EUMETSAT ROM SAF

RADIO OCCULTATION METEOROLOGY

The Radio Occultation Processing Package (ROPP) Utilities Module User Guide

Version 10.0

31 March 2020

The ROM SAF Consortium
Danish Meteorological Institute (DMI)
European Centre for Medium-Range Weather Forecasts (ECMWF)
Institut d'Estudis Espacials de Catalunya (IEEC)
Met Office (MetO)

Ref: SAF/ROM/METO/UG/ROPP/008

Document Author Table

	Name	Function	Date	Comment
Prepared by:	F. Warrick	ROM SAF Project Team	31 March 2020	
Reviewed by:	I. Culverwell	ROM SAF Project Team	31 March 2020	
Approved by:	K. B. Lauritsen	ROM SAF Project Manager	31 March 2020	

Document Change Record

Issue/Revision	Date	By	Description
Version 10.0	31 Mar 2020	FW	New document for tenth full release (v10.0)

Contents

1 Introduction

1.1 Purpose of this document

This document ([RD.2g]) provides a User Guide for the utilities library of the Radio Occultation Processing Package (ROPP), which includes the handling of time and date, coordinate transformations and geodesy calculations.

1.2 Applicable and reference documents

1.2.1 Applicable documents

The following documents have a direct bearing on the contents of this document.

[AD.1] Proposal for the Third Continuous Development and Operations Phase (ROM SAF CDOP-3) March 2017 – February 2022, as endorsed by Council 7th December 2016

[AD.2] Product Requirements Document (PRD). SAF/GRAS/METO/MGT/PRD/001

[AD.3] ROPP User Licence. SAF/ROM/METO/LIC/ROPP/002

1.2.2 Reference documents

The following documents provide supplementary or background information and could be helpful in conjunction with this document.

[RD.1] ROPP Architectural Design Document (ADD). SAF/ROM/METO/ADD/ROPP/001

[RD.2] The ROPP User Guides:

- [RD.2a] Overview. SAF/ROM/METO/UG/ROPP/001
- [RD.2b] ROPP_IO. SAF/ROM/METO/UG/ROPP/002
- [RD.2c] ROPP_PP. SAF/ROM/METO/UG/ROPP/004
- [RD.2d] ROPP_APPS. SAF/ROM/METO/UG/ROPP/005
- [RD.2e] ROPP_FM. SAF/ROM/METO/UG/ROPP/006
- [RD.2f] ROPP_1DVAR. SAF/ROM/METO/UG/ROPP/007
- [RD.2g] ROPP_UTILS. SAF/ROM/METO/UG/ROPP/008

1.3 Acronyms and abbreviations

AC	Analysis Correction (NWP assimilation technique)
API	Application Programming Interface
Beidou	Chinese GNSS navigation system. Beidou-2 also known as COMPASS
BG	Background
BUFR	Binary Universal Format for data Representation
CASE	Computer Aided Software Engineering
CDR	Climate Data Record
CF	Climate and Forecasts (CF) Metadata Convention
CGS	Core Ground Segment
CHAMP	Challenging Mini–Satellite Payload
CLIMAP	Climate and Environment Monitoring with GPS–based Atmospheric Profiling (EU)
CMA	Chinese Meteorological Agency
C/NOFS	Communications/Navigation Outage Forecasting System (US)
CODE	Centre for Orbit Determination in Europe
COSMIC	Constellation Observing System for Meteorology, Ionosphere & Climate
DMI	Danish Meteorological Institute
DoD	US Department of Defense
EC	European Community
ECF	Earth–centred, Fixed coordinate system
ECI	Earth–centred, Inertial coordinate system
ECMWF	The European Centre for Medium-Range Weather Forecasts
EGM–96	Earth Gravity Model, 1996. (US DoD)
EOP	Earth Orientation Parameters
EPS	EUMETSAT Polar System
ESA	European Space Agency
ESTEC	European Space Research and Technology Centre (ESA)
EU	European Union
EUMETSAT	European Organisation for the Exploitation of Meteorological Satellites
EUMETCast	EUMETSAT's primary dissemination mechanism for the NRT delivery of satellite data and products
FY-3C/D	GNSS radio occultation receivers (CMA)
GALILEO	European GNSS constellation project (EU)
GCM	General Circulation Model
GFZ	GFZ Helmholtz Centre (Germany)
GLONASS	Global Navigation Satellite System (Russia)
GNOS	GNSS Occultation Sounder (China)
GNSS	Global Navigation Satellite Systems (generic name for GPS, GLONASS, GALILEO and Beidou)
GPL	General Public Licence (GNU)
GPS	Global Positioning System (US)

GPS/MET	GPS Meteorology experiment, onboard Microlab-1 (US)
GPSOS	Global Positioning System Occultation Sensor (NPOESS)
GRACE–A/B	Gravity Recovery and Climate Experiment (US/Germany)
GRACE–FO	GRACE Follow-on experiment (US/Germany)
GRAS	GNSS Receiver for Atmospheric Sounding (onboard Metop)
GUI	Graphical User Interface
GTS	Global Telecommunications System
HIRLAM	High Resolution Limited Area Model
ICDR	Intermediate Climate Data Record
IERS	International Earth Rotation Service
ITRF	International Terrestrial Reference Frame
ITRS	International Terrestrial Reference System
IGS	International GPS Service
ISRO	Indian Space Research Organisation
JPL	Jet Propulsion Laboratory (NASA)
KMA	Korean Meteorological Agency
KOMPSAT–5	GNSS radio occultation receiver (KMA)
LAM	Local Area Model (NWP concept)
LEO	Low Earth Orbited
LGPL	Lesser GPL (*q.v.*)
LOS	Line Of Sight
Megha-Tropiques	Tropical water cycle (and RO) experiment (India/France)
METOP	Meteorological Operational polar satellites (EUMETSAT)
MKS	Meter, Kilogram, Second
MPEF	Meteorological Products Extraction Facility (EUMETSAT)
MSL	Mean Sea Level
N/A	Not Applicable or Not Available
NASA	National Aeronautics and Space Administration (US)
NMS	National Meteorological Service
NOAA	National Oceanic and Atmospheric Administration (US)
NPOESS	National Polar-orbiting Operational Environmental Satellite System (US)
NRT	Near Real Time
NWP	Numerical Weather Prediction
OI	Optimal Interpolation (NWP assimilation technique)
Operational ROM SAF	Team responsible for the handling of GRAS data and the delivery of meteorological products during the operational life of the instrument
PAZ	Spanish Earth Observation Satellite, carrying a Radio Occultation Sounder
PFS	Product Format Specifications
PMSL	Pressure at Mean Sea Level
POD	Precise Orbit Determination
Q/C	Quality Control

RO	Radio Occultation
ROC	Radius Of Curvature
ROM SAF	The EUMETSAT Satellite Application Facility responsible for operational processing of radio occultation data from the Metop satellites. Leading entity is DMI; collaborating entities are UKMO, ECMWF and IEEC.
ROPP	Radio Occultation Processing Package
ROSA	Radio Occultation Sounder for Atmosphere (on OceanSat-2 and Megha-Tropiques)
RMDCN	Regional Meteorological Data Communication Network
SAC–C	Satelite de Applicaciones Cientificas – C
SAF	Satellite Application Facility (EUMETSAT)
SAG	Scientific Advisory Group
SI	Système International (The MKS units system)
TAI	Temps Atomique International (International Atomic Time)
TanDEM–X	German Earth Observation Satellite, carrying a Radio Occultation Sounder
TBC	To Be Confirmed
TBD	To Be Determined
TDB	Temps Dynamique Baricéntrique (Barycentric Dynamical Time)
TDT	Temps Dynamique Terrestre (Terrestrial Dynamical Time)
TDS	True–of–date coordinate system
TerraSAR–X	German Earth Observation Satellite, carrying a Radio Occultation Sounder
TP	Tangent Point
UKMO	United Kingdom Meteorological Office
UML	Unified Modelling Language
UT1	Universal Time-1 (proportional to the rotation angle of the Earth)
UTC	Universal Time Coordinated
VAR	Variational analysis; 1D, 2D, 3D or 4D versions (NWP data assimilation technique)
VT	Valid or Verification Time
WEGC	Wegener Center for Climate and Global Change
WGS–84	World Geodetic System, 1984. (US DoD)
WMO	World Meteorological Organization
WWW	World Weather Watch (WMO)

1.4 Definitions, levels and types

RO data products from the Metop and Metop-SG satellites and RO data from other missions are grouped in data levels (Level 0, 1, 2, or 3) and product types (NRT, offline, CDR, or ICDR). The data levels and product types are defined below[1]. The lists of variables should not be considered as the complete contents of a given data level, and not all data may be contained in a given data level.

