Computer Methods in Applied Mechanics and Engineering 420 (2024) 116661

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

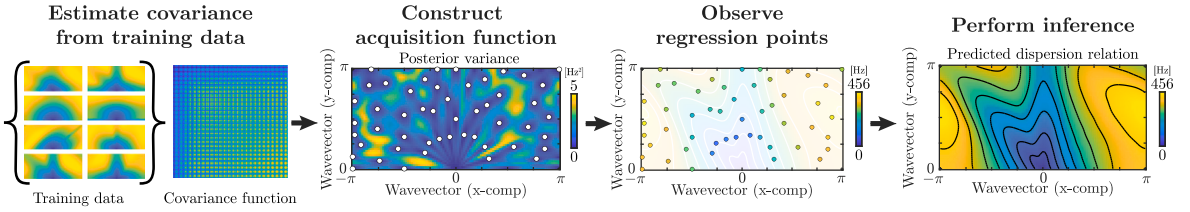
journal homepage: www.elsevier.com/locate/cma

Gaussian process regression as a surrogate model for the computation of dispersion relations

Alexander C. Ogren [a,*], Berthy T. Feng [b], Katherine L. Bouman [b], Chiara Daraio [a]

[a] Department of Mechanical and Civil Engineering, California Institute of Technology, 1200 E. California Blvd., Pasadena, CA, 91125, USA
[b] Department of Computing and Mathematical Sciences, California Institute of Technology, 1200 E. California Blvd., Pasadena, CA, 91125, USA

GRAPHICAL ABSTRACT



A graphical depiction of our surrogate model pipeline. The model first observes a dataset to construct the covariance function of the Gaussian process model. Using this covariance function, the model build sits acquisition function, determining the locations of the regression point sit will sample. Finally, the model observes the value of the dispersion relation at the selected regression points, and infers the dispersion solution everywhere in the irreducible Brillouin zone. The surrogate model makes this inference two orders of magnitude faster than the exact calculation, and one order of magnitude faster than an equivalent-error linear interpolation model.

ARTICLE INFO

Keywords:
Dispersion relation
Wave propagation
Surrogate model
Gaussian process regression
Machine learning

ABSTRACT

The ability to design materials for wave propagation behaviors has high potential for impact in medical imaging, telecommunications, and signal processing. The dispersion relation is the key mathematical object that describes linear elastic wave propagation behavior in a material. To date, the design of wave propagation behavior in materials has been limited to one or two design objectives, for example, designing for one or two bandgaps or wavespeeds. This is a result of the complicated relationship between a material and its dispersion relation, and the hefty computational cost of traditional dispersion computations. Decreasing the amount of time required to perform dispersion computations will streamline the design process of wave devices, allowing for more thorough design optimization workflows, more highly resolved design features, and multi-functional materials.

In this work, we demonstrate a specialized Gaussian Process Regressor as a surrogate model for the computation of dispersion relations to alleviate the immense computational cost of the traditional model. By measuring covariance information from relatively small training sets (10s to 1000s of dispersion relations), our surrogate model efficiently infers the full dispersion

* Corresponding author.
E-mail address: aogren@caltech.edu (A.C. Ogren).

https://doi.org/10.1016/j.cma.2023.116661
Received 19 August 2023; Received in revised form 16 November 2023; Accepted 21 November 2023
Available online 5 January 2024

relation after solving for the dispersion relation at only a sparse set of wavevector points. Further, we provide a mathematical framework to use this model for material design through gradient-based optimization methods.

1. Introduction

The structure and constitution of a material dictate the properties of the waves it can support, which are thoroughly described by the material's dispersion relation. A material's dispersion relation is the relationship between the frequencies and wavevectors of all linear elastic waves that can travel in the material. Dispersion relations contain information about bandgaps (frequencies at which waves experience spatial attenuation), group velocity (the velocity with which energy can be transported by waves in the material), and phase velocity (the velocity at which a wave peak will move in the material). Bandgap information can be leveraged to engineer filtering and reflection, and group velocity and phase velocity information can be leveraged to achieve refraction. These fundamental behaviors can in turn be utilized to design complex wave propagation devices such as wave guides [1–3], wave lenses [4–6], wave multiplexers [1,2,7], and more.

Current methods to calculate dispersion relations are based on numerically computing the eigenvalues of the elastic wave equation [8]. Computational limitations are especially restrictive when these expensive forward models are used to solve the inverse problem of design [9–13]. As a result, scientists and engineers are often forced to simplify design workflows by limiting the design space to be relatively narrow, considering only partial dispersion information and/or relatively basic design objectives [1,2,4,9,12–14].

Our method aims to increase the accessibility of full dispersion relations to alleviate the need for simplifications such as partial dispersion computations. Full dispersion relations are defined over a region of reciprocal space (wavevector space) called the *irreducible Brillouin zone* (IBZ). Due to computational costs, dispersion relations are often computed only along one-dimensional slices of these regions called *irreducible Brillouin zone contours* (IBZ contours). However it has been recently shown that using this simplification can lead to errors [15]. Further, plane symmetry groups without mirror symmetries do not have a uniquely defined IBZ contour. For unit cells belonging to these plane symmetry groups, dispersion analysis along a contour will yield different results depending arbitrarily on the choice of contour. As a second example, a common shortcut when designing for group velocity is to compute the dispersion relation at only a handful of wavevectors or frequencies of interest. This is useful for achieving desired wave behavior in specific directions of propagation [12], or at specific frequencies of interest [14]. Although these methods are sufficient for designing simple wave devices, the design of next-generation, multi-functional wave devices will require the efficient computation of full dispersion relations.

Techniques have been developed specifically to efficiently solve dispersion problems such as Bloch mode synthesis [16,17], spectral X-FEM [18], specialized mixed-variational formulations [19], reduced Bloch mode expansion [20], reduced Bloch operator finite element methods [21], and wave finite element methods (WFEM) [22,23]. However, even these specially tailored methods executed by a highly parallelized and optimized implementation on a multi-CPU multi-GPU cluster admit the need for more efficient ways to explore the design space [13]. To perform design optimization on a 48^3 voxel domain, a distributed computing setup using 16 NVIDIA GTX-780 GPUs takes 6.5 min per optimization iteration, even when computing the dispersion relation only along the IBZ contour rather than over the full IBZ [13]. Fundamentally, each of [16–23] reduces the cost of dispersion computations by reducing the number of effective degrees of freedom involved in the model, thereby reducing the size of the eigenvalue problem that needs to be solved at each wavevector point. In contrast, our surrogate model reduces the total number of eigenvalue problems that need to be solved. Due to this construction, our surrogate model can seamlessly integrate with any of these aforementioned techniques to produce an efficiency improvement that compounds with that of the physics-based technique being paired with our model.

Surrogate models, often involving machine learning, have been proposed in a variety of settings to reduce the computational cost of purely physics-based methods. The applications range from multi-scale modeling [24–27], to dynamics [28], statics [29], turbulence [30], optics [31], and many others. For dispersion relation computations specifically, a fixed-resolution convolutional neural network (CNN) surrogate model has been proposed [32]. This CNN model is shown to vastly outperform the traditional densely connected neural network, suggesting that the convolutional architecture is more appropriate for these types of problems. Because CNNs can make predictions so quickly once the model is trained, it seems like a very attractive option. However, neural networks often require large amounts of training data, which are expensive to compute (hundreds of thousands of precomputed dispersion relations [32]) and require a lengthy training period (13 min on a GPU reported just for the 1D problem [32]). Unless the trained model is subsequently used to make predictions for more dispersion relations than were included in the training set, the time used to generate the training data outweighs the time saved by using the model. Further, the CNNs architecture locks the user into a fixed unit cell resolution and fixed dispersion relation resolution. If a user desires to change the resolution of the discretization of the unit cell or IBZ, a new model architecture needs to be trained on a new training set with the corresponding resolution. Recently, there has been a push towards surrogate models that exhibit resolution independence, meaning that the model can train on data of one spatio-temporal resolution, and make predictions at another resolution [33–35]. In the spirit of this effort, the surrogate model we propose exhibits resolution independence both in terms of unit cell discretization and IBZ discretization.

We propose using Gaussian process regression [36] to obtain principled surrogate models of full dispersion relations. In particular, we propose a data-driven covariance function that only requires a small training set of dispersion relations (tens to thousands) and allows for an efficient acquisition function. In comparison to deep learning-based surrogate models, our model achieves higher accuracy with less training data and is invariant to unit cell representation.

The key features of our model are:

1. The model is independent of unit cell representation. For example, if the model learned covariance information from a dataset of parametrically defined unit cells, it can still be used to make predictions for pixelated designs, and vice versa.
2. The model requires a relatively low amount (tens to thousands) of training data samples to make accurate predictions. The model learns from data by measuring covariance information, which takes less than 20 s even for reasonably large training set sizes.
3. The model uses a physics-based method to sample the dispersion relation at various points in the IBZ. This physics-based method is a modular component of the model that can be swapped for other, faster physics-based methods for dispersion computations. Because of this, the efficiency gains provided by our model directly augment the efficiency gains of the physics-based model.
4. The model not only supplies predictions, but also offers uncertainty estimates.
5. The model is differentiable, allowing for its use in gradient-based topology optimization.

In this paper, we first discuss the traditional methods currently used for the computation of dispersion relations and gradient-based optimization of dispersion relations. We then go on to provide a brief overview of Gaussian process regression (GPR). Next, we describe the details of our model and demonstrate how it would be applied to one-dimensional problems. Finally, we apply our model to two-dimensional unit cells and evaluate its performance against traditional methods as well as another existing surrogate model.

2. Traditional computation of dispersion relations

To compute the dispersion relation of a periodic structured material, the physical domain of analysis is reduced to a single unit cell. The frequency domain dynamics of linear-elastic periodic solids are governed by the harmonic elastic wave equation with Bloch-Floquet periodic boundary conditions. The harmonic elastic wave equation looks for wave mode solutions that obey linear elasticity, and assumes that these solutions will be temporally sinusoidal. The Bloch-Floquet boundary conditions enforce that the solution is periodic (since the domain is periodic), but with a complex phase offset. The complex phase offset is used to make sure we are finding modal wave solutions with a variety of different wavelengths (not just wavelengths that are integer divisors of the unit cell's dimensions).

The harmonic elastic wave equation in D dimensions is

$$-\rho(x)\omega^2 u_i = \frac{\partial}{\partial x_j}\left[C_{ijkl}(x)\frac{1}{2}\left(\frac{\partial u_k}{\partial x_l}+\frac{\partial u_l}{\partial x_k}\right)\right], \quad i,j,k,l = 1,\dots,D, \tag{1}$$

and the Bloch-Floquet periodic boundary conditions are

$$u(x+a_n) = u(x)e^{\mathbf{i}\gamma\cdot a_n} \;\; \forall x \in \{x \in \Theta \mid x + a_n \in \Theta\}, \quad n = 1,\dots,D. \tag{2}$$

The density, $\rho(x)$, and the material stiffness tensor, $C_{ijkl}(x)$, define the linear-elastic material properties of the unit cell, Θ. The wavevector, γ, and the eigenfrequency, ω, inversely describe the length and time scales of the wave. The displacement field, $u(x)$, describes the vibrational mode by which the wave travels. Finally, the lattice vectors, a_n, define the lattice of the periodic material, and $\mathbf{i}$ is the imaginary unit. The equation is a generalized eigenvalue problem with $u(x)$ and ω^2 as the eigenvector and eigenvalue respectively. To compute the dispersion relation, the eigenvalues of this equation must be determined as a function of γ, $\forall\gamma \in \Gamma$ (letting Γ denote the irreducible Brillouin zone).

With the exception of a few simple cases, analytical solutions for dispersion relations are not obtainable. For this reason, numerical solutions are often used to discretize the generalized eigenvalue problem (Eq. (1)). Although there are many numerical methods to compute dispersion relations, one of the most well-known (though not necessarily the most efficient) is the Finite Element Method. Within the framework of FEM, the dispersion relation can be computed by assembling the stiffness and mass matrices, $K(\gamma)$, and $M(\gamma)$, and solving the generalized eigenvalue problem for each γ in a discretization of the IBZ:

$$\left[K(\gamma) - \omega(\gamma)^2 M(\gamma)\right] u(\gamma) = 0. \tag{3}$$

There are several other specially tailored methods that have been shown to be more efficient than the finite element method when solving this particular PDE [16–23]. We note that the surrogate model we have developed can be used in conjunction with any of these more highly-specialized methods.

3. Traditional gradient-based optimization of dispersion relations

Gradient-based optimization is a powerful design tool that can be used in many contexts (including wave propagation [9,12,13]) to obtain material designs that exhibit desired behaviors. Designs are represented by a design vector, θ, containing parameter values that uniquely determine the geometry and constitutive properties of the unit cell design. The goal of design optimization is to determine a θ that results in the desired wave behavior, often achieved by maximizing or minimizing an objective function (which encodes design objectives) via gradient-based methods. To use these gradient-based methods, gradients of the objective function are needed, which in turn depend on gradients of quantities embedded in the dispersion relation. In this section, we quickly step through a few known equations commonly used in optimization problems surrounding wave behavior. In this context, K and M are not only functions of the wavevector, but also of the design vector, θ. Derivations of these equations can be found in Appendix C.

