NPL Report CISE 44/00

Report to the National Measurement System Policy Unit, Department of Trade and Industry

From the Software Support for Metrology Programme

Specification of the METROS key functions

By
Robin Barker, NPL &
Geoff Morgan, NAG Ltd

February 2000

Specification of the METROS key functions

Robin Barker, NPL & Geoff Morgan, NAG Ltd

February 2000

ABSTRACT

The report describes the specification of the key functions of METROS and the issues concerning how the functions will be implemented. METROS is a web-based library of re-usable software, giving metrologists and software developers access to solutions of problems common to many metrology areas. These solutions are specifications of algorithms, implementations of those algorithms, and guidance on their use.

This report is a deliverable from the specification of key functions work package of the *Software Reuse* Project of the NMS Software Support for Metrology programme. This report was originally delivered by NAG Ltd to the NMSPU in July 1999.

ISSN 1361–407X

Authorised by Dr Dave Rayner,
Head of the Centre for Information Systems Engineering

National Physical Laboratory,
Queens Road, Teddington, Middlesex, United Kingdom. TW11 OLW

Contents

1 Introduction

This report describes the specification of the key functions of METROS (the METROlogy Software environment) and considers the issues of how they will be implemented. METROS is a web-based library of re-usable software, giving metrologists and software developers access to solutions of problems common to many metrology areas. These solutions are specifications of algorithms, implementations of those algorithms, and guidance on the use of the implementations. The METROS system is described in the NPL Report CISE 43/00: *"Design of the Metrology Software Environment (*METROS*)"*.

The bulk of the report is specification of a number of *key functions*. This list is not exhaustive, but equally we will not necessarily provide implementations of all these functions.

2 Implementation Issues

The specification of key functions and generic key functions given below are primarily in terms of a mathematical specification in order to be language and implementation independent. For an implementation of a key function it will be necessary to produce a language specific specification, e.g. a Fortran 77 specification or a C++ specification. If there are more than one implementation of a key function available in a language they should have the same specification. It is also desirable that implementations in different languages are as similar as possible so that users can easily switch languages as required.

It is envisaged that the results of the work package on mixed language programming may be used to design a generic language specification that can be used to create the individual language specifications. The experience gained on the initial specifications will be used to develop clear guidelines for the generic language and specific language specifications.

The issues that will need to be considered at the language implementation level include:

(a) Specification of data types, for example, DOUBLE PRECISION in Fortran will need to be specified. Generally the use of REAL in Fortran or its equivalent in other systems (i.e. single precision) will not be used in order to maintain accuracy.

(b) Two-dimensional arrays, these can be stored row-wise or column-wise depending on the language. Therefore, unnecessary use of two-dimensional arrays will be avoided. Clear guidelines will be developed in the light of experience with mixed-language programming. (Note also that discussions on this topic are currently underway in the Java community and the situation there should become clearer in the near future.)

(c) Specification of error handling may be implementation dependent. In many cases a simple integer value may make mixed-language programming more straightforward but for some systems, the most natural presentation is to return an error message string.

(d) Functions requiring functions as arguments (for example, non-linear least-squares solvers) will generally require the argument function to be written in the language of the calling function. The specification of the argument function will be language dependent. In some systems a reverse communication procedure may be adopted to overcome the problem.

(e) Optional arguments will have to be specified in an appropriate way depending on the language. For example, the argument can be defined as optional in Fortran 90 or Visual Basic, a null pointer can be used in C but in Fortran 77 an additional argument would be required to indicate the requirement for the option.

In developing the specifications for individual languages the emphasis will be on practicality. The aim being to produce a system that is most useful to the user rather than place too much emphasis on the elegance of the design. It is therefore possible that there may be notable differences between different implementations if that contributes to the ease-of-use of the implementations. However, the general rule will be that is there is not an overriding reason for using different interfaces the interfaces should be as similar as possible.

3 Key function specifications

3.1 Categories of key functions

We have collected the key functions into categories of similar functions.

Evaluation functions. These functions evaluate a simple function of their arguments. They are a number of sub-categories:

- **Geometric elements.** The functions calculate the distance from a point to another geometric object in two or three dimensions. The functions may be useful themselves or may be combined with solver routines to solve arbitrary geometric fitting problems.
- **Sums of series.** These functions calculate the value of standard mathematical functions which are commonly used to fit data: polynomials, splines, Fourier Series, etc.
- **Simple statistics.** These functions calculate standard statistical functions of sample data, and values and inverses of common statistical distributions.
- **Properties of materials.** These functions calculate properties of material which depend on the physical environment.

  The functions given here are for properties of materials which are common to most metrology areas.

Solver routines. These are iterative functions which solve generic equations. They can be supplied with user-functions to solve particular problems described by the functions.

Statistical routines. These are iterative routines for calculating mean and variance of measurands (computed quantities) from the mean and variance of the measurements.

Fitting routines. These are iterative functions for fitting geometric objects or mathematical functions to data.

3.2 Generic key functions

The METROS design document, NPL Report CISE 43/00: *"Design of the Metrology Software Environment (METROS)"*, describes both key functions and generic key functions. The generic key function specification contains: name, purpose/aim and, optionally, inputs and outputs. While the key function specification also contains method, computational aim and validation criterion as well as further details on inputs and outputs.

The specifications given below are intended to enable the construction of both key function and generic key function specifications. To this aim all the requirements for a generic key function have been given along with the computational aim and the further specification of input and output that is required for the final specification of the key functions. Also, where appropriate, indicative methods and valuation criterion have been given.

The material given below will be used to construct fully specified key functions as part of the next work package on implementation and acceptance testing of key functions.

