Measurement: Sensors xxx (xxxx) xxx

Contents lists available at ScienceDirect

Measurement: Sensors

journal homepage: www.sciencedirect.com/journal/measurement-sensors

# Preserving model structure and constraints in scientific computing

ARTICLE INFO

*Keywords:*
Dimensioned variable
Functional programming
Numerical computation
Units of measurement

ABSTRACT

In this paper, we look at how model structure and constraints can be incorporated into scientific computing using functional programming and, implicitly, category theory, in a way that constraints are automatically satisfied. Category theory is the study of different types of objects (e.g., sets, groups, vector spaces) and mappings between them (e.g., functions, homomorphisms, matrices) and is used in mathematics to model the underlying structure associated with systems we wish to describe and how this underlying structure is preserved under transformations. In this paper, we look at the structure associated with the representation of, and calculations using, quantitative data. In particular, we describe how measurement data can be represented in terms of the product $C \times D$ of two groups: the first, $C$, the counting algebra, and the second, $D$, the dimension algebra. Different but equivalent unit systems are related through group isomorphisms. The structure associated with this representation can be embedded in software using functional programming.

## 1. Introduction

Current practice in scientific computing in metrology usually involves a) writing a report or paper providing a model for the system under study and how measurement data can be used to extract information about quantities of interest, b) designing an algorithm that uses the model and measurement data to compute estimates of the relevant model parameters (and their associated uncertainties), c) implementing the algorithm in software, and d) testing software to increase the level of trust in the software. Good practice dictates that the software should be well commented and documented so that a user can follow the logic and intent of the written instructions. However, while the intent of a piece of software may be made clear in the comments section and supporting documentation, for nearly all computing languages used regularly in scientific computing, the compiler only sees the coded instructions and has no way of checking whether the code is actually performing the required tasks.

The way most of us write software has not changed much in 50 years and we use essentially the same text editors to write software as we use to write reports and papers. However, computer science has evolved enormously over the last 50 years, not just in terms of faster computation and larger memory, but also in terms of the semantic reach of computer languages that embed logic and inference. In particular, functional programming languages are designed to be able to represent model structure and constraints that we normally would write in mathematics.

Category theory [1,2] is the study of different types of objects (e.g., sets, groups, vector spaces) and mappings between them (e.g., functions, homomorphisms, matrices) and is used in mathematics to model the underlying structure associated with systems we wish to describe and how this underlying structure is preserved under transformations.

Model constraints can be used in two ways [3]. Firstly, software that claims to perform the required calculations can be checked by the compiler to see if the constraints are satisfied. The second point of view is that each model constraint reduces the set of programs/mappings, from all possible programs that could be written, that are consistent with the constraints. With enough model constraints, we can end up with only one program/mapping that satisfies the constraints and the compiler implicitly or explicitly constructs this program based on the constraints. In fact the role of a functional requirements specification in standard programming methodologies is to specify the required behaviour of the program. Such functional requirements, if they were written in appropriate language, act as the model constraints that help check or even construct the required software. By incorporating model constraints such as those defined by dimensioned variables, it is hoped that much more programming errors will be detected by the compiler, significantly increasing the trustworthiness of numerical computation.

In this paper, we are interested in model constraints and how they relate to functional programming, section 2, the type of constraints that arise from assigning dimensions (length, mass, etc.) to variables, section 3, and how these constraints can be incorporated into scientific computing, section 4. A discussion and our concluding remarks are given in section 5.

## 2. Functional programming: a motivation

There are two main computer programming paradigms, imperative programming and functional programming. Most standard languages used for scientific computation are imperative and programs are a list of instructions (imperatives, commands) that the computer has to perform

This article is part of a special issue entitled: Supp: IMEKO 2024 published in Measurement: Sensors.

https://doi.org/10.1016/j.measen.2024.101796

in order to complete the calculation. Functional programs regard functions as the primary element and programs are compositions of functions in which the behaviour of each function is constrained to achieve a specific result. As long as each function is specified correctly, the composite program is constrained to produce the correct result: 'correctness is built in'. A simple example below attempts to give a flavour of a functional programming approach.

### 2.1. Example: A collection of mass standards

Suppose a mass laboratory has a set of mass standards $S = \{s_1, s_2, \ldots, s_n\}$ that it uses to calibrate other mass artefacts using a mass balance. Each of the standards is a right circular cylinder in shape. In order to compensate for air buoyancy effects [4], it is necessary to know the volume and density of each artefact.

We suppose that the mass, height and radius of each artefact is known, defining three functions $m,h,r$: $S \to \mathbb{R}$. We denote of the set of mappings from $S$ to $\mathbb{R}$ by $\mathbb{R}^S$ so that $m$, $h$, and $r$ are three elements of $\mathbb{R}^S$. Since $S$ has $n$ elements, we note that $\mathbb{R}^S$ can be represented as $\mathbb{R}^n$ derived from the indexing of the elements of $S$. We say that $\mathbb{R}^S$ isomorphic to $\mathbb{R}^n$ and write $\mathbb{R}^S \cong \mathbb{R}^b$ (as sets) with $f \in \mathbb{R}^S$ identified with the $n$-tuple $(f(s_1), \ldots, f(s_n)) \in \mathbb{R}^n$. The functions $m$, $h$, and $r$ equivalently, specify three $n \times 1$ vectors $m$, $h$ and $r$, in $\mathrm{R}^n$. we can also think of $(m,h,r)$ as an element of

$$\mathbb{R}^S \times \mathbb{R}^S \times \mathbb{R}^S \cong \left(\mathbb{R}^3\right)^S \cong \left(\mathbb{R}^3\right)^n \cong \mathbb{R}^{3n} \cong \left(\mathbb{R}^n\right)^3$$

This chain of isomorphisms maps the three functions $(m,h,r)$ to the $3n$-vector $\left(\boldsymbol{m}^{\mathrm{T}}, \boldsymbol{h}^{\mathrm{T}}, \boldsymbol{r}^{\mathrm{T}}\right)^{\mathrm{T}}$ and the 3-tuple $(\boldsymbol{m}, \boldsymbol{h}, \boldsymbol{r})$.