To modify the frequency, ω, at a specific wavevector by incrementally changing the design variable, the following sensitivity equation is useful. Assume all eigenvectors are normalized by the mass matrix; i.e., $u^* M u = 1$ where $(\cdot)^*$ denotes conjugate transpose. For ease of reading, assume here that θ is a scalar. These equations can easily be extended to vector θ by repeating the following computations for each entry of θ. The gradient of one frequency in the dispersion relation with respect the design variable is given by

$$\frac{\partial \omega}{\partial \theta} = \frac{1}{2\omega} u^* \left[\frac{\partial K}{\partial \theta} - \omega^2 \frac{\partial M}{\partial \theta} \right] u. \tag{4}$$

To modify the group velocity, $\frac{\partial \omega}{\partial \gamma}$, at a specific wavevector, the sensitivity computation is more complicated and more computationally expensive. The group velocity can be computed analytically by

$$\frac{\partial \omega}{\partial \gamma} = \frac{1}{2\omega} u^* \left[\frac{\partial K}{\partial \gamma} - \omega^2 \frac{\partial M}{\partial \gamma} \right] u. \tag{5}$$

The design sensitivity of the group velocity can be computed analytically by

$$\frac{\partial}{\partial \theta}\left[\frac{\partial \omega}{\partial \gamma}\right] = \frac{\partial^2 \omega}{\partial \theta \partial \gamma} = -\frac{1}{2\omega^2}\frac{\partial \omega}{\partial \theta} u^* A u + \frac{1}{2\omega}\frac{\partial u}{\partial \theta}^* A u + \frac{1}{2\omega} u^* \frac{\partial A}{\partial \theta} u + \frac{1}{2\omega} u^* A \frac{\partial u}{\partial \theta} \tag{6}$$

where $A = \frac{\partial K}{\partial \gamma} - \omega^2 \frac{\partial M}{\partial \gamma}$, and $\frac{\partial u}{\partial \theta}$ can be computed by solving the linear system:

$$\begin{bmatrix} \mathrm{Re}[K-\omega^2 M] & -\mathrm{Im}[K-\omega^2 M] & \mathrm{Re}\,[-Mu] & -\mathrm{Im}[-Mu] \\ \mathrm{Im}[K-\omega^2 M] & \mathrm{Re}[K-\omega^2 M] & \mathrm{Im}\,[-Mu] & \mathrm{Re}\,[-Mu] \\ \mathrm{Re}\,[u^* M] & -\mathrm{Im}\,[u^* M] & 0 & 0 \\ 0 \cdots 0 & 1\ 0 \cdots 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} \mathrm{Re}[\frac{\partial u}{\partial \theta}] \\ \mathrm{Im}[\frac{\partial u}{\partial \theta}] \\ \mathrm{Re}[\frac{\partial \mu}{\partial \theta}] \\ \mathrm{Im}[\frac{\partial \mu}{\partial \theta}] \end{Bmatrix} = \begin{Bmatrix} \mathrm{Re}[-\left(\frac{\partial K}{\partial \theta} - \mu \frac{\partial M}{\partial \theta}\right) u] \\ \mathrm{Im}[-\left(\frac{\partial K}{\partial \theta} - \mu \frac{\partial M}{\partial \theta}\right) u] \\ \mathrm{Re}[\frac{1}{2} u^* \frac{\partial M}{\partial \theta} u] \\ 0 \end{Bmatrix} \tag{7}$$

Not only is it computationally expensive to compute dispersion relations, but it is also computationally expensive to compute the design sensitivities of dispersion relations. Lu et al. report sensitivity analysis accounting for nearly half the time consumed by one iteration of optimization [13]. Later, we will explain how our surrogate model gives multiple options for computing sensitivities, opening the possibility for further computational savings. We also provide explicit equations for how to compute these gradients when using the surrogate model.

4. Gaussian process regression

4.1. Overview

Gaussian process regression (GPR) is a probabilistic regression method that constructs a predictive distribution for an unknown function, $f(x)$, by combining observations of the function with prior information [36]. Because the function is unknown, we think of $f(x)$ as being a random function distributed according to a Gaussian process $\mathcal{GP}(m(x), k(x, x'))$ defined by mean function $m(x)$ and covariance function $k(x, x')$.

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')) \tag{8}$$

Evaluation of this function at any finite number of points is described by a joint Gaussian distribution. Let m and k operate row-wise when given matrix input, with $m(X)$ yielding the mean vector corresponding to the points indicated by the rows of X, and $k(X, X')$ yielding the covariance matrix for the points in X and X'. With matrix X_* and vector f_* denoting the set of points for which we would like to make evaluations, the prior distribution is

$$f_* \sim \mathcal{N}\left(m(X_*), k\left(X_*, X_*\right)\right) \tag{9}$$

where $\mathcal{N}$ denotes a normal distribution.

Given a dataset $R = \{(x_i, y_i) | y_i = f(x_i) + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma_f^2)\}_{i=1}^N$ of (possibly noisy) observations of $f(x)$, the predictive distribution is formed by conditioning the prior distribution on the observations. With X and y denoting the observations, and I denoting the identity matrix, the predictive distribution is

$$\begin{aligned} f_* \mid X_*, X, y \sim \mathcal{N}\ \Big(& m(X_*) + k\left(X_*, X\right)\left[k(X, X) + \sigma_f^2 I\right]^{-1} y, \\ & k\left(X_*, X_*\right) - k\left(X_*, X\right)\left[k(X, X) + \sigma_f^2 I\right]^{-1} k\left(X, X_*\right) \Big)\ . \end{aligned} \tag{10}$$

Numerically, Eq. (10) is often implemented in two steps. The first step is to find the model weights, α, which are independent of the prediction locations, by solving the linear system in Eq. (11). Subsequently, predictions can be made at arbitrary locations X_* using the pre-computed model weights α in Eq. (10) rather than solving the system anew.

$$\alpha = [k(X, X) + \sigma_f^2 I]^{-1} y \tag{11}$$

Because GPR yields a predictive *distribution*, we not only infer a prediction as the mean of the distribution, but also a measure of uncertainty as the variance of the distribution.

4.2. Eigenfunction analysis of the covariance function

In later sections, we perform eigenfunction analysis of our proposed covariance function. Eigenfunction analysis is often used to assess the convergence and expressive power of a Gaussian process regressor. A function $\phi(x)$ is an eigenfunction of the covariance function $k(x, x')$ with eigenvalue λ, with respect to measure $\mu(x)$ if it satisfies

$$\int k\left(x, x'\right) \phi(x) d\mu(x) = \lambda \phi\left(x'\right). \tag{12}$$

These eigenfunctions form a basis which spans the space of all possible functions that can be attained by the mean function of the Gaussian process regressor, which is called the *hypothesis space*. A Gaussian process regressor can exactly represent any function in the span of the eigenfunctions with nonzero eigenvalues. A covariance function with an infinite number of nonzero eigenvalues is said to be *nondegenerate*, while one with only a finite number of nonzero eigenvalues is said to be *degenerate*.

A Gaussian process regressor with a nondegenerate covariance function has almost unlimited expressive power since its hypothesis space is the span of an infinite eigenbasis. In the limit of infinite observations, the model can exactly fit any function that can be represented by the generalized fourier series $\sum_{i=1}^{\infty} a_i \phi_i(x)$. In other words, for a GP regressor with a nondegenerate covariance function, the posterior mean function will converge to the expected value of the true underlying distribution as the number of observations goes to infinity (assuming the true underlying distribution is reasonably well-behaved).

In contrast, a Gaussian process regressor with a degenerate covariance function has limited expressive power, since its eigenfunctions span only a finite dimensional hypothesis space. For a GP regressor with a degenerate covariance function, in the limit of infinite observations, the posterior mean function will instead converge to the best μ-weighted L^2 approximation of the expected value of the true underlying distribution within the span of the eigenfunctions. Practically, this means that although a Gaussian process regressor with a degenerate covariance function may not converge to the exact solution in the limit of infinite regression points, it can still make a highly accurate prediction if its eigenbasis is comprehensive enough for the desired application.

5. Methods

In this section, we give the specifics of our surrogate model, and demonstrate its application to relatively simple one-dimensional dispersion computations. This toy problem is meant only to serve as an example — the substance of our results can be found in the following section. Traditional dispersion computations require the solution of an eigenvalue problem at each wavevector in a discretization of the irreducible Brillouin zone (IBZ). The goal of our surrogate model is to predict the eigenvalue solutions for each wavevector after solving the eigenvalue problems at only a small subset of these wavevectors. For most unit cells, which have many physical degrees of freedom, it is common in dispersion analysis to only consider the smallest N eigenvalues. In our surrogate model framework, we use N separate GPR models to fit the N eigenvalue bands, $\{\omega_i(\gamma)\}_{i=1}^{N}$.

The structures used in this section for demonstration are periodic one-dimensional bars. They can only transmit waves along their axis and have relatively simple dispersion relations. We randomly generate unit cell designs which have smoothly varying constitutive properties, except at randomly selected interfaces where the property values are allowed to be discontinuous (details in Appendix A.1). These interfaces lead to wave scattering behavior in the material – a mechanism well-known to cause band gaps in the dispersion relation. Examples of these unit cells and their dispersion relations are shown below in Fig. 1.

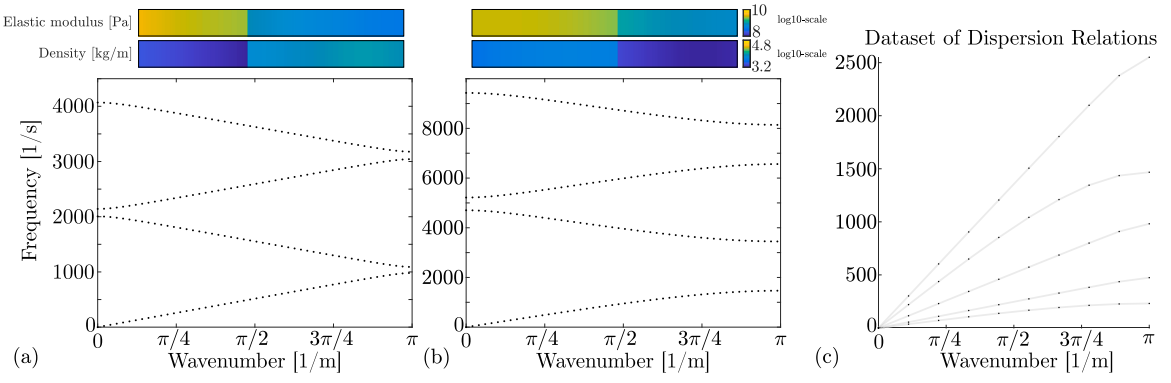


Fig. 1. (a) and (b) Two different 1-D unit cell designs and the first four bands of each design's dispersion relation, computed on a 50-point discretization of the IBZ. Each design in this figure has two interfaces of material property discontinuity (one in the interior of the unit cell and one at the boundary). (c) A dataset of the first band of the dispersion relations computed for 5 random 1-D structures drawn from the same distribution as the designs in (a) and (b), evaluated on a 10-point discretization of the IBZ.

5.1. Mean function

Our model uses a zero mean function ($m(x) = 0$). This is common in GPR, and does not mean that the predictive distribution will have a zero mean. Nonzero mean functions are often used when the function to be fit has an underlying trend that can be written

explicitly (such as a signal that, on average, increases linearly with time but has unpredictable variations around that average). Because dispersion relations have no such trends that can be explicitly written, we leave our mean function as zero. This leaves the entirety of the model's capacity to be carried by the covariance function.

5.2. Covariance function

The covariance function is usually the most crucial component in any Gaussian process model because it encodes the statistical coupling between function values at each point in the domain, defining underlying patterns and relationships in the data. Often, Gaussian process models use simple *isotropic* covariance functions such as the squared exponential or the Matérn covariance functions. Isotropic covariance functions are functions only of the distance between points in input space, meaning that they can only encode assumptions about the underlying function globally. These functions offer a convenient starting point for basic applications of Gaussian process modeling; however, they lack the expressiveness that can be offered by more thoughtful constructions of the covariance function. We propose a non-isotropic covariance function that encodes richer covariance information in a principled way that varies throughout the domain.

Our surrogate model uses a dataset of precomputed dispersion relations to construct its covariance function. Given a dataset $T = \{(\gamma, \omega)_i\}$ of dispersion relations evaluated at a fixed set of wavevectors Γ_T, we measure the covariance between the frequency values at all pairs of wavevectors in Γ_T. This covariance computation is extremely fast, especially when compared to the training process of neural networks. For example, to compute the covariance matrix for 10,000 dispersion relations with 10,000 points evaluated in each dispersion relation, only 6 s on a 9th generation Intel Core i7-9750H are required. Once the covariance matrix is computed, we use cubic interpolation of the measured covariance values to define the covariance function, $k : \Gamma^p \times \Gamma^q \to \mathbb{R}^{p \times q}$ mapping any two lists of wavevectors to the appropriate covariance matrix. Each eigenvalue band is described by its own separate covariance function. Although this data-driven covariance function is degenerate (i.e., in the limit of infinite regression points, the estimated dispersion relation converges to the best solution spanned by the finite eigenbasis rather than the exact numerical solution), it achieves higher accuracy than a data-agnostic covariance function given a limited number of regression points. Convergence to the exact numerical solution is not necessary in practical design problems, since even the exact solution does not perfectly represent real-world physics.