3.3 Evaluation functions: geometric

3.3.1 Key function: distance (2D)

Purpose calculate the distance of one point from another in 2D

Method standard geometric methods

Inputs

Name	Type	Description
$(x_0, y_0), (x_1, y_1)$	Double: pair of pairs	points

Outputs

Name	Type	Description
d	Double	distance
J	Double: array (2)	(OPTIONAL) Jacobian
Error	*	indicates presence/nature of error

Computational Aim $d = ||(x_1, y_1) - (x_0, y_0)||$
J is the Jacobian of d w.r.t. x_0, y_0

Validation Criteria reference data sets, geometric characterisation

3.3.2 Key function: distance (3D)

Purpose calculate the distance of one point from another in 3D

Method standard geometric methods

Inputs

Name	Type	Description
$(x_0, y_0, z_0), (x_1, y_1, z_1)$	Double: pair of triples	points

Outputs

Name	Type	Description
d	Double	distance
J	Double: array (3)	(OPTIONAL) Jacobian
Error	*	indicates presence/nature of error

Computational Aim $d = ||(x_1, y_1, z_1) - (x_0, y_0, z_0)||$
J is the Jacobian of d w.r.t. x_0, y_0, z_0

Validation Criteria reference data sets, geometric characterisation

3.3.3 Key function: distance to line (2D)

Purpose calculate the distance of a point from a line in 2D

Method standard geometric methods

Inputs

Name	Type	Description
(x, y)	Double: pair	point
$(x_0, y_0), (a, b)$	Double: pair, pair	parameterisation of a line

Outputs

Name	Type	Description
d	Double	distance
J	Double: array (4)	(OPTIONAL) Jacobian
Error	*	indicates presence/nature of error

Computational Aim $d = \min_{\mathbf{p} \in \ell} ||(x, y) - \mathbf{p}||$
where ℓ is the line $\{(x_0 + ta, y_0 + tb)\}$

J is the Jacobian of d w.r.t. $(x_0, y_0), (a, b)$

Constraint: $a^2 + b^2 = 1$

Validation Criteria reference data sets, geometric characterisation

3.3.4 Key function: distance to line (3D)

Purpose calculate the distance of a point from a line in 3D

Method standard geometric methods

Inputs

Name	Type	Description
(x, y, z)	Double: triple	point
(x_0, y_0, z_0), (a, b, c)	Double: triple, triple	parameterisation of a line

Outputs

Name	Type	Description
d	Double	distance
J	Double: array (6)	(OPTIONAL) Jacobian
Error	*	indicates presence/nature of error

Computational Aim $d = \min_{\mathbf{p}\in\ell} ||(x, y, z) - \mathbf{p}||$
where ℓ is the line $\{(x_0 + ta, y_0 + tb, z_0 + tc)\}$

J is the Jacobian of d w.r.t. $(x_0, y_0, z_0), (a, b, c)$

Constraint: $a^2 + b^2 + c^2 = 1$

Validation Criteria reference data sets, geometric characterisation

3.3.5 Key function: distance to plane

Purpose calculate the distance of a point from a plane

Method standard geometric methods

Inputs

Name	Type	Description
(x, y, z)	Double: triple	point
(x_0, y_0, z_0), (a, b, c)	Double: triple, triple	parameterisation of a plane

Outputs

Name	Type	Description
d	Double	distance
J	Double: array (6)	(OPTIONAL) Jacobian
Error	*	indicates presence/nature of error

Computational Aim $d = \min_{\mathbf{p}\in P} ||(x, y, z) - \mathbf{p}||$
where P is the plane $\{\mathbf{p} : (\mathbf{p} - (x_0, y_0, z_0)).(a, b, c) = 0\}$

J is the Jacobian of d w.r.t. $(x_0, y_0, z_0), (a, b, c)$

Constraint: $a^2 + b^2 + c^2 = 1$

Validation Criteria reference data sets, geometric characterisation

3.3.6 Key function: distance to circle (2D)

Purpose calculate the distance of a point from a circle in 2D

Method geometric elements

Inputs

Name	Type	Description
(x, y)	Double: pair	point
$(x_0, y_0), r$	Double: pair, scalar	parameterisation of a circle

Outputs

Name	Type	Description
d	Double	distance
J	Double: array (3)	(OPTIONAL) Jacobian
Error	*	indicates presence/nature of error

Computational Aim $d = \min_{\mathbf{p} \in \mathcal{C}} ||(x, y) - \mathbf{p}||$
where $\mathcal{C}$ is the circle $\{\mathbf{p} : ||\mathbf{p} - (x_0, y_0)|| = r\}$

J is the Jacobian of d w.r.t. x_0, y_0, r

Validation Criteria reference data sets, geometric characterisation

3.3.7 Key function: distance to circle (3D)

Purpose calculate the distance of a point from a circle in 3D

Method standard geometric methods

Inputs

Name	Type	Description
(x, y, z)	Double: triple	point
$\mathcal{P}$	Double: array	parameterisation of a circle

Outputs

Name	Type	Description
d	Double	distance
J	Double: array	(OPTIONAL) Jacobian
Error	*	indicates presence/nature of error

Computational Aim $d = \min_{\mathbf{p} \in \mathcal{C}} ||(x, y, z) - \mathbf{p}||$
where $\mathcal{C}$ is the circle defined by $\mathcal{P}$

e.g. $\mathcal{P} = ((x_0, y_0, z_0), (a, b, c), r)$, with $a^2 + b^2 + c^2 = 1$ defines the circle $\{\mathbf{p} : ||\mathbf{p} - (x_0, y_0, z_0)|| = r, (\mathbf{p} - (x_0, y_0, z_0)).(a, b, c) = 0\}$

J is the Jacobian of d w.r.t. the independent variables in $\mathcal{P}$

Validation Criteria reference data sets, geometric characterisation

3.3.8 Key function: distance to sphere

Purpose calculate the distance of a point from a sphere

Method standard geometric methods

Inputs

Name	Type	Description
(x, y, z)	Double: triple	point
$(x_0, y_0, z_0), r$	Double: triple, scalar	parameterisation of a sphere

