Deriving Equations from Sensor Data Using Dimensional Function Synthesis

Youchao Wang*
yw479@cam.ac.uk
University of Cambridge
United Kingdom

Sam Willis*
sjw238@cam.ac.uk
University of Cambridge
United Kingdom

Vasileios Tsoutsouras*†
vt298@cam.ac.uk
University of Cambridge
United Kingdom

Phillip Stanley-Marbell†
phillip.stanley-marbell@eng.cam.ac.uk
University of Cambridge
United Kingdom

ABSTRACT

We present a new method for deriving functions that model the relationship between multiple signals in a physical system. The method, which we call *dimensional function synthesis*, applies to data streams where the dimensions of the signals are known. The method comprises two phases: a compile-time synthesis phase and a subsequent calibration using sensor data.

We implement dimensional function synthesis and use the implementation to demonstrate efficiently summarizing multi-modal sensor data for two physical systems using 90 laboratory experiments and 10 000 synthetic idealized measurements. We evaluate the performance of the compile-time phase of dimensional function synthesis as well as the calibration phase overhead, inference latency, and accuracy of the models our method generates.

The results show that our technique can generate models in less than 300 ms on average across all the physical systems we evaluated. When calibrated with sensor data, our models outperform traditional regression and neural network models in inference accuracy in all the cases we evaluated. In addition, our models perform better in training latency (over 8660× improvement) and required arithmetic operations in inference (over 34× improvement). These significant gains are largely the result of exploiting information on the physics of signals that has hitherto been ignored.

1 INTRODUCTION

Physical systems instrumented with sensors can generate large volumes of data. These data are useful in understanding previous behaviors of the systems that generate them (e.g., monitoring properties of components in aircraft) as well as in predicting future behaviors of those systems (e.g., predicting failures of components in machinery).

Unlike data sources such as speech or text however, data from sensors of physical phenomena must obey the laws of physics. Existing methods for constructing predictive models from sensor data however do not fully exploit prior knowledge of the physical interpretation of sensor data. In this work, we use information about physical dimensions[1] of sensors to synthesize compact predictive models from sensor data. The state of the art in deriving compact models from such data streams today is to apply some form of machine learning [13, 25]. Blindly applying machine learning to data from physical systems however ignores important prior knowledge about the physical implications of the signals.

1.1 Contemporary Methods Ignore Physics

Despite its use in programming languages for tasks such as extending type systems with units of measure [1–3, 5, 8, 9, 12, 14, 15, 17, 18, 23, 30], physical information in the form of dimensions (e.g., time, temperature, and so on) has seen limited use in building models of physical systems from data. Physical constraints can be viewed as a form of Bayesian prior [4]. Kalman filters incorporate information about the physical constraints of systems but use this information primarily to guide their state update equations. Today, no principled techniques exist which learn models from sensor data while exploiting the requirements of dimensional consistency of sensors to learn more compact models.

1.2 Dimensional Function Synthesis

Dimensional function synthesis is a new method to efficiently derive functions relating the values from multiple streams of data when those data are from physical systems and as a result have known physical dimensions. The insight behind the method is that any equation relating physical quantities must obey the principle of *dimensional homogeneity* from dimensional analysis [6]: The two sides of an equation, an addition, or a subtraction, must have the same physical dimensions.

Based on this observation, dimensional function synthesis enables automatic correlation of physical measurements (e.g., mass, time, acceleration etc.) in a physically-consistent manner. In a first offline analysis phase, the method groups physical parameters according to their dimensions. Then, in a second run-time stage and using data from sensors of the physical parameters in question, dimensional function synthesis calibrates the set of dimensionally-plausible equations to obtain a final set of predictive models.

Figure 1 shows a schematic view of the process. The inputs to dimensional function synthesis are a list of signals with known dimensions relevant to the system under study and a set of data

*Contributed equally.
†Corresponding authors.
[1]In keeping with the convention in physics, we use the term *dimensions* to refer to quantities such as length or time and we use the term *units* to refer to a value in a standardized system for quantifying values of a given dimension, such as centimeters or miles for length and Pascals or mm Hg for pressure.

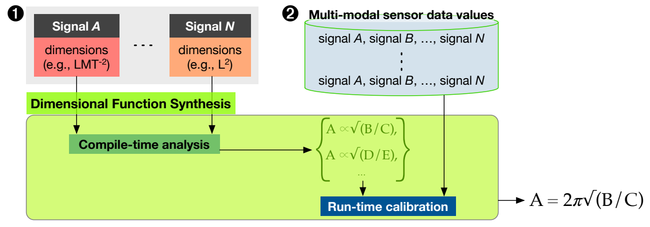


Figure 1: Dimensional function synthesis uses information about physical dimensions to generate a family of candidate equations. It then uses sensor measurements to calibrate the set of candidate equations.

values corresponding to instances of those signals. The outcome is a model relating the signals and predicting the expected physical system output. We developed the method of dimensional function synthesis with the objective of creating inference models that can fit within the memory, computation, and energy constraints of low-power embedded systems such as those powered by scavenged energy. The method may however also apply to computing systems that are not constrained by compute resources or by energy, but which nonetheless need simple models defined over a large parameter space.

1.3 Contributions

This article makes three main contributions to the state of the art in deriving analytic models from sensor data, whether for the purpose of summarization or for use in prediction:

- **A new method, which we call *dimensional function synthesis*, for deriving analytic models from sensor data**, based on the principle of dimensional homogeneity in physical systems (Section 3).
- **An implementation of dimensional function synthesis** and its performance evaluation (Section 4).
- **An evaluation comparing models generated by dimensional function synthesis** to models generated by regression and neural networks. The evaluation uses sensor data from 90 physical experiments augmented with 10 000 synthetic idealized experiments (Section 5).

2 MATHEMATICAL FOUNDATION

Dimensional analysis is often introduced in engineering curricula as a simple method for checking the validity of computations on physical quantities. It is frequently used in engineering, fluid mechanics, and electrodynamics in cases such as deflection of turbine blades in turbo machine designs [26]. The approach to dimensional analysis familiar to most researchers in computing systems and computer science involves taking some physical quantity (e.g., acceleration) and expressing it in terms of basic dimensions such as length (L) and time (T) to obtain its dimensions (LT^{-2} for acceleration). Dimensional analysis however has a well-developed mathematical framework that combines a few basic principles from physics with an analytic formulation based on linear algebra and group theory [6, 10, 21]. The remainder of Section 2 provides a brief overview of this mathematical formalization of dimensional analysis. We then use these definitions to derive the method for generating equations from sensor data (dimensional function synthesis), in Section 3.

Table 1: Examples of physical systems and their $\mathbb{S}_{\text{symbols}}$.

Physical System	Parameters, $\mathbb{S}_{\text{symbols}}$	Parameters	Dimensions
Altimeter in a fitness tracker	$\mathbb{S}_{\text{symbols}} = \{p, h\}$	Pressure, p	$\mathcal{D}(p) = ML^{-1}T^{-2}$
		Elevation, h	$\mathcal{D}(h) = L$
Pendulum	$\mathbb{S}_{\text{symbols}} = \{t, l, g\}$	Period, t	$\mathcal{D}(t) = T$
		Rod length, l	$\mathcal{D}(l) = L$
		Gravity, g	$\mathcal{D}(g) = LT^{-2}$

2.1 Parameters in Physical Equations and Dimensionless Products

Let i be an index over a set of symbols in a physical equation and let Q_i be one of those symbols in an equation describing a physical system. Typically, these symbols will correspond to parameters of some physical model and we will therefore use the term *parameter* and *symbol* interchangeably. Let $\mathcal{D}(\cdot)$ be a function from symbols to some product of basic dimensions. For any equation describing a physical system, we introduce the set $\mathbb{S}_{\text{symbols}}$, where

$$\mathbb{S}_{\text{symbols}} = \{Q_1, Q_2, \ldots, Q_n\}\,. \tag{1}$$

For the system described by $\mathbb{S}_{\text{symbols}}$ to be physically plausible, each member Q_i of $\mathbb{S}_{\text{symbols}}$ can be rewritten in terms of a set of basic *dimensions* (e.g., mass, length, time) or is otherwise *dimensionless*. In Section 3, we show how we obtain the set $\mathbb{S}_{\text{symbols}}$ from specifications of physical systems.

Example: For the equation

$$F = m \cdot a,$$

we have $\mathbb{S}_{\text{symbols}} = \{F, m, a\}$, $Q_1 = F$, $Q_2 = m$, $Q_3 = a$. The dimensions of the members of $\mathbb{S}_{\text{symbols}}$ are given by $\mathcal{D}(Q_1) = MLT^{-2}$, $\mathcal{D}(Q_2) = M$, and $\mathcal{D}(Q_3) = LT^{-2}$. Table 1 shows additional examples of parameters and their units for the data from sensors in several physical systems that can be instrumented with sensors to monitor their behavior. For example, the altimeter subsystem of a fitness tracker uses changes in atmospheric pressure to estimate changes in elevation and hence to estimate the number of flights of stairs climbed.

The key idea in the mathematical formulation of dimensional analysis is that for any set $\mathbb{S}_{\text{symbols}}$ such as in the example above, we can arrange the members Q_i of $\mathbb{S}_{\text{symbols}}$ into groups of products where the dimensions of the symbols in the product cancel out and as a result each product is dimensionless [6, 7].