To demonstrate how this works for the one-dimensional case, we compute the covariance between frequency values at each pair of wavenumbers for the dataset shown in the right-most panel of Fig. 1. The results of this computation are shown in Fig. 2. For this demonstration, the dataset is small, with only 5 dispersion relations, and coarse, with the domain discretized into only 10 points. Even so, we can see that we have already captured critical information about the type of function we are trying to fit. For example, we see that the variance is 0 at $\gamma = 0$, and that the frequency at this wavenumber has no covariance with the frequencies at any other wavenumber. This makes sense because, for the first eigenvalue band, this frequency is always zero, representing a rigid body mode, regardless of the rest of the dispersion relation. Conversely, the variance is maximum at $\gamma = \pi$, and the frequency at this wavenumber is covariant with the frequency at many other wavenumbers, especially those nearby.

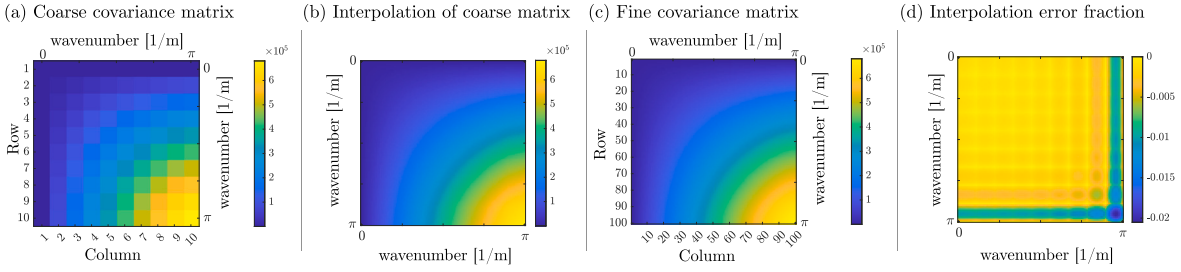


Fig. 2. (a) The covariance matrix computed directly from the coarse version of the dataset, using a 10-point discretization of Γ. (b) The covariance function generated by applying interpolation to the coarse covariance matrix. (c) the covariance matrix computed from the high resolution version of the same dataset, but using a 100-point discretization of Γ. This panel represents a visualization of the true covariance function. (d) The error fraction induced by approximating the high resolution covariance matrix with interpolation of coarse resolution covariance matrix. The error fraction is small, achieving a maximum absolute deviation of about 2%.

In later sections, when working with 2D structures, there are two wavevector components to consider. In these cases, the covariance function should be thought of as a continuous scalar-valued function of four scalar variables, and covariance interpolation must happen in this four-dimensional space. Regardless, for all computational operations other than interpolation, standard matrix operations can still be applied after reshaping the four-dimensional covariance array into a matrix.

With the covariance and mean functions defined, our Gaussian process prior is defined. We can visualize our prior by sampling it. With the exception that the samples are allowed to assume negative values, they have characteristics similar to the dispersion bands (see the first panel of Fig. 3). This shows that our covariance function yields a prior that encodes an appropriate hypothesis space, even before sampling any regression points.

5.3. Acquisition function

To fit the model to a specific dispersion relation, we supply it with a set of points (*regression points*) sampled from that dispersion relation. Because each regression point has a computational cost, it is beneficial to try to minimize the number of regression points needed, and maximize the information that each regression point contributes. An *acquisition function* is a function that tells the model how to select the locations of regression points. Here, we propose a greedy acquisition function that sequentially determines the most informative next point to sample. The acquisition function we have constructed uses the covariance function to evaluate the posterior variance over a discretization of the IBZ. The variance can be computed by evaluating the diagonal entries of the posterior covariance, $C_{posterior}$, given by

$$C_{posterior} = k\left(\Gamma_*, \Gamma_*\right) - k\left(\Gamma_*, \Gamma_R\right)\left[k(\Gamma_R, \Gamma_R) + \sigma_\omega^2 I\right]^{-1} k\left(\Gamma_R, \Gamma_*\right) \tag{13}$$

where Γ_* are the wavevectors constituting the discretization of the IBZ, Γ_R are the wavevectors representing the locations of the optimal regression points, and I denoting the identity matrix.

Starting with no regression points (Γ_R is empty, and the entire second term is zero), we consider the wavevector in Γ_* with the highest variance to be the most informative next wavevector to sample, so it becomes the next regression point location. We append this wavevector to Γ_R, re-evaluate $C_{posterior}$, and repeat. This process can be iterated until a desired number of regression point locations is obtained.

When this process is finished, Γ_R represents an ordered list of optimal regression points. During the fitting process (described in Section 5.4), subsets of Γ_R can be selected by taking the first N elements if the computational budget does not permit the solution of the phononic eigenvalue problem for every wavevector in Γ_R. Because our covariance function is degenerate, there is a finite limit on the number of regression points that can be effectively used by the model, and Γ_R will not be able to grow beyond this size (the inversion in Eq. (13) will run into conditioning issues). This limit can be quite large, and depends on the size and resolution of the dataset that was used to construct the covariance function.

Conveniently, our acquisition function is entirely independent of the observed values of the regression points. This means the locations (wavevectors) of the optimal regression points can be determined ahead of time without actually making observations of the regression points themselves, allowing for the parallel computation of regression point values (frequencies) later. Additionally, this means that the final Γ_R is dependent only on the covariance function, and does not change as a function of the specific dispersion relation that the model is being fit to.

We visualize how our acquisition function works for a simple one-dimensional problem in the bottom row of panels of Fig. 3. Without performing eigenvalue computations, we iteratively evaluate the posterior variance, and append the point of maximum variance to the ordered list of regression points.

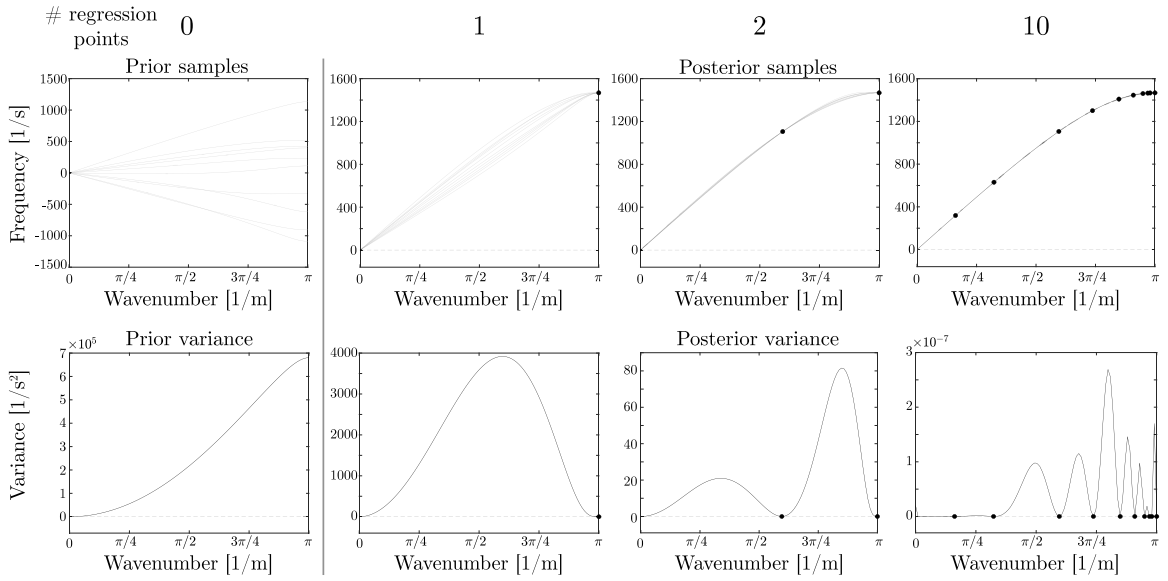


Fig. 3. Top row: samples generated from the posterior (predictive) distribution with 0, 1, 2, and 10 regression points sampled (note that when 0 points are sampled, the posterior is the prior). As more points are sampled, the posterior distribution becomes much tighter, as can be seen by the decrease in variation of samples drawn from the posterior distribution. Bottom row: the variance of the posterior distribution with 0, 1 2, and 10 points sampled. The first three panels of the bottom row demonstrate the first three sequential steps of the algorithm used to determine the order in which points are sampled.

5.4. Fitting the model

With the mean, covariance, and acquisition functions determined, we can fit the GP model to a specific dispersion relation. To fit the model, we supply it with a set of regression points sampled from that dispersion relation. Our acquisition function determines the

locations (wavevectors) of these points ahead of time. To perform the fit, we need to *observe* the eigenfrequency solutions associated with these wavevectors by solving the corresponding eigenvalue problem. Because the locations of the regression points have been determined ahead of time, these computations can be done in parallel. After observing regression points (Γ_R, Ω_R), we compute the model weights α by solving the linear system

$$\alpha = [k(\Gamma_R, \Gamma_R) + \sigma_\omega^2 I]^{-1} \Omega_R \tag{14}$$

where Γ_R and Ω_R are the wavevectors and frequencies of the regression points, σ_ω^2 is the assumed variance of the noise on the regression point measurements, and I is the identity matrix. Because our regression points are generated by a well-chosen finite element model, it is reasonable to assume values of σ_ω^2 as low as 0, if assuming the eigenvalue solutions are exact, or 10^{-16} to represent the uncertainty associated with using double precision floating point data format. We find that using $\sigma_\omega^2 = 0$ works well for our purposes, and does not produce significantly different behavior from using $\sigma_\omega^2 = 10^{-16}$.

Once the model is fit to the regression points, the predictive distribution can be computed (Eq. (15)). Because the predictive distribution is Gaussian, the maximum a posteriori estimate of the full dispersion relation is the mean of the predictive distribution. Further, the diagonal of the covariance matrix of the predictive distribution is the variance of each point being predicted. The variance gives us a metric for the model's predictive uncertainty at each wavevector in Γ_*. The predictive distribution is

$$\begin{aligned} \Omega_* \mid \Gamma_*, \Gamma_R, \Omega_R \sim \mathcal{N} \big(& k\left(\Gamma_*, \Gamma_R\right) \alpha, \\ & k\left(\Gamma_*, \Gamma_*\right) - k\left(\Gamma_*, \Gamma_R\right) \left[k(\Gamma_R, \Gamma_R) + \sigma_\omega^2 I\right]^{-1} k\left(\Gamma_R, \Gamma_*\right) \big) \end{aligned} \tag{15}$$

where Γ_* are the wavevectors at which we would like to predict the frequency values of the dispersion band, and Ω_* are the predicted frequencies. In the top row of Fig. 3, we draw samples from this predictive distribution to show its evolution as more regression points are used. When fewer regression points are used, the predictive distribution is quite broad, indicating uncertainty in the prediction. However, as more regression points are used, the predictive distribution becomes extremely tight, indicating much more certainty.

5.5. Gradients for gradient-based optimization

So far, we have detailed how to make predictions with our model. However, to use our model in gradient-based design optimization, we must be able to supply efficient design sensitivity computations. In this subsection, we describe how to compute gradients for frequency optimization and group velocity optimization.

5.5.1. Gradients for frequency optimization

Typically, computing the design sensitivity of the frequency at one wavevector requires not only the eigenvalue solution, but also the eigenvector solution (Eq. (4)). However, our model predicts only the eigenvalue, not the eigenvector. There are at least two ways to amend this.

First, we could simply use the surrogate model to infer the dispersion relation, and use this inference to determine the wavevector at which we need to optimize the frequency. Then we could solve the eigenvalue problem corresponding to that single wavevector, and compute the design sensitivity using Eq. (4). This is a great option if we only need to compute the design sensitivity of the frequency at a single wavevector, but do not know a priori at which wavevector (e.g. for bandgap optimization).

Alternatively, if the eigenvalue problems are particularly expensive, or if there are many wavevectors at which we desire to optimize the frequency, we can use a trick to avoid solving more eigenvalue problems. By employing the chain rule we can compute the gradient of any set of frequencies without performing additional eigenvalue computations. To do this, we first compute the gradient of the frequency at any wavevector with respect to the frequencies measured at the regression points. This gradient is defined by the GPR model. Then, since we do know the eigenvector at the regression points, we compute the gradient of the frequency at the regression point with respect to the design variables. This second gradient is defined by Eq. (4). We then multiply these two quantities to obtain the design sensitivity of any frequencies we are interested in optimizing. That is,

$$\frac{\partial \Omega_*}{\partial \theta} = \frac{\partial \Omega_*}{\partial \Omega_R} \frac{\partial \Omega_R}{\partial \theta} \tag{16}$$

where

$$\frac{\partial \Omega_*}{\partial \Omega_R} = k\left(\Gamma_*, \Gamma_R\right) \left(k(\Gamma_R, \Gamma_R) + \sigma_\omega^2 I\right)^{-1} \tag{17}$$

and

$$\left(\frac{\partial \Omega_R}{\partial \theta}\right)_{i,j} = \frac{1}{2(\Omega_R)_i} (U_R^*)_{i,:} \left[\frac{\partial K}{\partial \theta_j} - (\Omega_R)_i^2 \frac{\partial M}{\partial \theta_j}\right] (U_R)_{:,i} \tag{18}$$

where U_R is a matrix containing the eigenvectors of the regression points, $(\cdot)^*$ denotes conjugate transpose, and the colon subscript indicates every element along the corresponding axis of the array (e.g. $(U_R)_{:,i}$ represents the ith column of U_R). Note that although this method does not require further eigenvalue solutions, it does require traditional sensitivity computations for a potentially large number of regression points. In practice, these two costs should be weighed against each other.

