Machine learning materials physics: Integrable deep neural networks enable scale bridging by learning free energy functions

G.H. Teichert\textsuperscript{a}, A.R. Natarajan\textsuperscript{c}, A. Van der Ven\textsuperscript{c}, K. Garikipati\textsuperscript{a,b,d,*}

\textsuperscript{a}Department of Mechanical Engineering, University of Michigan
\textsuperscript{b}Department of Mathematics, University of Michigan
\textsuperscript{c}Materials Department, University of California, Santa Barbara
\textsuperscript{d}Michigan Institute for Computational Discovery & Engineering, University of Michigan

Abstract
The free energy of a system is central to many material models. Although free energy data is not generally found directly, its derivatives can be observed or calculated. In this work, we present an Integrable Deep Neural Network (IDNN) that can be trained to derivative data obtained from atomic scale models and statistical mechanics, then analytically integrated to recover an accurate representation of the free energy. The IDNN is demonstrated by training to the chemical potential data of a binary alloy with B2 ordering. The resulting DNN representation of the free energy is used in a mesoscopic, phase field simulation and found to predict the appropriate formation of antiphase boundaries in the material. In contrast, a B-spline representation of the same data failed to resolve the physics of the system with sufficient fidelity to resolve the antiphase boundaries. Since the fine scale physics harbors complexity that emerges through the free energy in coarser-grained descriptions, the IDNN represents a framework for scale bridging in materials systems.

Keywords: Deep Neural Networks, Chemical potential, Phase field, Multiscale physics

1. Introduction
An accurate description of the free energy plays a critical role in many physics-based models of materials systems. The Euler-Lagrange equations of stationary problems are obtained by requiring that the first variational derivative of the free energy functional vanish. In the example of elasticity, this leads to the equilibrium equation satisfied by the displacement field \cite{1}. Evolution

*Corresponding Author
Email address: krishna@umich.edu (K. Garikipati)
equations can also require first variations of the free energy as inputs. For example, the variational derivatives of the free energy with respect to composition and order parameters define the chemical potential fields used in phase field dynamics \[2, 3, 4\]. A related result is that the first derivative of the free energy density function with respect to appropriate strain measures gives the conjugate stress. Second derivatives with respect to the temperature yield the heat capacity, those with respect to strains define elasticities, and mixed second derivatives with respect to strain and temperature lead to the thermal expansivities \[5\].

High-fidelity evaluations of free energy density functions are attainable by a combination of atomic scale models (including quantum mechanics and various molecular models) and statistical mechanics. The use of such free energy density functions in the partial differential equations of continuum physics at coarse-grained scales, such as in phase field models, other mass and heat transport equations, and the equations of nonlinear elasticity is a rigorous, if classical approach to scale bridging. In principle, if highly accurate free energy density functions were available, quantitative predictivity would become accessible in a specialized but significant spectrum of mechano-chemically driven phenomena, such as phase transformations, nonlinear and strain gradient elasticity.

Implementing these computations, however, can be challenging for a number of reasons. Free energy data is often computed at individual points rather than as an analytic function directly. As such, it becomes necessary to use a fitting process to create a faithful mathematical model of the free energy. The free energy density can be a function of multiple variables, including composition, temperature, strain, and order parameters, thus leading to a high-dimensional function \[6, 7\]. Additionally, realistic data for the free energy can contain regions with rapid fluctuations, along with other regions with very gradual slopes \[8\]. Finally, because of the importance of the derivatives of the free energy, it is desirable that the fitting function be smooth. All of these considerations pose challenges to the fitting technique.

Machine learning methods are readily applicable to this problem, and we specifically consider Deep Neural Networks (DNNs) \[9\]. DNNs have been successfully applied to problems with large numbers of inputs, such as RGB pixel values in an image \[10\], the log-power spectra of speech data \[11\], the configurational energy of multicomponent alloys \[12\], predicting precipitate morphologies \[13\] and biomarkers for predicting human age \[14\], among numerous other applications. Because the activation functions used in DNNs are generally global functions with one local feature, DNNs are capable of capturing local phenomena without negatively affecting longer range attributes of the data. Also, with the proper choice of activation functions, DNNs are infinitely differentiable, thus allowing all necessary derivatives to be computed.

One potential disadvantage of DNNs is that they are not, in general, analytically integrable. This is a particular challenge in the case of fitting free energy data, because the free energy is often not directly measured or computed. Instead, the derivatives of the free energy (i.e. the chemical potentials) are first observed or computed, then integrated to find the free energy of the system \[15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 8\]. Such an approach is of partic-
ular importance in cases where the chemical potential representation must be integrated with respect to chemical variables to obtain the free energy density, whose derivatives with respect to strain then yield the stress for elasticity [6]. In order to preserve as much information about the derivatives as possible, it is ideal to train directly to the derivative data itself, as opposed to a numerically integrated data set. This requires an alternative form of DNN that can be trained to derivative data, then analytically integrated to represent the free energy itself. We present such an approach in this work.

It is possible to reduce the complexity required of the DNN by incorporating certain terms that are known beforehand. For example, the ideal solution energy contains logarithmic terms that cause the chemical potentials to diverge at certain locations in the composition-order parameter space. Rather than requiring the DNN to learn this behavior, it can be imposed by incorporating the ideal solution free energy terms themselves into the final expression of the free energy.

In this first presentation of our proposed framework, we consider the problem of chemistry, postponing coupled problems for later communications. To demonstrate the method of training a DNN to derivative data of the free energy, we consider a simple binary substitutional alloy with B2 ordering on a BCC parent structure [24]. The resulting free energy, as a function of composition and an order parameter, is then used in the Cahn-Hilliard and Allen-Cahn phase field equations.

In Section 2, we present the form of an Integrable Deep Neural Network (IDNN). Section 3 describes the B2 material system and the method of calculating the chemical potential data via a combination of atomic scale and statistical mechanics computations. In this context, the IDNN training to the chemical potential data and the resulting free energy DNN are shown in Section 3.1. We describe the phase field formulation in Section 3.2 and present the computational results obtained using the DNN representation of the free energy. Section 4 describes a process for fitting the chemical potential data using B-splines and compares the resulting fit and phase field simulation with those of the DNN. Concluding discussions are presented in Section 5.

