Refractive Computation: parallelizing logic gates across driving frequencies in a mechanical polycomputer.

Shawn Beaulieu[1], Piper Welch[1], Atoosa Parsa[1], Corey O'Hern[2], Rebecca Kramer-Bottligio[2], and Josh Bongard[1]
[1]University of Vermont, Burlington, VT 05404
[2]Yale University, New Haven, CT, 06520
Shawn.Beaulieu@uvm.edu

Abstract

Unconventional computing seeks to develop new means of acting on and interpreting the world. These emerge when new tools and computational substrates are built or discovered, or when existing artifacts are deployed in novel ways. Prior work designed sheets of vibrating particles to achieve mechanical polycomputation, wherein multiple logical operations were physically executed by the same parts at the same time. This works by exploiting the vibrational superposition of particles induced by external drives acting at multiple frequencies. In this paper, we introduce an idea called *refractive computation*, in which a sufficiently high density of polycomputed logic gates results in parallelized computations across driving frequencies. Parallelized logic gates are split across external drive frequencies in a single simulation, and emerge in the course of polycomputing sequential logic gates.

Introduction

Motivation and prior work.

Continued advances in machine capability will be underwritten by advances in materials science and hardware design. Material conditions set the terms of intellectual life; while at the same time, as the history of deep learning shows, useful ideas may greatly predate the possibility of their material instantiation (Pasquinelli, 2023). Ongoing research into unconventional substrates for computation aims both to improve extant machines, and push the boundaries of what is possible with artificial constructs (Pask, 1958, 1960; Sinapayen et al., 2017; Zhu et al., 2018; Wu et al., 2019; Nakajima, 2020; Bongard and Levin, 2021; Dillavou et al., 2022; Stern and Murugan, 2023; Jaeger et al., 2023).

Chemical and biological computing in particular have been characterized as intrinsically parallel, and possessed of multi-scale competence (Conrad, 1972, 1985; Gorecki et al., 2015; McMillen and Levin, 2024). Increasingly, however, purely mechanical systems are being shown to exhibit analogous properties when the right operational constraints are applied (Pashine et al., 2019; Serra-Garcia, 2019; Zangeneh-Nejad et al., 2020; Yasuda et al., 2021; Kaspar et al., 2021; El Helou et al., 2022; Jiao et al., 2023; Bongard and Levin, 2023; Mei and Chen, 2023). One such property is *polycomputation*, wherein multiple computations can executed by the same parts at the same time with respect to different factors (Parsa et al., 2023). This paper builds on the prior demonstration of polycomputation by first increasing the computational density of a simulated granular metamaterial (from two sequential logic gates to four) and next by reporting a qualitatively new property of polycomputational assemblies: *refractive computation*.

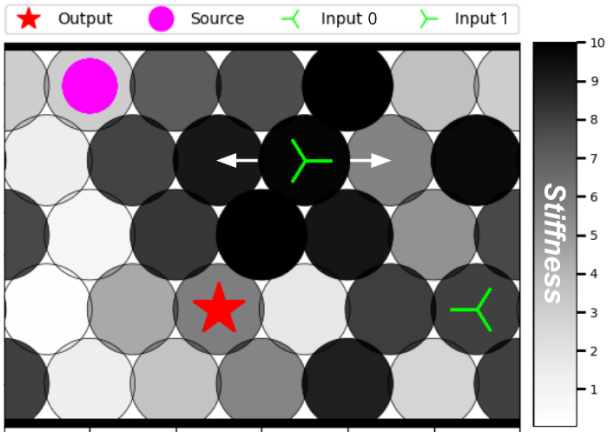

Figure 1: Visualization of an evolved granular assembly with two input particles (green sticks), one output particle (red star), and one source particle (pink circle). Evolution sets the stiffness ratios of all particles, represented by a grey-scale color map, as well as the indices of the input, output, and source particles. In this example, one of the input particles is being vibrated horizontally, while the other is not. This corresponds to an input of 1 and 0.

Utilizing the capacity of polycomputing systems to perform multiple operations simultaneously, we evaluate the set of logically possible inputs to a two-input logic gate all at once, in a single simulation. We show that when four distinct sequential logic gates are evaluated for each logically possible input in a specific order over four simulations, each simulation can be used to evaluate a unique *parallelized* logic gate. Refractive computation is then a byproduct or side-effect (Rosen, 1994) of the operations that realize poly-

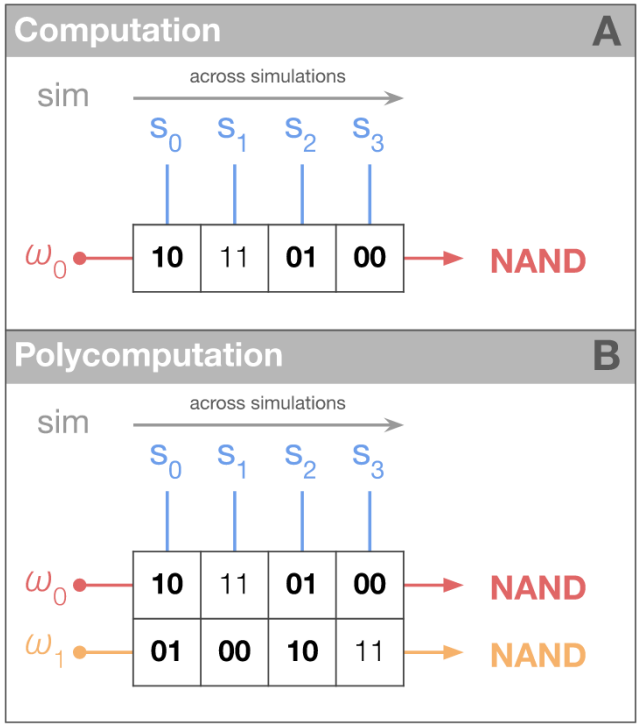

Figure 2: (**A**) For a sequential logic gate (e.g. NAND) computation is defined as the selective amplification of external drives on input particles, following stimulation patterns **s**=[s_0, s_1, s_2, s_3], represented as binary strings, at frequency ω_0. The k^{th} binary value in the j^{th} cell indicates whether the k^{th} input particle is externally driven during simulation s_j. For each pattern, or switch string, an independent simulation is run, and the resulting gain at the output particle is recorded. Sequential computation then occurs *across* simulations, and only instantiates a logic gate if the recorded gains from each simulation agree with the target gains for that gate. Bold values indicate the switch strings for which the target gain is equal to 1, while non-bold values indicate a target gain of 0. (**B**) Polycomputation for the visualized matrix, **S**, is said to occur when the same parts are used to compute two or more functions at the same time. Pictured are two sequential NAND gates, each operating at a different frequency, over four simulations. The k^{th} binary value in cell $\mathbf{S}_{cj}$ indicates whether the k^{th} input particle is externally driven during simulation j at frequency ω_c.

