# Lattice of Springs to Model Waves

Ghasemfar, Dadmehr
danielghasemfar@gmail.com

Chen, Boyuan
boyuan.chen@duke.edu

April 2024

**Abstract**

This work explores the use of simulated data to train convolutional neural networks (CNNs) for spatial understanding using acoustic vibrations, an approach analogous to biological echolocation. The focus is on artificially generating surface sound data using a custom multi-threaded wave propagation simulation based on lattices of spring-mass-dampers. It is tested across a wide range of parameters and characterized. It is also connected to real-world wave phenomenon through modal analysis of Chladni plates. Interfacing in real-time with a physics collision engine, it attempts to determine the unique sounds resulting from a collision.

## Contents

**1 Introduction** **2**

**2 Mathematical Background** **3**

**3 Simulation Iteration** **6**

**4 Testing and Evaluation** **10**

**5 Conclusion** **17**

# 1 Introduction

In attempting to decipher the physical topology and layout of a surrounding environment, both humans and robots rely on visible light as their primary means. Navigating through the corridors of a structure devoid of the senses of smell, hearing, taste, or touch, presents a minimal challenge. On the other hand, the moment a blindfold is introduced most individuals find themselves metaphorically and physically 'stumbling around in the dark'.

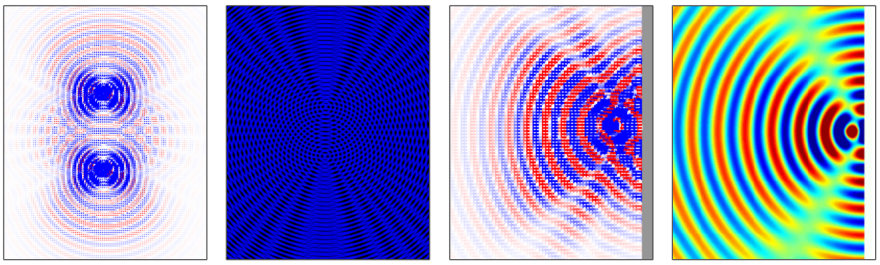


**Figure 1:** From left to right a) a snapshot of a simple 2D sound simulation (see part 1 of methods) with two synchronized drivers, b) the same setup but in a 2D light simulation [4], c) a single sound driver near a fixed edge, and d) the same setup in a 2D light simulation.

Numerous species have evolved to "see with sound". For instance, bats and dolphins use echolocation for obstacle avoidance and prey/predator detection [6]. Various insects utilize surface vibrations to perceive potential threats, and moles discern subterranean movements to infer surface activity.

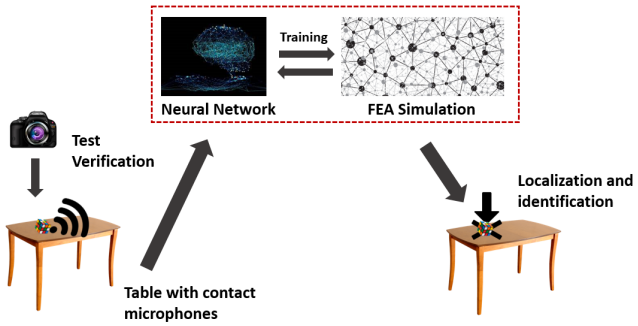



**Figure 2:** A visual schematic of the process for training on example object. Most of the training happens virtually between the simulation and the network (center). Real life testing refines the network (left), which will then be used for localization (right).

Both sound and light manifest as waves traversing through a medium, interacting with objects before being detected by sensors. Both undergo reflection, refraction, and energy dissipation in an analogous manner. Like light waves, sound carry information about the environment. The question that naturally arises: can we utilize sound data to infer spatial information in a similar manner?

Recent works have harnessed surface sound readings in tandem with trained convolutional neural networks to decipher spatial information. In *BoomBox: Visual Reconstruction from Acoustic Vibrations*[2], Dr. Chen showcased the feasibility of accurately reconstructing visual and height data from surface vibrations, using a specific plastic container for training and testing. The objective of this work is to streamline the training process by incorporating artificial/simulated vibration data, thereby facilitating initial network training on simulated datasets followed by refinement with real-world data.

This approach unveils numerous potential technological applications. Consider, for instance, a scenario where a table is equipped with four contact microphones, one at each corner. Through simulation and training, the network can discern the precise locations of finger taps and objects (pencils, cups, etc) positioned on it. Another illustration involves a robotic arm endowed with tactile sensors on its fingers. By employing an optimal array of contact microphones and interpreting the resulting surface sounds, the system can deduce all conceivable points of contact on the hand. Such capabilities furnish the manipulator with invaluable feedback, thereby enhancing object manipulation, particularly when integrated with artificial implant technology.

This advancement holds promise for a range of robotics applications conducted in environments devoid of light, as sound waves persist regardless of lighting. Nevertheless, perhaps the most significant outcome of this research lies in the expansion of sensing modality. For instance, if a robotic arm is equipped to both visually perceive a patient's limb and acoustically detect the waves propagating through it, it can execute delicate surgical procedures with heightened precision and confidence.

# 2 Mathematical Background

Let u(x,t) be a scalar field representing the disturbance to the intended wave field. For example, this may be: A) air pressure at a point in space, B) the height of the water across a calm pond, C) the absolute value of the electromagnetic field. All classical wave phenomenon is described by the wave equation shown

below. For scalar values (pressure in space, surface displacement at a location, etc) u(x,t) is a scalar field, while for electromagnetic waves, u(x,t) is a 3D vector field.

$$\ddot{u} = \frac{\partial^2 u}{\partial t^2} = c^2(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}) = c^2 \nabla^2 u \tag{1}$$

The operator on the right hand side is known as Laplacian operator. It is defined as the divergence of the gradient of u($\mathbf{x}$,t). To find a solution to the PDE we must have initial values $u(\mathbf{x}, 0) = u_{\mathrm{i}}(\mathbf{x})$, a set of boundary conditions $u(\partial\mathbf{x}, t) = u_{\partial}(x, t)$, and a defined forcing function F(x, t). The forcing function encapsulates outside disturbances to our system. It can be thought of as the additional input forces at position x and time t.