Data levels:

[1] Note that the level definitions differ partly from the WMO definitions: http://www.wmo.int/pages/prog/sat/dataandproducts_en.php.

- **Level 0**: Raw sounding, tracking and ancillary data, and other GNSS data before clock correction and reconstruction;
- **Level 1A**: Reconstructed full resolution excess phases, total phases, pseudo ranges, SNRs, orbit information, I, Q values, NCO (carrier) phases, navigation bits, and quality information;
- **Level 1B**: Bending angles and impact parameters, tangent point location, and quality information;
- **Level 2**: Refractivity, geopotential height, "dry" temperature profiles (Level 2A), pressure, temperature, specific humidity profiles (Level 2B), surface pressure, tropopause height, planetary boundary layer height (Level 2C), ECMWF model level coefficients (Level 2D), quality information;
- **Level 3**: Gridded or resampled data, that are processed from Level 1 or 2 data, and that are provided as, e.g., daily, monthly, or seasonal means on a spatiotemporal grid, including metadata, uncertainties and quality information.

Product types:

- **NRT product**: Data product delivered less than: (i) 3 hours after measurement (ROM SAF Level 2 for EPS); (ii) 150 min after measurement (ROM SAF Level 2 for EPS-SG Global Mission); (iii) 125 min after measurement (ROM SAF Level 2 for EPS-SG Regional Mission); item
- **Offline product**: Data product delivered from less than 5 days to up to 6 months after measurement, depending on the requirements. The evolution of this type of product is driven by new scientific developments and subsequent product upgrades;
- **CDR**: Climate Data Record generated from a dedicated reprocessing activity using a fixed set of processing software[2]. The data record covers an extended time period of several years (with a fixed end point) and constitutes a homogeneous data record appropriate for climate usage;
- **ICDR**: An Interim Climate Data Record (ICDR) regularly extends in time a (Fundamental or Thematic) CDR using a system having optimum consistency with and lower latency than the system used to generate the CDR[3].

1.5 Structure of this document

Section 2 briefly describes ROPP and its documentation.

Section 3 is the main part of this User Guide, giving an overview of the ROPP_UTILS functions and subroutines.

Appendices give brief instructions on how to build ROPP, list the files in the `ropp_utils` module, list the 'extra diagnostic data' that is produced by the various ROPP tools (usually by means of a '`-d`' option), record useful ROPP and other ROM SAF documentation, list the principal authors of ROPP, and state the copyright information that applies to various parts of the code.

[2] (i) GCOS 2016 Implementation Plan; (ii) http://climatemonitoring.info/home/terminology/.

[3] http://climatemonitoring.info/home/terminology/ (the ICDR definition was endorsed at the 9th session of the joint CEOS/CGMS Working Group Climate Meeting on 29 March 2018).

2 ROPP

2.1 ROPP introduction

The aim of ROPP is

> *. . . to provide users with a comprehensive software package, containing all necessary functionality to pre-process RO data from Level 1a (Phase), Level 1b (Bending Angle) or Level 2 (Refractivity) files, plus RO-specific components to assist with the assimilation of these data in NWP systems.*

ROPP is a collection of software modules (provided as source code), supporting data files and documentation, which aids users wishing to assimilate radio occultation data into their NWP models. As far as is practical, the ROPP software is generic, in that it can handle any standard GNSS–LEO configuration radio occultation mission (Metop, COSMIC, CHAMP, GRACE, C/NOFS, SAC–C, TerraSAR–X, TanDEM–X, Megha-Tropiques, PAZ, KOMPSAT–5 etc).

The software is distributed in the form of a source code library written in Fortran 90. ROPP is implemented using Fortran modules and derived types, enabling the use of object oriented techniques such as the overloading of routines. The software is split into several modules. Figure 2.1 illustrates the interrelationships between each module. Users may wish to integrate a subset of ROPP code into their own software applications, individually linking modules to their own code. These users may not require the complete ROPP distribution package. Alternatively, users may wish to use the executable tools provided as part of each module as stand-alone applications for RO data processing. These users should download the complete ROPP release.

ROPP contains support for a generic data format for radio occultation data (`ropp_io`), one- and two-dimensional forward models (`ropp_fm`), routines for the implementation of 1D–Var retrievals, including quality control routines (`ropp_1dvar`), pre-processing and wave optics propagator routines (`ropp_pp`), and various standalone applications (`ropp_apps`). Utility routines used by some or all of the ROPP modules are provided in an additional module (`ropp_utils`). This structure (Figure 2.1) reflects the various degrees of interdependence of the difference ROPP modules. For example, the subroutines and functions in `ropp_io` and `ropp_fm` modules are mutually indepdendent, whereas routines in `ropp_1dvar` depend on `ropp_fm`. Sample standalone implementations of `ropp_pp`, `ropp_fm` and `ropp_1dvar` (which then require `ropp_io` for file interfaces, reading and writing data) are provided with those modules and documented in the relevant User Guides.

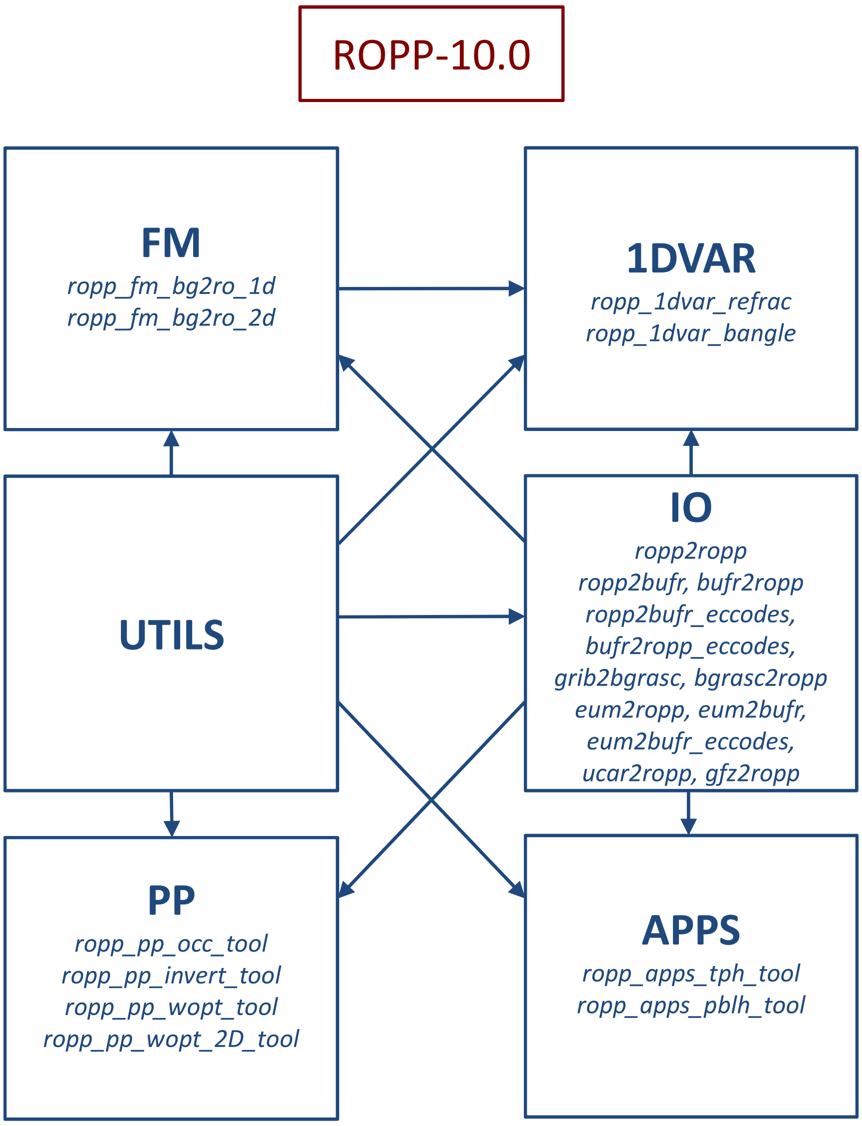

Figure 2.1: The **modules** and *tools* within ROPP-10.0. The module at the head of an arrow depends directly on the module at its tail.