computation. Parallelization may thus help scale mechanical computation to more complex functions, which might otherwise take prohibitively long to execute. Additionally, our work may be relevant to the philosophy of computation and embodiment, since it implies that auxiliary functions may be lurking in the specific operations that carry out a computation. For the observer who attends to the details of these operations, rather than just the computations they finally realize, additional information may be extracted about the computing system and the inputs over which it computes.

In the next section, we describe the system in which we simulate mechanical computation, before describing in detail what polycomputation is, and how it can be used for refractive computation.

Computational granular metamaterials.

Granular metamaterials are objects whose discrete parts, or granules, have been arranged in such a way as to yield unique properties resulting from their interaction (Wu et al., 2019; Pashine et al., 2019; Parsa et al., 2022b, 2023; Zangeneh-Nejad et al., 2020; Jiao et al., 2023). These differ from traditional composite materials in that their emergent properties are either qualitatively distinct from those found in nature, or that change with the activity of their parts and the environment in which they are embedded. *Computational* granular metamaterials (CGMMs) are those metamaterials whose emergent properties can be exploited for problems that are historically the ken of digital computers (Wu et al., 2019; Parsa et al., 2022b,a, 2023).

Prior work demonstrated that CGMMs can be built in simulation from two-dimensional sheets of idealized circular particles, whose sole means of interaction is the exertion of mechanical force. Using particle vibrations instead of voltage to represent inputs and outputs, these materials are configured to mechanically compute two-input logic gates. Externally applied vibrations along the x-axis of certain particles, realized by sinusoidal waves with amplitude, A, and frequency, ω, may be dampened or amplified in the material depending on the collective properties of its constituent parts. Differential amplification of external vibrations is measured as horizontal particle displacement at ω elsewhere in the material, and is taken to be the output computed by the sheet as a whole. For a driven particle, k, and initial horizontal position, $x_k(\mathrm{t}=0)$, the function that describes external horizontal driving under driving force, $\mathrm{d}_{\mathrm{k}}^{\omega}$, is given by:

$$\begin{aligned} x_k(t) &= x_k(0) + \mathrm{d}_k^{\omega} \\ &= x_k(0) + \eta_k A\, sin(\omega t) \end{aligned} \tag{1}$$

Where η_k is a binary switch for the k^{th} particle that either applies ($\eta_k = 1$) or withholds ($\eta_k = 0$) the external drive.

To evaluate a K-input logic gate over 2^{K} simulations, K input particles, $\{\mathrm{I}^0, ..., \mathrm{I}^{(\mathrm{K}-1)}\}$, are subject to external driving under all logically possible values for a binary string

of length K, where index k gives the binary switch for the k^{th} input particle. Input particles are thus either externally driven, or they are not. The 2^{K} possible switch strings for a K-input gate are contained in a list, $\mathbf{s}$, whose j^{th} index gives the switch string used in the j^{th} simulation. Thus, for a two-input logic gate with $\mathbf{s}$ = [‘00’, ‘01’, ‘10’, ‘11’], the switch string for the 2^{nd} simulation, $\mathbf{s}_2$, is equal to ‘10’, with binary switches $\mathbf{s}_2(0)$=1 and $\mathbf{s}_2(1)$=0 for respective inputs I^0 and I^1. This means that particle I^0 is driven, and particle I^1 is not. The external drive on the assembly for the j^{th} simulation is then represented as a function which takes a switch string as input, and generates K drives as output:

$$\text{F}_x^{\omega}(\mathbf{s}_j) = (\text{d}_0^{\omega}, ..., \text{d}_{\text{K}-1}^{\omega}) = \begin{cases} \mathbf{s}_j(0) A \sin(\omega t) \\ ... \\ \mathbf{s}_j(\text{K}-1) A \sin(\omega t) \end{cases} \tag{2}$$

If $\mathbf{s}_j(k) = 0$, the drive is not applied to particle k. Due to interparticle forces resulting from external drives, other particles in the assembly will exhibit varying amounts of horizontal displacement at the driving frequency, ω. For each of N designated output particles, $\{\text{O}^0, ..., \text{O}^{\text{N}-1}\}$, high displacement under $\text{F}_x^{\omega}(\mathbf{s}_j)$ is interpreted as signaling a value of ‘1’, while low displacement is interpreted as signaling a value of ‘0’. Displacement of the n^{th} output particle during the j^{th} simulation, at frequency ω, is calculated as the *gain* of that particle under input $\text{F}_x^{\omega}(\mathbf{s}_j)$:

$$G_{\mathbf{s}_j}^n(\omega) = \frac{\hat{f}_x^{\omega}(\text{O}_{\mathbf{s}_j}^n)}{\sum_{k=0}^{\text{K}-1} \hat{f}_x^{\omega}(\text{I}^{\text{k}})} \tag{3}$$

Since the object of concern is the vibrational response of the output particle at the driving frequency, ω, the fast Fourier transform, $\hat{f}_x^{\omega}$, is used to isolate particle behavior at just this frequency in the x-direction, ignoring its other vibrational modes. With a single output particle, we drop the superscript notation on $G_{\mathbf{s}_j}(\omega)$ and $\text{O}_{\mathbf{s}_j}$. Gain at ω in the absence of external drive ($\mathbf{s}_j$ =‘00’) can still be meaningfully quantified due to the inclusion of a source particle that vibrates constantly at the driving frequency for each simulation. Combining observed gains for all possible switch strings allows us to measure how well the granular assembly instantiates the desired logic gate at the n^{th} output particle.