The wave equation has one particularly nice property: it is linear. If the two solutions $f_1$(x, t) and $f_2$(x, t) both satisfy the equation and all associated BC's and IV's, then any solution of the form $Af_1$(x, t) + $Bf_2$(x, t) will also be a valid solution (assuming A and B are both scalars). This is called the superposition principle.

Furthermore, any solution that satisfies the equation with the forcing function F(x, t) = 0 (i.e the system is isolated and undisturbed), can also be a component of the full solution to the equation with a non-zero forcing term. A general solution is the solution to the unforced equation. A specific solution is the solution with the specific forcing function. Actual observed solutions can be a superposition of the two.

It would make sense if there was some set of fundamental solutions that compose all other solutions. These would be solutions that cannot be written as a linear sum of other solutions in the set. Such a solution is called an eigenmode or eigenfunction solution or harmonic mode. Such solutions have the form A(t)f($\mathbf{x}$) - a product of a space-dependent term and a time-dependent term. Intuitively, if the A(t) term is sinusoidal, the solution can be thought of as a standing wave.

the $\omega^{th}$ eigenmode is $u_\omega(\mathbf{x}, t) = e^{-i\omega t} f(\mathbf{x})$ where $e^{-i\omega t} = \cos(\omega t) - i \sin(\omega t)$

Plugging this into the wave equation and simplifying, we get the following ODE.

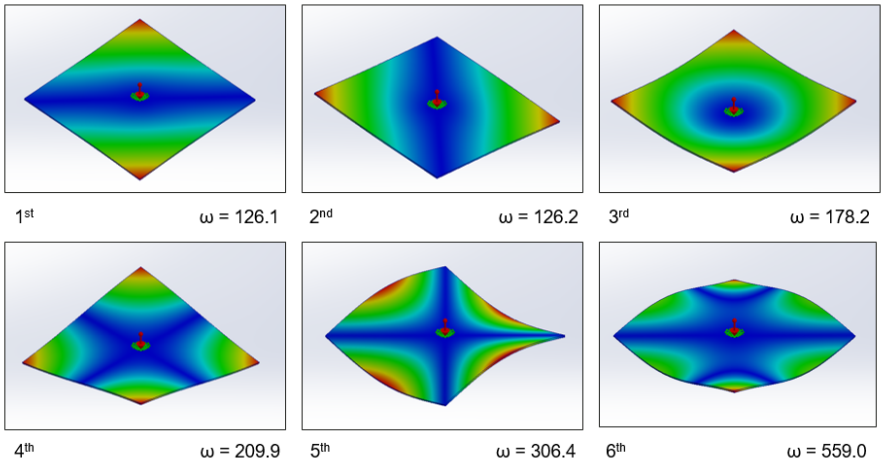



**Figure 3:** The first six harmonic modes of a flat steel plate, computed in SolidWorks. Angular frequency is in units of rad/sec.

$$\frac{\partial^2}{\partial t^2}u_\omega = \frac{\partial^2}{\partial t^2}(e^{-i\omega t}f(\mathbf{x})) = -\omega^2 e^{-i\omega t}f(\mathbf{x}) = c^2(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2})(e^{-i\omega t}f(\mathbf{x}))$$

in x direction we get $\frac{d^2}{dx^2}(e^{-i\omega t}f(x)) = -(\frac{\omega}{c})^2 f(x)$ and similarly for y and z

The three equations (one for x, y, and z) are now a system of ODE's. These are far easier to solve with numerical analysis techniques than the original PDE. Critically, this ODE is in a well known form. It is the Helmholtz equation - or the eigenvalue problem for the Laplace operator. It has well known solutions of the form $f(\mathbf{x}) = Ae^{-i\omega\mathbf{x}} + Be^{i\omega\mathbf{x}}$. With the harmonic modes of the equation solved, we know that a true solution will be a linear sum of harmonics modes. The continuous version of this statement can be expressed as:

$$u(\mathbf{x},t) = \int_{-\infty}^{\infty} s(\omega)u_\omega(\mathbf{x},t)d\omega = \int_{-\infty}^{\infty} s(\omega)(Ae^{-i\omega\mathbf{x}} + Be^{i\omega\mathbf{x}})d\omega \qquad (2)$$

The important realization here is that all solutions are a (possibly infinite) sum of harmonic standing waves. When you hit an object at two different spots, or with two different forces, you hear different sounds. You are changing

the linear combination of harmonic modes s($\omega$), but not the harmonic modes themselves. Almost all modern CAD software can easily calculate the first hundred or so harmonic modes of a 3D object. A useful simulation for this task should analytically approximate s($\omega$), either directly or in the background.

# 3 Simulation Iteration

The simulation is based around modelling solutions to this equation as a set of connected spring-mass-damper systems. In other words, a collection of points connected to an arbitrary number of neighbors by linear springs (Hooke's law) and dampeners, each with a point mass. The goal was to capture the way individual molecules in a solid are held in equilibrium in a lattice. In this way, a vibration in one particle is propagated across the medium. Each individual system is only $2^{nd}$ order, but the cascaded network can model much more complex phenomenon.

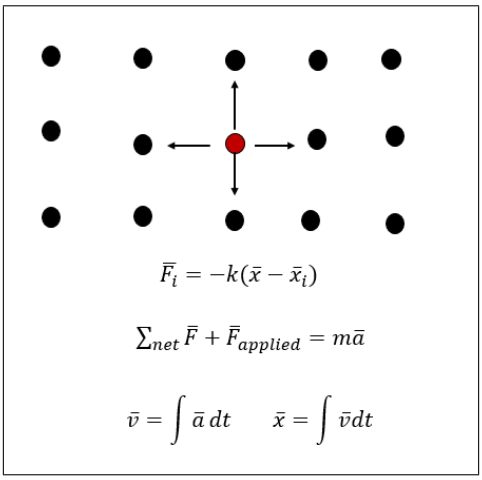
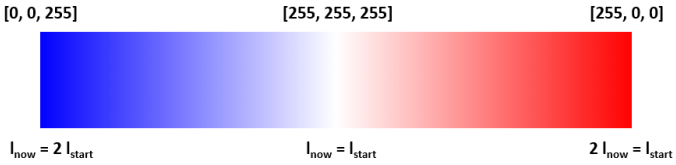



**Figure 4:** First mathematical model. Neighbor forces are summed to find acceleration. This is integrated twice for position.

