Computationally Modeling and Optimizing a Tuned Mass Damper

Steven VanCamp, Patrick Tutt
Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48824

(Dated: April 28, 2023)

I. ABSTRACT

Large, man-made structures such as bridges or skyscrapers face many problems associated with ensuring stability and safety. One option to counteract oscillations induced by external factors is a tuned mass damper (TMD). The goal of this project is to present a mathematical formulation of a TMD for a multi-story building using ordinary differential equations, create a simulation to model the movement of each floor, and then optimize the parameters associated with the TMD for the largest reduction in oscillation amplitude for the top floor. Techniques such as RK4 will be used for numerical integration and Markov Chain Monte Carlo will be used to optimize this problem.

II. INTRODUCTION

Tuned mass dampers (TMDs) are mechanical devices that are used to reduce the vibrations of structures subjected to dynamic loads. Harmonic absorbers or seismic dampers are other names for these types of devices. They are typically used in tall buildings, long-span bridges, and offshore structures that are exposed to wind, earthquakes, and other environmental forces. These external factors induce vibrations that can cause harm to the building structure and jeopardize the safety of occupants. Structures like bridges or buildings have a natural frequency at which they oscillate. Coupling this fundamental mode with excitation from an earthquake or other factor, the building may be excited to a higher-order mode where vibrations are more dangerous. The basic principle of a TMD is to dampen the harmonic motion of a structure by having a component of the structure oscillate out of phase with the harmonic motion. The desired effect is to see a reduction of the oscillation amplitude in time. [1].

The first ever vibration damping device, which was an early form of the tuned mass damper, was shown in 1909 by a patent filed by Heramann Frahm entitled dynamic vibration absorber. Extensions of Frahm's work were done by Ormondroyd and Den Hartog in 1928. They demonstrated that the quality of the vibration absorber's performance could be greatly improved by adding damping elements [2]. From 1909 to today, TMDs have grown in complexity and applicability. Figure 1 shows a TMD installed in the Taipei 101 skyscraper in the capital city of Taiwan. The left image shows the TMD installed at the top of the skyscraper and demonstrates its suspension to the frame of the building. One of the many advantages of TMDs is their small mass in comparison to the overall size of the structure they are installed within. This is especially true for Taipei 101, where the TMD weighs 660 metric tons. The right image shows a close-up display of the TMD. Fundamentally, the TMD here can be represented as a mass coupled with a spring. To begin our computational model of a TMD, this basic model is where we will start. From there, we will extend the mathematical description and computational model to a multi-story building.

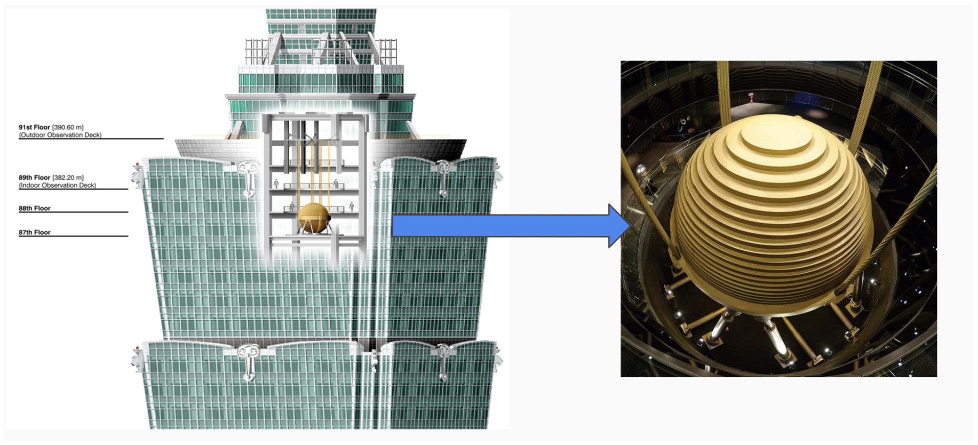

FIG. 1. Diagram of Taipei 101 and its TMD. [3].

The structure of this report is as follows: The concept of tuned mass damping has been introduced in II and will be subsequently motivated in II.1. The methodology of the model in terms of mathematical approach, computational modeling, and optimization will be discussed in III. In section IV, the results of the simulation will be discussed. Concluding remarks will then be presented in V. References are included in VI. Supporting images, code, and workload description are contained in I.

II.1. Motivation

As discussed above, TMDs can be a powerful method for damping oscillations in various types of structures. There are four main ways that systems benefit from reduced oscillations via the tuned damper.

First, there is a reduction in fatigue and damage to the structure. Vibrations can cause fatigue and damage to structures over time, which in turn can reduce their lifespan and increase maintenance costs. TMDs can help reduce the amplitude of vibrations and prevent long-term damage to the structure. Second, the comfort of occupants on or within the structure can be improved by damping oscillations. It is logical to assume that by decreasing these amplitudes the driver on a bridge or a worker within a tall structure will feel safer. Third, they increase safety. In extreme cases, excessive vibrations can lead to structural failure and collapse, posing a significant safety risk. TMDs reduce the likelihood of these events by increasing the force required to perturb the building. Fourth, there can be a reduction in construction costs. Incorporating TMDs into the design of a structure can help reduce construction costs by allowing for lighter and more economical designs. By reducing the size and weight of the structure, TMDs can also help minimize the number of materials required for construction. Overall, a TMD has a wide range of advantages, stretching from building cost and design to the safety and comfort of occupants or users.

As discussed above, TMDs have been a part of damping structural oscillations for some time. The construction of the Citigroup Center in 1977 contained one of the largest TMDs at the time, weighing nearly 400 tons. Additional motivation can be found for the use of TMDs by considering the events following the completed Citigroup Center. In 1978, serious structural flaws were discovered in the design and construction of the Citigroup Center. The firm responsible for the structural engineering of the building found that with a TMD present in the building, the structural concerns were limited; however, if an event occurred causing the Citigroup Center to lose power and no longer be able to operate an active TMD, then structural integrity was in extreme jeopardy because of the structural flaws. This example goes to show the importance of having a TMD not only to reduce oscillations but also to counteract possible structural flaws during construction[4];[5].