2.2 User documentation

A full list of user documentation is provided in Tables D.1, D.2 and D.4. These documents are available via the ROM SAF website at http://www.romsaf.org.

The ROPP distribution website has a Release Notes file in the root directory which provides a 'Quick Start' guide to the package. This should be read before downloading the package files. Detailed build and install instructions are contained in the release notes of the individual ROPP software modules.

Module-specific user guides for the utilities (ROM SAF, 2020f), input/output (ROM SAF, 2020d), pre-processor (ROM SAF, 2020e), forward model (ROM SAF, 2020c), 1D–Var (ROM SAF, 2020a) and applications (ROM SAF, 2020b) modules describe the algorithms and routines used in those modules. These provide the necessary background and descriptions of the ROPP software for users to process radio occultation data from excess phase to bending angle or refractivity, to forward model background fields to refractivity and bending angle profiles, to simulate the propagation of GNSS radio waves through idealised atmospheric refractivity structures, and to perform 1D–Var retrievals of radio occultation data, as well as advice on how to implement ROPP in their own applications.

More detailed Reference Manuals are also available for each module for users wishing to write their own interfaces to the ROPP routines, or to modify the ROPP code. These are provided in the associated module distribution files.

Further documentation can be downloaded from the ROPP section of the ROM SAF web site http://www.romsaf.org. The full user documentation set is listed in Table D.1.

In addition to these PDF documents, most of the stand-alone application programs have Unix-style 'man page' help files which are installed during the build procedures. All such programs have summary help information which is available by running the command with the `-h` switch.

Any comments on the ROPP software should in the first instance be raised via the ROM SAF Helpdesk at http://www.romsaf.org.

References

ROM SAF, The Radio Occultation Processing Package (ROPP) 1D–Var module User Guide, SAF/ROM/METO/UG/ROPP/007, Version 10.0, 2020a.

ROM SAF, The Radio Occultation Processing Package (ROPP) Applications module User Guide, SAF/ROM/METO/UG/ROPP/005, Version 10.0, 2020b.

ROM SAF, The Radio Occultation Processing Package (ROPP) Forward model module User Guide, SAF/ROM/METO/UG/ROPP/006, Version 10.0, 2020c.

ROM SAF, The Radio Occultation Processing Package (ROPP) Input/Output module User Guide, SAF/ROM/METO/UG/ROPP/002, Version 10.0, 2020d.

ROM SAF, The Radio Occultation Processing Package (ROPP) Pre-processor module User Guide, SAF/ROM/METO/UG/ROPP/004, Version 10.0, 2020e.

ROM SAF, The Radio Occultation Processing Package (ROPP) Utilities module User Guide, SAF/ROM/METO/UG/ROPP/008, Version 10.0, 2020f.

3 Overview of the `ropp_utils` module

The `ropp_utils` module contains a collection of utility routines which are used by other ROPP modules. They are not intended to be called directly by user applications, so they are not documented in any detail here. This chapter gives only an overview of the library which is divided into sub-libraries by related functionality. The reader is directed to the ROPP Utils Reference Manual (SAF/ROM/METO/RM/ROPP/001).

Since the `ropp_utils` does not have any executables of its own, most users will not need to interact with it directly. However, those users wishing to modify or add to ROPP may find this User Guide gives a useful description of the functions and subroutines available in the `ropp_utils` module.

3.1 `Missing data values`

The `ropp_utils` module defines a set of parameters to indicate and test a 'missing' or 'invalid' data value used by any ROPP routine. These are set in the main module file `common/ropp_utils.f90`.

- `ropp_MDFV` (=`-99999000.0`) is used to set missing (invalid) data for real ROPP parameters.
- `ropp_MDTV` (=`-9999.0`) is used for testing invalid real parameter values. Any real number less than this value can be assumed to be 'missing'.
- `ropp_ZERO` (=`0.0`) is used to set parameters to zero.
- `ropp_ZDTV` (=`1.0e-10`) is used to test for (almost) zero values.
- `ropp_MIFV` (=`-999`) is used to set missing (invalid) data for integer ROPP parameters.
- `ropp_MITV` (=`-99`) is used for testing invalid integer parameter values. Any integer less than this value can be assumed to be 'missing'.

3.2 `ropp_messages`

These routines provide an interface to write information and error messages to stdout or stderr from ROPP routines. The utilities were originally written by Christian Marquardt as personal code, independent of the ROM SAF. The author grants a free-use licence for all of this code. The utilties have since been modifed and extended for ROPP.

A `msg_MODE` parameter is used to control the level of output diagnostic information output by ROPP routines. The available options are

- QuietMode - only output error messages to stdout, no info/warnings
- NormalMode - output all info and warnings and errors to stdout
- VerboseMode - as NormalMode, but also output diagnostic/debug messages to stdout

The required `msg_MODE` may be altered either within a program routine, e.g.

```
USE messages
msg_MODE = VerboseMode            ! Enable all messages
CALL message(msg_diag, ``The result is...'')
msg_MODE = NormalMode             ! Re-set to normal output level
```

or by implicitly setting the default value in the `ropp_messages/messages.f90` file before compiling, or by setting the environment variable `ROPP_MSG_MODE` on the command line. Note that VerboseMode can be selected when running any of the stand-alone tools provided with ROPP by running it with a '`-d`' command-line option.

With effect from ROPP8.0, the interface to these routines has been extended to ensure that the unix return code $? produced by running any of the ROPP tools equals (0, 1, 2, 3) if the tool executed (OK, with at least one warning, with at least one error, with a fatal error) respectively.

3.3 Unitconvert

A collection of low-level F90 routines for converting data between standard units used within ROPP modules. The conversion scaling factors and offsets for a given unit conversion operation are defined in routine `ropp_unit_conversion.f90`. The set of available unit conversions provided may be extended by a user as required. These unit conversion routines are called automatically from within `ropp_io` module read and write routines, and from within `ropp_fm` and `ropp_1dvar` routines to ensure variables have required units.

Structure of the code:

- `unitconvert.f90` is the module file containing the interface block
- `ropp_unit_conversion.f90` contains all the conversion factors (slopes and intercepts)
- `ut_convert.f90` contains the subroutines that do the convsersion, with separate routines for each form of input data, for example the number of array dimensions.

3.4 Geodesy

A collection of low-level F90 geodetic conversion routines to convert to/from geometric/geopotential heights and compute Earth's effective radius and gravity. These routines are based on Somigliana's equation to compute height scales relative to the WGS-84 reference ellipsoid. A more thorough guide to the geodesy in ROPP is provided in the ROM SAF Report 14 (Culverwell, 2013), but an overview of the key functions and their equations is provided here:

- `geodesy.f90`: module file containing interface blocks to the geodesy functions.
- `gravity.f90`: returns gravity (m/s^2) for a given latitude ϕ and, optionally, for a given height h above the WGS-84 reference ellipsoid. Somigliana's equation for the surface gravity $g(0, \phi)$, as currently used in ROPP (`ropp_utils/geodesy/gravity.f90`), takes the form

$$g(0,\phi) = g_{\text{eq}}(1+k_s\sin^2\phi)/\sqrt{1-e^2\sin^2\phi}$$

where

$$k_s(\text{Somigliana's constant}) = (cg_{\text{po}} - ag_{\text{eq}})/ag_{\text{eq}} \approx 1.9\times10^{-3}$$

and

$$e(\text{eccentricity}) = \sqrt{(a^2-c^2)/a^2} \approx 8.2\times10^{-2}.$$

The optional adjustment for height h uses the effective radius function from this module and is performed as follows:

$$R_{\text{lat}} = R_{\text{eff}}(\phi)$$

$$g(h,\phi) = g(0,\phi)\frac{R_{\text{lat}}^2}{(R_{\text{lat}}+h)^2}$$

- `great_circle_distance.f90`: A great circle is one that circles the sphere and encloses a plane containing the sphere's centre. This function, assuming a spherical Earth, calculates the great circle distance between two lat/lon points (or two arrays of lat/lon points). The great circle distance between two latitudes $\phi_{1,2}$ and longitudes $\theta_{1,2}$ is calculated as follows:

$$y = \sqrt{(\cos\phi_2\sin\Delta\theta)^2 + (\cos\phi_1\sin\phi_2 - \sin\phi_1\cos\phi_2\cos\Delta\theta)^2}$$

$$x = \sin\phi_1\sin\phi_2 + \cos\phi_1\cos\phi_2\cos\Delta\theta$$

$$\Delta\phi = \arctan\left(\frac{y}{x}\right)$$

$$\text{Distance} = \Delta\phi\times R_{\text{Earth}}$$

- `geometric2geopotential.f90` converts geometric to geopotential heights using the equation below, where h is geometric height, g_{ref} is 9.80665m/s^2, and ϕ is latitude. Geopotential height adjusts gravity for height and latitude. Calls `gravity` and effective radius routines. Output heights keep same reference as input i.e. vs ellipsoid or vs geoid.

$$Z(h,\phi) = \left(\frac{g(0,\phi)}{g_{\text{ref}}}\right)\left(\frac{R_{\text{eff}}(\phi)}{R_{\text{eff}}(\phi)+h}\right)h.$$

- `geopotential2geometric.f90`: Inverse of the above routine.

$$h(Z,\phi) = \left(\frac{R_{\text{eff}}(\phi)}{(g(0,\phi)/g_{\text{ref}})R_{\text{eff}}(\phi) - Z} \right) Z,$$

- `r_eff.f90`: function to calculate effective radius at a given latitude. In ROPP, R_{eff} for latitude ϕ is given by:

  $$R_{\text{eff}}(\phi)/a = 1/(1 + f + m - 2f \sin^2 \phi)$$

  where

  $$a = 6378137.0\text{m}; \;\; f = 0.003352811; \;\; m = 0.003449787$$

3.5 Arrays

A collection of low-level Fortran-90 array manipulation routines supporting all data types. The utilities were written by Christian Marquardt as personal code, independent of the ROM SAF. The author grants a free-use licence for all of this code.

As with the rest of the ROPP UTILS module, ROPP users do not need to call these routines diectly. However, the following code description serves as a reference for those wishing to modify the code or add new code. Overview of the code:

- `arrays.f90`: module file containing interfaces for the array functions and subroutines.
- `blend.f90`: Calculate the cosine weights for blending two arrays. Then, weights w, arrays A and B can then be blended as follows:

  $$\text{Blended Array} = A(1-w) + B * w$$

- `cross_product.f90`: given two 3x1 arrays, return cross product in new 3x1 array.
- `getrun.f90`: Return Nth run of consecutive integers from an integer array.
- `iminloc.f90` and `imaxloc.f90`: functions to find location of first minimum/maximum element in array. Uses Fortran intrinsic minloc/maxloc but returns scalar instead of rank one array.
- `isinrange.f90`: Function to check that all elements of input array are within bounds set by 2x1 array 'range', returns result as a logical.
- `l2norm.f90`: Calculate the L2 norm (Euclidean) of a 1D vector, which is the result of a dot product of a vector with itself, square rooted.
- `locate.f90`: Function which, given monotonically sorted array, finds the index/indices of input 'point(s)' in that array.

- `outer_product.f90`: Given two numerical 1D arrays (x and y), return 2D matrix z containing product of each pair of elements from input array:

$$z(i, j) = x(i) \times y(j)$$

- `outer_and.f90`: Similar to `outer_product.f90`, given two 1D logical arrays, return 2D matrix of .AND. operation on their elements.
- `sort.f90`: based on quick sort: choose a pivot point and move those elements less than or greater than pivot into two smaller lists, do the same on those lists, and so on. Functions 'sort' and 'sorted' return result as indices or as new array. They each call subroutine 'quick_sort' that does the actual sorting.
- `uniq.f90`: needs a monotonic array from `sort` so that repeated elements are adjacent. Returns indices of unique elements.
- `callocate.f90`: Allocate array to number of counts given and initialise to supplied value (default is to initialise to zero).
- `copy_alloc.f90`: Allocate 'newarray' to the dimensions of the input array, and copy the contents.
- `nruns.f90`: Function that finds number of runs of consecutive integers in an array by using the `CSHIFT` Fortran intrinsic on an array and comparing to the original array.
- `preallocate.f90`: Function that changes size of array, keeping old contents if possible and returns a new pointer array. If new array size is larger, the extra elements will just be filled by whatever values the memory has. If new array is smaller, elements that don't fit into the new array will be lost.
- `reallocate.f90`: As `preallocate` above, but a subroutine creating array 'newarray' which input array ends up pointing to.
- `reverse.f90`: Function that reverses order of input 'array' and returns as array 'reversed'. The dimension to reverse can be given as an optional argument, otherwise dimension 1 is reversed.
- `setminus.f90`: Remove elements in array b from array a and return resulting array c.
- `swap.f90`: Subroutine that swaps the values of its two input arguments.
- `unit_vector.f90`: Function that calculates unit vector by calling `L2norm` function detailed above on input vector.
- `where.f90`: Given logical mask, return array of pointers to true elements of mask.

3.6 `Datetime`

A collection of low-level F90 date and time conversion routines. The utilities were developed within the Met Office outside the context of the ROM SAF and represents pre-existing software (PES). This code is Crown copyright.

- `DateTimeTypes`: Set parameters and format strings.
- `DateTimeProgs`: Interface block.
- `GpsSecond`: Calculate GPS time (total number of seconds since the midnight between 5/1/1980 to 6/1/1980) from calendar date, or vice versa.
- `CalToJul`: Convert between calendar date and Julian date - number of days since 1-Jan-4713 BCE. Julian date can be fractional and also considers midday as start of day.
- `ConvertDT`: Convert between date formats long1 (dd-MMM-yyyy hh:mm:ss.ttt), long2 (yyyy-MM-dd hh:mm:ss.ttt) and short (yyyymmddhhmmss).
- `Date_and_Time_UTC`: Similar to Fortran intrinsic `DATE_AND_TIME`, but returns system time in UTC.
- `DateTimeOffset`: Apply supplied time offset to input calendar date. Uses CalToJul.
- `MonthOfYear`: Convert month either direction between character string or and number e.g. 'March' to 3.
- `TimeSince`: Given a calendar date, calculate time since supplied 'base' time, or vice-versa.
- `LeapSeconds`: Calculates the total number of leapseconds to have accrued up to a given calendar date.
- `GPStoUTC`: Converts between GPS and UTC times, by accounting for leapseconds.
- `AugmentUTC`: Add a specified length of time to a given UTC, accounting for leapseconds.

The last three, which were introduced at ROPP-9.1, are fully described in the ROPP 9.1 Change Log document (ROM SAF, 2019).

3.7 Coordinates

A collection of low-level F90 coordinate manipulation routines. Functionality includes routines to convert cartesian position vectors between Earth Centred Fixed and Earth Centred Inertial reference frames, to convert between cartesian and geodetic coordinates, to compute impact parameter, occultation point, radius of curvature and undulation (i.e., the difference between the EGM96[1] geoid and the WGS84[2] ellipsoid). Vector manipulation routines (vector product, vector angle, rotation) are also included. With effect from ROPP9.1, this sub-library contains a routine to convert coordinates from the J2000 ECI frame used by EUMETSAT to an intermediate ECI frame, either the classical equinox-based 'true-of-date' frame or the CIO-based frame.[3] The last are fully described in the ROPP 9.1 Change Log document (ROM SAF, 2019).

[1] See http://cddis.nasa.gov/926/egm96/egm96.html.
[2] See http://earth-info.nga.mil/GandG/wgs84/index.html.
[3] See IERS Technical Note 36 (https://www.iers.org/nn_11216/IERS/EN/Publications/TechnicalNotes/tn36.html) for details.