More generally, for any three sets $A$, $B$ and $C$, with $C^A$ denoting the set of functions from $A \to C$, $C^{A\times B} \cong \left(C^B\right)^A \cong \left(C^A\right)^B$. The exponential notation suggests the analogous arithmetical identities for numbers $a$, $b$ and $c$: $c^{ab} = (c^a)^b = \left(c^b\right)^a$. Here, and above, $A \times B$ denotes the Cartesian product of $A$ with $B$ given by the set of pairs $\{(a,b)\colon a \in A, b \in B\}$. The Cartesian product has two associated mappings, also referred to as projections, $\pi_A\colon A \times B \to A$ and $\pi_B\colon A \times B \to B$ defined by $\pi_A(a,b) = a$ and $\pi_B(a,b) = b$.

The density associated with a right circular mass artefact can be inferred from the following facts: a) density is mass over volume, b) the volume of a right cylinder is its height times its cross-sectional area, c) the cross-section of a right circular cylinder is a circle, and d) the area of circle is $\pi$ times its radius squared. These facts can be represented by functions $d(m,v)$ calculating density, $v(h,a)$ calculating the volume of a cylinder, and $a(r)$ calculating the area of a circle. The density calculation $d\colon S \to \mathbb{R}$ can therefore be written as

$$d(s) = d(m(s), v(h(s), a(r(s)))) = \frac{m(s)}{\pi r^2(s) h(s)} \tag{1}$$

Implicit in this calculation are the deductions about the volume of a right circular cylinder and its density. The density function is a function of the volume function which is in turn a function of the area function. It can also been seen that the density function can be applied to $m$, $h$ and $r$ where now $m,h,r \in \mathbb{R}^S$ are themselves functions, defining $d \in \mathbb{R}^S$ and, along the way, $v \in \mathbb{R}^S$. We can generalise to other right cylinders having square or elliptical cross sections, for example. The density function $d$ can be written as $d = d(m,v(h,a)))$ where the area function $a$ has yet to be specified. With the $m$ and $v$ functions specified, such a function maps functions calculating area to functions calculating densities and is specified by facts a) and b) above. Adding facts c) and d) reduces the set of density functions satisfying the constraints to a set with one element given by (1).

These simple examples involve sets (objects), Cartesian products, functions between sets, sets of functions, functions of functions, applying model constraints to determine subsets of functions that meet the model constraints and so on. The Curry-Howard-Lambek (CHL) [5–7] correspondence relates concepts in logic, computing and category theory and says, loosely, that proofs are programs and programs are mappings. Much of science involves finding symmetries, invariances, conservation laws, etc., associated with the systems under study. The appeal of using category theory as a tool for scientific discourse derives in part from the CHL correspondence since it allows the model constraints and contextual information such as controlled vocabularies and ontologies to be encoded [8], enables logical inferences to be made, and supports constructive computation. Functional programming can use categorical concepts to ensure software respects the model constraints in an appropriate context.

## 3. Type constraints and the representation of scientific data in terms of dimensions and units

This section is concerned with the representation of physical quantities in terms of numerical values and associated units [9–11], particularly from the point of view of enabling machine-actionable interoperability, an issue of importance in the move towards a digital SI [12]. We are concerned with a measurable attribute or quantity $Q$ associated with an object: its mass, its length, etc. A methodology for defining a measurement representation scheme has the following elements.

### 3.1. A set of base dimensions

The first component of the representation system involves a set of base attributes, $\mathrm{A}_1, \ldots, \mathrm{A}_p$. The base attributes should be chosen so that any quantity $Q$ of interest is associated with a unique dimension vector $D(Q)$ expressed as

$$D_{A(Q)} = A^d = A_1^{d_1} A_2^{d_2} \ldots A_p^{d_p},$$

where $d$ is an $n$-vector of integers. The set of dimensions associated with $A$ can be identified with the free Abelian group on the $n$ generators $A_1, \ldots, A_p$, which, in turn is isomorphic to $\mathbb{Z}^p$ through the isomorphism

$$A_1^{d_1} A_2^{d_2} \ldots A_p^{d_p} \mapsto d \in \mathbb{Z}^p.$$

Below, we use the symbol $D$ to represent $\mathbb{Z}^p$ The unit of the group A associated with so-called 'dimensionless' quantities is mapped to $0_D$, the $p-$vector of zeros in $\mathbb{Z}^p$.

As a primary example, according to the SI brochure [13], the dimension vector can be written as

$$D_{\mathrm{SI}}(Q) = \dim(Q) = \mathrm{T}^{\alpha}\mathrm{L}^{\beta}\mathrm{M}^{\gamma}\mathrm{I}^{\delta}\Theta^{\epsilon}\mathrm{N}^{\zeta}\mathrm{J}^{\eta}, \tag{2}$$

where T, L, M, I, Θ, N and J represent time, length, mass, electric current, thermodynamic temperature, amount of substance, and luminous intensity, respectively.

### 3.2. A counting algebra used to specify numerical values

The second component of a representation system is required to represent the numerical value associated with a (measurement of) a quantity. Most usually, we use the positive real numbers $\mathbb{R}^{>} = \{r \in \mathbb{R} : r > 0\}$ to specify numerical numbers but it is possible to use other algebraic structures for a counting algebra. A requirement for the counting algebra is that it supports the operation of multiplication. We will denote the counting algebra by $C$ and the multiplicative identity by $1_C$. In practice, it is necessary to extend the counting algebra to support addition and subtraction. Usually, the real numbers $\mathbb{R}$ performs this role.

### 3.3. Representation of a measured value as a mapping to $\boldsymbol{R} = \boldsymbol{C} \times \boldsymbol{D}$

Given base dimensions $\mathrm{A}_k$ and counting algebra $\mathbf{C}$, the representation of measured values can be thought of as a mapping from quantities to $R = C \times D$ which we will denote by

$$Q \mapsto (c_A(Q), d_A(Q)), c_A(Q) \in C, d_A(Q) \in D,$$

and refer to $c_A(Q)$ as the count associated with $Q$ and $d_A(Q)$ the dimension associated with $Q$ for the particular choice of base dimensions, indicated by the subscript $A$.