Rational design of granular assemblies can be achieved by optimizing the properties of particles that collectively amplify or dampen supplied forces, like mass, modulus, position, and shape. In this paper, we vary only particle stiffness, as well as the indices of input, output, and source particles. Choosing these values for an arbitrary logic gate involves non-intuitive design considerations, and thus was delegated to an evolutionary algorithm.

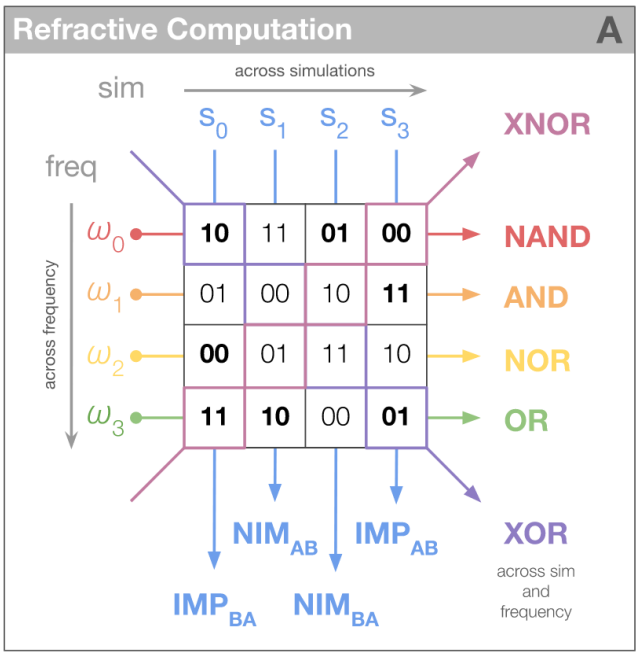

Figure 3: Evaluation of a K-input logic gate can be parallelized across *frequencies* in the event that (*i*) a polycomputational material instantiates at least 2^{K} K-input sequential logic gates over 2^{K} different frequencies; and, (*ii*) when the set of switch strings used in the j^{th} simulation covers the set of logically possible inputs to the K-input parallel gate. Parallelized computations are read along the columns of the visualized matrix, $\mathbf{S}$ (where A=I^0, B=I^1). As before, the k^{th} binary value in cell $\mathbf{S}_{cj}$ indicates whether the k^{th} input particle is externally driven during the j^{th} simulation at frequency ω_c. When a granular assembly polycomputes the sequential logic gates NAND, AND, NOR, and OR at respective frequencies ω_0, ω_1, ω_2, ω_3, following switch strings, $\mathbf{S}_{0:}$, $\mathbf{S}_{1:}$, $\mathbf{S}_{2:}$, $\mathbf{S}_{3:}$, a new logic gate, parallelized over all driving frequencies, can be defined for each of the j simulations. A parallel logic gate exists for each simulation just in case $\mathbf{S}$ realizes a Latin square, for which each switch string occurs exactly once in a given column and row. Finally, logic gates may be defined across *both* simulations and frequencies, e.g. along the diagonals of $\mathbf{S}$, yielding XOR and XNOR. The recipe for an XOR gate would then consist in attending to frequency ω_0 at j=0, ω_1 at j=1, ω_2 at j=2, and ω_3 at j=3, or [$\mathbf{S}_{00}$, $\mathbf{S}_{11}$, $\mathbf{S}_{22}$, $\mathbf{S}_{33}$]. Recipes for the computation of the six remaining logically possible gates: 0=[$\mathbf{S}_{10}$, $\mathbf{S}_{01}$, $\mathbf{S}_{32}$, $\mathbf{S}_{23}$] , 1=[$\mathbf{S}_{20}$, $\mathbf{S}_{31}$, $\mathbf{S}_{02}$, $\mathbf{S}_{13}$], A=[$\mathbf{S}_{00}$, $\mathbf{S}_{21}$, $\mathbf{S}_{32}$, $\mathbf{S}_{13}$], ¬A=[$\mathbf{S}_{20}$, $\mathbf{S}_{01}$, $\mathbf{S}_{12}$, $\mathbf{S}_{33}$], B=[$\mathbf{S}_{30}$, $\mathbf{S}_{11}$, $\mathbf{S}_{02}$, $\mathbf{S}_{23}$], ¬B=[$\mathbf{S}_{10}$, $\mathbf{S}_{31}$, $\mathbf{S}_{22}$, $\mathbf{S}_{03}$]. Although one may interpret assembly behavior as instantiating logic gates this way, each such recipe requires the application of input drives in the precise manner given by $\mathbf{S}$. This argument also applies to parallelized gates.

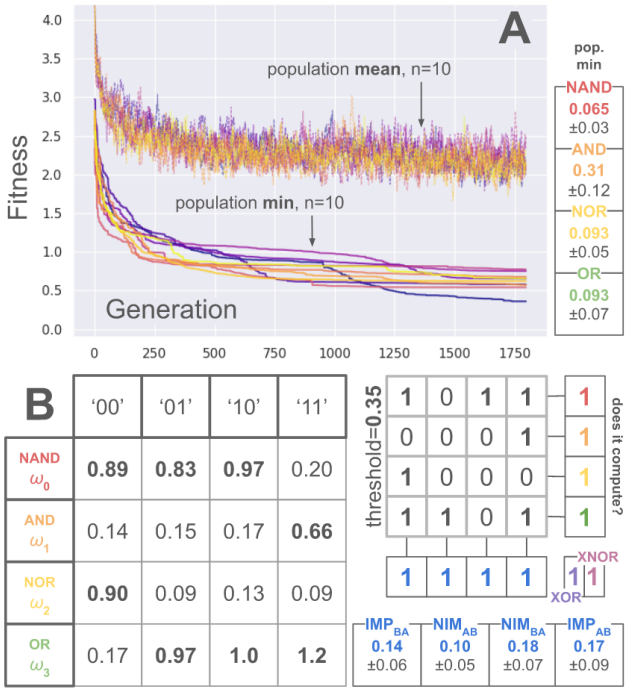

Figure 4: (**A**) Evolutionary history of the population mean (*dashed* lines) and population minimum (*solid* lines) for each independent trial (n=10). Mean and standard deviation for each logic gate in the population minimums are reported in the column on the right. (**B**) Observed gain values for each of the four evolved logic gates in the fittest materials over all evolutionary trials. Bold values indicate a target gain of 1, while non-bold values indicate a target gain of 0. For the corresponding matrix of switch strings, **S** (Fig. 3), we report a binary matrix of gains for an arbitrary threshold=0.2, and report whether the logic gates of **S** are faithfully executed by the material (0=no, 1=yes). Mean and standard deviation for each parallel logic gate in the fittest material from each trial is reported in the bottom right.

