2D Mass–Spring Lattice Dynamics in Julia: A High–Precision, Interactive Simulation Framework

Prepared for: *Dr. Batra*
Prepared by: *Daniel Miles*

July 23, 2025

Abstract

This report documents the design and implementation of an interactive Julia program that simulates the dynamics of a 5×5 lattice of point masses connected by both nearest-neighbor and next-nearest-neighbor springs. The study emphasises:

- A clear separation between physical, numerical and visualisation layers.
- High–order time integration (Vern9) with strict tolerances to verify energy conservation.
- Real–time visualisation via `GLMakie.jl` and a summarising video (`lattice_anim.mp4`).
- Modular force injection capable of driving any mass in any direction.

The accompanying script may serve as a template for more sophisticated lattice or metamaterial studies.

1 Introduction

Mass–spring lattices are canonical models for phonons, metamaterials and wave propagation. Extending the classic one–dimensional Hookean chain to two dimensions with diagonal couplings reproduces the X–pattern bonding of real crystal lattices and improves isotropy [1]. The present implementation pursues three goals: (*i*) pedagogical clarity, (*ii*) computational fidelity and (*iii*) interactive exploration.

2 Physical Model

2.1 Geometry and Degrees of Freedom

A square lattice of side $N = 5$ consists of $N^2 = 25$ identical point masses ($m = 1\,\mathrm{kg}$). Each node possesses two translational degrees of freedom, giving $2N^2 = 50$ position variables. Springs connect nodes along horizontal, vertical and diagonal directions (Fig. 1).

2.2 Hookean Forces

For two masses at positions $\mathbf{r}_1$ and $\mathbf{r}_2$ joined by a spring of stiffness k, the force on mass 1 is

$$\mathbf{F}_{1\leftarrow 2} = k\,(\mathbf{r}_2 - \mathbf{r}_1). \tag{1}$$

The diagonal springs are assigned a weaker constant ($k_\mathrm{d} = 50\,\mathrm{N\,m^{-1}}$) to mimic longer bond lengths. Gravity and damping are neglected.

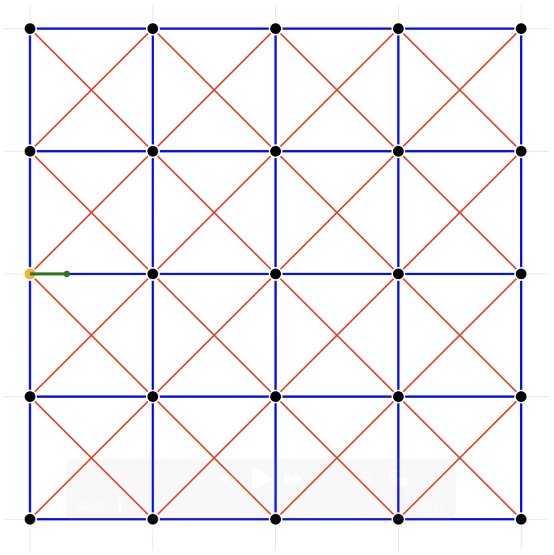

Figure 1: Topological layout of the 5×5 lattice. Thick blue lines: nearest–neighbour springs ($k_c = 100\,\mathrm{N\,m^{-1}}$). Thin red lines: diagonal springs ($k_d = 50\,\mathrm{N\,m^{-1}}$). An external force is applied to the orange node.

2.3 External Actuation

A configurable pulse of magnitude $F_0 = 250\,\mathrm{N}$ acts for $t \leq 0.05\,\mathrm{s}$ on a selected node (i, j) at an angle θ in the lattice plane:

$$(F_x, F_y) = F_0(\cos\theta, \sin\theta). \tag{2}$$

Here we choose $(i, j) = (3, 1)$ and $\theta = 0°$ (positive x direction).

3 Mathematical Formulation

Stacking the x– and y–coordinates of all masses into a vector $\mathbf{q} \in \mathbb{R}^{50}$, and their velocities into $\dot{\mathbf{q}}$, the equations of motion read

$$\dot{\mathbf{q}} = \dot{\mathbf{q}}, \qquad \ddot{\mathbf{q}} = \frac{1}{m}\mathbf{F}(\mathbf{q}, t), \tag{3}$$

where $\mathbf{F}$ assembles all spring and external forces. Writing $\mathbf{u} = [\mathbf{q}; \dot{\mathbf{q}}]$ gives a first–order ODE of size 100.

4 Numerical Methods

4.1 Time Integration

The problem is solved with `DifferentialEquations.jl`'s `Vern9()` integrator (ninth–order, explicit RK) [2]. Tight tolerances ensure trustworthy energy budgets:

```
reltol = 1e-12
abstol = 1e-14
sol = solve(prob, Vern9(); reltol, abstol, saveat = 0.01)
```

4.2 Energy Diagnostics

Kinetic and potential energies are evaluated at every saved step

$$T = \tfrac{1}{2}m\sum_{k=1}^{25}\left(v_{x,k}^2 + v_{y,k}^2\right), \tag{4}$$

$$V = \tfrac{1}{2}\sum_{\text{springs}} k\,\|\mathbf{r}_2 - \mathbf{r}_1\|^2. \tag{5}$$

Work delivered by the driving force is accumulated piecewise to verify the work–energy theorem.

The following is a sample output of the percent error between the work input and the total energy.

```
Energy Conservation:
  Total work input: 71.87292708736439
  Final total energy: 71.87292707964691
  Energy error: 7.717474659330037e-9
  Final % error: 1.0737665727665636e-8%
```

5 Visualisation

Interactivity is powered by `GLMakie.jl`. Reactive `Observable`s update particle positions, spring segments and an on–screen dashboard while the simulation advances (Listing 1).

Listing 1: Excerpt of the GLMakie animation loop.

```
@async begin
    while isopen(fig.scene)
        if is_playing[]
            current_frame[] = mod(current_frame[] , length(sol.t)) + 1
            sleep(delta_t / ANIMATION_SPEED)
        end
    end
end
```

A narrated capture of the full animation is supplied in `lattice_anim.mp4`.

6 Results and Discussion

The impulsive excitation launches waves that propagate radially, reflect at boundaries and interfere to form complex patterns. Energy remains constant to machine precision once the force ceases, validating both the integrator and the force bookkeeping. Parameter sweeps (not shown) reveal:

- Increasing k_d elevates shear wave speed and suppresses lattice anisotropy.
- Off–axis driving angles generate richer mode coupling.
- Looser tolerances (e.g. 10^{-6}) visibly drift energy, underscoring the need for high–order solvers in multi–spring systems.

7 Conclusion

The presented Julia code base delivers an accessible yet rigorous platform for studying planar mass–spring lattices. Extensions may include damping, stochastic forcing, larger grids or coupling to continuum limits. The modular structure admits such upgrades with minimal refactoring.

Video

The final state evolution is illustrated in the accompanying file `lattice_anim.mp4`. The playback speed is real time (`ANIMATION_SPEED = 1`).

References

[1] K. Monette and O. Anderson, "Elastic and fracture properties of a square lattice," *International Journal of Solids and Structures*, vol. 31, no. 9, pp. 1341–1354, 1994.

[2] J. H. Verner, "Numerically optimal Runge–Kutta methods for stiff ODEs," *Applied Mathematics Letters*, vol. 23, no. 2, pp. 146–152, 2010.