The fact that $C$ and $D$ are both Abelian groups with respect to multiplication and addition, respectively, means that the Cartesian product $R = C \times D$ is also an Abelian group with respect to the operation $*_R$ defined component-wise as

$$(c,d)*_R(e,f) = (c \times d, e+f) \tag{3}$$

We denote the identity element by $1_R$ with $1_R = (1_C, 0_D)$. The inverse $(c,d)^{-1}$ of $(c,d)$ is given by $(c^{-1}, -d)$. We also use the symbol $*_D$ to denote the group operation associated with $D$.

The group $R$, as a Cartesian product, has associated projections that are also group homomorphisms

$$\pi_C(c,d) \mapsto c \in C, \pi_D(c,d) \mapsto d \in D.$$

There are also group homomorphisms $\iota_C : C \longrightarrow R$ and $\iota_D : C \longrightarrow R$ given by

$$\iota_C(c) = (c, 0_D), \iota_D(d) = (1_C, d).$$

with the properties that $\pi_C \circ \iota_C$ and $\pi_D \circ \iota_D$ are the identity homomorphisms on $C$ and $D$, respectively. By group homomorphism, we mean a mapping that preserves the group operation: a mapping $f : G \longrightarrow H$ between two is a groups is a homomorphism if $f(x *_G y) = f(x) *_H f(y)$ where $*_G$ and $*_H$ are the group operations of $G$ and $H$, respectively.

The Abelian group structure of $R$ enables the dimensions of products of quantities to be calculated automatically. Representing physical quantities as elements of $R$ allows us to make sure that the numerical value and associated unit are always aligned.

### 3.4. Equivalent systems of measurement representation

Two systems of representing measurement results

$$Q \mapsto (c_A(Q), d_A(Q)), Q \mapsto (c_B(Q), d_B(Q)),$$

using $R = C \times D$ can be said to be equivalent if there is a group isomorphism, i.e., an invertible homomorphism $F : C \times D \longrightarrow C \times D$ from $C \times D$ to itself such that $(c_B, d_B) = F(c_A, d_A)$. Such an automorphism has the form

$$F = \begin{bmatrix} F_{CC} & F_{DC} \\ F_{CD} & F_{DD} \end{bmatrix}$$

where $F_{CC} : C \longrightarrow C$ and $F_{DD} : D \longrightarrow D$ are group isomorphisms and $F_{CD} : C \longrightarrow D$ and $F_{DC} : D \longrightarrow C$ are group homomorphisms.

### 3.5. Example: $\boldsymbol{R} = \mathbb{R}^> \times \mathbb{Z}^p$

The only isomorphism of $\mathbb{R}^>$ to itself as a group under multiplication is the identity mapping. Isomorphisms of $D = \mathbb{Z}^p$ can be represented by a $p \times p$ matrix $M$ with integer entries and determinant $\pm 1$. For example, suppose a dimension system has base dimensions momentum, P, force, F, and energy, E. These are related to the SI base dimensions for time, T, length, L, and mass, M through

$$\begin{bmatrix} d_T \\ d_L \\ d_M \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 \\ 0 & -1 & 1 \\ -2 & 0 & -1 \end{bmatrix} \begin{bmatrix} d_P \\ d_F \\ d_E \end{bmatrix},$$

so that time T has dimensions $PF^{-1}$, and length L has dimensions $F^{-1}E$, etc.

The only homomorphism from $\mathbb{R}^> \longrightarrow \mathbb{Z}^p$ is the trivial mapping that maps all elements to the unit element $0_D = (0, \ldots, 0) \in \mathbb{Z}^p$. Homomorphisms $\mathbb{Z}^p \longrightarrow \mathbb{R}^>$ take the form

$$(d_1, \ldots, d_p) \mapsto \prod_{k=1}^{p} a_k{}^{d_k}, \tag{4}$$

where $a_k$ are fixed numbers in $\mathbb{R}^>$. Thus, an automorphism of $\mathbb{R}^> \times \mathbb{Z}^p$ maps $(c,d)$ to $(e,f)$ with

$$e = c\prod_{k=1}^{p} a_k{}^{d_k}, f = Md.$$

In principle, the mapping between two equivalent unit systems based on $\mathbb{R}^> \times \mathbb{Z}^p$ is defined a $p$-vector of positive real numbers $\boldsymbol{a} = (a_1, \ldots, a_p)^{\mathrm{T}}$ and a $p \times p$ integer matrix with determinant $\pm 1$. This vector and matrix enables the interoperability of the two unit systems. For two representation systems with the same base dimensions $\mathrm{A}_k$, an isomorphism between the two is defined by the $p$-vector $\boldsymbol{a} = (a_1, \ldots, a_p)^{\mathrm{T}}$ where the numbers $a_k$ are conversion factors, for example, converting metres to inches. We note that degrees Kelvin and degrees Celsius are not related by such a conversion factor (but degrees Kelvin and degrees Rankine are with 1 K = 1.8° Rankine).

### 3.6. Base units associated with the base dimensions

The base dimensions give a way of specifying attributes associated with a system and the counting algebra enables numerical calculations associated with quantities. The role of the base units is to specify, directly or indirectly, quantities $Q_k$ whose representation $r_{Q_k}$ is such that $r_{Q_k} = (1_C, e_k)$, where $e_k$ is the $k$th generator of the free Abelian group $D$. For $D = \mathbb{Z}^p$, $e_k$ is the $p$-vector with one in the $k$th element and zeros elsewhere. Thought of in this way, measurement units are names of specific elements of $R$. In the SI, the second is the name for $(1, e_1)$, the metre is the name for $(1, e_2)$, the Newton is the name for $(1, d_{\mathrm{N}}$ where $d_{\mathrm{N}} = (-2, 1, 1, 0, 0, 0, 0)^{\mathrm{T}}$, etc.