2. Integrable Deep Neural Network

For a fully connected DNN, the following equations are commonly used to define the activation value of unit i in layer \( \ell \), denoted here by \( a^\ell_i \):

\[
a^\ell_i = f(z^\ell_i) \tag{1}
\]

\[
z^\ell_i = b^\ell_i + \sum_{j=1}^{m_{\ell-1}} W^\ell_{ij} a^\ell_{j-1} \tag{2}
\]

where \( b^\ell_i \) is the bias, \( W^\ell_{ij} \) is the weight, \( m_{\ell-1} \) is the number of units in layer \( \ell - 1 \), and \( f(\cdot) \) is the activation function (see Figure 1). In a DNN used for
regression, the output $Y_i$ is computed using the activation units from the final layer without an activation function:

$$Y_i = b_{i,n+1}^{n+1} + \sum_{j=1}^{m_n} W_{i,j}^{n+1} a_j^n$$  \hspace{1cm} (3)

The DNN can be thought of as a function of inputs $x$, weights $W$, and biases $b$, i.e. $Y = Y(x, W, b)$. Training the DNN consists of optimizing the weights and biases to minimize the loss function for a given dataset $\{(\hat{x}_\theta, \hat{Y}_\theta)\}_{\theta=1}^{N_{\text{data}}}$. The loss function is generally the mean square error (MSE) for regression problems. We can represent this as follows:

$$\hat{W}, \hat{b} = \arg \min_{W, b} \text{MSE} \left( Y(x, W, b) \bigg| \hat{x}_\theta, \hat{Y}_\theta \right)$$ \hspace{1cm} (4)

where the sum over $\theta = 1, \ldots N_{\text{data}}$ is implied in the definition of the MSE.

As argued in the Introduction, it can be desirable to train a function with data describing the derivatives. While it is possible to train a standard DNN directly to the derivative data, there are at least two drawbacks. The first is that the standard DNN cannot, in general, be analytically integrated to recover the originating function. Furthermore, in the multidimensional case with multiple sets of partial derivative data, the trained standard DNNs representing the partial derivatives would not necessarily be themselves the derivatives of the same function. This inconsistency poses a problem even in situations where mathematical representations of only the partial derivatives are needed and not the integral itself.

Both of these difficulties can be overcome by differentiating the standard DNN with respect to the desired input variables, say $x_k$ with $k = 1, \ldots, n$, then
training these differentiated functions to the (partial) derivative data $\{(\hat{x}_\theta, \hat{y}_{k_\theta})\}$, as represented by the following:

$$\hat{W}, \hat{b} = \arg \min_{W, b} \sum_{k=1}^{n} \text{MSE} \left( \frac{\partial Y(x, W, b)}{\partial x_k} \bigg|_{\hat{x}_\theta}, \hat{y}_{k_\theta} \right)$$  \hspace{1cm} (5)

The functional form that results from differentiating the standard DNN is, of course, analytically integrable. As such, it will be referred to in this work as an IDNN (Integrable Deep Neural Network). The antiderivative of the IDNN is simply a standard DNN with the weights and biases of the trained IDNN. The IDNN has the following form:

$$\frac{\partial a^{\ell}}{\partial x_k} = f'(z^{\ell}_i) \frac{\partial z^{\ell}_i}{\partial x_k}$$  \hspace{1cm} (6)

$$a^{\ell}_i = f(z^{\ell}_i)$$ \hspace{1cm} (7)

$$\frac{\partial z^{\ell}_i}{\partial x_k} = \sum_{j=1}^{m_{\ell-1}} W^{\ell}_{i,j} \frac{\partial a^{\ell-1}_j}{\partial x_k}$$ \hspace{1cm} (8)

where $z^{\ell}_i$ is given by Equation (2). In a slightly more abstracted form with two sets of activation units, $\alpha$ and $\beta_k$, we have:

$$\beta^{\ell}_{k_i} = f'(z^{\ell}_i) \sum_{j=1}^{m_{\ell-1}} W^{\ell}_{i,j} \beta^{\ell-1}_{k_j}$$ \hspace{1cm} (9)

$$a^{\ell}_i = f(z^{\ell}_i)$$ \hspace{1cm} (10)

The values of the trained derivative function $y_i,k := \partial Y_i / \partial x_k$ and its integral (within an integration constant) $Y_i$ are found as follows:

$$y_i,k = \sum_{j=1}^{m_{\ell-1}} W^{\ell}_{i,j} \beta^{n}_{k_j}$$ \hspace{1cm} (11)

$$Y_i = \sum_{j=1}^{m_{\ell-1}} W^{\ell}_{i,j} a^{n}_j$$ \hspace{1cm} (12)

Note that the following are used to compute the activation values for the first hidden layer:

$$\beta^{1}_{k_i} = f'(z^{1}_i) W^{1}_{i,k}$$ \hspace{1cm} (13)

$$z^{1}_i = b^{1}_i + \sum_{j=1}^{m_{1}} W^{1}_{i,j} x_j$$ \hspace{1cm} (14)

$$a^{1}_i = f(z^{1}_i)$$ \hspace{1cm} (15)

The activation function, $f(\cdot)$, can chosen such that its derivative, $f'(\cdot)$, is also a common activation function. For example, with the SoftPlus activation function, $f(x) := \ln(1 + e^x)$, the derivative, $f'(x)$ is the commonly used logistic function, i.e. $f'(x) = 1/(1 + e^{-x})$.  

5
We emphasize that the form of the IDNN in Eq. (6)–(8) has been chosen such that its integral has the form of a standard DNN, where both the IDNN and its integral use the same weights and biases, as is clear from the optimizations (5) and (4). Thus, once the weights and biases of an IDNN have been trained using (5), the analytically integrated DNN is constructed by simply using the IDNN’s weights and biases in a standard DNN given by (1) and (2).