Outputs

Name	Type	Description
d	Double	distance
J	Double: array (4)	(OPTIONAL) Jacobian
Error	*	indicates presence/nature of error

Computational Aim $d = \min_{\mathbf{p} \in \mathcal{S}} ||(x, y, z) - \mathbf{p}||$
where $\mathcal{S}$ is the sphere $\{\mathbf{p} : ||\mathbf{p} - (x_0, y_0, z_0)|| = r\}$

J is the Jacobian of d w.r.t. x_0, y_0, z_0, r

Validation Criteria reference data sets, geometric characterisation

3.3.9 Key function: distance to cylinder

Purpose calculate the distance of a point from a cylinder

Method standard geometric methods

Inputs

Name	Type	Description
(x, y, z)	Double: triple	point
$\mathcal{P}$	Double: array	parameterisation of a cylinder

Outputs

Name	Type	Description
d	Double	distance
$\mathbf{J}$	Double: array	(OPTIONAL) Jacobian
Error	*	indicates presence/nature of error

Computational Aim $d = \min_{\mathbf{p} \in \mathcal{C}} ||(x, y, z) - \mathbf{p}||$
where $\mathcal{C}$ is the cylinder defined by $\mathcal{P}$

e.g. $\mathcal{P} = ((x_0, y_0, z_0), (a, b, c), r)$, with $a^2 + b^2 + c^2 = 1$, defines the cylinder with radius r and axis defined by $((x_0, y_0, z_0), (a, b, c))$:

$$\{\mathbf{p} : ||\mathbf{p} - (x_0, y_0, z_0)||^2 = ((\mathbf{p} - (x_0, y_0, z_0)).(a, b, c))^2 + r^2\}$$

$\mathbf{J}$ is the Jacobian of d w.r.t. the independent variables in $\mathcal{P}$

Validation Criteria reference data sets, geometric characterisation

3.3.10 Key function: distance to cone

Purpose calculate the distance of a point from a cone

Method standard geometric methods

Inputs

Name	Type	Description
(x, y, z)	Double: triple	point
$\mathcal{P}$	Double: array	parameterisation of a cone

Outputs

Name	Type	Description
d	Double	distance
$\mathbf{J}$	Double: array	(OPTIONAL) Jacobian
Error	*	indicates presence/nature of error

Computational Aim $d = \min_{\mathbf{p} \in \mathcal{C}} ||(x, y, z) - \mathbf{p}||$
where $\mathcal{C}$ is the cone defined by $\mathcal{P}$

e.g. $\mathcal{P} = ((x_0, y_0, z_0), (a, b, c), \phi)$, with $a^2 + b^2 + c^2 = 1$, defines a cone with centre (x_0, y_0, z_0), axis in the direction (a, b, c), and slope ϕ:

$$\left\{ \mathbf{p} : \frac{(\mathbf{p} - (x_0, y_0, z_0)).(a, b, c)}{||\mathbf{p} - (x_0, y_0, z_0)||} = \cos\phi \right\}$$

J is the Jacobian of d w.r.t. the independent variables in $\mathcal{P}$

Validation Criteria reference data sets, geometric characterisation

3.4 Evaluation functions: series

3.4.1 Key function: evaluation polynomial (by powers)

Purpose To evaluate a polynomial, given by its powers.

Method see Computational Aim

Inputs

Name	Type	Description
n	Integer, ≥ 1	order of polynomial $(degree + 1)$
A_i	Double: array (n)	coefficients
x	Double	abscissa

Outputs

Name	Type	Description
y	Double	value
Error	*	indicates presence/nature of error

Computational Aim

$$y = \sum_{i=1}^{n} A_i x^{i-1}$$

Validation Criteria reference data-sets

3.4.2 Key function: evaluation polynomial (Chebychev)

Purpose To evaluate a polynomial, given by Chebychev representation.

Method see Computational Aim

Inputs

Name	Type	Description
n	Integer, ≥ 1	order of polynomial $(degree + 1)$
$[x_{\min}, x_{\max}]$	Double: pair	interval of definition
A_i	Double: array (n)	Chebychev coefficients
x	Double	abscissa

Outputs

Name	Type	Description
y	Double	value
Error	*	indicates presence/nature of error

Computational Aim

$$y = \sum_{i=1}^{n} A_i T_{i-1}(x)$$

where T_i is the Chebychev polynomial of degree i on $[x_{\min}, x_{\max}]$.

Validation Criteria reference data-sets

3.4.3 Key function: spline evaluation

Purpose To evaluate a spline function

Method see Computational Aim

Inputs

Name	Type	Description
n	Integer, ≥ 1	order ($degree + 1$) of splines
N	Integer, ≥ 1	number of interior knots
$[x_{\min}, x_{\max}]$	Double: pair	interval of definition
$(c_i)_{i=1}^{n+N}$	Double: array $(n + N)$	spline coefficients
$(\lambda_j)_{j=1}^{N}$	Double: array (N)	interior knot values
x	Double	abscissa

Outputs

Name	Type	Description
y	Double	value
Error	*	indicates presence/nature of error

Computational Aim

$$y = \sum_{i=1}^{n+N} c_i B_i(x)$$

where the B_i are the spline functions defined by the λ_j.

