MAGPIE: a web-based, open-source framework for plate vibration analysis

Matthew Hamilton[1]
DIN, University of Bologna
Viale del Risorgimento 2, 40136, Bologna, Italy

Michele Ducceschi[2]
DIN, University of Bologna
Viale del Risorgimento 2, 40136, Bologna, Italy

Alexis Mousseau[3]
Laboratoire d'Acoustique de l'Université du Mans (LAUM), UMR 6613, Institut d'Acoustique - Graduate School (IA-GS), CNRS, Le Mans Université
Av. Olivier Messiaen, 72085 Le Mans, France

Sebastian Duran[4]
DIN, University of Bologna
Viale del Risorgimento 2, 40136, Bologna, Italy

ABSTRACT

The analysis and simulation of plate vibration are central to many branches of science and engineering, specifically acoustics and musical acoustics. Well-developed plate theories and numerous simulation software packages exist, but they are seldom available in open-source form and are not easily accessible to a greater audience. MAGPIE is a framework for interacting with a novel finite difference scheme for plates under general elastic boundary conditions. Free, clamped, and simply supported boundary conditions are obtainable by setting stiffness coefficients to very small or large values. Boundary conditions are parameterised, which allows for intermediate possibilities. A web browser-based implementation of the framework includes interactive elements. Users begin by providing parameters for an isotropic plate. These parameters are then used to visualise the mode shapes from the finite difference scheme simulation. Elastic boundary conditions can be changed continuously to see their effect on mode shapes. Results are then available for export in common image and data formats. MAGPIE aims to make it easier to explain concepts such as mode shapes to a wider audience and also provide a tool for research that is accessible to students, scientists and musical instrument builders.

1. INTRODUCTION

Plate vibration is a longstanding topic of research. Plates are common engineering elements finding application in industry and, particularly, in acoustics. Musical instrument soundboards are accurately described using appropriate plate theories such as Kirchhoff-Love [1] and Mindlin-Reissner [2]. Furthermore, plates lend themselves naturally to understanding vibration through eigenshape visualisation. They are often used as teaching tools in classrooms for their ability

[1]matthew.hamilton2@unibo.it

[2]michele.ducceschi@unibo.it

[3]alexis.mousseau.etu@univ-lemans.fr

[4]sebastian.duran2@unibo.it

to produce accurate and fascinating Chladni patterns [3]. Eigenshape visualisation techniques are still used today to measure the elastic properties of tonewood [4, 5] parallel to numerical simulation. Despite such an ever-growing body of applications, plate simulation is still mostly accessible through dedicated, proprietary software, often out of reach for the typical student's budget or even for small-sized research groups. This work illustrates the theoretical foundation and a first implementation of MAGPIE. MAGPIE is designed to be an open-source platform to simulate the physics of plates, offering easy-to-use visualisation tools and several analysis techniques. The tools can be equally used as a teaching aid for lecturers and as an analysis tool for researchers and industry stakeholders. Functionalities include setting elastic boundary conditions along the plate edges, visualising eigenshapes, exporting eigenshapes and frequencies, performing a time-domain simulation of the plate equation for sound synthesis, and estimating the elastic modulus of experimental plates. For brevity, not all functionalities can be described in this work, and attention is here restricted to the eigenvalue problem and related implementation. In Section 2, the background plate physics is described, along with the numerical approach used to perform the simulations; in Section 3, the implementation details of MAGPIE are discussed; finally, Section 4 draws some final reflections and illustrates future alleys of development.

2. THEORETICAL BACKGROUND

The Kirchhoff-Love theory of plates describes the dynamics of thin sheet materials under bending. Assuming constant thickness and density, and with the further hypothesis of isotropy, a linear partial differential equation describing wave propagation is, thus [6, 7]:

$$\rho\zeta\frac{\partial^2 u}{\partial t^2} = -D\Delta\Delta u - 2\rho\zeta\sigma\frac{\partial u}{\partial t} + \delta(x-x_p)\delta(y-y_p)f(t). \tag{1}$$

In the above, $u = u(x,y,t) \in \mathscr{D} \times \mathbb{R}_0^+ \to \mathbb{R}$ is the flexural displacement of the plate, occupying a rectangular domain $\mathscr{D} := [0, L_x] \times [0, L_y]$, where L_x, L_y are the side lengths along the x and y axes, respectively. In the above, constants appear as: ρ, the volume density of the plate (in $\mathrm{kg}\cdot\mathrm{m}^{-3}$); ζ, the plate thickness, assumed uniform throughout $\mathscr{D}$ (in m); $D := E\zeta^3/12(1-\nu^2)$, the rigidity of the plate, where E is Young's modulus (in Pa), and ν is Poisson's ratio. In the equation, Δ represents the two-dimensional Laplace operator:

$$\Delta := \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}, \tag{2}$$

$\Delta\Delta$ is, thus, the two-dimensional biharmonic operator. σ, measured in s^{-1}, is a loss factor. $f(t)$ is an external input, and (x_p, y_p) is the input location on the plate surface. The input is distributed according to a double Dirac delta and is, hence, pointwise.

