A GENERAL APPROACH FOR LEARNING CONSTITUTIVE RELATIONSHIPS (PHYSICAL LAWS) FROM NEURAL NETWORKS

ANDREW TEMPLE, AARON STEBNER, PETER COLLINS, AND BRANDEN KAPPES

1. Abstract

While it is possible to extract equations from machine learning strategies such as artificial neural networks that relate the independent variables (inputs) to a dependent variable (output), these equations do not naturally reflect known physical phenomena. Here, we present a general approach to learn and produce natural (and perhaps well-known) constitutive relationships (physical laws) from neural networks. The approach is presented in this paper as a comparison of the polynomial expansions of the optimized formula from neural networks and a learned/optimized physical relationship, whether or not known a priori. This linkage between the powerful infrastructure of neural networks and the fundamental physical phenomena and laws has many implications that will be introduced. General expansions for many physical phenomena are presented.

2. Introduction

With apologies to the Bard of Avon,

Two methods, both alike in dignity,
In fair sciences, where we lay our scene.
From ancient elegance break to methods mutinous,
Where machine learning makes physical laws unclean,
From forth the noble loins of these two foes
A pair of cross-term'd numbers seek their place;
Whose serial expansions, equally covariant
Do with their projections, bury their parents strife.
Our fearful assessment of their dual-space truth,
And the continuance of their parents, sage,
Which, for their children's union, sought to achieve,
Is now the two hours' traffic of our stage;
The which, if you with patent eyes attend,
What here shall miss, our toil shall strive to mend.

With the proliferation of increasingly powerful machine learning strategies, and their promise of the automatic discovery of new physics, phenomena, properties, and correlations, it is seductive to trust machine learning to solve our most complex problems. The unintended consequence of relinquishing one's data to machine learning is that our native physical understanding of currently-understood phenomenon may atrophy, and our development of descriptions of new physics may stagnate. These risks could have profound impacts on numerous scientific fields. Consider the instruction of machine learning, where students, and hence future scientists, engineers and educators, may, unintentionally, understand machine learning to be a cudgel against which every problem which has a solution falls. When one tool is so powerful as to render other methods seemingly obsolete, it is natural to "disarm" oneself of the tools that seem obsolete. Yet, such tools are not obsolete. Various physical phenomenon are exceptionally well described by laws (the laws of Thermodynamics, kinetics, motion, ...) or theories (e.g., dislocations which underpin our understanding of strength in metals and alloys). Those laws and theories permit us to creatively explore spaces where data does not exist, hypothesize, theorize, and test our hypotheses. In the absence a mastery of our understanding of physical processes, the scientific method in its currently and widely accepted form, becomes harder to apply. Thus, we must: retain mastery of current theory; postulate logically new theories; and leverage mathematical powers to understand complex hyperspaces describing n-variables.

The problem of whether to apply our most powerful tools in our analytical arsenal to a problem, or rigorously develop theories or calculate previously unknown physical constants seems to become an either-or problem. Much like the houses of the Montagues and Capulets in Shakespeare, the tools of Machine Learning and Phenomenological Relationships seem irrenconcilable. Yet, the problem lies with neither approach (both alike in dignity), rather it stems from two very human limitations. Firstly, we have great difficulty interpreting problems in higher-order space. It is easy to understand problems which have a single independent variable against which a phenomena depends, resulting in y-x plots of data. It is not much harder to understand problems which have two independent variables against which a phenomena depends, as these can be represented in Euclidean space, and with imagination we can understand problems with three independent variables (say x,y and time) against which a phenomena (say, z=f(xy,t)) depends. Beyond this, we lack tools to understand or visualize an n-variable hyperspace. Secondly, we can develop an implicit bias when wrestling with mathematical expressions, and may begin to believe that one (and hence, only one) functional expression can be used to fit a relationship. We develop mathematical "favorites", tools that, quite rationally, we turn to when solving certain problems. We turn to these expressions for two reasons: they work, and they are reduced to the simplest form possible. In materials science, against which the framework presented in this paper has been conceived and for which a simple discipline-specific problem will be tackled, it is common to turn to the Arrhenius relationship for phenomena which have temperature dependencies. This simple form, $A = A_O \exp -\frac{Q}{kT}$, is taught in extensively in a wide variety of classes, and yet, there is nothing in nature that dictates the exponential form be used. It is used because it is the simplest mathematical expression to fit the observed data. And yet, it could be expanded, in, e.g., polynomial form...

If this recognition that our most trusted expressions can be re-written in polynomial form, then so too must the reduced functions describing the most flexible and powerful neural networks out there. The next logical step is to postulate that these expanded forms of different representations of the same physical phenomena can be compared, and assessed for their self-similarity, providing a *translation* between one form and another. A Rosetta Stone, if you will, where the mathematical functions (i.e., the language) in one reference frame can be translated and assessed for equivalency to the mathematical functions of the second reference frame[1].

The precursors to this thought process began when H.L. Fraser et al. began conducting so-called virtual experiments, which permitted a well-trained neural network to be probed to determine the influence of one variable on a physical property while all of the other potential independent variables were artificially held at their average values [refs]. These virtual experiments were slices through an n-dimensional hyperspace, and it became obvious that the lower dimension slices could be described using simpler functions than the full neural network. This early work was built upon the initial efforts by H.K.D.H. Bhadeshia to solve complex materials science problems using artificial neural networks [refs] and the work of D.J.C. MacKay [refs] to incorporate Bayesian statistics and feedback loops. Following this prior work, with the assumption that a mapping was possible between the basis vectors of the n-variable hyperspace equally represented by the expansion of terms of an artificial neural network and the basis vectors representing the physical processes, Ghamarian and Collins, seeking to establish a constitutive equation for the room temperature yield strength of a titanium alloy given variations in its compositional and microstructural states, applied a hybrid artificial neural network-genetic algorithm method that optimized the unknowns in a postulated physically-based equation, testing the optimized physically-based model against slices through hyperspace function representing the neural network model and the data [refs]. This latter effort was quite inefficient, but resulted in a solution that proved, in subsequent work, to be generalized for multiple different variations of processing and with compositional variations [refs].