2.1. Enforcing symmetries

Neural networks have the property of being uniform approximations of continuous functions over compact domains [25]. However, underlying symmetries of the domain are not guaranteed to be reproduced exactly by the DNN. For instance, consider a function, \( f(x, y) \), defined on variables \( x \) and \( y \), such that they are symmetric under the inversion operation. Given the symmetry of the domain of \( (x, y) \), the function is also symmetric under the same operation, i.e. \( f(x, y) = f(\pm x, \pm y) \). Any artificial neural network that approximates \( f \) must reproduce this symmetry exactly. This can be enforced on the DNN by first transforming the inputs to a set of symmetric functions. For example, within a neural network to approximate \( f \), rather than use the values of \((x, y)\) as input, the symmetric functions \( (x^2, y^2) \) can be used to parameterize the DNN. These functions map all symmetrically equivalent points in the \((x, y)\) space on to the same value, while also differentiating symmetrically distinguishable points. As another example consider functions \( f(x, y) \) that are invariant under an inversion about the \( y = 0 \) line. A DNN that approximates any function in this class can be guaranteed to obey the required symmetry by using \((x, y^2)\) as the input functions.

In general, a set of invariant inputs to the neural network may be defined by generating symmetry invariant polynomial functions with algorithms described by Thomas and Van der Ven [26]. We will denote these functions as \( h(\cdot) \). Since the given derivative data is likely differentiated with respect to, for example, \( y \) and not \( y^2 \), we must incorporate the symmetrized inputs, \( h(\cdot) \), into the loss function, e.g. \( \text{MSE} \left( \frac{\partial Y}{\partial x_1}, \hat{y} \right) \), where \( \hat{y} = \frac{\partial Y}{\partial x_1} \) is the derivative data.

3. Bridging atomic to continuum scales via an IDNN representation of the free energy density

The prototypical example of microstructure evolution in materials science is the ordering phase transformation from a bcc solid-solution to B2 order. This transformation usually involves not only an ordering reaction, but also the migration and interaction of anti-phase boundaries within the ordered phase. In technologically important alloys such as Fe-Al and Ni-Al, these types of order-disorder transformations can be exploited to improve material properties. We benchmark our techniques by generating thermodynamic data using a model Hamiltonian that exhibits this order-disorder phase transition.

Our model at the atomic scale is an attractive nearest neighbor pair cluster expansion on the bcc crystal structure was used to simulate the order-disorder
Figure 2: (a) Formation energies (solid points) of orderings on the bcc crystal structure calculated with an attractive nearest neighbor interaction. (b) Temperature-composition phase diagram calculated for the attractive nearest neighbor cluster expansion. Second order phase transitions are estimated from the divergence of the heat capacity.

Figure 3: Conventional cell of bcc showing two sublattices labeled 1 and 2.

phase transitions (Figure 2a). The ground state for this cluster expansion is the well known B2 phase [27]. The phase transition between the disordered and ordered bcc phase is second-order as shown in the phase diagram of Figure 2b.

Order parameters to distinguish the B2 phase from the disordered solid solution are well known and may be derived as a linear combination of sublattice compositions [3, 24]. The compositions, \( x_1 \) and \( x_2 \) of species B on the two sublattices of the conventional bcc cell shown in Figure 3 are used to define order parameters as:

\[
\begin{align*}
\eta &= \frac{x_1 - x_2}{\sqrt{2}} \\
\epsilon &= \frac{x_1 + x_2}{\sqrt{2}}
\end{align*}
\]  

The first order parameter, \( \eta \) corresponds to the homogeneous composition of the alloy, while \( \eta \) can be used to distinguish long-range B2-like order from a com-
pletely disordered phase. Throughout the composition space, the disordered solid solution corresponds to the loci of $\eta = 0$, while the maximum degree of ordering is the line connecting the origin to the point $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$. Unlike conventional phase field models where “perfect” ordering is the locii of points with $\eta = 1$ the definitions of the order parameter as linear combinations of sublattice compositions imposes particular limits on the domain of allowed values in the $(c, \eta)$ space.

Conventional grand-canonical Monte-Carlo techniques restrict the calculated free energy to only the thermodynamically stable regions. However, since the unstable parts of the free energy are critical to describing the temporal and spatial evolution of the system, we employed biased Monte-Carlo techniques to sample the free energy throughout the composition, order-parameter domain [24]. An ordering of A and B atoms in a crystal is denoted by the vector $\mathbf{\sigma}$ with components $\sigma_i = \pm 1$ depending on whether site $i$ is occupied by species A or B. An ordering of A and B atoms is referred to as a “configuration”. The biased ensemble is defined with the following partition function:

$$Z(\phi_c, \phi_\eta, \kappa_c, \kappa_\eta, T) = \sum_\sigma \exp\left(-\frac{E(\mathbf{\sigma}) + M\phi_c(c(\mathbf{\sigma}) - \kappa_c)^2 + M\phi_\eta(\eta(\mathbf{\sigma}) - \kappa_\eta)^2}{k_B T}\right)$$

(18)

where $E(\mathbf{\sigma})$ is the energy for the configuration given by $\mathbf{\sigma}$, $M$ is the number of unit cells, $c(\mathbf{\sigma})$ and $\eta(\mathbf{\sigma})$ are the homogeneous composition and the order parameter defined by Eq. (17) for the configuration. The quantities $\phi_c$, $\phi_\eta$, $\kappa_c$, and $\kappa_\eta$ define bias potentials, with curvatures are given by $\phi_c$ and $\phi_\eta$, and are centers are at $\kappa_c$ and $\kappa_\eta$, respectively. The biased ensemble is sampled with Metropolis Monte-Carlo, and statistical averages of the homogeneous composition $\langle c \rangle$, and order parameter $\langle \eta \rangle$ are measured for different values of $\phi$, $\kappa$, and $T$. The statistical averages can be related to the derivatives of the free energy per atom (denoted $g = G/M$, where $G$ is the total free energy) as [24]:

$$\mu_c = \frac{\partial g}{\partial c}\bigg|_{\langle c \rangle, \langle \eta \rangle} = -2\phi_c(\langle c \rangle - \kappa_c)$$

(19)

$$\mu_\eta = \frac{\partial g}{\partial \eta}\bigg|_{\langle c \rangle, \langle \eta \rangle} = -2\phi_\eta(\langle \eta \rangle - \kappa_\eta)$$

(20)

where $\mu_c$ and $\mu_\eta$ are the chemical potentials with respect to the composition and order parameter respectively. The cluster expansions, and statistical mechanics calculations were performed with the CASM code [28, 29, 30, 21]. The measured ensemble averages were then used to calculate free energy derivatives.