Equation 1 may be arrived at after inspection of the energy of a bent plate. To that end, consider the definitions of inner product and associated norm for two smooth functions defined over $\mathscr{D}$:

$$\langle f, g\rangle := \int_{\mathscr{D}} f\, g \,\mathrm{d}x\mathrm{d}y, \quad \|f\| = \sqrt{\langle f, f\rangle}. \tag{3}$$

Furthermore, consider the following bilinear operator:

$$\mathscr{L}(f,g) := \frac{\partial^2 f}{\partial x^2}\frac{\partial^2 g}{\partial y^2} + \frac{\partial^2 f}{\partial y^2}\frac{\partial^2 g}{\partial x^2} - 2\frac{\partial^2 f}{\partial x \partial y}\frac{\partial^2 g}{\partial x \partial y}. \tag{4}$$

With this notation, the form of the energy can be expressed compactly as:

$$\mathfrak{H} := \frac{\rho}{2}\left\|\frac{\partial u}{\partial t}\right\|^2 + \frac{D}{2}\left(\|\Delta u\|^2 + (\nu-1)\,\langle \mathscr{L}(u,u), 1\rangle\right), \tag{5}$$

and the energy balance reads:

$$\frac{d\mathfrak{H}}{dt} = -2\rho\zeta\sigma \left\| \frac{\partial u}{\partial t} \right\|^2 + \frac{\partial u}{\partial t}(x_p, y_p, t) f(t). \tag{6}$$

In particular, energy is conserved in the absence of dissipation ($\sigma = 0$) and external forcing ($f = 0$). The form of the energy is more complicated than what one would expect after a cursory examination of Equation 1. In fact, one may have trouble bounding the energy from below by just glancing at the form of the energy, as the inner product involving the bilinear operator $\mathscr{L}$ is not, by definition, non-negative. Bilbao [7, Chapter 11] noted that the inner product involving the operator $\mathscr{L}$ in the energy expression only contributes to boundary terms, and it does not change the energy in the domain interior, as one may show after integrating by parts the inner product in the energy expression. See also [8]. The boundary terms appear as the energy-conjugate products force/velocity and moment/angular velocity. For a side perpendicular to the x axis, the analytic expressions for the force and moment are derived as:

$$\mathfrak{F}_x := D\left(\frac{\partial^3 u}{\partial x^3} + (2-\nu)\frac{\partial^3 u}{\partial x \partial y^2}\right), \quad \mathfrak{M}_x := -D\left(\frac{\partial^2 u}{\partial x^2} + \nu\frac{\partial^2 u}{\partial y^2}\right). \tag{7}$$

The boundary terms along the edges located at $x = 0, L_x$ are, thus:

$$\mathfrak{B}_{x=L_x} = \int_0^{L_y} \left(\frac{\partial u}{\partial t}\mathfrak{F}_x + \frac{\partial^2 u}{\partial t \partial x}\mathfrak{M}_x\right) \mathrm{d}y \Big|_{x=L_x}, \quad \mathfrak{B}_{x=0} = -\int_0^{L_y} \left(\frac{\partial u}{\partial t}\mathfrak{F}_x + \frac{\partial^2 u}{\partial t \partial x}\mathfrak{M}_x\right) \mathrm{d}y \Big|_{x=0}. \tag{8}$$

The boundary terms for the edges at $y = 0, L_y$ are obtained using analogous expressions for the force and moment $\mathfrak{F}_y$, $\mathfrak{M}_y$: one only needs to swap x and y in the expressions for $\mathfrak{F}_x$, $\mathfrak{M}_x$. With these in mind, one can easily recover the expressions for free, clamped and simply supported edge conditions: in the free case, the force and the moment vanish; in the clamped case, the displacement and the slope vanish; in the simply supported case, the displacement and the moment vanish. For all such cases, the boundary terms vanish, so the total plate energy follows the energy balance above. One further condition arises at the corner of two free edges, namely:

$$\frac{\partial^2 u}{\partial x \partial y} = 0, \tag{9}$$

see, for example, the book by Szilard for an explanation [9].

2.1. Elastically restrained plates

The boundary conditions of the classic type form a set of consistent conditions leading to the energy balance expressed in Equation 6. In some cases, the edges can themselves store energy, such as when the plate is elastically restrained. This case has received some attention in previous literature, particularly in Galerkin-type applications; see, for example, [10–12]. One reason for such interest lies in the ability to recover the classic edge conditions for limiting values of the edge stiffness constants. When a plate is elastically restrained, two spring-like devices control the displacement and the rotation of the edge by delivering counteracting elastic force and moment. At the simulation level, one can observe the passage from one end condition to another by varying the stiffness of one or more springs along the edges. This leads to modal crossings, including distortion of the modal shapes, which may be a useful analysis tool in various scenarios, particularly in modal-based vibration analysis. The boundary conditions of an elastically restrained plate along $x = L_x$ and $x = 0$ are as follows:

$$K_{xL_x} u(L_x, y, t) = \mathfrak{F}_x(L_x, y, t), \quad R_{xL_x}\frac{\partial u}{\partial x}(L_x, y, t) = \mathfrak{M}_x(L_x, y, t); \tag{10a}$$

$$K_{x0} u(0, y, t) = -\mathfrak{F}_x(0, y, t), \quad R_{x0}\frac{\partial u}{\partial x}(0, y, t) = -\mathfrak{M}_x(0, y, t). \tag{10b}$$