To test the feasibility of this model, a very simple 2D simulation in Java is created. All particles (mass m) are in a uniform Cartesian grid with. Each particle is connected to its four neighbors via springs of stiffness k. Only the bonding springs are shown on screen. A spring in compression is more red, while one in compression is more blue. A perfectly neutral spring is white. This convention is followed in later simulations. Finally, dampening is given by the equation $m\frac{d}{dt}\mathbf{v(x)} = d\mathbf{v(x)}$ where d is the dampening constant.




Although the limited complexity of the algorithm produced visually pleasing results, the simulation is very limited. For one, it is completely two dimensional. The medium of choice can only be rectangular with side lengths $[x_l, y_l]$. It is limited to rectilinear lattices, and all edge cases fall into one of four sets. Each edge set can either be fully fixed, fixed in one axis, driven by a vector forcing function $F_i(\mathbf{x}, t)$, or free. Finally, it is running on the CPU and is hence much slower than a GPU capable simulation. The idea worked well enough to pursue further however.

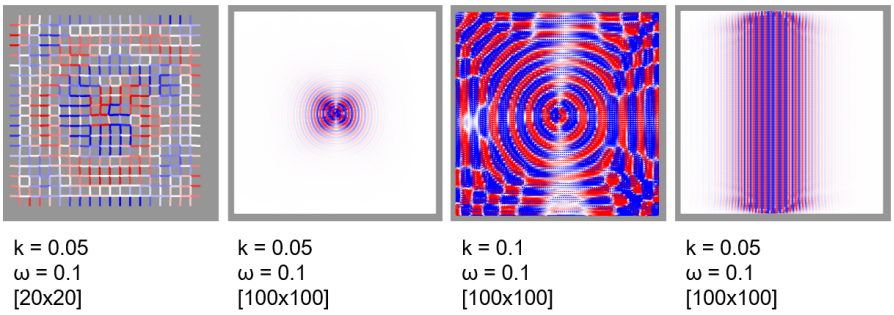



**Figure 5:** Four test cases with parameters labelled. The first has fewer particles to show the lattice. The second and third are driven at the center in a circular fashion. The fourth is the center column moving back and forth. All have mass m = 0.05 and dt = 0.1.

For the next simulation we switch to Python, specifically the **Taichi package** (to incorporate GPU multi-threading). This made computation an order of magnitude faster. We also switch to using a 3D set of points. However, the simulation was limited to rectilinear lattices with 6 local neighbors and the medium limited to being a cuboid of size $[x_l, y_l, z_l]$. This means that all edge cases fall into one of six sets. As before any set can be fixed in any axis, driven in any axis, or free.

The simulation needed to incorporate one more critical feature to be useful for artificial data generation - interfacing with a 3D Physics and Collision Simulation. This would allow us to model two 3D objects colliding, and to then separately model the sound waves in each object based on the contact points

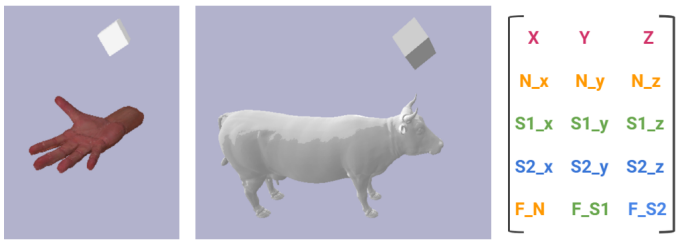



**Figure 6:** Snapshots of the collision between a cube and two test objects. At each time $t_i$, there are $n(t_i) < n_{max}$ contact points in the form shown to the right. So for an entire test with $t_{max}$ time frames, the data has size $[t_{max}, n_{max}, 4, 3]$. Typical numbers are $t_{max} = 1500$ and $n_{max} = 15$.

and forces. The **PyBullet library** was perfect for this task. It gave us the list of locations of contacts with respect to time, a list of normal force components, a list of orthogonal components. All force components also come with a corresponding 3D unit vector pointing in the necessary direction.

It did not make sense to run the wave-propagation simulation simultaneously with the physics-collision simulation. One is on GPU and the other on CPU. One would just slow down the other. Instead, we run the physics simulation, save the collision locations and forces in a .NPY file, and use this data in our wave propagation simulation.

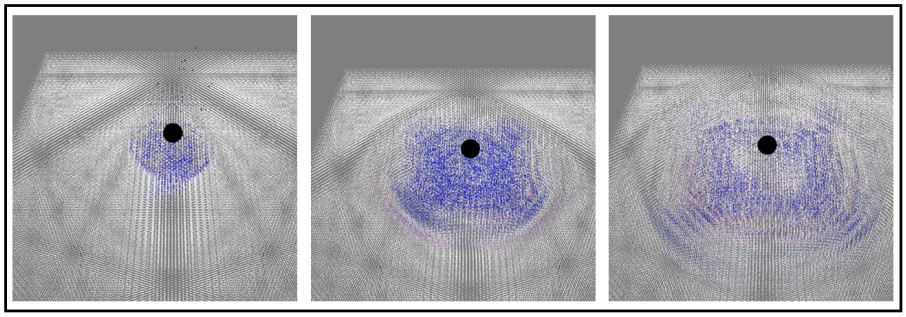


**Figure 7:** Three consecutive snapshots of a simulation as a marble impacts a table and creates a propagated wave. The object itself then bounces back up and hits the table again. Here $m = 10$, $k = -3$, $d = 0.98$, $dt = 1$, and $t_{max} = 1500$.

With a decent amount of tuning and debugging, this second simulation was able to simulate basic 3D harmonics. However, it was flawed. One issue was stability - with high spring stiffness k and low mass m and dampening d, each cell system enters an uncontrolled and unbounded series of oscillations. Once

neighboring particles begin to shoot out of their own $[dx, dy, dz]$ cells, things risk falling apart.