3.8 `Misc`

Miscellaneous utilities used by other ROPP modules. `FileDelete.f90` and `GetIOUnit.f90` are low-level F90 file handling routines. `ToUpper.f90` and `ToLower.f90` are low-level F90 string handling routines.

3.8.1 `typeSizes`

`typeSizes.f90` is a public-domain F90 module written by Robert Pincus (Cooperative Institute for Meteorological Satellite Studies, Madison) which provides named kind parameters for use in declarations of real and integer variables with specific byte sizes. It is a copy of the same file included in the 3rd party netCDF package, but is bundled with, and used by, ROPP as a stand-alone tool to provide a standardised type naming convention. This is ‘freeware’ provided complete and ‘as-is’ under the terms of usage. Users of `ropp_utils` must respect the same conditions in turn.

References

ROM SAF, ROPP Change Log: v9.0 to v9.1, SAF/ROM/METO/SRN/ROPP/016, Version 1.0, 2019.

A Installing and using ROPP

A.1 Software requirements

ROPP is written in standard Fortran 95. Thus, compilation and use of the routines forming ROPP require the availability of standard ISO-conforming compilers. Fortran 95 was preferred over Fortran 90 because it has a number of convenient features. In particular, it allows elemental functions and pointers can be nullified when they are declared.

A.2 Software release notes

The latest ROPP distribution is available for download via the ROM SAF website http://www.romsaf.org. The ROPP Release Notes available from the ROPP download page and provided with the main ROPP download tarfile gives instructions for unpacking and installing the complete ROPP package, or individual modules. Users are strongly recommended to refer to the ROPP Release Notes and use the build and configure tools described therein. The information contained here is intended to complement the ROPP Release Notes. Where any contradiction between the User Guide and ROPP Release Notes exist, the ROPP Release Notes page is considered to be the most up-to-date latest information.

A.3 Third-party packages

To fully implement ROPP, the code uses some standard third-party packages. These are all non-commercial and cost-free. Note that third-party codes are only needed by the `ropp_utils`, `ropp_io` and `ropp_pp` modules, so are optional if these modules are not required by the user.

All third-party code or packages used by ROPP are, by definition, classed as 'Pre-Existing Software' and all rights remain with the originators. Separate rights licences may be part of these distributions — some may have a licence which may impose re-distribution restrictions — and such licences must be adhered to by users.

If a third-party package is required, this must be built and installed before attempting to build the ROPP code. For convenience, these packages should be installed to the same root path as ROPP. It is highly recommended that the package is compiled using the same compiler and using the same compiler flags as will be used to build the ROPP code. Example configure scripts for supported compilers are provided in the `ropp_build` module available from the ROPP download website. See Section A.4 for further details.

A.3.1 NetCDF (optional in principle)

The input/output library `ropp_io` uses Unidata's `netCDF` data format. Thus, the `netCDF` library and its associated utility programs (like `ncdump`, `ncgen`) are required and must be properly installed on the user's system before the compilation of the `ropp_io` package can be attempted. `netCDF` may also be used for reading MSIS or BAROCLIM climatology data as part of the `ropp_pp` module.

The SAF provides versions of the `netCDF` distribution, which have been successfully integrated with ROPP, alongside the ROPP distribution. This may not be the most recent distribution. Latest versions are freely available from

http://www.unidata.ucar.edu/software/netcdf/

With effect from ROPP9.0, ROPP netCDF build support for 'classic' netCDF-4 has been dropped, which implies a need for HDF5 and, optionally, ZLIB libraries. These last two can be found at

https://support.hdfgroup.org/HDF5/

and

http://www.zlib.net/

respectively.

In addition, the supported versions of the netCDF library are now split into two parts: a netCDF-Core library, written in C, and a netCDF-Fortran interface. The ROPP `buildpack` script (see Sec A.4 for more details) allows installation of these libraries as follows:

```
> buildpack zlib <compiler>
> buildpack hdf5 <compiler>
> buildpack netcdf <compiler>    (the netCDF-Core library)
> buildpack netcdff <compiler>   (the netCDF-Fortran library)
```

These packages need to be installed in this order, since each depends on the previous one. Note, however, that the `zlib` and the `HDF5` libraries may already be installed as part of a standard Linux distribution, in which case, of course, the user need not build a local version.

Note that the `tests` subdirectory of the `ropp_io` distribution contains a simple test to check if the `netCDF` installation works; see Section A.7 for details.

A very useful complementary set of tools for handling and manipulating netCDF data files are the netCDF Operators `nco`[1]. While the latter are not required for using `ROPP` libraries and sample applications, we highly recommend them.

Some example and test programs provided with the `ropp_pp`, `ropp_apps`, `ropp_fm` and `ropp_1dvar` packages read data via `ropp_io`. A complete installation of the `ropp_io` library is therefore required if the test programs or one of the sample applications are to be run. As a consequence, the complete installation of these packages also requires the availability of `netCDF`. Note, however, that the libraries `libropp_pp.a`, `libropp_apps.a`, `libropp_fm.a` and `libropp_1dvar.a` can be compiled and installed without `ropp_io`

[1] See `http://nco.sourceforge.net/`.

and therefore without `netCDF`; the configuration script will recognise the absence of these libraries and only compile and install the core pre-processor, forward model or 1DVar routines (i.e. those with no dependencies on `netCDF` or `ropp_io`).

A.3.2 BUFR (optional)

The GNSS-RO BUFR encoder/decoder tools `ropp2bufr` and `bufr2ropp` in `ropp_io` require either the Met Office's 'MetDB' or the ECMWF BUFR library to be pre-installed. Alternatively, the BUFR encoder/decoder tools `ropp2bufr_eccodes` and `bufr2ropp_eccodes` can be used if the ECMWF ecCodes library is pre-installed. If no BUFR library is detected by the installation configure script, then these tools will not be built.

The tools to BUFR-encode EUMETSAT-format grouped netCDF data, `eum2bufr` and `eum2bufr_eccodes` in `ropp_io`, require the ECMWF BUFR library or ECMWF ecCodes library to be pre-installed, respectively.

The MetDB BUFR package is available without charge on request from the ROPP Development Team but with some licence restrictions. The ECMWF BUFR package is licensed under the GNU/GPL and can be downloaded from:

https://software.ecmwf.int/wiki/display/BUFR

The ECMWF ecCodes package is licensed under Apache (2.0), and can be downloaded from:

https://confluence.ecmwf.int/display/ECC/ecCodes+Home

Both libraries generate essentially identical data when decoded (there may be non-significant round-off differences due to use of single– vs. double–precision interfaces). While the MetDB library is easier to install from a portability point of view, the ROPP `buildpack` script makes the ECMWF installation compatibly with ROPP more transparent. Therefore users can employ whichever BUFR package they prefer. Thus, the MetDB library could be built with

```
> buildpack bufr <compiler>
```

or

```
> buildpack mobufr <compiler>
```

while the ECMWF BUFR library would be be built with

```
> buildpack ecbufr <compiler>
```

and the ECMWF ecCodes library would be be built with

```
> buildpack eccodes <compiler>
```

In order to install BUFR tables and related files, and for the applications to find them at run-time, an environment variable must be pre-defined to the path to these files. For instance, for the MetDB library:

```
> export BUFR_LIBRARY=<path>/data/bufr/
```

or for the ECMWF library:

```
> export BUFR_TABLES=<path>/data/bufr/
```

Note that in both cases, the path must currently be terminated with a '/' character, although this restriction has been relaxed for later (v20+) releases of the MetDB BUFR library. By default, the `buildpack` script will set `<path>` to be `ROPP_ROOT`.

A.3.3 GRIB (optional) - either GRIB_API or ecCodes

The GRIB background reading tool `grib2bgrasc` in `ropp_io` requires either the ECMWF GRIB_API library or ECMWF ecCodes library to be pre-installed. If neither is detected by the installation configure script, then this tool will not be built.