This previous work strongly suggests that the hypothesis we propose below is valid, and that a more fulsome mathematical treatment is merited and which is the subject of this paper. We recognize that this approach lies neither fully in a mathematics space nor in an

[1]It is critically important to recognize that while we are describing the translation between two reference frames, that of Machine Learning and that of Constitutive Laws of certain physical phenomena, there is no limitation to the number of reference frames (languages) among and between which the relationships can be established...which provides a way for different disciplines, describing the same phenonmena using different expressions, a way to translate their work - EXPAND THIS IN CONCLUSIONS, ... different data, equipment, labs, ...

engineering/science space. Consequently, we aim to provide both sufficiency of the derivations and sufficiency of motivation and impact, so that the paper is accessible to readers of various communities. While we demonstrate the approach on a materials science problem, we hope that the general applicability will be apparent, as it is easy to imagine this approach impacting a diverse range of disciplines, including: genetics, public health, biological sciences, earth sciences, information sciences (including signals analysis), physical sciences, applied variants thereof (medicine, environmental activities, engineering), space sciences, and economics. We finally include an appendix that includes the expansions of the activation functions and the generating functions of common functions that form the basis of some relevant physics. We recognize that this appendix is far from complete, but hope that the logic presented will permit those interested in identifying and developing additional functions as the specific applications demand.

The hypothesis: We hypothesize that the physics of an arbitrary and complex process can be extracted by fitting the polynomial expansions of known/postulated/potential physical relationships to the polynomial coefficients of a polynomial series expansion of an arbitrary and complex neural network.

NOTE: DISCUSS IN TEXT - if any physical constants are known, they can be directly entered into the expansions...

3. Methods

Fully dense neural network (NN) architectures, such as the one shown in Figure 1, perform a sequence of affine transformations, $\mathbf{z}_i \leftarrow \boldsymbol{\theta}_i \mathbf{x}^{(i)}$, followed by element-wise functional operations, $\sigma(\mathbf{z}_i)$ to introduce non-linearity at each layer; that is, each layer stretches and distorts the underlying space.

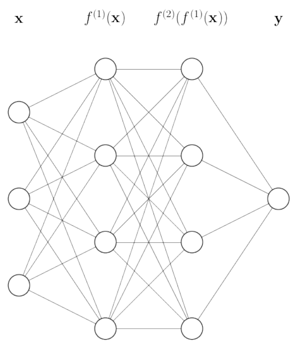

Figure 1. Schematic view of a fully dense neural network. Each sequence of affine and non-linear transformations are captured in the function, $f_i(\mathbf{x}) : \mathbf{x}^{(i+1)} \leftarrow \sigma(\boldsymbol{\theta}_i \mathbf{x}^{(i)})$

The resulting network,

$$f(x) = \sigma(\boldsymbol{\theta}_n \sigma(\boldsymbol{\theta}_{n-1} \sigma(\dots \boldsymbol{\theta}_2 \sigma(\boldsymbol{\theta}_1 \mathbf{x})))) \tag{1}$$

is an arbitrary function generator, but at present, the network weights $\boldsymbol{\theta}_i$ can not map back to analytic forms that capture and describe the underlying physics. There are, however, many such mappings through polynomial series expansions,

$$f(x) = \sum_{n=0}^{\infty} a_n x^n \tag{2}$$

The variable $\mathbf{x}$ in Equation (1) represents a column vector containing $\boldsymbol{n}$ input elements, and thus, the output from each node is multplied by the respective weight before the application of the subsequent activation function, e.g.

$$\mathbf{z} = \theta^{(1)} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \Rightarrow \begin{bmatrix} \theta_{11}x_1 + \theta_{12}x_2 + \dots \theta_{1n}x_n \\ \theta_{21}x_1 + \theta_{22}x_2 + \dots \theta_{2n}x_n \\ \vdots \\ \theta_{m1}x_1 + \theta_{m2}x_2 + \dots \theta_{mn}x_n \end{bmatrix} \tag{3}$$

and $\boldsymbol{\sigma}(\mathbf{z})$ serves as the input of the next layer in the network, where $\mathbf{z} = (\boldsymbol{\theta}^{(1)}\mathbf{x})$.

We hypothesize that the physics of a process can be extracted by fitting the polynomial expansions of known physical relationships to the polynomial coefficients of a polynomial series expansion of Equation (1).

3.1. **Softplus Activation.**

For example, consider the Softmax activation function which is given in Equation (???)

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum\limits_{j=1}^{k} e^{z_j}} \tag{4}$$

The goal is to obtain a activation generating function for the chosen activation function in the neural network.

The partion function, $\mathbf{Z}$, is defined below as

$$\mathbf{Z} = \sum_{j=1}^{k} e^{z_j} \tag{5}$$

The series representation for the exponential is given by the following

$$e^{z_i} = \sum_{k=0}^{\infty} \frac{1}{k!} z_i^k \tag{6}$$

Therefore

$$\sigma(\mathbf{z})_i = \sum_{k=0}^{\infty} \alpha_k z_i^k \tag{7}$$

Where $\alpha_k = \frac{1}{\mathbf{Z}k!}$ is the generating function for the Softmax

3.2. **ReLU Activation.** The ReLU (rectified linear units) function is another commonly used activation function and is given below in Equation (8):

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

The generating function for the ReLU is given in Equation (9) and it is dependent on both the input variables and the network weights.

$$\alpha_n = \begin{cases} 1 & \mathbf{z} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Similarly, the linear generating function is given in Equation (10) but it is only dependent upon **n**.

$$\alpha_n = \begin{cases} 1 & \mathbf{n} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Although ReLU (rectified linear units) have become a more common activation function, its discontinuity at $x = 0$ requires an infinite series to fully capture the behavior at this transition.

However, the softplus function,

$$
\begin{aligned}
f(x) &= log(1+e^x) \\
&= \frac{1}{\alpha} log(1+e^{\alpha x}), \quad \alpha > 0 \\
&= \frac{1}{\alpha} log(z), \quad z = 1+e^{\alpha x} \\
&= \frac{1}{\alpha}\sum_{k=1}^{\infty}\frac{1}{k}\left(\frac{z-1}{z}\right)^k \\
&= \frac{1}{\alpha}\sum_{k=1}^{\infty}\frac{1}{k}(z-1)^k z^{-k} \\
&= \frac{1}{\alpha}\sum_{k=1}^{\infty}\frac{1}{k}e^{\alpha x k}(1+e^{\alpha x})^k \\
&= \frac{1}{\alpha}\sum_{k=1}^{\infty}\frac{1}{k}\left(\sum_{m=0}^{\infty}\frac{\alpha^m x^m}{m!}\right)^k\left(1+\sum_{n=0}^{k}\frac{\alpha^n x^n}{n!}\right)^{-k}
\end{aligned}
$$

From Gradshteyn and Ryzhik - 0.314 Power series raised to powers.