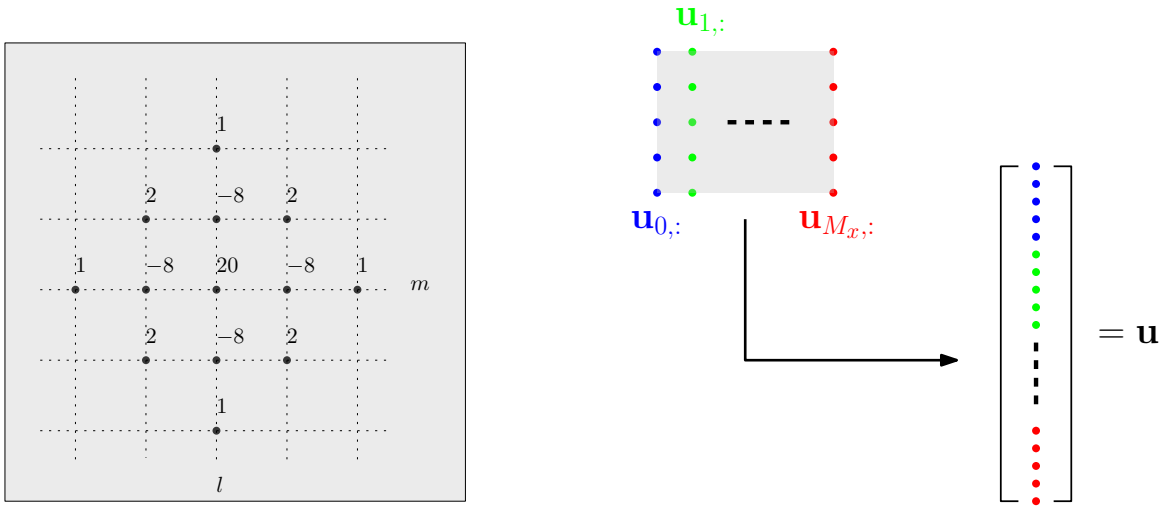


Figure 1: Sketch of the finite difference method discretising the plate equation. On the left, the 13-point biharmonic coefficients are scaled by h^{-4} [13].

Here, $K_\circ$, $R_\circ$ denote the flexural and rotational spring stiffness constants, respectively. Remark that the units of $K_\circ$ are $\mathrm{N}\cdot\mathrm{m}^{-2}$, as the boundary force $F_\circ$ is per unit length. Similarly, $R_\circ$ is measured in $\mathrm{N}\cdot\mathrm{m}^{-1}$. The expressions for the boundary conditions at the edges along $y = 0, L_y$ can be given analogously. Note that the boundary force vanishes whenever $K_\circ$ vanishes. Conversely, the displacement vanishes as $K_\circ \to \infty$. The moment vanishes whenever $R_\circ$ vanishes and the slope vanishes whenever $R_\circ \to \infty$. Note that the presence of the elastic edge supports modifies the expression of the energy. Thus, Equation 5 is not the overall energy of the bent plate. The complete expression is here omitted for brevity but is available in [9], depending on eight elastic constants (i.e. four flexural spring constants $K_\circ$ and four rotational spring constants $R_\circ$).

2.2. Semi-discrete equations

The method of finite differences can be employed to simulate Equation 1. To that end, space is first discredited using a grid of points with equal spacing along x and y. Let h be such grid spacing. Thus, the domain $\mathscr{D}$ is sampled with a two-dimensional grid of size $M := (M_x + 1)(M_y + 1)$, with M_x being the number of grid subintervals along x, and M_y being the number of grid subintervals along y. The continuous displacement $u(x, y, t)$ is thus approximated by a two-dimensional grid function $u_{l,m}$, which may itself be turned into a column vector $\mathbf{u}(t)$ of length M. Turning the two-dimensional grid function into a column vector can be achieved as in Figure 1, by stacking consecutive strips of grid points sharing the same index l. Approximating derivatives is easily accomplished via the following definitions of the difference operators:

$$\delta_{x\cdot} u_{l,m} := (2h)^{-1}\left(u_{l+1,m} - u_{l-1,m}\right) \approx \partial u/\partial x, \tag{11a}$$

$$\delta_{xx} u_{l,m} := h^{-2}\left(u_{l+1,m} - 2u_{l,m} + u_{l-1,m}\right) \approx \partial^2 u/\partial x^2, \tag{11b}$$

$$\delta_{xxxx} u_{l,m} := \delta_{xx}(\delta_{xx} u_{l,m}) \approx \partial^4 u/\partial x^4, \tag{11c}$$

with similar definitions holding for the derivatives along y. Thus, a discretisation of the biharmonic operator is obtained as:

$$\Delta\Delta \approx \delta_{xxxx} + \delta_{yyyy} + 2\delta_{xx}\delta_{yy}. \tag{12}$$

When applied to a grid point away from the boundary, the resulting 13-point biharmonic difference operator has coefficients as per Figure 1. Closer to the boundary, the coefficients must be specialised to account for the boundary conditions. A discretisation of, e.g. Equation 10b is

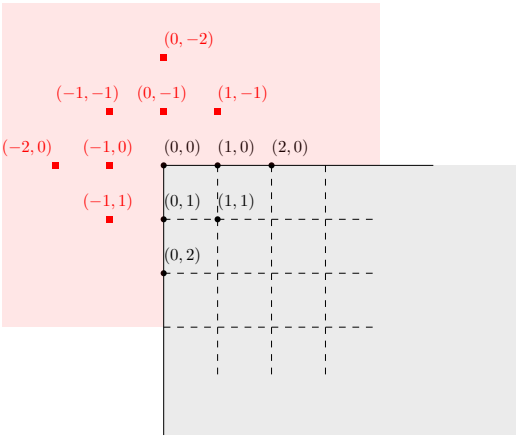


Figure 2: Ghost points resulting from applying the biharmonic difference operator to the upper-left corner.