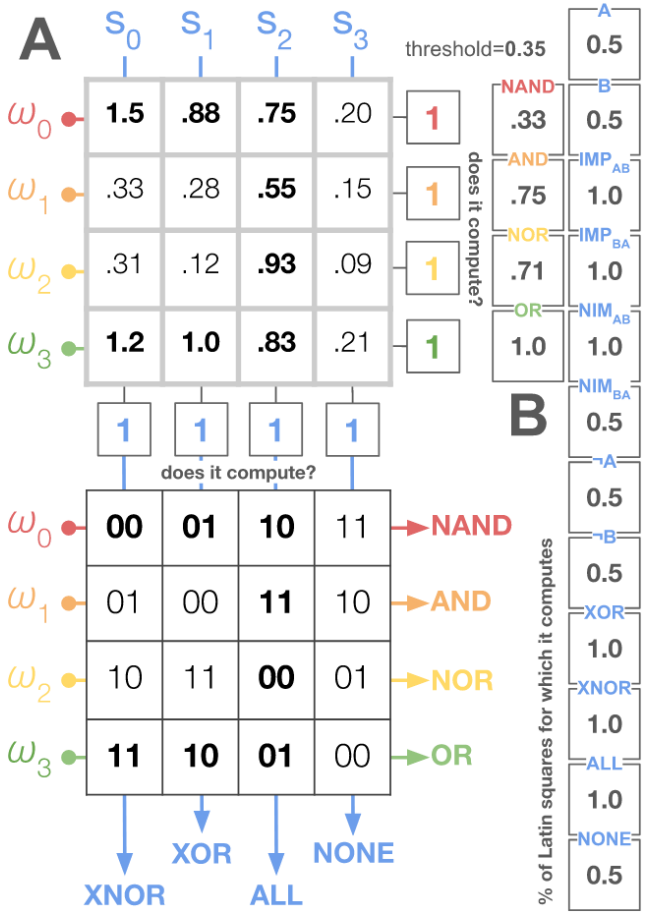

Figure 5: Successful computation under changes to the switch matrix, **S**.

Polycomputation.

When a single two-input sequential logic gate is desired, the driving force and vibrational response of the granular assembly involve just one frequency, ω, and yield an observed gain for each of the four possible switch strings. Yet the full vibrational response of a particle to the mechanical forces that act on it is equal to the *superposition*, or linear sum, of all its individual vibrational normal modes. Since the principle of superposition enables independent analysis of each vibrational mode, the potential arises for multiple gates operating *simultaneously* in the same material, each associated with a different driving frequency.

Prior work exploited the principle of superposition to accommodate two simultaneous NAND gates sharing the same input and output particles (Parsa et al., 2023). In this work, a NAND gate operating over frequency ω_0 was evidenced by the observed gains, $\{G_{\mathbf{s}_j}(\omega_0)\}$ for $j = 0, 1, 2, 3$, at the same time that a NAND gate operating over frequency ω_1 was evidenced by the observed gains, $\{G_{\mathbf{s}_j}(\omega_1)\}$ for $j = 0, 1, 2, 3$. Generically, the term "polycomputation" denotes the phenomenon in which the same parts are used to compute two or more functions at the same time.

External drives acting on *polycomputational* CGMMs in the j^{th} simulation are calculated as the sum of driving forces at C different frequencies, where C represents the number of logic gates embedded in the material:

$$\mathrm{x}_k(t) = x_k(0) + \sum_{c=0}^{\mathrm{C}-1} \mathrm{d}_k^{\omega_c} \tag{4}$$

Input drives for C logic gates can be abbreviated as $\mathrm{F}_x^{\omega_0}(\mathbf{s}_j)\mathrm{F}_x^{\omega_1}(\mathbf{s}_j)...\mathrm{F}_x^{\omega_{C-1}}(\mathbf{s}_j)$.

Refractive computation.

For polycomputational materials, the value of $\mathbf{s}_j$ for C simultaneous computations now becomes salient, since every input drive for a given computation coincides with $\mathrm{C} - 1$

