DOI:10.1145/3465216

Deriving Equations from Sensor Data Using Dimensional Function Synthesis

By Vasileios Tsoutsouras, Sam Willis, and Phillip Stanley-Marbell

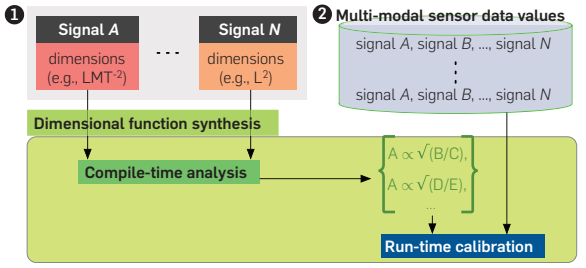
Abstract

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 (e.g., length, mass, etc.) 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 multimodal sensor data for two physical systems using 90 laboratory experiments and 10,000 synthetic idealized measurements. The results show that our technique can generate models in less than 300 ms on average across all the physical systems we evaluated. This is a marked improvement when compared to an average of 16 s for training neural networks of comparable accuracy on the same computing platform. 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 (up to 1096× improvement) and required arithmetic operations in inference (up to 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, 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 of sensor signals to synthesize compact predictive models from sensor data. 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 mmHg for pressure. The state of the art in deriving models from such data streams is to apply some form of machine learning.[11, 19] Blindly applying machine learning to data from physical systems however ignores important prior knowledge about the physical implications of the signals.

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.



1.1. Contemporary methods ignore physics

Despite its use in programming languages for tasks such as extending type systems with units of measure,[1, 2, 3, 5, 8, 12, 10, 14, 17, 23] physical information in the form of dimensions (e.g., time, temperature, etc.) 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 by 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 from physical systems with 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.

The original version of this paper was published in *ACM Transactions on Embedded Computing Systems*, October 2019.

In a first offline analysis phase, dimensional function synthesis forms monomials of physical parameters whose dimensions, when combined in a monomial, cancel out. Then, in a second run-time stage and using data from sensors of the physical parameters in question, the method calibrates dimensionally plausible equations formed from those monomials to obtain a 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 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. The method may 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.

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.[20] 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, 9, 16] The remainder of Section 2 provides a brief overview of this mathematical formalization of dimensional analysis.

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

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

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*. For the example equation $F = m \cdot a$, relating a force F applied to a mass m and its resulting acceleration, a, we have $\mathbb{S}_{\text{symbols}} = \{F, m, a\}$, $Q_1 = F$, $Q_2 = m$, and $Q_3 = a$. The dimensions of the members of $\mathbb{S}_{\text{symbols}}$ are $\mathcal{D}(Q_1) = MLT^{-2}$, $\mathcal{D}(Q_2) = M$, and $\mathcal{D}(Q_3) = LT^{-2}$. Table 1 shows additional examples of parameters and their dimensions for data from sensors in 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.

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	$\mathcal{D}(g) = LT^{-2}$

The key idea in the mathematical formulation of dimensional analysis is that for a set $\mathbb{S}_{\text{symbols}}$ such as in the example above, we can often 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 monomial 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 other parameters in the dimensionless product.

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 $m \leq n$. 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,*

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

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 *$\prod$ groups*.[6, 7]

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

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

we can equate the dimensionless product to a constant to obtain

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

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

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

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 $\mathbb{S}_{\text{symbols}}$.

2.2. Groups of dimensionless products and the Buckingham Π theorem

The primary insight exploited in many contemporary applications of dimensional analysis[18, 21] is that for any physical system represented by a set of physical parameters $\mathbb{S}_{\text{symbols}}$, it is often possible to reparametrize 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]:

Theorem 1. *Let n be the number of parameters in a description of a physical system, that is,* $n = |\mathbb{S}_{\text{symbols}}|$. *Let r be the number of dimensions from some orthogonal dimensional bases 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 parameters.*

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

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

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

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

when $n - r$ equals 1, that is, when there is only one Π product in the Π groups, then

$$\Phi(\Pi_1) = 0. \tag{5}$$

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

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

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).

