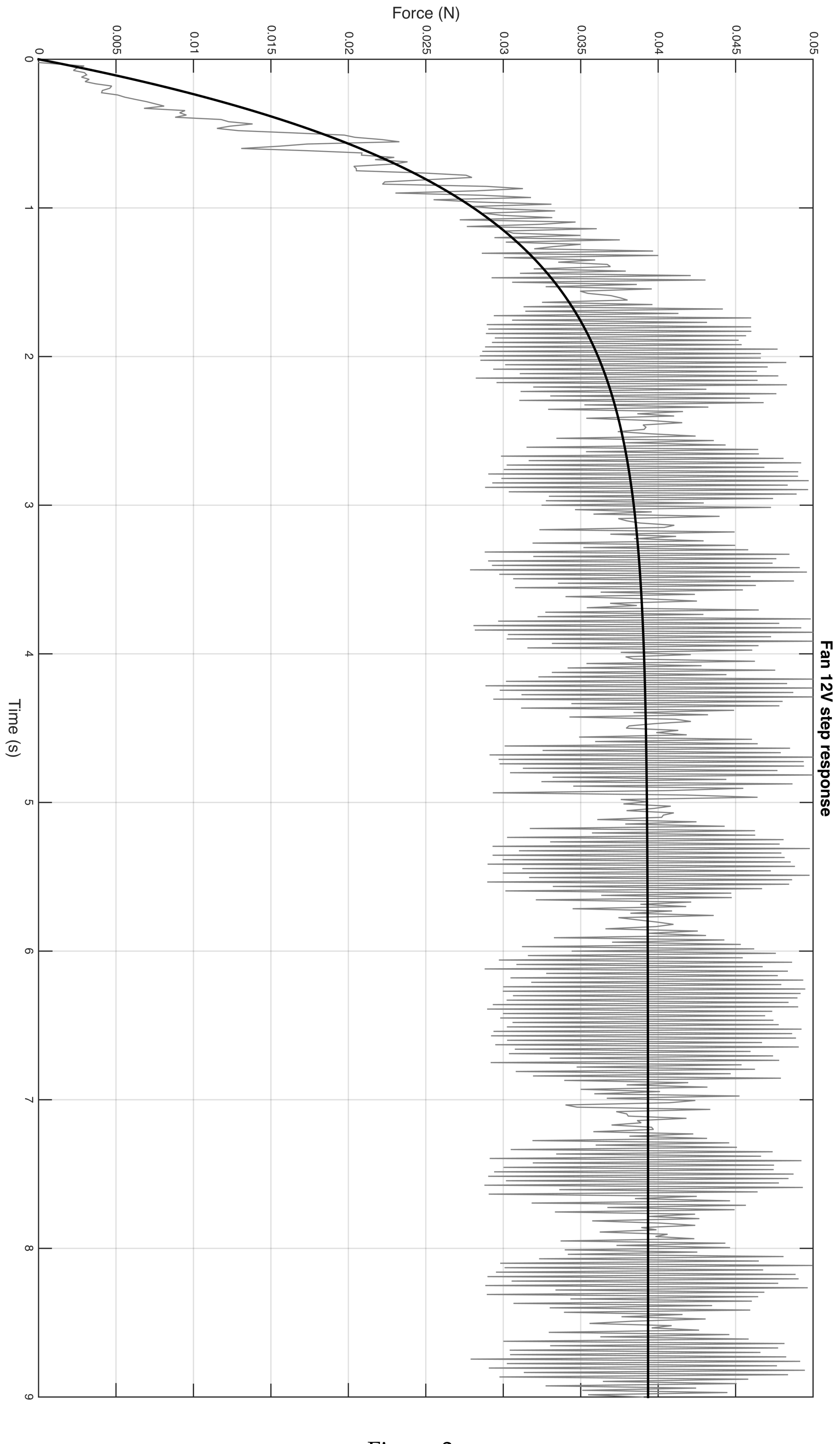

Figure 3: