A novel technique for minimizing energy functional using neural networks

Sanjeeb Poudel[a,*], Xiaoqiang Wang[a], Sanghyun Lee[b]

[a]*Department of Scientific Computing,*
[b]*Department of Mathematics, Florida State University, Tallahassee, 32306, FL, United States*

Abstract

An energy functional describes the equilibrium state of a system. In this work, we present a novel technique, Functional Optimization using Neural Networks (FONN), for minimizing the system's energy. FONN utilizes neural networks to process information at discrete grid points, considering their interactions with neighboring grid points, to update the state of the system. The training process involves formulating a loss function based on the system's energy, and with the help of multiple fine-tuning steps, the method employs a progressive energy reduction technique that decreases the energy in multiple steps. FONN's effectiveness is demonstrated across various problems, including the minimization of the heat and Lyapunov energy. Moreover, the paper explores the minimization of the elastic bending energy with an area constraint.

Keywords: neural networks, energy minimization, heat energy, Lyapunov energy, elastic bending energy

1. Introduction

The energy functional of a system, which depends on the system's state, is a scalar quantity representing the total energy contained within the system. The system naturally transitions from a state of higher energy to one of lower energy, which corresponds to a stable or stationary state. These energy

*Corresponding Author
Email addresses: spoudel@fsu.edu (Sanjeeb Poudel), wwang3@fsu.edu (Xiaoqiang Wang), lee@math.fsu.edu (Sanghyun Lee)

functionals are used in various disciplines of engineering; for instance, the Lyapunov energy functional [1] finds applications in simulating multiphase problems [2, 3, 4, 5, 6], and the elastic bending energy [7] represents the equilibrium shape of a lipid vesicle membrane [8, 9, 10, 11].

The classical approach to solving these problems includes utilizing the standard gradient flow of the energy functional to formulate a time-dependent partial differential equation (PDE). Then, various numerical techniques are employed to approximate the solutions. These include the adoption of various orders of implicit, semi-implicit, or explicit schemes for time discretization [1, 5, 9, 12, 13], while the spatial discretization is achieved using finite difference or finite element or spectral methods [8, 5, 14, 15, 10, 12, 16]. On the other hand, one can consider energy minimization using techniques such as Newton's method or the conjugate gradient method.

In this paper, we propose a novel approach to utilize neural networks to minimize energy functionals instead of changing the problem to a time-dependent PDE. Neural networks are gaining traction in various fields due to their ability to approximate non-linear functions in high dimensions [17], and advancements in computing power have enabled the large-scale use of neural networks [18, 19]. The emerging field of solving classical problems by employing neural networks, known as scientific machine learning [20], finds application in diverse fields, including but not limited to fluid mechanics [21, 22, 23], solid mechanics [24, 25], heat transfer [26, 27], and many others. The utilization of neural networks in various contexts underscores the versatility of this technique and offers exciting prospects for the future of problem-solving in the field of science and engineering.

The novel approach presented in this paper utilizes neural networks that take the system's current state as input and produce an updated state as output. The network's architecture is similar to the one used for processing graph data [28, 29]. This network structure comprises two fundamental steps. First, it collects information from neighboring vertices of a node, and then, in the second step, it employs this gathered information to update the node's value. The motivation to use the structure stems from the traditional explicit method for time-dependent problems. For example, the finite difference spatial approximation with the forward Euler method in time operates based on the local information of the grid. The size of the stencil determines the number of points in the grid used for the estimation. In our approach, we employ neural networks to extract and utilize this local information for updating the system's state.

To implement our method, we begin by discretizing the computational domain using a uniform grid. Then, a neural network is used to extract local information for each grid point. This information is used by another neural network in conjunction with the current value of the grid point to update its state. The training process involves a loss function defined using the energy functional and the change between the input and the output states. Additionally, we use a multi-step approach to minimize the energy. Starting from the initial state, the network lowers the energy to obtain a different state, which is used as an input for the subsequent energy reduction step. The main contributions of the paper can be summarized in the following points:

- Our proposed method shares a structural resemblance with an existing network architecture which is known as Propagation Networks (PropNet) [29] that is based on directed graphs. In this work, we have formulated a network tailored to work efficiently for a regular grid.
- We have developed a training strategy that updates the system's state by reducing its total energy through a loss function based on the energy.
- We have formulated a progressive energy reduction technique, which involves decreasing the energy in multiple steps.

2. Background

In this section, we briefly recapitulate the Interaction Network (IN) [28] and Propagation Network (PropNet) [29]. To be more specific, we discuss "Vanilla PropNet" [29], which we refer to as "PropNet" for simplicity.

2.1. Interaction Network

Interaction Network (IN) is a learnable physics engine that infers the dynamics of a physical system, represented as graphs. Given a graph representation G^n of a physical system at some state n, IN computes the system's subsequent state (G^{n+1}), i.e.,

$$G^{n+1} = \mathrm{IN}(G^n). \tag{1}$$

The graph $G = \langle O, R \rangle$ is a directed graph, where O (nodes) represents the objects, and R (edges) represents their relations. For a relation $r_j \in R$,

where j ranges from 1 to the number of relations $|R|$, there are two objects: $o_{a_j} \in O$ is the sender, and $o_{b_j} \in O$ is the receiver of the relation r_j.

IN incorporates a relation-centric function (f_R) and an object-centric function (f_O). At a given state n, f_R takes three input parameters: $o^n_{a_j}, o^n_{b_j}$, and r_j, and computes the effect of interaction (e^n_j) between the two objects:

$$f_R(o^n_{a_j}, o^n_{b_j}, r_j) = e^n_j, \qquad j = 1, 2, \ldots |R|. \tag{2}$$

On the other hand, f_O takes two inputs: the current state of an object o^n_i, where i ranges from 1 to the number of objects $|O|$, and the total effects for which the object is the receiver. Then, it computes the future state of the object:

$$f_O(o^n_i, \sum_{k \in \mathcal{N}_k} e^n_k) = o^{n+1}_i, \qquad i = 1, 2, \ldots |O|, \tag{3}$$

where $\mathcal{N}_k$ denotes the total number of the relations for which the node o^n_i is the receiver.

A simple example where $|R| = 2$ and $|O| = 3$ is given in the following. The graph has three nodes and two directed edges (see Fig. 1 for more details). In the context of relation r_1, o_1 is the sender, while o_3 is the receiver (i.e., $a_1 = 1$ and $b_1 = 3$). Similarly, in the relation r_2, the sender is o_2 and the receiver is o_3.

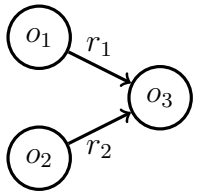


Figure 1: A directed graph with three nodes and two edges

The relation-centric function (f_R) determines the effect of interaction (e^n_j) between two nodes. Thus, the calculation of the effect can be expressed as follows:

$$\begin{aligned} f_R(o^n_3, o^n_1, r_1) &= e^n_1, \\ f_R(o^n_3, o^n_2, r_2) &= e^n_2. \end{aligned} \tag{4}$$

Since o_3 is the receiver for both relations, the total effect of interaction for o_1 and o_2 is 0. On the other hand, for o_3, the total effect is the sum of e^n_1

and e_2^n. The object-centric function (f_O) updates the nodes as follows:

$$\begin{aligned} f_O(o_1^n, 0) &= o_1^{n+1}, \\ f_O(o_2^n, 0) &= o_2^{n+1}, \\ f_O(o_3^n, e_1^n + e_2^n) &= o_3^{n+1}. \end{aligned} \tag{5}$$

2.2. Propagation Network

PropNet extends the idea of IN so that the effect of interaction propagates beyond a pairwise interaction. It uses a multistep message passing along the directed edges in graph G^n. To initiate this process, PropNet introduces a new variable for each node in the graph, denoted as h_i^m, where m is a message passing step. The index i ranges from 1 to the number of nodes in the graph $|O|$. At the initial message passing step ($m = 0$), the value of h_i^0 is set to 0.