With the need to understand TMDs properly motivated, a discussion of the important physical questions we want to answer can be had. Our goals in this project are as follows:

- Can we model a simple coupled-mass spring system with damped oscillations?
- Can we extend this model to a generalized case of a one-dimensional building?
- Can we optimize the parameters of the TMD for this one-dimensional building case?

III. METHODOLOGY

III.1. Simple coupled mass-spring system

In order to model a building with a TMD, we begin by considering a simpler model to use as a building block for our later work. The simplest case of a TMD is that of a coupled mass-spring system. This system, as shown in Figure 2, consists of two masses and two springs. The first mass m_1, serves as our "building" in this simple case. m_2 will serve as the TMD. When m_1 is either compressed or stretched, the mass will begin to oscillate. We can imagine that if the second mass was not present and there was no energy loss due to friction or other effects, the mass would continue to oscillate back and forth forever as energy is interchanged between kinetic and potential forms. Even with some natural damping from friction that may reduce the amplitude of oscillation, additional damping is needed as motivated by earlier sections. m_2 provides this additional damping.

The spring constants k_1 and k_2 each represent a coupling of the masses. k_1 is the coupling of m_1 to a stationary point. This can be thought of as the coupling of a building to the ground. k_2 represents the coupling of each of the masses to each other. This implies that in order to achieve the best damping possible, both the mass of the TMD and the spring constant associated with its coupling to the building are important to optimize. Values b_1 and b_2 are

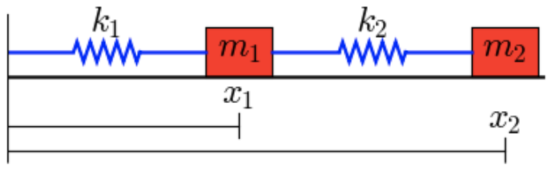

FIG. 2. A coupled mass-spring system [6].

damping constants for each mass and spring constant. Tuning this damping factor is also essential for optimizing to a low oscillation amplitude.

In order to apply numerical integration techniques such as Runge-Kutta 4 (RK4), the pictorial representation of this system must be translated into a mathematical form [7]. Two equations, one for the motion of x_1 and one for the motion of x_2, can be written down from the realization of Figure 2 [6]. These equations are second-order ordinary differential equations (ODEs) and are shown below:

$$\begin{aligned} m_1\ddot{x_1} + b_1\dot{x_1} + k_1(x_1 - L_1) - k_2(x_2 - x_1 - L_2) = 0 \\ m_2\ddot{x_2} + b_2\dot{x_2} + k_2(x_2 - x_1 - L_2) = 0 \end{aligned} \tag{1}$$

In this case, L represents the respective displacement from the equilibrium position for each mass. It is important to note that these equations are not independent of one another but rather the position of one mass affects the equations of motion of the other. This can be seen by recognizing that in the equation for the motion of x_1, there is a term containing x_2. The same relation can be seen in the equation of motion of x_2. In order to solve these second-order ODEs, the system must be uncoupled into a series of first-order ODEs. From here, numerical integration methods may be used to solve these uncoupled equations. In our case, the odeint python package will be sufficient; however, the RK4 method will be utilized for the more complex building description later on.

III.2. A 1D Building Model

With the structure of the simple coupled mass-spring system in mind, the mathematical model can be extended such that we are considering a multi-story building. The model of a multi-story building can be seen in Figure 3. At the top of the structure, the TMD device is installed and the mass oscillates against some fixed boundary. In our model, parameters m_d, the mass of the damper, k_d, the spring constant of the damper, and c_d, the stiffness of the damper, will be the optimized for best reduction in oscillation amplitude. It should be noted that in reality, the TMD does not sit on top of the structure. Considering Figure 1 of the Taipei 101, it is shown that the TMD sits close to the top floor, but sits several floors below. In our computations and models, we will make the assumption that the TMD sits on the top floor for simplicity.

Translation of the model in Figure 3 to a mathematical form can be realized when considering how each of the individual floors is modeled. Each floor has its own mass, spring constant, and stiffness parameter to represent coupling between floors. For simplicity, we will assume the parameters of each floor are identical such that:

$$\begin{aligned} m_1 = m_2 = m_3 = ... = m_n \\ k_1 = k_2 = k_3 = ... = k_n \\ c_1 = c_2 = c_3 = ... = c_n \end{aligned} \tag{2}$$

where n refers to the number of building floors. Considering each floor and the damper separately, one can imagine writing down a differential equation for the motion of each floor and the damper; however, similar to the coupled spring-mass system, the equations of motions for one specific floor or damper will contain position terms from other floors. Due to this coupling, a more complicated description of motion is required. From a substantial literature review, the following mathematical description can be used to model a one-dimensional building:

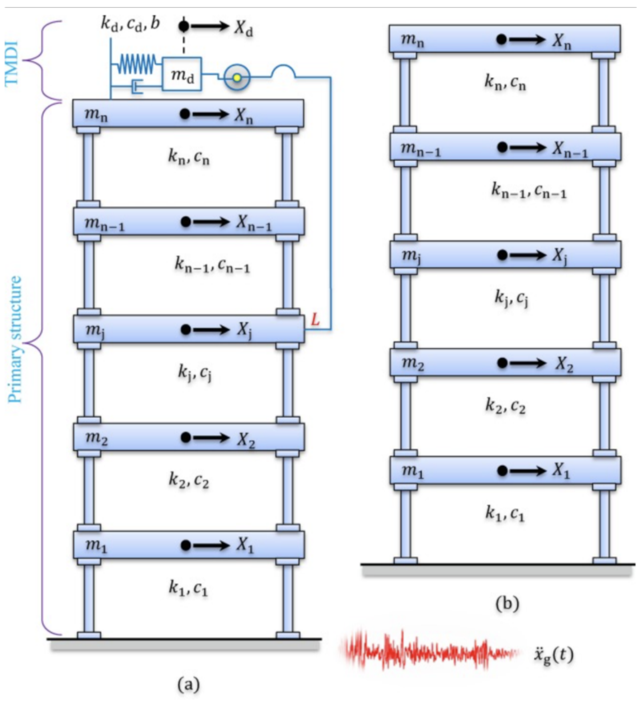

FIG. 3. 1D building model [8].