3. DIMENSIONAL FUNCTION SYNTHESIS

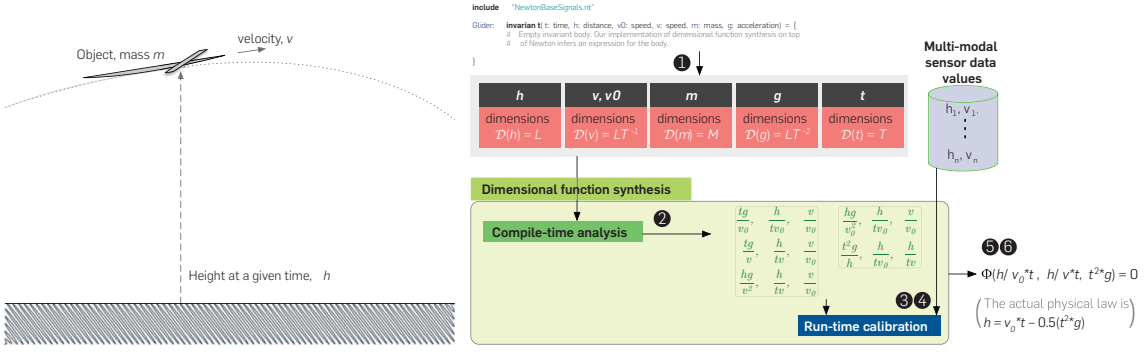
From the set $\mathbb{S}_{\text{symbols}}$ of parameters defining a physical system, we can construct a matrix representation of the system, where the columns are the parameters that are members of $\mathbb{S}_{\text{symbols}}$, the rows are base dimensions such as length, mass, or time, returned by 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 automatically computes *all the valid* $\prod$ *products across all possible* $\prod$ *groups*. Then, a run-time step 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. Figure 2 shows the steps using the terminology introduced in this section and a physical system comprising an unpowered flying object (glider) as an example.

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,

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.



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 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, 9, 13]

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* **A**, *as*

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

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 product. The solution of $\mathbf{Ak} = \mathbf{0}$ is the null space $N(\mathbf{A})$.

Physical restrictions on solutions of $N(\mathbf{A})$: because of our objective of finding physically plausible dimensionless groups that are efficiently computable, we restrict the solutions to the null space computation to rational powers of a_{ji} 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_{ji} values that are ratios of integers. To compute the rational null space of **A**, we first use Gauss-Jordan elimination to reduce the matrices to their reduced row-echelon form (RREF), where all pivots equal one, with zeros below each pivot.[22] 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 nontrivial null space is available and as a result it is not possible to form a dimensionless product with rational exponents 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(**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, 13] Our final set of dimensionless product groups is the union of all the unique dimensionless product groups resulting from computing the null spaces.

3.2. Calibration: using sensor data to transform Π groups to equational models

The dimensionless groups obtained by analyzing a description of the physical system in the form of the set $\mathbb{S}_{\text{symbols}}$ give proportionality relations between the parameters in $\mathbb{S}_{\text{symbols}}$. In the general case where more than one of the dimensionless products are not constant, then, from Equation (4), there is a function Φ' that relates the values of one of the Π products to the rest of them. We can use a data-driven approach to find the form of Φ' and we call this step *calibration*. In this case, we apply the generated Π products to transform the data at calibration-time and achieve dimensionality reduction. This allows simpler models to perform better, allowing smaller models to be learned with less data for a given prediction performance.

When a dimensionless product group contains a single item, Equation (6) showed that we can equate the dimensionless product to a constant and obtain a proportionality relation between the symbols in the dimensionless product. We still need to determine the value of the constant of proportionality and we can do so given one or more values of the parameters in the dimensionless group. 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.

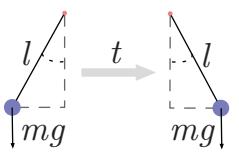
Like any model construction method, dimensional function synthesis will produce incomplete results if the inputs to the method do not fully describe the problem being modeled: an incomplete $\mathbb{S}_{\text{symbols}}$ can result in an empty set of dimensionless products.

3.3. Implementation using Newton language

We implemented dimensional function synthesis by extracting the set $\mathbb{S}_{\text{symbols}}$ from the intermediate representation of descriptions of physical systems written in Newton,[15] 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.