5.5.2. Gradients for group velocity optimization

Frequency is not the only subject of dispersion optimization. Group velocity is also a common optimization objective [12]. Computing the design sensitivity of group velocity again requires eigenvector information at the wavevector of interest. It should be noted that traditional sensitivity computations for group velocities are more expensive than traditional sensitivity computations for frequencies because they require solving an extra system of equations. As with frequency optimization, there are two methods that we can use which will be advantageous in different situations.

First, we can use the surrogate model to infer the dispersion relation, and identify a wavevector of interest. Then we can solve the eigenvalue problem at that wavevector, and use Eqs. (6), (7) to compute the group velocity design sensitivities.

Alternatively, we can avoid group velocity sensitivity computations altogether by employing the chain rule. Using the surrogate model, the analytical expression for the predicted group velocity, Ψ_*, is

$$\Psi_* = g(\Gamma_*, \Gamma_R)\alpha \tag{19}$$

where g is the gradient of the covariance function with respect to its first input. Since the covariance function is fixed, this can be estimated numerically once, and stored for further use. Subsequently, we apply the chain rule to write the sensitivity of the predicted group velocity with respect to the design variables:

$$\frac{\partial \Psi_*}{\partial \theta} = \frac{\partial \Psi_*}{\partial \Omega_R}\frac{\partial \Omega_R}{\partial \theta} \tag{20}$$

where

$$\frac{\partial \Psi_*}{\partial \Omega_R} = g(\Gamma_*, \Gamma)(k(\Gamma, \Gamma) + \sigma_\omega^2 I)^{-1} \tag{21}$$

and $\frac{\partial \Omega_R}{\partial \theta}$ is again given by Eq. (18).

Using the second method requires traditional frequency sensitivity computations for a potentially large number of regression points. However, it gracefully avoids the need for any traditional group velocity sensitivity computations, which require solving a rather large linear system. This second method is advantageous to use when we desire to optimize group velocity at a large number of wavevector points.

6. Results

In this section, we apply our surrogate model to dispersion relations of two-dimensional materials and evaluate its performance against traditional methods and an existing deep-learning-based surrogate model.

6.1. Two-dimensional dispersion relations

When modeling materials in two dimensions, the evaluation of the dispersion relation becomes more complicated and more computationally costly than in one dimension. This is for two reasons. First, a two-dimensional material requires more degrees of freedom to be accurately modeled. Second, wave propagation in a 2-D material must be described by a *wavevector*, rather than a *wavenumber*, since wave propagation may occur in any combination of the two directions. This means that one must sweep over a 2-D parameter space of wavevectors, solving an eigenvalue problem at each point in this 2-D space. These issues are further exacerbated when working with 3-D materials.

For this reason, when considering 2-D and 3-D materials, engineers and scientists often evaluate the dispersion relation only on the IBZ contour or a "virtual IBZ contour" rather than the full IBZ. The definition of the IBZ contour is dependent on the symmetries of the unit cell, and is defined as the boundary of the IBZ for symmetry groups with reflection or glide-reflection symmetries. For symmetry groups without reflection or glide-reflection symmetries, the IBZ contour is not defined since the IBZ can be arbitrarily rotated, giving an infinite number of possible choices for the IBZ contour, and an infinite number of possible resulting dispersion band diagrams.

For example, Fig. 4 shows a unit cell lacking symmetry. Unit cells lacking symmetry belong to the $p1$ plane symmetry group (or *wallpaper group*). The IBZs of unit cells in the $p1$ symmetry group can be rotated arbitrarily, so a unique IBZ contour is not defined. As a substitute, the authors of [15] use a "virtual IBZ contour". Although the construction of this contour is entirely artificial, we include it because many people in the acoustics and phononics field are accustomed to seeing dispersion information in this format. Even for unit cells that have well-defined IBZ contours, the commonly used *contour shortcut* yields only partial information about wave propagation through the medium. The goal of our surrogate model is to increase the accessibility of dispersion information over the *full* IBZ.

In Fig. 4, we show the unit cell along with the first three bands of its dispersion relation evaluated along the "virtual IBZ contour", and over the full IBZ. We also show the inference of the surrogate model over these domains.

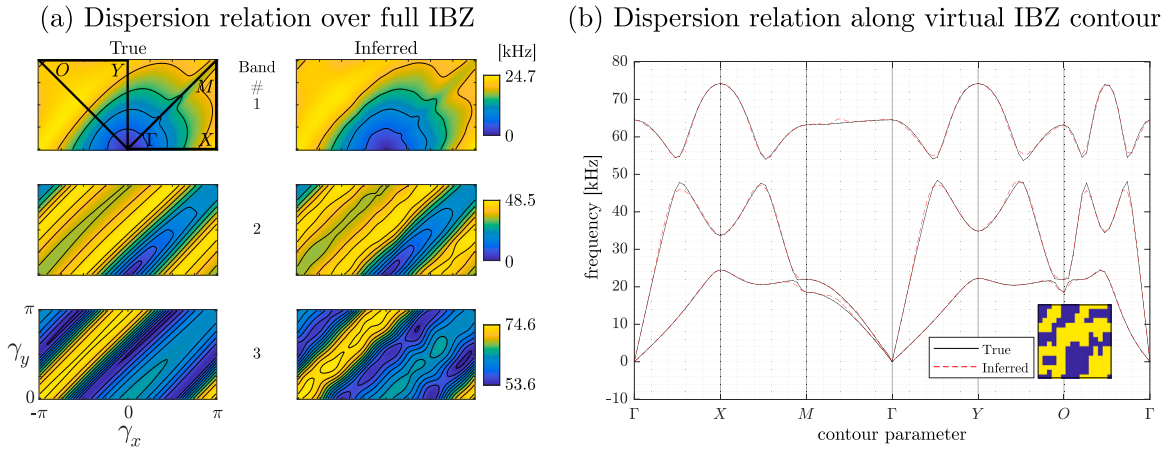


Fig. 4. (a) The dispersion relation of a bimaterial unit cell evaluated over the full IBZ. The true dispersion band is shown in the left column, and the inference given by the surrogate model is shown in the right column. Here, the surrogate model uses a training set of 2000 dispersion relations to construct the covariance function, and uses 50 regression points (regression points not shown). (b) The same dispersion relation evaluated along the "virtual IBZ contour", $\Gamma X M \Gamma Y O \Gamma$. The unit cell design is shown in the inset, where yellow represents a stiff and heavy material ($E = 200$ GPa, $\rho = 8000$ kg/m^3) and blue represents a softer and less dense material ($E = 200$ MPa, $\rho = 800$ kg/m^3). The side length of the unit cell is $a = .01$ m. This unit cell has no symmetry, making it part of symmetry group $p1$ [15]. The IBZ of this symmetry group can be arbitrarily rotated, so a well-defined IBZ contour does not exist [15]. However, since so many are accustomed to visualizing dispersion information in this format, we have included the dispersion relation evaluated on the "virtual IBZ contour" as defined in [15]. Here, we again show the true dispersion relation along with the inferred dispersion relation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6.2. Generating random designs

In this section, we consider 2D unit cell designs of the $p1$ symmetry group (i.e. no symmetry) using a plane stress model. To generate random unit cell designs, we sample a Gaussian process with a 2-D periodic covariance function. To construct this covariance function, we start with a 1-D periodic covariance function,

$$k_{\text{per-1D}}\left(x, x'\right) = \sigma_f^2 \exp\left(-\frac{2}{\ell^2}\sin^2\left(\pi\frac{|x - x'|}{p}\right)\right) \tag{22}$$

where σ_f controls the overall variance of the Gaussian process, ℓ controls the lengthscale of the randomness, and p is the period. In our case, we set p to be the side length, a, of the unit cell. We form the 2-D periodic covariance function as the product of the 1-D periodic covariance function applied to each direction:

$$k_{\text{per-2D}}\left(x, y, x', y'\right) = k_{\text{per-1D}}\left(x, x'\right) k_{\text{per-1D}}\left(y, y'\right) \tag{23}$$

where x and y are the horizontal and vertical coordinates of locations in the unit cell. We emphasize that the Gaussian process used to generate unit cell designs has no direct connection with the Gaussian process used by the surrogate model for inference.

Beyond the periodic covariance parameters (which control the variance and the lengthscale of features in the unit cell), there are several other parameters that define the unit cell. The lattice parameter a dictates the side length of the square unit cell. The resolution of the unit cell is determined by the integer N_{pix}, which defines the number of pixels both vertically and horizontally. The constitutive properties of each pixel are bounded by E_{min}, E_{max}, ρ_{min}, ρ_{max}, ν_{min}, and ν_{max}, corresponding to elastic modulus, mass density, and Poisson's ratio respectively. The number of unique materials in each design is controlled by N_{mat} (for example, a bimaterial unit cell would be defined by $N_{mat} = 2$). Further details of unit cell design generation are given in Appendix A.2. Examples of how these unit cells might look are provided in Section 6.6.

6.3. Visualization of the acquisition function for the two-dimensional problem

In this section, we visualize the covariance and acquisition functions for the two-dimensional problem. In Fig. 5, we show the evolution of the acquisition function and the resulting variance fields for a covariance function constructed from a dataset of 2000 dispersion relations.[1] As we compute the order of regression point sampling, we can observe that points that are not near to each other can be highly covariant. For example, the covariance function is taking advantage of the periodicity of the dispersion relation. In the example shown in Fig. 5, the unit cells in the dataset used to construct the covariance function belong to the *p1* plane symmetry group (they have no symmetry), which is a superset of all other plane symmetry groups. Due to the construction of this symmetry group's IBZ and the periodicity of the dispersion relation, the solution along the boundary of the IBZ must be symmetric

[1] The dataset was constructed with the following parameters: $\sigma_f = 1$, $\ell = 0.5$, $a = 1$ m, $N_{pix} = 16$, $N_{mat} = 2$, $E_{min} = 200$ MPa, $E_{max} = 200$ GPa, $\rho_{min} = 800$ kg/m^3, $\rho_{max} = 8000$ kg/m^3, $\nu_{min} = 0$, $\nu_{max} = 0.5$. The wavevector resolution of each dispersion relation in the dataset is 101×51.

across the line $\gamma_x = 0$. Indeed, we see from the $(1,1)$ and $(2,1)$ panels that a single regression point on the far left side of the IBZ causes a "cold spot" in the variance field on the right side of the IBZ in the region that is "close" to the regression point when considering the periodicity of the dispersion surface. This indicates that the model gains insight about the function value on the right side of the domain from information observed on the left. Our covariance function is able to implicitly learn crucial patterns such as periodicity and symmetry from the data, without the need for a user to manually encode these assumptions.

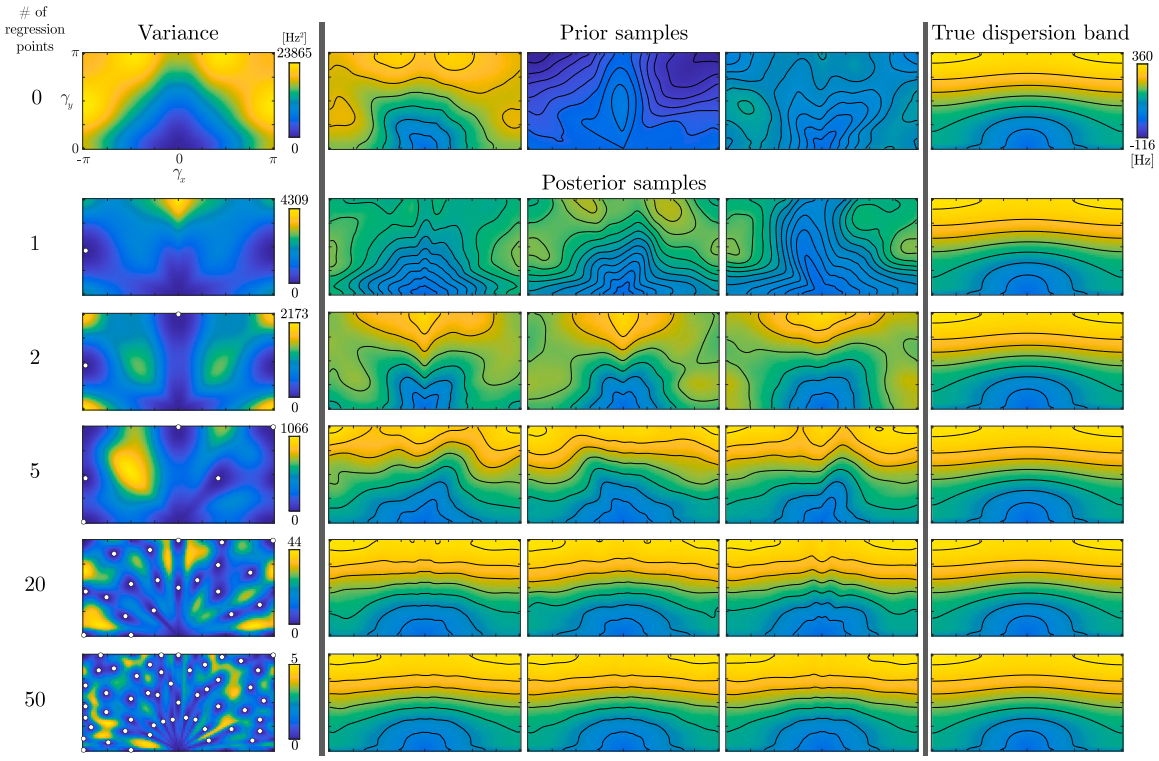


Fig. 5. Left column: from top to bottom, we show the progression of regression point selection for 0, 1, 2, 5, 20, 50 regression points. Right column: the true dispersion relation for one particular unit cell design. Each image in this column is identical. Center three columns: three random samples drawn from the posterior distribution for 0, 1, 2, 5, 20, 50 regression points. Note that when the model uses 0 regression points, the posterior distribution is identical to the prior distribution. Also, we observe that as more regression points are used, the samples become more similar to each other, and to the true dispersion relation in the right column. Further, note that the posterior mean function is the final hypothesis of the surrogate model, not a randomly drawn posterior sample.