$$M\ddot{x}(t) + C\dot{x}(t) + Kx(t) = -m\ddot{x}_g(t) \tag{3}$$

where M, C, and K are matrices that contain the associated mass, stiffness, and spring constant for each floor and the damper. Each of these matrices also contains information about how the coupling between floors functions. A general form of the K and C matrices is shown in Figure 4. Examining the diagonal of each matrix, we can see that the diagonal is equal to the current floor parameter added to the parameter of the floor above. Looking at the off-diagonal terms, the parameter of the next floor is contained there with a negative sign. The rest of the matrix is sparse with zeros. These diagonal and off-diagonal terms represent the coupling of each floor to one another. The only exception to this trend is the bottom right-hand parameter which contains the associated parameter for the TMD.

$$\mathbf{K} = \begin{bmatrix} (k_1+k_2) & -k_2 & & & & \\ -k_2 & (k_2+k_3) & -k_3 & & 0 & \\ & -k_3 & . & . & & \\ & & . & . & -k_n & \\ & 0 & & -k_n & (k_n+k_d) & -k_d \\ & & & & -k_d & k_d \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} (c_1+c_2) & -c_2 & & & & \\ -c_2 & (c_2+c_3) & -c_3 & & 0 & \\ & -c_3 & . & . & & \\ & & . & . & -c_n & \\ & 0 & & -c_n & (c_n+c_d) & -c_d \\ & & & & -c_d & c_d \end{bmatrix}$$

FIG. 4. Spring constant and stiffness coupling matrices [9].

The matrix M only has elements along the diagonal. These terms are just the masses of each floor and the damper mass. The vector m is simply a row vector containing each of the masses for the floor and damper mass similar to the matrix M. The only remaining piece of this description is the force perturbing the system so that the TMD can dampen the resulting oscillations. This perturbing force is contained within the $\ddot{x}_g(t)$ term. In our case, this term represents the ground acceleration due to an earthquake. For all of the results contained in section IV, the signal from an earthquake in El Centro, California in 1940 was used. This acceleration data is displayed in Figure 5.

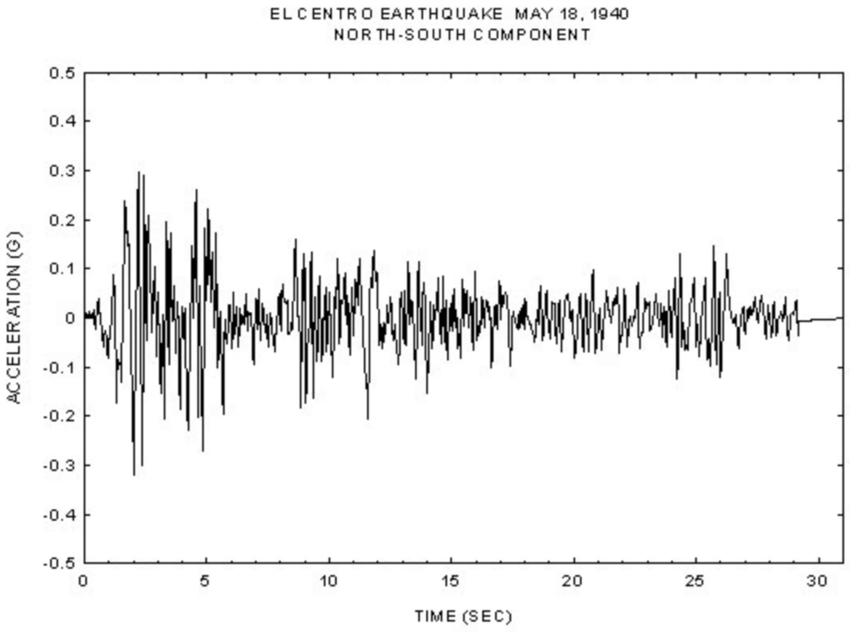

FIG. 5. Ground acceleration from a 1940 earthquake in El Centro, California. [10]

III.3. Markov Chain Monte Carlo Optimization of TMD

As part of our project, we sought to find the optimal value of the parameters that define the TMD. These parameters being m_d, k_d, and c_d. Each parameter represents a different property of the TMD and as such affects how it behaves in the system differently. Due to the complexity of modeling these systems, there is no simple analytical solution to this problem, and as such numerical optimization techniques must be used. The primary technique used to complete the optimization of our problem was Bayesian Markov Chain Monte Carlo (MCMC).

MCMC was chosen as it is a robust and well-understood technique. It has many types of implementations each designed for a different type of problem. In our case, we choose to use a Metropolis-Hastings algorithm (MHa) for the core of the MCMC algorithm. The MHa is used to generate a Markov Chain with the desired posterior distribution from which we pull samples used in the MCMC algorithm. Since we are optimizing 3 parameters, each with a well-understood parameter space we don't need a complex sampling algorithm. We also want to avoid overhead as much as possible as every MCMC iteration will require a new building simulation to be run. This is why we've chosen to use the MHa.

In some cases, MCMC algorithms can be designed to include prior knowledge about the system or problem. This generally limits the parameter space so that nonphysical or previous knowledge about the shape of the parameter space can be taken into account. When implemented in this way, this information is known as a prior. This is a Bayesian technique, which allows for the inclusion of previous knowledge attained in experiments or research in our optimization algorithm. The priors used in our project are discussed in section III.3.1.

When using an MCMC algorithm to optimize a problem a definition for the optimal set of parameters must be defined, before picking them from the posterior distributions generated by the MCMC algorithm. In our case we defined the optimal set of parameters to be those defined by the single point/bin in the 3-dimensional posterior distribution that was "visited" the most during the MCMC algorithm.