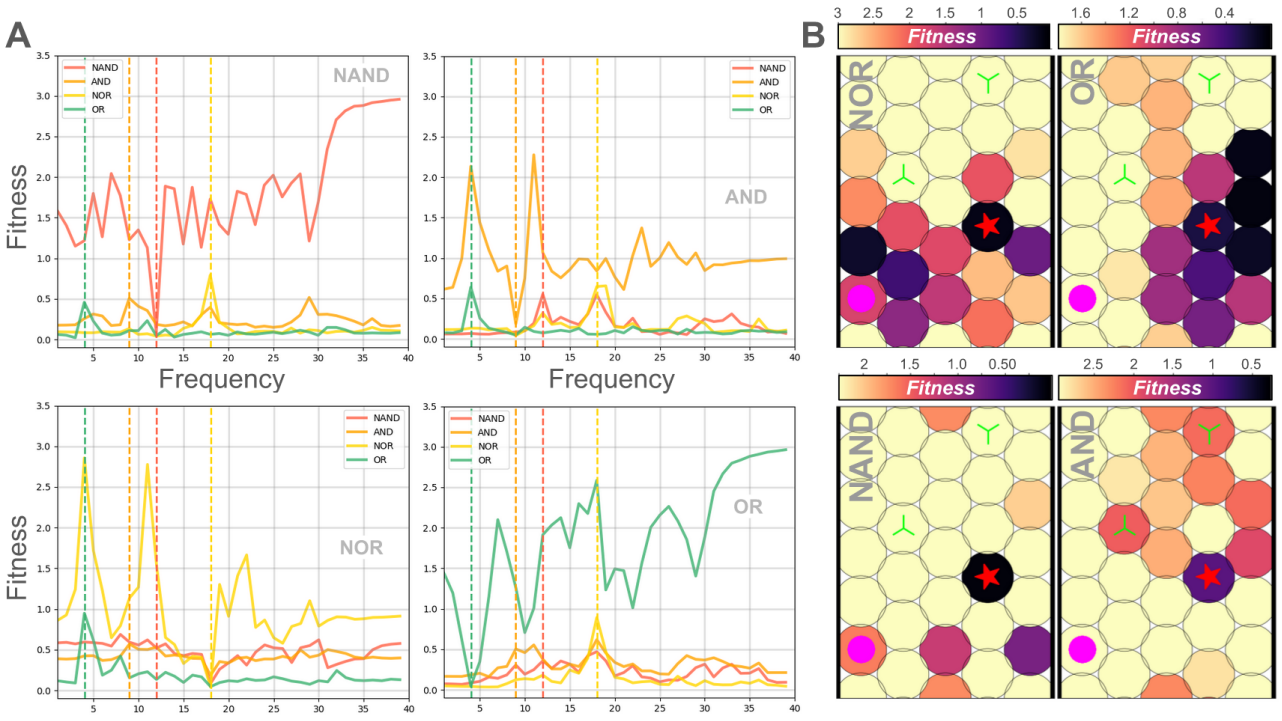

Figure 6: (**A**) For each sequential logic gate, we vary the frequency at which its input patterns are evaluated, holding fixed all other gates. Inset text indicates which gate we vary the frequency for. Vertical dashed lines denote the native frequency of each gate. (**B**) Distributed computation in the granular assembly whose stiffness ratios are visualized in Fig. 1.

other input drives that modulate the mechanical activity of the input particles. This is significant because the behavior of the assembly when the switch strings of two simultaneous computations are identical, e.g. input drive $= \mathrm{F}_x^{\omega_0}(\text{‘10’})\mathrm{F}_x^{\omega_1}(\text{‘10’})$, may diverge from the behavior of the assembly when their switch strings are different, e.g. input drive $= \mathrm{F}_x^{\omega_0}(\text{‘00’})\mathrm{F}_x^{\omega_1}(\text{‘10’})$. No special consideration for the order of **s** was given in prior work, as logic gates were simultaneously evaluated for the same input pattern only after evolutionary optimization (Parsa et al., 2023). This approach requires 4C sequential evaluations during evolution for two-input logic gates, and thus scales linearly in time with the number of gates desired.

Evolution can be made more efficient by evaluating logic gates simultaneously, collapsing $2^{\mathrm{K}}\mathrm{C}$ evaluations into just 2^{K} evaluations, but this may frustrate the optimization of any single logic gate. Conversely, simultaneous evaluation could render gates susceptible to corruption if their evolved mechanisms takes advantage of mechanical effects induced by input drives at other frequencies. In this case, withholding input frequency ω_0 may disable a logic gate operating over frequency ω_1. Such ‘entwining’ of physical processes has been reported previously under different circumstances, but is not unambiguously bad from the perspective of the engineer (Pask, 1958; Thompson, 1997). We reserve for a later date analysis of how simultaneous logic gates might interfere or synergize, and instead ask whether, and under what conditions, qualitatively new properties can be surfaced by a quantitative increase in polycomputational density.

We propose here that one such property is the parallel computation of logic gates across the set of forces that concurrently act on the assembly. When the number of sequential logic gates embedded in a polycomputational material equals the number of logically possible switch strings to another logic gate (e.g. 4 gates for a two-input logic gate), and each switch string occurs exactly once at a unique frequency for a given simulation (e.g. ‘00’ at ω_0, ... , ‘11’ at ω_3), the set of gains at each driving frequency during that simulation can be said to represent the full set of outputs for a new logic gate computed in parallel over the set of concurrent driving forces. We refer to this phenomenon as *refractive computation*.

With two-input (sequential) logic gates, we can represent the switch strings for a $\mathrm{C} = 4$ polycomputational assembly as a 4 by 4 matrix, $\mathbf{S}$, where each row is a list of j switch strings for the computation occurring at the c^{th} frequency. A parallelized logic gate for the j^{th} simulation is then possible if $\mathbf{S}$ is constructed such that all C external drives, $\mathrm{F}_x^{\omega_c}$, at j receive a unique switch string. The observed gain of the output particle at the c^{th} frequency during the j^{th} simulation is then the output computed by the sheet for the input specified by cell $\mathbf{S}_{cj}$, and the set of all gains for the j^{th} sim-

ulation represents the full set of outputs for the parallelized gate. Computation of a parallelized logic gate thus occurs down the j^{th} column of $\mathbf{S}$, or *across* frequencies, for the j^{th} simulation.

For example, using switch matrix $\mathbf{S}$ in Fig. 3 at the 0^{th} simulation, a parallelized logic gate is defined over the observed gains:

$$\{G_{\mathbf{S}_{00}}(\omega_0), G_{\mathbf{S}_{10}}(\omega_1), G_{\mathbf{S}_{20}}(\omega_2), G_{\mathbf{S}_{30}}(\omega_3)\} \quad (5)$$

When a unique switch string is used for each of the four driving frequencies in the j^{th} simulation, the identity of the parallelized gate for that simulation is determined by the target gains associated with each cell in the corresponding column. These are set by the choice of *sequential* logic gates occurring over fixed frequencies, and their associated target gains. For example, if a sequential NAND gate is chosen for frequency ω_0 with $\mathbf{S}_{0:}$ = [‘10’, ‘11’, ‘01’, ‘00’], the parallelized logic gate defined at the 0^{th} simulation is constrained to exhibit high gain at the output particle under switch string ‘10’ for frequency ω_0. This constraint thereby eliminates e.g. NOR from the set of candidate gates to be computed in parallel at the 0^{th} simulation. The full identify of the parallel gate for the j^{th} simulation is then determined by the choice of sequential logic gates at the remaining three frequencies.