A more significant issue was discovered when testing non-rectilinear lattices. Although the distance between two points is constrained by a spring, the angle between neighboring bonds is not. In other words, a pair of points could rotate around one another even when fully constrained. To remedy this an angular spring, moment of inertia, and angular dampener are added to the system. This way the correcting torque is proportional to the angular displacement from the equilibrium angle.

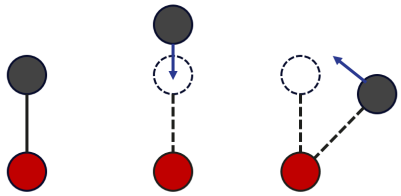


**Figure 8:** The new mathematical model has a linear component and an angular component. At equilibrium, all forces are zero. With some linear displacement the force is $F_{axial} = k_l(x_f - x_i) - d_1 v(t_f)$ while with some angular displacement the force is $F_{tangential} = k_2(\theta_f - \theta_i) - d_2\omega(t_f)$. This is done for each neighbor.

With these insights it was time to create a generalized 3D simulation. This should be capable of testing for any 3D object (not just cuboids), with any 3D lattice (cubic, hexagonal, random, etc), and with any arbitrarily defined number of boundary points. It should interface directly with a collision detecting physics engine (PyBullet as before) and run parallel processed as kernels on the GPU.

One road block is converting a 3D object into a point cloud and reading this point cloud at initialization. To do this we must know if a point is contained within a 3D object. This can be resolved by projecting a line, starting at the point, going in any direction and counting the number of times it crosses the object. If it is odd the point is inside, if it is even the point is outside. PyBullet uses URDF's but Trimesh (the library used for 3D object manipulation) requires an OBJ file. The scaling, translation, and rotation of these three must line up for the collision data to line up with the correct point cloud data. This does not always happen, since URDF uses varying origins and since the two libraries use different axis and rotations. Extra care must be taken to ensure both simulations line up in angle and translation and scaling.

The results of this final simulation are shown below. This version can successfully model any shape and is stable for a wide range of stiffness values (k = 40 - 40,000). Some of the tests on a flat plate are shown below. Before this

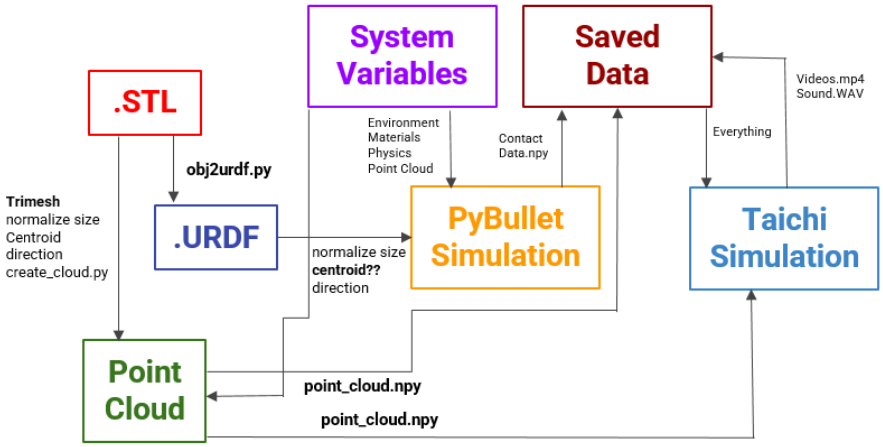



**Figure 9:** A system diagram of the different files involved in running a simulation. Material properties and test settings are set in a YAML in system variables.

model can be tested and evaluated, we must convert the time-series particle position data (from simulation) into a sound signal over time.

Sound files we listen to are just a scalar value over time - most likely 64 bit integers at a rate of 44.1 kHz. We define the pressure at $\mathbf{x}$ as the average relative change in bond length amongst all the bonds connected to $\mathbf{x}$. In other words $P(\mathbf{x}, t) = \frac{1}{n}\sum_{i=1}^{n} \frac{x_i(0)}{x_i(t_f)}$. The intensity of the sound heard is then just some constant multiplied by this pressure squared. This way, the sound can be "recorded" at any point within the solid or on the surface.

# 4 Testing and Evaluation

A simulation that does not reflect reality is not scientifically useful. To verify the validity of the spring-mass-dampener model, the simulation must be both A) tested and characterized across a wide set of parameters and B) compared to real-life vibration.

A major complication is that the simulation settings determine the material properties. Although the model may be accurate, its accuracy depends

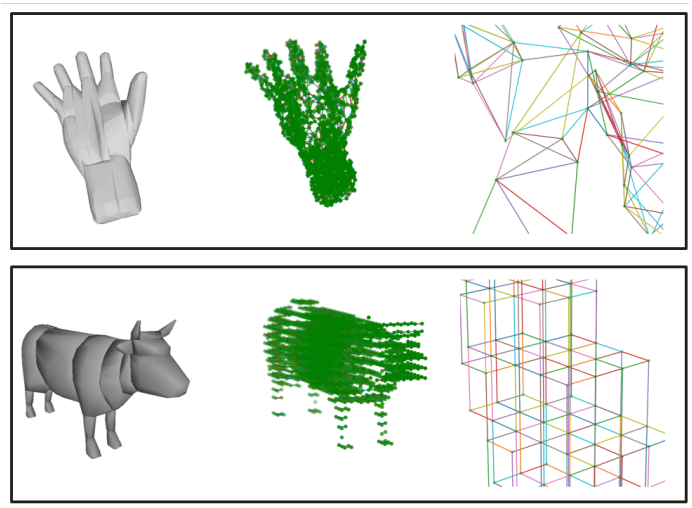


**Figure 10:** Row one shows the original STL file of a hand and the corresponding random point cloud. The points are selected randomly and connected to the 5 closest neighbors. Row two shows the same but for an STL of a cow. The points are in a rectilinear lattice this time.

significantly on parameters such as $(k_{spring}, m, d, dt, \text{density})$. Real solids can be characterized by properties such as the Young's modulus E, Poisson ratio $\nu$, Bulk modulus $\kappa$, shear modulus G, speed of sound c, etc. In theory there is a mathematical relationship between the simulation parameters and real life material properties. In other words, there exists some $f(k_{spring}, m, d, dt, \text{density})$ that returns $(E, \nu, \kappa, G, c)$. To understand the behaviour of the simulation across a range of parameters, a handful of simulated tests are created. The first one of these is a speed of sound measurement.