### 3.1. Training free energy DNN

The free energy of an alloy can be partitioned into ideal and non-ideal contributions. For the prototypical example of ordering on bcc, we consider the ideal solution free energy in terms of the sublattice compositions $x_1$ and $x_2$, as follows:

$$\hat{g}(x_1, x_2) = k_B T[x_1 \log x_1 + (1 - x_1) \log(1 - x_1) + x_2 \log x_2 + (1 - x_2) \log(1 - x_2)]$$

(21)
where \( k_B \) is Boltzmann’s constant and \( T \) is the temperature. The sublattice compositions can be written in terms of \( c \) and \( \eta \) with Equations (16) and (17). The ideal solution free energy diverges as the sub-lattice compositions \( x_1 \) and \( x_2 \) approach 0 and 1, as shown in Equation (21). The partial derivatives of the ideal solution free energy with respect to \( c \) and \( \eta \), then, have the following forms:

\[
\frac{\partial \hat{g}}{\partial c} = \frac{k_B T}{\sqrt{2}} \log \left( \frac{c^2 - \eta^2}{(\sqrt{2} - c)^2 - \eta^2} \right) \tag{22}
\]

\[
\frac{\partial \hat{g}}{\partial \eta} = \frac{k_B T}{\sqrt{2}} \log \left( \frac{(c + \eta)(\sqrt{2} - c + \eta)}{(c - \eta)(\sqrt{2} - c - \eta)} \right) \tag{23}
\]

Subtracting the right-hand side expressions in the partial derivatives, Equations (22) and (23), from the chemical potential data leaves the excess chemical potential. Training against the excess chemical potential allows the IDNN to resolve the non-divergent, and thus smoothly varying data, rather than the high gradient points as \( x_1 \) and \( x_2 \) approach 0 and 1, which pose challenges for DNNs in general. Upon training, the ideal solution contributions are added back to the IDNN for chemical potentials and free energy. Thus the divergence in the neighborhood of \{0,1\} is exactly represented, also.

An IDNN representing the excess chemical potential data, \( \Delta \mu_c \) and \( \Delta \mu_\eta \), was trained to the derivative data, i.e. the excess chemical potential values, as described in Section 2, such that the IDNN could be analytically integrated to recover the excess free energy. Data points within \( 0.01 < x_1, x_2 < 0.99 \), were used for training and cross-validation, with the logarithmic terms of the ideal solution energy exactly capturing the appropriate divergence. The IDNN was implemented as a custom Estimator using the TensorFlow library [31], and was defined by two hidden layers with 10 units per layer. The IDNN was trained for 700,000 epochs using the AdagradOptimizer, with learning rates of 0.1, 0.05, and 0.01 applied at different stages of training. A batch size of 10 was used, with 65,601 points in the training set and 21,867 points used for cross-validation. The resulting learning curve is plotted in Figure 4, showing the decrease of both the training and cross-validation mean square errors as training progressed. Symmetry with respect to the order parameter about \( \eta = 0 \) was enforced.

Since the IDNN has a more complex form than the standard DNN, it is reasonable to expect that it may require more training to achieve comparable errors. To demonstrate any differences in training, ten IDNNs and ten standard DNNs were trained to the same chemical potential data, with the same symmetry conditions imposed. All twenty neural networks consisted of two hidden layers of ten neurons, each with different initial values for the weights and biases. They were trained for 100,000 epochs with a learning rate of 0.1. The resulting learning curves appear in Figure 5. It is clear that the IDNN is more sensitive to the initial values of the weights and biases, resulting in a larger spread in MSE values. About half of the IDNNs had MSE values comparable to the standard DNNs, while the remaining five IDNNs had higher MSE values. However, the highest IDNN error at 100,000 epochs is of the same order of magnitude—only
Figure 4: Learning curve showing the decrease in mean square error for testing and cross-validation datasets over training epochs. The cross-validation error is nearly indistinguishable from the training error.

Figure 5: The training of a ten standard DNNs is compared with the training of ten IDNNs. Each neural network was trained for 100,000 epochs, with different initial values for the weights and biases.

a factor of $\sim 3.5\times$—of the highest MSE from a standard DNN. Thus, the added complexity of the IDNN does not significantly inhibit the training.

Figure 6 shows the original chemical potential data compared with the associated IDNN that was trained to the corresponding excess chemical potential data, and supplemented by the ideal solution terms. It also shows two views of the free energy surface as represented by the analytically integrated DNN. Perhaps the most significant feature of the free energy surface is the two energy wells, located at about $c = 1/\sqrt{2}$, $\eta = \pm 1/\sqrt{2}$. Given that the wells exist at the same composition, the material will not separate into multiple phases, but instead form anti-phase domains. This reflects the expected physics of the system, described at the beginning of Section 3.

3.2. Phase field computation

To demonstrate the use of the DNN representation of the free energy in computations, the analytically integrated free energy DNN was used in phase
field computations. The phase field model was based on the coupled Cahn-Hilliard and Allen-Cahn equations, solved using isogeometric analysis (IGA) \cite{32}. The simulation was performed using the mechanChemIGA code\footnote{Code available at github.com/mechanoChem/mechanoChem} which is based on the PetIGA \cite{33} and PETSc \cite{34, 35, 36} libraries, and run on the XSEDE Stampede2 HPC cluster \cite{37}.