Validation Criteria reference data-sets

3.4.4 Key function: Fourier series evaluation

Purpose To evaluate a Fourier series

Method see Computational Aim

Inputs

Name	Type	Description
n	Integer, ≥ 0	number of harmonics
$(A_i)_{i=0}^{n}$	Double: arrays $(0 \ldots n)$	(cosine) Fourier coefficients
$(B_i)_{i=1}^{n}$	Double: arrays (n)	(sine) Fourier coefficients
f	Double, > 0	(OPTIONAL) frequency
x	Double	abscissa

Outputs

Name	Type	Description
y	Double	value
Error	*	indicates presence/nature of error

Computational Aim

$$y = \frac{A_0}{2} + \sum_{i=1}^{n} (A_i \cos(2\pi i f x) + B_i \sin(2\pi i f x))$$

Validation Criteria reference data-sets

3.4.5 Key function: sum of exponentials

Purpose To evaluate a sum of exponentials

Method see Computational Aim

Inputs

Name	Type	Description
n	Integer, ≥ 1	number of terms
$(A_i)_{i=1}^n$	Double: array (n)	coefficients of the exponentials terms
$(b_i)_{i=1}^n$	Double: array (n)	"exponents"
x	Double	abscissa

Outputs

Name	Type	Description
y	Double	value
Error	*	indicates presence/nature of error

Computational Aim

$$y = \sum_{i=1}^{n} A_i e^{b_i x}$$

Validation Criteria reference data-sets

3.5 Evaluation functions: simple statistics

3.5.1 Key function: sample mean

Purpose to calculate the sample mean of a set of data

Method see Computational Aim

Inputs

Name	Type	Description
m	Integer, ≥ 1	number of data points
$(x_i)_{i=1}^m$	Double: array (m)	data
$(f_i)_{i=1}^m$	Double: array (m)	(OPTIONAL) frequencies

Outputs

Name	Type	Description
$\bar{x}$	Double	sample mean
Error	*	indicates presence/nature of error

Computational Aim

$$\bar{x} = \frac{\sum_{i=1}^{m} f_i x_i}{\sum_{i=1}^{m} f_i}$$

Validation Criteria generated data sets

3.5.2 Key function: sample standard deviation

Purpose to calculate the sample standard deviation of a set of data

Method two-pass or stable one-pass algorithm

Inputs

Name	Type	Description
m	Integer, ≥ 2	number of data points
$(x_i)_{i=1}^{m}$	Double: array (m)	data
$(f_i)_{i=1}^{m}$	Double: array (m)	(OPTIONAL) frequencies

Outputs

Name	Type	Description
s	Double	sample standard deviation
Error	*	indicates presence/nature of error

Computational Aim

$$s^2 = \frac{\sum_{i=1}^{m} f_i (x_i - \bar{x})^2}{(\sum_{i=1}^{m} f_i) - 1}$$

where $\bar{x}$ is the sample mean.

Validation Criteria generated data sets

3.5.3 Key function: geometric mean

Purpose to calculate the geometric mean of a set of data

Method see Computational Aim

Inputs

Name	Type	Description
m	Integer, ≥ 1	number of data points
$(x_i)_{i=1}^{m}$	Double: array (m), > 0	data
$(f_i)_{i=1}^{m}$	Double: array (m)	(OPTIONAL) frequencies

Outputs

Name	Type	Description
m_{geom}	Double	geometric mean
Error	*	indicates presence/nature of error

Computational Aim

$$\log m_{\mathrm{geom}} = \frac{\sum_{i=1}^{m} f_i \log x_i}{\sum_{i=1}^{m} f_i}$$

Validation Criteria generated data sets

3.5.4 Key function: harmonic mean

Purpose to calculate the harmonic mean of a set of data

Method see Computational Aim

Inputs

Name	Type	Description
m	Integer, ≥ 1	number of data points
$(x_i)_{i=1}^{m}$	Double: array (m), > 0	data
$(f_i)_{i=1}^{m}$	Double: array (m)	(OPTIONAL) frequencies

Outputs

Name	Type	Description
m_{harm}	Double	harmonic mean
Error	*	indicates presence/nature of error

Computational Aim

$$\frac{1}{m_{\mathrm{harm}}} = \frac{\sum_{i=1}^{m} (f_i/x_i)}{\sum_{i=1}^{m} f_i}$$

Validation Criteria generated data sets

3.5.5 Key function: Huber mean

Purpose to calculate mean of a set of data w.r.t. the Huber norm

Method iterative procedure

Inputs

Name	Type	Description
γ	Double	coefficient of Huber norm
m	Integer, ≥ 1	number of data points
$(x_i)_{i=1}^{m}$	Double: array (m), > 0	data
$(f_i)_{i=1}^{m}$	Double: array (m)	(OPTIONAL) frequencies

Outputs

Name	Type	Description
m_{Huber}	Double	Huber mean
s_R	Double	(OPTIONAL) Robust standard deviation
Error	*	indicates presence/nature of error

Computational Aim m_{Huber} minimises $\sum_{i=1}^{m} f_i||(x_i - m_{\text{Huber}})/s_R||_{\text{Huber}}$

where

$$||x||_{\text{Huber}} = \begin{cases} \frac{1}{2}x^2 & \text{if } |x| \leq \gamma \\ \frac{1}{2}\gamma^2 & \text{if } |x| \geq \gamma \end{cases}$$

The robust standard deviation maybe computed either using the median absolute deviation or the Huber M–estimator for variance.