$$
\left(\sum_{k=0}^{\infty} a_k x^k\right)^n = \sum_{k=0}^{\infty} c_k x^k \tag{11}
$$

where

$$
c_0 = a_0^n, \quad c_m = \frac{1}{m a_0}\sum_{k=0}^{\infty} c_k x^k
$$

And therefore it can be seen that

$$
c_0 = \frac{\alpha^0}{0!} = 1, \quad c_l = \frac{1}{l}\sum_{j=1}^{l}(jk-l+j)\frac{\alpha^l}{l!}c_{l-j}
$$

$$
= \frac{1}{\alpha}\sum_{k=1}^{\infty}\frac{1}{k}\sum_{m=0}^{\infty} c_m x^m \left(1+\sum_{n=0}^{\infty}\frac{\alpha^n x^n}{n!}\right)^{-k}
$$

From Gradshteyn and Ryzhik - 1.1 Power of Binomials, 1.111 Power Series

$$
(a+x)^n = \sum_{k=0}^{n}\binom{n}{k} x^k a^{n-k} \tag{12}
$$

And similarly

$$= \frac{1}{\alpha}\sum_{k=1}^{\infty}\frac{1}{k}\sum_{m=0}^{\infty}c_m x^m\left[\sum_{i=0}^{k}\binom{k}{i}\left(\sum_{n=0}^{\infty}\frac{\alpha^n x^n}{n!}\right)^i\right]^{-1}$$

Once again the substitution for a power series raised to a power is applied.

$$d_0 = 1, \quad d_l = \frac{1}{l}\sum_{j=1}^{l}(ji - l + j)\frac{\alpha^l}{l!}d_{l-j}$$

$$= \frac{1}{\alpha}\sum_{k=1}^{\infty}\frac{1}{k}\left[\frac{\sum\limits_{m=0}^{\infty} c_m x^m}{\sum\limits_{i=0}^{k}\frac{1}{\binom{k}{i}}\sum\limits_{n=0}^{\infty} d_n x^n}\right]$$

After some rearranging of term, the result is given below.

$$= \frac{1}{\alpha}\sum_{k=1}^{\infty}\frac{1}{k}\left[\frac{\sum\limits_{m=0}^{\infty} c_m x^m}{\sum\limits_{n=0}^{\infty}\sum\limits_{i=0}^{k}\frac{1}{\binom{k}{i}} d_n x^n}\right]$$

And therefore the following is obtained.

$$= \frac{1}{\alpha}\sum_{k=1}^{\infty}\frac{1}{k}\left[\frac{\sum\limits_{m=0}^{\infty} c_m x^m}{\sum\limits_{n=0}^{\infty} d'_n x^n}\right]$$

Where

$$d'_n = \sum_{i=0}^{k}\frac{1}{\binom{k}{i}}d_n$$

From Gradshteyn and Ryzhik - 0.313 Division of power series.

$$\frac{\sum\limits_{k=0}^{\infty} b_k x^k}{\sum\limits_{k=0}^{\infty} a_k x^k} = \frac{1}{a_0}\sum_{k=0}^{\infty} c_k x^k \tag{13}$$

where

$$c_n + \frac{1}{a_0}\sum_{k=1}^{n} c_{n-k}a_k - b_n = 0 \tag{14}$$

And therefore

$$b_m = \sum_{j=1}^{m} c_m - b_{m-j}d_j$$

$$= \frac{1}{\alpha}\sum_{k=1}^{\infty}\frac{1}{k}\sum_{n=0}^{\infty} b_n x^n$$
$$= \sum_{n=0}^{\infty}\sum_{k=1}^{\infty}\frac{b_n}{\alpha k}x^n = \frac{1}{\alpha}log(1+e^{\alpha x})$$

where $\sum_{k=1}^{\infty}\frac{b_n}{\alpha k}$ is the generating function for the softplus

3.3. **Iterative Determination of ANN Series Expansion Coefficients.** Any series expansion description of a deep neural network, which is necessarily multilayered, requires a explicit generating function for the coefficients of each layer; a generating function that is a function only of the coefficients of the previous layer and the coefficient generating function of the series expansion of the current layer's activation function.

A derivation for the polynomial expansion coefficient generating function for vector-valued function (layer 1 to layer 2) is presented below, which is an extension of a scalar expansion, provided in **??**.

3.4. **Polynomial Expansion of a Scalar Layer.** We can represent the recursive structure of an ANN given by Equation (1) as a series of transformations on a particular power series. Using this method, we are able to write an algorithm that will compute the coefficients of a power series determined by some ANN up to an arbitrary order of approximation.

For the sake of illustration, we present the next derivation in terms of a simplified ANN with one neuron in each layer. For the treatment of a unsimplified ANN, see Section 4.1.

3.4.1. *Notation.* Suppose that $x \in \mathbb{R}$ is a scalar, $y \in \mathbb{R}$ is a scalar, $\theta \in \mathbb{R}$ is a scalar, and $\sigma : \mathbb{R} \to \mathbb{R}$ is an analytic function. Since σ is analytic, it can be represented as

$$\sigma(x) = s_0 + s_1 x + s_2 x^2 + \cdots$$
$$= \sum_{k=0}^{\infty} s_k x^k. \tag{15}$$

Suppose that we can represent x as a power series of another scalar z.

$$\begin{aligned} x &= a_0 + a_1 z + a_2 z^2 + \cdots \\ &= \sum_{n=0}^{\infty} a_n z^n. \end{aligned} \tag{16}$$

Finally, suppose that we have the relation

$$y = \sigma(\theta x). \tag{17}$$

This represents a single-neuron layer of an ANN with input x, weight θ, output y, and activation σ. The value of z can be intrepreted as the input to the entire ANN.

3.4.2. *Objective.* We wish to represent y as a power series of z. That is,

$$\begin{aligned} y &= b_0 + b_1 z + b_2 z^2 + \cdots \\ &= \sum_{n=0}^{\infty} b_n z^n. \end{aligned} \tag{18}$$

We will rewrite the relation for y given by Equation (17) to find the coefficients b_i for $i = 0, 1, 2, \ldots$ for Equation (**??**).

$$\begin{aligned} y &= \sigma(\theta x) \\ &= \sum_{k=0}^{\infty} s_k (\theta x)^k \\ &= \sum_{k=0}^{\infty} s_k \theta^k \left(\sum_{n=0}^{\infty} a_n z^n \right)^k \\ &= \sum_{k=0}^{\infty} s_k \theta^k \left(\sum_{k_0+k_1+k_2+\cdots=k} \binom{k}{k_0, k_1, k_2, \ldots} \prod_{n=0}^{\infty} (a_n z^n)^{k_n} \right) \end{aligned} \tag{19}$$

Note that we used the multinomial theorem to expand the penultimate term in the equation.

3.4.3. *Coefficient Extraction.* At this point, it is infeasible to attempt to write Equation (19) directly as a power series of z. Instead, we will extract the coefficients b_i for $i = 0, 1, 2, \ldots$ by observing constraints on the coefficients.