### 3.7. Conventional representation of measurements

The SI brochure [13, Section 5.4] and the NIST publication on the use of the international system of units [14, Section 7] emphasise that the value of a quantity $Q$ can be written as $Q = \{Q\}[Q]$ where $\{Q\}$ is the numerical value and $[Q]$ is the (name of the) associated unit, and that $\{Q\}[Q]$ should be regarded as the product of mathematical entities.

The statement $L = 5$ m implies $\{L\} = 5$ and $[L] = \mathrm{m}$ and is equivalent to the statement $L/\mathrm{m} = 5$. The representation of the measurement result $Q$ as $r_Q \in R$ provides a formal, mathematical realisation of the idea of a product of a numerical value and a unit. Regarding m as the name for $(1_C, e_2)$ the 'product' 5 m is realised as the product $(5, e_2) = \iota_C(5) *_R \iota_D(e_2) = (5, 0_D) *_R (1_C, e_2)$ of elements or $R$. The representation $r_Q \in R$ also supports the notion that the numerical value and associated unit should be conjoined and are not to be separated without caution.

### 3.8. Quantity kinds, dimensional constraints and counting algebras

The advantage of representing quantitative data in $R$ within the context of richly typed programming languages is that it enables the interoperability of systems of dimensions and associated units to be implemented at a high level. It also allows the compiler to check that all calculations are dimensionally consistent, a capability that has been sought after for many decades, see, e.g., Ref. [15]. For example, the programming language F♯ [16] which supports functional programming allows variables to be assigned dimensions and units, based on the work of Kennedy [17] and others. Examples of the implementation of dimension checking in other computer languages are given in Refs. [3, 18].

While dimensional consistency represents a necessary constraint, it is by no means the only constraint relating to the representation of measurement results [19] as quantities that are different can have the same dimension. For example, in the SI [13], both the derived units the hertz, the unit for frequency, and the becquerel, the unit for radioactivity, have

dimension $\mathrm{T}^{-1}$ but represent different quantities. If we regard the hertz as a rate of one cycle per second and the becquerel as a rate of one nucleus decay per second, then it is not unreasonable to regard the hertz and becquerel as instances of the more general concept of a rate of one event per second. In urban environments, it might be required to estimate the average number of vehicles – motorbikes, cars and trucks, etc. – passing along a street per day. The frequency of cars is a different quantity from the frequency of trucks. In order to maintain clarity in describing the real world it is necessary to have a controlled vocabulary to describe different types of events. Similarly, in chemistry, it is necessary to have a controlled vocabulary to discuss counts relating to different chemical elements, molecules, etc.

In the example of a right circular cylinders, section 2.1, both the radius and height of a cylinder are associated with dimension L but represent different quantities and getting them mixed up in calculating the volume $v = \pi r^2 h$ of the cylinder would lead to errors. Adding a height to a radius also seems wrong but the complete surface area $s$ of a solid cylinder is given by $s = 2\pi rh + 2\pi r^2 = 2\pi r(r + h)$ so in this context (and others) adding them together is fine. In order to make appropriate inferences, it is necessary to encode or model in a formal language/controlled vocabulary/ontology what is meant by cylinder, radius (or diameter), height etc., so that using the height to calculate the cross-sectional area of the cylinder would detected as inconsistent with the model.

Even if two quantities have the same dimension and are of the same kind, it might not make sense to add them together. The main examples are intensive quantities such as density $d = m/v$ which is the ratio of two extensive quantities mass $m$ and volume $v$ and has dimension $\mathrm{L}^{-3}\mathrm{M}$. Given two densities $d_k = m_k/v_k$, $k = 1,2$, the sum $d = d_1 + d_2$ might not represent anything physically meaningful. However representing $d_k$ as $(m_k, v_k) \in R \times R$ gives

$$d = (m, v) = (m_1 + m_2, v_1 + v_2) = d_1 + d_2 \in R \times R$$

and is valid.

Is a difference in two masses (lengths, temperatures) the same kind of quantity as mass (length, temperature)? We can use $\mathbb{R}^{>}$ as a counting algebra to represent a mass measurement but for a difference in masses we need to use $\mathbb{R}$. Different counting algebras can reflect differences in quantity kinds and it is possible to consider counting algebras that are more general such as the complex numbers $\mathbb{C}$, quaternions, Lie groups, etc., and can include measurement uncertainty by constructing counting algebras that are probability distributions defined on $\mathbb{R}^{>}$, $\mathbb{R}$, etc.

## 4. Numerical calculations and $R = C \times D$

Current practice in numerical computing is usually implemented in software in languages in which variables have only a limited number of types: Boolean, integer, real (single, double precision), complex, string, etc. In this section, we look at issues in implementing numerical calculations in which variables are represented in $R = C \times D$, where $R = \mathbb{R} \times \mathbb{Z}^p$. We use $\mathbb{R}$ rather than $\mathbb{R}^{>}$ as we want to accommodate differences of quantities with the same dimension.

### 4.1. Functions $R \longrightarrow R$

A function $f : R \longrightarrow R$ defines a subset $F = \{(x,y): y = f(x)\}$ of $R \times R$. We assume that all functions $f : R \longrightarrow R$ involved have a fixed dimension defined by $(d,e) \in D \times D$ and are such that $(x, d) \mapsto (f_C(x), e)$ , with $f_C : \mathbb{R} \longrightarrow \mathbb{R}$. Usually, we will denote $f_C$ also by $f$ if there is no confusion. If $f(x)$ is differentiable and has dimension $(d,e)$, the derivative $f'(x) = df/dx$ is a function of dimension $(d,e-d)$. Similarly, the second derivative $f''(x)$ has dimension $(d,e - 2d)$. It follows that the Taylor expansion for $f$ about $x_0$ given by

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \frac{f'''(x_0)}{3!}(x - x_0)^3 + \ldots$$

is consistent with the dimension constraints. In the same vein, if $p(x)$ is a model response function of dimension $(d,e)$ modelled as a polynomial $p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$, then the coefficients are such that $a_0$ has dimension $e$, $a_1$ has dimension $e - d$, $a_2$ dimension $e - 2d$, and so on.