Validation Criteria reference data sets

3.5.6 Key function: Gaussian distribution

Purpose to calculate the area under a Gaussian distribution

Method implementation dependent

Inputs

Name	Type	Description
μ	Double	(OPTIONAL) mean of distribution
σ^2	Double	(OPTIONAL) variance of distribution
x	Double	abscissa

Outputs

Name	Type	Description
P	Double	lower tail probability
Error	*	indicates presence/nature of error

Computational Aim

$$P = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\frac{x-\mu}{\sigma}} e^{-t^2/2} dt$$

Validation Criteria reference values

3.5.7 Key function: inverse Gaussian

Purpose to calculate the percentage point for the Gaussian distribution

Method implementation dependent

Inputs

Name	Type	Description
μ	Double	(OPTIONAL) mean of distribution
σ^2	Double	(OPTIONAL) variance of distribution
α	Double	upper tail probability

Outputs

Name	Type	Description
x	Double	abscissa
Error	*	indicates presence/nature of error

Computational Aim

$$\alpha = \frac{1}{\sqrt{2\pi}} \int_{\frac{x-\mu}{\sigma}}^{\infty} e^{-t^2/2} dt$$

Validation Criteria reference values

3.5.8 Key function: inverse Student's t-distribution

Purpose to calculate the percentage point for Student's t-distribution

Method implementation dependent

Inputs

Name	Type	Description
ν	Double	degrees of freedom
α	Double	probability

Outputs

Name	Type	Description
t	Double	abscissa
Error	*	indicates presence/nature of error

Computational Aim α is the upper tail of Student's t-distribution on ν degrees of freedom, starting from t.

Validation Criteria reference values

3.5.9 Key function: inverse χ^2 distribution

Purpose to calculate the percentage point for χ^2 distribution

Method implementation dependent

Inputs

Name	Type	Description
ν	Double	degrees of freedom
α	Double	probability

Outputs

Name	Type	Description
x	Double	abscissa
Error	*	indicates presence/nature of error

Computational Aim α is the upper tail of χ^2 distribution on ν degrees of freedom, starting from x.

Validation Criteria reference values

3.5.10 Key function: inverse F-distribution

Purpose to calculate the percentage point for F-distribution

Method implementation dependent

Inputs

Name	Type	Description
ν_1, ν_2	Double: scalar, scalar	degrees of freedom
α	Double	probability

Outputs

Name	Type	Description
F	Double	abscissa
Error	*	indicates presence/nature of error

Computational Aim α is the upper tail of F-distribution on ν_1 and ν_2 degrees of freedom, starting from F.

Validation Criteria reference values

3.6 Evaluation functions: properties of materials

3.6.1 Key function: refractive index of air

Purpose to estimate the refractive index of air

Method K.P.Birch and M.J.Downs corrections to Edlén Equation

Inputs

Name	Type	Description
t	Double, > 0	ITS-90 temperature
p	Double, > 0	atmospheric pressure
σ	Double, > 0	vacuum wavenumber
f	Double, > 0	water vapour pressure
x	Double, > 0	proportion by volume of carbon dioxide

Outputs

Name	Type	Description
n	Double, > 0	refractive index
Error	*	indicates presence/nature of error

Computational Aim Calculations in Metrologica 1993, **30**, 155–162 and Metrologica 1994, **31**, 315–316.

Validation Criteria reference data sets (comparison with tabulated values)

3.6.2 Key function: partial water vapour pressure (humidity)

Purpose to calculate partial water vapour pressure from the relative humidity

Inputs

Name	Type	Description
t	Double, > 0	temperature
r	Double, > 0	relative humidity

Outputs

Name	Type	Description
e	Double, > 0	partial water vapour pressure
Error	*	indicates presence/nature of error

Computational Aim $e = re_d(t)/100$ where e_d is the partial vapour presure as a function of dew-point temperature (see KF: partial vapour pressure (dew-point)).

Validation Criteria reference data sets (comparison with tabulated values)

3.6.3 Key function: partial water vapour pressure (dew point)

Purpose to calculate partial water vapour pressure from the dew point temperature

Inputs

Name	Type	Description
t_d	Double, > 0	dew point temperature

Outputs

Name	Type	Description
e	Double, > 0	partial water vapour pressure
Error	*	indicates presence/nature of error

Computational Aim $e = e_d(t_d)$ where e_d is given in D. Sontag *Z. Meteorol.*, 40(5):340–344, 1990.

$$
\begin{aligned}
e_d(t) = \exp\{ & -6096.9385/(t + 273.15) + 21.2409642- \\
& 2.711193(t + 273.15)/100 + 1.673952(t + 273.15)^2/10^5+ \\
& 2.433502 \log(t + 272.15) \quad \}
\end{aligned}
$$

Validation Criteria reference data sets (comparison with tabulated values)

3.6.4 Key function: density of air

Purpose to estimate the density of air

Method IBWM

Inputs

Name	Type	Description
t	Double, > 0	ITS-90 temperature
p	Double, > 0	atmospheric pressure
f	Double, > 0	water vapour pressure

Outputs

Name	Type	Description
ρ	Double, > 0	density of air
Error	*	indicates presence/nature of error

Computational Aim See 15th Edition of Tables of Physical and Chemical Constants

Validation Criteria reference data sets (comparison with tabulated values)

3.6.5 Key function: density of water

Purpose to estimate the density of water

Method various experimental formulae available

Inputs

Name	Type	Description
t	Double, > 0	ITS-90 temperature

Outputs

Name	Type	Description
ρ	Double, > 0	density of water
Error	*	indicates presence/nature of error

Computational Aim estimate of ρ as a function of t either by extrapolation from experimental data or by evaluation of formulae from curves fitted to experimental data

Validation Criteria reference data sets (comparison with tabulated values)

3.6.6 Key function: ITS-90 temperature

Purpose to calculate temperature from platinum resistance

Method ITS-90

Inputs

Name	Type	Description
σ	Double, > 0	platinum resistivity

Outputs

Name	Type	Description
t_{90}	Double, > 0	ITS-90 temperature
Error	*	indicates presence/nature of error

Computational Aim ITS-90 International Standard defines t_{90} as a function of σ, using two polynomials (of degrees 7 and 13) on disjoint intervals.