First, as a result of the multinomial theorem, we have a constraint on the inner sum in Equation (19) that is

$$k = \sum_{n=0}^{\infty} k_n \tag{20}$$

where each k_n must be a non-negative integer. This is a constraint on the index of the sum so we shall call this an index constraint.

Second, the coefficient b_i is associated with the term z^i so we must form an equality between i and the power on z in Equation (19). To satisfy this equality, we note that

$$Cz^{0k_0+1k_1+2k_2+\cdots} = (a_0z^0)^{k_0}(a_1z^1)^{k_1}(a_2z^2)^{k_2}\cdots = \prod_{n=0}^{\infty}(a_nz^n)^{k_n} \tag{21}$$

where C is some constant. This implies that

$$i = \sum_{n=0}^{\infty} nk_n. \tag{22}$$

This is a constraint on the power of the scalar z so we shall call this a power constraint.

Notice that the power constraint implies $k_{i+1} = k_{i+2} = \cdots = 0$. Therefore, both the index constraint and power constraint can be reduced to a finite series instead of an infinite series.

$$k = \sum_{n=0}^{i} k_n, \qquad i = \sum_{n=0}^{i} nk_n \tag{23}$$

If a collection of k_j for $j = 0, 1, 2, \ldots$ satisfy both of these constraints, then, the constant coefficients of z^i which are

$$s_k\theta^k \binom{k}{k_0, k_1, \cdots, k_i} \prod_{n=0}^{\infty} a_n^{k_n}$$

may be pulled out and added to the sum of b_i. In concise terms we have derived that

$$b_i = \sum_{k=0}^{\infty} s_k\theta^k \sum_{\substack{k_0+k_1+\cdots+k_i=k \\ 0k_0+1k_1+\cdots+ik_i=i}} \binom{k}{k_0, k_1, \cdots, k_i} \prod_{n=0}^{\infty} a_n^{k_n}. \tag{24}$$

Thus, we have derived a tractable method of computing the coefficients of a power series after a linear and nonlinear transformation. We have included some worked examples of these coefficients in Appendix 6.3.

4. Discussion

Many non-trivial problems in materials science, and in science more broadly, are explained not through a single constitutive relationship, but through a combination of contributing physics.

The goal of this approach is to identify the coefficients of a hypothesized constitutive relationship, coefficients that capture the specific physics of a process, through a least-squares fit between the covector space of a neural network series expansion, $\boldsymbol{\alpha}(\boldsymbol{\theta})$, which is a function of the model parameters, and the covector space of the constitutive relationship, $\boldsymbol{\beta}(C_k)$, which is a function of the physical constants of the model.

Having fit the model parameters, $\boldsymbol{\theta}$, on a vector space spanned by the column vectors of $\mathbf{x}$, the coefficients (the covector basis) of the neural network expansion, $\boldsymbol{\alpha}(\boldsymbol{\theta})$, capture the functional relationship between the input space and the response space, both affine and non-linear contributions, introduced through those parameters, $\boldsymbol{\theta}$, and the coefficient generating functions for the activation, e.g. Equations **??** (Rectified Linear Unit, ReLU) and **??** (softmax), respectively.

Naturally, the activation generating functions must match the activation function chosen in the neural network model architecture. Equation **??** is derived for ReLU activation, the most common hidden layer activation. (Generating functions for other activations are provided in the appendix.) In addition to the hidden layers, activation functions must also be chosen for the output layer. The two most common output activations are linear (Eq. **??**) and softmax (Eq. **??**) for regression and classification, respectively.

ReLU activation,

$$\sigma(z) = \begin{cases} z & \text{if } z > 0, \\ 0 & \text{otherwise} \end{cases}$$

is discontinuous in the first derivative at $z_i = 0$. Therefore, the coefficient generating function of this activation must either be either a function of the input data, $\mathbf{z}$ or a small modification must be made to the softplus,

$$\sigma(z; \alpha) = \frac{1}{\alpha} \ln\left(1 + e^{\alpha x}\right). \tag{25}$$

In the limit as α approaches infinity, this converges to the ReLU. Practically, though, α can be assigned a large value and the coefficient generating function no longer depends on the input data, see Equation **??** in the appendix. However, because of the high computational cost of expressing the coefficients using the modified softplus, and the relative low cost of forward evaluation of the trained neural network in order to apply Equation **??**, expressing the coefficient generating function of ReLU in terms of the input data, as in Equation **??**, rather than the training-data-agnostic approach in Equation **??** would seem more practical.

Condition 1. *Both the neural network and the constitutive relationship must depend on the same independent variables.*

That is, they must be described on the same basis vectors. The fit between the coefficients of the neural network expansion–$\boldsymbol{\alpha}$, the covector space of the neural network's basis vectors–and the coefficients of the series expansion of the constitutive relationship (its covector space, $\boldsymbol{\beta}$) is only possible because both span the same subspace and share a common description of the solution within that subspace, that is, the covector spaces are colinear. That is, suppose that A maps between a vector space and its covector space. If $A : A(X) = X^*$ and $A : A(Y) = Y^*$, then $X^* = Y^*$ if and only if $X = Y$.

If $\mathbf{x} = ([c], d)$ are the concentration of a solute and grain size, respectively, then the constitutive relationship expansion would be,

$$\begin{aligned}
\sigma_y &= \sigma_f + C[c]^{2/3} + \frac{k}{\sqrt{d}} \\
&= \sigma_f + \underbrace{\sum_{k=0}^{\infty} a_k [c]^k}_{\text{solute}} + \underbrace{\sum_{k=0}^{\infty} b_k d^k}_{\text{Hall-Petch}} \\
&= \sigma_f + \sum_{k=0}^{\infty} a_k (\mathbf{S}_{\text{solute}}\mathbf{x})^{\circ k} + \sum_{k=0}^{\infty} b_k (\mathbf{S}_{\text{Hall-Petch}}\mathbf{x})^{\circ k} \qquad (26)
\end{aligned}$$

where

σ_f	= Matrix flow stress.
a_k	= Coefficient generating function for $x^{2/3}$.
b_k	= Coefficient generating function for $x^{-1/2}$.
$\mathbf{S}_{\text{solute}}$	= Selection vector/matrix for choosing the solute concentration from the input vector.
$\mathbf{S}_{\text{Hall-Petch}}$	= Selection vector/matrix for choosing the grain size from the input vector.