3.2.1. Formulation

Given the homogeneous free energy density $g(c, \eta)$ as a function of concentration, $c$, and order parameter, $\eta$, we define the total free energy as the following:

$$\Pi[c, \eta] = \int_\Omega \left[ g(c, \eta) + \frac{1}{2} \chi_1 |\nabla c|^2 + \frac{1}{2} \chi_2 |\nabla \eta|^2 \right] \, dV$$ \hspace{1cm} (24)
where the homogeneous free energy can be additively decomposed into the ideal solution free energy and the excess free energy, as in
\[ g(c, η) = \hat{g}(c, η) + \Delta g(c, η), \]
respectively. The corresponding chemical potentials are given by the variational derivatives of the total free energy, namely
\[ \mu_c := \frac{\delta \Pi}{\delta c} \quad \text{and} \quad \mu_η := \frac{\delta \Pi}{\delta η}. \]
Using standard variational methods, and incorporating the known form of \( \hat{g}(c, η) \), results in the following equations for the chemical potentials:
\[ \mu_c = \frac{k_B T}{\sqrt{2}} \log \left( \frac{c^2 - η^2}{(\sqrt{2} - c)^2 - η^2} \right) + \frac{\partial \Delta g}{\partial c} - \chi_1 \nabla^2 c \quad (25) \]
\[ \mu_η = \frac{k_B T}{\sqrt{2}} \log \left( \frac{(c + η)(\sqrt{2} - c + η)}{(c - η)(\sqrt{2} - c - η)} \right) + \frac{\partial \Delta g}{\partial η} - \chi_2 \nabla^2 η \quad (26) \]
The phase field model consists of the Cahn-Hilliard \[2\] and Allen-Cahn \[3\] equations, given by the following, respectively:
\[ \frac{\partial c}{\partial t} = \nabla \cdot M \nabla \mu_c \quad (27) \]
\[ \frac{\partial η}{\partial t} = -L \mu_η \quad (28) \]
The Cahn-Hilliard equation is in conservation form, with the flux defined as \( J := -M \nabla \mu_c \). It models the overall composition of the system through \( c \), while conserving mass. The Allen-Cahn equation models the time evolution of the long-range ordering of the system through the non-conserved order parameter \( η \). The two equations are coupled through the chemical potentials being derived from the same free energy. We take the following conditions on the boundary \( \partial \Omega \):
\[ M \nabla \mu_c \cdot n = J_n \quad (29) \]
\[ \nabla η \cdot n = 0 \quad (30) \]
\[ \nabla c \cdot n = 0 \quad (31) \]
where the conditions in Equations \(30\) and \(31\) arise from enforcing equilibrium at the boundary.

The weak form of the equations, as solved by the IGA formulation, take the following form:
\[ 0 = \int_{Ω} \left[ w \frac{\partial c}{\partial t} + M (\nabla w \cdot \nabla g_c, \chi_1 \nabla^2 w \nabla^2 c) \right] \, dV \]
\[ - \int_{\partial Ω} \left[ w J_n + M \chi_1 (\nabla^2 c (\nabla w \cdot n) + \nabla^2 w (\nabla c \cdot n)) \right] \, dS \]
\[ + \int_{\partial Ω} \tau (\nabla w \cdot n) (\nabla c \cdot n) \, dS \]
\[ 0 = \int_{Ω} \left[ w \frac{\partial η}{\partial t} + L (w g_η, \chi_2 \nabla w \cdot \nabla η) \right] \, dV \quad (32) \]
where the higher order Dirichlet boundary condition in Equation \(31\) is applied using Nitsche’s method \[38, 39, 6\] in the last term of Equation \(32\).
3.2.2. Phase field results

We considered a two-dimensional domain, discretized by a $200 \times 200$ element mesh. The initial and boundary value problem was initialized with a uniform composition field of $c = 1/\sqrt{2}$ and an order parameter field randomly perturbed about $\eta = 0$, representing a material that had just been quenched from a higher temperature. Zero flux boundary conditions were applied. The initial time step was $\Delta t = 0.1$, with an adaptive time stepping scheme being applied to modify the time step based on the convergence of the nonlinear solver at each time step.

As shown in Figure 7, the initially disordered domain gradually forms regions of the two ordered states. The antiphase boundary has completely formed within 900 time steps. The antiphase domains take on order parameter values of $\pm 0.67$, while the composition field remains nearly uniform, with values within 3% of $1/\sqrt{2}$.

As the simulation progresses, the microstructure coarsens with curvature fluctuations from a straight antiphase boundary decreasing, as shown in Figure 8.
Figure 8: Curvature fluctuations away from straight antiphase boundaries decrease, and the domains coarsen with time. The chemical potentials are represented as IDNNs, integrated to yield the free energy.

4. Comparison with B-spline surface fit

For comparison with the IDNN’s representation properties of complex surfaces, we consider two-dimensional B-splines, motivated by their wide use in mathematics, and diverse applications in engineering [32]

4.1. Formulation

The equation for a B-spline surface \( Y(\xi_1, \xi_2) \), with knots on a tensor grid can be written in the following form

\[
Y(\xi_1, \xi_2) = \sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij} N_{i,p}(\xi_1) M_{j,p}(\xi_2)
\]

(34)

where \( \xi_1 \in [\xi_1^1, \xi_1^{m+p+1}] \), \( \xi_2 \in [\xi_2^1, \xi_2^{n+p+1}] \), and \( N_{i,p} \) is the B-spline basis function of order \( p \). The basis functions are defined by the Cox-de Boor recursion formula
\begin{align}
N_{i,p}(\xi_1) &= \frac{\xi_1 - \xi_1^i}{\xi_1^{i+p} - \xi_1^i} N_{i,p-1}(\xi_1) + \frac{\xi_1^{i+p+1} - \xi_1}{\xi_1^{i+p+1} - \xi_1^{i+1}} N_{i+1,p-1}(\xi_1) \\
N_{i,0}(\xi_1) &= \begin{cases} 
1 & \text{if } \xi_1^i \leq \xi_1 < \xi_1^{i+1} \\
0 & \text{otherwise}
\end{cases}
\end{align}

(35)

(36)

using the knot vector \( \Xi_1 = \{\xi_1^1, \xi_1^2, \ldots, \xi_1^{m+p+1}\} \). \( M_{j,p}(\xi_2) \) is similarly defined using the knot vector \( \Xi_2 = \{\xi_2^1, \xi_2^2, \ldots, \xi_2^{n+p+1}\} \).