Pendulum example: Figure 3a shows a pendulum instrumented with a sensor that measures movement. By measuring, for example, angular movement with a gyroscope or 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 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

Figure 3. (a) 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*. (b) Physical description for the ideal pendulum written in Newton.



(a) Simple pendulum.

```
include      "NewtonBaseSignals.nt"

Pendulum : invariant ( l : distance,
                       g : acceleration,
                       m: mass,
                       t : time) =
{
    # Empty invariant body to force
    # Newton compiler to use only
    # units/dimensions of parameters
}
```

(b) Newton description.

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. Figure 3b shows a physical description for the ideal pendulum written in Newton. Dimensional function synthesis, implemented as a new backend for the Newton compiler, takes this description as input and performs the following steps.

Step 1: Dimensional matrix construction. For the system in Figure 3a, 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 **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 $n = |\mathbb{S}_{\text{symbols}}| = 4$. 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 Π product:

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

From Equation (6) (Section 2.2), it follows that we can equate the corresponding monomial to 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. MODEL EVALUATION

To demonstrate the potential of dimensional function synthesis, we compare it against black-box data-driven approaches for the characterization of a physical system. The fundamental idea is that a scientist has assembled a physical system and is able to measure a subset of its parameters either by inspection (e.g., measuring the length of a component) or by using sensors (e.g., accelerometers, tachometers, etc.). Given that a complex physical system requires effort and expertise to be analytically defined, its data-driven characterization is a promising idea. The designer can collect a large dataset of observations from the physical system and then use regression and machine learning to derive a model that fits the measured parameters to an expected output.

However, deriving an effective data driven model requires good sampling of the physical system's parameters and extensive exploration of the design space of available data fitting models. In practice, both these requirements are hard or impossible to meet, especially in the case of complex, multiparametric systems. On the contrary, the outcome of dimensional function synthesis can either *fully characterize the system or act as a starting point for targeted data-driven analysis*. In addition, simple dimensional functions have significantly less computational requirements compared to the majority of data-driven characterization techniques.

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.

Physical system	Input to our technique		Dimensions	Example of one dimensionless group generated by our automated method
Vibrating string	$\mathbb{S}_{\text{symbols}} =$	String tension, t	$\mathcal{D}(t) = MLT^{-2}$	$\left\{\frac{(L^2)(\mu)(f^2)}{(t)}, \frac{(t)}{(\mu)(f^2)(\rho^2)(\theta^2)}\right\}$
	$\{t, L, \mu, f, \rho, \theta\}$	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 flying object	$\mathbb{S}_{\text{symbols}} =$	Object elevation, h	$\mathcal{D}(\text{h}) = L$	$\left\{\frac{(t^2)(g)}{(h)}, \frac{(h)}{(t)(v_0)}, \frac{(h)}{(t)(v)}\right\}$
	$\{h, v_0, v, m, g, t\}$	Object initial velocity, v_0	$\mathcal{D}(v_0) = LT^{-1}$	
		Object velocity, v	$\mathcal{D}(v) = LT^{-1}$	
		Object mass, m	$\mathcal{D}(\text{m}) = M$	
		Acceleration due to gravity, g	$\mathcal{D}(g) = LT^{-2}$	
		Time, t	$\mathcal{D}(t) = T$	
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	$\mathcal{D}(\text{g}) = LT^{-2}$	
		Mass, m	$\mathcal{D}(m) = M$	
		Oscillation period, t	$\mathcal{D}(t) = T$	

4.1. Evaluation for synthetic data

We first compare dimensional function synthesis to regression and neural networks using synthetic idealized data. We examine several neural network topologies from the FitNet family of curve-fitting neural network architectures, which are optimized for equation fitting. We target an unpowered flying vehicle (glider) with initial velocity v_0, mass m, acceleration due to gravity g, and, trajectory height h at time t, similar to the example of Figure 2. We examine the ability of our method to find the relation between trajectory height and the rest of the physical parameters of the glider. The parameters used to describe the glider result in multiple Π groups, each of which includes multiple Π products. In this case, the form of the function Φ' for combining the Π products into an equational model is unknown and we must use a data-driven approach to find its form. Dimensional function synthesis provides two options for the calibration phase: (1) performing calibration on the target embedded system; and (2) performing 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 $<v_0, m, g, t>$ 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 0.1 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, that is, 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** = $\{\Pi_1 = t^2 \cdot g/h, \Pi_2 = h/t \cdot v_0\}$. In Π group 0, h appears only in Π_2, thus according to Equation (4), we can express h as a function Φ' of Π_1:

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

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 single-variable function Φ' of Equation (11). This simpler form is particularly valuable when our goal is to perform the final calibration on a resource-constrained embedded system. Figure 4 shows the comparative performance of using 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%, although having similar computational requirements. Neural networks are capable of minimizing the prediction error, at the expense of over 80× greater required computation. We quantify the computational requirements of each network as the total number of floating-point operations (additions, multiplications) that it requires per inference instance. Overall, the neural network models require between 0.3 and 50 s training per model for 5-fold cross-validation, with an average of 16 s. The total training latency was approximately 240 minutes on an Intel Core i7-7820X CPU at 3.60 GHz, with 32 GB RAM. This is 1096× slower than our approach which requires 1.5 ms on average for the examined physical system running on the same workstation. We have examined a total of 16 physical systems of increasing complexity and our method requires less than 300 ms on average to generate the dimensional functions, with a maximum of 3428.7 ms.

Figure 5 shows model approximation performed by neural networks trained against 20 data points, with (Figure 5b) and without (Figure 5a) dimensionally reducing the number of input parameters by making use of dimensional function synthesis. The most accurate neural network for

Figure 4. 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).

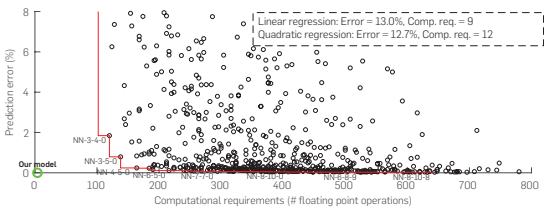
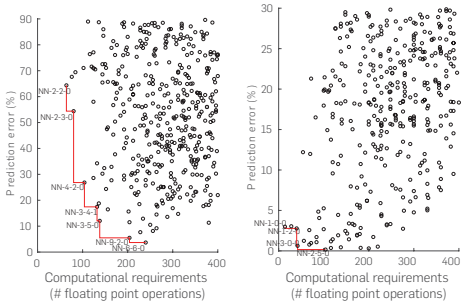
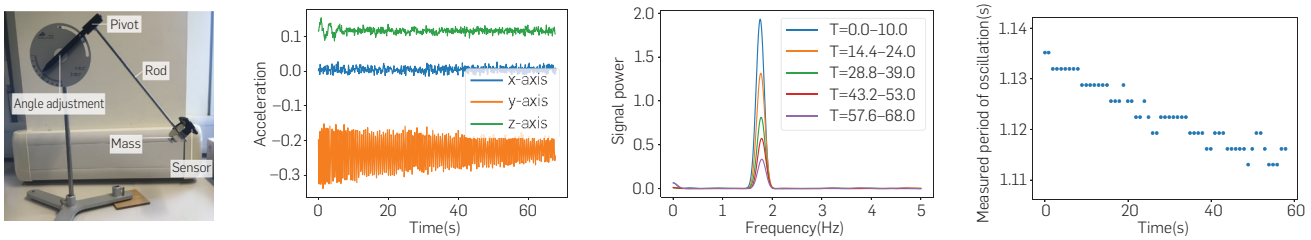


Figure 5. Prediction error versus computational requirements for predicting the trajectory of a glider. Subfigure (a) corresponds to the straightforward application of neural networks for fitting function Ψ of Equation (10). Subfigure (b) corresponds to our approach using a neural network for fitting function Φ' of Equation (11). We train all models against a set of 20 input data points. Our method achieves prediction error of 0.17% via an approximately 2.5× less computationally demanding model.



(a) Black box approach. (b) Our approach.

Figure 6. (a) Our experimental setup for the variable-*g* pendulum. (b) Data collected from the 3-axis accelerometer over time using 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) (b) (c) (d)

fitting the function Φ' over the four-dimensional space $\langle v_0, m, g, t\rangle$ has prediction error of 0.17%. It consists of two layers with 2 and 5 neurons, whereas 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 where there are insufficient data to train more complex models.