(Coefficient generating functions can be found in Table 1.) Combining Equations 26 and **??**,

$$\begin{aligned}
\sigma_y = \sigma_f + \sum_{k=0}^{\infty} a_k &\left[\sum_{l_1=0}^{k} \binom{k}{l_1, l_2} \prod_{m=1}^{2} ((\mathbf{S}_{\text{solute}})_m \mathbf{x}_m)^{l_m} \right] \\
&+ \sum_{k=0}^{\infty} b_k \left[\sum_{l_1=0}^{k} \binom{k}{l_1, l_2} \prod_{m=1}^{2} ((\mathbf{S}_{\text{Hall-Petch}})_m \mathbf{x}_m)^{l_m} \right], \; l_2 = k - l_1
\end{aligned}$$

which further simplifies to

$$\begin{aligned}
\sigma_y = \sigma_f + \sum_{k=0}^{\infty} \sum_{l_1=0}^{k} a_k \binom{k}{l_1, l_2} \prod_{m=1}^{2} (\mathbf{S}_{\text{solute}})_m^{l_m} \prod_{m=1}^{2} \mathbf{x}_m^{l_m} \\
+ \sum_{k=0}^{\infty} \sum_{l_1=0}^{k} b_k \binom{k}{l_1, l_2} \prod_{m=1}^{2} (\mathbf{S}_{\text{Hall-Petch}})_m^{l_m} \prod_{m=1}^{2} \mathbf{x}_m^{l_m}
\end{aligned}$$

such that the term $\prod_{m=1}^{2}\mathbf{x}_m^{l_m}$ serves as the common basis set over which the summation occurs, so that now, having a common basis, this simplifies to,

$$\sigma_y = \sigma_f + \sum_{k=0}^{\infty}\sum_{l_1=0}^{k}\binom{k}{l_1, l_2}\left(a_k\prod_{m=1}^{2}(\mathbf{S}_{\text{solute}})_m^{l_m} + b_k\prod_{m=1}^{2}(\mathbf{S}_{\text{Hall-Petch}})_m^{l_m}\right)\prod_{m=1}^{2}\mathbf{x}_m^{l_m} \tag{27}$$

It should be noted here that this common basis is a function of the length (dimension) of the input vector and of the order of the expansion. Therefore, two series will share the same basis vector and the same covector space if and only if they are taken to the same order. In addition, because there is no guarantee that the input vector space directions are orthogonal, there is no guarantee that the cross-term interactions will vanish and, therefore, must be included explicitly. This expansion includes all cross-terms and, through the element-wise exponentiation, also explicitly captures all combinations of powers of all cross-terms. Equation **??** shows that the polynomial series expansion for the first layer of a neural network,

$$y^{(1)} = \sum_k \alpha_k^{(1)}\left(\boldsymbol{\theta}^{(1)}\mathbf{x}\right)^{\circ k},$$

similarly relies on the element-wise exponential, $(\bullet)^{\circ n}$, as does the expansion of all layers. Unlike scalar exponentiation, element-wise exponentiation does not distribute, as seen in Equation **??**, and because element-wise exponentiation does not distribute, this equation explicitly captures all possible (second, $x_i^m x_j^n$; third, $x_i^m x_j^n x_k^p$; fourth, $x_i^m x_j^n x_k^p x_l^r$; etc.) cross-interactions of each term in $(\boldsymbol{\theta}\mathbf{x})$ at all polynomial orders. This is equivalent, then, to expanding over the basis set that includes all cross-interactions in the input vector space for both the constitutive relation and neural network polynomial expansions.

Data preprocessing is an important step in training a neural network to avoid implicit bias. Commonly, data is whitened, also known as scaling or standardization, $\mathbf{z}_s^{(k)} : \mathbf{z}_s^{(k)} = \frac{\mathbf{z}^{(k)} - \overline{\mathbf{z}^{(k)}}}{\sigma}$, where $\overline{\mathbf{z}^{(k)}}$ is the arithmetic mean and σ the standard deviation of data in layer, k. However, a model trained on such scaled data would no longer share the vector space of the constitutive relationship. To ensure that both the neural network and constitutive relation expansions share a common vector space, and thus a common covector space, this whitening procedure must be integrated into the construction of the neural network architecture.

Procedurally, the neural network expansion proceeds as described in Section 3. The input to each layer, which is the output from the previous layer, is subjected first to an affine transformation, then to an activation function. The activation function is completely described though its polynomial expansion and the corresponding coefficient generating function. Using this same structure, then, data whitening can be applied to any layer, including the input layer, as an identity tranform ($\boldsymbol{\theta} = \mathbf{I}_d$, where d is the dimension of the source layer) followed by the whitening expansion whose coefficient generating function is simply,

$$\alpha_k(\mathbf{z}) = \begin{cases} -\overline{\mathbf{z}}/\sigma & \text{if } k = 0, \\ 1/\sigma & \text{if } k = 1, \\ 0 & \text{otherwise} \end{cases} \tag{28}$$

where

$\overline{\mathbf{z}}$ = The mean of the data into the standarization layer.

σ = The standard deviation of the data into the standardization layer.

By introducing such a whitening layer, data standardization can be included at any point in the neural network architecture.

TBW. This must answer the question: how do we know if we've measured the right things–not the number of measurements, but that we have enough information? How do we know that the solution has converged? Example: A model is to be fit to the number of cakes produced by a bakery. If we are given weights of flour and sugar and number of eggs, our model can accurately tell us the *volume* of cakes produced, but not the number. If this bakery makes cupcakes, but the model is trained across a spectrum of bakers, such as purveyors of wedding cakes and catering companies who work with large sheet cakes, then our dimensions (flour, sugar, eggs) are insufficient to fit the number of cakes produced. If, however, we also include number of orders and revenue, some information about the *quanta* of cakes is baked into those two additional dimensions (sorry, I couldn't help myself). Therefore, a model based only on (flour, sugar, eggs) is dimensionally insufficient, but a model based on (flour, sugar, eggs, order size, revenue) is dimensionally sufficient.

This is a broader question that may be beyond the scope of this paper. Let's return to this if, after completing the first pass, we feel that this can be addressed by what we've done.

This leads to a seven-step process for systematically and incrementally extracting physics information from an ANN:

(1) Collect data–features and targets–for which relationships are expected to exist.
(2) Design and train a fully dense multi-layer perceptron network (ANN).
(3) Build a power series expansion from the architecture of this ANN, using Equations (48) and (**??**) to populate the coefficients using the trained weights from the neural network.
(4) Hypothesize a constitutive relationship between the feature space and the target space.
(5) Recast the terms in the hypothesis function from #4 as power series expansions, creating power series coefficient generating functions that are functions of the constitutive model fitting parameters. An example of this process is provided below, and a table of select power series expansions relevant to materials research are provided in Table (**??**).
(6) Perform an optimization, *e.g.* least squares, fit to find the fitting parameters from #5

(7) Calculate the residuals of the ANN power series expansion coefficient vector, and from this residual vector, the error in the model. If the accuracy is sufficient for the application, stop; otherwise, expand the constitutive relationship from step #4 and repeat.