6.4. Inference examples

We now show examples of our surrogate model infering the dispersion relation for two-dimensional structures using a sparse set of regression points in Fig. 6. In each case, the model uses a dataset of 2000 dispersion relations (the same dataset used to generate Fig. 5) and 50 regression points (the same regression points shown in the bottom left panel of 5). The dataset was constructed with the following parameters: $\sigma_f = 1$, $\ell = 0.5$, $a = 1\,\text{m}$, $N_{pix} = 16$, $N_{mat} = 2$, $E_{min} = 200\,\text{MPa}$, $E_{max} = 200\,\text{GPa}$, $\rho_{min} = 800\,\text{kg/m}^3$, $\rho_{max} = 8000\,\text{kg/m}^3$, $\nu_{min} = 0$, $\nu_{max} = 0.5$. The wavevector resolution of each dispersion relation in the dataset is 101×51. In Fig. 6 we use 50 regression points to achieve a relatively accurate prediction; however, in practice, the number of regression points can be adjusted fluidly to strike the right balance of accuracy versus computational cost depending on the application.

6.5. Eigenanalysis of the covariance function

In this section, to examine the relationship between the size of the covariance function training set and the expressivity of the model, we examine the spectra of covariance functions based on datasets of different sizes. Recall from Section 4.2 that the number of nonzero eigenvalues of the covariance function corresponds to the size of the covariance function's eigenbasis, which is the basis the model uses to fit to the regression points. A covariance function with a more comprehensive eigenbasis results in a model with more expressive power, and a model that can make use of a larger number of regression points.

For a range of logarithmically spaced dataset sizes, we compute the spectrum of the corresponding covariance function, and plot the spectra in Fig. 7. As the size of the dataset increases, the number of nonzero eigenvalues increases, meaning that the covariance function's eigenbasis is expanding to be more expressive. Eventually, however, the addition of more dispersion relations to the dataset has a decreasing effect on the spectrum of the covariance function. For the largest dataset size (20,000 dispersion relations) we plot the eigenfunctions corresponding to the 10 largest eigenvalues of the covariance function in Fig. 8. These eigenfunctions

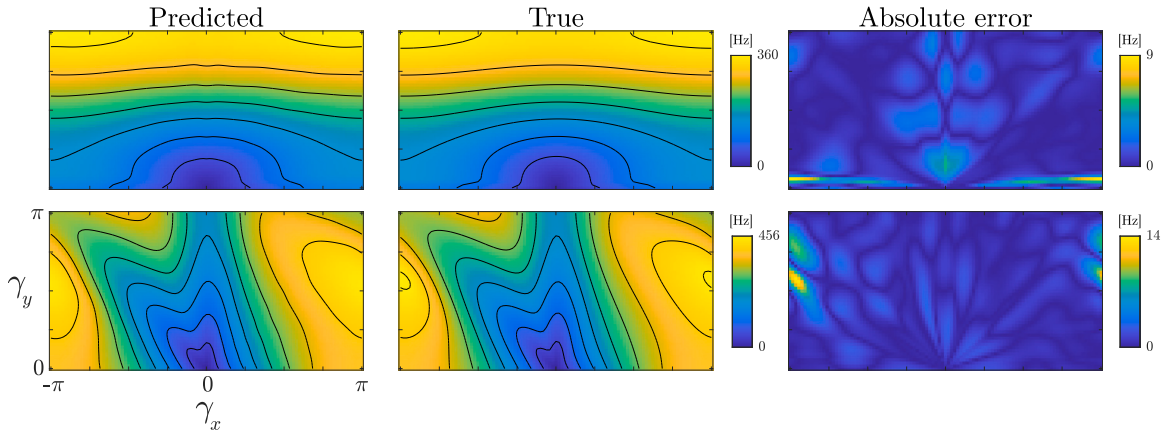


Fig. 6. A comparison of the prediction against the truth for the first eigenvalue band of two different unit cells over the full IBZ. In each case, the surrogate model uses a covariance function trained on 2000 dispersion relations and 50 regression points. The absolute error is shown on the right for each scenario.

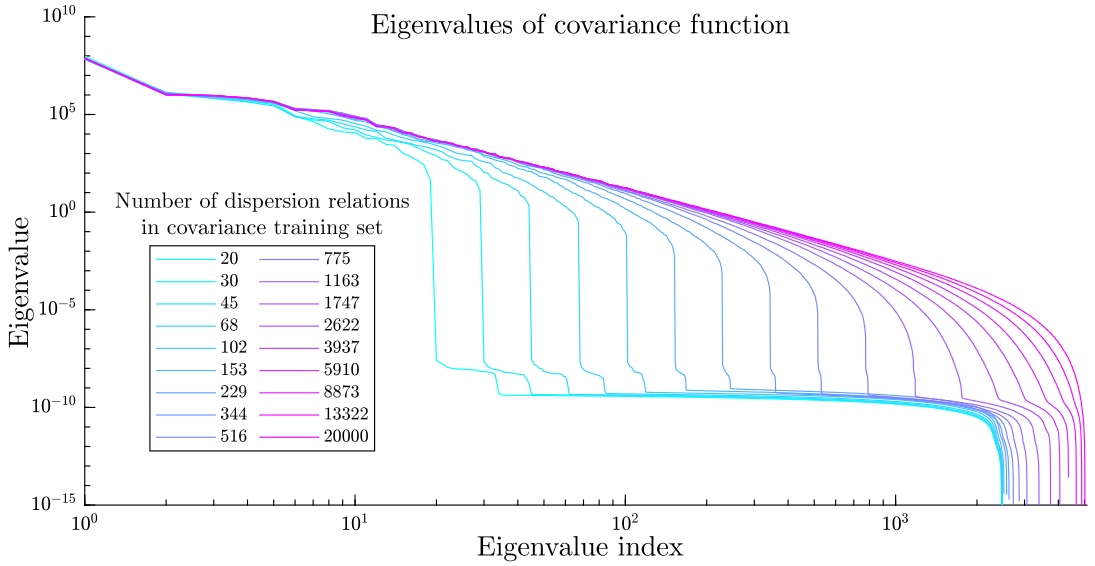


Fig. 7. The spectra of covariance matrices computed from datasets containing different numbers of dispersion relations. The wavevector resolution of the datasets is 101 × 51, and the unit cell designs are 16 × 16 pixels. Because we plot in log scale, eigenvalues that are zero are not plotted. As expected from basic statistics, we observe that datasets with larger numbers of dispersion relations produce covariance matrices with more nonzero eigenvalues. This indicates that covariance functions generated from larger datasets will have larger eigenbases, and therefore have more expressive power. However, we observe that the change in spectrum appears to slow down at some point, and we get diminishing returns after about 1700 dispersion relations. We can expect this "saturation" effect to happen at different points in different scenarios, and will depend on many factors including the resolution of the dispersion relations in the datasets and the method of generating unit cell designs.

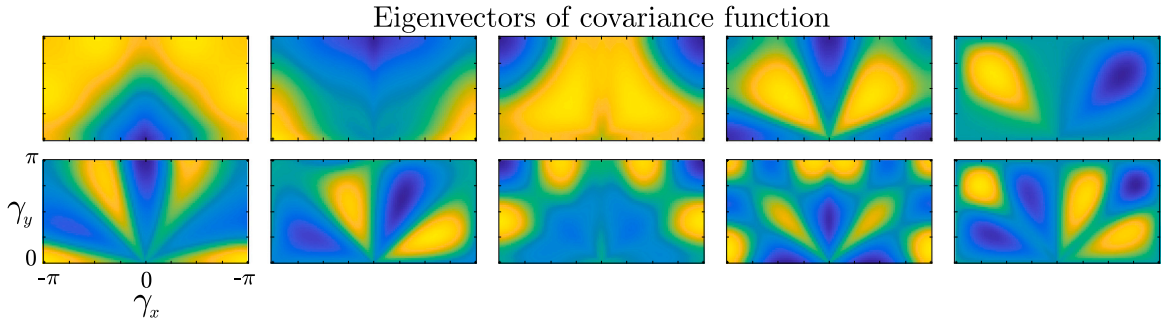


Fig. 8. The first 10 eigenfunctions of the covariance function that uses training set containing 20,000 dispersion relations. The eigenfunctions form a basis for the hypothesis space of the Gaussian process regressor.

are the first ten elements of the eigenbasis, and can be thought of as the ten most prominent and fundamental function shapes that the model uses to fit a dispersion relation.

The expressivity of the model (i.e. the broadness of the hypothesis space) depends on the dataset used to construct the covariance function. In general, larger datasets will allow for more expressivity. This is not surprising, since other data-driven methods such as neural networks require larger amounts of data to characterize more complex functions. However, there is a difference in how the model complexity is determined. For a neural network, the hypothesis space is determined by the network architecture, which is typically determined before training the model. If too little training data is supplied, the hypothesis space of the network remains large, but the model's performance will be low since the training data was not sufficient to narrow down the hypothesis space. In contrast, the expressivity of our model's data-driven covariance function is determined by the dataset itself. With our method of covariance function construction, a small dataset will automatically result in a smaller hypothesis space, and a larger dataset will automatically result in a larger hypothesis space, without the need to adjust complicated model hyperparameters (such as network architecture).

6.6. Model evaluation

In the following subsections, we compare the performance of the GPR model against two other models in the context of the two-dimensional problem. The first model is linear interpolation, representing the traditional model for computing dispersion relations.[2] The second model is an existing convolutional neural network surrogate model from the literature [32].

6.6.1. The datasets

To evaluate the performance of the GPR model, we consider the accuracy of the model as a function of the size of the covariance training set and the number of regression points for both in-distribution and out-of-distribution scenarios. We generate three datasets, one for training the covariance function, one for testing in-distribution performance, and one for testing out-of-distribution performance. The training dataset contains dispersion relations of 5000 unit cells generated as described in Section 6.2, with periodic kernel parameters $\sigma_f = 1$ and $l = 0.5$ at a resolution of 32×32 ($N_{pix} = 32$). The unit cells have a lattice parameter $a = 1$ m, are divided into 2 materials ($N_{mat} = 2$), and have material properties $E_{min} = 200$ MPa, $E_{max} = 200$ GPa, $\rho_{min} = 800\,\mathrm{kg/m^3}$, $\rho_{max} = 8000\,\mathrm{kg/m^3}$, $\nu_{min} = 0$, and $\nu_{max} = 0.5$. The in-distribution test dataset contains dispersion relations of 100 unit cells randomly generated using the same parameters as the training set (hence, *in-distribution*). The out-of-distribution test dataset contains dispersion relations of 100 unit cells randomly generated with the same parameters as the training set, except $l = 1$, $N_{mat} = 3$, $E_{min} = 200$ GPa, $E_{max} = 200$ TPa. These four parameters cause the distributional shift between the training set and the out-of-distribution test set. Although the elastic moduli in the out-of-distribution dataset are outside of the range of realistic materials, we use them to show that our model works when the material properties and frequency ranges of the training set and test set are vastly different. The training set uses a wavevector resolution of 101×51, and the test sets use a wavevector resolution of 501×251. Examples of unit cells drawn from each distribution are shown in Fig. 9.

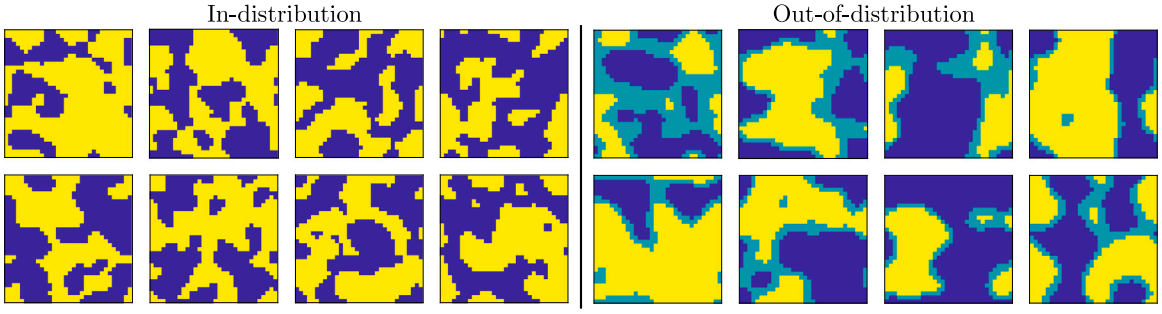


Fig. 9. A visualization of the unit cell designs for the two distributions used to generate the datasets. Blue represents 0, teal represents 0.5, and yellow represents 1. On the left we show in-distribution designs, generated with the same parameters as the training set. On the right, we show out-of-distribution designs, generated by parameters different from the training set. The out-of-distribution dataset is generated with a larger length scale parameter, an extra material, and elastic moduli 1000× larger (note that the colorscale does not show this stiffness increase). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6.6.2. Performance metrics

To measure the error of the model, we use the L^2 norm and the H^1 norm, defined in Eqs. (24), (25). The L^2 norm allows us to assess how well the model infers frequency values, and the H^1 norm to assess how well the model infers group velocity values. Errors are evaluated at the resolution of the test sets, which are about five times more resolved along each of the two wavevector

[2] For the traditional linear interpolation model, one simply evaluates the dispersion relation on a grid of evenly spaced points and linearly interpolates if evaluation is needed at points that do not belong to the grid.

directions, as described in Section 6.6.1. After evaluating model error on each eigenvalue band of each dispersion relation in the test set, we average over eigenvalue bands, and take the 0.95 quantile over dispersion relations to get a sense for worst-case error. The norms are given as

$$\|f\|_{L^p}^p = \int_D \|f(x)\|^p \, \mathrm{d}x \tag{24}$$

$$\|f\|_{H^k} = \left(\sum_{0 \le |\alpha| \le k} \|\partial^\alpha f\|_{L^2}^2 \right)^{1/2} \tag{25}$$

where ∂^α denotes the multi-index partial differential operator, $|\alpha| = \sum_i \alpha_i$, and D is some domain over which we are interested in measuring the function. In our case, D will be the IBZ, and the function f will be the error of the model.