In probability theory, a cumulative distribution function (CDF) must have dimension of the form $(d,0_D)$ since probabilities are dimensionless. The corresponding probability density function (PDF) is the derivative of the CDF and therefore has dimension $(d,-d)$.

We also consider multivariate functions $f : R^n \longrightarrow R$ that satisfy dimensional constraints. For example, suppose $z = f(x,y) = ax^2 + bxy + cy^2$ is of fixed dimension $(d(x),d(y),d(z))$ involving coefficients $a$, $b$ and $c$, elements of $R$. The dimensions of these coefficients must satisfy $d(a) = d(z) - 2d(x)$, $d(b) = d(z) - d(x) - d(y)$ and $d(c) = d(z) - 2d(y)$ as elements of $D = \mathbb{Z}^p$. The partial derivative function $\partial f/\partial x = 2ax + by$ has dimension $(d(x),d(y),d(z) - d(x))$. Note that if $\pi_C(a) = \pi_C(b) = \pi_C(c) = 1$ as elements of $\mathbb{R}$, while we can write $f(x,y) = x^2 + xy + y^2$ as a function $\mathbb{R}^2 \longrightarrow \mathbb{R}$, we cannot ignore the coefficients when regarding $f$ as a function $R^2 \longrightarrow R$.

### 4.2. Vector spaces and matrices

See also [10,20]. We can construct the equivalent of vector spaces and mappings between vector spaces involving the Cartesian product $R^n = (R \times \mathbb{Z}^p)^p \cong \mathbb{R}^n \times (\mathbb{Z}^p)^n$ where now we regard $\mathbb{R}^n$ as a vector space with inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{y} = \sum_{i=1}^n x_i y_i$ for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$. For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ with associated dimension vectors $d(\boldsymbol{x})$ and $d(\boldsymbol{y})$ in $(\mathbb{Z}^p)^n$, we can only form the inner product $z = \langle \boldsymbol{x}, \boldsymbol{y} \rangle_R$ if $d(x_i) + d(y_i) = d(z)$ is constant, $i = 1, \ldots, n$. We can also consider matrices $A \in R^{m \times n} = \mathbb{R}^{m \times n} \times (\mathbb{Z}^p)^{m \times n}$ where $c(A)$ is our usual concept of a real-valued $m \times n$ matrix and $d(A)$ is an $m \times n$ array of elements of $\mathbb{Z}^p$. For $n$-vector $\boldsymbol{x} \in R^n$, we can construct the matrix-vector product $\boldsymbol{y} = A\boldsymbol{x}$ if we can form all the inner products $\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle_R$, where $\boldsymbol{a}_i$ is the $i$th row of $A$. The dimensions associated with $A$, $\boldsymbol{x}$ and $\boldsymbol{y}$ must satisfy $d(a_{ij}) + d(x_j) = d(y_i)$, $j = 1, \ldots, n$, where $a_{ij} = A(i,j)$. We can summarise this constraint by $d(A) = d(\boldsymbol{y}) *_D (-d(\boldsymbol{x}^\top))$ where the term on the right is the outer 'product' of $m \times 1$ vector of dimensions $d(\boldsymbol{y})$ with the $1 \times n$ vector of dimensions $-d(\boldsymbol{x}^\top)$, remembering that the group operation $*_D$ in $\mathbb{Z}^p$ is given by addition.

If $A$ is a matrix with associated dimensions given by $\boldsymbol{e} *_D (-\boldsymbol{d}^\top)$ then $A$ can act on vectors of dimension $\boldsymbol{d} + g = (d_1 + g, d_2 + g, \ldots, d_n + g)^\top \in (\mathbb{Z}^p)^n$ for any $g \in \mathbb{Z}$. Note that the dimensions of $A$ can also be expressed as $((\boldsymbol{e} + h) *_D (\boldsymbol{d} - h)$ for $h \in \mathbb{Z}$. A representation of such dimensions that removes this degree of freedom in discussed in Ref. [20].

In standard matrix algebra involving matrices with real elements, if $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{k \times \ell}$ we can from the matrix product $C = AB$ if and only if $n = k$. For matrices with elements in $R$, constraints relating to dimension are much more binding. Suppose $A \in R^{m \times n}$ and $B \in R^{n \times \ell}$ are such that the dimensions of $A$ and $B$ are given by outer products: $d(A) = \boldsymbol{d} *_D \boldsymbol{e}^\top$ and $d(B) = \boldsymbol{f} *_D \boldsymbol{g}^\top$. The matrix product $C = AB$ in $R$ can be formed if $d(e_j) + d(f_j) = h$ is constant, $j = 1, \ldots, n$, in which case, $d(C) = h *_D (\boldsymbol{d} *_D \boldsymbol{g}^\top)$, i.e., $d(c_{ij}) = h + d(d_i) + d(g_j)$, $i = 1, \ldots, n$, $j = 1, \ldots, \ell$.

The dimensional constraints on matrix-vector and matrix-matrix multiplications have a significant bearing in what we mean by the identity matrix and the inverse of a matrix. In standard numerical linear algebra the $n \times n$ identity matrix $I$ is such that for any $n$-vector $x$, $Ix = x$. For dimensioned vectors, we can only form $Ix$ if the dimensions conform. Suppose $\boldsymbol{d}$ is a $n \times 1$ dimension vector and let $\boldsymbol{x} \in R^n$ be any $n$-vector with dimension $\boldsymbol{d}$. We would like an identity matrix $I_{\boldsymbol{d}}$ to have the property that $I_{\boldsymbol{d}}\boldsymbol{x} = \boldsymbol{x}$ as elements of $R^n$. This property holds if $I_{\boldsymbol{d}} = (I, \boldsymbol{d} *_D (-\boldsymbol{d}^\top)) \in \mathbb{R}^{n \times n} \times (\mathbb{Z}^p)^{n \times n}$. Furthermore, $I_{\boldsymbol{d}} I_{\boldsymbol{d}} = I_{\boldsymbol{d}}$ and $I_{\boldsymbol{d}}^{\mathrm{T}} = I_{-\boldsymbol{d}}$.