Why finding dimensionless products is useful: Given a set of parameters $\mathbb{S}_{\text{symbols}}$ for a physical system, each of the dimensionless products we can form from a subset of $\mathbb{S}_{\text{symbols}}$ directly gives us a dimensionally-valid equation between those parameters: We can equate the dimensionless product to any dimensionless quantity to obtain a dimensionally-correct equation; if we then rearrange that equation to move one of the parameters to be the only term on one side of the equation, we have a dimensionally valid equation of that parameter in terms of the remainder of $\mathbb{S}_{\text{symbols}}$.

Example: For $\mathbb{S}_{\text{symbols}} = \bigcup_i \{Q_i\}$ and the dimensionless product

$$\frac{Q_1^{k_1} Q_2^{k_2} \ldots Q_m^{k_m}}{Q_{m+1}^{k_{m+1}} Q_{m+2}^{k_{m+2}} \ldots Q_n^{k_n}},$$

we can equate the dimensionless product to a constant to obtain

$$\frac{Q_1^{k_1} Q_2^{k_2} \ldots Q_m^{k_m}}{Q_{m+1}^{k_{m+1}} Q_{m+2}^{k_{m+2}} \ldots Q_n^{k_n}} = C.$$

We can then obtain an expression for any of the $Q_i \in \mathbb{S}_{\text{symbols}}$. For example, for Q_1,

$$Q_1 = \sqrt[k_1]{\frac{C Q_{m+1}^{k_{m+1}} Q_{m+2}^{k_{m+2}} \ldots Q_n^{k_n}}{Q_2^{k_2} \ldots Q_m^{k_m}}}.$$

This simple idea generalizes to a method for obtaining a function relating all the parameters $Q_i \in \mathbb{S}_{\text{symbols}}$ relevant to a system, in terms of one or more dimensionless products that we can form from $\mathbb{S}_{\text{symbols}}$.

Definition 1. *Let i be an index over the set $\mathbb{S}_{\text{symbols}}$ of symbols in the description of a physical system, let n be the cardinality of $\mathbb{S}_{\text{symbols}}$, and let m be an index such that $m < n$. Let k_i be a value drawn from the set of rational numbers $\mathbb{Q}$. A dimensionless product Π of parameters $Q_i \in \mathbb{S}_{\text{symbols}}$ is a monomial of parameters raised to powers such that $\mathcal{D}(\Pi) = 1$, that is,*

$$\Pi = \frac{Q_1^{k_1} Q_2^{k_2} \ldots Q_m^{k_m}}{Q_{m+1}^{k_{m+1}} Q_{m+2}^{k_{m+2}} \ldots Q_n^{k_n}}. \quad (2)$$

Without loss of generality, we formulate our method in terms of rational powers k_i. This influences how we solve for the powers k_i needed to form a dimensionless product from the parameters in a set $\mathbb{S}_{\text{symbols}}$. Rather than using techniques such as singular value decomposition which are numerically stable and which provide a unique orthonormal null space [28], because our goal is to obtain monomials that are easily evaluated on resource-constrained embedded systems, we instead use methods based on finding the rational null space of a matrix by reduced row-echelon form (RREF). We revisit this observation in Section 3.2.

For a physical system defined by a set of parameters $\mathbb{S}_{\text{symbols}}$, we can define groups of one or more dimensionless products based on Definition 1. Because of the form of Equation 2, these groups of dimensionless products are often referred to as *Π groups* [6, 7].

2.2 Groups of Dimensionless Products and the Buckingham Π Theorem

The primary insight exploited in many contemporary applications of dimensional analysis [24, 27] is that for any physical system represented by a set of physical parameters $\mathbb{S}_{\text{symbols}}$, it is often possible to re-parametrize the system in terms of a smaller number of parameters. This basic observation is often used in the engineering and design of mechanical systems to reduce the number of parameters needed in experimentation. The principle behind the observation is what is commonly known as the Buckingham Π theorem [6][2]:

[2]Buckingham [6], Carlsson [7], and others provide proofs of the Π theorem.

Theorem 1. *Let n be the number of parameters in a description of a physical system, i.e., $n = |\mathbb{S}_{\text{symbols}}|$. Let r be the number of dimensions from some orthogonal dimensional basis that are sufficient to express the dimensions of the parameters in $\mathbb{S}_{\text{symbols}}$. Then, $n - r$ dimensionless products Π_i can be formed from the n parameters.*

The $n - r$ dimensionless products Π_i are the roots of some function Φ, that is,

$$\Phi(\Pi_1, \Pi_2, \ldots, \Pi_{n-r}) = 0. \quad (3)$$

Let Φ' be a function over the dimensionless products Π_i. It follows for the i-th product, Π_i, that,

$$\Pi_i = \Phi'(\Pi_1, \Pi_2, \ldots, \Pi_{i-1}, \Pi_{i+1}, \ldots, \Pi_{n-r}). \quad (4)$$

When $n - r = 1$, i.e., when there is only one Π product in the Π groups, then

$$\Phi(\Pi_1) = 0. \quad (5)$$

It follows that there exists some real-valued constant C such that

$$\Pi_1 = \frac{Q_1^{k_1} Q_2^{k_2} \ldots Q_m^{k_m}}{Q_{m+1}^{k_{m+1}} Q_{m+2}^{k_{m+2}} \ldots Q_n^{k_n}} = C. \quad (6)$$

There are multiple possible Π groups: For the same parameter set $\mathbb{S}_{\text{symbols}}$, of cardinality n, there are multiple possible groups of dimensionless products (i.e., multiple possible Π groups).

Dimension function synthesis, which we introduce next in Section 3, automates the process of finding *all the valid Π products across all possible Π groups.* Figure 2 shows the steps using the terminology introduced in this section and a physical system comprising an unpowered flying object (a glider) as a working example.

3 DIMENSIONAL FUNCTION SYNTHESIS

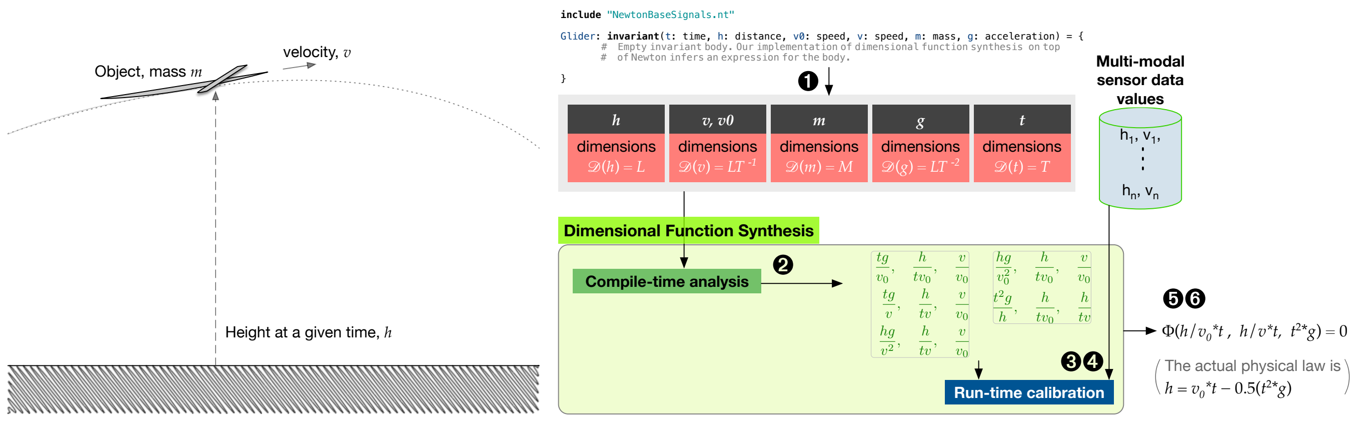
From the set $\mathbb{S}_{\text{symbols}}$ of parameters defining a physical system, we can construct a matrix representation of the system where the columns of the matrix are the parameters that are members of $\mathbb{S}_{\text{symbols}}$, the rows of the matrix are base dimensions such as length, mass, or time, returned by the function $\mathcal{D}$ (Section 2.1), and the elements in the matrix are the exponents of the base dimensions.

Dimensional function synthesis consists of a compile time step which computes the Π groups and a run-time step which calibrates the functional relationship between the derived Π products. Similar to other data-driven techniques, it uses sensor measurements as inputs and produces a model that maps those measurements to an expected output. Its advantage is the use of dimensional information to learn a simpler model than would otherwise be possible. Because of the small size of the produced model and the small amount of data required to calibrate it, dimensional function synthesis is well-suited for execution on resource-constrained embedded systems.

3.1 Deriving the Dimensionless Product Groups

Let the set of base dimensions be $\mathbb{S}_{\text{base dimensions}}$. We assume without loss of generality that $\mathbb{S}_{\text{base dimensions}} = \{I, \Theta, T, L, M, J, N\}$ corresponding to the base S.I. dimensions for electric current, thermodynamic temperature, time, length, mass, luminous intensity, and amount of matter, respectively.