4.1. **Polynomial Expansion of a Vector Layer.**

This derivation will be a generalization of the derivation given in Section 3.4 where each layer of the ANN may have an arbitrary number of neurons. We show a similar result to before where we find coefficients for a power series after linear and nonlinear transformations.

4.1.1. *Notation.* Suppose that $\mathbf{x} \in \mathbb{R}^d, (d \in \mathbb{Z})$ is a vector, $\mathbf{y} \in \mathbb{R}^c, (c \in \mathbb{Z})$ is a vector, $\mathbf{\Theta} \in \mathbb{R}^{c \times d}$ is a matrix, and $\sigma : \mathbb{R} \to \mathbb{R}$ is an analytic function. Since σ is analytic, it can be represented as

$$\begin{aligned}\sigma(x) &= s_0 + s_1 x + s_2 x^2 + \cdots \\ &= \sum_{k=0}^{\infty} s_k x^k. \end{aligned} \tag{29}$$

Suppose that we can represent each entry of $\mathbf{x}$ as a power series of the entries of a vector $\mathbf{z} \in \mathbb{R}^w, (w \in \mathbb{Z})$.

$$\begin{aligned} &\forall i = 1, \cdots, d, \\ &x_i = a^{(i)}_{0,0,\cdots,0} + a^{(i)}_{1,0,\cdots,0} z_1 + a^{(i)}_{2,0,\cdots,0} z_1^2 + \cdots \\ &\quad + a^{(i)}_{0,1,\cdots,0} z_2 + a^{(i)}_{1,1,\cdots,0} z_1 z_2 + a^{(i)}_{2,1,\cdots,0} z_1^2 z_2 + \cdots \\ &\quad + a^{(i)}_{0,0,\cdots,1} z_w + a^{(i)}_{1,0,\cdots,1} z_1 z_w + a^{(i)}_{2,0,\cdots,1} z_1^2 z_w + \cdots \\ &\quad = \sum_{n_1, n_2, \cdots, n_w}^{\infty, \infty, \cdots, \infty} a^{(i)}_{n_1, n_2, \cdots, n_w} z_1^{n_1} z_2^{n_2} \cdots z_w^{n_w}. \end{aligned} \tag{30}$$

Finally, suppose that we have the relation

$$\mathbf{y} = \sigma(\mathbf{\Theta x}) \tag{31}$$

where σ is being applied element-wise to a vector.

Similar to the scalar case, this represents a multi-neuron layer of an ANN with input $\mathbf{x}$, weights θ, output $\mathbf{y}$, and activation σ. The value of z can be intrepreted as the input to the entire ANN.

4.1.2. *Objective.* We wish to represent each entry of $\mathbf{y}$ as a power series of the entries of $\mathbf{z}$. That is,

$$\begin{aligned}
&\forall i = 1, \cdots, c,\\
y_i &= b^{(i)}_{0,0,\cdots,0} + b^{(i)}_{1,0,\cdots,0} z_1 + b^{(i)}_{2,0,\cdots,0} z_1^2 + \cdots\\
&\quad + b^{(i)}_{0,1,\cdots,0} z_2 + b^{(i)}_{1,1,\cdots,0} z_1 z_2 + b^{(i)}_{2,1,\cdots,0} z_1^2 z_2 + \cdots\\
&\quad + b^{(i)}_{0,0,\cdots,1} z_w + b^{(i)}_{1,0,\cdots,1} z_1 z_w + b^{(i)}_{2,0,\cdots,1} z_1^2 z_w + \cdots\\
&= \sum_{n_1,n_2,\cdots,n_w}^{\infty,\infty,\cdots,\infty} b^{(i)}_{n_1,n_2,\cdots,n_w} z_1^{n_1} z_2^{n_2} \cdots z_w^{n_w}.
\end{aligned} \tag{32}$$

Rewriting Equation (31) in terms of Equations (29) and (30), we obtain

$$\begin{aligned}
&\forall i = 1, \cdots, c,\\
y_i &= [\sigma(\boldsymbol{\Theta}\mathbf{x})]_i\\
&= \sigma(\theta_i \mathbf{x})\\
&= \sum_{k=0}^{\infty} s_k (\theta_i \mathbf{x})^k\\
&= \sum_{k=0}^{\infty} s_k \left(\sum_{j=1}^{d} \theta_{ij} x_j \right)^k\\
&= \sum_{k=0}^{\infty} s_k \left(\sum_{k_1+\cdots+k_d=k} \binom{k}{k_1, \cdots, k_d} \prod_{j=1}^{d} (\theta_{ij} x_j)^{k_j} \right)\\
&= \sum_{k=0}^{\infty} s_k \left(\sum_{k_1+\cdots+k_d=k} \binom{k}{k_1, \cdots, k_d} \prod_{j=1}^{d} \theta_{ij}^{k_j} \left(\sum_{n_1,\cdots,n_w}^{\infty,\cdots,\infty} a^{(j)}_{n_1,\cdots,n_w} z_1^{n_1} \cdots z_w^{n_w} \right)^{k_j} \right)\\
&= \sum_{k=0}^{\infty} s_k \left(\sum_{k_1+\cdots+k_d=k} \binom{k}{k_1, \cdots, k_d} \prod_{j=1}^{d} \theta_{ij}^{k_j} \left(\sum_{\|\mathbf{L}\|_1=k_j} \binom{k_j}{\mathbf{L}} \prod_{n_1,\cdots,n_w}^{\infty,\cdots,\infty} (a^{(j)}_{n_1,\cdots,n_w} z_1^{n_1} \cdots z_w^{n_w})^{l_{n_1,\cdots,n_w}} \right) \right).
\end{aligned} \tag{33}$$

where $\mathbf{L}$ is a collection of non-negative integers that are indices such that

$$\begin{aligned}
\mathbf{L} &= l_{\underbrace{0,\cdots,0}_{\times w}},\ldots,l_{\underbrace{\infty,\cdots,\infty}_{\times w}}, \\
\|\mathbf{L}\|_1 &= l_{0,\cdots,0}+\cdots+l_{\infty,\cdots,\infty}, \\
\binom{k_j}{\mathbf{L}} &= \binom{k_j}{l_{0,\cdots,0},\ldots,l_{\infty,\cdots,\infty}}.
\end{aligned}$$