given here as:

$$K_{x0}u_{0,m} = -D(\delta_{xx} + (2-\nu)\delta_{yy})\delta_{x\cdot}u_{0,m}, \quad R_{x0}\delta_{x\cdot}u_{0,m} = D(\delta_{xx} + \nu\delta_{yy})u_{0,m}. \tag{13}$$

Analogous conditions can be given at the other edges. The corner condition Equation 9 may be discredited at the upper-left corner as:

$$\delta_{x\cdot}\delta_{y\cdot}u_{0,0} = 0, \tag{14}$$

and similarly at the other three corners. Figure 2 shows the seven "ghost" points resulting from applying the biharmonic operator to $u_{0,0}$. Such seven values are set by applying the moment boundary condition to $u_{0,1}$ and $u_{1,0}$, plus the moment and force conditions along x and y to $u_{0,0}$. These are six conditions in total. The seventh condition is given by Equation 14. The resulting biharmonic coefficients have extremely lengthy expressions, omitted here for brevity, but presented on the GitHub page of MAGPIE [14]. However, the resulting biharmonic matrix is sparse, with a sparsity pattern as per Figure 2.

2.3. Eigenvalue problem and time stepping scheme

Let $\mathbf{B}$ denote the resulting biharmonic matrix of size $M \times M$. $\mathbf{B}$ is generally not symmetric, but it possesses a set of M independent, real eigenvectors for sufficiently small h. The eigenvalues are also real and bounded from below. For a completely free plate, the eigenvalues are found in the range

$$0 \lessapprox \lambda_{\mathbf{B}} \leq 64h^{-4}, \tag{15}$$

where the approximate left inequality is due to the rigid-body eigenvalues becoming negative in a very small neighbourhood around zero. For elastically restrained plates, the eigenvalues are bounded according to:

$$0 \lessapprox \lambda_{\mathbf{B}} \leq \mathcal{O}(\max(R_\circ, K_\circ)), \tag{16}$$

where the "big-Oh" notation was used. Thus, when the plate is restrained using very stiff springs, the spectral radius of the matrix becomes large. Whilst most of the eigenvalues will still be found within the bounds in Equation 15, a handful of "spurious" eigenvalues and associated modal shapes may result from applying the elastic boundary conditions using very large stiffness constants. In any case, the matrix $\mathbf{B}$ is factorised according to:

$$\mathbf{B} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1}, \tag{17}$$

where $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues, and $\mathbf{Q}$ is a matrix of associated column eigenvectors. The plate eigenfrequencies, in radians per second, are obtained as:

$$\mathbf{\Omega} = \sqrt{\frac{D}{\rho\zeta}}\sqrt{\mathbf{\Lambda}}, \tag{18}$$

where taking the square root of the eigenvalues is justified according to the bounds above. A semi-discrete version of Equation 1 results as:

$$\rho\zeta\ddot{\mathbf{u}}(t) = -D\mathbf{B}\mathbf{u}(t) - 2\rho\zeta\sigma\dot{\mathbf{u}}(t) + h^{-2}\mathbf{J}f(t). \tag{19}$$

Note that overdots now denote total time derivatives. In the above, $\mathbf{J}$ is a sparse vector discretising the double Dirac delta. Many options are available, such as via Lagrange interpolants, see [7]. Using the factorisation Equation 17, diving both sides of Equation 19 by $\rho\zeta$ and multiplying both sides by $\mathbf{Q}^{-1}$, the following system results:

$$\ddot{\mathbf{w}}(t) = -\mathbf{\Omega}^2\mathbf{w}(t) - 2\mathbf{C}\dot{\mathbf{w}}(t) + \rho^{-1}\zeta^{-1}h^{-2}\mathbf{Q}^{-1}\mathbf{J}f(t), \tag{20}$$

where $\mathbf{w}(t) := \mathbf{Q}^{-1}\mathbf{u}(t)$. This is a completely parallel (i.e. diagonal) system of M lossy, forced oscillators, with natural frequency Ω_m, $1 \leq m \leq M$. Note that the loss matrix $\mathbf{C}$ can now contain different loss factors σ_m for different modes, such as from experimental measurements.

One can truncate the system to an integer $N < M$, and work with the modes belonging to a given frequency band. One may, for instance, solve the eigenvalue problem Equation 19 using a very fine grid for increased precision but only retain the modes for which the natural frequency falls within the audible range. System 20 may then be updated in time using the "exact" integrator for the lossy oscillator, see e.g. [15] thus, yielding numerical simulations with very low numerical dispersion.

3. IMPLEMENTATION

MAGPIE collects together functionality for computing a discrete biharmonic operator, eigensolver, and impulse response generation through modal time integration and Young's modulus estimation. As discussed previously, the full extent of the functionality cannot be fully covered in this paper. This section aims to cover the main development steps and choices taken during the development of MAGPIE, technical underpinnings, justification and a case study of a hypothetical library application.