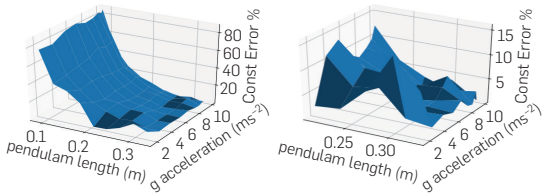
The simpler models and higher prediction accuracy of dimensional function synthesis are the result of its ability to use the physical information available. This enables better training of simpler models with less data. Most importantly, these reductions are not based on ad-hoc assumptions or approximations, but are dictated by physical laws. Models from dimensional function synthesis are more efficient for resource-constrained embedded systems as they require fewer computations during inference and less data for their training.

4.2. Evaluation on a physical pendulum

We evaluate our method in the presence of nonsynthetic data where the underlying relationship is more complex than a simple closed-form equation. We perform a series of experiments in our laboratory using an apparatus known as a *variable-g pendulum* (Figure 6a). 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 3–33 cm in steps of 3 cm and for a range of effective gravitational acceleration g resulting from pendulum pivot angles of 0°–80°, in 10° increments.

Figure 6b shows an example of the sensor data over 1 minute of pendulum oscillation. We recorded a time series of pendulum swing data such as that in Figure 6b for each of the 90 experiments we performed. We then used these time series data to calculate the oscillation period via its discrete Fourier transform (DFT). Figure 6c shows the resulting DFT output for one experiment, for four different processing windows of recorded data. Figure 6d shows the oscillation period over the duration of one 1-minute experiment, estimated using the DFT.

Figure 7. Percentage error of the predicted period of the variable-*g* pendulum, *t* for a given length *l* and gravitational acceleration, *g*. Subfigure (a) includes all experimental instances in a subset of which the ideal pendulum model assumptions are violated leading to high deviations. Subfigure (b) zooms in the region, where the error of synthesized dimensional functions is minimized.



(a) All experimental data. (b) Zoom for l > 20 cm.

Figure 7 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 estimated 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 four floating-point operations.

For pendulum lengths less than 20 cm, the error in the model increases due to nonidealities, such as friction, that are not captured by the form of the proportionality relation generated by our technique. The accuracy of the synthesized dimensional function is limited by the number of utilized parameters that describe the physical system. A richer choice in the set of parameters (e.g., such as 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, whereas the rest was used as testing samples. We used a 5-fold cross-validation policy to train the models. Figure 8 summarizes the prediction error of the period of

Figure 8. Percentage error of the predicted period of the variable-*g* pendulum, *t* for a given length *l*, and gravitational acceleration, *g*. We predict using neural networks and regression models.

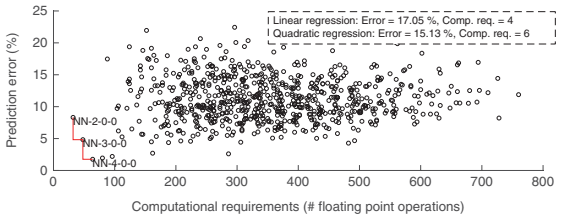
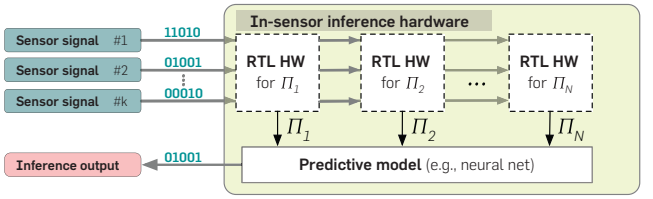


Figure 9. The hardware generated by dimensional circuit synthesis preprocesses *k* sensor signals to calculate *N* < *k* *dimensionless products* $\Pi_1 \ldots \Pi_N$. A predictive model takes the calculated product values as input and generates an inference output.



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 7b. 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 we train the black-box models against data points derived from the entire range of the pendulum experiments, they can effectively capture the nonideal characteristics of the oscillation, thus achieving high accuracy.

5. 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. Such as many existing approaches for constructing models based on human-chosen parameters, it 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 functions.