Suppose $A \in \mathbb{R}^{n \times n}$ has inverse $B$. If $A = (A, \boldsymbol{e} *_D (-\boldsymbol{d})^\top) \in R^n$, set $B = (B, \boldsymbol{d} *_D (-\boldsymbol{e}^\top))$. Then $BA = (I, \boldsymbol{d} *_D (-\boldsymbol{d})^\top) = I_{\boldsymbol{d}}$ so that $B$ acts as the left inverse of $A$ in $R^n$. The product $AB$ can only be formed if $\boldsymbol{e} = \boldsymbol{d}$, in which case, $AB = BA = I_{\boldsymbol{d}}$.

### 4.3. Regression problems

Many data analysis problems in metrology involve finding the best-fit model to data $\boldsymbol{y} = (y_1, \ldots, y_m)^\top$. Typically, the model response may be a function $\phi(x,\boldsymbol{a})$ depending on covariates $x$ and parameters $\boldsymbol{a} = (a_1, \ldots, a_n)^\top$, and the optimal values of the parameters [21] are found by minimising some measure $F(\boldsymbol{a},\boldsymbol{y})$, for example

$$F(\boldsymbol{a},\boldsymbol{y}) = \sum_{i=1}^{m} f^2(\boldsymbol{x}_i,\boldsymbol{a}), f_i = f(\boldsymbol{x}_i,\boldsymbol{a}) = y_i - \phi(\boldsymbol{x}_i,\boldsymbol{a}).$$

The Gauss-Newton algorithm can be used to perform this least-squares minimisation [22]: given estimates of $a$, updated estimates of $\boldsymbol{a}$ are given by $\boldsymbol{a} + \boldsymbol{p}$ where $\boldsymbol{p} = -(J^\top J)^{-1} J^\top \boldsymbol{f}$. Here, $J$ is the $m \times n$ Jacobian matrix of partial derivatives $J_{ij} = \partial f_i/\partial a_i$ and $\boldsymbol{f} = (f_1, \ldots, f_m)^\top$ evaluated at $\boldsymbol{a}$. If $J$ has QR factorisation [23] given by $J = QU$, where $Q$ is an $m \times n$ orthogonal matrix and $U$ is an $n \times n$ upper-triangular matrix, then $\boldsymbol{p}$ solves $U\boldsymbol{p} = -Q^\top \boldsymbol{f}$. We are interested in how these types of calculation can be made if all quantities are represented in $R = C \times D$ rather than just as real numbers.

**Example: estimation of parameters associated with a mass artefact.**

Suppose a cylindrical mass artefact is characterised by three parameters $\boldsymbol{a} = (m,r,h)^\top$ where $m$ is its mass, $r$ the cylinder radius and $h$ the cylinder height. In addition to estimates $y_1$, $y_2$ and $y_3$ of these three parameters, we have two other measurements, an estimate $y_4$ of its volume $v = \pi r^2 h$ and an estimate $y_5$ of its density $d = m/v$. An estimate of $\boldsymbol{a}$ using all five pieces of information can be found by minimising $F(\boldsymbol{a}) = \sum_{i=1}^{5} \big(y_i - \phi_i(\boldsymbol{a})\big)^2$ where $\phi_j(\boldsymbol{a}) = a_j$, $j = 1, 2, 3$, $\phi_4(\boldsymbol{a}) = \pi a_2^2 a_3$, and $\phi_5(\boldsymbol{a}) = a_1\big(\pi a_2^2 a_3\big)^{-1}$. Using standard programming languages we can implement a Gauss-Newton algorithm to perform this minimisation, regarding $a_j$, $y_i$ and $\phi_j(\boldsymbol{a})$ as real variables. However, as soon as we apply the model constraints that $\boldsymbol{a}$ and $\boldsymbol{y}$ are members of $R = C \times D$, then the algorithm as it stands cannot be implemented since $F(\boldsymbol{a})$ involves adding a variable of dimension $\mathrm{M}^2$ to a variable of dimension $\mathrm{L}^2$, etc. We can choose to ignore the dimension information by effectively projecting all variables in $R$ to $C = \mathbb{R}$ using $\pi_C$. (Implementing the calculations in a standard programming language essentially involves applying this 'forgetful' projection.) However, this defeats the purpose of using the model constraints. A better approach is as follows.

Least-squares optimisation arises often as maximum likelihood estimation for problems involving Gaussian noise. Suppose we write the observation equations as $y_i \in \mathrm{N}(\phi_i(a),\sigma_i^2)$, where $\sigma_i$ has the same dimension as $y_i$. Maximum likelihood estimates of $\boldsymbol{a}$ are found by minimising

$$F(\boldsymbol{a}) = \sum_{i=1}^{5} f_i^2(\boldsymbol{a}), f_i(\boldsymbol{a}) = z_i - \psi_i(\boldsymbol{a}),$$

where $z_i = y_i/\sigma_i$ and $\psi_i(\boldsymbol{a}) = \phi_i(\boldsymbol{a})/\sigma_i$. Each summand function $f_i(\boldsymbol{a})$ is now dimensionless so that forming $F(\boldsymbol{a})$ is consistent with the model constraints relating to dimensions. (Applying the same change of units to $y_i$, $\phi_i$ and $\sigma_i$ means the $F(\boldsymbol{a})$ is also invariant with respect to choice of units.) The Jacobian matrix of partial derivatives is such that the dimensions are homogeneous column-wise: if $a_j$ has dimension vector $d_j$ then $J_{ij}$ has dimension vector $-d_j$, $i = 1, \ldots, m$. In fact, $d(J)$ is given by the outer product $d(J) = \mathbf{0}_D *_D (-d(\boldsymbol{a}^\top))$. The matrix product $H = J^\top J$ is such that $H_{jk}$ has dimension $-d_j - d_k$, i.e., $d(H) = (-d(\boldsymbol{a})) *_D (-d(\boldsymbol{a}^\top))$. $H$ can act on vectors of dimension $d(\boldsymbol{a})$. If $H$ has numerical inverse $V \in \mathrm{R}^{n\times n}$, set $V = (V, d(\boldsymbol{a}) *_D d(\boldsymbol{a})^\top) \in R^{n\times n}$ which can act on vectors of dimension $-d(\boldsymbol{a})$. The vector $J^\top \boldsymbol{f}$ has dimension $-d(\boldsymbol{a})$ and $\boldsymbol{p} = -V J^\top \boldsymbol{f}$ has dimension $d(\boldsymbol{a})$, so that forming $\boldsymbol{a}+\boldsymbol{p}$ is consistent with the dimension constraints.