A 100x5x5 cm rod with a rectilinear lattice of approximately 50x5x5 particles is created. An impulse is sent through one end at t=0. The response time is measured at the other end. This is repeated across different stiffness's and masses. Units are intentionally left out since the simulation is not dimensioned - one unit mass could represent 1 kg or 1 gram for example.

These results match intuition; a higher stiffness results in a faster wave and a higher mass results in a slower wave. From curve fitting it appears that the relationship here is $c = c_1\sqrt{\frac{k}{m}}$ for constant $c_1$. Fascinatingly, if you refer to a physics textbook[1] you will find that the speed of sound in a solid is given by $c = \sqrt{\frac{E}{\rho}}$ for Young's modulus E and density $\rho$.

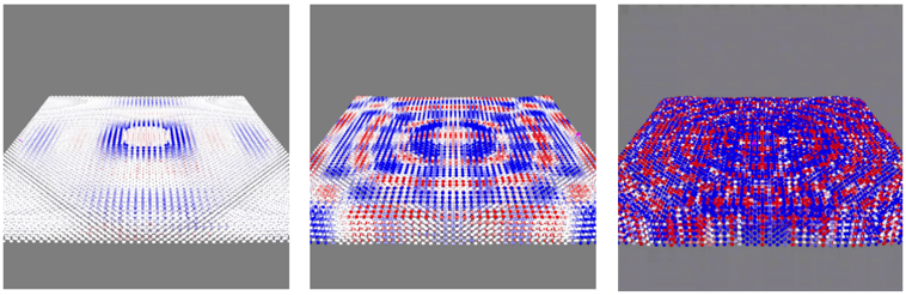


**Figure 11:** Screen captures of simulation at steady state. All three are set to k=-200, m=40, d=0.98, have 50x50x6 particles, are driven at the center, and have free boundaries. The first is driven at 8Hz, then 16Hz, then 32Hz.

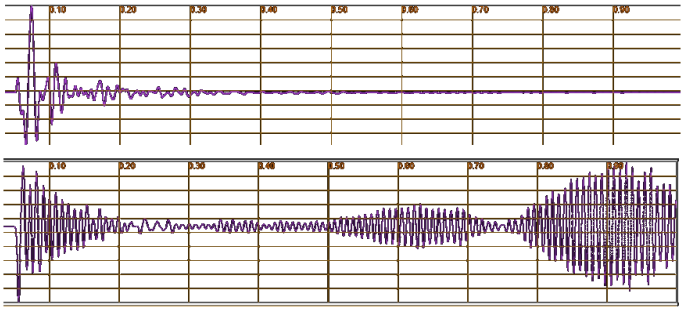


**Figure 12:** Two .WAV sound files created from testing on a 100x5x5 cm rod, with the sensor placed at one end and the driver at the other. In the first we see an impulse response, in the second we see a frequency response. The sound data is sampled up to 44 kHz using linear interpolation.

We also examined the effects of point-cloud density and point-neighbor density on the speed of sound. To test another aspect of the model, an amorphous lattice was used (N points randomly placed and connected to the closest M neighbors). The density of points is quantified by the average length of bonds. Shorter average bonds means more points in the set volume. The number of neighboring bonds is quantified by a bond-cutoff. When the bond-cutoff is 0.5 any bond longer than $0.5l_{avg}$ is removed (where $l_{avg}$ is the average bond length). When the bond-cutoff is 1.0, any bond longer than the average is removed.

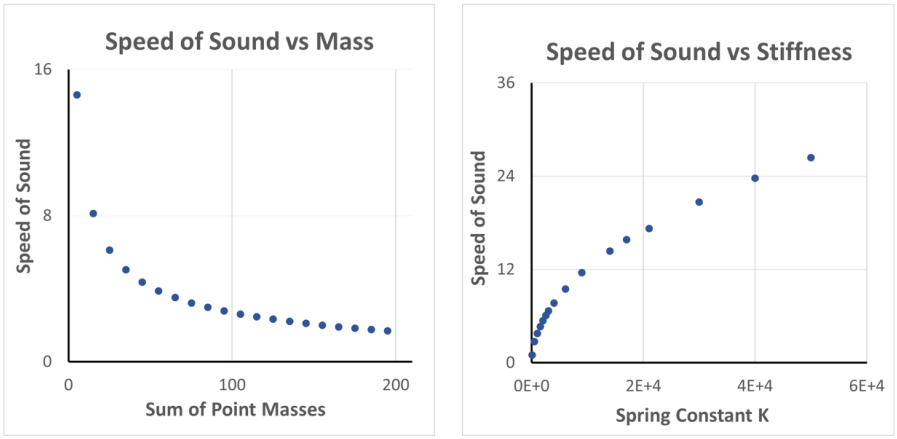



**Figure 13:** Plots of speed of sound against the sum of point masses (over 270 points) and against stiffness. The dt = 0.001 and d = 0.98 for both. For the mass test, the stiffness is set to k=-1500 while for the stiffness test the sum mass is set to M=40. Fitting an arbitrary exponential of the form $Ax^B$, both $R^2$ values are 0.99. The B coefficients of both are very close to $\pm\frac{1}{2}$.