6.6.3. Comparison against linear interpolation model

To compare against the linear interpolation model, we consider the number of regression points used by each model and the error that each model incurs in its prediction. In this subsection, because of the 1000× shift in elastic modulus of the out-of-distribution test set, we divide the out-of-distribution errors by $\sqrt{1000}$ so that they can be readily compared to the in-distribution errors.[3] We find that the GPR model always uses fewer regression points than the corresponding linear interpolation model achieving the same accuracy. At most, in this particular study, the GPR model uses 7.75 times fewer regression points than the equivalent linear interpolation model. Because almost all of the computational cost is due to regression point eigenvalue computations, this can be seen as a speed up of approximately the same factor. The results are shown below in Fig. 10.

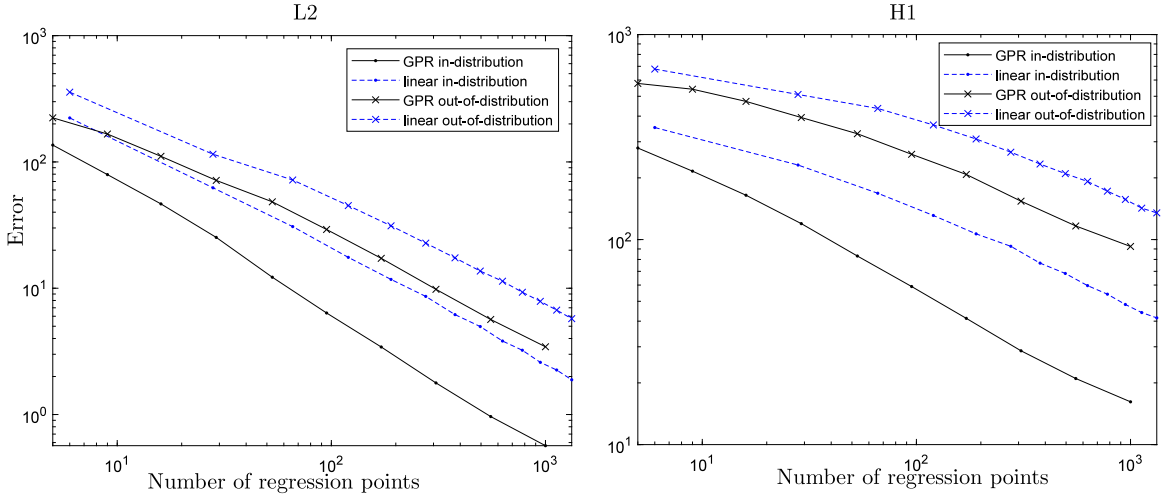


Fig. 10. A comparison of the GPR model against the traditional linear interpolation model. Performance is evaluated on two datasets: in-distribution and out-of-distribution. In both cases, the GPR model performs substantially better than the linear interpolation model. In the H^1 plot for in-distribution performance, the GPR model is shown to achieve the same error as the linear interpolation model while using up to 7.75× fewer regression points.

6.6.4. Comparison against existing CNN surrogate model

We compare our model performance to the performance of the CNN surrogate model proposed by Finol et al. [32]. Please refer to Appendix B for details about the CNN architecture and training. Note that the CNN uses the material properties of the unit cell in the input array. This is in contrast to the GPR model, which does not directly observe the material properties of the unit cell, but instead observes regression points of the dispersion relation.

In this subsection, we allow the GPR models to observe only 50 regression points, and limit both the CNN and GPR models training set sizes to range from only 50 to 5000 dispersion relations. For each training set size, we train the CNN for a maximum of 30 epochs with early stopping based on a validation set. Training the CNN on the training set of 5000 dispersion relations takes approximately 24 minutes/epoch on an NVIDIA GeForce RTX 3080 Ti GPU. For each training set size, we also compute a new covariance function for the GPR, taking only 1 s for the largest training set size (5000). Because training a neural network depends on the randomly initialized weights of the network, we repeat the training 10 times with different random initializations to get a sense for the variation in results. After measuring the L^2 and H^1 norms of the error, we average over all 10 eigenvalue bands, and

[3] This exactly transforms the errors to what their values would be if the modulus range were 1000× smaller, since the dispersion relation of a unit cell whose modulus is increased by 1000× will experience an upshift in frequency at a factor of $\sqrt{1000}$, and because both the linear interpolation and GPR models' predictions (and therefore errors) are linearly proportional to the frequency values of the regression points.

take the 95% quantile error over unit cells. Additionally, for the CNN, we measure the standard deviation of this error over the ten trials and plot the ± one standard deviation spread.

The results (Fig. 11) show that the GPR model achieves errors significantly lower than the CNN for all training set sizes. This is because we are using relatively small training sets, and neural networks typically require more data to train effectively. For reference, Finol et al. used a minimum of 20,000 and up to 100,000 dispersion relations to train their CNN. In general, it is a nontrivial task to adjust the architecture (and therefore expressivity) of a neural network to be appropriate for the amount of training data that you have. This process generally involves a substantial amount of trial-and-error and hyperparameter tuning, which can be costly in terms of time and effort. Note that in this section, we do not adjust the out-of-distribution errors by the factor of $\sqrt{1000}$ because this transformation would not be valid for the CNN, so in-distribution and out-of-distribution results cannot be directly compared against each other. For both in-distribution and out-of-distribution, we see the CNN gradually improve with training set size in the L^2 norm, but not in the H^1 norm. This makes sense, because the loss function that the CNN uses (mean squared error) is not sensitive to mismatches in gradients (H^1 error).

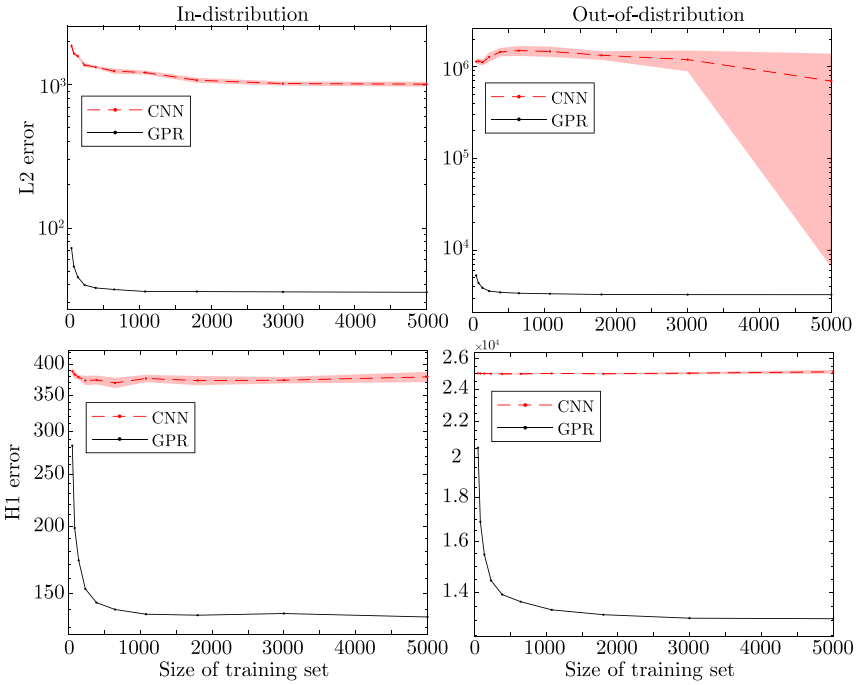


Fig. 11. Comparison of GPR surrogate model against CNN surrogate model for in-distribution and out-of-distribution performance in L^2 and H^1 norms. The GPR model performs with much lower error than the CNN for these relatively small training set sizes.

7. Discussion

Interest in the design of multi-functional wave devices and the high computational cost of dispersion relation computations motivate the need for more efficient methods of dispersion relation evaluation, including surrogate models. We have introduced a new GPR-based surrogate model for the accelerated computation of dispersion relations over the full irreducible Brillouin zone.

The GPR model builds its covariance function from data and is able to learn from datasets that are 2–3 orders of magnitude smaller than datasets required by the demonstrated deep learning approaches to date. Instead of involving a typical deep learning training process, the model uses only a simple covariance calculation to capture the data trends. The low requirement for data, along with the avoidance of a typical training process, massively reduces the overhead cost to set up the model, allowing the model to be more easily repurposed for new design tasks.

Because the model never directly observes the parameters governing the unit cell design, it can seamlessly train on unit cells represented in one format (e.g. pixelated/voxelated) and subsequently generalize its predictions to unit cells of other representations (e.g. parametric). This inherent quality makes the model versatile, emphasizing its potential to be effectively applied across a wide spectrum of design scenarios.

Finally, the model can supply analytically computed gradients for use with gradient-based design optimization, along with uncertainty estimates — both highly useful tools for design tasks. For optimization problems where deviation from the solution of the variational method cannot be tolerated, the surrogate model can simply be used for the first several iterations of optimization and then handed-off to a slower, but more exact dispersion evaluation method. Further, the model can work in tandem with existing, highly specialized variational methods for dispersion computations. We hope that by increasing the computational accessibility of

full dispersion relations we will alleviate the need for current wave analysis shortcuts and design simplifications, and enable the design of next-generation, highly multi-functional wave devices.

A current limitation to our surrogate model is that the covariance function treats each eigenvalue band separately, rather than considering information from all bands together to make more informed predictions. Not only does this likely reduce the accuracy-per-regression-point of the model, but it also means that the order in which regression points are acquired by the model is different for each eigenvalue band, which is inconvenient for computations where multiple bands are to be considered simultaneously. As a first order improvement, this could be explored by simply expanding the covariance function to consider the covariance between eigenfrequencies in different bands. However, if this method is taken to the extreme, the covariance matrix that the covariance function interpolates from can quickly become large enough to be a concern for memory requirements.[4] For this reason, it may be important to examine more efficient ways of storing covariance information. For example, it may be reasonable to only consider covariance information between bands that are within N bands of each other. Another alternative could be to characterize the covariance function in a compressed representation as a neural network rather than an explicit interpolation function.

Further, integration of this GPR model with existing physics-based techniques [16–23] has the potential to enable the hyper-efficient computation of dispersion relations. This could be done simply by using these techniques to carry out regression point computations within the GPR model, or even to generate the training sets that are used to inform the covariance function.

Other future work may involve determining more optimal patterns for regression point sampling. Although the current method works well, it is greedy, and more optimal acquisition functions likely exist. For example, one may consider at each iteration selecting the point that most reduces the variance field in some L_p norm, or may even consider setting up an optimization problem to jointly determine the optimal locations of a fixed number of regression points.

These improvements may improve the practical applicability of this surrogate model and may further decrease the computational barrier to accessing full dispersion information, enabling exciting advances in the design of wave devices.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments & code

A.C.O. is funded by DOE, USA award no. DE-SC0021253. B.T.F. is supported by an NSF GRFP Fellowship, USA and a Kortschak Scholarship, USA.

Code generated during this project for computing dispersion relations of pixelated structures using the finite element method, and for using the GPR-dispersion model can be found on github at the first two links below. Analogous codes for computation of dispersion relations in three-dimensions can be found at the third and fourth links below.

- https://github.com/aco8ogren/2D-dispersion
- https://github.com/aco8ogren/2D-dispersion-GPR
- https://github.com/aco8ogren/3D-dispersion-matlab
- https://github.com/aco8ogren/3D-dispersion-GPR

Appendix A. Generation of random designs

A.1. One dimension

We generate random 1-D unit cell designs with material properties that vary smoothly except at interfaces where the material properties are allowed to be discontinuous. To accomplish this, we draw the exponents of the material properties from a Gaussian process (distinct from the Gaussian process used for regression) using a Matérn covariance function modified to have block-diagonal support. The Matérn covariance structure allows us to impose smoothness semiglobally over the unit cell, while the block diagonal support allows for the interfaces of discontinuity in material property values.

[4] For example, if a dataset of dispersion relations has a resolution in reciprocal space of 201 × 201, the covariance matrix is of size 40 401 × 40 401, occupying 13.06 GB of memory when stored in double precision. This means that if the covariance between even two bands at this resolution is considered, the resulting covariance matrix will be 80 802 × 80 802, occupying 52.24 GB of memory.

A.2. Two dimensions

To generate a random design, we draw a random function from the constitutive Gaussian process (described in Section 6.2) at the resolution determined by N_{pix}. We use a mean function of $m(x) = 0.5$ for the constitutive Gaussian process. Any values below 0 or above 1 are trimmed to 0 and 1 respectively. Finally, each pixel's value is rounded to the nearest of N_{mat} equally spaced values between 0 and 1. The material properties of each pixel are then linearly interpolated between the maximum and minimum constitutive property values.