Suppose a QR factorisation approach $J = QU$ is used to determine the update step $\boldsymbol{p}$. The orthogonal matrix $Q$ can be expressed as a product of $2 \times 2$ Givens rotations, for example. The fact that all the elements of any column of $J$ have the same dimension means that each Givens rotation and hence all the elements of $Q$ are dimensionless. Correspondingly, each element of the $j$th column of $U$ has dimension $-d_j$. The vector $\boldsymbol{g} = -Q^\top \boldsymbol{f}$ can be formed since both $Q$ and $\boldsymbol{f}$ (and hence $\boldsymbol{g}$) are dimensionless. In terms of solving

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} \\ 0 & u_{22} & u_{23} & u_{24} & u_{25} \\ 0 & 0 & u_{33} & u_{34} & u_{35} \\ 0 & 0 & 0 & u_{44} & u_{45} \\ 0 & 0 & 0 & 0 & u_{55} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ g_5 \end{bmatrix}$$

for $\boldsymbol{p}$, we have $p_5 = g_5/u_{55}$ has dimension $d_5$, $p_5$=$(g_4 - u_{45}p_5)/u_{44}$ and has dimension $d_4$, recalling that $u_{45}p_5$ has dimension $-d_5 + d_5 = 0$, etc. In other words, the QR factorisation approach can also be implemented in a way that is consistent with the dimension constraints.

In standard numerical analysis, we would estimate the (un-dimensioned) variance matrix $V_{\boldsymbol{a}}$ associated with the fitted parameters by $V_{\boldsymbol{a}} = (J^\top J)^{-1}$. For $m > n$, a posterior adjustment to this estimate is given by

$$\widehat{V}_a = \widehat{\sigma}^2 \big(J^{\mathrm{T}} J\big)^{-1}, \qquad \widehat{\sigma}^2 = \frac{1}{m-n} \sum\nolimits_i f_i^2,$$

evaluated at the solution $\mathbf{a}$, an estimate that takes into account the estimated noise associated with the observed data. These variance matrix calculations can also be made for dimensioned variables with $\mathrm{d}(\mathrm{V_a}) = \mathrm{d}(\mathrm{V}) = \mathrm{d}(\mathbf{a}) *_{\mathrm{D}} \mathrm{d}(\mathbf{a}^\top)$ and $\mathrm{d}(\widehat{\sigma}) = 0_{\mathrm{D}}$. The variance matrix V can also be computed as $\mathrm{U}^{-1}\mathrm{U}^{-\mathrm{T}}$ using the QR factorisation of J where $\mathrm{U}^{-1}$ has dimension $\mathbf{d} *_{\mathrm{D}} \mathbf{0}^{\mathrm{T}}$ and $\mathrm{U}^{-\mathrm{T}}$ has dimension $\mathbf{0} *_{\mathrm{D}} \mathbf{d}^{\mathrm{T}}$.

### 4.4. Law of propagation of uncertainty

Suppose $f(\boldsymbol{a})$ is a multivariate function of dimensioned variables $\boldsymbol{a}$ with associated dimension vector $\boldsymbol{d}$ and $f$ is such that $d(f(\boldsymbol{a})) = e$. We assume that $\boldsymbol{a}$ has been estimated with associated variance matrix $V$ with $d(V) = \boldsymbol{d} *_R \boldsymbol{d}^\top$. Given an estimate of $\boldsymbol{a}$, let $\boldsymbol{g}$ be the vector of sensitivity coefficients with $g_j = \partial f(\boldsymbol{a})/\partial a_j$. Then $d(g_j) = e - d(a_j)$. The variance associated with $f(\boldsymbol{a})$ is given by $u^2(f) = \boldsymbol{g}^\top V \boldsymbol{g}$ with $d(u^2(f)) = 2e$, where all matrix-vector calculations obey the dimensional constraints. This calculation shows how the law of propagation of uncertainty that underlies the GUM uncertainty framework [24,25] can be implemented using dimensioned variables.

## 5. Discussion and concluding remarks

The examples above show that calculations that we routinely undertake in the analysis of data can also be implemented in a way that takes into account the dimensions and associated units of the quantities through representing them in $R$. (With care, the calculations can also be made invariant to changing the dimension and units using an isomorphism of $R$.) Functional programming promotes the embedding of model constraints into software. However, to embed model constraints relating to dimension it is also necessary to represent the algebraic nature of the dimension constraints represented here in terms of the free Abelian group $\mathbb{Z}^p$, as is already done in languages such as F♯. However, F♯ only caters for matrices for which all elements have the same dimension, so that calculations associated with regression problems, uncertainty evaluation and other calculations common in metrology cannot easily be implemented for dimensioned variables or checked for dimensional consistency. Recent research [26] has shown how the functional programming toolset can be extended to support general matrix algebra associated with dimensioned variables and used to check programs implementing calculations common in data science. Importantly, the toolset can be applied to programs written in widely used languages such as MATLAB, through the use of auxiliary directives, given in comments, that enable variable dimensions to be specified, propagated and checked [3,27].