III.3.1. Priors

The priors used in the MCMC algorithm are discussed in this section. Our priors take into account the physical aspects of the TMD as well as some practical limits. Due to physical limits each parameter m_d, k_d, and c_d must be a positive real number. The basis of our simulated system was developed from [9], as such we also include their optimal parameter choices as a broad baseline for our own optimization algorithm. We chose to use gaussian distributions, bounded to be above 0, as the underlying prior distribution for each parameter. The means were chosen to be the optimal values presented in [9] for the case of a 10-story building. Equation 4 shows the form of the gaussian used in our algorithm. Where x is the parameter guess, x_σ is the gaussian variance, and x_g is the gaussian mean.

$$prior(x) = \begin{Bmatrix} x < 0 : 0 \\ x >= 0 : (2\pi x_\sigma)^{-0.5} \exp - \frac{(x-x_g)^2}{(2x_\sigma)^2} \end{Bmatrix} \tag{4}$$

IV. RESULTS AND DISCUSSION

As discussed above in section III, there are two relevant models. The first is a coupled mass-spring system and the second is a more complicated formulation of a one-dimensional building. Simulations were run for both systems and the position of the damper and floors (when relevant) are plotted. Minimal results are shown for the mass-spring system, as this model only serves as a way of understanding the basic principles of a TMD. Multiple simulations were run for the building case and optimization was done to find the best parameters of the damper (m_d, k_d, and c_d).

IV.1. Simulation of coupled mass-spring system

For a coupled mass-spring system, python tools like odeint are capable of handling the decoupling of second-order ODEs to a series of first-order ODEs. Simulation results for this system are based on equations 1 and are contained within Figure 6. In this simulation, the values of the relevant parameters are shown below:

$$\begin{aligned} &m_1 = 10.0; m_2 = .3 \\ &k_1 = 8.0; k_2 = 3.0 \\ &b_1 = .2; b_2 = .8 \end{aligned} \tag{5}$$

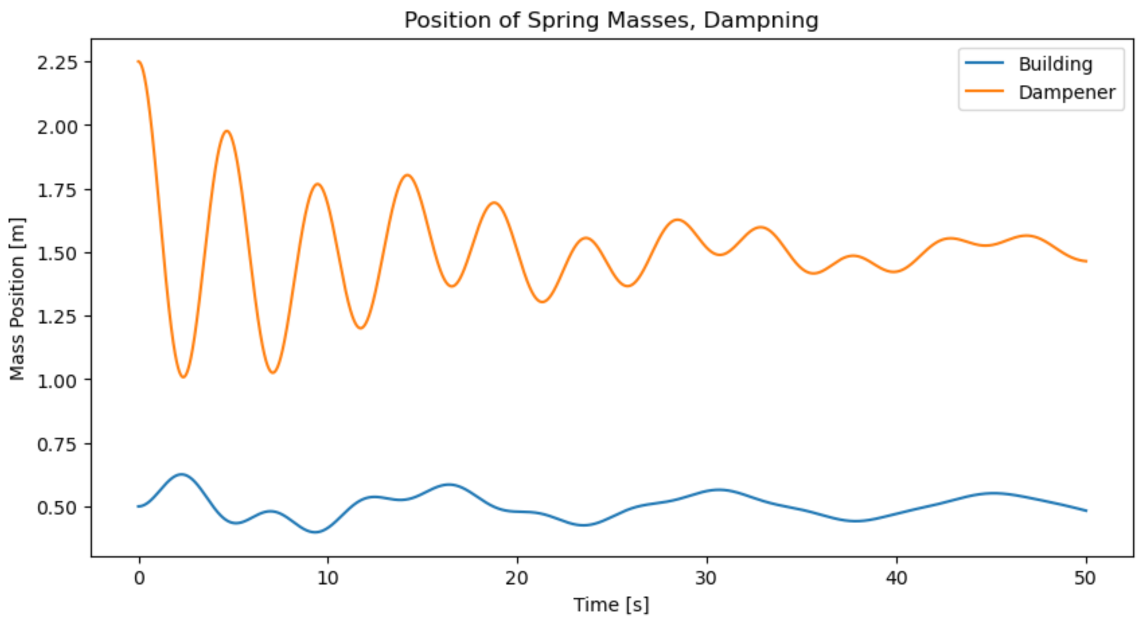

FIG. 6. Simulation result for the coupled mass-spring system.

Complex analysis and optimization are not required for this case since we are merely trying to observe qualitatively if the damper is working. To do this, we observe the behavior of the "building" which is represented by m_1 in our pictorial representation in section III.1. Just after $t = 0$, the building reaches the highest amplitude of the entire plot. As time is allowed to progress, we see that the amplitude of the building is damped from the maximum amplitude it originally reaches. In combination with this, we see that the damper also begins to oscillate less over time. With these two pieces of information together, we can confidently say that the damper is working, although it may not be the most effective damping.

IV.2. Optimizing Dampener Parameters for the Case of a 1D Building

As part of our project, we are optimizing the parameters that define the behavior of a TMD coupled to a multi-story building. To complete this an MCMC algorithm is used following the methodology described in section III.3. This does require that the simulation as described in section IV.3 is complete and working so that the results can be compared to a desired state at each MCMC iteration. For our problem, we defined the desired state of the system to be where the top floor displacement is described by Eq.6. Where ΔD_{top} is the displacement of the top floor.

$$\Delta D_{top}(t) = 0 \tag{6}$$

The required inputs for the MCMC algorithm are as follows: N_{iter}, N_{burn}, σ_{data}, p^i_{prior}, p^i_{var}, p^i_{guess}, p^i_{δ}. N_{iter} is the number of iterations the MCMC algorithm will take and N_{burn} is the number of iterations (as a percentage) the algorithm will throw away at the beginning as the walker is finding the optimal region. These were set to $1E5$ and 10% respectively. σ_{data} is the estimated error on the desired state, and was set to 0.2. The remaining parameters are required for each TMD parameter being optimized, so in our case m_d, k_d, and c_d. These describe the prior related to that parameter and the initial guess and how much the MCMC algorithm can vary that parameter each iteration. The parameters used in our project are shown in Table I.