Achieving a parallelized logic gate at each simulation in a $\mathrm{C} = 4$ polycomputational assembly limits $\mathbf{S}$ to the set of 4 by 4 Latin squares (n=576), for which a switch string occurs exactly once in each row and column. If the entries of $\mathbf{S}$ were randomly permuted, it is unlikely that the capacity for parallelization would be preserved, since the number of logically possible matrices far exceeds the number of Latin squares. Thus, refractive computation may be a property that ‘flickers’ in and out of existence when input drives don’t obey fixed patterns (Varley, 2022).

By varying the choice of sequential logic gates, different parallelized logic gates can be defined over the columns of $\mathbf{S}$. Our experimental proof of concept evolves the design of granular assemblies to compute sequential logic gates [NAND, AND, NOR, OR] with respective frequencies $\boldsymbol{\omega}$ =[12Hz, 9Hz, 18Hz, 4Hz], for a fixed Latin square (Fig. 3), yielding four distinct parallelized logic gates, [$\mathrm{IMPLY}_{\mathrm{BA}}$, $\mathrm{NIMPLY}_{\mathrm{AB}}$, $\mathrm{NIMPLY}_{\mathrm{BA}}$, $\mathrm{IMPLY}_{\mathrm{AB}}$], all of which are universal. Here, $\mathrm{IMPLY}_{\mathrm{AB}}$ symbolizes material implication, equal to $\neg A \vee B$, for input particles A=I^0 and B=I^1; while $\mathrm{NIMPLY}_{\mathrm{AB}}$ symbolizes nonimplication, $A \wedge \neg B$.

Finally, we note that if logic gates can be parallelized across frequencies, $\boldsymbol{\omega}$, under switch strings, $\mathbf{S}_{:j}$ in the j^{th} simulation, by postulating that every input to a logic gate can be simultaneously evaluated at a different frequency, then logic gates may be defined for which each input occurs at a different frequency in a different simulation. Examples of this variety are visualized in Fig. 3 on the diagonals of the matrix, $\mathbf{S}$ (XOR and XNOR).

Methods

Simulation.

We follow the experimental protocol detailed in (Parsa et al., 2023). Granular assemblies are simulated according to the Discrete Element Method. For each assembly, 30 frictionless particles of variable stiffness ratios $\in [0.05, 10]$, having mass=1 and diameter=0.1, are placed on a 5 by 6 2D triangular lattice with a periodic boundary condition along the x-axis. Particle-particle interactions are then modeled as repulsive forces with a Lennard-Jones potential. Prior to the application of external drives, the initial positions of particles are adjusted by the Fast Inertial Relaxation Engine (FIRE) to ensure stable particle packing (Bitzek et al., 2006). Under external drives, a simulation time of $5e3$ was used. Code for these experiments can be found at https://github.com/shawnbeaulieu/Refractive-Computation.

Evolution.

Designing a granular assembly that computes a sequential logic gate is a complex engineering problem for which an evolutionary algorithm was previously used (Parsa et al., 2023). To simplify the task of evolution in this paper, only the locations of the input, output, and source particles, as well as the stiffness ratios of all particles, were evolved using the Age-Fitness Pareto Optimization algorithm, or AFPO (Schmidt and Lipson, 2010). Other properties, like particle mass (1.0) and diameter (0.1), were held constant. Genomes were represented as vectors containing 30 stiffness ratios $\in$ [0.05 (*soft*) , 10 (*stiff*)] and four indices, giving the locations of I^0, I^1, O, and source particle (*src*), respectively.

Genomes were then realized in simulation as 5 by 6 granular assemblies, and subject to all logically possible switch strings for four two-input logic gates, each occurring at one of four different frequencies $\boldsymbol{\omega}$=[12Hz, 9Hz, 18Hz, 4Hz]. At 12 Hz, we selected for a NAND gate using ordered switch strings $\mathbf{S}_{0:}$ (Fig. 3), at the same time that we selected for an AND gate, a NOR gate, and an OR gate at respective frequencies [9 Hz, 18Hz, and 4Hz] using respective switch strings $\mathbf{S}_{1:}$, $\mathbf{S}_{2:}$, $\mathbf{S}_{3:}$. Given these frequencies, gates (NAND, OR) and (AND, NOR) are harmonic pairs. Fitness was defined as the cumulative L2 loss of the observed gains, $G_{\mathbf{S}_c}(\omega_c)$, with respect to target gains, $T_{\mathbf{S}_c}(\omega_c)$, at the c^{th} gate, for all 4 gates evaluated simultaneously:

$$\text{Fitness} = \sum_{c=0}^{C-1} \sum_{k=0}^{2^{\mathrm{K}}-1} \left(T_{\mathbf{S}_c(k)}(\omega_c) - G_{\mathbf{S}_c(k)}(\omega_c) \right)^2 \quad (6)$$

Raw gains are are clipped at a value of 1. For stiffness ratios, a Gaussian mutation operator with a standard deviation of 0.5 was applied with probability=0.75; while mutations to the indices of ports (I^0, I^1, O, *src*), occurred with probability=0.25, and changed a randomly sampled port location to another random particle taken from the set of particles not

currently being used as any of the four ports. 10 independent trials were conducted, each starting with a randomly initialized population of size 175. Each trial was run for 1800 generations. Parallelized logic gates, defined across frequencies, as well as any logic gate defined across simulations *and* frequencies (i.e. logic gates defined on the diagonals of **S**) were not explicitly selected for.

Comprehensive analysis of the optimal choice of driving frequencies will be the subject of future research. However, we observed mean fitness scores for the best performing material over 5 independent runs for the following treatments: (*i*) all frequencies set to prime numbers [3Hz, 5Hz, 7Hz, 11Hz] for gates 1-4 (mean=1.03 $\pm$ 0.166); (*ii*) all frequencies set to harmonic multiples of 4, [4Hz, 8Hz, 12Hz, 16Hz] (mean=0.82 $\pm$ 0.03); (*iii*) harmonic pairs (NAND, AND) and (NOR, OR), [12Hz, 4Hz, 18Hz, 9Hz] (mean=0.70 $\pm$ 0.12); (*iv*) harmonic pairs (NAND, NOR) and (AND, OR) (mean=0.83 $\pm$ 0.084); and (*v*) harmonic pairs used in the main treatment, but with frequencies swapped within each pair, [4Hz, 18Hz, 9Hz, 12Hz] (mean=0.82 $\pm$ 0.05). For the main treatment: mean=0.62 $\pm$ 0.11.