Let r be the cardinality of $\mathbb{S}_{\text{base dimensions}}$, let j be an index over r, and let $q_j \in \mathbb{S}_{\text{base dimensions}}$ be one of the base dimensions. As we



```
include "NewtonBaseSignals.nt"

Glider: invariant(t: time, h: distance, v0: speed, v: speed, m: mass, g: acceleration) = {
        #  Empty invariant body. Our implementation of dimensional function synthesis on top
        #  of Newton infers an expression for the body.

}
```

Object, mass m
velocity, v
Height at a given time, h
❶
h dimensions $\mathscr{D}(h) = L$
$v, v0$ dimensions $\mathscr{D}(v) = LT^{-1}$
m dimensions $\mathscr{D}(m) = M$
g dimensions $\mathscr{D}(g) = LT^{-2}$
t dimensions $\mathscr{D}(t) = T$
Multi-modal sensor data values
$h_1, v_1,$ ⋮ h_n, v_n
Dimensional Function Synthesis
Compile-time analysis
❷
$\frac{tg}{v_0}, \frac{h}{tv_0}, \frac{v}{v_0}$
$\frac{tg}{v}, \frac{h}{tv}, \frac{v}{v_0}$
$\frac{hg}{v^2}, \frac{h}{tv}, \frac{v}{v_0}$
$\frac{hg}{v_0^2}, \frac{h}{tv_0}, \frac{v}{v_0}$
$\frac{t^2g}{h}, \frac{h}{tv_0}, \frac{h}{tv}$
❸❹
Run-time calibration
❺❻
$\Phi(h/v_0{}^*t\ ,\ h/v^*t,\ t^{2*}g) = 0$
The actual physical law is $h = v_0{}^*t - 0.5(t^{2*}g)$

Figure 2: A glider of mass m launched with initial velocity v_0 moves through space with velocity v under gravitational acceleration g. Dimensional function synthesis can derive a set of candidate equations relating its height h to time t. Next, using sensor data, it can calibrate that set of candidate equations to obtain the model for height as a function of time and gravity.

did previously in Section 2.1 and Equation 1, let i be an index over the set of parameters for a physical system and let Q_i be one such parameter. Let a_{ij} be an exponent of one of the base dimensions of Q_i as returned by the function $\mathcal{D}$ from Section 2.1. We can express the dimensions of any Q_i in terms of the base dimensions q_j:

$$Q_i = q_1^{a_{i1}} q_2^{a_{i2}} \cdots q_r^{a_{ir}}. \tag{7}$$

We can represent the system of $n = |\mathbb{S}_{\text{symbols}}|$ equations, one for each of the $1 < i \leq n$ instances of Equation 7 with a matrix called the *dimensional matrix* [7, 10, 16].

Definition 2. *Let n be the number of parameters in $\mathbb{S}_{\text{symbols}}$ and let r be the number of fundamental dimensions required to express them. Let i be an index over the set of n parameters for a physical system and let j be an index over r. Then we define the dimensional matrix* $\mathbf{A}$, *as*

$$\mathbf{A} = (a_{ij})_{(r,n)}. \tag{8}$$

The products Π from Definition 1 and Equation 2 will be dimensionless (i.e., the dimensions in the monomial will cancel out) if and only if $\mathbf{Ak} = \mathbf{0}$, where the matrix k contains the exponents of the base dimensions needed to yield a dimensionless group. The solution of $\mathbf{Ak} = \mathbf{0}$ is the null space $N(\mathbf{A})$.

3.2 Physical Restriction on Solution of $N(\mathrm{A})$

Because of our objective of finding physically-plausible and efficiently-computable dimensionless groups, our goal is to restrict the solutions to the null space computation to rational powers of a_{ij} as opposed to permitting arbitrary real-valued exponents. As a result of this insight, we compute the *rational null space* of A which will by definition give us a_{ij} values that are ratios of small integers. To compute the rational null space of A, we first use Gauss-Jordan elimination [22] to reduce the matrices to their reduced row-echelon form (RREF), where all pivots equal one, with zeros below each pivot [28]. Once the matrix is in RREF, we find the special solutions to $\mathbf{Ak} = \mathbf{0}$. If, for a specific A, the only solution is the zero vector, then we conclude that no non-trivial null space is available and as a result it is not possible to form a dimensionless group from the set of parameters in $\mathbb{S}_{\text{symbols}}$.

The number of linearly-independent columns of the dimensional matrix A is equal to rank$(\mathbf{A})$. Thus, to find all possible solutions to $\mathbf{Ak} = \mathbf{0}$ and hence all possible groups of dimensionless products, we can rearrange the n columns of A in $\binom{n}{\text{rank}(\mathbf{A})}$ ways to yield different null space solutions [6][16]. Our final set of dimensionless product groups is the intersection of all the dimensionless product groups resulting from computing the null spaces.

Why this differs from traditional linear-algebraic formulations of dimensional analysis: The implementation we describe in Section 4.2 is the first system to completely automate identifying all possible dimensionless groups given a physical system description. We show in Section 4 that our implementation takes minimal time, less than 4 s when running single-threaded on a modern workstation, even for the largest dimensional matrices we encountered. Because this step is only performed once, at design time, this overhead is insignificant. Table 2 shows examples of five physical systems, their parameter sets $\mathbb{S}_{\text{symbols}}$, the dimensions of the parameters that are the members of $\mathbb{S}_{\text{symbols}}$, and examples of the dimensionless groups of those parameters which we compute from their dimensional matrix. In the general case the analysis results in more than one Π group. The remaining steps of the dimensional function synthesis must then fit a model to expressions derived from these Π groups.

3.3 Calibration: Using Sensor Data to Transform Π Groups to Equational Models

The dimensionless groups obtained offline by analyzing a description of the physical system in the form of the set $\mathbb{S}_{\text{symbols}}$ gives proportionality relations between the parameters in $\mathbb{S}_{\text{symbols}}$. Like any model construction method, dimensional function synthesis will produce incomplete results if the inputs to the method do not fully describe the problem being modeled: An incomplete $\mathbb{S}_{\text{symbols}}$ can result in an empty set of derived dimensionless products.

Table 2: Examples of physical system descriptions ($\mathbb{S}_{\text{symbols}}$) and the dimensionless groups our technique generates for them. Our implementation generates the LaTeX for the equations shown in the last column in colored text as one of its side effects.

Physical System	Input to Our Technique		Dimensions	Example of one Dimensionless Group Generated by Our Automated Method
Waves [11]	$\mathbb{S}_{\text{symbols}}$ =	Angular frequency, ω	$\mathcal{D}(\omega) = T^{-1}$	$\left\{\frac{(d^2)(\rho)(g)}{(\tau)}, \frac{(\omega^2)(d^3)(\rho)}{(\tau)}, \frac{(h)}{(d)}, \frac{(k)(h)}{1}\right\}$
	$\{\omega, k, h, d, \rho, \tau, g\}$	Length, k	$\mathcal{D}(k) = L^{-1}$	
		Length, h	$\mathcal{D}(h) = L$	
		Length, d	$\mathcal{D}(d) = L$	
		Density, ρ	$\mathcal{D}(\rho) = ML^{-3}$	
		Surface tension, τ	$\mathcal{D}(\tau) = MT^{-2}$	
		Acceleration, g	$\mathcal{D}(g) = LT^{-2}$	
Vibrating	$\mathbb{S}_{\text{symbols}}$ =	String tension, t	$\mathcal{D}(t) = MLT^{-2}$	$\left\{\frac{(L^2)(\mu)(f^2)}{(t)}, \frac{(t)}{(\mu)(f^2)(\rho^2)(\theta^2)}\right\}$
string	$\{t, L, \mu, f, \rho, \theta\}$	String length, L	$\mathcal{D}(L) = L$	
		String mass per unit length, μ	$\mathcal{D}(\mu) = ML^{-1}$	
		String vibration frequency, f	$\mathcal{D}(f) = T^{-1}$	
		Thermal expansion coefficient, ρ	$\mathcal{D}(\rho) = \Theta^{-1}$	
		String temperature, θ	$\mathcal{D}(\theta) = \Theta$	
Unpowered	$\mathbb{S}_{\text{symbols}}$ =	Object elevation, h	$\mathcal{D}(h) = L$	$\left\{\frac{(t^2)(g)}{(h)}, \frac{(h)}{(t)(v_0)}, \frac{(h)}{(t)(v)}\right\}$
flying object	$\{h, v_0, v, m, g, t\}$	Object initial velocity, v_0	$\mathcal{D}(v_0) = LT^{-1}$	
		Object velocity, v	$\mathcal{D}(v) = LT^{-1}$	
		Object mass, m	$\mathcal{D}(m) = M$	
		Acceleration due to gravity, g	$\mathcal{D}(g) = LT^{-2}$	
		Time, t	$\mathcal{D}(t) = T$	
Fluid	$\mathbb{S}_{\text{symbols}}$ =	Pressure gradient, ∇P	$\mathcal{D}(\nabla P) = ML^{-2}T^{-2}$	$\left\{\frac{(v^2)(\rho)}{(\nabla P)(D)}, \frac{(v^2)(\rho)}{(\nabla P)(e)}, \frac{(v^3)(\rho)}{(\nabla P)(\nu)}\right\}$
flow in pipe	$\{\nabla P, v, D, e, \nu, \rho\}$	Fluid velocity, v	$\mathcal{D}(v) = LT^{-1}$	
		Pipe diameter, D	$\mathcal{D}(D) = L$	
		Pipe roughness, e	$\mathcal{D}(e) = L$	
		Fluid kinematic viscosity, ν	$\mathcal{D}(\nu) = L^2T^{-1}$	
		Fluid density, ρ	$\mathcal{D}(\rho) = ML^{-3}$	
Pendulum	$\mathbb{S}_{\text{symbols}}$ =	Rod length, l	$\mathcal{D}(l) = L$	$\left\{\frac{(g)(t^2)}{(l)}\right\}$
	$\{l, g, m, t\}$	Acceleration due to gravity, g	$\mathcal{D}(g) = MT^{-2}$	
		Mass, m	$\mathcal{D}(m) = M$	
		Oscillation period, t	$\mathcal{D}(t) = T$	

When a dimensionless product group contains a single item, Equation 6 (Section 2.2) showed that we can equate the dimensionless product to a constant and hence obtain an equation between the symbols in the dimensionless group. We however still need to determine the value of the constant and we can do so given one or more values of the parameters in the dimensionless group. We call this step *calibration.*

When a dimensionless product group contains more than one dimensionless product, we can still apply this method if we can determine that all but one of the products in any of the dimensionless groups are effectively constant for the range of values of the parameters of interest.