MAGPIE was originally planned as a library for Python, as it is a free open-source platform. During the development, it became apparent that MAGPIE would have greater reach as a teaching aid and analysis tool if implemented across multiple programming languages. Given its prevalence within academia and industry, MATLAB was the first choice; therefore, it is more likely to reach the intended audience in acoustics education. Jupyter Notebooks are currently used as the graphical interface for the Python implementation of MAGPIE. Jupyter Notebooks were chosen for their graphical interface library, which allows easy deployment in the web browser. Users can change the plate parameters by typing the values directly or using sliders for the boundary conditions. A graphical result of the plate's mode shapes automatically responds to parameter changes. This allows continuous exploration of the parameter space and its effects on mode shapes and frequency response.

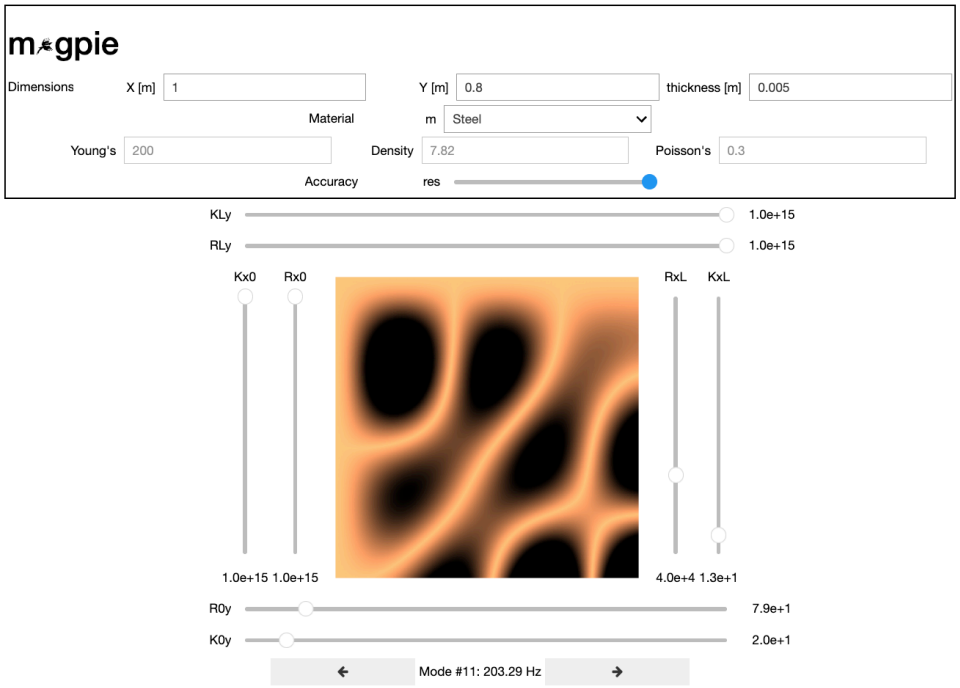


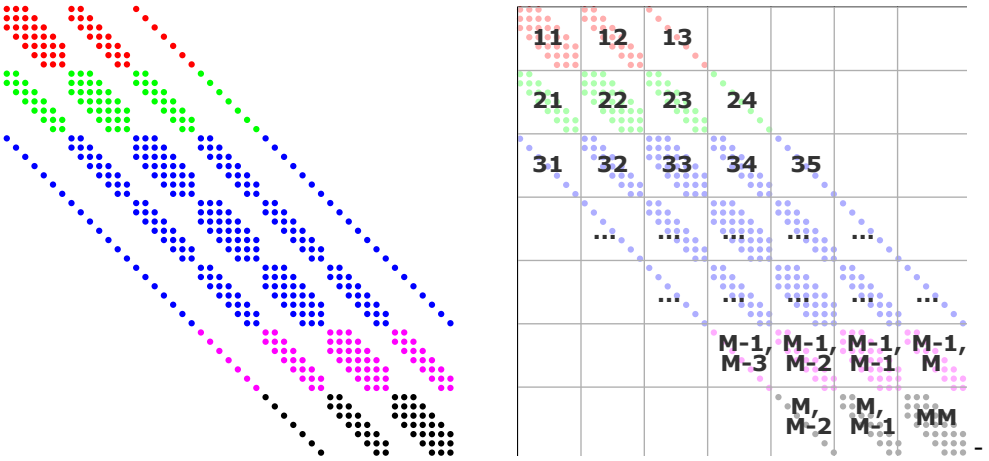
Figure 3: Beta version user interface for MAGPIE in Jupyter Notebooks

In addition, this format of interacting with the library means that only affected coefficients need to be recalculated and is, therefore, more intuitive than a script structure. Users can alter the number of grid points within the biharmonic to trade-off between responsiveness and accuracy. First, users can explore the parameter space with fewer grid points for quicker computation. When an area of interest is found, the effective resolution can be increased for higher-quality visualisations. Lastly, Jupyter Notebooks allows for the potential to provide MAGPIE as a service on a public server. This would remove the need for installation of additional libraries and dependencies, lowering the barrier of entry to MAGPIE.

3.1. Computing the biharmonic operator

One of the first steps for MAGPIE is creating a discrete biharmonic operator. Repeated calculation of the biharmonic operator is central to some of the functionality throughout the rest of MAGPIE. Therefore, optimal construction of the operator was a key factor during development. The biharmonic operator sparsity pattern, described in Figure 2, can be decomposed into several methods, of which two will be described. It is worth noting that explicitly declaring the set of coefficients in a loop formulation, though a valid method of using the biharmonic operator, was not considered an option for the current implementations as it does not output a matrix. It is advantageous to declare the biharmonic operator as a matrix as this is the input format used by the eigensolvers in both MATLAB and Python's SciPy package utilised in other parts of MAGPIE. Conveniently, the biharmonic operator is multi-diagonal and lends itself to the sparse matrix formats of MATLAB and SciPy.