To convert the matrix \( C \) to a vector (to use the standard form for least squares fitting), we use the following index conversion:

\[ I = n_i + j, \quad i = 0, \ldots, m - 1, \quad j = 0, \ldots, n - 1 \]

so that \( I = 0, \ldots, mn - 1 \). Then we can rewrite Eq. (34) as the following:

\[ Y(\xi_1, \xi_2) = \sum_{I=0}^{mn-1} c_I P_{I,p}(\xi_1, \xi_2) \]

(37)

where \( c_I := C_{ij} \) and \( P_{I,p}(\xi_1, \xi_2) := N_{i,p}(\xi_1) M_{j,p}(\xi_2) \). If evaluating multiple data points \( \{ (\hat{\xi}_1^k, \hat{\xi}_2^k) \} \), we can write the resulting vector of function evaluations using the following matrix-vector form, written in coordinate notation:

\[ Y_k = A_{kI} c_I \]

(38)

where

\[ A_{kI} = P_{I,p}(\hat{\xi}_1^k, \hat{\xi}_2^k) \]

(39)

We now consider fitting to two sets of derivative data. For derivative datasets contained in the two vectors \( \hat{\mu}_1 \) and \( \hat{\mu}_2 \), the following matrices are defined:

\[ B_{kI} = \frac{\partial}{\partial \xi_1} P_{I,p}(\hat{\xi}_1^k, \hat{\xi}_2^k) \]

(40)

\[ C_{kI} = \frac{\partial}{\partial \xi_2} P_{I,p}(\hat{\xi}_1^k, \hat{\xi}_2^k) \]

(41)

Then, we have the following least squares formulation, with some regularization added for numerical stability:

\[ \hat{c} = \arg \min_c \left[ (Bc - \hat{\mu}_1)^T (Bc - \hat{\mu}_1) + (Cc - \hat{\mu}_2)^T (Cc - \hat{\mu}_2) + \lambda c^T c \right] \]

(42)

where \( \lambda \) is the regularization coefficient. Setting the gradient with respect to \( c \) equal to the zero vector leads to the following least squares solution:

\[ \hat{c} = (B^T B + C^T C + \lambda I)^{-1} \left( B^T \hat{\mu}_1 + C^T \hat{\mu}_2 \right) \]

(43)

where \( I \) is the identity matrix. Equations (35, 41) and (43) are applied, with datasets in the \((c, \eta)\) space corresponding to the \((\xi_1, \xi_2)\) space for B-spline surfaces. Chemical potential data for \( \mu_c \) and \( \mu_\eta \) are contained in the vectors \( \hat{\mu}_1 \) and \( \hat{\mu}_2 \), respectively.
4.2. Selection of knots

While the method presented in the previous section can be used to optimize the values of the control points for given knot vectors, the locations of the knots also can be optimized to minimize the error. A variety of approaches are available, including nonlinear least squares, bisection, and genetic algorithms [42]. Our approach was performed in two steps.

First, we divided the data into training data (75%) and validation data (25%). As before, symmetry of the function about \( \eta = 0 \) was imposed. Unlike the DNN example, where using only data points within \( 0.01 < x_i < 0.99, \) \( i = 0, 1 \) improved the fit, we found that the B-spline fits had lower error when all data were retained. The optimal control points were found for a number of uniformly spaced knots, as well as knots placed at Chebyshev nodes, and the mean square error (MSE) using the validation data was reported, as shown in Figure 9. Higher numbers of knots were not used due to the resulting oscillatory behavior of the fit, particularly in regions of missing data. In comparing the resulting cross-validation error, it was found that nine uniformly distributed interior knots gave the best solution.

In the second step, we used Matlab’s genetic algorithm optimization routine to attempt to improve the B-spline fit using nine interior knots per knot vector (eighteen total variables). Appropriate inequality constraints were applied to maintain monotonically increasing knot vectors. The algorithm terminated after 57 generations, with a MSE of \( 1.312 \times 10^{-3} \). Since this was not an improvement over the error with uniformly spaced knots, the B-spline fit with nine uniformly spaced interior knots per knot vector was taken as the best fit using B-splines.

4.3. B-spline results, and comparison with DNN

The resulting B-spline representations of the chemical potentials and the free energy density are plotted in Figure 10. Visually, these fits seem very similar to those of the DNN in Figure 6.
However, the behavior of the phase field dynamics is dictated primarily by the value of $\frac{\partial^2 g}{\partial \eta^2}$ at $c = 1/\sqrt{2}$. These values are plotted in Figure 11 along with numerically differentiated data for comparison. These numerically differentiated data were obtained by first selecting data points within $c = 1/\sqrt{2} \pm 1.1 \times 10^{-4}$. These selected points were smoothed using Matlab’s smooth function with the rloess option, then numerically differentiated with respect to $\eta$ using a central difference scheme.

Antiphase segregation occurs where there is a negative value for $\frac{\partial^2 g}{\partial \eta^2}$, corresponding to a loss of convexity of the free energy density, $g$, with respect to $\eta$. In order for antiphase domains to occur in the phase field simulations, the initial conditions, which as in Section 3.2.2 consist of random perturbations about $\eta = 0$, must lie in a nonconvex region of the free energy. While this condition holds true for the DNN representation, it does not hold for the B-spline representation. In fact, the B-spline representation shows three convex regions (at $\eta \approx -0.3, 0, 0.3$) that do not exist in the data or in the DNN representation.
This results in a phase field solution that does not produce antiphase regions, but instead reaches an equilibrium solution with uniform values of $\eta = 0$ and $c = 1/\sqrt{2}$ (see Figure 12). It is possible, however, that other methods for knot selection might produce a better fit with B-splines.