If there is more than one dimensionless product which is not constant, then, from Equation 4, there is a function Φ' which relates the values of one of the Π products to the rest of them; a data-driven approach can then be used to find the form of Φ'. In this case the compile time step of dimensional function synthesis plays an important role of dimensionality reduction of the input data. This allows simpler models to perform better, allowing smaller models to be learned with less data for a given prediction performance.

3.4 Example

Figure 3 shows a system comprising a pendulum instrumented with a sensor that measures movement. By measuring, e.g., angular movement with a gyroscope or by measuring acceleration with an accelerometer, we can measure the period of oscillation t by computing the Fourier transform of time series data from the sensor. Our goal is to obtain a model relating t, the length of the rod l, and the component g of the acceleration due to gravity in the plane

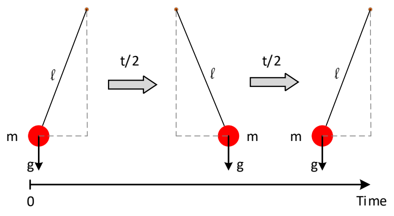


Figure 3: A simple pendulum with mass m, rod of length l, period of swing t, and with the component of the acceleration due to gravity in its plane of motion being g.

of rotation of the pendulum. The insights from this example are applicable to many sensor-instrumented mechanical systems such as ones where the period of oscillation might be affected when lengths of system parts expand or contract with temperature, or when the component of gravitational acceleration affecting the system changes due to the system being tilted at an angle.

Step 1: Dimensional matrix construction. For this system, the parameter set is $\mathbb{S}_{\text{symbols}} = \{l, g, m, t\}$. The last row of Table 2 shows the dimensions of the members of the parameter set $\mathbb{S}_{\text{symbols}}$ along with the dimensionless group computed by the method described above in Section 3.1. Following the formulation in Section 3.1, the dimensional matrix $\mathbf{A}$ for the pendulum's parameter set $\mathbb{S}_{\text{symbols}}$ is

$$\mathbf{A} = \begin{array}{c} \\ T \\ L \\ M \end{array} \begin{array}{c} \begin{array}{cccc} l & g & m & t \end{array} \\ \begin{bmatrix} 0 & -2 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Step 2: Dimensional matrix column permutation and Π group computation. The total number of parameters is $n = |\mathbb{S}_{\text{symbols}}| = 4$. The rank of dimensional matrix $\mathbf{A}$ is $\text{rank}(\mathbf{A}) = 3$ and there are $\binom{n}{\text{rank}(\mathbf{A})} = \binom{4}{3} = 4$ number of ways to permute the columns of the dimensional matrix to yield different null space solutions. The four permuted dimensional matrices are:

$$\mathbf{A}_1 = \begin{bmatrix} -2 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{A}_2 = \begin{bmatrix} 0 & 0 & 1 & -2 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{A}_3 = \begin{bmatrix} 0 & -2 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{A}_4 = \begin{bmatrix} 0 & -2 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

We then apply the Gauss-Jordan algorithm to reduce these matrices to reduced row-echelon form:

$$\text{RREF}(\mathbf{A}_1) = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{bmatrix} \text{RREF}(\mathbf{A}_2) = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -2 \end{bmatrix}$$

$$\text{RREF}(\mathbf{A}_3) = \begin{bmatrix} 1 & 0 & 1/2 & 0 \\ 0 & 1 & -1/2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{RREF}(\mathbf{A}_4) = \begin{bmatrix} 1 & 0 & 0 & 1/2 \\ 0 & 1 & 0 & -1/2 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

We then calculate the null space solutions $\mathbf{N}(\mathbf{A}_i)$ $(i = 1, 2, 3, 4)$ for each corresponding RREF dimensional matrix.

$$\mathbf{N}(\mathbf{A}_1) = \begin{bmatrix} -1 \\ 0 \\ -2 \\ 1 \end{bmatrix} \begin{array}{c} g \\ m \\ t \\ l \end{array} \quad \mathbf{N}(\mathbf{A}_2) = \begin{bmatrix} -1 \\ 0 \\ 2 \\ 1 \end{bmatrix} \begin{array}{c} l \\ m \\ t \\ g \end{array}$$

$$\mathbf{N}(\mathbf{A}_3) = \begin{bmatrix} -1 \\ 1 \\ 2 \\ 0 \end{bmatrix} \begin{array}{c} l \\ g \\ t \\ m \end{array} \quad \mathbf{N}(\mathbf{A}_4) = \begin{bmatrix} -1 \\ 1 \\ 0 \\ 2 \end{bmatrix} \begin{array}{c} l \\ g \\ m \\ t \end{array}$$

The null space solutions are functionally identical. From Definition 1 (Section 2.2), the pendulum system has $n = 4$ physical quantities and $r = 3$ base dimensions. Consequently, $n - r = 1$ and there is a single unique kernel:

$$\Pi_0 = \begin{array}{c} \begin{array}{cccc} g & l & m & t \end{array} \\ \begin{bmatrix} 1 & -1 & 0 & 2 \end{bmatrix} \end{array}$$

From Equation 6 (Section 2.2), it follows that this kernel equals some constant C:

$$\frac{g * t^2}{l} = C. \tag{9}$$

Given sensor measurements for different values of l, g, and t, we can determine the value of the constant C.

4 IMPLEMENTATION AND EVALUATION

We implemented the method of Section 3 by extracting the set $\mathbb{S}_{\text{symbols}}$ from the intermediate representation of descriptions of physical systems written in Newton [20], a domain-specific language for describing physical systems. We use Newton solely as a convenient way to obtain the set $\mathbb{S}_{\text{symbols}}$ from a human-readable description. Figure 4 shows the flow of our implementation.

4.1 Dimensional Function Synthesis versus Function Evaluation

Dimensional function synthesis includes a compile-time step comprising Π group generation followed by a run-time calibration step of function fitting. In the case of a single Π group and under the constraints covered in Section 5.2, this reduces to finding a constant. When the compile-time step results in multiple Π groups, the run-time step involves using measurement data to calibrate a model relating the multiple Π groups (i.e., not simply learning a proportionality constant). A person using an implementation of dimensional function synthesis invokes the method once to generate a model. An embedded system on which the model is subsequently deployed may execute the model billions of times (or more).

4.2 Implementation Methodology

Compile-time analysis phase: The input of dimensional function synthesis is the set $\mathbb{S}_{\text{symbols}}$. The first step (**Step 1**) creates the dimensional matrix from the list of members of $\mathbb{S}_{\text{symbols}}$ and their dimensions. Next (**Step 2**), the method permutes the columns of the dimensional matrix and computes the null space of the permuted matrices to obtain the dimensionless Π groups. As the example

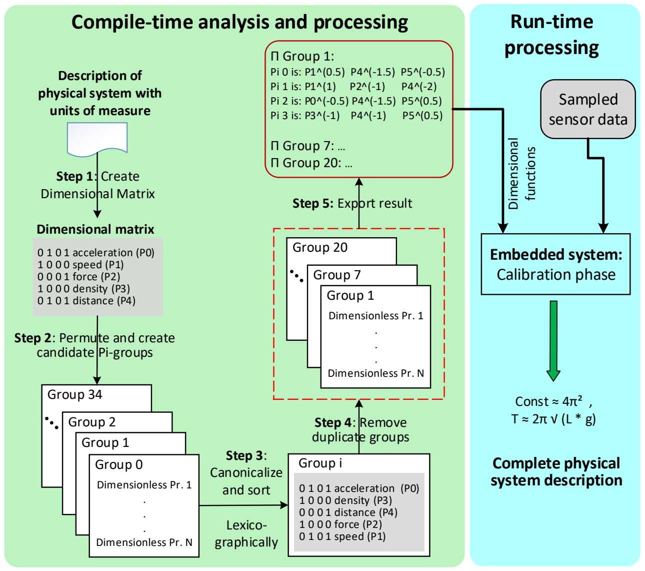

Compile-time analysis and processing
Run-time processing
Description of physical system with units of measure
Step 1: Create Dimensional Matrix
Dimensional matrix
0 1 0 1 acceleration (P0)
1 0 0 0 speed (P1)
0 0 0 1 force (P2)
1 0 0 0 density (P3)
0 1 0 1 distance (P4)
Step 2: Permute and create candidate Pi-groups
Group 34
Group 2
Group 1
Group 0
Dimensionless Pr. 1
Dimensionless Pr. N
Step 3: Canonicalize and sort
Lexico-graphically
Group i
0 1 0 1 acceleration (P0)
1 0 0 0 density (P3)
0 0 0 1 distance (P4)
1 0 0 0 force (P2)
0 1 0 1 speed (P1)
Step 4: Remove duplicate groups
Group 20
Group 7
Group 1
Dimensionless Pr. 1
Dimensionless Pr. N
Step 5: Export result
Π Group 1:
Pi 0 is: P1^(0.5) P4^(-1.5) P5^(-0.5)
Pi 1 is: P1^(1) P2^(-1) P4^(-2)
Pi 2 is: P0^(-0.5) P4^(-1.5) P5^(0.5)
Pi 3 is: P3^(-1) P4^(-1) P5^(0.5)
Π Group 7: ...
Π Group 20: ...
Dimensional functions
Sampled sensor data
Embedded system: Calibration phase
Const ≈ 4π² ,
T ≈ 2π √ (L * g)
Complete physical system description

Figure 4: Our implementation of the dimensional function synthesis method of Section 3 extracts the set $\mathbb{S}_{\text{symbols}}$ from the intermediate representation of descriptions of physical systems written in Newton [20]. Next, it uses the linear-algebraic steps described in Section 3.1 to compute all possible dimensionless groups. Finally, we use sensor data to calibrate the compile-time-generated functions.

in Section 3.4 showed, there may be redundant Π groups. The method therefore canonicalizes the representation of the computed Π groups by transforming the Π groups so that lexicographical smaller parameter names are always in the numerator of all Π product monomials. The method next sorts the canonicalized Π products in each Π group (**Step 3**). With the Π products in each group in canonical lexicographic form and sorted, the method removes duplicate Π groups (**Step 4**). The remaining unique Π groups constitute the synthesized dimensional proportionality relations. These proportionality relations can then be calibrated using sensor data to obtain compact models relating the signals in the original input set $\mathbb{S}_{\text{symbols}}$ (**Step 5**).