Validation Criteria reference data sets (comparison with tabulated values)

3.6.7 Key function: mass correction

Purpose determine the true mass based on reported mass

Inputs

Name	Type	Description
m'	Double, > 0	reported mass
ρ_{m}	Double, > 0	density of the object
ρ'_{a}	Double, > 0	air density at equilibrium
ρ_{s}	Double, > 0	density of the standard weights

Outputs

Name	Type	Description
m	Double, > 0	true mass
Error	*	indicates presence/nature of error

Computational Aim

$$m = m' \left[1 + \rho'_{\mathrm{a}} \left(\frac{1}{\rho_{\mathrm{m}}} - \frac{1}{\rho_{\mathrm{s}}} \right) \right]$$

Validation Criteria reference data sets (comparison with tabulated values)

3.7 Solver routines

3.7.1 Key function: linear least-squares

Purpose find the least-squares solution of a linear system of equations with constraints

Method implementation dependent

Inputs

Name	Type	Description
m	Integer, ≥ 1	number of equations
n	Integer, ≥ 1	number of unknowns
$\mathbf{A}$	Double: array ($m \times n$)	equation matrix
$\mathbf{B}$	Double: array (m)	equation vector
$m_{\mathbf{E}}$	Integer, ≥ 0	number of equality constraints
$\mathbf{E}$	Double: array ($m_{\mathbf{E}} \times n$)	equality constraints matrix
$\mathbf{F}$	Double: array ($m_{\mathbf{E}}$)	equality constraints vector
$m_{\mathbf{G}}$	Integer, ≥ 0	number of inequality constraints
$\mathbf{G}$	Double: array ($m_{\mathbf{G}} \times n$)	inequality constraints matrix
$\mathbf{H}$	Double: array ($m_{\mathbf{G}}$)	inequality constraints vector

Outputs

Name	Type	Description
$\mathbf{X}$	Double: array (n)	solution vector
$\mathbf{R}$	Double: array (n)	residual values
Δ	Double	norm of residual values
Error	*	indicates presence/nature of error

Computational Aim $\mathbf{X}$ minimises $\Delta = ||\mathbf{R}||_2$ where $\mathbf{R} = \mathbf{B} - \mathbf{AX}$, subject to $\mathbf{EX} = \mathbf{F}$ and $\mathbf{GX} \geq \mathbf{H}$.

Validation Criteria reference data sets

3.7.2 Key function: linear ℓ_1

Purpose find the least "ℓ_1" solution of a linear system of equations with constraints

Method implementation dependent

Inputs

Name	Type	Description
m	Integer, ≥ 1	number of equations
n	Integer, ≥ 1	number of unknowns
$\mathbf{A}$	Double: array ($m \times n$)	equation matrix
$\mathbf{B}$	Double: array (m)	equation vector
$m_{\mathbf{E}}$	Integer, ≥ 0	number of equality constraints
$\mathbf{E}$	Double: array ($m_{\mathbf{E}} \times n$)	equality constraints matrix
$\mathbf{F}$	Double: array ($m_{\mathbf{E}}$)	equality constraints vector
$m_{\mathbf{G}}$	Integer, ≥ 0	number of inequality constraints
$\mathbf{G}$	Double: array ($m_{\mathbf{G}} \times n$)	inequality constraints matrix
$\mathbf{H}$	Double: array ($m_{\mathbf{G}}$)	inequality constraints vector

Outputs

Name	Type	Description
$\mathbf{X}$	Double: array (n)	solution vector
$\mathbf{R}$	Double: array (n)	residual values
Δ	Double	norm of residual values
Error	*	indicates presence/nature of error

Computational Aim $\mathbf{X}$ minimises $\Delta = ||\mathbf{R}||_1$ where $\mathbf{R} = \mathbf{B} - \mathbf{AX}$, subject to $\mathbf{EX} = \mathbf{F}$ and $\mathbf{GX} \geq \mathbf{H}$.

Validation Criteria reference data sets

3.7.3 Key function: linear Chebychev

Purpose find the least minimax (ℓ_∞) solution of a linear system of equations with constraints

Method implementation dependent

Inputs

Name	Type	Description
m	Integer, ≥ 1	number of equations
n	Integer, ≥ 1	number of unknowns
$\mathbf{A}$	Double: array $(m \times n)$	equation matrix
$\mathbf{B}$	Double: array (m)	equation vector
$m_{\mathbf{E}}$	Integer, ≥ 0	number of equality constraints
$\mathbf{E}$	Double: array $(m_{\mathbf{E}} \times n)$	equality constraints matrix
$\mathbf{F}$	Double: array $(m_{\mathbf{E}})$	equality constraints vector
$m_{\mathbf{G}}$	Integer, ≥ 0	number of inequality constraints
$\mathbf{G}$	Double: array $(m_{\mathbf{G}} \times n)$	inequality constraints matrix
$\mathbf{H}$	Double: array $(m_{\mathbf{G}})$	inequality constraints vector

Outputs

Name	Type	Description
$\mathbf{X}$	Double: array (n)	solution vector
$\mathbf{R}$	Double: array (n)	residual values
Δ	Double	norm of residual values
Error	*	indicates presence/nature of error

Computational Aim $\mathbf{X}$ minimises $\Delta = ||\mathbf{R}||_\infty$ where $\mathbf{R} = \mathbf{B} - \mathbf{AX}$, subject to $\mathbf{EX} = \mathbf{F}$ and $\mathbf{GX} \geq \mathbf{H}$.