Parameters	p^i_{prior}	p^i_{var}	p^i_{guess}	p^i_{δ}
$m_{\%tot}$	0.05	0.03	0.03	0.005
k [MN/m]	3.7	1.5	3.0	0.005
c [MN - s/m]	0.197	0.10	0.12	0.005

TABLE I. Parameters used in our MCMC algorithm for the optimization of the TMD

The results from our MCMC algorithm are shown as a triptych in Fig.9. Each component of the figure is a posterior distribution related to 1 or 2 parameters. It can be seen that the parameter space defined by c and m, which is shown in the bottom left of Fig.9, has a posterior distribution that is well-defined. However, the posterior distribution of k is not well defined. This makes picking the optimal parameter values slightly more complicated.

To facilitate picking the optimal parameter values a full parameter space was created and binned. Since we have 3 parameters a 3d visualization can be made, this is shown in Fig.10. The bins were assigned a value that corresponds to the number of points within the bin volume. The optimal parameter values were chosen as those defined by the bin with the highest value. This corresponds to the parameter space that the MCMC algorithm spends the most time in. The optimal dampener parameters were found to be the following: $m_d = 0.0453$, $k_d = 2.7758$, $c_d = 0.2124$.

IV.3. Simulation of 1D Building

With parameters m_d, k_d, and c_d optimized using MCMC, comparisons can be made between simulation results for the 1D building case. In each case, ten building floors were simulated, including the damper. Figure 7 contains the 1D displacement of the top floor for a "no damp" case and a case where reference parameters are used. These reference parameters were obtained from a literature review and were found to be the best parameters using an optimized transfer function [9]. The reference parameters taken from the literature are contained in Table 2.

Note that "floor" refers to the mass, spring constant, and stiffness factor for all floors but the damper. Also note that the m_d value is not optimized, but rather is taken to be 3% of the total building mass. Investigating Figure 7 more closely, qualitatively it can be seen that the no dampener and dampened case are relatively similar for the first

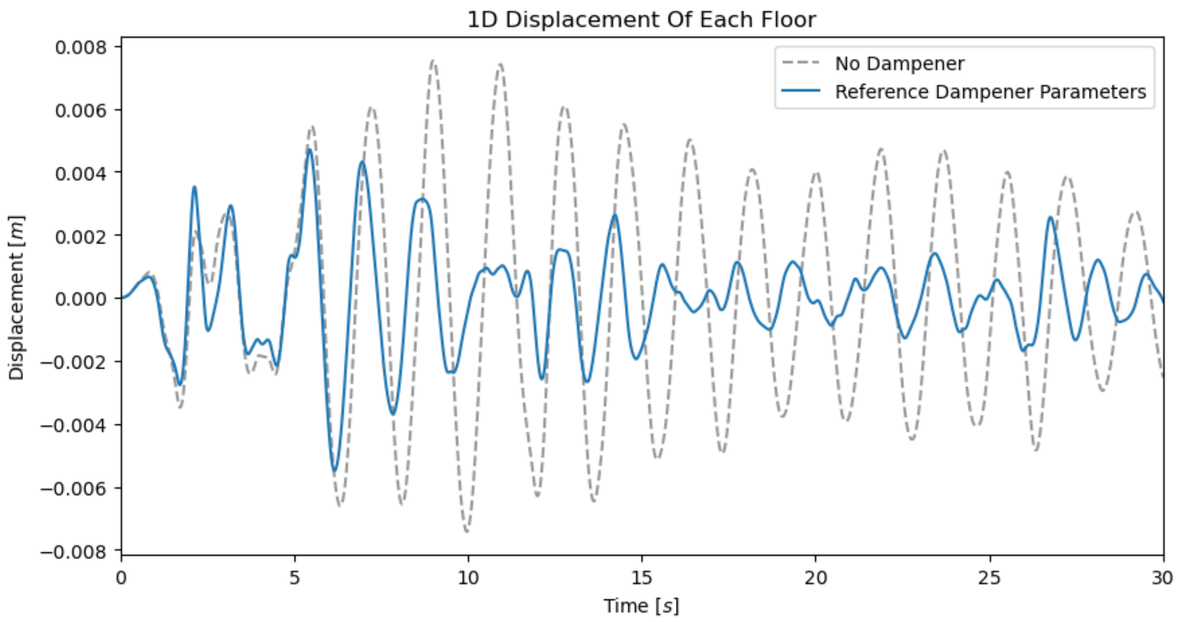

FIG. 7. Simulation result for reference parameters and undamped case.

Parameters	Damper	Floor
m [tons]	108	360
k [MN/m]	3.7129	650
c [MN - s/m]	6.2	.197

TABLE II. Parameters from literature [9].

5 seconds. There doesn't appear to be any significant reduction in amplitude during this period. After this initial 5 seconds, the dampening effects are clear. The dampened case, colored in blue, shows significant amplitude reductions compared to the no-dampener case. These reductions can be as large as 0.008 meters like at the t = 10 seconds mark where the dampened building position at the top floor is nearly 0. Reduction in oscillation amplitude is seen from beyond 5 seconds until the end of the simulation run at 30 seconds.

Using the optimization result from section IV.3, the simulation was rerun and compared with the optimal parameters obtained from literature and the no dampening case. This simulation result is shown in Figure 8. Similar to the previous simulation, within 5 seconds the no damper, reference parameters, and optimized parameters cases all have roughly the same displacement. As expected, as time is allowed to progress from 5 to 30, the no damper case has a much higher amplitude compared to the other two cases. Comparing the reference parameters case to the MCMC optimized case, their behavior is quite similar, especially from 12 seconds onward. The only time there is a major difference in amplitude is 10 seconds when the reference parameters appear to sit around 0 while the MCMC simulation appears to swing back and forth up to 0.003 meters. Despite this discrepancy, the MCMC simulation is clearly offering significant dampening at t = 10 seconds as the no-damped case is nearly 0.008 meters.

For what reasons may the reference parameter simulation perform better than MCMC? During the literature review as briefly mentioned early, the reference values are obtained from a paper that used the optimization of the transfer function in order to determine what parameters were best for dampening. This optimization style was not attempted in our simulations as MCMC was preferred due to our knowledge of the method. Additional analysis could be done regarding the transfer function to determine if slightly better parameters could have been selected.