The ECMWF GRIB_API package is licensed under Apache (2.0), and can be downloaded from:

https://software.ecmwf.int/wiki/display/GRIB/

The ROPP `buildpack` script allows installation of the GRIB_API by typing:

```
> buildpack grib <compiler>
```

The ECMWF ecCodes package is licensed under Apache (2.0), and can be downloaded from:

https://confluence.ecmwf.int/display/ECC/ecCodes+Home

The ROPP `buildpack` script allows installation of ecCodes by typing:

```
> buildpack eccodes <compiler>
```

A.3.4 SOFA (optional)

The routines in `ropp_utils` that transform coordinates between reference frames have the option of using the IAU Standards of Fundamental Astronomy (SOFA) library to convert between some frames. If this library is unavailable, less sophisticated formula-based versions of the routines will be used instead.

The SOFA libraries are freely available for use, provided the routines are not modified in any way. They can be downloaded from

http://www.iausofa.org/

The ROPP `buildpack` script allows installation of the SOFA library by typing:

```
> buildpack sofa <compiler>
```

A.3.5 RoboDoc (optional)

The ROPP Reference Manuals have been auto-generated using the RoboDoc documentation tool[2] All source code, scripts, etc. have standardised header comments which can be scanned by RoboDoc to produce various output formats, including LaTeX and HTML. If code (and in particular the header comments) is modified, RoboDoc can optionally be used to update the documentation. This tool is not required in order to build the ROPP software.

A.3.6 autoconf and automake (optional)

The `automake` and `autoconf` tools, common on most Linux and Unix systems, are not necessary to build the ROPP package as provided, but are useful if any modifications are made to the code or build systems to re-generate the package `configure` files. Versions at, or higher than, v1.9 are required to support some of the m4 macros defined in the ROPP build system.

A.4 BUILDPACK script

The ROPP package distribution includes a collection of configure and build scripts for a number of compilers and platforms suitable for ROPP and the dependency packages. A top-level BASH shell script `buildpack` is provided which may be used to automate the build of any ROPP module or dependency package in a consistent way, using the appropriate configure scripts. Use of `buildpack` is therefore highly recommended for first time build and less experienced users. Summary usage can be obtained using

```
> buildpack -h
```

In general, to build and install a package,

```
> buildpack <package> <comp> [[NO]CLEAN]
```

where <package> is one of the supported package names (e.g. `ropp_fm`, `ropp_io`, `netcdf`, `mobufr`, etc.) and <comp> is the required compiler (e.g. `ifort`, `gfortran`, etc.).

The `buildpack` script assumes that all tarball files and configure scripts provided with the ROPP distribution are placed in the same working directory. Packages will be decompressed here and installed to the `ROPP_ROOT/<comp>` target directory. The script automates the `configure` – `make` – `make install` build cycle described below. Further information on the `buildpack` script are provided in the ROPP Release Notes.

The shell scripts `build*_ropp`, `build_deps` and `build_ropp` have also been provided to help automate the build process by calling `buildpack` with a pre-determined sequence of packages or compilers, and to save a copy of all screen output to a disk log file. Users should review and edit these to suit their requirements. Using these tools, a complete check out of ROPP from scratch can be effected by running (in order):

[2] See http://rfsber.home.xs4all.nl/Robo/robodoc.html.

```
> buildzlib_ropp <compiler>
> buildhdf5_ropp <compiler>
> buildnetcdf_ropp <compiler>  (note that this builds the Core and Fortran libs)
> buildmobufr_ropp <compiler> or buildecbufr_ropp <compiler> or buildeccodes_ropp <compile
> buildgrib_ropp <compiler> or buildeccodes_ropp <compiler>
> buildsofa_ropp <compiler>
> build_ropp <compiler>
```

Or, even more quickly:

```
> build_deps <compiler> zlib hdf5 netcdf netcdff mobufr/ecbufr grib sofa
> build_ropp <compiler>
```

A.5 Building and installing ROPP manually

The low-level build sequence performed by `buildpack` may be implemented manually by more experienced users. After unpacking, all packages are compiled and installed following the `configure` – `make` – `make install` cycle.

1. First run the command `configure` to check for the availability of all required libraries. `configure` allows the user to specify compiler options, paths to libraries and the location where the software shall eventually be installed, on the command line or as environment variables. Based on this information, `configure` generates user specific `Makefiles`, allowing a highly customised configuration and installation of the software.
2. Compilation is then initiated with the command `make`.
3. If building the software was successful, a `make install` will install libraries, header and module files as well as any executables in the directories specified by the user via the configure step.

Note that the ROPP modules partially depend on each other. In particular, all packages require that `ropp_utils` has been installed successfully. This package therefore needs to be compiled and installed first. Most packages make use of the `ropp_io` package for sample applications and testing, and should therefore be installed next if these are required. Note that users wishing to use ROPP source code directly in their own applications need not install the `ropp_io` module. If the `ropp_io` module is not available at build time, only the source code libraries will be compiled. We thus recommend the following build order:

i) Third-party packages: `zlib`, `hdf5`, `netcdf`, `netcdff`, `mo/ecbufr`, `grib` (as required)
ii) `ropp_utils`
iii) `ropp_io` (if required)
iv) `ropp_pp` (if required)
v) `ropp_apps` (if required)
vi) `ropp_fm` (if required)
vii) `ropp_1dvar` (if required)

Note that *all* libraries need to be built with the same Fortran compiler, and preferably with the same version of the compiler as well.

Supported Fortran (and C) compilers are listed in the Release Notes distributed with the ROPP package.

A.5.1 Unpacking

Once the required third-party software packages have been installed successfully, the ROPP packages can be installed. The complete ROPP package and individual modules are distributed as gzipped tar (.tar.gz) files. The complete package file name consists of the version name (e.g. ropp-9.1.tar.gz). This file contains the complete ROPP distribution. The module file names consist of the package's name (e.g. ropp_utils) and version (e.g. 9.1), as in ropp_utils-9.1.tar.gz. If GNU tar is available (as on Linux systems), gzipped tar files can be unzipped with

```
> tar -xvzf ropp-9.1.tar.gz
```

Older, or non-GNU, versions of tar might need

```
> gunzip -c ropp-9.1.tar.gz | tar -xv
```

In all cases, a new subdirectory named (in the above example) ropp-9.1 will be created which contains the source code of the complete package.

A.5.2 Configuring

Details on the installation procedure for the individual packages can be found in the files README.unix and README.cygwin for the installation under Unix and Windows (with Cygwin), respectively. Here, we provide a brief example for a Unix or Linux system.

Unpacking the ropp_build package will create the configure/ sub-directory containing a number of mini-scripts for local build configuration. The files have names <package>_configure_<compiler>_<os> where <package> is the package name (ropp, netcdf), <compiler> is the compiler ID (ifort, nagfor, pgf95, ...) and <os> is the operating system ID, as output by the uname(1) command but entirely in lower case (linux, cygwin, ...). Note these configure mini-scripts are also used by the high-level buildpack script. The example configure scripts for specific platforms and compilers may need to be edited for optimal local use, or users may create their own following one of the examples.

The main configure scripts provided assume that the external libraries and individual ROPP modules are all installed under $ROPP_ROOT, i.e. the libraries can be found in the directory $ROPP_ROOT/lib and/or $ROPP_ROOT/lib64, and header and module files in $ROPP_ROOT/include. The $ROPP_ROOT location should be specified as an environment variable, e.g,

```
> export ROPP_ROOT=$HOME (for sh, ksh and bash users)
> setenv ROPP_ROOT $HOME (for csh and tcsh users)
```

For most compilers, this means that the two paths to the header and module files need to be specified via the proper compiler options — usually via the -I option. The linker also needs to know where libraries are located; on most Unix systems, this can be achieved by specifying the -L option at link time. Users are referred to the examples provided in the configure package for further details.

Running the appropriate script from configure/ will set the required compiler flags and specify the header, module and library paths before running the configure script. For example if the Fortran 95 compiler is named (say) ifort, the following command would be sufficient to configure a package for later compilation:

```
> cd ropp_<module>
> ../configure/ropp_configure_ifort_linux
```

The configure script will check for all required libraries and add the required options for the linker. If configure is not successful finding the required libraries, an error message will be produced, and further compilation will not be possible. Should the configuration step fail entirely, the file config.log created during the run of configure usually gives some clues on what went wrong; the most likely reason for failing is that compiler or linker options (and in particular paths to include files or libraries) are not set correctly.