Run-time phase: The output of the final stage of the compile-time analysis, in the form of the set of Π groups, is directed to the calibration phase and processed according to which of the conditions satisfied by Section 3.3 the system falls into. In the case where there is only a single Π product, the calibration process uses sensor data to find the constant of proportionality which relates the Π products. When there is more than one Π product, the calibration process uses sensor data in combination with an approximate function estimator to find an approximate relationship between the Π products. The output after this stage is a model which can be used for inference during the normal operation of the embedded system. Because, in addition to function evaluation, our objective is to perform the run-time phase of function synthesis on the target resource-constrained computing system, we have explored the use of simple regression models (e.g., linear or quadratic regression) for the approximate function estimator. Dimensional function synthesis is however a generally-applicable method, suitable for integration with more complex function estimators such as neural networks. When paired with neural networks, a user of dimensional function synthesis can decide whether the training of the network will be performed on a powerful computing device or on the target embedded hardware platform, if it can support it.

To evaluate the performance of dimensional function synthesis, we consider two metrics: (1) the additional one-time cost of running the compile-time stage to find the dimensionless groups in Section 4.3 and (2) the performance of the models trained using this technique in Section 5.1 (after both compile time Π group synthesis and run-time calibration/function approximation).

4.3 Implementation Evaluation

We first evaluate the performance of the compile-time dimensional function synthesis. We then evaluate the calibration overhead, inference latency, and prediction accuracy of the models our method generates using 90 laboratory experiments which we augment with 10 000 synthetic idealized measurements.

Evaluation setup: We evaluate our implementation of dimensional function synthesis by applying it to the specifications of 16 physical systems. Table 3 lists these benchmark physical systems along with their descriptions, demonstrating the variety of systems able to be described by means of their physical parameters. We provide information about the number of parameters that describe each system (e.g., mass, time, length, and gravitational acceleration in the ideal pendulum example), as well as the size of the resulting dimensional matrix. In most of the systems, there are fewer rows than columns because there are fewer unique units of measurement than physical parameters. We evaluate the run time of our implementation using nanosecond-granularity timestamps provided by the operating system. We perform the evaluation on an Intel Core i3 processor operating at 3.6 GHz, equipped with 8 GB of DDR4 memory and a solid-state disk drive.

Compile-time model synthesis effectiveness and overhead: Table 4 shows the results from applying the compile-time steps of dimensional function synthesis to the physical systems of Table 3. For example, for the physical system describing a selective laser melting system (second row from the bottom row) there are 1365 Π groups that result from computing the null space of all the possible permutations of the columns of the dimensional matrix (**Step 2** of the compile-time phase of the method). After canonicalization, sorting and removing duplicates (**Step 3 and 4** of the compile-time phase of the method), this reduces to 94 Π groups. For this system, the Π groups have 11 dimensionless products or Πs per group. In all the physical systems listed in Table 4, the compile-time stage completes in under 4 s, with most cases taking under 10 ms.

Figure 5 shows the breakdown of the time spent in the five compile-time phases of the dimensional function synthesis method, averaged over five executions of the benchmarks of Table 4. The core steps of the method (**Step 2:** computing the null space for all permutations; **Step 3:** canonicalization, and **Step 4:** duplicate removal) require less than 3% of the total execution latency. **Step 5:** corresponds to the export latency of the toolflow.

Table 3: The specifications of 16 different physical systems, the cardinalities of their symbol sets $|\mathbb{S}_{\text{symbols}}|$, and the size of their dimensional matrices A.

Name	Description	Number of parameters, $\lvert\mathbb{S}_{\text{symbols}}\rvert$	Size of the dimensional matrix, $\lvert A\rvert$
Cable hydrodynamic drag force	Hydrodynamic drag force for a cable	6	6 columns × 3 rows
Explosion, v1	Explosion model [29] including explosion radius	5	5 columns × 3 rows
Explosion, v2	Explosion model [29] excluding explosion radius	4	4 columns × 3 rows
Hot sphere	Heat transfer between a hot sphere and the environment	8	8 columns × 4 rows
Ideal gas	Ideal gas model	4	4 columns × 4 rows
Pendulum, static	Simple pendulum excluding dynamics and friction	4	4 columns × 3 rows
Pendulum, dynamic	Simple pendulum including dynamics and friction	8	8 columns × 3 rows
Fluid in Pipe, v1	Pressure drop of a fluid through a pipe [11]	6	6 columns × 3 rows
Fluid in Pipe, v2	Pressure drop of a fluid through a pipe [11] substituting dynamic viscosity with kinematic viscosity	6	6 columns × 3 rows
Unpowered flight	Unpowered flight (e.g., catapulted drone)	4	4 columns × 3 rows
Vibrating string, v1	Vibrating string	4	4 columns × 3 rows
Vibrating string, v2	Vibrating string with model of thermal expansion	6	6 columns × 4 rows
Fluid wave dispersion	Dispersion of waves in a fluid	7	7 columns × 3 rows
Spring and mass system	Vertical spring with attached mass	4	4 columns × 3 rows
Selective Laser Melting (SLM), v1	Selective laser melting 3D printer with sensors	15	15 columns × 4 rows
Selective Laser Melting (SLM), v2	Selective laser melting 3D printer with sensors	11	11 columns × 4 rows

Table 4: The static compile-time analysis phase of dimensional function synthesis, applied to 16 problems, implemented based on Section 3.

Name	# of Π groups	Πs per Π group	Unique Π groups	Analysis time
Hydrodynamic drag	20	3	3	7.4 ms
Explosion, v1	10	2	6	8.8 ms
Explosion, v2	4	1	1	1.5 ms
Hot sphere	70	4	23	81.5 ms
Ideal gas	4	1	1	1.5 ms
Pendulum, static	4	1	1	1.5 ms
Pendulum, dynamic	56	5	25	104.1 ms
Fluid in pipe, v1	20	3	11	23.7 ms
Fluid in pipe, v2	20	3	8	17.1 ms
Unpowered flight	4	1	1	1.5 ms
Vibrating string, v1	4	1	1	1.4 ms
Vibrating string, v2	15	2	2	4.2 ms
Wave dispersion	35	4	8	25 ms
Spring and mass	4	1	1	1.4 ms
SLM, v1	1365	11	94	834.6 ms
SLM, v2	165	8	106	3428.7 ms

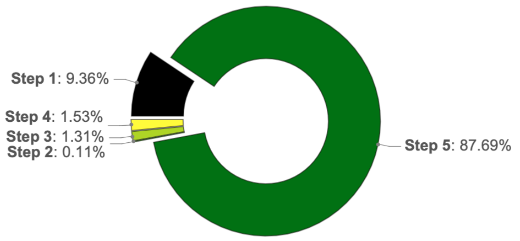


Figure 5: Latency percentage breakdown for the compile-time steps of the dimensional function synthesis. The core steps constitute less than 3% of the overall time.

5 MODEL EVALUATION

We demonstrate the advantages of constructing models from sensor data using our method by using it to derive functions modeling properties of three physical systems:

❶ A pendulum with rod length l, component of the acceleration due to gravity g in the plane of swing of the pendulum, and period of oscillation t. We run experiments for several values of l and g to obtain values for the period of oscillation t.

❷ An oscillating mass-loaded spring with period of oscillation t, spring constant K_{spring}, and mass m attached to the spring.

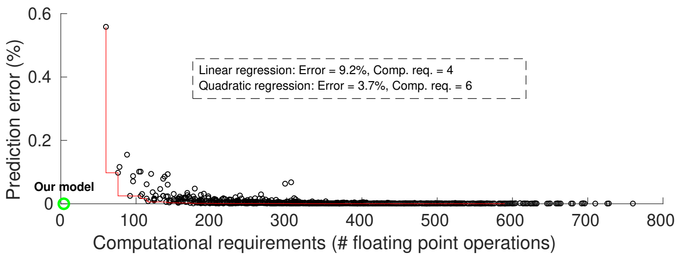
❸ An unpowered flying vehicle (glider) with initial velocity v_0, mass m, acceleration due to gravity g, and height h at time t.

For the pendulum system, we evaluate the ability of our method to build a model for t as a function of l and g. For the spring, we evaluate the ability of our method to build a model for K_{spring}, as a function of m and t. And for the glider we examine the ability of our method to find the relation between trajectory height and the other variables.