In the first approach, the operator can be composed of sparse sub-blocks calculated separately and then concatenated. Figure 4 shows the internal pattern repetition of blocks and how each block can be indexed. After each block is computed separately, they can all be concatenated together. Colour illustrates boundary sections of the biharmonic, with the central repeating section highlighted in blue. This approach simplifies code structure by avoiding the need for zero padding of the computed matrices but has implications for performance.



(a) Block Structure (b) Block Indexing

Figure 4: Biharmonic block structure. Each indexed block is computed separately and then concatenated together. Colour illustrates boundary sections of the biharmonic, with the central repeating section highlighted in blue.

The second approach is to decompose the operator across the 13 diagonals and to separate them into non-repeating sectors. Each diagonal array is split into 3-5 sectors; each diagonal sector contains between 3-5 unique coefficients. Care must be taken to zero pad diagonals where necessary. The two boundary conditions (flexural and rotational spring stiffness) per edge of the plate mean 8 boundary stiffness coefficients (BSP), which impacts the repetition of diagonal patterns in the distribution of coefficients. Figure 5 illustrates the distribution of unique coefficients within the operator. As in Figure 4, Figure 5 shows a repeating central section but an overall lack of repeating diagonal patterns.

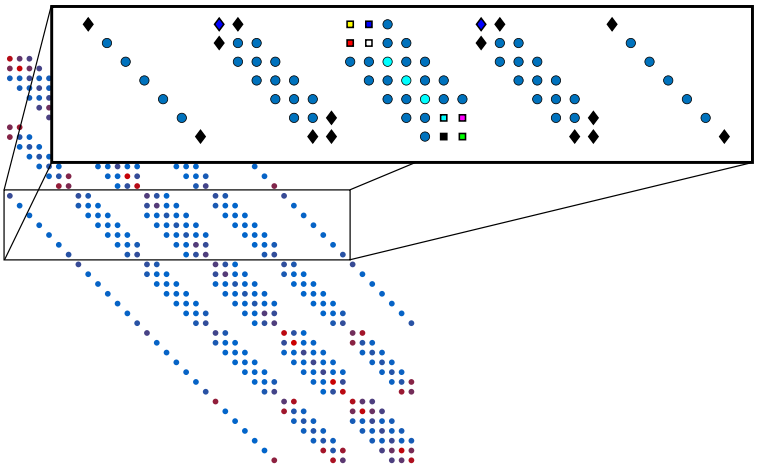


Figure 5: Detail of biharmonic block structures illustrating the impact of 8 BSP on internal symmetry. Each marker/colour combination represents a unique coefficient.

The 13-point biharmonic stencil, as shown in Figure 2, only has 4 unique coefficients. In addition, two extra coefficients are required for the case of clamped and simply supported boundary conditions, implying that the biharmonic operator in those classical cases is composed of 6 unique coefficients. By contrast, the biharmonic operator with generalised elastic boundary conditions can contain up to 229 unique coefficients. As a result, each block or diagonal sector cannot be constructed through the repetition of a stencil pattern but must be declared explicitly.

This lack of pattern repetition also impacts computational performance when constructing the operator. Since the biharmonic operator with generalised boundary condition provides very useful functionality in and of itself, the functionality to compute it is provided with an open interface as part of the library through the `bhmat` function [14]. The operator can then be used in further user-defined Finite Difference Time Domain schemes or if the users wish to take a more bespoke approach to eigenvalue/eigenvector calculation.

3.2. Performance Across Implementations

It was initially unclear which approach to composing the operator would yield the best performance. At the time of writing, the diagonal composition method was not yet finished with the Python implementation.

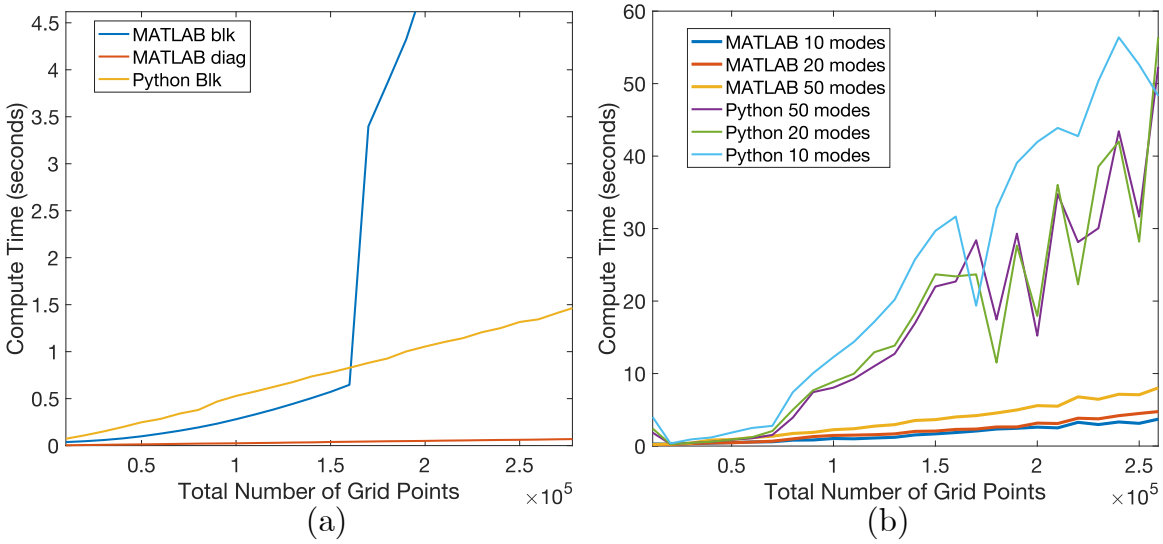


Figure 6: (a) Biharmonic operator computation performance comparison between diagonal formulation in MATLAB and block formulation in both MATLAB and python, (b) Eigensolver performance comparison between MATLAB and python for three different number of modes.