Note that ropp_io may optionally use other external libraries in order to support additional features. For example, the ropp_io library will provide two conversion tools from ROPP to BUFR and back if a supported BUFR library is found. The existence of such additional libraries is also checked during configure. If these libraries are missing, however, the installation will proceed without building the parts related to the missing library. Should the build process fail to find usable BUFR libraries, for example, and therefore fail to build the BUFR tools, config.log should again provide evidence on what went wrong.

A.5.3 Compiling

If configuration was successful, the software can be built with the command

```
> make
```

This will compile all relevant source code, but may take several minutes. The resulting object library archive will be located in the build subdirectory. It will be named similar to the package following usual Unix conventions; for example, the ropp_utils library is named libropp_utils.a. Sample applications and test programs or scripts will also have been built in the relevant subdirectories. Sample and test runs can be performed without installing the software; for details on available test programs, see A.7.

Currently supported Fortran compilers include (on Linux unless otherwise stated): Intel's ifort (v12), v16); NAG's nagfor (v5.2); Portland Group's pgf95 (v15); GNU gfortran (v4.4.7); SUN's sunf95 (v8 with SunStudio 12). For the authoritative list please refer to the ROPP Release Notes and README files in each sub-package.

A.5.4 Installing

After building the software successfully, the command

```
> make install
```

will install libraries in {prefix}/lib, Fortran modules in {prefix}/include, and any application programs in {prefix}/bin. Here, {prefix} is the prefix directory given as argument to the --prefix option of the configure command. By default, this is $ROPP_ROOT. If no --prefix is given, the installation root directory defaults to /usr/local which would normally require root (sudo) privileges.

A.5.5 Cleaning up

The temporary files created during the compilation of any ROPP package can be removed from the package directory tree with

```
> make clean
```

Note that this will keep the information gathered during configuration as well as the build libraries and executables intact. Thus, a new build can be attempted using make without the need for another configure. To remove all data related to the build and install process, run

```
> make distclean
```

which will restore the original state of the unpacked package, but with all potential user modifications to the source code still in place.

If the software has been installed previously, but shall be removed from the user's computer, this can be accomplished with the command

```
> make uninstall
```

performed in the source code distribution directory. Note that this requires a configuration which is identical to the one used for the original installation of the software. It is not necessary to rebuild the software again before uninstalling it.

A.6 Linking

If one (or more) ROPP packages have been installed successfully, linking your application's code against the ROPP libraries requires the specification of all ROPP and all external libraries. For example, to create an executable from your own application.f90 and the ropp_io libraries, something like

```
> ifort -o application application.f90 -L/usr/local/lib -L$ROPP_ROOT/lib  \
        -L$ROPP_ROOT/lib64 -lropp_io -lropp_utils -lnetcdf (-lnetcdff)
```

will be required. (Since netCDF-4.1.1, the netCDF C and Fortran routines have been split, with the latter held in libnetcdff.a. Hence, if compiling Fortran routines against a recent version of netCDF, -lnetcdff must be included in the list of libraries to be linked. Note that the netCDF libraries recommended for use with ROPP are now split in this way.)

A.7 Testing

The ROPP software has undergone formal testing before distribution, as will all future modifications and improvements. A subset of the test procedures and some reference files are provided with the source code in order to facilitate quick tests whether the compilation was completed successfully. Users can run these tests to ensure that there are no major problems. It should be kept in mind, though, that not all of the functionality of the corresponding package is fully tested. Note also that several of the test scripts attempt to run IDL to generate output which can be compared against existing reference plots. Generally the user would only do this if one of the tests failed. If IDL is unavailable the tests will bypass this step.

A.7.1 `ropp_utils`

Tested as part of the other modules, mainly with `ropp_io`.

A.7.2 `ropp_io`

The subdirectory `tests` of the `ropp_io` distribution contains several test programs and scripts to test various aspects of the software. A test is provided to check the user's installation of the `netCDF` library. They can be run after a successful compilation of the `ropp_io` package with

```
> make test_netcdf
```

from within the `tests` subdirectory. The program executed for this test does not use `ropp_io`, but is exclusively based on the native Fortran 90 interfaces for `netCDF`. Failure of this test strongly indicates that there is a problem with the installation or setup of the external library, which needs to be fixed before `ropp_io` can be used.

A second test can be run with

```
> make test_ropp
```

which runs a script performing several conversions between ROPP data files. Running this test through `make` has the advantage that the results of the conversions are interpreted properly and result in 'success' or 'failure' messages.

If a supported BUFR library is available, the `tests` subdirectory will also contain a test script for the two programs `ropp2bufr` and `bufr2ropp` which convert ROPP data files to and and from BUFR format data files. Issuing the command

```
> make test_bufr
```

will run a number of conversions and provide some verbose information on the content of the BUFR files and the encoding and decoding process. The script finally also compares the results. Its output should be self-explanatory. Note that due to limitations of the BUFR format, non-significant loss of precision may be detected and flagged as differences from the reference file; this is normal.

The gfz2ropp and ucar2ropp tools to convert GFZ native text files or UCAR netCDF files to ropp-standard netcdf are tested with the commands

```
> make test_gfz
> make test_ucar
```

The grib2bgrasc and bgrasc2ropp tools, which extract background profiles from GRIB-format gridded data and convert to ascii format, and then convert this to a ROPP-format netCDF file, are respectively tested with the commands

```
> make test_grib
> make test_bgrasc
```

The eum2ropp and eum2bufr tools to convert 'EUMETSAT-format' RO data into standard ROPP netCDF or BUFR files, are tested with the commands

```
> make test_eum
> make test_eumbufr
```

Finally, the command

```
> make test
```

will run all of the above described tests.

The test of the ropp_io library and tools can also be tested manually by running, for example,

```
> t_ropp2ropp -t -n
```

which will create a series of different files. These should be compared (e.g., using diff) according to the advice given through the program's execution. Users can safely ignore numerical differences in the order of the cutoff in the text representation of the ROPP data files. Also note that different file names will show up in the first line of the text representation of netCDF data files (files created by the test script with the extension .cdl) and can be ignored. The test_ropp target actually does the same, but interprets the differences between the files with the above issues in mind. Note that the output of t_ropp2ropp can be found in the file t_ropp2ropp.log when run through make.

A.7.3 ropp_pp

The subdirectory tests of the ropp_pp distribution contains testing software, to compare the geometric optic and wave optic processing with known output, check the consistency of the Abel integral routines and their inverses, and compare the ionospheric correction processing with known output. It also tests a low resolution of the wave optics propagator code, which resides in the ropp_pp module. Run

```
> make test
```

to check if solutions agree with precalculated solutions to within expected small tolerances. If IDL is available on the user's machine, plots of the results are made and can be compared against reference plots. A table summarising the results of the tests is written to stdout after they have all run.

A.7.4 `ropp_apps`

The subdirectory `tests` of the `ropp_apps` distribution contains testing software, to calculate tropopause height, and planetary boundary layer height, from a variety of profile data: bending angles, refractivities, background temperatures etc. Run

```
> make test
```

to check if solutions agree with precalculated solutions to within expected small tolerances. A table summarising the results of the tests is written to stdout after they have all run.

A.7.5 `ropp_fm`

The subdirectory `tests` of the `ropp_fm` distribution contains testing software. Run

```
> make test
```

to check if everything is working correctly. A series of tests are run to run the 1D and 2D operator applications to generate simulated refractivity and bending angle profiles, which are compared with precalculated data. Also included are tests of the consistency of the 1D and 2D tangent linear and adjoint routines. Warning messages are written to `stdout` if the operator, tangent linear and adjoint routines do not meet the expected (demanding) consistency checks. If IDL is available on the user's machine, plots of the results are made and can be compared against reference plots. A table summarising the results of the tests is written to stdout after they have all run.

A.7.6 `ropp_1dvar`

A simple test is provided to check the correct running of the 1D–Var stand-alone application. This inputs a file of 'observations' (refractivity profiles) simulated from a set of ECMWF model background profiles. The same backgrounds are used in the 1D–Var retrieval. Hence the expected retrieved output profiles should be identical to the background (within rounding errors).