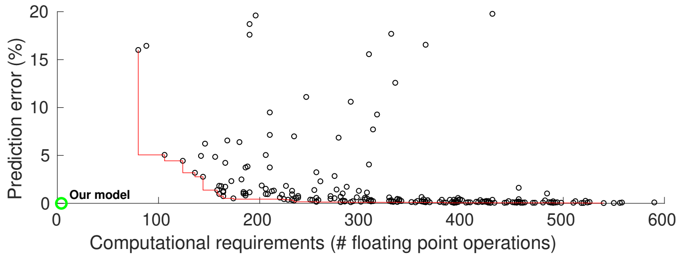
Because our target domain is resource-constrained embedded systems, we focus our evaluation on the ability of our method to produce models with minimal training and to produce models whose computational requirements during inference are small. We compare our method against regression over the data, as well as against training several neural network topologies from the FitNet family of curve-fitting neural network architectures, which are optimized for equation fitting.

5.1 Model Evaluation Results

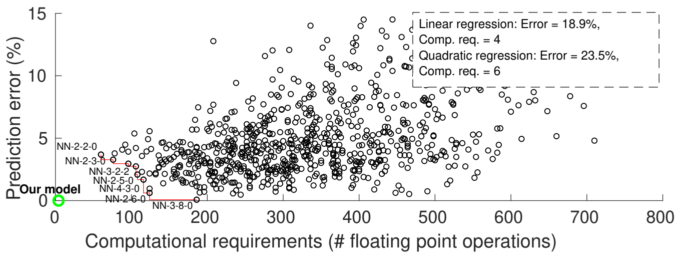
We first compare dimensional function synthesis to regression and neural networks using synthetic idealized data (Section 5.1.1). For the multiple Π product relationship such as the glider example (Figure 2), the calibration stage of dimensional function synthesis also involves a data-driven function approximation (Section 5.1.2).



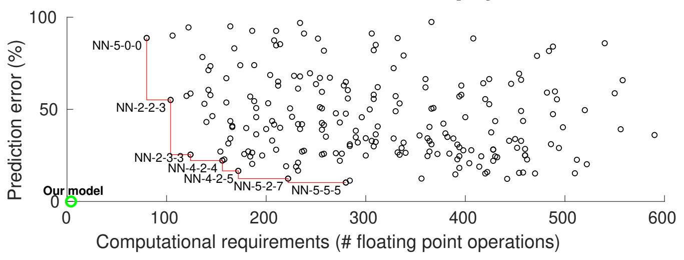
(a) In-sample error, neural network: pendulum period.



(b) In-sample error, neural network: spring oscillator constant K_{spring}.



(c) Out-of-sample error, neural network: pendulum period.



(d) Out-of-sample error, neural network: spring oscillator constant K_{spring}.

Figure 6: Prediction error versus computational requirements for predicting the period of pendulum (subfigures (a) and (c), 810 different network topologies) and K_{spring} constant of the spring oscillator (subfigures (b) and (d), 265 different network topologies). The Pareto front is indicated in red. Subfigures (a) and (b) are evaluated on data within the range of the training set. Subfigures (c) and (d) are evaluated on data outside of the training range (out-of-sample performance). Our method Pareto-dominates all the neural network variants ("our model" in lower left corner of the plots).

We then present an evaluation using laboratory experiments in Section 5.1.3.

5.1.1 Evaluation results for synthetic data. Our evaluation methodology for synthetic data includes two steps. First, we train the models against 10 000 input vectors generated based on the ideal equations for the pendulum and spring systems. For the pendulum, the data are vectors of $\{t, l, g\}$ where $t = 2\pi\sqrt{l/g}$. For the spring, the data are vectors of $\{t, K_{\text{spring}}, m\}$ where $t = 2\pi/\sqrt{K_{\text{spring}}/m}$. We refer to the range of the values we use for $\{l, g\}$ for the pendulum training and $\{t, m\}$ for the spring oscillator as being *inside the training range* and we refer to the performance of the models for parameters in the training range as the *in-sample performance*. Next, we evaluate the trained models against new input vectors, resulting from values of the controllable parameters *outside the training range*. We refer to the performance of the models for parameters outside the training range as the *out-of-sample performance*.

We apply three data-driven modeling techniques on the training data: linear regression, quadratic regression, and neural networks. For the neural networks, we evaluate networks with a range of hidden layers and a range of number of nodes per hidden layer. We performed 5-fold cross validation of the models.

Neural networks: Figure 6 shows the prediction error versus the computational requirements of the neural network models for predicting the period t of the pendulum (Figure 6a and Figure 6c) and the spring constant K_{spring} of the oscillating spring (Figure 6b and Figure 6d).

For each examined neural network, we calculate the total number of floating-point operations (additions, multiplications) required per inference instance. We use this information to quantify the computational requirements of each network, independently of the target hardware execution platform. We annotate points on the Pareto frontier with information about the number of nodes in each of the network's three hidden layers. Thus, e.g., NN-5-5-5 indicates a network of three hidden layers and five nodes per layer, while NN-5-0-0 indicates a network with only one hidden layer comprising 5 nodes.

Figures 6a and 6b show the in-sample performance Pareto front of the design exploration for the two examined physical systems. In both systems, because our method is able to generate simple functions that exactly match the equations of the physical equation used to generate the training data, our method achieves lower computational cost and lower prediction error than all points on the Pareto frontier of the neural network model results.

The neural network models achieve significantly different in-sample and out-of-sample performance (top row versus bottom row in Figure 6). For the pendulum (Figure 6c), although the prediction errors span a large range, there are configurations that achieve prediction error below 5%. For the spring-mass system (Figure 6d), the prediction error is greater than 50% in more than half of the evaluated models. The smallest prediction error (under 10%) in the spring-mass system is achieved by the network labeled NN-5-5-5. Its smaller prediction error comes at the cost of greater computation

(280 floating-point operations) compared to NN-5-0-0 (80 floating-point operations) and compared to our method (4 floating-point operations). Overall, the neural network models require between 1 s and 30 s training per model, with an average of 13 s. The total training latency was approximately 170 minutes for the pendulum model, and 60 minutes for the spring model, on an Intel Core i7-7820X CPU running at 3.60 GHz, equipped with 32 GB RAM. This is 8660× slower than our approach which requires 1.5 ms on average for the two examined physical systems (Table 4) running on the same workstation. The neural network models require between 60 and 280 floating-point operations per inference, with average inference cost of 34× worse than our method.

Linear and quadratic regression: We next compare the overhead and prediction error of the model our method generates for the pendulum period against models generated by linear and quadratic regression. Both linear and quadratic regression models show in-sample prediction error of less than 10% but exhibit out-of-sample prediction errors of 18.9% and 23.5% respectively and require 4 and 6 floating-point operations, respectively, per inference. In contrast, our method learns the original equation used to generate the data exactly and requires only 4 floating-point operations per prediction.

5.1.2 Evaluation results for multiple Pi-groups. The previous examples outlined the ability of dimensional function synthesis to derive the exact form of the equation of a physical system when the dimensional matrix results in a single dimensionless product. However, the parameters used to describe the physical systems under analysis may result in multiple Π groups which each include multiple Π products. The glider example of Section 2.1 is one such example. In this case, the form of the function Φ' for combining the Π products into an equational model is unknown and a data-driven approach must be used to find its form. Dimensional function synthesis provides two options for the calibration phase: (1) perform calibration on the target embedded system; (2) perform calibration offline on a computing system that is not constrained by resources. In both cases the calibrated models target the embedded platform, so final model complexity is still a key restriction.

In contrast to Φ and Φ' which are functions of dimensionless products, let Ψ be a function directly relating the parameters of a system. For the glider example, we compare our approach to a data-driven approach for fitting the feature vector $\langle v_0, m, g, t\rangle$ to a predicted height h through the function Ψ:

$$h = \Psi(v_0, m, g, t). \tag{10}$$

The ideal trajectory equation of a glider is $h = v_0 \cdot t - 0.5 \cdot (t^2 \cdot g)$. Using the ideal trajectory equation, we synthesize a dataset by uniformly sampling the initial velocity of the glider (v_0) in the range of 1 m/s to 10 m/s with a step size of 0.5 m/s. We considered acceleration due to gravity (g) from 6.0 m/s^2 to 9.5 m/s^2, with 0.5 m/s^2 step size, and a time window for gliding (t) ranging from of 0.1 s to 100 s, with a step of 0.1 s.

Using dimensional function synthesis, the chosen description of the system leads to three Π groups, each with two Π products, i.e., Π **group 0** = $\{\Pi_1 = t \cdot g/v_0, \Pi_2 = h/t \cdot v_0\}$, Π **group 1** = $\{\Pi_1 = h \cdot g/v_0^2, \Pi_2 = h/t \cdot v_0\}$, Π **group 2** = $\{\Pi_1 = t^2 \cdot g/h, \Pi_2 = h/t \cdot v_0\}$.

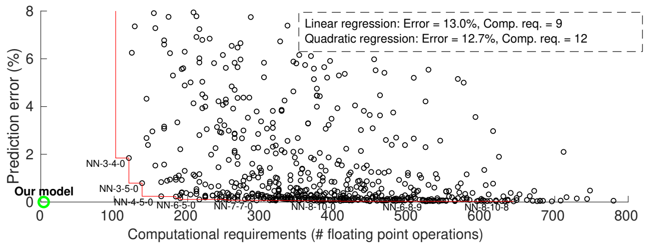


Figure 7: Prediction error versus computational requirements for predicting the trajectory of a glider. Our model uses linear regression for fitting function Φ' of Equation 11 (denoted as "our model" in the lower left corner). It Pareto-dominates all the neural network variants (891 different network topologies), which are used for fitting function Ψ of Equation 10.