Results

Fitness at each generation for the population *average* and population *minimum* in each evolutionary trial is reported in Fig. 4A. Since only *cumulative* loss is reported as fitness, proficiency on any single computation is obscured. For example, the fitness for the fittest individual across all trials resolves into L2 losses of 0.02, 0.28, 0.13, and 0.04 for NAND, AND, NOR, and OR, respectively. While for the parallelized gates we find L2 losses of 0.03, 0.07, 0.09, and 0.15 for $\mathrm{IMP}_{\mathrm{BA}}$, $\mathrm{NIM}_{\mathrm{AB}}$, $\mathrm{NIM}_{\mathrm{BA}}$, and $\mathrm{IMP}_{\mathrm{AB}}$. The observed gains from which loss is calculated under the switch strings, **S** (Fig. 3), are reported in Fig. 4B. Mean and standard deviation for logic gates defined across the rows of **S** are given in the top right column for the fittest individuals (pop. min). We find that evolution struggles to uniformly minimize loss on all logic gates, and struggles specifically to optimize the AND gate. Persistent trouble with computing AND across evolutionary trials may indicate a systemic bias caused by hyperparameter settings, such as the choice of driving frequencies.

Next we evaluate sensitivity to changes in driving frequency for the fittest evolved material (Fig. 6A). For each evolved gate, we vary the frequency of the driving force for that gate in the range [1Hz, 40Hz], and observe the effect this has on fitness for all four logic gates. Changing the driving frequency for any single logic gate is always deleterious relative to the native frequency of that gate. However, the effect on other logic gates appears marginal, except in the case that the new frequency for the altered gate coincides with the native frequency of another gate. Here we observe error spikes that are likely the result of interference between gates.

Fig. 6B visualizes how the computation of sequential logic gates at their native frequency is distributed in the granular assembly for an evolved individual. For each sequential logic gate, we compute the fitness for that gate should the index of the output port be changed to that of any other particle. The fitness calculated for an alternative output particle is then used to color that particle in the heat map. This allows us to visualize where, and to what extent, each gate is computed by the material. We find that computation is not confined to a single location, but varies depending on the identity of the gate, and tends not to be concentrated in the stiffest particles (Fig. 1). At the same time, distributed computation reveals candidate locations from which to read out logic gates in the event of injury, malfunction, or design modifications to the material.

Finally, we document the performance of the fittest evolved material under changes to the switch matrix, **S**. We consider only the set of remaining Latin squares (n=575) as alternative switch matrices. Because Latin squares differ in the order of switch strings along their columns and rows, the identity of parallelized logic gates for each simulation will likewise change, under the constraint that NAND, AND, NOR, and OR are evaluated sequentially at frequencies $\boldsymbol{\omega}$. In Fig. 5A (bottom) we show a randomly sampled switch matrix, whose permuted entries result in the parallelized gates [XNOR, XOR, ALL, NONE]. Different Latin squares may be used to generate different parallelized logic gates. Indeed, we find that the remaining 12 logically possible gates emerge as parallel gates within some subset of Latin squares, with each two-input logic gate appearing in precisely 192 cases (or 33% of Latin squares).

The top matrix reports the observed gains at each frequency and simulation under the new pattern of inputs. Since the evolved material was only ever exposed to a fixed pattern of inputs, and the new pattern of inputs may cause previously independent input drives to co-occur at the j^{th} simulation, there is likely to be either mechanical interference or loss of evolved synergy between input drives. Nevertheless, for the visualized switch matrix we observe behavior adequate to regard as computing all four sequential logic gates, and all four parallel logic gates. Gates on the diagonal are no longer defined for this matrix.

In Fig. 5B, we give the mean performance for all 16 logically possible two-input gates over all 576 Latin squares. For this metric, we threshold gain values at 0.35, where $G{<}0.35 = 0$ and $G{\geq}0.35 = 1$, and report a value of 1 if all relevant gains for a given logic gate are above this threshold, and a value of 0 otherwise. This threshold was selected arbitrarily by inspection and then applied uniformly across the set of Latin squares. The numbers reported are then the percent of Latin squares for which a logic gate appears and is computed by the material. Boxes with blue text give the performance of parallelized gates, while boxes with the labels of the evolved sequential gates demonstrate the robustness

of each such gate to changes in the ambient forces acting on the material. For example, the computation of NAND at frequency ω_0 successfully occurs 33% of the time when the input drives at $\boldsymbol{\omega}_{1:4}$ are shuffled by the use of a new Latin square. While the computation of OR at ω_3 is successfully computed under all considered changes to the pattern of input drives.

Conclusion

In the preceding sections, we reported on the automated design of a simulated granular metamaterial which instantiates logic gates that are parallelized over driving frequencies. Parallel logic gates are possible because of a previously demonstrated property called *polycomputation*, in which multiple operations can be performed at the same time in the same place. Polycomputation is a consequence of the principle of superposition, which holds when simultaneous external drives on a system can be resolved into each drive acting independently. In this paper, polycomputation of multiple logic gates happens at different frequencies of driving force acting on a sheet of idealized particles. *Parallelized* logic gates exploit the fact that when polycomputational density is sufficiently high, and when external drives act on particles in a particular way, additional computations can be defined across driving frequencies and computed all at once in a single simulation. We therefore achieve with these results greater computational *density* and *efficiency* than prior work. More provocatively, if the specific operations that realize a given computation can themselves be used to realize additional computations, depending on how these operations are interpreted, our work may impinge on the notion that we can simulate how a living system executes and exploits specific computations without exactly reproducing the operations it performs. Or that the details of the mechanism responsible for a given computation can be ignored insofar as this computation is successfully executed. Under an observer-dependent view of computation, it may not be obvious whether the behaviors of a living system we observe and label as instantiating particular computations are those that the system itself uses—now or in the future. One might look at our granular assembly and see only four sequential logic gates. Another observer may see something entirely different.