Validation Criteria reference data sets

3.7.4 Key function: non-linear least-squares

Purpose find the least-squares solution of a system of non-linear equations or to fit a non-linear model by least-squares

Method implementation dependent

Inputs

Name	Type	Description
m	Integer, ≥ 1	number of equations
n	Integer, ≥ 1	number of unknowns
f	Double: function	functions defining the residuals
$\mathbf{J}$	Double: function	Jacobian of f
$\mathbf{x}_0$	Double: array (n)	initial estimate of solution

Outputs

Name	Type	Description
$\mathbf{x}$	Double: array (n)	solution vector
$\mathbf{R}$	Double: array (n)	residual values
Δ	Double	norm of residual values
Error	*	indicates presence/nature of error

Computational Aim $\mathbf{x}$ minimises $\Delta = ||\mathbf{R}||_2$ where $\mathbf{R} = f(\mathbf{x})$

Validation Criteria reference functions and data sets

3.7.5 Key function: ODE solver

Purpose find a solution of an ordinary differential equation

Method e.g. Runga-Kutta

Inputs

Name	Type	Description
f	Double: function	function defining the equation
$[t_0, t_{\text{final}}]$	Double: pair	interval for solution
x_0	Double: array (n)	initial value of solution

Outputs

Name	Type	Description
$(t_i, \mathbf{x}_i)$	Double: array	sequence of points on the solution curve
Error	*	indicates presence/nature of error

Computational Aim $\mathbf{x}_i = x(t_i)$ where $x(t)$ is the solution of

$$x'(t) = f(x(t), t), \quad t \in [t_0, t_{\text{final}}]; \qquad x(t_0) = \mathbf{x}_0$$

Validation Criteria reference functions and values

3.7.6 Key function: zero-finder

Purpose find a zero of a function in an interval in which the function changes sign

Method e.g. bisection method

Inputs

Name	Type	Description
F	Double: function	target function
$[x_{\min}, x_{\max}]$	Double: pair	search interval
$y, z, \ldots$	Doubles	(OPTIONAL) parameters to F

Outputs

Name	Type	Description
x	Double	zero of target function
Error	*	indicates presence/nature of error

Computational Aim $F(x, y, z, \ldots) = 0$, $x \in [x_{\min}, x_{\max}]$

Validation Criteria reference functions and values

3.8 Statistics routines

3.8.1 Key function: standard variance (explicit)

Purpose evaluate measurand and compute its variance

Method delta method

Inputs

Name	Type	Description
$\mathbf{x}$	Double: array (m)	measurements
$\mathbf{V}_x$	Double: array ($m \times m$)	measurement covariance
f	Double: function	measurand function
$\mathbf{J}_f$	Double: function	Jacobian function of f
$x', \ldots$	Doubles	(OPTIONAL) parameters to f

Outputs

Name	Type	Description
y, V_y	Double: scalar, scalar	measurand and its variance
Error	*	indicates presence/nature of error

Computational Aim $y = f(\mathbf{x}, x', \ldots)$, $V_y = \mathbf{J}\mathbf{V}_\mathbf{x}\mathbf{J}'$
where $\mathbf{J} = \mathbf{J}_f(\mathbf{x}, x', \ldots)$

Validation Criteria reference functions and values

3.8.2 Key function: standard variance (implicit)

Purpose find measurand value and compute its variance

Method iterative procedure with delta method

Inputs

Name	Type	Description
$\mathbf{x}$	Double: array(m)	measurements
$\mathbf{V}_x$	Double: array($m \times m$)	measurement covariance
$[y_{\min}, y_{\max}]$	Double: pair	interval containing measurand
f	Double: function	implicit measurand function
$\mathbf{J}_f$	Double: function	Jacobian function of f
$x', \ldots$	Doubles	(OPTIONAL) parameters to f

Outputs

Name	Type	Description
y, V_y	Double: scalar, scalar	measurand and its variance
Error	*	indicates presence/nature of error

Computational Aim $f(y, \mathbf{x}, x', \ldots) = 0$, $\mathbf{J}_y V_y \mathbf{J}_y' = \mathbf{J}_\mathbf{x}\mathbf{V}_\mathbf{x}\mathbf{J}_\mathbf{x}'$;
where $y \in [y_{\min}, y_{\max}]$, $[\mathbf{J}_\mathbf{x}, \mathbf{J}_y] = \mathbf{J}_f(y, \mathbf{x}, x', \ldots)$.

Validation Criteria reference functions and values

3.8.3 Key function: expanded uncertainty (explicit, Gaussian)

Purpose evaluate measurand and compute a confidence interval, assuming a Gaussian distribution

Method delta method

Inputs

Name	Type	Description
$\mathbf{x}$	Double: array (m)	measurements
$\mathbf{V}_x$	Double: array $(m \times m)$	measurement covariance
f	Double: function	measurand function
$\mathbf{J}_f$	Double: function	Jacobian function of f
p	Double	confidence level
$x', \ldots$	Doubles	(OPTIONAL) parameters to f

Outputs

Name	Type	Description
y	Double: scalar	measurand
c	Double: scalar	measurand confidence
SE_y	Double: scalar	(OPTIONAL) std. error of measurand
Error	*	indicates presence/nature of error

Computational Aim $y = f(\mathbf{x}, x', \ldots)$, with confidence interval $[y - c, y + c]$ for a confidence level of p.

Validation Criteria reference functions and values

3.8.4 Key function: Monte-Carlo simulation (parametric)

Purpose calculate measurand value and compute its variance by Monte-Carlo simulation

Method implementation dependent

Inputs

Name	Type	Description
$\mathbf{x}$	Double: array (m)	measurements
$\mathbf{V}_x$	Double: array $(m \times m)$	measurement covariance
$\mathcal{D}$	*	distribution type
f	Double: function	measurand function
N	Integer, > 0	number of Monte-Carlo trials
$x', \ldots$	Doubles	(OPTIONAL) parameters to f
R	Double: array ?	(OPTIONAL) random numbers

Outputs

Name	Type	Description
y, V_y	Double: scalar, scalar	measurand, and its variance
y^*	Double: array (N)	measurand distribution
Acc	Double: scalar	(OPTIONAL) std. error of variance
Error	*	indicates presence/nature of error