Figure 6 (a) compares the computing time necessary to build the biharmonic with increasing size, using a block structure, diagonal in MATLAB, and a block structure in Python. Each size of biharmonic was computed 10 times and the shortest time was recorded. The speed tests for the biharmonic operator computation in Figure 6 show a clear performance superiority of the diagonal formulation over the block formulation in both MATLAB and Python languages. For the largest grid size ($3e^5$) tested in MATLAB, the diagonal method was forty times faster than the block method (results showing 0.2 seconds against 8 seconds respectively). Given this result, it can be projected that a diagonal composition in Python will likely outperform the block composition method.

Figure 6 (b) shows the MATLAB eigensolver outperforming the SciPy package's `eigs` function. It is possible that performance may improve when using a diagonal spare compression format. A non-intuitive result is that fewer modes can compute slower than a higher one. This is due to the search space being proportional to the number of eigenvalues requested, which may lead to eigenvalues being found faster for a higher number of modes. Interestingly enough, there is a similar phenomenon in MATLAB, and it will be of interest when optimizing the speed of MAGPIE in later stages of development. Performance benchmarking of alternative eigensolvers in Python is still ongoing.

3.3. Case Study: Generalised Boundary Conditions

This section presents a use-case of the MAGPIE library where a user wishes to explore the transition from a simply supported condition case to a clamped case by gradually increasing rotational spring stiffness across one edge of a plate and observing the effects on modal patterns and frequency response. The user first defines the plate material parameters, which are Young's modulus, density, Poisson number, and dimensions. These parameters compute the discrete biharmonic operator in a sparse matrix format. The biharmonic operator is then used as input for an eigensolver, which returns the eigenvalues and eigenvectors.

The user wishes to compare with Chladni patterns recorded during the course of experiments on a real plate. As such, the user chooses the 'Chladni' format option for plotting mode shapes. The 'Chladni' option renders a top-down 2D visualisation of mode patterns where points at or near zero are highlighted. This allows the user to render Figure 7, which shows the effect of transition in the rotational stiffness. From these plots obtained by MAGPIE, the user can illustrate how the mode shapes develop between simply supported and clamped cases. As expected, an increase in rotational stiffness increases modal frequency.

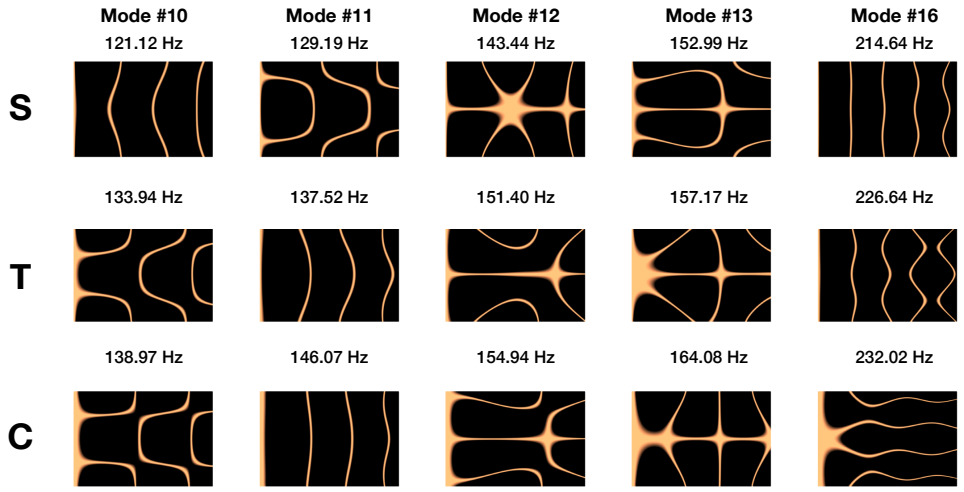


Figure 7: Simulated Chladni patterns of modes 10, 11, 12, 13, and 16, for a plate with free boundary conditions along all edges except the leftmost one. The last condition moves from simply supported (**S**) to a fully clamped cantilever case (**C**), with an intermediary step corresponding to a high rotation stiffness (**T**). The choice of specific modes is the result of an exploration using the Jupyter Notebook graphical interface depicted in Figure 3

Secondly, the eigenvalues returned from the eigensolver are used to calculate modal frequencies. The user can then use the modal time integration functionality of MAGPIE using these modal frequencies, producing a simulated impulse response of the plate. The impulse response of the simulated is then used to create a series of FRF plots contrasting the transition between boundary conditions, as can be created as shown in 8.