Similar to IN, at each message passing step m, PropNet relies on two key functions, namely f_O^m and f_R^m; however, the method extends the parameters of f_R^m by two additional variables, $h_{a_j}^{m-1}$ and $h_{b_j}^{m-1}$. Similar to the previous context, a_j and b_j denote the receiver and sender of a relation r_j, respectively. The calculation of e_j^m is done using the following equation:

$$f_R^m(o_{a_j}^n, o_{b_j}^n, r_i, h_{a_j}^{m-1}, h_{b_j}^{m-1}) = e_j^m, \quad j = 1, 2, \ldots |R|. \tag{6}$$

Instead of directly updating the object's state using the function f_O^m, this method stores the updates within h_i^m. The calculation is expressed as follows:

$$f_O^m(o_i^n, \sum_{k \in \mathcal{N}_k} e_k^m) = h_i^m, \quad i = 1, 2, \ldots |O| \tag{7}$$

Then, in the subsequent message passing step ($m + 1$), it uses f_O^{m+1} and f_R^{m+1} along with h_i^m to compute h_i^{m+1}. This technique allows the network to propagate the effects beyond the pairwise interactions.

In summary, Algorithm 1 provides a clear demonstration of this approach. We can notice that setting $M = 1$ restricts the method to consider only pairwise interactions. However, increasing the value of M allows the propagation of information beyond the pairwise interaction.

In previous literature [29], the authors demonstrated the superior performance of the multi-step effect propagation technique over pairwise interaction in a wide range of examples, like Newton's cradle and string manipulation.

Algorithm 1 PropNet

Require: $o_i^n, \quad i = 1, 2, \ldots |O|$

$h_i^0 \leftarrow 0, \quad i = 1, 2, \ldots |O|$

for $m = 1, ..., M$ **do**

  $e_j^m \leftarrow f_R^m(o_{a_j}^n, o_{b_j}^n, r_j, h_{a_j}^{m-1}, h_{b_j}^{m-1}), \quad j = 1, 2, \ldots, |R|$

  $h_i^m \leftarrow f_O^m(o_i^n, \sum_{k \in \mathcal{N}_k} e_k^m), \quad i = 1, 2, \ldots, |O|$

end for

$o_i^{n+1} \leftarrow h_i^M, \quad j = 1, 2, \ldots |O|$

3. Functional Optimization using Neural Networks (FONN) Method

In this section, we introduce a novel method, which we name Functional Optimization using Neural Networks (FONN). We explain the structural details of the method, the loss function employed, and the training strategy for the networks.

3.1. Updating the System's State

The computational domain $\Omega \subset \mathbb{R}^2$ is discretized using a uniform grid, as shown in Fig. 2. Each grid point is associated with a state denoted as $\phi_{i,j}^n$, corresponding to a node located at (i, j). Here, i ranges from 1 to the number of grid points in the x-direction (N_x), and similarly, j ranges from 1 to the number of grid points in the y-direction (N_y). Given that the grid does not have a sender-receiver relationship, the function f_R is used for all the edges, considering the two adjacent nodes, to calculate the effect.

Similar to PropNet, we define a variable $\psi_{i,j}^m$ for each node. Throughout the message passing steps, denoted as step m, we consider the same functions: f_R to compute the interacting effect and f_O to update the state. If $e_{i,j,i',j'}^m$ denotes the propagating effect between two adjacent nodes at (i, j) and (i', j'), we have

$$\begin{aligned} f_R(\phi_{i,j}^n, \phi_{i-1,j}^n, \psi_{i,j}^{m-1}, \psi_{i-1,j}^{m-1}) &= e_{i,j,i-1,j}^m, \\ f_R(\phi_{i+1,j}^n, \phi_{i,j}^n, \psi_{i+1,j}^{m-1}, \psi_{i,j}^{m-1}) &= e_{i+1,j,i,j}^m, \\ f_R(\phi_{i,j}^n, \phi_{i,j-1}^n, \psi_{i,j}^{m-1}, \psi_{i,j-1}^{m-1}) &= e_{i,j,i,j-1}^m, \\ f_R(\phi_{i,j+1}^n, \phi_{i,j}^n, \psi_{i,j+1}^{m-1}, \psi_{i,j}^{m-1}) &= e_{i,j+1,i,j}^m. \end{aligned} \tag{8}$$

As seen in Eqn. 8, the order of arguments for the function f_R is taken based on the relative position of the node. The node that is in the positive direction

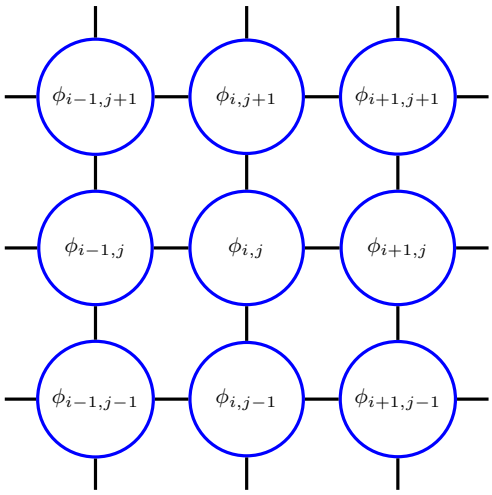


Figure 2: Regular grid in a two-dimensional domain

precedes the node in the negative direction. Finally, the effect is considered equally and oppositely for the two nodes. Thus, for each node, the effect is combined using:

$$c_{i,j} = e^m_{i,j,i-1,j} - e^m_{i+1,j,i,j} + e^m_{i,j,i,j-1} - e^m_{i,j+1,i,j}. \tag{9}$$

Once the total interacting effect $c_{i,j}$ is calculated, the function f_R updates the parameter $\psi^m_{i,j}$ as:

$$f_O(\phi^n_{i,j}, c_{i,j}) = \psi^m_{i,j}. \tag{10}$$

After the final message passing step $m = M$, the system's state is updated with $\psi^M_{i,j}$. Algorithm 2 summarizes the steps involved in calculating the output (ϕ^{n+1}) for a given input (ϕ^n).

The functions f_R and f_O can be defined in various ways. In our implementation, we use fully-connected neural networks to model f_R and f_O for updating the system's state [28, 29]. This enables the method to adapt and optimize the weights of the networks through gradient-based optimization techniques.