The model constraints defined by dimensioned variables vastly reduces (but does not eliminate, of course) the scope for making programming errors, including conceptual errors, that are undetected by the compiler. The extension of functional programming technologies to support dimensioned variables and other model constraints could have a significant impact in promoting trustworthy numerical computation.

## Funding statement

This work was undertaken jointly by the Mathematically Structured Programming Group of the University of Strathclyde and the National Physical Laboratory's Data Science department funded by the UK's National Measurement System programme for Data Science 2023–2024 as part of the *Tools for Trustworthiness* project.

## Acknowledgements

We thank NPL colleague Peter Harris for reviewing this paper, and are grateful to Ian Smith, NPL, and Professor Neil Ghani, Strathclyde, for their input and support. We also thank the reviewers for their many helpful comments and suggestions.

## References

[1] S. MacLane, Categories for the Working Mathematician, vol. 5, Springer-Verlag, New York, 1971. Graduate Texts in Mathematics.
[2] D.I. Spivak, Category Theory for Scientists, MIT Press, Cambridge, Mass, 2013.
[3] C. McBride, G. Nakov, F. Nordvall Forsberg, Expressive type systems for metrology, in: IMEKO TC6 International Conference on Metrology and Digital Transformation, IMEKO, Berlin, Germany, 2022, 19–21 September, 2022.
[4] L. Nielsen, Evaluation of mass measurements in accordance with the GUM, Metrologia 51 (2014) S183.
[5] Haskell Brooks Curry, Robert M. Feys, Combinatory Logic, vol. 1, North-Holland Publishing Company, 1958.
[6] William Alvin Howard, The formulae-as-types notion of construction, in: Haskell Curry, B. Hindley, Seldin J. Roger, P. Jonathan, To H.B. Curry (Eds.), Essays on Combinatory Logic, Lambda Calculus, and Formalism, Academic Press, 1980.
[7] Joachim Lambek, Philip J. Scott, Introduction to Higher Order Categorical Logic, Cambridge University Press, 1986.
[8] David I. Spivak, Robert E. Kent Ologs, A categorical framework for knowledge representation, PLoS One 7 (1) (January 2012) e24274.
[9] L. Finkelstein, M.S. Leaning, A review of the fundamental concepts of measurement, Measurement 2 (1) (1984) 25–34.
[10] G.W. Hart, Multidimensional Analysis: Algebras and Systems for Science and Engineering, Springer, 1995.
[11] B.D. Hall, Software support for physical quantities, in: Proceedings of the 9th Electronics New Zealand Conference, January 2002, 2002.
[12] R. Hanisch, S. Chalk, R. Coulon, S. Cox, S. Emmerson, Unclear units stymie science, Nature 605 (May 2022), 12 others.
[13] BIPM, The International System of Units (SI Brochure (EN)), ninth ed., 2019. May 2019.
[14] A. Thompson, B. Taylor, NIST Special Publication 811: Guide for the Use of the International System of Units (SI), National Institute of Standards and Technology, Gaithersburg, 2008. Technical report.
[15] N. Gehani, Units of measure as a data attribute, Comput. Lang. 2 (3) (January 1977) 93–111.
[16] M.R. Hansen, H. Rischel, Functional Programming Using F♯, Cambridge University Press, 2013.
[17] A.J. Kennedy, Types for units-of-measure: theory and practice, in: Zoltan Horváth, Rinus Plasmeijer, Viktória Zsók (Eds.), Central European Functional Programming School: Hungary, May 21-23, 2009, Springer Berlin Heidelberg, 2010, pp. 268–305.
[18] Oscar Bennich-Bjorkman, Steve McKeever, The next 700 unit of measurement checkers, in: Software Language Engineering, ACM, 2018, pp. 121–132, 2018.
[19] B.D. Hall, Software representation of measured physical quantities, in: F. Pavese, A. B. Forbes, N.F. Zhang, A.G. Chunovkina (Eds.), Advanced Mathematical Tools in Metrology and Testing, XII, World Scientific, Singapore, 2022, pp. 273–284.
[20] C. McBride, F. Nordvall Forsberg, Type systems for programs respecting dimensions, in: F. Pavese, A.B. Forbes, N.F. Zhang, A.G. Chunovkina (Eds.), Advanced Mathematical Tools in Metrology, vol. XII, World Scientific, Singapore, 2022, pp. 331–345.
[21] A.B. Forbes, Parameter estimation based on least squares methods, in: F. Pavese, A. B. Forbes (Eds.), Data Modeling for Metrology and Testing in Measurement Science, Birkhauser-Boston, New York, 2009, pp. 147–176.
[22] P.E. Gill, W. Murray, M.H. Wright, Practical Optimization, Academic Press, London, 1981.
[23] G.H. Golub, C.F. Van Loan, Matrix Computations, fourth ed., Johns Hopkins University Press, Baltimore, 2013.
[24] JCGM, Evaluation of measurement data — guide to the expression of uncertainty in measurement, Joint Committee for Guides in Metrology, JCGM 100 (2008) 2008.
[25] JCGM, Evaluation of measurement data — supplement 2 to the "Guide to the expression of uncertainty in measurement" — extension to any number of output quantities, Joint Committee for Guides in Metrology, JCGM 102 (2011) 2011.
[26] Conor McBride, Georgi Nakov, Fredrik Nordvall Forsberg, Measuring with confidence: leveraging expressive type systems for correct-by-construction software, Acta IMEKO 12 (1) (2023).
[27] C. McBride, G. Nakov, F. Nordvall Forsberg, A. Videla, A.B. Forbes, K. Lines, LabMate: a prospectus for types for MATLAB, in: IMEKO World Congress, August, 2024, pp. 26–29. Hamburg, Germany, 2024. To appear.

Alistair Forbes[a,*], Keith Lines[a], Fredrik Nordvall Forsberg[b], Conor McBride[b], Andre Videla[b]
[a] *National Physical Laboratory, Teddington, UK*
[b] *University of Strathclyde, Glasgow, UK*

[*] Corresponding author.
*E-mail address:* alistair.forbes@npl.co.uk (A. Forbes).