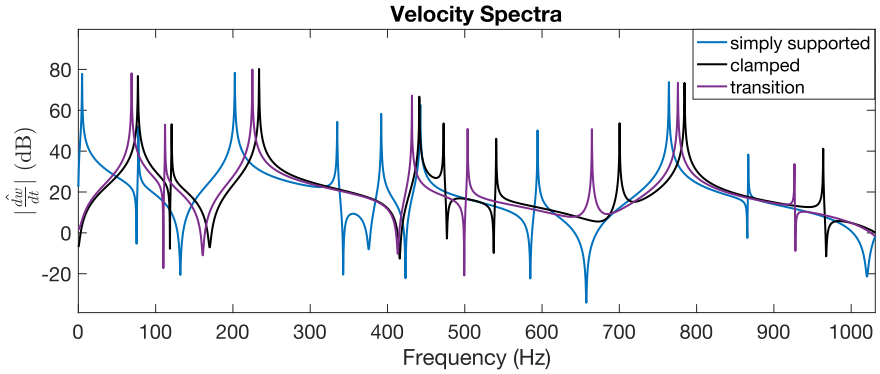


Figure 8: FRF of impulse response from simulations of the same plate under different boundary conditions.

4. FINAL COMMENTS AND CONCLUSIONS

The first version of an open source library for the modal analysis of plates was authored. The full code of the project is available on the Nemus project's GitHub https://github.com/orgs/Nemus-Project/teams/magpie/repositories. Care and attention has been given to the structure of the code, in order to make the library as modular and reusable to as wide an audience as possible. Central functionality was replicated in both MATLAB and Python, while taking care to accommodate for the conventions used by both programming languages. This functionality was namely the computation of the biharmonic operator in a sparse matrix format with generalized elastic boundary conditions, the computation the eigenfrequencies and eigenmodes, the visualisation the mode shapes. All of these data can then be used in time domain simulations. A graphical interface was designed using the Jupyter Notebooks platform by utilising the Python implementation of the library.

Going forward, a broader set of models will be considered. The ongoing work concerns the Föppl–von Kármán model for nonlinear thin plates [16], which calls for the implementation of other operators in sparse matrix form. These operators are used in a variety of other models, such as the nonlinear bar and nonlinear string, which are other models that will be developed in MAGPIE. The orthotropic Kirchoff-Love plate is another direction to explore. The orthotropic model is the most interesting from a musical instrument manufacturing perspective as it would be a better approximation of wooden soundboard vibration. In combination with generalised boundary conditions, an orthotropic model would allow for easier experimental comparison than the tools currently available. In terms of time efficiency, parallel computation of sectors and block have not yet been tested, but may yield better performances. The biharmonic operator computational performance could be further improved by separately considering specific cases such as square plate and symmetric boundary conditions where repeating coefficient patterns are more prevalent. Future development will include the implementation of a MATLAB package, a C / C++ library in addition to the expansion of the existing Jupyter Notebooks interface.

ACKNOWLEDGEMENTS

This work received funding from the European Research Council (ERC) under the Horizon2020 framework, grant number 950084 - StG - NEMUS.

REFERENCES

1. A. E. H. Love. The Small Free Vibrations and Deformation of a Thin Elastic Shell. *Philosophical Transactions of the Royal Society of London Series A*, 179:491–546, January 1888.
2. R. D. Mindlin. Influence of Rotatory Inertia and Shear on Flexural Motions of Isotropic, Elastic

Plates. *Journal of Applied Mechanics*, 18(1):31–38, March 1951.

3. E.F.F. Chladni. *Treatise on Acoustics: The First Comprehensive English Translation of EFF Chladni's Traité d'Acoustique*. Springer, 2015.
4. M. Ducceschi, S. Duran, H. Tahvanainen, and L. Ausiello. A method to estimate the rectangular orthotropic plate elastic constants using least-squares and chladni patterns. *Applied Acoustics*, 220:109949, 2024.
5. N. H Fletcher and T.D. Rossing. *The physics of musical instruments*. Springer Science & Business Media, 2012.
6. K.F. Graff. *Wave Motion in Elastic Solids*. Dover Publications, 1991.
7. S. Bilbao. *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics*. Wiley, October 2009.
8. O. Thomas and S. Bilbao. Geometrically nonlinear flexural vibrations of plates: In-plane boundary conditions and some symmetry properties. *Journal of Sound and Vibration*, 315(3):569–590, 2008.
9. R. Szilard. *Theories and applications of plate analysis*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2004.
10. W.L. Li, X. Zhang, J. Du, and Z. Liu. An exact series solution for the transverse vibration of rectangular plates with general elastic boundary supports. *Journal of sound and vibration*, 321(1-2):254–269, 2009.
11. X. Zhang and W.L. Li. Vibrations of rectangular plates with arbitrary non-uniform elastic edge restraints. *Journal of Sound and Vibration*, 326(1-2):221–234, 2009.
12. Q.S. Li. Free vibration of elastically restrained flexural-shear plates with varying cross-section. *Journal of sound and vibration*, 235(1):63–85, 2000.
13. G. J. Tee. A novel finite-difference approximation to the biharmonic operator. *Comput. J.*, 6(2):177–192, 1963.
14. Magpie jupyter notebook github repository. https://nemus-project.github.io/projects/magpie. Accessed: 2024-04-12.
15. J. L. Cieśliński. On the exact discretization of the classical harmonic oscillator equation. *Journal of Difference Equations and Applications*, 17(11):1673–1694, 2011.
16. T. von Kármán. Festigkeitsprobleme im Maschinenbau. In *Encyklopädie der Mathematis- chen Wissenschaften*. 1910.