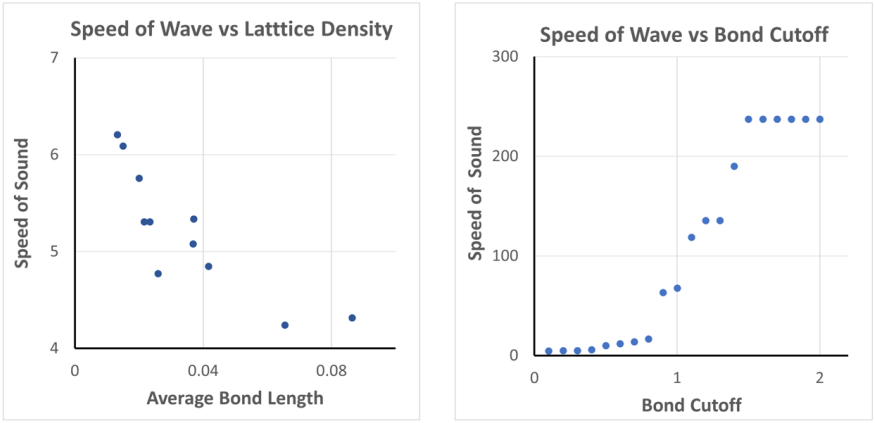



**Figure 14:** Plots of the speed of sound against lattice density and bond density. The lattice density relationship appears linear while the bond density relationship appears to be a sigmoid curve. This makes sense because after a certain bond-cutoff, no more bonds are removed.

For the next test we were hoping to measure the transfer function, or **frequency response**, of the system. We again used the same 100x5x5 cm rod

with a cubic lattice of approximately 50x5x5 particles. To do this, we send in a signal of the form $L_{disp} \cdot \sin(\omega t)$ with some maximum displacement $L_{disp} = 0.03$. We then measure the output sound recorded and fit this output to some $A \cdot \sin(\omega t + \phi)$. From this, the magnitude response is given by $\frac{A}{L_{disp}}$ while the phase response is simply $\phi$. This test was repeated for several different stiffness levels.

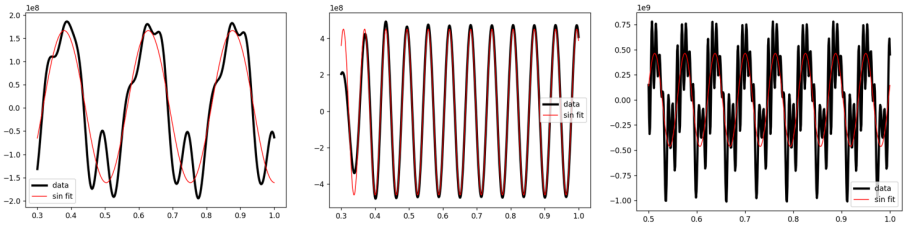



**Figure 15:** Some example frequency response outputs. Black is the simulation data and red is the sinusoidal fit. All three runs are using M = 40, d = 0.98, and dt = 0.001. The first two are at k=-3000 while the last is at k=-80000. From left to right, the driving frequencies are 4 Hz, 16 Hz, and 16 Hz.

As expected, each individual curve has a single peak. They start at one and go out to zero for higher frequencies. The peak can be thought of as the frequency of the natural mode of the rod. As the material becomes stiffer, this frequency increases. It is important to note that the output of this transfer function is the sound measured at the furthest end, not the displacement. That is why we are seeing such a large gains.

Although tests one and two successfully characterized the simulation, they failed to connected the model to real life data. To correct this, we could measure the true speed of sound and true frequency responses across a 100x5x5 cm steel rod and calibrate the k, M, d, and dt such that the simulated response and real-life response match. Due to time constraints, this is left for future research.

The final test of the simulation is the most fascinating. Physics undergraduate courses will often demo the **Chladni plate experiment** in lab. In this experiment a thin metal sheet is connected, by its center, to an oscillator driver (essentially a speaker without the diagram). Sand is sprinkled uniformly across the metal sheet. Certain driving frequencies will result in 3D vibrational modes on the approximately 2D surface. As discussed in *Mathematical Background*, such harmonic modes are standing waves on the surface. This means some parts of the surface are oscillating from peak to trough (**nodes**) while others are not moving at all (**anti-nodes**). Sand will quickly accumulate wherever the surface is not moving. This way the harmonic modes can be visualized. Examples of such an experiment are shown below.

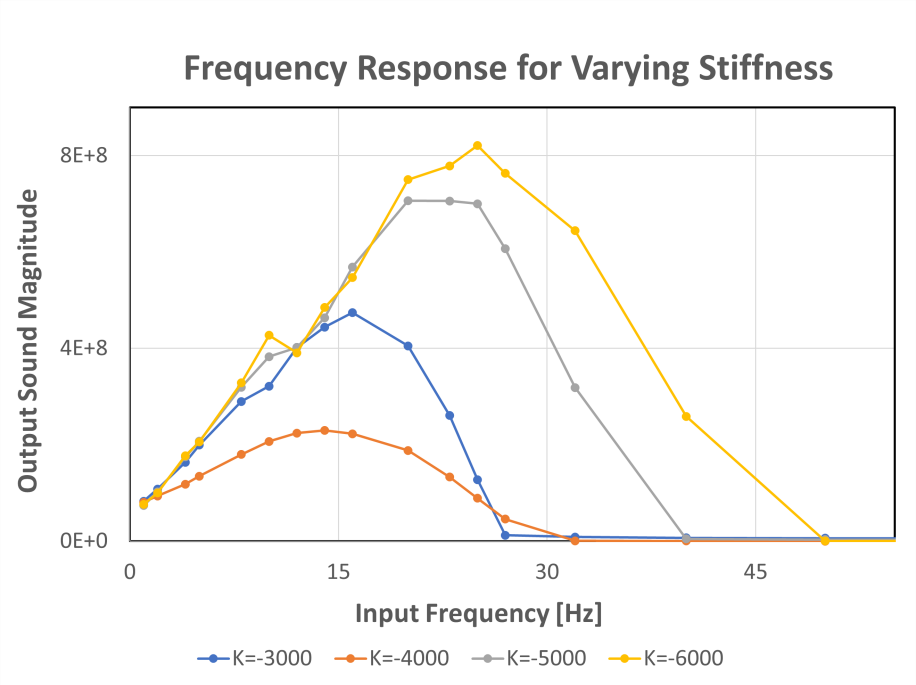



**Figure 16:** Magnitude plot of the transfer function output sound vs input displacement across several stiffness'. Sound is stored as a 64-bit integer, hence why the gain is so high. Unlike normal bode magnitude plots, a linear scale was selected as the simulation is only stable for a small range of k values.