The B-spline representation is more computationally efficient than the DNN. The phase field code evaluates the free energy and all first and second derivatives at each quadrature point. Using the de Boor algorithm for B-spline evaluation [41], the FLOP count for each function evaluation is about 1700. The highest order term in the count is $\sim 165p^2$, with polynomial order $p$. Significantly, the FLOP count for the B-spline evaluation does not depend on the size of the knot vector. The DNN representation, on the other hand, requires about 4500 FLOPs per evaluation of the DNN and its derivatives, using a naive implementation. The highest order term of the count is $\sim 18m^2n$, where $m$ is the neurons per layer and $n$ is the number of layers. However, an improved evaluation algorithm could potentially reduce the FLOP count. The use of accelerators could also speed up the function evaluations, without reducing the total FLOP count.

While the FLOP count for the free energy evaluation is more than 2.5 times greater for the DNN than the B-spline, this affects only the computation time for the assembly of the residual vector and tangent matrix. The wall time for the matrix-vector solution will be equivalent for the two methods, assuming both fitted functions are similar enough to give comparable condition numbers. For the DNN and B-spline fits presented here, the average computation time over the first 25 time steps using B-splines was 21.4 s, while it was 28.5 s using the DNN. The total computation time was, then, only about 1.3 times greater when using the DNN instead of the B-spline to represent the free energy. For larger problems, the matrix size increases and solver time comes to dominate the average computation time. In this limit the wall times will converge.
Figure 12: The lack of nonconvexity in the B-spline representation of the free energy at \( \eta = 0 \), \( c = 1/\sqrt{2} \) prevents the appearance of antiphase domains. The equilibrium solution in this case shows a uniform order parameter field of \( \eta = 0 \).

### 5. Conclusions

This communication adds to our nascent, but growing body of work in machine learning and artificial intelligence targeting higher fidelity models of materials physics [12, 13]. We have explored machine learning as an approach to bridging scales, by focusing on the representation of complexity emerging from fine scale physics. Here, the fine scales come from atomic and statistical mechanics descriptions, which can parameterize thermodynamic functions, such as free energy densities with high fidelity. At the core of this work is the idea of an analytically integrable Deep Neural Network (IDNN) to represent such functions. The IDNN is of particular use in the context of mathematically representing the free energy density of a material, where only the derivative data is originally known and the trained function must be integrated to recover the free energy. It is highly relevant to multidimensional systems, where multiple sets of partial derivative data must be trained against simultaneously under the constraint that all trained functions are the partial derivatives of one common function.

Using as a prototypical case a binary alloy, an IDNN was trained to two sets of chemical potential data, which were found using first principles calculations. Since these datasets are derivatives of the same free energy density, the analytic integrability of the IDNN, which exactly preserves consistency of representation, assumes importance. Symmetry with respect to the order parameter was embedded in the IDNN. Furthermore, recovery of the form of the ideal chemical potential, which is fundamental to statistical physics, also was guaranteed: This term was subtracted from the data, and the IDNN trained only against data representing the excess chemical potential. We anticipate an expansion of these approaches to embedding fundamental aspects of the physics into machine learning models as this field develops.

Phase field simulations with the analytically integrated DNN representing
the free energy recovered the proper physics of the system, showing the creation and subsequent coarsening of antiphase domains in the material. This example demonstrates the ability of the IDNN to capture the relevant physics of a material system. Interestingly, the IDNN was able to represent the physics of the system more faithfully than a B-spline representation, even in the current, relatively simple case, of a two-dimensional input. We note that in earlier work, we have demonstrated that B-spline representations themselves are superior to the more traditional Redlich-Kister polynomials at resolving rapidly varying thermodynamic functions [8]. The present work is a continuation of that thread, and establishes that even for only two-dimensional functions, the B-spline approach also may be inadequate. It is natural to expect that high-dimensional chemical potentials and free energy densities will present greater challenges.

Future work will consider systems of greater complexity, where the free energy density has such higher-dimensional dependence on variables that number $\sim O(10)$. In this regime, high fidelity representations of the free energy density will be crucial to reproducing the physics through scale bridging, and the uniform approximation property of DNNs, inherited by the IDNNs, will deliver greater advantages.

6. Acknowledgements

We gratefully acknowledge the support of Toyota Research Institute, Award #849910, “Computational framework for data-driven, predictive, multi-scale and multi-physics modeling of battery materials”. This work has also been supported in part by NSF DMREF grant #1729166, “Integrated Framework for Design of Alloy-Oxide Structures”. Simulations in this work were performed using the Extreme Science and Engineering Discovery Environment (XSEDE) Stampede2 at the Texas Advance Computing Center through allocations TG-MSS160003 and TG-DMR180072. XSEDE is supported by National Science Foundation grant number ACI-1548562.

References

[1] Jerrold E Marsden and Thomas JR Hughes. Mathematical foundations of elasticity. Courier Corporation, 1994.

[2] J. W. Cahn and J. E. Hilliard. Free energy of a nonuniform system. i interfacial energy. The Journal of Chemical Physics, 28:258–267, 1958.

[3] S. M. Allen and J. W. Cahn. A microscopic theory for antiphase boundary motion and its application to antiphase boundary coarsening. Acta Metallurgica, 27:1085–1091, 1979.

[4] Nikolas Provatas and Ken Elder. Phase-field methods in materials science and engineering. John Wiley & Sons, 2011.
[5] M. Hillert. *Phase Equilibria, Phase Diagrams and Phase Transformations*. Cambridge University Press, Cambridge, 2nd edition, 2007.

[6] S. Rudraraju, A. Van der Ven, and K. Garikipati. Mechano-chemical spinodal decomposition: A phenomenological theory of phase transformations in multi-component crystalline solids. *Nature Computational Materials*, 2:16012, 2016.

[7] K. Sagiyama, S. Rudraraju, and K. Garikipati. Unconditionally stable, second-order accurate schemes for solid state phase transformations driven by mechano-chemical spinodal decomposition. *Computer Methods in Applied Mechanics and Engineering*, 311:556–575, 2016.

[8] Gregory H. Teichert, N.S. Harsha Gunda, Shiva Rudraraju, Anirudh Raju Natarajan, Brian Puchala, Krishna Garikipati, and Anton Van der Ven. A comparison of redlich-kister polynomial and cubic spline representations of the chemical potential in phase field computations. *Computational Materials Science*, 128:127 – 139, 2017.