3.2. Neural Networks

For $L \in \mathbb{N}$ and $\vec{\boldsymbol{n}} = (n_0, n_1, \ldots, n_L) \in \mathbb{N}^{L+1}$, a L-layer feed-forward neural network is the mapping $\mathbb{R}^{n_0} \ni x \mapsto z^L \in \mathbb{R}^{n_L}$ where z^L is defined recursively by

$$z^\ell = \sigma(W^\ell z^{\ell-1} + b^\ell), \quad 2 \le \ell \le L, \quad z^1 = \sigma(W^1 x + b^1).$$

Algorithm 2 Updating the system's state

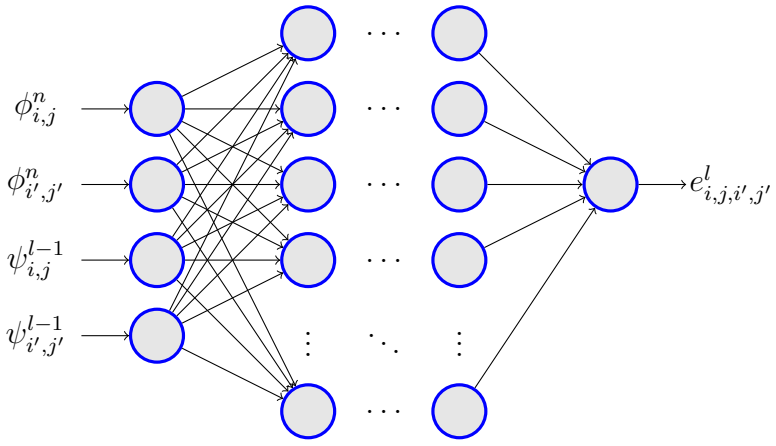
Require: $\phi^n_{i,j}, \quad i = 1, 2, \ldots, N_x \quad j = 1, 2, \ldots, N_y$

$\psi^0_{i,j} \leftarrow 0, \quad \forall i, j$
for $m = 1, ..., M$ **do**
  for each adjacent pair of nodes (i, j) and (i', j') **do**
    $e^m_{i,j,i',j'} \leftarrow f_R(\phi^n_{i,j}, \phi^n_{i',j'}, \psi^{m-1}_{i,j}, \psi^{m-1}_{i'j'})$
  end for
  $c_{i,j} = e^m_{i,j,i-1,j} - e^m_{i+1,j,i,j} + e^m_{i,j,i,j-1} - e^m_{i,j+1,i,j}, \quad \forall i, j$
  $\psi^m_{i,j} \leftarrow f_O(\phi^n_{i,j}, c_{i,j}), \quad \forall i, j$
end for
$\phi^{n+1}_{i,j} \leftarrow \psi^M_{i,j}, \quad \forall i, j$

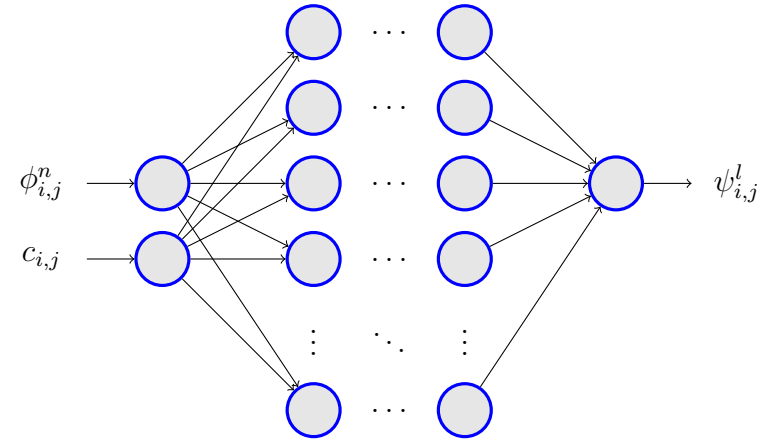
Here, σ is a non-linear activation function that applies element-wise, $W^\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and $b^\ell \in \mathbb{R}^{n_\ell}$ are the weight matrix and the bias vector of the ℓ-th layer, respectively. The vector $\vec{\boldsymbol{n}}$ is referred to as the feature dimensions at different layers and $\{W^\ell, b^\ell\}_{\ell=1}^L$ is to as the network parameters.

Thus, we have two networks $\vec{\boldsymbol{n}}_R$ and $\vec{\boldsymbol{n}}_O$ for f_R and f_O respectively. The network $\vec{\boldsymbol{n}}_R$ for f_R has four input nodes and one output node; similarly, the network $\vec{\boldsymbol{n}}_O$ for f_O has two input nodes and one output node. Fig. 3 shows the schematic diagram of the networks.

Two networks are simultaneously trained using a loss function based on energy, which will be discussed in Section 3.3, and they are interconnected by algebraic relation between $e^l_{i,j,i',j'}$ and $c_{i.,j}$ from (9).



(a) $\vec{\boldsymbol{n}}_R$ (f_R)



(b) $\vec{\boldsymbol{n}}_O$ (f_O)

Figure 3: Schematic diagram of the two functions: f_R and f_O

3.3. Loss Function and Progressive Training

We are utilizing neural networks to minimize the energy functional of a given system. Directly training the network to learn a mapping from an initial state to the minimum energy state of the system poses challenges. To address this, we adopt a progressive approach that reduces the energy in multiple steps. This involves formulating a training loss function that is given by:

$$\mathcal{L} = \left[\frac{1}{N_x N_y} \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} (\phi_{i,j}^{out} - \phi_{i,j}^{in})^2 \right] + \omega \frac{E^{out}}{E^{in}}, \tag{11}$$

where ω is a constant, and E^{out} and E^{in} are the energies of the output and input, respectively. The computation of these energies on the regular grid of the domain involves the utilization of the finite difference method [30] to compute the necessary derivatives and the Riemann sum [31] to approximate the definite integrals.

The loss function consists of two distinct terms. The first term is the mean squared difference between the input and the output, which constrains the output to remain close to the input. Meanwhile, the subsequent term is the ratio of the output energy E^{out} to the input energy E^{in}. Since the input energy E^{in} remains constant, the loss function guides the network to learn a mapping that keeps the output close to the input while achieving a lower energy state.

Initially, we train the networks f_R and f_O to learn the mapping of the initial state of the system ($\phi^0 \rightarrow \phi^0$) using the mean squared error. During this initial training, the Gaussian noise (X) with mean (μ) = 0 and a small standard deviation (σ) is added to the initial state at each iteration, enhancing the robustness of the initial mapping. Once the initial training is complete, the networks are fine-tuned for a few iterations (K) with the given loss function $\mathcal{L}$ (11). This process helps the networks to learn a mapping that not only keeps the output near the input but also achieves a lower energy state. With this fine-tuned network we obtain an updated state of the system. Using this new state, the network is again fine-tuned to get a subsequent state of the system. We continue this iterative process of energy reduction until the termination criterion is met. In contrast to conventional optimization techniques, the FONN method eliminates the need to specify a step size for energy functional minimization. With the fine-tuning steps, the networks reduce energy without explicitly specifying the step size. Algorithm 3 shows the steps involved in progressively decreasing the energy.