The benefit of such a Chladni setup is that it is simple. Because it is a simple shape (approximately flat square plate) and the boundary condition is simple (free boundary around the edges, fixed at the center), the exact analytical solution can be derived from the wave equation. Let $h(x, y, t) = \sin(\omega t)s(x, y)$ represent the height of the standing wave, separated into time-dependent and space-dependent components. Taking the Fourrier series of s(x,y), we find that $s(x, y) = \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} a_{nm} \sin(\pi n x) \sin(\pi m y)$ for some constants $a_{nm}$. Plugging this back into the wave equation, we find the solution for a generic Chladni pattern - at some frequency f.

$$s(x, y) = a \sin(\pi n x) \sin(\pi m y) + b \sin(\pi m x) \sin(\pi n y) \tag{3}$$

$$\text{with the constraint that } n^2 + m^2 = 4(\frac{f}{c})^2$$

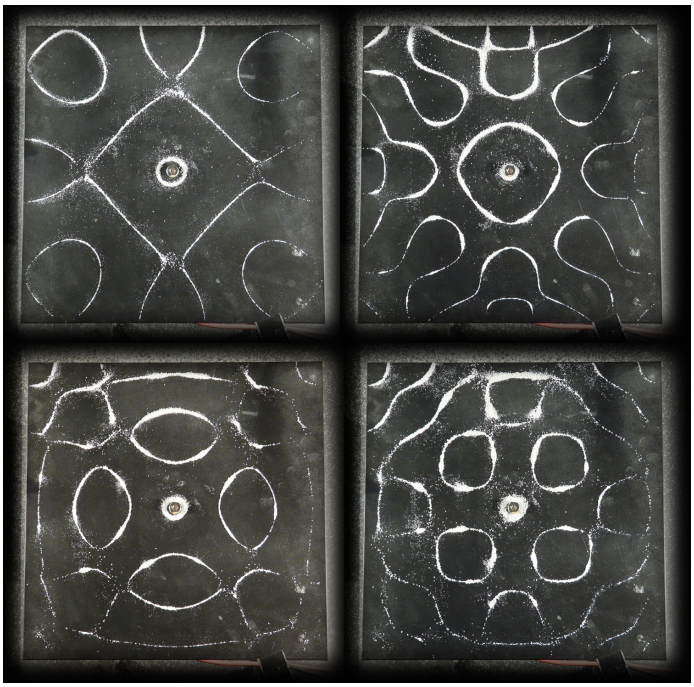


**Figure 17:** Photographs of real life Chladni patterns, courtesy of Rian Hunter[3].

By selecting the constants a, b, n, and m such that the constraint above is satisfied, we find an infinite set of Chladni pattern solutions. Separately, we can also determine the stationary points of the standing wave in simulation. The goal is to find the combination of constants that will result in the best possible fit. If we can find multiple matching modes, we can calibrate the simulation to real-life. For example, say a simulated plate at (k, M, d, dt) with frequency $f_S$ behaves like a real Chladni plate at frequency $f_R$ and is stable. Now assume the same plate is still stable at $2f_S$ and behaves like a real Chladni plate at frequency $2f_R$. With these two matches, we expect the simulated behaviour in the range $[f_S, 2f_S]$ to match real life.

To find the simulated stationary points, we use the sine fitting code from earlier to accurately find the amplitude of oscillations at that point. We then remove all points which have amplitude greater than a threshold. We can then run a least-squares brute force program to check all Chladni constants against this data. This way the Chladni pattern of best fit is determined.

This test gave mixed results. For one, all simulations either matched their corresponding Chladni pattern very well or did not at all. The best few runs are

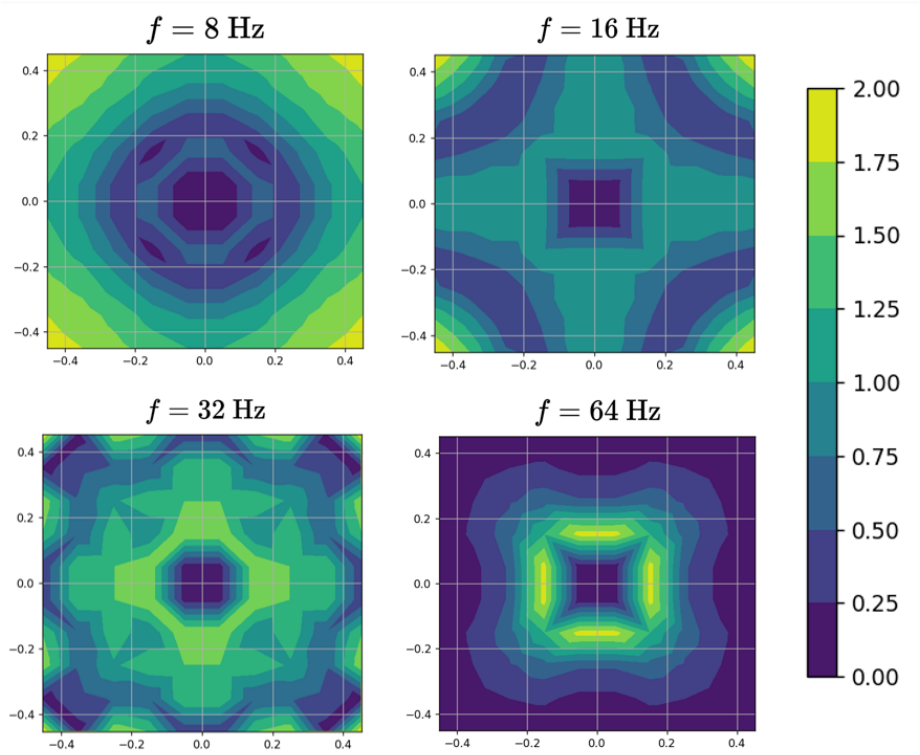



**Figure 18:** Scalar field heat maps showing the amplitude of oscillations at all points for different driving frequencies. All tests are done with k=-8000, M=40, d=0.98, dt = 0.001, and $L_{disp} = 0.03$

shown above, but these are just six of twenty. The other fourteen had either no discernible match to the Chladni fit or where repeats of the modes seen above.