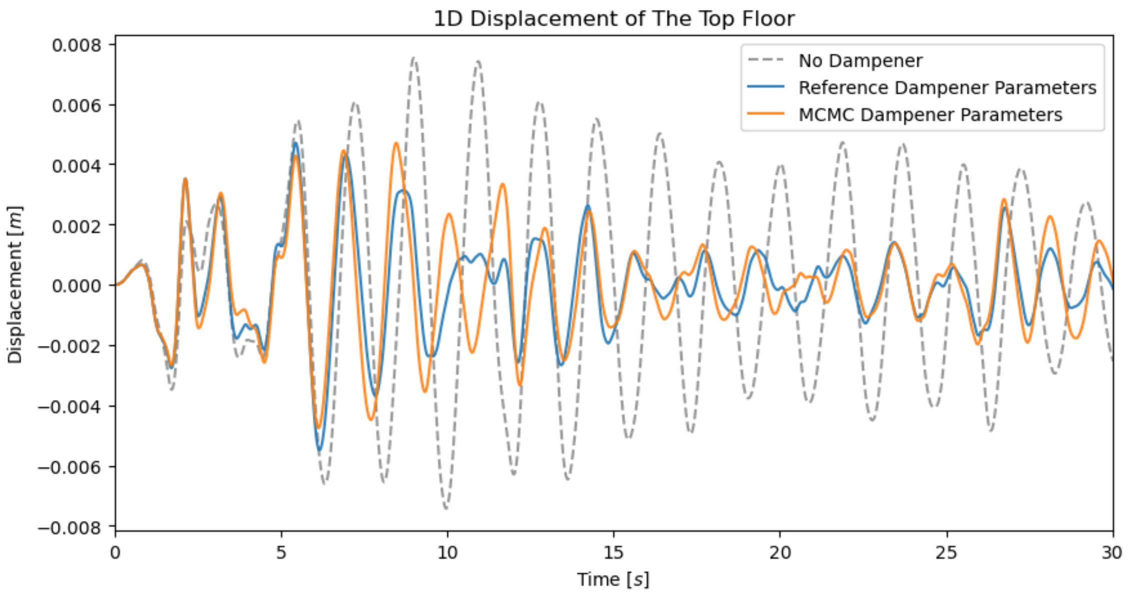

FIG. 8. Simulation result for the optimized parameters.

V. CONCLUSIONS AND PERSPECTIVES

In this project, we were able to develop two mathematical models based on literature in order to simulate a tuned mass damper. The first, a coupled mass-spring system, served as a basic model for a TMD. In this case, simulation results show a dampening of oscillation amplitude with time due to the coupling of a small mass to a larger mass. In this example, the larger mass represents our building and the smaller mass represents a TMD. This model was simple but allowed us to explore dampened oscillations and build a mathematical base for our problem. From here, we extended this model to a multi-story building by constructing a more complicated mathematical description. This description included matrices that describe the coupling of each floor to each other and to the dampener. Our simulation of a 10-story building plus a dampener yielded the expected results. Comparing the oscillations of an undampened building to that of a dampened building using optimized parameters from literature, the oscillation amplitude is observed to be much lower for the dampened case. Performing our own optimization utilizing an MCMC algorithm also produced the expected results and compared well with the literature-optimized parameters.

In regard to future work, one area of interest would be the comparison of optimized parameters using MCMC for different numbers of floors. In our simulation, we used a building height of 10 floors plus the dampener. This created matrices that were 11 by 11 in size. Increasing the number of floors to 20, 50, or even 100 to model more modern buildings with TMDs would increase the size of the matrix greatly. Simulating these solutions could be computationally expensive; however, the matrices are mostly sparse with zeros. Another area of interest would be modifying the prior information handed to MCMC. For each parameter, m_d, k_d, and c_d, a gaussian distribution was used. It is possible that different distributions, for example, a uniform one, may produce better simulation results that are closer to the literature parameters.

There are several areas of this project that could be improved for future work. One area of potential improvement, as discussed in the previous section, is to do more research into an appropriate selection of a prior distribution for dampener parameters. As discussed above, the prior was a gaussian distribution for all parameter cases; however, it may be advantageous to look into other distributions to improve the optimization. Reasoning as to why another distribution may be more appropriate requires additional research. Another area of improvement would be to take a deeper look into the parameters provided by the literature and determine if they are still applicable to more modern buildings. It is possible that a more refined mathematical description is required to accurately describe modern buildings that contain more advanced structural elements. A final area of improvement could be to test additional numerical integration methods to see how they compare to both literature and to RK4.

Overall, the project goals of mathematically describing a mass-spring system and 1D building were achieved. Simulation results were run for both cases and agreed with expectations and with the literature results. Consequently, we have shown the importance of TMDs in dampening oscillation amplitudes induced by external forces.

VI. REFERENCES

[1] Chapter 4 - tuned mass damper systems.
[2] Structural control chapter 13 – tuned-mass dampers.
[3] Taipei 101 (2023).
[4] C. Whitbeck and E. Plosky, William lemessurier - the fifty-nine-story crisis: A lesson in professional behavior.
[5] M. P. SINGH, E. E. MATHEU, and L. E. SUAREZ, Active and semi-active control of structures under seismic excitation, Earthquake Engineering & Structural Dynamics **26**, 193 (1997).
[6] Coupled spring-mass system¶.
[7] Runge-kutta algorithm.
[8] S. Djerouni, A. Ounis, S. Elias, M. Abdeddaim, and R. Rupakhety, Optimization and performance assessment of tuned mass damper inerter systems for control of buildings subjected to pulse-like ground motions, Structures **38**, 139 (2022).
[9] S. Özsarıyıldız and A. Bozer, Finding optimal parameters of tuned mass dampers, The Structural Design of Tall and Special Buildings **24**, 461 (2015), https://onlinelibrary.wiley.com/doi/pdf/10.1002/tal.1174.
[10] Vibration data el centro earthquake.