Algorithm 3 The FONN Method

Require: $\phi^0_{i,j}, \quad i = 1, 2, \dots N_x, \quad j = 1, 2, \dots N_y$

Given $n = 0, M, K, \omega, \sigma$
for step $= 1, \dots,$ initial training steps **do**
  $X_{i,j} \sim \mathcal{N}(0, \sigma^2), \quad \forall i, j$
  $\phi^{in}_{i,j} \leftarrow \phi^0_{i,j} + X_{i,j}, \quad \forall i, j$
  Use Algorithm 2 with $\phi^{in}_{i,j}$ to obtain $\phi^{out}_{i,j}$
  MSE $\leftarrow \frac{1}{N_x N_y} \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} (\phi^{out}_{i,j} - \phi^{in}_{i,j})^2$
  Train the networks f_R and f_O for MSE
end for
Approximate E^0 for $\phi^0_{i,j}$
while $E^n > tolerance$ **do**
  $\phi^{in}_{i,j} \leftarrow \phi^n_{i,j}, \quad \forall i, j$
  Approximate E^{in} from $\phi^{in}_{i,j}$
  for $k = 1, \dots, K$ **do**
    Use Algorithm 2 with $\phi^{in}_{i,j}$ to obtain $\phi^{out}_{i,j}$
    Approximate E^{out} for $\phi^{out}_{i,j}$
    $\mathcal{L} \leftarrow \left[\frac{1}{N_x N_y} \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} (\phi^{out}_{i,j} - \phi^{in}_{i,j})^2 \right] + \omega \frac{E^{out}}{E^{in}}$
    Fine-tune f_R and f_O for $\mathcal{L}$
  end for
  Use Algorithm 2 with $\phi^n_{i,j}$ to obtain $\phi^{n+1}_{i,j}$
  $n \leftarrow n + 1$
end while

4. Numerical Experiments

In this section, we illustrate the capabilities of the proposed novel FONN algorithm by testing it with a different set of problems. For all the tests (if not specified), we used two hidden layers for both f_R and f_O, i.e., $\vec{\boldsymbol{n}}_R = (4, 32, 32, 1)$ and $\vec{\boldsymbol{n}}_O = (2, 32, 32, 1)$. For the initial training, a standard deviation of 0.05 was considered for the Gaussian noise, and the networks were trained for 1000 steps. We applied the ReLU activation function to the hidden layers and tanh for the output neurons for both networks. The networks were initialized using the Xavier initialization method [32]. We used TensorFlow [33] to implement the model and Adam [34] to optimize the networks. All codes accompanying this manuscript are available on GitHub at https://github.com/Sanjeeb-Poudel/FONN.git

4.1. Heat energy

The steady-state heat equation is represented as follows:

$$-\nabla^2\phi = 0 \quad \text{in } \Omega, \tag{12}$$

where $\phi : \Omega \to \mathbb{R}$ is the scalar temperature. The equivalent form in the energy functional formulation is

$$E(\phi) := \int_\Omega \frac{1}{2}|\nabla\phi|^2 d\mathbf{x}, \tag{13}$$

and the minimizer of (13) will be the solution of (12).

In Algorithm 2, the functions f_R and f_O can be defined in various ways. For the heat energy, we show specific formulations for f_R and f_O that make the algorithm identical to the Forward Time Centered Space (FTCS) method. Using the gradient flow of the energy (13) gives the heat equation,

$$\frac{\partial\phi}{\partial t} = \nabla^2\phi. \tag{14}$$

Approximating the heat equation (14) using the FTCS, we have

$$\phi^{n+1} = \phi^n + \frac{\Delta t}{h^2}(\phi^n_{i+1,j} + \phi^n_{i-1,j} + \phi^n_{i,j+1} + \phi^n_{i,j-1} - 4\phi^n_{i,j}), \tag{15}$$

where Δt is the time step and h is the grid size. After reorganizing (15), we get

$$\begin{aligned}\phi^{n+1} = \phi^n - \frac{\Delta t}{h^2}[&(\phi^n_{i,j} - \phi^n_{i-1,j}) - (\phi^n_{i+1,j} - \phi^n_{i,j}) \\ &+ (\phi^n_{i,j} - \phi^n_{i,j-1}) - (\phi^n_{i,j+1} - \phi^n_{i,j})].\end{aligned} \tag{16}$$

In Algorithm 2, taking $M = 1$, and comparing (16), we get

$$f_R(\phi^n_{i,j}, \phi^n_{i',j'}, 0, 0) = \phi^n_{i,j} - \phi^n_{i',j'} \tag{17}$$

and

$$f_O(\phi^n_{i,j}, c_{i,j}) = \phi^n_{i,j} - \frac{\Delta t}{h^2}c_{i,j}. \tag{18}$$

The above representations for f_R and f_O are linear, and the method is effective only under the condition that the stability criterion ($\Delta t/h^2 \leq 1/4$) is satisfied. However, in our implementation, we use fully connected neural

networks to model f_R and f_O. By fine-tuning steps, our method dynamically adjusts the functions, eliminating the need for the selection of an optimal step size in minimization.

The steady-state solution of the heat equation (12) is independent of the initial condition of the domain and only depends on the boundary condition. We used a random temperature distribution across the domain as the initial condition. On the other hand, the boundary conditions that we used for our experiment are given as follows:

$$\begin{aligned} \phi(0, y) &= \sin(5\pi y/2), \\ \phi(1, y) &= -4(y - 1/2)^2 + 1, \\ \phi(x, 1) &= -x + 1, \\ \phi(x, 0) &= 0. \end{aligned} \tag{19}$$

The energy functional (13) includes the squared magnitude of the gradient of the temperature. In computing $|\nabla\phi|^2$, we took the average of the first-order forward and backward difference schemes, mitigating the possibility of checkerboard distribution of ϕ caused by the centered difference scheme. For the experiment, we discretized the domain with $N_x = 101$ and $N_y = 101$ grid points and used a single message-passing step (i.e., $M = 1$).

To illustrate the advantage of the progressive method of energy reduction, we initially trained the networks using the energy functional (13) itself as the loss function. This training approach optimizes the weights and biases of the networks to minimize the energy functional at the output. While the training effectively decreases the energy of the output, as shown in Fig. 4, the resulting state has a higher energy compared to the reference solution, which is computed using the FTCS method. Moreover, after training the networks, the output state of the domain is significantly different from the reference solution as shown in Fig. 5. The networks learned to make the gradient of the temperature small, so the temperature distribution inside the domain is near constant. However, the large gradients at the boundary correspond to a higher energy state. The mean absolute difference between the reference solution and the output of the network is 0.1908.

Subsequently, we minimized the energy functional given in (13) using the FONN method. For the fine-tuning, we specified the number of steps (K) to 10 and the weight (ω) of the loss function (11) to 0.1. Specifying $K = 10$ implies that for each energy reduction step (n), there are 10 fine-tuning steps.

Fig. 6 shows the training loss for the first ten steps ($n < 10$), where

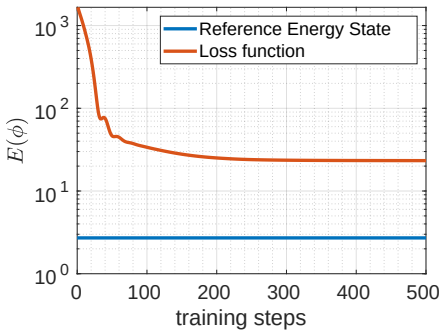
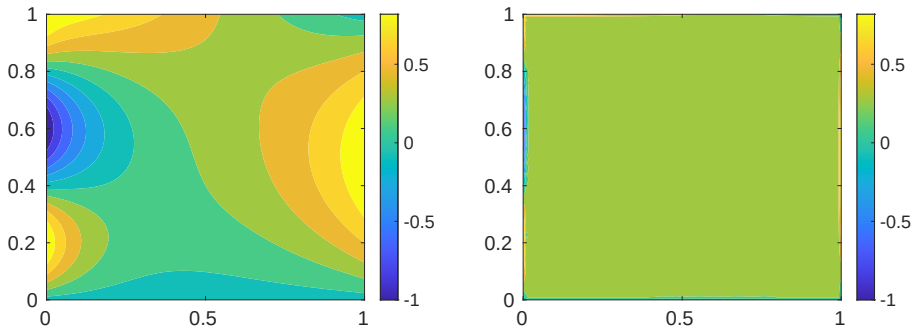


Figure 4: Loss function (energy functional) at different training steps.