4.1.3. *Coefficient Extraction.* To find the coefficients $b^{(i)}_{m_1,\cdots,m_w}$ from Equation (33), we must find terms satisfying index constraints

$$k=\sum_{n=1}^{d}k_n \tag{34}$$

$$k_j=\|\mathbf{L}\|_1 \tag{35}$$

and power constraints

$$\begin{aligned}
m_1 &= \sum_{n_1,\cdots,n_w}^{\infty,\cdots,\infty} n_1 l_{n_1,\cdots,n_w} \\
m_2 &= \sum_{n_1,\cdots,n_w}^{\infty,\cdots,\infty} n_2 l_{n_1,\cdots,n_w} \\
\vdots &= \quad \vdots \\
m_3 &= \sum_{n_1,\cdots,n_w}^{\infty,\cdots,\infty} n_w l_{n_1,\cdots,n_w}. \qquad (36)
\end{aligned}$$

Similar to the scalar case, notice that the power constraints can be simplified since any solution must also satisfy

$$l_{n_1,\cdots,n_{p-1},m_p+1,n_{p+1},\cdots,n_w}=l_{n_1,\cdots,n_{p-1},m_p+2,n_{p+1},\cdots,n_w}=\cdots=0. \tag{37}$$

resulting in

$$\begin{aligned} m_1 &= \sum_{n_1,\cdots,n_w}^{m_1,\cdots,m_w} n_1 l_{n_1,\cdots,n_w} \\ m_2 &= \sum_{n_1,\cdots,n_w}^{m_1,\cdots,m_w} n_2 l_{n_1,\cdots,n_w} \\ \vdots &= \quad \vdots \\ m_w &= \sum_{n_1,\cdots,n_w}^{m_1,\cdots,m_w} n_w l_{n_1,\cdots,n_w}. \end{aligned} \tag{38}$$

If a collection of $k_1, \ldots, k_d$ and $\mathbf{L}$ satisfy all of these constraints, then the value

$$s_k \binom{k}{k_1, \cdots, k_d} \binom{k_j}{\mathbf{L}} \theta_{ij}^{k_j} (a_{n_1,\cdots,n_w}^{(j)})^{l_{n_1,\cdots,n_w}}$$

may be pulled out to the sum of $b_{m_1,\cdots,m_w}^{(i)}$. In concise terms we have derived

(39)
$$b_{m_1,\cdots,m_w}^{(i)} = \sum_{k=0}^{\infty} s_k \sum_{k_1+\cdots+k_d=k} \binom{k}{k_1, \cdots, k_d} \prod_{j=1}^{d} \theta_{ij}^{k_j} \sum_{\substack{l_{0,\cdots,0}+\cdots+l_{\infty,\cdots,\infty}=k_j \\ \sum_{n_1,\cdots,n_w}^{m_1,\cdots,m_w} n_1 l_{n_1,\cdots,n_w}=m_1 \\ \vdots \\ \sum_{n_1,\cdots,n_w}^{m_1,\cdots,m_w} n_w l_{n_1,\cdots,n_w}=m_w}} \binom{k_j}{\mathbf{L}} (a_{n_1,\cdots,n_w}^{(j)})^{l_{n_1,\cdots,n_w}}$$

The only issue is that of finding solutions to the index and power constraints.

4.1.4. *Interpretation.* We can use this method for deriving power series for repeatedly composed functions with linear transformations (i.e. $\mathbf{y} = f(\mathbf{\Theta}_3 f(\mathbf{\Theta}_1 f(\mathbf{\Theta}_1 \mathbf{x})))$). We represent our original input variable as $\mathbf{x} = \mathbf{z}$ so that we have coefficients $a_{1,0,\cdots,0} = a_{0,1,\cdots,0} = \cdots = a_{0,0,\cdots,1} = 1$ with all other coefficients zero. Then, we update $\mathbf{a}$ for each composition of the function to obtain $\mathbf{b}$ for $\mathbf{y}$. This can be easily applied to neural networks.

In order to make this method computationally feasible, we must truncate the power series at some precision. Say the maximum power we wish on any variable is K, then, we simply replace instances of $\sum_{k=0}^{\infty} s_k x^k$ with $\sum_{k=0}^{K} s_k x^k$. Thus, coefficients of order $0, 1, \cdots, K$ will have no error and our approximation will only have truncation error from dropping terms with order $> K$.

4.1.5. *Constraint Analysis.* The constraints form a system called a restricted partition which is a concept within number theory. Fel has found an explicit solution to the constraints found in the simplified ANN case. We can construct an algorithm that operates in $\mathcal{O}(K^w)$ time by iterating through each of the indices $l_{0,\cdots,0}, \cdots, l_{m_1,\cdots,m_w}$ from 0 to K and checking if they satisfy the constraints. This warrants some future study to see if there are ways to reduce the operation time and find solutions to the system of constraints more quickly.

5. Conclusions

A generalized mathematical framework for proposing a constitutive relationship and fitting the physical constants associated with that constitutive relationship to a data corpus using the generalized fitting framework provided by machine learning/artificial intelligence has been presented. The proposed method maps between an arbitrary constitutive relationship and an artificial neural network model. The resulting covector spaces (coefficients) of the series are colinear. The generating function for the constitutive relationship is in terms of physical constants while that of neural network is determined through the trained model parameters. The method of least-squares is used to fit the physical constants to the trained neural network model parameters through this colinear covector space.

A simplified yield strength model, which includes flow stress, solute concentration, and Hall-Petch strengthening, is provided as an example to demonstrate the more general form for constructing the vector space for these two models.

Neural network series expansions are constructed layer-wise, which allows this framework to handle arbitrarily complex network architectures, including drop out, whitening, and any element-wise activation function. Mathematical descriptions of the rectified linear unit (ReLU) activation commonly used in neural network hidden layers, linear activation for regression networks, and softmax activation for classification networks are derived.

A mathematical framework to create constitutive relationship series expansions is also provided. Select coefficient generating functions for functional forms commonly found in materials physics are presented in the Appendix (Table 1) to serve as a starting point, but any other functions that can be described by a polynomial series may be used.

As a single framework capable of describing arbitrarily complex relationships, this approach is intended to facilitate fits between existing data and any hypothesized constitutive relationships built upon the same vector space as the trained neural network model.

The authors wish to thank Linus, Martin, Richard, ... for fruitful discussions. Support through XYZ grant ABC123456 is gratefully acknowledged.