VII. APPENDICES

VII.1. Work Divison

- Report: Patrick, supported by Steven
- Analysis: Patrick, Steven
- Mathematical model development: Patrick, supported by Steven
- Runge-Kutta development: Steven, supported by Patrick
- MCMC development: Steven, supported by Patrick

VII.2. Images

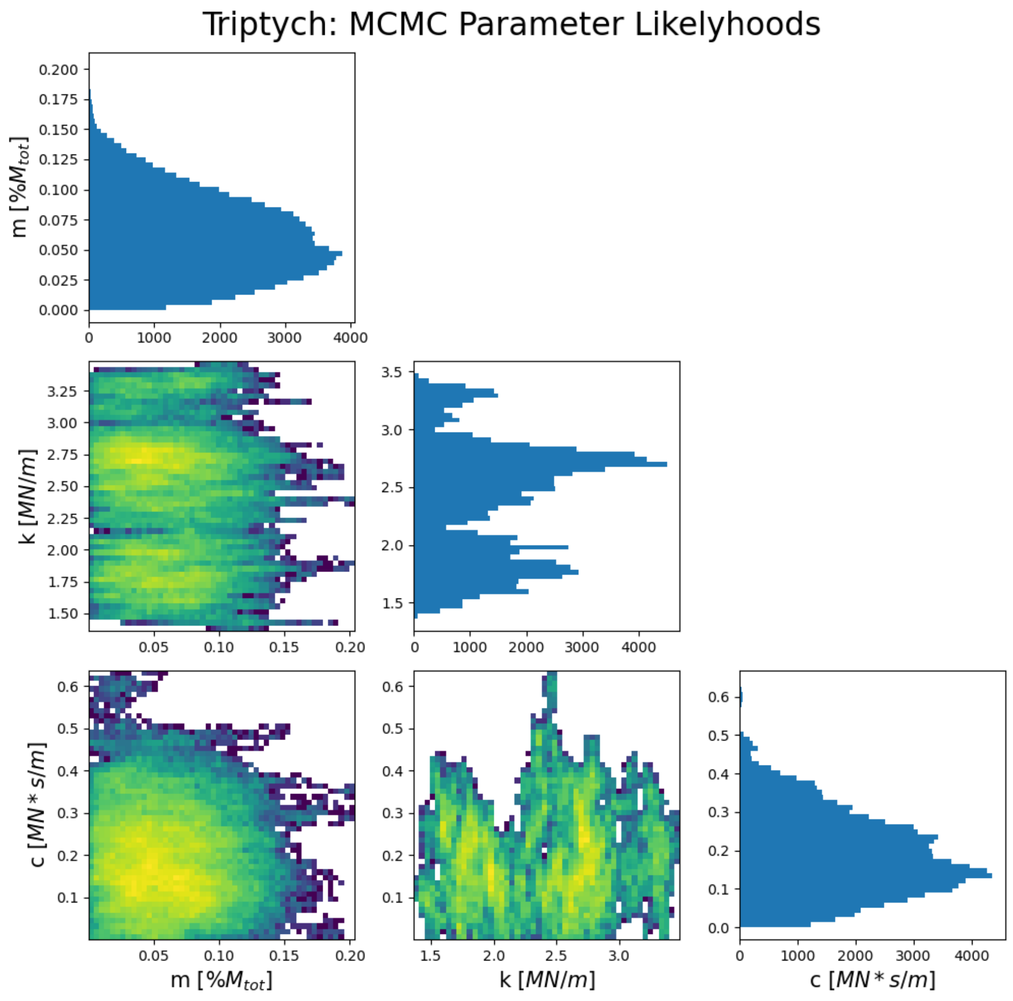

FIG. 9. Posterior distributions generated by an MCMC optimization algorithm for the parameters m_d, k_d, and c_d. Using the priors discussed in III.3.1. Lighter colours represent an area ”visited” more often by the MCMC algorithm

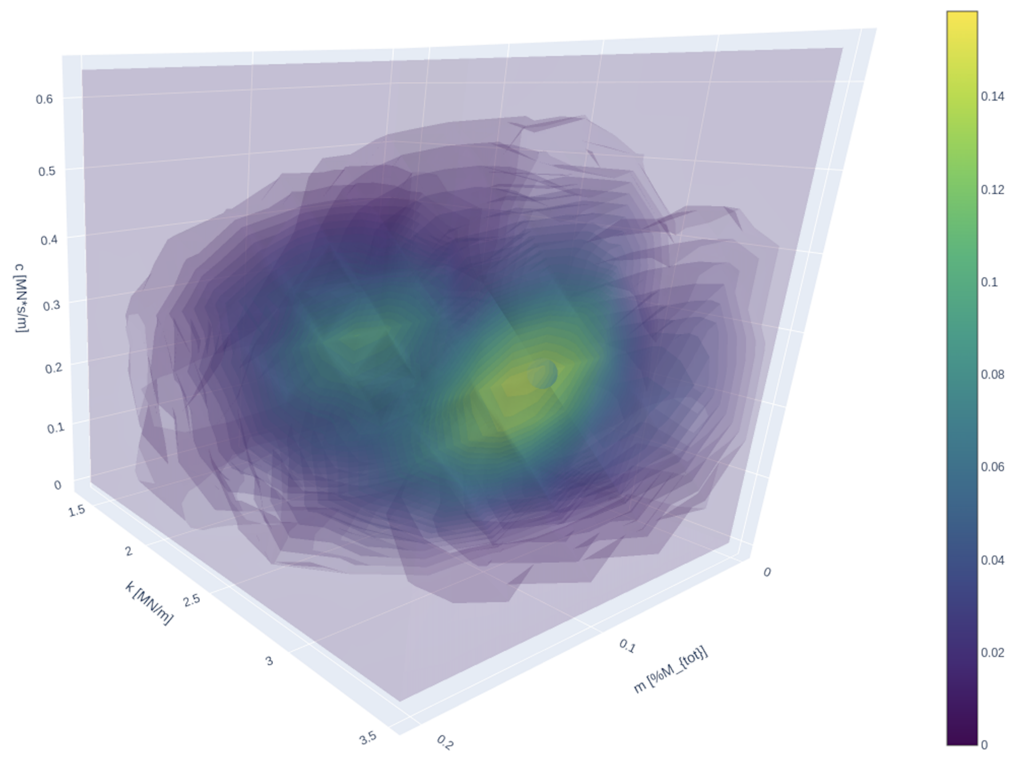

FIG. 10. Posterior distribution space where lighter colours represent an area "visited" more often by the MCMC algorithm

VII.3. Code

The following sections VII.3.1VII.3.2 show parts of our code for the RK4 and MCMC algorithm. The code below shows snippets of our full code and may be missing variable or function definitions. To view the full code please view the source files.