(a) Reference Solution

(b) Networks' output

Figure 5: The solution of the heat equation obtained using the energy functional itself as the loss function and comparing it with the reference solution.

the vertical red lines separate individual steps (n). Note that each step (n) comprises 10 fine-tuning substeps. The figure shows decrease in the loss function (11) within each step (n). However, between the steps, there is a sudden increase (from $n = 0$ to $n = 1$, there is a decrease) in the loss, indicating an update in the system's state. As depicted in the figure, the loss function converges to $\omega = 0.1$. This convergence reflects the system approaching the minimum energy state, where the system undergoes minimal changes, resulting in the loss function value close to 0.1.

On the other hand, Fig. 7 shows the energy of the system at different steps (n) of the FONN method. The figure illustrates a rapid decrease in the system's energy, ultimately converging towards the reference minimum energy. Moreover, the minimum energy state of the system is close to the

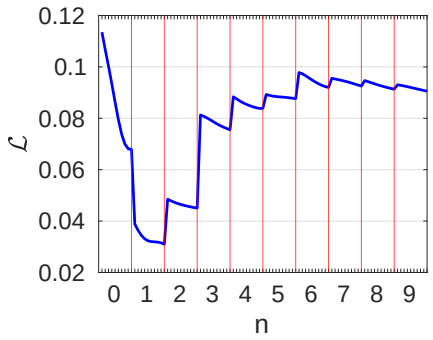


Figure 6: The loss function ($\mathcal{L}$) for the first ten fine-tuning steps ($n < 10$).

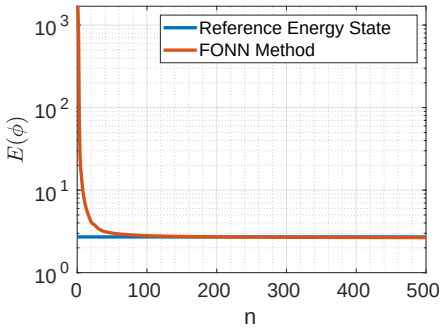


Figure 7: Comparing energy output at different steps of the FONN method with the reference minimum energy state.

reference solution as demonstrated in Fig. 8. The mean absolute difference between the reference solution and the minimum energy state obtained from the method is 0.0342.

Additionally, we examined the functions f_R and f_O that were learned by the FONN method at different steps. Fig. 9 shows the functions at the initial and the final steps of the method. The function f_R is plotted for various values of $\phi_{i,j}$ and $\phi_{i',j'}$, whereas the function f_O is plotted for various values of $\phi_{i,j}$ and the total interacting effect $c_{i,j}$. As expected, the initial and the final functions are different. The fine-tuning step updates the functions to make them suitable for the given state of the system.

The FONN method learn to mitigate the total interacting effect $c_{i,j}$ at the minimum energy state. Fig. 10 shows the total effect across the domain at the initial and final step. At the initial step, the total interacting effect looks

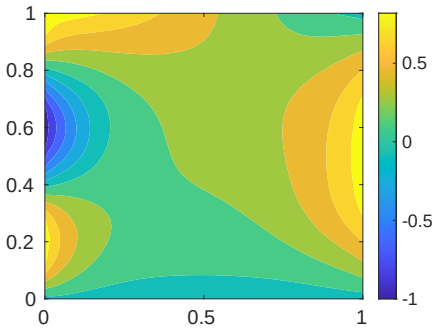
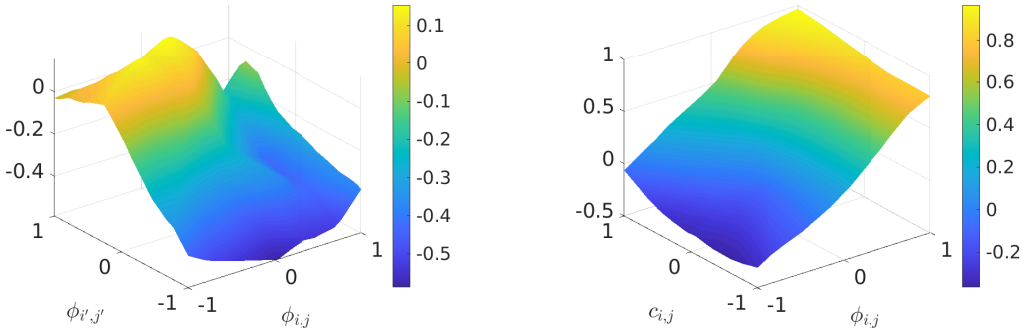


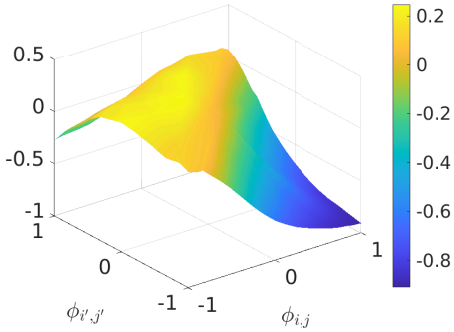
Figure 8: The solution of the heat equation using the FONN method.

noisy because of the random initialization of the domain. However, at the final iteration, the total effect is small throughout the domain. Moreover, for $c_{i,j} = 0$, the output of the function f_O is almost linear (Fig. 11) with a slope of 1, which means that the output of $f_O \sim \phi_{i,j}$. In other words, through fine-tuning, the networks demonstrate the ability to nearly neutralize the interacting effect while maintaining an output similar to the input at the minimum energy state.

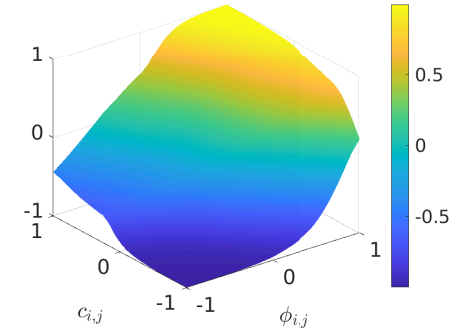


(a) f_R at $n = 0$

(b) f_O at $n = 0$

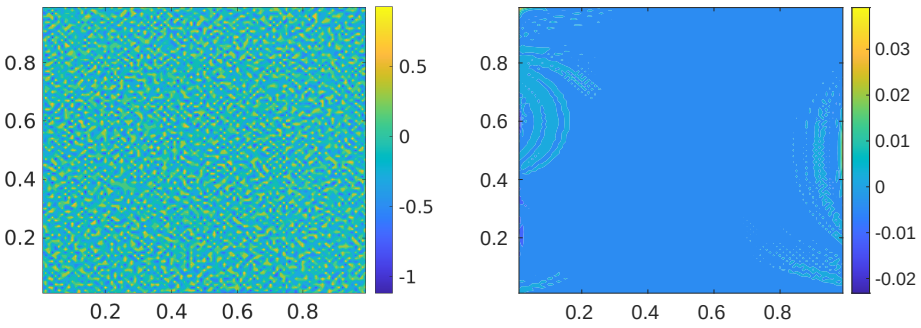


(c) f_R at $n = 500$



(d) f_O at $n = 500$

Figure 9: The functions f_R and f_O learned by the FONN method at the initial and the final step.



(a) $c_{i,j}$ at $n = 0$

(b) $c_{i,j}$ at $n = 500$

Figure 10: The total interacting effect ($c_{i,j}$) across the grid for the initial and final step in the FONN method.