6. Appendix

6.1. **Activation Functions.**

6.1.1. *The logistic sigmoid.*

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{40}$$

is a special case of the generating function for the Euler polynomial coefficients,

$$\frac{2e^{xt}}{1+e^{t}} = \sum_{n=0}^{\infty} E_n(x)\frac{t^n}{n!} \tag{41}$$

where, for $x = 0$,

$$\sigma(x) = \frac{1}{2} \sum_{n=0}^{\infty} E_n(0) \frac{(-1)^n}{n!}. \tag{42}$$

The Euler polynomials at $x = 0$,

$$E_n(0) = -2(n+1)^{-1} \left(2^{n+1} - 1\right) B_{n+1} \tag{43}$$

where B_n is the nth Bernoulli number. Since Bernoulli numbers of odd index, with the exception of B_1, are zero, $E_i(0) = 0$ for $i = 2, 4, 6, \ldots, 2n$. Therefore, the summand and limits of Equation (42) change to

$$\sigma(x) = \frac{1}{2} - \frac{1}{2} \sum_{n=1}^{\infty} \left(\frac{E_{2n-1(0)}}{(2n-1)!} \right) x^{2n-1}. \tag{44}$$

The series representation of $E_{2n-1}(x)$

$$E_{2n-1}(x) = \frac{(-1)^n 4(2n-1)!}{\pi^{2n+1}} \sum_{k=0}^{\infty} \frac{\cos[(2k+1)\pi x]}{(2k+1)^{2n}} \tag{45}$$

such that,

$$E_{2n-1}(0) = \frac{(-1)^n 4(2n-1)!}{\pi^{2n+1}} \sum_{k=0}^{\infty} \frac{1}{(2k+1)^{2n}} \tag{46}$$

and therefore,

$$\begin{aligned}
\sigma(x) &= \frac{1}{2} - \sum_{n=1}^{\infty} 2 \frac{(-1)^n}{\pi^{2n}} \left(\sum_{k=0}^{\infty} \frac{1}{(2k+1)^{2n}} \right) x^{2n-1} && (47)\\
&= \frac{1}{2} - \sum_{n=1}^{\infty} 2 \frac{(-1)^n}{\pi^{2n}} \left(4^{-n} \left(4^n - 1\right) \zeta(2n)\right) x^{2n-1} \\
&= \frac{1}{2} - \sum_{n=1}^{\infty} \underbrace{2 \left(\frac{-1}{4\pi^2} \right)^n \left(4^n - 1\right) \zeta(2n)}_{a_n} x^{2n-1} \\
&= \sum_{n=0}^{\infty} a_n x^n, \ a_n = \begin{cases} 1/2 & n = 0 \\ -2 \left(\frac{-1}{4\pi^2}\right)^{(n+1)/2} \left(4^{(n+1)/2} - 1\right) \zeta(n+1) & n \text{ odd} \\ 0 & n \text{ even} \end{cases} && (48)
\end{aligned}$$

6.2. **Coefficient Generating Functions for Common Functions.** Central to this approach is the connection ability to represent a constitutive relationship and a neural network on the same vector/covector space. This is done through a polynomial series expansion of both the neural network, covered in the text, and the constitutive relationship. Select generating functions are provided here.

k	Ca^x	Cx^n	$Ce^{-\beta x}$	$Cx^{-1/2}$	$C(1+x)^\alpha$	$C\ln(1+x)$
0	1	–	C	C	C	0
1	$C\ln a$	–	$-\beta C$	$-\frac{1}{2}C$	$C\alpha$	C
2	$\frac{(\ln a)^2}{2}C$	–	$\frac{\beta^2}{2}C$	$\frac{3}{8}C$	$C\frac{\alpha(\alpha-1)}{2!}$	$\frac{-C}{2}$
$\vdots$			$\vdots$			
n	$\frac{(\ln a)^n}{n!}C$	$\begin{cases} C & \text{if } k=n \\ 0 & \text{otherwise} \end{cases}$	$(-1)^n\frac{\beta^n}{n!}C$	$C\prod_{i=1}^{n}(-1)\frac{2i-1}{2i}$	$C\frac{\prod_{i=1}^{n}\alpha-n+1}{n!}$	$C\frac{(-1)^{n+1}}{n}$
Constraint					$-1 < x < 1$	$-1 < x \leq 1$

6.3. **Coefficients of the Scalar Polynomial Series.** A result of the coefficient derivation of polynomial series is that we may construct formulae for the coefficients of an arbitrary function by simply solving for the constraints in Equation (24). We provide a few instances of worked coefficients here.

We find b_0. $i = 0 \implies k_1 = k_2 = \cdots = 0$ so $k_0 = k$. Therefore,

$$\begin{aligned} b_0 &= \sum_{k=0}^{\infty} s_k \theta^k \sum_{\substack{k_0+k_1+\cdots=k \\ 0k_0+1k_1+\cdots=0}} \binom{k}{k_0, k_1, \ldots} \prod_{n=0}^{\infty} a_n^{k_n} \\ &= \sum_{k=0}^{\infty} s_k \theta^k \binom{k}{k} a_0^k \\ &= \sum_{k=0}^{\infty} s_k \theta^k a_0^k \end{aligned} \tag{49}$$

We find b_1. $i = 1 \implies k_2 = k_3 = \cdots = 0$ so $k_0 = k - 1$ and $k_1 = 1$. Further, $k \geq 1$. Therefore,

$$\begin{aligned} b_1 &= \sum_{k=0}^{\infty} s_k \theta^k \sum_{\substack{k_0+k_1+\cdots=k \\ 0k_0+1k_1+\cdots=1}} \binom{k}{k_0, k_1, \ldots} \prod_{n=0}^{\infty} a_n^{k_n} \\ &= \sum_{k=1}^{\infty} s_k \theta^k \binom{k}{k-1, 1} a_0^{k-1} a_1 \\ &= \sum_{k=1}^{\infty} k s_k \theta^k a_0^{k-1} a_1 \end{aligned} \tag{50}$$

We find b_2. For this coefficient, there are multiple solutions to the constraints.

- $i = 2 \implies k_3 = k_4 = k_5 = \cdots = 0$ and either
  - $k_2 = 0, k_1 = 2, k_0 = k - 2$ and $k \geq 2$ or
  - $k_2 = 1, k_1 = 0, k_0 = k - 1$ and $k \geq 1$

Therefore,

$$
\begin{aligned}
b_2 &= \sum_{k=0}^{\infty} s_k \theta^k \sum_{\substack{k_0+k_1+\cdots=k \\ 0k_0+1k_1+\cdots=2}} \binom{k}{k_0, k_1, \ldots} \prod_{n=0}^{\infty} a_n^{k_n} \\
&= \sum_{k=2}^{\infty} s_k \theta^k \left(\binom{k}{k-2, 2, 0} a_0^{k-2} a_1^2 \right) + \sum_{k=1}^{\infty} s_k \theta^k \left(\binom{k}{k-1, 0, 1} a_0^{k-1} a_2 \right) \\
&= s_1 \theta a_2 + \sum_{k=2}^{\infty} s_k \theta^k \left(\frac{k(k-1)}{2} a_0^{k-2} a_1^2 + k a_0^{k-1} a_2 \right) \qquad (51)
\end{aligned}
$$