# 5 Conclusion

This work has barely scratched the surface of this complex and powerful field of study. The idea of using a lattice of spring-mass-dampers started out as a trivial little 2D experiment, but every development brought more questions. In an attempt to avoid the complexity of solving partial differential equations we resorted to a simpler concept. However, the difficulty then became in showing that such a simple model closely represents real life. At the end of the day, there is not enough evidence to prove that such a model works.

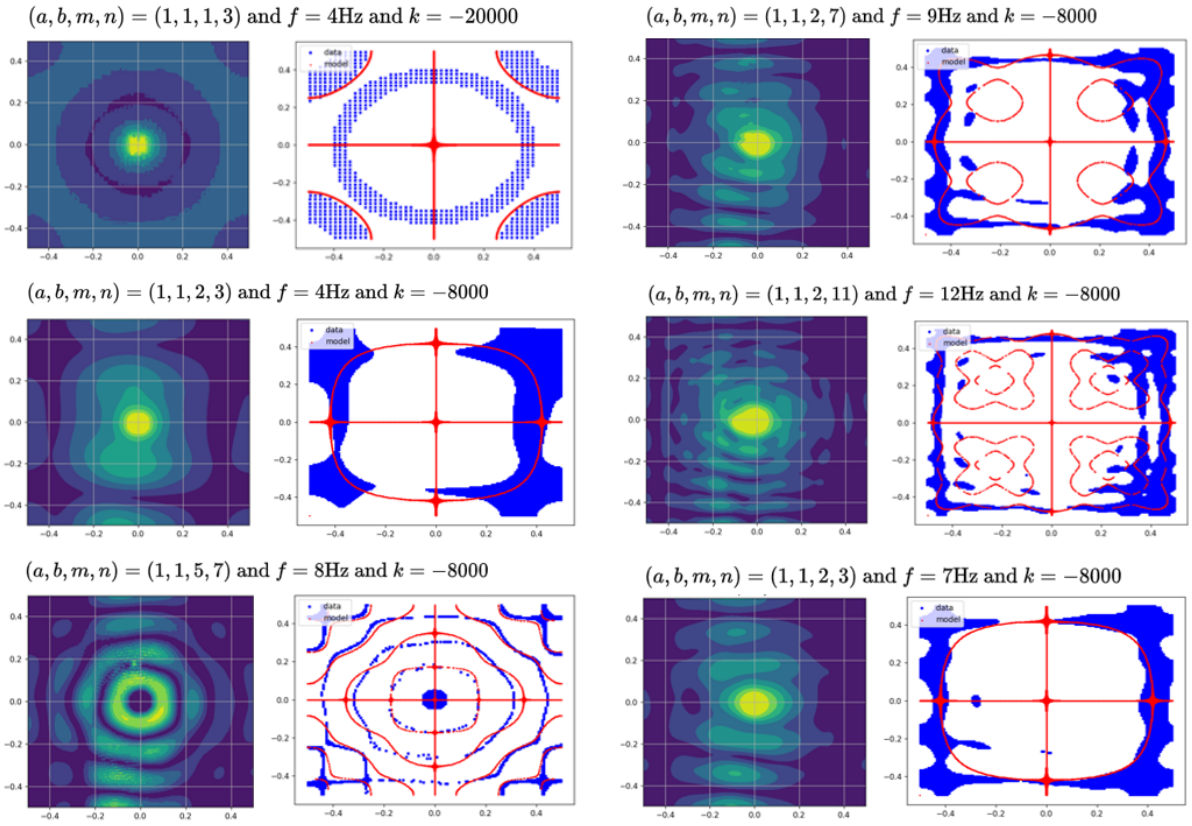



**Figure 19:** Six different runs along with the best-fit Chladni solutions. All tests were at d = 0.98, M = 40 and dt = 0.001. The corresponding Chladni constants, driving frequency, and stiffness' are labeled above each run. To save on computing time the Chladni constants are assumed to be integer, although they do not have to be in real life.

The videos, photos, and sounds produced from this method are intriguing and promising (many can be found at www.danielghasemfar.com). The parameter testing and frequency testing showed overall trends similar to those seen in real solids. However, to really utilize this method we must be able to A) select material properties and B) have more confidence in the physical accuracy of the simulation. In other words, we need some function $f(k_{spring}, m, d, dt, \text{density})$ that returns the material properties $(E, \nu, \kappa, G, c)$.

Future work can expand on this in many ways. Just a few potential ideas include:

- Calculate the first few hundred modes of a 3D object. Train a neural network to predict the correct linear combination of these standing waves that results in an expected sound post-collision. Such a method requires much less training and a much simpler network. The heavy lifting would be in quickly adding hundreds of vector fields. Difficulties include knowing the exact surface vibration at all points of the boundary while training.
- Train a neural network to approximate the behavior of the function that connects simulation parameters to real solid properties. Use data collected from axial-loading stress tests, speed of sound measurements, torsional stress test, and bulk modulus test to determine $(k_{spring}, m, d, dt)$.
- Place artificial sensors on an artificial moving robotic limb, and use the simulated vibrations to train a network to determine joint angles. Such a robotic limb would know its position in space without encoders or a camera.
- As before, use the simulation to train on a flat table (four contact microphone on each corner) with the goal of determining the location of a finger tap. This table is now a touchscreen!
- Circular Chladni plates have a very unique and special pattern relating the numbers of diametric (linear) nodes and of radial (circular) nodes. See *Neville, 2004* [5, pg.72-75]. This could be used to calibrate the simulation parameters to solid properties.

# References


[1] Dictionary of physics, speed of sound. *Oxford Reference*, 2024.

[2] Boyuan Chen, Mia Chiquier, Hod Lipson, and Carl Vondrick. The boombox: Visual reconstruction from acoustic vibrations, 2021.

[3] Rian Hunter. The light - creating digital chladni patterns. 2014.

[4] Amy Rouinfar: Noah Podolefsky. Phet interactive simulations - wave interference.

[5] Neville H. Rossing, Thomas D.; Fletcher. *Principles of Vibration and Sound*. 2004.

[6] Emily Willingham. Bats beat dolphins in the battle ovver who has the best sonar.