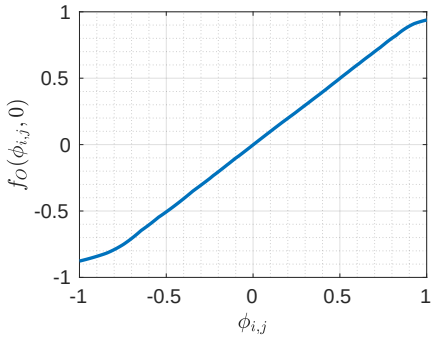


Figure 11: Output of f_O for different $\phi_{i,j}$ at $c_{i,j} = 0$.

4.2. Lyapunov energy

In this section, we address a problem characterized by multiple local minima in the energy functional. It is common for traditional methods to become trapped in one of the local minima depending on the initial conditions. Here, we consider the Lyapunov energy functional [1] that is given by

$$E(\phi) = \int_{\Omega} \left(\frac{1}{2} |\nabla \phi|^2 + \frac{1}{\epsilon^2} F(\phi) \right) d\mathbf{x}, \tag{20}$$

which is often employed for simulating multi-phase problems. The above energy has two components. The first term is the squared magnitude of the gradient of a scalar function $\phi : \Omega \to \mathbb{R}$ on the domain Ω. The second component $F(\phi)$ is the energy potential, and ϵ is a small positive constant. The gradient flow of this energy gives the Allen-Cahn equation. In the context, $F(\phi)$ is the Ginzburg-Landau double-well potential, $F(\phi) = \frac{1}{4}(\phi^2 - 1)^2$, and the parameter ϵ is related to the interface thickness. Here we set $\epsilon = 0.02$.

The computational domain $\Omega \in [0, 1]^2$ was discretized with $h = 0.005$, $N_x = N_y = 201$, and the initial condition for the domain was set as $\phi(x, y) = \sin(4\pi x) cos(4\pi y)$. We used the second-order finite difference scheme for approximating the gradient of the scalar function ϕ with a periodic boundary condition. We observe that when implementing a periodic boundary condition in the simulation, two global minima of the energy functional exist. These minima correspond to constant scalar functions, denoted as ϕ, with values of either +1 or -1 throughout the domain, resulting in a 0 energy value. These are considered as trivial solutions. In this section, we investigate the efficacy of the FONN method for such problems and explore a method to attain non-trivial energy states.

For the FONN method, we used two message-passing steps (i.e., $M = 2$), and similar to the previous section, the initial training had 1000 steps, and the fine-tuning steps (K) was 10.

Initially, we conducted multiple simulations, varying the weight parameter ($\omega = 0.01, 0.1, 0.2,$ and 0.5). Fig. 12 shows the energy plots at different steps of the FONN method for each ω. The figure illustrates that there is an initial drop in energy, followed by a period of relative stability before a rapid decrease in energy to zero. The weight parameter (ω) influences the steps taken by the method to get to the trivial solution because it determines the contribution of the output energy to the loss function. A higher value of ω requires fewer steps to reach a zero energy state, whereas a lower ω takes more steps.

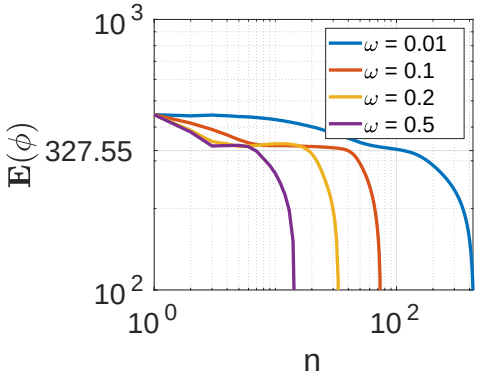


Figure 12: Energy plots for various ω

After the initial drop of the energy $E(\phi)$, the energy exhibits a slow descent as the system's state approaches the local minimum (see Fig. 12). The local minimum of the system was determined to be $E(\phi) \approx 327.55$ using the standard gradient descent method as a reference. Extending the simulation steps leads to a further reduction in energy until it eventually reaches the global minimum. This observation highlights the ability of the FONN method to avoid getting trapped in a local minimum. The iterative fine-tuning of the networks for each energy reduction step allows them to retain the previous momentum, which might help to escape the local minimum. Notably, in all experiments, the method consistently converges to the global minimum. Consequently, early stopping of the simulation is necessary to get non-trivial energy states.

In all experiments, the FONN method effectively reduces the energy of the system, ultimately reaching the global minimum. However, due to the stochastic nature inherent in the neural networks' initialization and initial training, the method's path to the global minimum might be different. For $\omega = 0.1$, we ran ten simulations using the same parameters. Fig. 13 illustrates the mean path and the range of different paths taken by the method to reach the global minimum. Initially, the range is small, but as the iteration progresses, the range increases.

Next, Fig. 14 shows the side-by-side comparison of the states using the standard gradient method and the FONN method. For the comparison, we made contour plots of a state close to the local minimum. Specifically, we selected the state closest to the local minimum among the mean states from the preceding ten simulations for the FONN method. Simultaneously, the

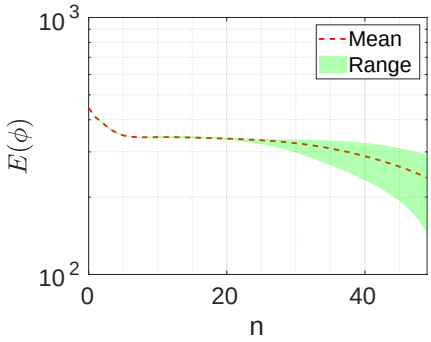
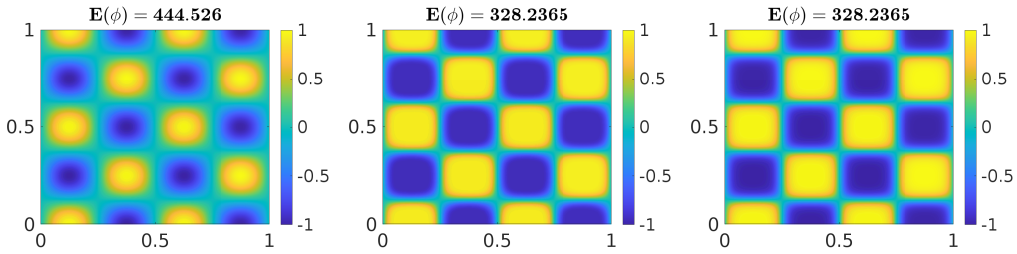


Figure 13: Multiple simulations for $\omega = 0.1$

standard gradient descent method was employed to derive the state corresponding to the same energy level. The figure shows that the FONN method is close to the standard gradient method, and the mean absolute difference between the results is 0.0296.



(a) Initial condition (b) FONN method (c) Standard gradient method

Figure 14: Comparison of an energy state obtained using the FONN method with the corresponding energy state using the standard gradient descent method.

4.3. Elastic Bending Energy

In this example, we study a constrained minimization problem to explore the performance of the method in addressing constraints. We consider the modified elastic bending energy based on a phase-field function given by

$$E(\phi) = \int_\Omega \frac{\epsilon}{2}|\Delta\phi - \frac{1}{\epsilon^2}(\phi^2 - 1)\phi|^2 d\mathbf{x}, \tag{21}$$

where $\phi : \Omega \to \mathbb{R}$ on the domain Ω is the scalar phase-field function and ϵ is a small constant characterizing the width of the transition layer. The

elastic bending energy is widely used in the modeling of vesicle conformation changes [8, 35, 36, 37].