VII.3.1. Runge Kutta 4

The following block of code is a snippet of the runge kutta algorithm. Be aware not all variables used or functions used are shown below. To see the full source code please view the source file *funcs.h*

```
vector<vector<double>> rk4_build(vector<double>& state_vars,
                                 vector<vector<double>>& pars,
                                 vector<double>& t,
                                 double h) {
    /*
    rk4 algorithm

    inputs:
        state_vars : vector of the state variables;
                     vector<double> [x1,y1,xv_2,yv_2]

        spring_pars : vector of the spring parameters;
                      vector<double> [m1,m2,k1,k2,L1,L2,b1,b2]

        t : vector contaning timing data; vector<double> [...]
        h : rk4 stepsize; double

    returns:
        p_hist: state variable history; vector<vector<double>>
                [[x1,y1,x2,y2...]...]
    */

    // initialize the time array [a,b] with stepsize h

    int n_times = t.size();
    int n_vars = state_vars.size();

    vector<vector<double>> p_hist(n_times, vector<double>(n_vars, 0));

    vector<double> k_1, k_2, k_3, k_4;

    vector<double> temp;
    vector<double> temp_o;

    p_hist[0] = state_vars;

    for (int i=0; i<n_times-1; i++) {
        // calc k_1
        k_1 = build_f(p_hist[i], pars, t[i]);

        // calc k_2
        temp = vec_mult_scal(k_1, h/2);
        temp = vec_add(p_hist[i], temp);

        k_2 = build_f(temp, pars, t[i]);

        // calc k_3
        temp = vec_mult_scal(k_2, h/2);
        temp = vec_add(p_hist[i], temp);

        k_3 = build_f(temp, pars, t[i]);

        // calc k_4
        temp = vec_mult_scal(k_3, h);
        temp = vec_add(p_hist[i], temp);

        k_4 = build_f(temp, pars, t[i]);

        // calc x+1
        temp_o = vec_mult_scal(k_2, 2);
        temp = vec_add(k_1, temp_o);

        temp_o = vec_mult_scal(k_3, 2);
        temp = vec_add(temp, temp_o);

        temp = vec_add(k_4, temp);

        temp = vec_mult_scal(temp, h/6);

        p_hist[i+1] = vec_add(p_hist[i], temp);
    }

    return p_hist;
```

```
}
```

VII.3.2. Markov Chain Monte Carlo

The following block of code is a snippet of the algorithm used in the MCMC algorithm. Be aware not all variables used ore functions used are shown below. To see the full source code please view the source file *optim_building.cpp*

```
// initialize variables for use in mcmc loop
vector<vector<double>> p_g_hist;
vector<double> sim_err_hist; // history of the error in the top floor position

vector<double> p_g_new (p_g_old);

double p_factor;
double p_prior_old;
double p_prior_new;
double p_accept;

vector<double> sim_vals_new(sim_vals_old);
double sim_err_new;

default_random_engine rand_gen;

normal_distribution<double> md_norm_dist(0.0,p_d[0]);
normal_distribution<double> kd_norm_dist(0.0,p_d[1]);
normal_distribution<double> cd_norm_dist(0.0,p_d[2]);

uniform_real_distribution<double> r_accept(0, 1);

// begin mcmc loop
int iter_count = 0;
for (int i=0; i<mcmc_points; i++) {

    // update optimization parameters
    p_g_new[0] = p_g_old[0] +  md_norm_dist(rand_gen);
    p_g_new[1] = p_g_old[1] +  kd_norm_dist(rand_gen);
    p_g_new[2] = p_g_old[2] +  cd_norm_dist(rand_gen);

    // calculate new simulation results and error
    // based on new parameter values

        // model building position values based on the new
        // mass and spring coefficent values of the
        // dampener
    update_top_floor_pos(
        xg_data_fname,
        init_sim_cond_fname,
        p_g_new,
        sim_res,
        sim_vals_new
    );

        // calculate the "sum of squares" value comparing the
        // data and model given our estimate of the errors
        // in the data
    update_error_est(
        des_vals,
        sim_vals_new,
        est_des_err,
        3,
        sim_err_new
    );

    // calculate the probability of acceptance for both
    // the mass and spring coeficient parameters, this uses
    // bayesian stats to include the assumption that the
    // paramter space for each parameter is gaussian
    p_factor = exp(-sim_err_new+sim_err_old);
    p_prior_old = p_factor * prior_build(p_g_old, p_prior, p_sig);
    p_prior_new = p_factor * prior_build(p_g_new, p_prior, p_sig);

        // calculate current p_accept
    p_accept = p_prior_new/p_prior_old;

    if (r_accept(rand_gen) < p_accept) {
        p_g_old[0] = p_g_new[0];
        p_g_old[1] = p_g_new[1];
        p_g_old[2] = p_g_new[2];
        sim_err_old = sim_err_new;

        if (iter_count > mcmc_burn) {
            p_g_hist.push_back(p_g_new);
            sim_err_hist.push_back(sim_err_new);
        }
    }

    iter_count += 1;
}

ofstream mcmc_output;
mcmc_output.open(output_fname);

for (int i=0; i<p_g_hist.size(); i++) {
    mcmc_output << p_g_hist[i][0] << ", "
                << p_g_hist[i][1] << ", "
                << p_g_hist[i][2] << ", "
                << sim_err_hist[i] << "\n";
}
mcmc_output.close();
```