[9] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436 EP –, 05 2015.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[11] Yong Xu, Jun Du, Li-Rong Dai, and Chin-Hui Lee. A regression approach to speech enhancement based on deep neural networks. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 23(1):7–19, January 2015.

[12] Anirudh Raju Natarajan and Anton Van der Ven. Machine-learning the configurational energy of multicomponent crystalline solids. *npj Computational Materials*, 4(1), 2018.

[13] G. Teichert and K. Garikipati. Machine learning materials physics: Surrogate optimization and multi-fidelity algorithms predict precipitate morphology in an alternative to phase field dynamics. *Computer Methods in Applied Mechanics and Engineering*, 344:666–693, 2019.

[14] Evgeny Putin, Polina Mamoshina, Alexander Aliper, Mikhail Korzinkin, Alexey Moskalov, Alexey Kolosov, Alexander Ostrovskiy, Charles Cantor, Jan Vijg, and Alex Zhavoronkov. Deep biomarkers of human aging: Application of deep neural networks to biomarker development. *Aging*, 8(5):1021–1033, 05 2016.

[15] R. T. DeHoff. *Thermodynamics in Materials Science*. McGraw-Hill, 2006.
[16] J. M. Sanchez, F. Ducastelle, and D. Gratias. Generalized cluster description of multicomponent systems. *Physica A*, 128:334–350, 1984.

[17] D. De Fontaine. Cluster approach to order-disorder transformations in alloys. *volume 47 of Solid State Physics*, pages 33 – 176. Academic Press, 1994.

[18] A. Van der Ven, M. K., Aydinol, G. Ceder, G. Kresse, and J. Hafner. First-principles investigation of phase stability in Li$_2$CoO$_2$. *Phys. Rev. B*, 58:2975–2987, 1998.

[19] A. van de Walle and M. Asta. Self-driven lattice-model Monte Carlo simulations of alloy thermodynamic properties and phase diagrams. *Modelling and Simulation in Materials Science and Engineering*, 10:521–538, 2002.

[20] A. Van der Ven, J. C. Thomas, Q. Xu, and J. Bhattacharya. Linking the electronic structure of solids to their thermodynamic and kinetic properties. *Mathematics and Computers in Simulation*, 80(7):1393–1410, 2010.

[21] B. Puchala and A. Van der Ven. Thermodynamics of the Zr-O system from first-principles calculations. *Phys. Rev. B*, 88:094108, Sep 2013.

[22] M.-H. Chen, B. Puchala, and A. Van Der Ven. High-temperature stability of δ’-ZrO. *Calphad: Computer Coupling of Phase Diagrams and Thermochemistry*, 51:292–298, 2015.

[23] Anirudh Raju Natarajan, Ellen L.S. Solomon, Brian Puchala, Emmanuel A. Marquis, and Anton Van der Ven. On the early stages of precipitation in dilute Mg–Nd alloys. *Acta Materialia*, 108:367 – 379, 2016.

[24] Anirudh Raju Natarajan, John C. Thomas, Brian Puchala, and Anton Van der Ven. Symmetry-adapted order parameters and free energies for solids undergoing order-disorder phase transitions. *Phys. Rev. B*, 96:134204, Oct 2017.

[25] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[26] John C. Thomas and Anton Van der Ven. The exploration of nonlinear elasticity and its efficient parameterization for crystalline materials. *Journal of the Mechanics and Physics of Solids*, 107:76 – 95, October 2017.

[27] F. Ducastelle. *Order and Phase Stability in Alloys*. Cohesion and structure. Elsevier, 1991.

[28] https://github.com/prisms-center/CASMcode. CASM: A Clusters Approach to Statistical Mechanics, 2018.

[29] Anton Van der Ven, John C. Thomas, Brian Puchala, and Anirudh Raju Natarajan. First-principles statistical mechanics of mult-component crystals. *Annual Review of Materials Research*, 48:27–55, 2018.
[30] John C. Thomas and Anton Van der Ven. Finite-temperature properties of strongly anharmonic and mechanically unstable crystal phases from first principles. *Physical Review B*, 88(21):214111–214111, December 2013.

[31] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[32] J. Cottrell, T. J. R. Hughes, and Y. Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA*. Wiley, Chichester, 2009.

[33] L. Dalcin, N. Collier, P. Vignal, A.M.A. Córtes, and V.M. Calo. Petiga: A framework for high-performance isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 308:151–181, 2016.

[34] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Todd Munson, Karl Rupp, Patrick Sanan, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc Web page. [http://www.mcs.anl.gov/petsc](http://www.mcs.anl.gov/petsc), 2018.

[35] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Todd Munson, Karl Rupp, Patrick Sanan, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.10, Argonne National Laboratory, 2018.

[36] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[37] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Latham, D. Lifka, G. D. Peterson, R. Roskies, J. R. Scott, and N. Wilkins-Diehr. Xsede: Accelerating scientific discovery. *Computing in Science & Engineering*, 16(5):62–74, Sept.-Oct. 2014.
[38] J. A. Nitsche. Über ein Variationsprinzip zur Lösung von Dirichlet Problemen bei Verwendung von teilräumen, die keinen Randbedingungen unterworfen sind. Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg, 36:9–15, 1971.

[39] D. Arnold, F. Brezzi, B. Cockburn, and L. Marini. Unified analysis of discontinuous galerkin methods for elliptic problems. SIAM Journal on Numerical Analysis, 39(5):1749–1779, 2002.

[40] M. G. Cox. The numerical evaluation of B-splines. IMA Journal of Applied Mathematics, 10(2):134–149, 1972.

[41] Carl de Boor. On calculating with B-splines. Journal of Approximation Theory, 6(1):50 – 62, 1972.

[42] Van Than Dung and Tegoeh Tjahjowidodo. A direct method to solve optimal knots of B-spline curves: An application for non-uniform B-spline curves fitting. PLOS ONE, 12(3):1–24, 03 2017.