Our objective in this context is to minimize the energy (21) while incorporating an area constraint. The interior area of the vesicle membrane is calculated using the following expression:

$$A(\phi) = \int_{\Omega} \frac{1}{2}(\phi + 1)\, d\mathbf{x}. \tag{22}$$

To enforce the constraint in our method, we employ the penalty method with a penalty coefficient M_1. The problem with the penalty coefficient M_1 can be represented as:

$$\min_{\phi} \; E_M(\phi) = E(\phi) + M_1(A(\phi) - A_0)^2, \tag{23}$$

where A_0 is the prescribed area.

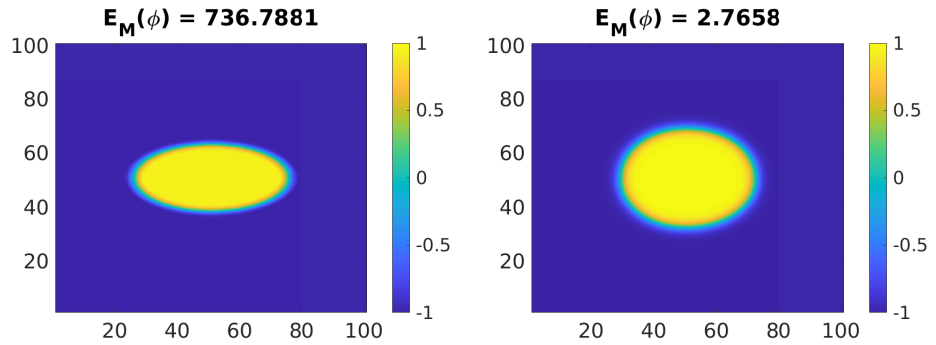
The domain was given as the range $\Omega = [-\pi, \pi]^2$, discretized with $N_x = N_y = 101$, where $h = 2\pi/100$, and $\epsilon = 1.5h$. We used two message passing steps ($M = 2$), trained the initial condition for 1000 steps, fixed $\omega = 0.1$. We solved the problem with the constant $A_0 = 5$, and $M_1 = 100$. For the boundary condition, all boundary values of ϕ were fixed to -1, i.e., $\phi_{1,j} = \phi_{N_x,j} = \phi_{i,1} = \phi_{i,N_y} = -1$. The variable ψ in Algorithm 2 was fixed to 0 at the boundary for all message-passing steps. The phase-field function ϕ was initialized as:

$$\phi(x, y) = \tanh\left(\frac{1 - (2x/\pi)^2 - (4y/\pi)^2}{\sqrt{2}\epsilon}\right). \tag{24}$$

Fig. 15 is a visual representation, offering a side-by-side comparison of the vesicle membrane's initial and final configurations. Notably, the final configuration exhibits an increase in membrane size due to the imposed area constraint. Fig. 16a shows the system's energy throughout the minimization steps. The consistent decrease in the energy underscores the method's effectiveness in optimizing the system's configuration. In the experiments, we noticed that running the simulation longer may lead to an increase in energy and potential entry into a noisy state. Thus, early stopping might be necessary in this situation.

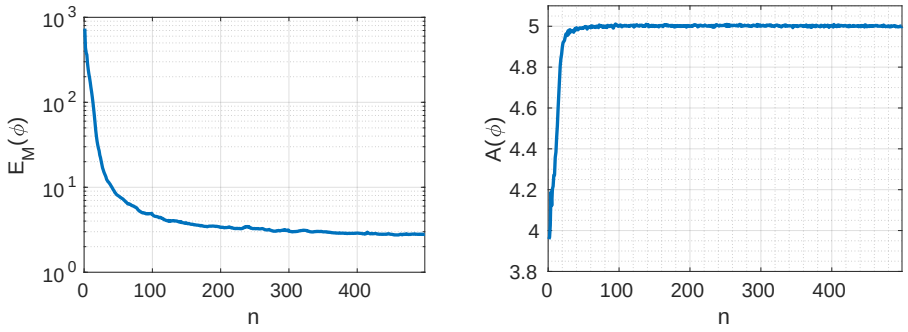
Moreover, the method's ability to uphold the constraint is demonstrated in Fig. 16b. The figure shows that, initially, the member has a lower area

value. Due to the imposed constraint, the membrane's area increases to the specified value, and throughout the rest of the simulation, the membrane's area remains constant. This graph highlights the robustness of the method in fulfilling the given constraint.



(a) Initial condition (b) Final configuration

Figure 15: Solution of the two-dimensional elastic bending energy using the FONN method



(a) Energy (b) Inside area

Figure 16: The energy $E_M(\phi)$ and the inside area of the membrane $A(\phi)$ at different steps of the FONN method

5. Conclusion

In this paper, we developed a novel FONN method to minimize energy functional using neural networks. The proposed method is described with a network architecture to work on a regular grid and a loss function based on the energy functional. In addition, a progressive energy reduction technique to decrease energy in multiple steps is established. Several numerical experiments for a wide variety of problems demonstrated the ability of the FONN method for energy minimization. For example, we employed the proposed FONN method to compute the steady state distribution of temperature in a domain and the minimum state of the Lyapunov energy. We also confirmed the method's effectiveness in minimizing the elastic bending energy with an area constraint.

The notable advantage of the FONN method is that it eliminates to need to specify a step size for the minimization problem, unlike traditional energy minimization methods. Moreover, with an example of Lyapunov energy, we demonstrated that the method avoids getting trapped in a local minimum. Although we only considered two-dimensional problems, the method can be easily used for higher or lower dimensions.

Acknowledgements

The authors would like to thank Prof. Gordon Erlebacher for his valuable feedback. This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] J. Shen, X. Yang, Numerical approximations of allen-cahn and cahn-hilliard equations 28 (4) 1669–1691, publisher: Discrete and Continuous Dynamical Systems. doi:10.3934/dcds.2010.28.1669.
URL https://www.aimsciences.org/en/article/doi/10.3934/dcds.2010.28.1669

[2] C. Liu, J. Shen, A phase field model for the mixture of two incompressible fluids and its approximation by a fourier-spectral method 179 (3) 211–228. doi:10.1016/S0167-2789(03)00030-7.
URL https://www.sciencedirect.com/science/article/pii/S0167278903000307

[3] X. Yang, J. J. Feng, C. Liu, J. Shen, Numerical simulations of jet pinching-off and drop formation using an energetic variational phase-field method 218 (1) 417–428. doi:10.1016/j.jcp.2006.02.021.
URL https://www.sciencedirect.com/science/article/pii/S002199910600091X

[4] J. Shen, X. Yang, A phase-field model and its numerical approximation for two-phase incompressible flows with different densities and viscosities 32 (3) 1159–1179. doi:10.1137/09075860X.
URL http://epubs.siam.org/doi/10.1137/09075860X

[5] J. Kim, Phase-field models for multi-component fluid flows 12 (3) 613–661. doi:10.4208/cicp.301110.040811a.
URL https://www.cambridge.org/core/product/identifier/S1815240600003169/type/journal_article

[6] S. Lee, M. F. Wheeler, T. Wick, Pressure and fluid-driven fracture propagation in porous media using an adaptive finite element phase field model, Computer Methods in Applied Mechanics and Engineering 305 (2016) 111–132.