Appendix B. CNN description

Recall that each unit cell is 32×32 pixels, and each pixel has its own elastic modulus, density, and Poisson ratio. Therefore each unit cell can be represented by a constitutive array, A_{const}, of size $32 \times 32 \times 3$. To form the CNN input array, we concatenate A_{const} with two more matrices, γ_x and γ_y, which are matrices of size 32×32 with all elements equal to the x and y wavevector components respectively. The input array, X_{CNN}, has size $32 \times 32 \times 5$. The output array, y_{CNN}, is a vector of length 10 containing the 10 lowest eigenfrequencies corresponding to the given unit cell and wavevector. For model architecture, we borrow from Finol et al. [32]. The model architecture is as follows: an input layer matching the size of the input, followed by two Conv2D ReLU layers with 275 filters of size 2×2, followed by a MaxPooling2D layer of size 2×2, followed by a flattening layer (to transition to fully connected layers), followed by two fully connected ReLU layers each with 500 nodes, finally followed by the output layer of size 10.

Appendix C. Derivation of gradients for optimization of dispersion relations

In order to optimize the dynamic properties of material designs, we need to be able to efficiently compute the gradient of those properties with respect to design variables. To do this, we start with the generalized eigenvalue equation used for dispersion analysis of periodic materials.

$$(K(\gamma) - \mu M)u = 0 \tag{C.1}$$

Note that, due to the hermitian symmetry of K and M, the following is also true (and will be used later).

$$u^*(K(\gamma) - \mu M) = 0 \tag{C.2}$$

where * denotes conjugate transpose.

By taking a gradient with respect to a generic variable θ, we obtain Eq. (C.3). For now, consider θ to be a scalar. To generalize to vector θ (possibly as a design vector or a wavevector), simply repeat these computations for each entry of θ. The product rule is used here.

$$\frac{\partial K}{\partial \theta}u - \frac{\partial \mu}{\partial \theta}Mu - \mu\frac{\partial M}{\partial \theta}u + (K - \mu M)\frac{\partial u}{\partial \theta} = 0 \tag{C.3}$$

There are two things that can be done with this equation. First, if we only want the gradient of the eigenvalue with respect to a design vector, we can left-multiply Eq. (C.3) by u^*. Then, applying Eq. (C.2), we can see that

$$\frac{\partial \mu}{\partial \theta} = u^*(\frac{\partial K}{\partial \theta} - \mu\frac{\partial M}{\partial \theta})u \tag{C.4}$$

Alternatively, if the gradient of the eigenvector u is also desired, Eq. (C.3) can be rearranged to

$$(K - \mu M)\frac{\partial u}{\partial \theta} - \frac{\partial \mu}{\partial \theta}Mu = -(\frac{\partial K}{\partial \theta} - \mu\frac{\partial M}{\partial \theta})u \tag{C.5}$$

If $\theta \in \mathbb{R}$ and $K, M \in \mathbb{R}^n$, then this system has n equations but $2n$ unknowns (or in general if $\theta \in \mathbb{R}^d$, and $K, M \in \mathbb{R}^n$, then this system has nd equations but $n(d+1)$ unknowns). This is because eigenvectors are only determined up to their scale. To provide the extra n equations required to solve, we can use an eigenvector normalization equation. It is standard to normalize eigenvectors by their mass matrix.

$$u^*Mu = 1 \tag{C.6}$$

By differentiating, we arrive at

$$\frac{\partial(u^*)}{\partial \theta}Mu + u^*\frac{\partial M}{\partial \theta}u + u^*M\frac{\partial u}{\partial \theta} = 0$$

$$(\frac{\partial u}{\partial \theta})^*Mu + u^*\frac{\partial M}{\partial \theta}u + u^*M\frac{\partial u}{\partial \theta} = 0$$

$$(\frac{\partial u}{\partial \theta})^*Mu + u^*M\frac{\partial u}{\partial \theta} = -u^*\frac{\partial M}{\partial \theta}u \tag{C.7}$$

If our eigenvector u were purely real, we could say:

$$(\frac{\partial u}{\partial \theta})^TMu = ((\frac{\partial u}{\partial \theta})^TMu)^T = u^TM\frac{\partial u}{\partial \theta}$$

This would make Eq. (C.7) easy to rewrite so that the unknowns on the LHS only receive operations from the left. Then we could solve

$$\begin{bmatrix} K-\mu M & -Mu \\ -u^T M & 0 \end{bmatrix} \begin{Bmatrix} \frac{\partial u}{\partial \theta} \\ \frac{\partial \mu}{\partial \theta} \end{Bmatrix} = \begin{Bmatrix} -\left(\frac{\partial K}{\partial \theta} - \mu \frac{\partial M}{\partial \theta}\right) u \\ \frac{1}{2} u^T \frac{\partial M}{\partial \theta} u \end{Bmatrix} \tag{C.8}$$

However, in this case, our Bloch-Floquet periodic boundary conditions cause our eigenvector u to be complex, so in general

$$(\frac{\partial u}{\partial \theta})^* M u \neq ((\frac{\partial u}{\partial \theta})^* M u)^* = u^* M \frac{\partial u}{\partial \theta} \tag{C.9}$$

and therefore

$$(\frac{\partial u}{\partial \theta})^* M u = (u^* M \frac{\partial u}{\partial \theta})^* \tag{C.10}$$

By substituting Eq. (C.10) into Eq. (C.7), we can arrive at

$$(u^* M \frac{\partial u}{\partial \theta})^* + u^* M \frac{\partial u}{\partial \theta} = -u^* \frac{\partial M}{\partial \theta} u \tag{C.11}$$

In other words,

$$2\,\mathrm{Re}\left[u^* M \frac{\partial u}{\partial \theta}\right] = -u^* \frac{\partial M}{\partial \theta} u \tag{C.12}$$

And there is no constraint on the imaginary component. In other words,

$$\mathrm{Im}\left[u^* M \frac{\partial u}{\partial \theta}\right] = \text{anything} \tag{C.13}$$

To solve this, we can break up the imaginary and real equations, and write them individually. This turns 3 complex equations into 5 real equations. The reason we do not obtain 6 real equations from the 3 complex equations (as one may expect) is that Eq. (C.11) actually only provides one real equation (note that of Eqs. (C.12), (C.13), only Eq. (C.12) tells us something useful). Luckily, because our eigenvectors are complex, they not only need a constraint in magnitude, but also in complex angle. Note that if u is an eigenvector, then $uRe^{i\beta}$ is also an eigenvector for any $R, \beta \in \mathbb{R}$. In other words, the eigenvector can be multiplied by any complex scalar, and still be an eigenvector. For this reason, we not only need to constrain the magnitude of u (as we have done with Eq. (C.6)), we also need to constrain the complex angle of the entries of u. To do this, we can simply enforce that the first entry of u is purely real. This can be written as

$$\mathrm{Im}\left[\frac{\partial u}{\partial \theta}\right] = 0 \tag{C.14}$$

Finally, we convert Eq. (C.12) into a form conducive to being encoded into a matrix by noting that

$$\mathrm{Re}\left[u^* M \frac{\partial u}{\partial \theta}\right] = \mathrm{Re}\left[u^* M\right] \mathrm{Re}\left[\frac{\partial u}{\partial \theta}\right] - \mathrm{Im}\left[u^* M\right] \mathrm{Im}\left[\frac{\partial u}{\partial \theta}\right] \tag{C.15}$$

Finally, we can combine Eqs. (C.5), (C.12), (C.14) to write the following matrix equation.

$$\begin{bmatrix} \mathrm{Re}[K-\mu M] & -\mathrm{Im}[K-\mu M] & \mathrm{Re}[-Mu] & -\mathrm{Im}[-Mu] \\ \mathrm{Im}[K-\mu M] & \mathrm{Re}[K-\mu M] & \mathrm{Im}[-Mu] & \mathrm{Re}[-Mu] \\ \mathrm{Re}[u^* M] & -\mathrm{Im}[u^* M] & 0 & 0 \\ 0 \cdots 0 & 1\,0 \cdots 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} \mathrm{Re}[\frac{\partial u}{\partial \theta}] \\ \mathrm{Im}[\frac{\partial u}{\partial \theta}] \\ \mathrm{Re}[\frac{\partial \mu}{\partial \theta}] \\ \mathrm{Im}[\frac{\partial \mu}{\partial \theta}] \end{Bmatrix} = \begin{Bmatrix} \mathrm{Re}[-\left(\frac{\partial K}{\partial \theta} - \mu \frac{\partial M}{\partial \theta}\right) u] \\ \mathrm{Im}[-\left(\frac{\partial K}{\partial \theta} - \mu \frac{\partial M}{\partial \theta}\right) u] \\ \mathrm{Re}[\frac{1}{2} u^T \frac{\partial M}{\partial \theta} u] \\ 0 \end{Bmatrix} \tag{C.16}$$

Now, let us say we want to optimize the group velocity of a wave. The group velocity is the gradient of the frequency, ω, with respect to the wavevector, γ. As with θ, it is helpful to think of γ as a scalar initially, then generalize to vector γ by repeating the computation for each component of the wavevector. First, we know from Eq. (C.4) that the gradient of the eigenvalue, ω^2, with respect to the wavevector, γ, for a given branch and wavevector can be determined by

$$\frac{\partial \mu}{\partial \gamma} = u^* (\frac{\partial K}{\partial \gamma} - \mu \frac{\partial M}{\partial \gamma}) u \tag{C.17}$$

Note that Eq. (C.17) gives $\frac{\partial \omega^2}{\partial \gamma}$, and does not directly give the group velocity, which is $\frac{\partial \omega}{\partial \gamma}$. To obtain the group velocity, we can use the chain rule. Consider two functions $f(\gamma)$ and $g(\gamma)$, such that $f(\gamma) = g(\gamma)^2$.

$$g(\gamma) = \sqrt{f(\gamma)}$$

$$\frac{\partial g(\gamma)}{\partial \gamma} = \frac{1}{2} f^{-\frac{1}{2}} \frac{\partial f}{\partial \gamma} \tag{C.18}$$

Letting $f(\gamma) = \mu = \omega^2$ (so $g(\gamma) = \omega$), we arrive at

$$\frac{\partial \omega}{\partial \gamma} = \frac{1}{2\omega} \frac{\partial \mu}{\partial \gamma} \tag{C.19}$$

Using Eqs. (C.17), (C.19), we can write an analytical equation to compute the group velocity.

$$\frac{\partial \omega}{\partial \gamma} = \frac{1}{2\omega} u^* (\frac{\partial K}{\partial \gamma} - \mu \frac{\partial M}{\partial \gamma}) u \tag{C.20}$$

Note that this same equation can be used to compute the design sensitivity of the eigenfrequency, ω, (in contrast to Eq. (C.4), which computes the gradient of the eigenvalue, μ).

$$\frac{\partial \omega}{\partial \theta} = \frac{1}{2\omega} u^* (\frac{\partial K}{\partial \theta} - \mu \frac{\partial M}{\partial \theta}) u \tag{C.21}$$

By taking the gradient of Eq. (C.17) with respect to a design vector, we can aim to obtain the design sensitivity that would be used for a gradient-based optimization of group velocity.

First, to simplify notation, define the matrix A as follows.

$$A = \frac{\partial K}{\partial \gamma} - \mu \frac{\partial M}{\partial \gamma} \tag{C.22}$$

Substituting into Eq. (C.20) shows

$$\frac{\partial \omega}{\partial \gamma} = \frac{1}{2\omega} u^* A u \tag{C.23}$$

Next, take the gradient with respect to a design vector, θ.

$$\begin{aligned} \frac{\partial}{\partial \theta}\left[\frac{\partial \omega}{\partial \gamma}\right] = \frac{\partial^2 \omega}{\partial \theta \partial \gamma} = & \frac{\partial}{\partial \theta}\left[\frac{1}{2\omega}\right] u^* A u \\ & + \frac{1}{2\omega} \frac{\partial u^*}{\partial \theta} A u \\ & + \frac{1}{2\omega} u^* \frac{\partial A}{\partial \theta} u \\ & + \frac{1}{2\omega} u^* A \frac{\partial u}{\partial \theta} \end{aligned} \tag{C.24}$$

Next, let us note how to find some of the quantities needed to use Eq. (C.24).