Although parallelized logic gates depend on a precise ordering of input drives, our analysis shows that parallelized gates are modestly robust to permuted patterns of driving force (Fig. 5B). Additional limitations of this work include that physical simulations are highly idealized, to the point where particles are assumed to be frictionless, and that computations more complex than two-input logic gates aren't considered. Future work will elaborate on all of these shortcomings with the aim of real world implementation.

Acknowledgements

We would like to acknowledge financial support from the National Science Foundation under the DMREF program (award number: 2118810). Computations were performed, in part, on the Vermont Advanced Computing Center (VACC).

References

Bitzek, E., Koskinen, P., Gähler, F., Moseler, M., and Gumbsch, P. (2006). Structural relaxation made simple. *Physical Review Letters*, 97(17).

Bongard, J. and Levin, M. (2021). Living things are not (20th century) machines: Updating mechanism metaphors in light of the modern science of machine behavior. *Frontiers in Ecology and Evolution*, 9.

Bongard, J. and Levin, M. (2023). There's plenty of room right here: Biological systems as evolved, overloaded, multi-scale machines. *Biomimetics*, 8(1):110.

Conrad, M. (1972). Information processing in molecular systems. *Biosystems*, 5(1):1–14.

Conrad, M. (1985). On design principles for a molecular computer. *Communications of the ACM*, 28(5):464–480.

Dillavou, S., Stern, M., Liu, A. J., and Durian, D. J. (2022). Demonstration of decentralized physics-driven learning. *Physical Review Applied*, 18(1).

El Helou, C., Grossmann, B., Tabor, C. E., Buskohl, P. R., and Harne, R. L. (2022). Mechanical integrated circuit materials. *Nature*, 608(7924):699–703.

Gorecki, J., Gizynski, K., Guzowski, J., Gorecka, J. N., Garstecki, P., Gruenert, G., and Dittrich, P. (2015). Chemical computing with reaction–diffusion processes. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 373(2046):20140219.

Jaeger, H., Noheda, B., and van der Wiel, W. G. (2023). Toward a formal theory for computing machines made out of whatever physics offers. *Nature Communications*, 14(1).

Jiao, P., Mueller, J., Raney, J. R., Zheng, X., and Alavi, A. H. (2023). Mechanical metamaterials and beyond. *Nature Communications*, 14(1).

Kaspar, C., Ravoo, B. J., van der Wiel, W. G., Wegner, S. V., and Pernice, W. H. P. (2021). The rise of intelligent matter. *Nature*, 594(7863):345–355.

McMillen, P. and Levin, M. (2024). Collective intelligence: A unifying concept for integrating biology across scales and substrates. *Communications Biology*, 7(1).

Mei, T. and Chen, C. Q. (2023). In-memory mechanical computing. *Nature Communications*, 14(1).

Nakajima, K. (2020). Physical reservoir computing—an introductory perspective. *Japanese Journal of Applied Physics*, 59(6):060501.

Parsa, A., Wang, D., O'Hern, C. S., Shattuck, M. D., Kramer-Bottiglio, R., and Bongard, J. (2022a). Evolving programmable computational metamaterials. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 122–129.

Parsa, A., Wang, D., O'Hern, C. S., Shattuck, M. D., Kramer-Bottiglio, R., and Bongard, J. (2022b). Evolution of acoustic logic gates in granular metamaterials. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, pages 93–109. Springer.

Parsa, A., Witthaus, S., Pashine, N., O'Hern, C., Kramer-Bottiglio, R., and Bongard, J. (2023). Universal mechanical polycomputation in granular matter. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 193–201.

Pashine, N., Hexner, D., Liu, A. J., and Nagel, S. R. (2019). Directed aging, memory, and nature's greed. *Science Advances*, 5(12).

Pask, G. (1958). Physical analogues to the growth of a concept. *Mechanization of Thought Processes*, pages 769–764.

Pask, G. (1960). The natural history of networks. *Self-Organizing Systems*, pages 232–263.

Pasquinelli, M. (2023). *The Eye of the Master: A Social History of Artificial Intelligence*. Verso Books.

Rosen, R. (1994). On psychomimesis. *Journal of Theoretical Biology*, 171(1):87–92.

Schmidt, M. and Lipson, H. (2010). *Age-Fitness Pareto Optimization*, page 129–146. Springer New York.

Serra-Garcia, M. (2019). Turing-complete mechanical processor via automated nonlinear system design. *Physical Review E*, 100(4).

Sinapayen, L., Masumori, A., and Ikegami, T. (2017). Learning by stimulation avoidance: A principle to control spiking neural networks dynamics. *PLOS ONE*, 12(2):e0170388.

Stern, M. and Murugan, A. (2023). Learning without neurons in physical systems. *Annual Review of Condensed Matter Physics*, 14(1):417–441.

Thompson, A. (1997). *An evolved circuit, intrinsic in silicon, entwined with physics*, page 390–405. Springer Berlin Heidelberg.

Varley, T. F. (2022). Flickering emergences: The question of locality in information-theoretic approaches to emergence. *Entropy*, 25(1):54.

Wu, Q., Cui, C., Bertrand, T., Shattuck, M. D., and O'Hern, C. S. (2019). Active acoustic switches using two-dimensional granular crystals. *Physical Review E*, 99(6).

Yasuda, H., Buskohl, P. R., Gillman, A., Murphey, T. D., Stepney, S., Vaia, R. A., and Raney, J. R. (2021). Mechanical computing. *Nature*, 598(7879):39–48.

Zangeneh-Nejad, F., Sounas, D. L., Alù, A., and Fleury, R. (2020). Analogue computing with metamaterials. *Nature Reviews Materials*, 6(3):207–225.

Zhu, L., Kim, S.-J., Hara, M., and Aono, M. (2018). Remarkable problem-solving ability of unicellular amoeboid organism and its mechanism. *Royal Society Open Science*, 5(12):180396.