[7] W. Helfrich, Elastic properties of lipid bilayers: Theory and possible experiments 28 (11) 693–703, publisher: De Gruyter. doi:10.1515/znc-1973-11-1209.
URL https://www.degruyter.com/document/doi/10.1515/znc-1973-11-1209/html

[8] Q. Du, C. Liu, X. Wang, A phase field approach in the numerical study of the elastic bending energy for vesicle membranes 198 (2) 450–468. doi:10.1016/j.jcp.2004.01.029.
URL https://www.sciencedirect.com/science/article/pii/S0021999104000373

[9] Q. Du, C. Liu, X. Wang, Simulating the deformation of vesicle membranes under elastic bending energy in three dimensions 212 (2) 757–777. doi:10.1016/j.jcp.2005.07.020.
URL https://www.sciencedirect.com/science/article/pii/S0021999105003566

[10] X. Wang, Q. Du, Modelling and simulations of multi-component lipid membranes and open membranes via diffuse interface approaches 56 (3) 347–371. doi:10.1007/s00285-007-0118-2.
URL https://doi.org/10.1007/s00285-007-0118-2

[11] Q. Du, C. Liu, R. Ryham, X. Wang, Energetic variational approaches in modeling vesicle and fluid interactions 238 (9) 923–930. doi:10.1016/j.physd.2009.02.015.
URL https://www.sciencedirect.com/science/article/pii/S0167278909000669

[12] X. Wang, L. Ju, Q. Du, Efficient and stable exponential time differencing runge–kutta methods for phase field elastic bending energy models 316 21–38. doi:10.1016/j.jcp.2016.04.004.
URL https://www.sciencedirect.com/science/article/pii/S0021999116300365

[13] R. Gu, X. Wang, M. Gunzburger, Simulating vesicle–substrate adhesion using two phase field functions 275 626–641. doi:10.1016/j.jcp.2014.07.010.
URL https://www.sciencedirect.com/science/article/pii/S0021999114004902

[14] Q. Du, L. Zhu, Analysis of a mixed finite element method for a phase field bending elasticity model of vesicle membrane deformation 24 (3) 265–280, publisher: Institute of Computational Mathematics and Scientific/Engineering Computing.
URL https://www.jstor.org/stable/43693290

[15] X. Feng, H.-j. Wu, A posteriori error estimates and an adaptive finite element method for the allen–cahn equation and the mean curvature flow 24 (2) 121–146. doi:10.1007/s10915-004-4610-1.
URL https://doi.org/10.1007/s10915-004-4610-1

[16] M. F. Wheeler, T. Wick, S. Lee, Ipacs: Integrated phase-field advanced crack propagation simulator. an adaptive, parallel, physics-based-discretization phase-field framework for fracture propagation in porous media, Computer Methods in Applied Mechanics and Engineering 367 (2020) 113124.

[17] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators 2 (5) 359–366. doi:10.1016/0893-6080(89)90020-8.
URL https://www.sciencedirect.com/science/article/pii/0893608089900208

[18] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, B. Kingsbury, Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups 29 (6) 82–97, conference Name: IEEE Signal Processing Magazine. doi:10.1109/MSP.2012.2205597.

[19] A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet classification with deep convolutional neural networks 60 (6) 84–90. doi:10.1145/3065386.
URL https://dl.acm.org/doi/10.1145/3065386

[20] N. Baker, F. Alexander, T. Bremer, A. Hagberg, Y. Kevrekidis, H. Najm, M. Parashar, A. Patra, J. Sethian, S. Wild, K. Willcox, S. Lee, Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence. doi:10.2172/1478744.
URL https://www.osti.gov/biblio/1478744

[21] Z. Mao, A. D. Jagtap, G. E. Karniadakis, Physics-informed neural networks for high-speed flows 360 112789. doi:10.1016/j.cma.2019.112789.
URL https://www.sciencedirect.com/science/article/pii/S0045782519306814

[22] X. Jin, S. Cai, H. Li, G. E. Karniadakis, NSFnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations 426 109951. doi:10.1016/j.jcp.2020.109951.
URL https://www.sciencedirect.com/science/article/pii/S0021999120307257

[23] S. Lee, Y. Shin, On the training and generalization of deep operator networks, arXiv preprint arXiv:2309.01020 (2023).

[24] E. Haghighat, M. Raissi, A. Moure, H. Gomez, R. Juanes, A physics-informed deep learning framework for inversion and surrogate modeling

in solid mechanics 379 113741. doi:10.1016/j.cma.2021.113741.
URL https://www.sciencedirect.com/science/article/pii/S0045782521000773

[25] J. Bai, T. Rabczuk, A. Gupta, L. Alzubaidi, Y. Gu, A physics-informed neural network technique based on a modified loss function for computational 2d and 3d solid mechanics 71 (3) 543–562. doi:10.1007/s00466-022-02252-0.
URL https://doi.org/10.1007/s00466-022-02252-0

[26] S. Cai, Z. Wang, C. Chryssostomidis, G. E. Karniadakis, Heat transfer prediction with unknown thermal boundary conditions using physics-informed neural networks, American Society of Mechanical Engineers Digital Collection. doi:10.1115/FEDSM2020-20159.
URL https://dx.doi.org/10.1115/FEDSM2020-20159

[27] S. Cai, Z. Wang, S. Wang, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks for heat transfer problems 143 (60801). doi:10.1115/1.4050542.
URL https://doi.org/10.1115/1.4050542

[28] P. W. Battaglia, R. Pascanu, M. Lai, D. Rezende, K. Kavukcuoglu, Interaction networks for learning about objects, relations and physics. arXiv:1612.00222[cs].
URL http://arxiv.org/abs/1612.00222

[29] Y. Li, J. Wu, J.-Y. Zhu, J. B. Tenenbaum, A. Torralba, R. Tedrake, Propagation networks for model-based control under partial observation, in: 2019 International Conference on Robotics and Automation (ICRA), pp. 1205–1211, ISSN: 2577-087X. doi:10.1109/ICRA.2019.8793509.

[30] G. D. Smith, G. D. Smith, Numerical solution of partial differential equations: finite difference methods, Oxford university press.

[31] V. Sealey, Definite integrals, riemann sums, and area under a curve: What is necessary and sufficient, in: Proceedings of the 28th annual meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education, Vol. 2, Universidad Pedagógica Nacional Mérida, México, pp. 46–53, issue: 1991.

[32] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, pp. 249–256, ISSN: 1938-7228.
URL https://proceedings.mlr.press/v9/glorot10a.html

[33] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous distributed systems. arXiv:1603.04467[cs], doi:10.48550/arXiv.1603.04467.
URL http://arxiv.org/abs/1603.04467

[34] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization. arXiv:1412.6980[cs], doi:10.48550/arXiv.1412.6980.
URL http://arxiv.org/abs/1412.6980

[35] Q. Du, C. Liu, X. Wang, Retrieving topological information for phase field models 65 (6) 1913–1932. doi:10.1137/040606417.
URL http://epubs.siam.org/doi/10.1137/040606417

[36] Q. Du, C. Liu, R. Ryham, X. Wang, Modeling the spontaneous curvature effects in static cell membrane deformations by a phase field formulation 4 (3) 537–548, publisher: Communications on Pure and Applied Analysis. doi:10.3934/cpaa.2005.4.537.
URL https://www.aimsciences.org/en/article/doi/10.3934/cpaa.2005.4.537

[37] Q. Du, C. Liu, R. Ryham, X. Wang, A phase field formulation of the willmore problem 18 (3) 1249–1267, publisher: IOP Publishing. doi:10.1088/0951-7715/18/3/016.
URL https://doi.org/10.1088/0951-7715/18/3/016