- $\frac{\partial}{\partial \theta}\left[\frac{1}{2\omega}\right] = -\frac{1}{2\omega^2}\frac{\partial \omega}{\partial \theta} = -\frac{1}{2\omega^2}\frac{1}{2\omega} u^* \left(\frac{\partial K}{\partial \theta} - \mu \frac{\partial M}{\partial \theta}\right) u = -\frac{1}{4\omega^3} u^* \left(\frac{\partial K}{\partial \theta} - \mu \frac{\partial M}{\partial \theta}\right) u$
- $\frac{\partial u}{\partial \theta}$ can be found by solving the linear system given by Eq. (C.16).
- $\frac{\partial A}{\partial \theta} = \frac{\partial}{\partial \theta}\left[\frac{\partial K}{\partial \gamma} - \mu \frac{\partial M}{\partial \gamma}\right] = \frac{\partial^2 K}{\partial \theta \partial \gamma} - \frac{\partial \mu}{\partial \theta}\frac{\partial M}{\partial \gamma} - \mu \frac{\partial^2 M}{\partial \theta \partial \gamma}$

As for computing the derivatives of the system matrices K, M, we can recall that they are constructed by applying wavevector-dependent transformation matrices to design-dependent full system matrices K_f, M_f.

$$\begin{aligned} K(\theta, \gamma) &= T^*(\gamma) K_f(\theta) T(\gamma) \\ M(\theta, \gamma) &= T^*(\gamma) M_f(\theta) T(\gamma) \end{aligned} \tag{C.25}$$

$$\begin{aligned} K &= T^* K_f T \\ M &= T^* M_f T \end{aligned} \tag{C.26}$$

$$\begin{aligned} \frac{\partial K}{\partial \gamma} &= \frac{\partial T^*}{\partial \gamma} K_f T + T^* K_f \frac{\partial T}{\partial \gamma} \\ \frac{\partial M}{\partial \gamma} &= \frac{\partial T^*}{\partial \gamma} M_f T + T^* M_f \frac{\partial T}{\partial \gamma} \end{aligned} \tag{C.27}$$

$$\begin{aligned} \frac{\partial K}{\partial \theta} &= T^* \frac{\partial K_f}{\partial \theta} T \\ \frac{\partial M}{\partial \theta} &= T^* \frac{\partial M_f}{\partial \theta} T \end{aligned} \tag{C.28}$$

$$\begin{aligned} \frac{\partial^2 K}{\partial \theta \partial \gamma} &= \frac{\partial T^*}{\partial \gamma} \frac{\partial K_f}{\partial \theta} T + T^* \frac{\partial K_f}{\partial \theta} \frac{\partial T}{\partial \gamma} \\ \frac{\partial^2 M}{\partial \theta \partial \gamma} &= \frac{\partial T^*}{\partial \gamma} \frac{\partial M_f}{\partial \theta} T + T^* \frac{\partial M_f}{\partial \theta} \frac{\partial T}{\partial \gamma} \end{aligned} \tag{C.29}$$

References

[1] M. Miniaci, R.K. Pal, R. Manna, M. Ruzzene, Valley-based splitting of topologically protected helical waves in elastic plates, Phys. Rev. B 100 (2) (2019) 024304, http://dx.doi.org/10.1103/PhysRevB.100.024304, URL https://link.aps.org/doi/10.1103/PhysRevB.100.024304.

[2] Z. Chen, A. Ogren, C. Daraio, L.C. Brinson, C. Rudin, How to see hidden patterns in metamaterials with interpretable machine learning, Extreme Mech. Lett. 57 (2022) 101895, http://dx.doi.org/10.1016/j.eml.2022.101895, URL https://www.sciencedirect.com/science/article/pii/S2352431622001717.

[3] W.J. Zhou, M.N. Ichchou, Wave propagation in mechanical waveguide with curved members using wave finite element solution, Comput. Methods Appl. Mech. Engrg. 199 (33) (2010) 2099–2109, http://dx.doi.org/10.1016/j.cma.2010.03.006, URL https://www.sciencedirect.com/science/article/pii/S0045782510000824.

[4] G. Kim, C.M. Portela, P. Celli, A. Palermo, C. Daraio, Poroelastic microlattices for underwater wave focusing, Extreme Mech. Lett. 49 (2021) 101499, http://dx.doi.org/10.1016/j.eml.2021.101499, URL https://www.sciencedirect.com/science/article/pii/S2352431621001930.

[5] T.-X. Ma, Z.-Y. Li, C. Zhang, Y.-S. Wang, Energy harvesting of Rayleigh surface waves by a phononic crystal Luneburg lens, Int. J. Mech. Sci. 227 (2022) 107435, http://dx.doi.org/10.1016/j.ijmecsci.2022.107435, URL https://www.sciencedirect.com/science/article/pii/S0020740322003393.

[6] R. Yu, H. Wang, W. Chen, C. Zhu, D. Wu, Latticed underwater acoustic Luneburg lens, Appl. Phys. Express 13 (8) (2020) 084003, http://dx.doi.org/10.35848/1882-0786/aba7a7, Publisher: IOP Publishing.

[7] M. Lott, P. Roux, M. Rupin, D. Colquitt, A. Colombi, Negative index metamaterial through multi-wave interactions: numerical proof of the concept of low-frequency Lamb-wave multiplexing, Sci. Rep. 11 (1) (2021) 561, http://dx.doi.org/10.1038/s41598-020-79572-9, URL https://www.nature.com/articles/s41598-020-79572-9, Number: 1 Publisher: Nature Publishing Group.

[8] Y. Lu, A. Srivastava, Variational methods for phononic calculations, Wave Motion 60 (2016) 46–61, http://dx.doi.org/10.1016/j.wavemoti.2015.08.004, URL https://www.sciencedirect.com/science/article/pii/S0165212515001183.

[9] O. Sigmund, J. Søndergaard Jensen, Systematic design of phononic band–gap materials and structures by topology optimization, in: R.T. Bonnecaze, G.J. Rodin (Eds.), Phil. Trans. R. Soc. A 361 (1806) (2003) 1001–1019, http://dx.doi.org/10.1098/rsta.2003.1177, URL https://royalsocietypublishing.org/doi/10.1098/rsta.2003.1177.

[10] G.A. Gazonas, D.S. Weile, R. Wildman, A. Mohan, Genetic algorithm optimization of phononic bandgap structures, Int. J. Solids Struct. 43 (18) (2006) 5851–5866, http://dx.doi.org/10.1016/j.ijsolstr.2005.12.002, URL https://www.sciencedirect.com/science/article/pii/S0020768305006281.

[11] O.R. Bilal, M.I. Hussein, Ultrawide phononic band gap for combined in-plane and out-of-plane waves, Phys. Rev. E 84 (6) (2011) 065701, http://dx.doi.org/10.1103/PhysRevE.84.065701, URL https://link.aps.org/doi/10.1103/PhysRevE.84.065701, Publisher: American Physical Society.

[12] E. Andreassen, H.R. Chang, M. Ruzzene, J.S. Jensen, Optimization of directional elastic energy propagation, J. Sound Vib. 379 (2016) 53–70, http://dx.doi.org/10.1016/j.jsv.2016.03.002, URL https://www.sciencedirect.com/science/article/pii/S0022460X16002315.

[13] Y. Lu, Y. Yang, J.K. Guest, A. Srivastava, 3-D phononic crystals with ultra-wide band gaps, Sci. Rep. 7 (1) (2017) 43407, http://dx.doi.org/10.1038/srep43407, URL https://www.nature.com/articles/srep43407, Number: 1 Publisher: Nature Publishing Group.

[14] S.S. Injeti, P. Celli, K. Bhattacharya, C. Daraio, Tuning acoustic impedance in load-bearing structures, 2021, arXiv:2106.10573 [cond-mat, physics:physics], URL http://arxiv.org/abs/2106.10573, ArXiv:2106.10573.

[15] F. Maurin, C. Claeys, E. Deckers, W. Desmet, Probability that a band-gap extremum is located on the irreducible Brillouin-zone contour for the 17 different plane crystallographic lattices, Int. J. Solids Struct. 135 (2018) 26–36, http://dx.doi.org/10.1016/j.ijsolstr.2017.11.006, URL https://linkinghub.elsevier.com/retrieve/pii/S0020768317305103.

[16] D. Krattiger, M.I. Hussein, Bloch mode synthesis: Ultrafast methodology for elastic band-structure calculations, Phys. Rev. E 90 (6) (2014) 063306, http://dx.doi.org/10.1103/PhysRevE.90.063306, URL https://link.aps.org/doi/10.1103/PhysRevE.90.063306.

[17] D. Krattiger, M.I. Hussein, Generalized Bloch mode synthesis for accelerated calculation of elastic band structures, J. Comput. Phys. 357 (2018) 183–205, http://dx.doi.org/10.1016/j.jcp.2017.12.016, URL https://www.sciencedirect.com/science/article/pii/S0021999117309002.

[18] E.B. Chin, A.A. Mokhtari, A. Srivastava, N. Sukumar, Spectral extended finite element method for band structure calculations in phononic crystals, J. Comput. Phys. 427 (2021) 110066, http://dx.doi.org/10.1016/j.jcp.2020.110066, URL http://arxiv.org/abs/2009.06913, arXiv:2009.06913.

[19] A. Srivastava, S. Nemat-Nasser, Mixed-variational formulation for phononic band-structure calculation of arbitrary unit cells, Mech. Mater. 74 (2014) 67–75, http://dx.doi.org/10.1016/j.mechmat.2014.03.002, URL http://arxiv.org/abs/1411.2996, arXiv:1411.2996.

[20] M.I. Hussein, Reduced Bloch mode expansion for periodic media band structure calculations, Proc. R. Soc. A 465 (2109) (2009) 2825–2848, http://dx.doi.org/10.1098/rspa.2008.0471, URL https://royalsocietypublishing.org/doi/10.1098/rspa.2008.0471, Publisher: Royal Society.

[21] A. Palermo, A. Marzani, A reduced Bloch operator finite element method for fast calculation of elastic complex band structures, Int. J. Solids Struct. 191–192 (2020) 601–613, http://dx.doi.org/10.1016/j.ijsolstr.2019.12.011, URL https://www.sciencedirect.com/science/article/pii/S0020768319304950.

[22] B.R. Mace, D. Duhamel, M.J. Brennan, L. Hinke, Finite element prediction of wave motion in structural waveguides, J. Acoust. Soc. Am. 117 (5) (2005) 2835–2843, http://dx.doi.org/10.1121/1.1887126.

[23] B.R. Mace, E. Manconi, Modelling wave propagation in two-dimensional structures using finite element analysis, J. Sound Vib. 318 (4) (2008) 884–902, http://dx.doi.org/10.1016/j.jsv.2008.04.039, URL https://www.sciencedirect.com/science/article/pii/S0022460X0800391X.

[24] J. Knap, N.R. Barton, R.D. Hornung, A. Arsenlis, R. Becker, D.R. Jefferson, Adaptive sampling in hierarchical simulation, Internat. J. Numer. Methods Engrg. 76 (4) (2008) 572–600, http://dx.doi.org/10.1002/nme.2339, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.2339, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.2339.

[25] K.W. Leiter, B.C. Barnes, R. Becker, J. Knap, Accelerated scale-bridging through adaptive surrogate model evaluation, J. Comput. Sci. 27 (2018) 91–106, http://dx.doi.org/10.1016/j.jocs.2018.04.010, URL https://www.sciencedirect.com/science/article/pii/S1877750317313807.

[26] B. Liu, N. Kovachki, Z. Li, K. Azizzadenesheli, A. Anandkumar, A.M. Stuart, K. Bhattacharya, A learning-based multiscale method and its application to inelastic impact problems, J. Mech. Phys. Solids 158 (2022) 104668, http://dx.doi.org/10.1016/j.jmps.2021.104668, URL https://www.sciencedirect.com/science/article/pii/S0022509621002982.

[27] Y.S. Teh, S. Ghosh, K. Bhattacharya, Machine-learned prediction of the electronic fields in a crystal, Mech. Mater. 163 (2021) 104070, http://dx.doi.org/10.1016/j.mechmat.2021.104070, URL https://www.sciencedirect.com/science/article/pii/S0167663621002921.

[28] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707, http://dx.doi.org/10.1016/j.jcp.2018.10.045, URL https://linkinghub.elsevier.com/retrieve/pii/S0021999118307125.

[29] S. Lee, Z. Zhang, G.X. Gu, Deep learning accelerated design of mechanically efficient architected materials, ACS Appl. Mater. Interfaces 15 (18) (2023) 22543–22552, http://dx.doi.org/10.1021/acsami.3c02746, Publisher: American Chemical Society.

[30] R. Maulik, H. Sharma, S. Patel, B. Lusch, E. Jennings, A turbulent eddy-viscosity surrogate modeling framework for Reynolds-averaged Navier-Stokes simulations, Comput. & Fluids 227 (2021) 104777, http://dx.doi.org/10.1016/j.compfluid.2020.104777, URL https://www.sciencedirect.com/science/article/pii/S0045793020303479.

[31] X. Li, Y. Zhang, H. Zhao, C. Burkhart, L.C. Brinson, W. Chen, A transfer learning approach for microstructure reconstruction and structure-property predictions, Sci. Rep. 8 (1) (2018) 13461, http://dx.doi.org/10.1038/s41598-018-31571-7, URL https://www.nature.com/articles/s41598-018-31571-7, Number: 1 Publisher: Nature Publishing Group.

[32] D. Finol, Y. Lu, V. Mahadevan, A. Srivastava, Deep convolutional neural networks for eigenvalue problems in mechanics, Internat. J. Numer. Methods Engrg. 118 (5) (2019) 258–275, http://dx.doi.org/10.1002/nme.6012, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.6012.

[33] K. Bhattacharya, B. Hosseini, N.B. Kovachki, A.M. Stuart, Model reduction and neural networks for parametric PDEs, 2020, arXiv:2005.03180 [cs, math, stat], URL http://arxiv.org/abs/2005.03180, ArXiv:2005.03180.

[34] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Neural Operator: Graph kernel network for partial differential equations, 2020, arXiv:2003.03485 [cs, math, stat], URL http://arxiv.org/abs/2003.03485, ArXiv:2003.03485.

[35] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, 2021, arXiv:2010.08895 [cs, math], URL http://arxiv.org/abs/2010.08895, ArXiv:2010.08895.

[36] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning, in: Adaptive computation and machine learning, MIT Press, Cambridge, Mass, 2006, oCLC: ocm61285753.