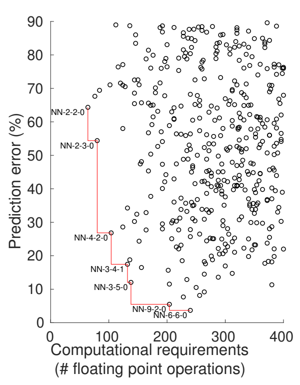
In Π group 0, h is included in only one Π, thus according to Equation 4 we can express h as function Φ' of the other Π:

$$\frac{h}{t \cdot v_0} = \Phi'(\frac{t \cdot g}{v_0}). \tag{11}$$

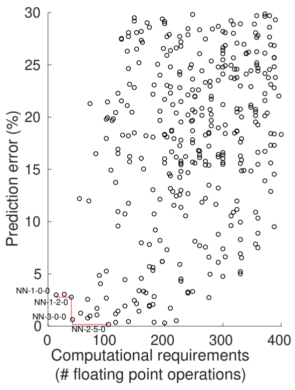
In contrast to traditional methods that must learn a function over a four-dimensional space $\langle v_0, m, g, t\rangle$, dimensional function synthesis only needs to use data to learn the function of the single-variable function Φ' of Equation 11. This simpler form is particularly valuable when our goal is to perform final calibration on a resource-constrained embedded system. Figure 7 shows the comparative performance of using a linear regression to find the dimensionally reduced Φ', against linear, quadratic, and neural network based regression to find Ψ. Linear regression on Φ' outperforms the same technique on Ψ by more than 12%, while having similar computational requirements. Neural networks are capable of minimizing the prediction error, at the expense of over 80× greater required computation.

Figure 8 shows model approximation performed by neural networks trained against 20 data points, with (Figure 8b) and without (Figure 8a) dimensionally reducing the number of input parameters by making use of dimensional function synthesis. The prediction error of the model produced from dimensional function synthesis is 0.17% and it is at the same time much simpler. The most accurate neural network for fitting the function Φ' over the four-dimensional space $\langle v_0, m, g, t\rangle$ is composed of two layers with 2 and 5 neurons, while the most accurate for fitting function Ψ is composed of two layers of 6 neurons each. This highlights dimensional function synthesis as a tool for training models in situations were there are insufficient data to train more complex models.

The simpler models and higher prediction accuracy of dimensional function synthesis is the result of being able to use the physical information available. This enables simpler models which train better with less data. Most importantly, these reductions are not based on ad-hoc assumptions or approximations, but are dictated by physical laws. This results in models from dimensional function synthesis which can be executed more efficiently on resource-constrained embedded systems as they require less computation during inference and less data to train.



(a) Black box approach.



(b) Our approach.

Figure 8: Prediction error versus computational requirements for predicting the trajectory of a glider. Sub-figure (a) corresponds to the straightforward application of neural networks for fitting function Ψ of Equation 10. Sub-figure (b) corresponds to our approach using a neural network approximation function for fitting function Φ' of Equation 11. All models are trained against a set of 20 input data points. Our method achieves prediction error of 0.17% via an approximately 2.5$\times$ less computationally demanding model.

5.1.3 Evaluation results for physical pendulum experiments: We evaluate our method in the presence of non-synthetic data where the underlying relationship is more complex than a simple closed-form equation. We perform a series of experiments in our laboratory and we vary the component of g and l using an apparatus known as a *variable-g pendulum* (Figure 9a). This apparatus uses a mass on a stiff rod swinging about a pivot which is at an angle that is not perpendicular to the horizon. We instrument this apparatus with a wireless sensor containing a 3-axis accelerometer at the "bob" end of the pendulum to provide a data stream from which we automate measuring the period of oscillation, t. We run 90 physical experiments on this apparatus for different values of the pendulum rod length l in the range of 3 cm to 33 cm in steps of 3 cm and for a range of effective gravitational acceleration g resulting from pendulum pivot angles of 0° to 80°, in 10° increments.

Figure 9b shows an example of the sensor data over 1 minute of pendulum operation. We recorded a time series of pendulum swing data such as that in Figure 9b for each of the 90 experiments we performed. We then used this time series data to calculate the oscillation period via its Discrete Fourier Transform (DFT). Figure 9c shows the resulting DFT output for one experiment, for four different processing windows. Figure 9d shows the oscillation period over the duration of one 1-minute experiment, calculated from the DFT. Figure 9d shows how the period of oscillation diminishes slightly over the course of a 1-minute experiment.

Figure 10 shows the ability of our method to generate a model that accurately predicts the period of oscillation of the variable-g pendulum. The calibration step of our method takes as input the periods measured from the actual experiment. Our method requires minimal calibration data. For pendulum lengths greater than 20 cm, the prediction error is always less than 15% even though each prediction requires only 4 floating-point operations. This error is within the range achieved by the neural networks (Figure 6c) but which require two orders of magnitude more floating-point operations.

For pendulum lengths less than 20 cm, the error in the model increases as non-idealities such as friction are not captured by the form of the proportionality relation generated by the compile-time step of our technique. The accuracy of the synthesized dimensional function is limited by the number of utilized parameters which describe the physical system. A richer choice in the set of parameters, e.g., including the friction of the pivot and mass of the rod is a possible solution to derive more accurate dimensional functions.

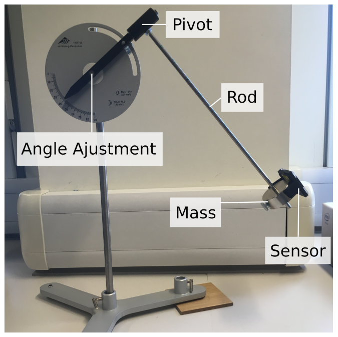
We also applied the black-box data driven techniques on the assembled data of the pendulum experiment. Of this dataset, 75% was randomly sampled to act as training data, while the rest were used as testing samples. We used a 5-fold cross validation policy to build the models. Figure 11 summarizes the prediction error of the period of pendulum oscillation averaged for all models in the case of the testing dataset. Regression models have prediction error comparable to our method but our method outperforms regression models in the zoomed area of Figure 10b. Neural networks exhibit a wide distribution of prediction error, but simple networks are able to achieve very high accuracy within the same range as our proposed model. Because the black-box models are trained against data points derived from the entire range of the pendulum experiments, they can effectively capture the non-ideal characteristics of the oscillation, thus achieving high accuracy.

5.2 Scope, Limitations and Extensions

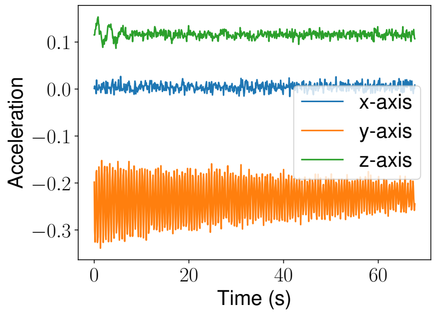
Dimensional function synthesis uses information on the physical dimensions and units of measure of the signals relevant to a physical system, to derive a set of candidate equations relating those signals.

Like many existing approaches for constructing models based on human-chosen parameters, dimensional function synthesis depends on a valid set of parameters in the set $\mathbb{S}_{\text{symbols}}$ (introduced in Section 2.1) for describing the system to be modeled. When provided with a set of parameters insufficient to generate a model that captures a system's behavior, the method will unsurprisingly generate a model that is, at best, only an approximation to the true behavior. Exciting areas of further development include automating the process of identifying parameters in $\mathbb{S}_{\text{symbols}}$ rather than extracting them from a human-written description and incorporating integrals and derivatives in formulations for Φ.

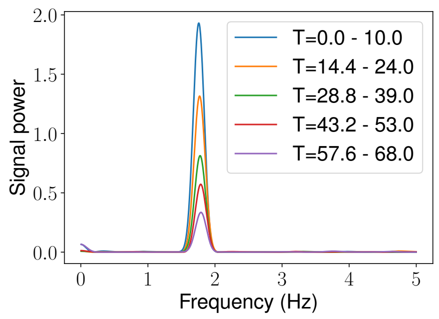
For physical parameters that cannot be directly measured, dimensional function synthesis faces the same challenges faced by traditional modeling approaches. In practice, for parameters that cannot be measured, designers measure surrogates which correlate to the missing parameters, e.g., measuring acceleration and elapsed time instead of velocity. In this case, dimensional function synthesis has the net effect of exploiting information on the physical units of the parameters in question, while traditional modeling techniques have no option but to attempt to fit data with ever more complex non-linear models. Dimensional function synthesis enables the combination of both approaches in the case of multiple Π groups as examined in Section 5.1.2.



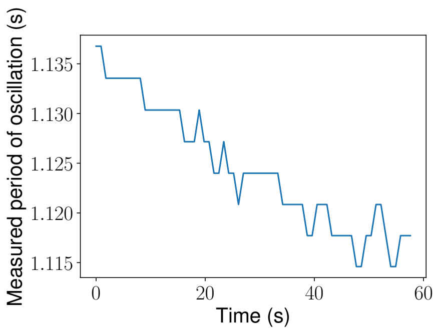
(a)



(b)

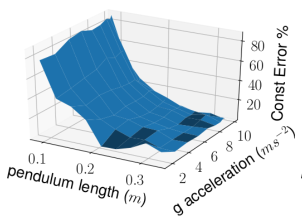


(c)

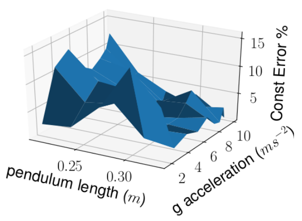


(d)

Figure 9: (a) Our experimental setup for the variable-g pendulum. (b) Data collected from the 3-axis accelerometer over time by the wireless sensor on the pendulum. The largest component of oscillation is due to the motion of the pendulum. (c) Discrete Fourier Transform (DFT) of 10 s windows of the sampled acceleration data. Despite the variation of signal properties over time, the dominating frequency remains around 2 Hz. (d) The time period of the pendulum, calculated according to the dominating frequency in each time window of DFT, exhibits a small variation of about 20 ms over a 1-minute interval.