Computational Aim y^* are the values of $f(\mathbf{x}^*, x', \ldots)$ where $\mathbf{x}^*$ is drawn from a random distribution of type $\mathcal{D}$ with mean $\mathbf{x}$ and covariance $\mathbf{V_x}$.

y and V_y are the mean and variance of y^*

Validation Criteria reference functions and tolerance values

3.8.5 Key function: Monte-Carlo simulation (non-parametric)

Purpose calculate measurand value and compute its variance by Monte-Carlo simulation

Method implementation dependent

Inputs

Name	Type	Description
$\mathbf{x}$	Double: array of array (m)	distribution of measurements
$corr$	Boolean	measurements are correlated
f	Double: function	measurand function
N	Integer, > 0	number of Monte-Carlo trials
$x', \ldots$	Doubles	(OPTIONAL) parameters to f

Outputs

Name	Type	Description
y, V_y	Double: scalar, scalar	measurand, and its variance
y^*	Double: array	measurand and distribution
Acc	Double: scalar	(OPTIONAL) std. error of variance
Error	*	indicates presence/nature of error

Computational Aim y^* is the values of $f(\mathbf{x}^*, x', \ldots)$ where $\mathbf{x}^*$ are drawn from $\mathbf{x}$, correlated or independent according to the input parameter $corr$.

y and V_y are the mean and variance of y^*

Validation Criteria reference functions and tolerance values

3.9 Fitting routines

3.9.1 Key function: linear orthogonal distance regression

Purpose to fit a straight line to data, with errors in both variables

Method least-squares, decomposition methods

Inputs

Name	Type	Description
$(x_i, y_i)_{i=1}^m$	Double: array (m) of pairs	data points

Outputs

Name	Type	Description
$(x, y), (a, b)$	Double: pair, pair	parameters specifying a line
Error	*	indicates presence/nature of error

Computational Aim $\{(x, y), (a, b)\}$ minimises $\sum_{i=1}^m d_i^2$,

where $d_i =$ distance of (x_i, y_i) from the line defined by $\{(x, y), (a, b)\}$

Validation Criteria reference data-sets

3.9.2 Key function: circle fit

Purpose to fit a circle to data

Method non-linear least-squares

Inputs

Name	Type	Description
$(x_i, y_i)_{i=1}^m$	Double: array (m) of pairs	data points
$(w_i)_{i=1}^m$	Double: array (m)	(OPTIONAL) weights

Outputs

Name	Type	Description
$(x, y), r$	Double: pair, scalar	parameters specifying a circle
Error	*	indicates presence/nature of error

Computational Aim $\{(x, y), r\}$ minimises $\sum_{i=1}^m w_i^2 d_i^2$,

where d_i = distance of (x_i, y_i) from the circle defined by $\{(x, y), r\}$

Validation Criteria reference data-sets

3.9.3 Key function: polynomial fit

Purpose to fit a polynomial to data

Method least-squares

Inputs

Name	Type	Description
$(x_i, y_i)_{i=1}^m$	Double: array (m) of pairs	data points
$(w_i)_{i=1}^m$	Double: array (m)	(OPTIONAL) weights
n	Integer, ≥ 0	order of fitting polynomial

Outputs

Name	Type	Description
$(A_j)_{j=1}^n$	Double: array (n)	coefficients
Error	*	indicates presence/nature of error

Computational Aim $(A_j)_{j=1}^n$ minimises $\sum_{i=1}^m w_i^2 (y_i - P(x_i))^2$,

where $P(x) = \sum_{j=1}^n A_j x^{j-1}$

Validation Criteria reference data-sets

3.9.4 Key function: spline fit

Purpose to fit a spline to data

Method least-squares

Inputs

Name	Type	Description
$(x_i, y_i)_{i=1}^m$	Double: array (m) of pairs	data points
$(w_i)_{i=1}^m$	Double: array (m)	(OPTIONAL) weights
n	Integer, ≥ 0	order of spline polynomial
N	Integer, ≥ 0	number of knots
$(\lambda_j)_{j=1}^N$	Double: array (N)	knot values

Outputs

Name	Type	Description
$(c_k)_{k=1}^{n+N}$	Double: array$(n+N)$	spline coefficients
Error	*	indicates presence/nature of error

Computational Aim $(c_k)_{k=1}^{n+N}$ minimises $\sum_{i=1}^m w_i^2(y_i - s(x_i))^2$,

where $s(x) = \sum_{k=1}^{n+N} c_k B_k(x)$, and the B_k are the spline functions defined by $(\lambda_j)_{j=1}^N$

Validation Criteria reference data-sets

3.9.5 Key function: Fourier series fit

Purpose to fit a Fourier series to data

Method least-squares

Inputs

Name	Type	Description
$(x_i, y_i)_{i=1}^m$	Double: array (m) of pairs	data points
$(w_i)_{i=1}^m$	Double: array (m)	(OPTIONAL) weights
n	Integer, ≥ 0	number of harmonics
f	Double, > 0	(OPTIONAL) frequency

Outputs

Name	Type	Description
$(A_j)_{j=0}^n$	Double: array $(0..n)$	(cosine) Fourier coefficients
$(B_j)_{j=1}^n$	Double: array (n)	(sine) Fourier coefficients
Error	*	indicates presence/nature of error

Computational Aim $(A_j)_{j=0}^n, (B_j)_{j=1}^n$ minimises $\sum w_i^2(y_i - F(x_i))^2$,

where $F(x) = \frac{A_0}{2} + \sum_{j=1}^n (A_j \cos(2\pi j f x) + B_j \sin(2\pi j f x))$

Validation Criteria reference data-sets

4 Conclusions

We have specified a large number of key functions which are important for metrologists. These key functions will form the core of METROS, encouraging its extension to cover all significant areas of computation in metrology.