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 that correlate to the missing parameters, for example, 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, whereas traditional modeling techniques have no option but to attempt to fit data with ever more complex nonlinear models. Dimensional function synthesis enables the combination of both approaches in the case of multiple Π groups as examined in Section 4.1.

5.1. Dimensional circuit synthesis

Dimensional circuit synthesis is an extension of dimensional function synthesis that provides a compile-time method to generate digital logic circuits for the calculation of Π groups. We have implemented a Verilog register transfer level (RTL) synthesis backend in Newton, which uses the information of the calculated Π groups of dimensional function synthesis and generates the RTL description of hardware modules, each of which computes a Π monomial (Equation (2)) of a selected Π group. The hardware modules take sensor signals as input and perform the pre-inference processing of the calibrated predictive module that we derive from dimensional function synthesis. An on-device (in-sensor) inference engine will integrate the synthesized dimensional circuits with the module that executes the calibrated predictive model using, for example, a neural network. This inference module can either be a custom RTL component or a programmable core. Figure 9 shows an in-sensor inference hardware system generated using dimensional function synthesis and dimensional circuit synthesis.

We evaluated the hardware generated by the dimensional circuit synthesis backend using a Lattice Semiconductor iCE40 FPGA. The iCE40 is a low-power miniature FPGA in a wafer-scale WLCSP package of 2.15×2.50 mm, which targets sensor interfacing tasks and on-device machine learning. We used a fully open-source FPGA design flow, comprising the YoSys synthesis tool (`version 0.8+456`) for synthesis and NextPNR (`version git SHA1 5344bc3`) for placing, routing, and timing analysis.

We performed our measurements on an iCE40 Mobile Development Kit (MDK) which includes a 1Ω current sense resistor in series with each of the supply rails of the FPGA (core, PLL, I/O banks). We measure the current drawn by the FPGA core by measuring the voltage drop across the FPGA core supply rail (1.2 V) resistor using a Keithley DM7510, a laboratory-grade 7½ digital multimeter that can measure voltages down to 10 nV. Using these voltage drop measurements, we computed the power dissipated by the FPGA core for each configured RTL design. We used a pseudorandom number generator to feed the Π monomials computation circuit modules under evaluation with random input data.

We evaluated dimensional circuit synthesis on seven different physical systems described in Newton. Table 3 presents the total FPGA resource utilization for all the generated Π product computation modules, expressed in terms of the number of four-input lookup tables (LUT4 cells) required for their synthesis. These resource utilization values also include the required resources for the synthesis of the fixed-point arithmetic modules,

Table 3. Experimental evaluation on iCE40 FPGA of dimensional circuit modules generated from descriptions of physical systems.

Name	LUT4 cells	Maximum frequency	Execution latency	Avg. power at 12 MHz	Avg. power at 6 MHz
Beam	2958	16.88 Mhz	115 cycles	3.5 mW	1.8 mW
Pendulum, static	1402	17.07 Mhz	115 cycles	2.0 mW	1.1 mW
Fluid in pipe	4258	15.65 Mhz	188 cycles	5.8 mW	3.0 mW
Unpowered flight	1930	16.44 Mhz	81 cycles	2.3 mW	1.2 mW
Vibrating string	2183	16.67 Mhz	183 cycles	2.5 mW	1.3 mW
Warm vibrating string	3137	16.77 Mhz	269 cycles	1.9 mW	1.0 mW
Spring-mass system	1419	16.67 Mhz	115 cycles	3.4 mW	1.8 mW

which we integrated in the computation module of each Π product.

The execution latency column lists the required cycles for completing the calculations of the critical path of each of the generated RTL modules. We obtained the number of cycles by simulating the execution of the RTL modules for pseudorandom inputs generated by linear feedback shift registers (LFSRs). In each RTL module, we parallelize the calculation of different Π products but the required operations per Π product are executed serially.

The last column of Table 3 shows the measured power dissipation of each design configured in the iCE40 FPGA. In all cases, the power dissipation is less than 6 mW and as low as 1 mW, demonstrating the suitability of our method for small-form-factor, battery-operated on-device inference at the edge.