(a) All experimental data.



(b) Zoom for $l > 20\,cm$.

Figure 10: Percentage error of the predicted period, t for a given length l and gravitational acceleration component in the plane of pendulum, g. Sub-figure (a) includes all experimental instances in a subset of which the ideal pendulum model assumptions are violated leading to high deviations. Sub-figure (b) zooms in the region of interest, where the error of synthesized dimensional functions is minimized.

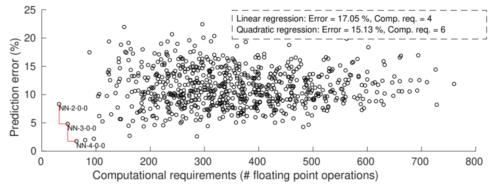


Figure 11: Percentage error of the predicted period, t for a given length l and gravitational acceleration component in the plane of pendulum, g. Prediction is performed using Neural Networks and regression models.

6 RELATED RESEARCH

The majority of existing research on applying dimensional analysis to computing systems has focused on extending the type systems of programming languages with support for units of measure [1–3, 5, 8, 9, 12, 14, 15, 17, 18, 23, 30]. This line of work, exemplified by the work of Kennedy [17, 18] can be thought of as tackling the problem of *verification* of dimensional homogeneity: how to validate that an equation containing a set of parameters is dimensionally homogenous. In contrast, in this work, we address the dual problem of *synthesis*: how to generate an equation that is dimensionally homogenous from a list of parameters.

In parallel to research on dimensions and units of measure in computing systems, there have been several efforts to learn physical laws or invariants from experimental measurements or sensor data. This line of research, exemplified by the work of Schmidt and Lipson [25] and extended by Hills [13], uses information about the general form of physical laws, such as the principle of stationary action, to infer functions relating physical signals from measurements of those signals. In contrast, the method we present in this work takes the unconventional approach of bootstrapping a function model purely from the constraints imposed by the principle of dimensional homogeneity. Unlike the work of early efforts to derive models based solely on dimensional analysis, such work by Buckingham [6], Jonsson [16], and Taylor [29], the approach we present is the first system to fully automate the computation of all possible dimensionally-consistent equations that could be formed from a set of dimensioned parameters.

Our comparison of our method against data-driven methods such as linear and polynomial model fitting and neural networks sheds new light on how prior heuristic approaches for inferring physical laws from measurement data (e.g., BACON [19] and DISCOVER [31]) are at the same time potentially computationally more expensive as well as potentially less accurate.

7 CONCLUSIONS

Existing methods for constructing retrospective or predictive models for data from physical systems do not fully exploit information about the physics of the systems in question. In this work, we present an automated method for generating the family of functions from which to learn a model, based on information about the physical dimensions of the signals in the system. The method, which we call *dimensional function synthesis*, applies to data streams where the dimensions of the signals are known.

We implement dimensional function synthesis and apply our implementation to 16 physical system descriptions. The results show that we can analyze even the largest of the 16 systems in under 4 s of execution time, with most of the analyses taking under 10 ms. For two of these 16 systems, we demonstrate efficiently summarizing multi-modal sensor data using 10 000 synthetic idealized measurements augmented with 90 laboratory experiments. We evaluate the model evaluation cost and accuracy of the models our method generates and compare it to models generated by linear regression, quadratic regression, and neural networks.

The results show that our technique can generate models in less than 300 ms on average across all the physical systems we evaluated. When calibrated with sensor data, our models outperform traditional regression and neural network models in inference accuracy in all the cases we evaluated. In addition, our models perform better in training latency (over 8660× improvement) and required arithmetic operations in inference (over 34× improvement). These significant gains are largely the result of exploiting information on the physics of signals that has hitherto been ignored.

ACKNOWLEDGMENTS

This research is supported by an Alan Turing Institute award TU/B/000096 under EPSRC grant EP/N510129/1, by Royal Society grant RG170136, and by EPSRC grants EP/P001246/1 and EP/R022534/1.

REFERENCES

[1] Allen, E., Chase, D., Luchangco, V., Maessen, J.-W., and Steele, Jr., G. L. Object-oriented units of measurement. In *Proceedings of the 19th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications* (New York, NY, USA, 2004), OOPSLA '04, ACM, pp. 384–403.

[2] Antoniu, T., Steckler, P. A., Krishnamurthi, S., Neuwirth, E., and Felleisen, M. Validating the unit correctness of spreadsheet programs. In *Proceedings of the 26th International Conference on Software Engineering* (Washington, DC, USA, 2004), ICSE '04, IEEE Computer Society, pp. 439–448.

[3] Babout, M., Sidhoum, H., and Frecon, L. Ampere: a programming language for physics. *European Journal of Physics 11*, 3 (1990), 163.

[4] Barber, D. *Bayesian reasoning and machine learning.* Cambridge University Press, 2012.

[5] Biggs, G., and Macdonald, B. A. A pragmatic approach to dimensional analysis for mobile robotic programming. *Auton. Robots 25*, 4 (Nov. 2008), 405–419.

[6] Buckingham, E. On physically similar systems; Illustrations of the use of dimensional equations. *Physical Review 4*, 4 (1914), 345–376.

[7] Carlson, D. E. A mathematical theory of physical units, dimensions, and measures. *Archive for Rational Mechanics and Analysis 70*, 4 (1979), 289–305.

[8] Chen, F., Roşu, G., and Venkatesan, R. P. Rule-based analysis of dimensional safety. In *Proceedings of the 14th International Conference on Rewriting Techniques and Applications* (Berlin, Heidelberg, 2003), RTA'03, Springer-Verlag, pp. 197–207.

[9] Cmelik, R. F., and Gehani, N. H. Dimensional analysis with c++. *IEEE Softw. 5*, 3 (May 1988), 21–27.

[10] E. Carlson, D. On Some New Results in Dimensional Analysis.

[11] Hanche-Olsen, H. BuckinghamâĂŹs pi-theorem. *NTNU: http://www. math. ntnu. no/~ hanche/notes/buckingham/buckingham-a4. pdf* (2004).

[12] Hilfinger, P. N. An ada package for dimensional analysis. *ACM Trans. Program. Lang. Syst. 10*, 2 (Apr. 1988), 189–203.

[13] Hills, D. J. A., Grütter, A. M., and Hudson, J. J. An algorithm for discovering Lagrangians automatically from data. *PeerJ Computer Science 1* (nov 2015), e31.

[14] Hills, M., Chen, F., and Roşu, G. A rewriting logic approach to static checking of units of measurement in c. *Electron. Notes Theor. Comput. Sci. 290* (Dec. 2012), 51–67.

[15] House, R. T. A proposal for an extended form of type checking of expressions. *Comput. J. 26*, 4 (Nov. 1983), 366–374.

[16] Jonsson, D. Dimensional Analysis: A Centenary Update. 1–16.

[17] Kennedy, A. Dimension types. In *Proceedings of the 5th European Symposium on Programming: Programming Languages and Systems* (London, UK, UK, 1994), ESOP '94, Springer-Verlag, pp. 348–362.

[18] Kennedy, A. J. Relational parametricity and units of measure. In *Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (New York, NY, USA, 1997), POPL '97, ACM, pp. 442–455.

[19] Langley, P. W. BACON: A Production System That Discovers Empirical Laws. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 1* (San Francisco, CA, USA, 1977), IJCAI'77, p. 344.

[20] Lim, J., and Stanley-Marbell, P. Newton: A language for describing physics. *CoRR abs/1811.04626* (2018).

[21] Lord Rayleigh. The Principle of Similitude. *Nature 95* (dec 1915), 66–68.

[22] Press, W. H., Teukolsky, S. a., Vetterling, W. T., and Flannery, B. P. *Numerical Recipes in Fortran 77: the Art of Scientific Computing. Second Edition*, vol. 1. 1996.

[23] Rittri, M. Dimension inference under polymorphic recursion. In *Proceedings of the Seventh International Conference on Functional Programming Languages and Computer Architecture* (New York, NY, USA, 1995), FPCA '95, ACM, pp. 147–159.

[24] Rudy, S. H., Brunton, S. L., Proctor, J. L., and Kutz, J. N. Data-driven discovery of partial differential equations. *Science Advances 3*, 4 (2017), e1602614.

[25] Schmidt, M., and Lipson, H. Distilling Free-Form Natural Laws from Experimental Data. *Science 324*, 5923 (2009), 81–85.

[26] Simon, V., Weigand, B., and Gomaa, H. *Dimensional analysis for engineers.* Springer, 2017.

[27] Sonin, A. A. A generalization of the π-theorem and dimensional analysis. *Proceedings of the National Academy of Sciences 101*, 23 (2004), 8525–8526.

[28] Strang, G. *Introduction to Linear Algebra*, fifth ed. Wellesley-Cambridge Press, Wellesley, MA, 2016.

[29] Taylor, G. The Formation of a Blast Wave by a Very Intense Explosion. II. The Atomic Explosion of 1945. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 201*, 1065 (1950), 175–186.

[30] Umrigar, Z. D. Fully static dimensional analysis with c++. *SIGPLAN Not. 29*, 9 (Sept. 1994), 135–139.

[31] Wu, Y. Discovering Natural Laws by Reduction. *Journal of Computer Science and Technology 4*, 1 (1989), 35–51.