6. CONCLUSION

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 evaluate the execution cost and accuracy of the models our method generates compared against regression models and neural networks. 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 (up to 1096× improvement) and required arithmetic operations in inference (up to 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 grant EP/V004654/1. Youchao Wang, Vlad-Mihai Mandric, and James Rhodes contributed to the implementation of the linear-algebraic methods within the Newton compiler.

References

1. Allen, E., Chase, D., Luchangco, V., Maessen, J.-W., Steele, G.L., Jr. Object-oriented units of measurement. In *Proceedings of the 19th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications,* OOPSLA'04 (2004), ACM, New York, NY, USA, 384–403.
2. Antoniu, T., Steckler, P.A., Krishnamurthi, S., Neuwirth, E., Felleisen, M. Validating the unit correctness of spreadsheet programs. In *Proceedings of the 26th International Conference on Software Engineering,* ICSE'04 (2004), IEEE Computer Society, Washington, DC, USA, 439–448.
3. Babout, M., Sidhoum, H., Frecon, L. Ampere: A programming language for physics. *European J. Phys. 11*, 3 (1990):163.
4. Barber, D. *Bayesian Reasoning and Machine Learning.* Cambridge University Press, Cambridge, 2012.
5. Biggs, G., 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. *Phys. Rev. 4*, 4 (1914), 345–376.
7. Carlson, D.E. A mathematical theory of physical units, dimensions, and measures. *Arch. Rational Mechanics Anal. 70*, 4 (1979), 289–305.
8. Cmelik, R.F., Gehani, N.H. Dimensional analysis with C++. *IEEE Softw. 5*, 3 (1988), 21–27.
9. Carlson, D.E. On some new results in dimensional analysis. *Arch. Ration. Mech. Anal. 68*, 3 (1978), Springer, 191–210.
10. Hilfinger, P.N. An ada package for dimensional analysis. *ACM Trans. Program. Lang. Syst. 10*, 2 (Apr. 1988), 189–203.
11. Hills, D.J.A., Grütter, A.M., Hudson, J.J. An algorithm for discovering Lagrangians automatically from data. *PeerJ Comput. Sci. 1*, (Nov. 2015), e31.
12. Hills, M., Chen, F., Roşu, F. A rewriting logic approach to static checking of units of measurement in C. *Electron. Notes Theor. Comput. Sci. 290*, (Dec. 2012), 51–67.
13. Jonsson, D. Dimensional analysis: A centenary update. arXiv preprint arXiv:1411.2798 (2014).
14. Kennedy, A. Dimension types. In *Proceedings of the 5th European Symposium on Programming: Programming Languages and Systems,* ESOP'94 (1994), Springer-Verlag, London, UK, 348–362.
15. Lim, J., Stanley-Marbell, P. Newton: A language for describing physics. *CoRR,* abs/1811.04626 (2018).
16. Rayleigh, L. The principle of similitude. *Nature 95* (Dec. 1915), 66–68.
17. Rittri, M. Dimension inference under polymorphic recursion. In *Proceedings of the Seventh International Conference on Functional Programming Languages and Computer Architecture,* FPCA'95 (1995), ACM, New York, NY, USA, 147–159.
18. Rudy, S.H., Brunton, S.L., Proctor, J.L., Kutz, J.N. Data-driven discovery of partial differential equations. *Sci. Adv. 3*, 4 (2017), e1602614.
19. Schmidt, M., Lipson, H. Distilling free-form natural laws from experimental data. *Science 324*, 5923 (2009), 81–85.
20. Simon, V., Weigand, B., Gomaa, H. *Dimensional Analysis for Engineers.* Springer, Gewerbestrasse, Cham, Switzerland, 2017.
21. Sonin, A.A. A generalization of the *Π*-theorem and dimensional analysis. *Proc. Natl. Acad. Sci. 101*, 23 (2004), 8525–8526.
22. Strang, G. *Introduction to Linear Algebra*, 5th edn. Wellesley-Cambridge Press, Wellesley, MA, 2016.
23. Umrigar, Z.D. Fully static dimensional analysis with C++. *SIGPLAN Not. 29*, 9 (Sept. 1994), 135–139.

Vasileios Tsoutsouras, Sam Willis, and **Phillip Stanley-Marbell** ({vt298, sjw238, phillip.stanley-marbell}@eng.cam.ac.uk), University of Cambridge, Cambridge, U.K.