Further tests are run of retrievals based on COSMIC observations (refractivities and bending angles) and co-located Met Office background profiles, and of retrievals based on GRAS observations (refractivities and bending angles) and co-located ECMWF background profiles. A simple test of a retrieval using L1 and L2 bending angles is also included.

The subdirectory `tests` of the `ropp_1dvar` distribution contains the testing software. Run

```
> make test
```

to check if everything is working correctly. The results of each test are numerically compared to reference results, and a PASS/FAIL message issued to stdout if the differences are smaller/greater than some small tolerance. If IDL is available on the user's machine, plots of the results are made and can be compared against reference plots. A table summarising the results of the tests is written to stdout after they have all run.

A.8 Troubleshooting

If something goes wrong during the configuration step, carefully check the full output of the last unsuccessful `configure` run to get an idea why the software could not be built; this can be found in the file `config.log`. This also applies if parts of ROPP are not built (e.g. the `BUFR` tools), even though the required additional libraries are available.

During compilation, warnings that indicate unused variables (e.g. with the NAG compiler) or the potential trimming of character variables (with Intel compilers) can safely be ignored. If the compilation is successful, but installation fails, make sure you have write permissions on the installation directories.

If linking against ROPP libraries fails because of unresolved externals, make sure that *all* relevant libraries – *including all external ones* – are specified in the correct order (some linkers are not able to recursively browse through several libraries in order to resolve externals) with lower-level libraries following higher-level (ROPP) ones.

If the BUFR encoding or decoding fail with messages about missing run-time BUFR tables, check that the appropriate environment variable `BUFR_LIBRARY` (for the MetDB library) or `BUFR_TABLES` (for the ECMWF library) have been correctly set to the path of the installed BUFR tables, and that the path ends with a '/' character.

Forward modelling of, and retrievals using, L1 and L2 bending angles impose heavier memory requirements than the more standard use of neutral bending angles. Users should therefore be prepared to increase the local memory available on their machines if using this feature.

If an ROPP module compiles and runs satisfactorily, but produces unexpected results, an easy first step in tracking down the problem is to print out extra diagnostic information. Most of the ROPP tools provide the facility to do this by means of the '`-d`' option. `ropp_pp`, `ropp_1dvar`, `ropp_apps` and `ropp_fm` also allow the user to add sets of pre-defined variables to the `ROprof` structure, which are written out in `netCDF` format with the usual variables. The first two modules do this by means of an option in a configuration file; the last two by means of a command line option in (some of) the tools. In fact, all ROPP modules allow the user to add specified variables to the `ROprof` structure in this way, by calling `ropp_io_addvar`, as described in the ROPP I/O user Guide. This obviously requires the code to be recompiled.

B ropp_io program files

```
arrays/
    arrays.f90
    blend.f90
    callocate.f90
    copy_alloc.f90
    cross_product.f90
    getrun.f90
    imaxloc.f90
    iminloc.f90
    isinrange.f90
    l2norm.f90
    locate.f90
    nruns.f90
    outer_and.f90
    outer_product.f90
    preallocate.f90
    reallocate.f90
    reverse.f90
    setminus.f90
    sort.f90
    swap.f90
    uniq.f90
    unit_vector.f90
    where.f90

common/

    FileDelete.f90
    GetIOUnit.f90
    ToLower.f90
    ToUpper.f90
    ropp_utils.f90
    ropp_utils_version.f90
    typeSizes.f90

compilers/

    nag_interfaces.f90
    xlf_interfaces.f90

coordinates/

    cart2geod.f90
    coordinates.f90
    curvature.f90
    earth.f90
    ecf2eci_nosofa.f90
    ecf2eci_sofa.f90
    eci2ecf_nosofa.f90
    eci2ecf_sofa.f90
    eci2eci_nosofa.f90
    eci2eci_sofa.f90
    geod2cart.f90
    impact_parameter.f90
    occ_point.f90
    plane_coordinates.f90
    tangent_point.f90
    vectors.f90

datetime/

    GpsSecond.f90
    augmentutc.f90
    caltojul.f90
    convertdt.f90
    date_and_time_utc.f90
    datetimeoffset.f90
    datetimeprogs.f90
    datetimetypes.f90
    gpstoutc.f90
    leapseconds.f90
    monthofyear.f90
    timesince.f90

geodesy/

    geodesy.f90
    geometric2geopotential.f90
    geopotential2geometric.f90
    gravity.f90
    great_circle_distance.f90
    r_eff.f90

messages/

    assert.f90
    message.f90
    message_bs_escape.f90
    message_get_addinfo.f90
    message_get_program.f90
    message_get_routine.f90
    message_set_addinfo.f90
    message_set_program.f90
    message_set_routine.f90
    messages.f90

tests/

    t_version.f90

unitconvert/

    ropp_unit_conversion.f90
    unitconvert.f90
    ut_convert.f90
```

C ROPP extra diagnostic data

For reference and for completeness, the listings of the all ROPP modules' extra variables are listed below.

C.1 ropp_io_addvar

The general form of the extra data, appended to the `RO_prof` structure by `ropp_io_addvar`, is described in Table C.1.

ROprof (Additional variables requested by call to `ropp_io_addvar`, throughout ROPP)	
Structure element	Description
`...%vlist%VlistD0d%name`	Name of 1st 0D extra variable
`...%vlist%VlistD0d%long_name`	Long name of 1st 0D extra variable
`...%vlist%VlistD0d%units`	Units of 1st 0D extra variable
`...%vlist%VlistD0d%range`	Range of 1st 0D extra variable
`...%vlist%VlistD0d%DATA`	Value of 1st 0D extra variable
`...%vlist%VlistD0d%next%name (etc)`	Name (etc) of 2nd 0D extra variable
`...%vlist%VlistD0d%next%next%name (etc)`	Name (etc) of 3rd 0D extra variable
`...%vlist%VlistD1d%name`	Name of 1st 1D extra variable
`...%vlist%VlistD1d%long_name`	Long name of 1st 1D extra variable
`...%vlist%VlistD1d%units`	Units of 1st 1D extra variable
`...%vlist%VlistD1d%range`	Range of 1st 1D extra variable
`...%vlist%VlistD1d%DATA`	Value of 1st 1D extra variable
`...%vlist%VlistD1d%next%name (etc)`	Name (etc) of 2nd 1D extra variable
`...%vlist%VlistD1d%next%next%name (etc)`	Name (etc) of 3rd 1D extra variable
`...%vlist%VlistD2d%name`	Name of 1st 2D extra variable
`...%vlist%VlistD2d%long_name`	Long name of 1st 2D extra variable
`...%vlist%VlistD2d%units`	Units of 1st 2D extra variable
`...%vlist%VlistD2d%range`	Range of 1st 2D extra variable
`...%vlist%VlistD2d%DATA`	Value of 1st 2D extra variable
`...%vlist%VlistD2d%next%name (etc)`	Name (etc) of 2nd 2D extra variable
`...%vlist%VlistD2d%next%next%name (etc)`	Name (etc) of 3rd 2D extra variable

Table C.1: Additional elements of `ROprof` structure, available throughout ROPP

C.2 PPDiag

The extra data which are output to the netCDF file if `config%output_diag` is set to `.TRUE.` in `ropp_pp`, are described in Table C.2.

PPDiag (`config%output_diag` = `TRUE` in `ropp_pp`)	
Structure element	Description
`...%CTimpact`	CT processing impact parameter (m)
`...%CTamplitude`	CT processing amplitude
`...%CTamplitude_smt`	CT processing smoothed amplitude
`...%CTimpactL2`	CT processing L2 impact parameter (m)
`...%CTamplitudeL2`	CT processing L2 amplitude
`...%CTamplitudeL2_smt`	CT processing smoothed L2 amplitude
`...%ba_ion`	Ionospheric bending angle in L1 (rad)
`...%err_neut`	Error covariance of neutral bending angle (rad^2)
`...%err_ion`	Error covariance of ionospheric bending angle (rad^2)
`...%wt_data`	Weight of data (data:data+clim) in profile
`...%sq`	SO badness score: `MAX[err_neut`$^{1/2}/\alpha_N$`]`$\times 100\%$
`...%L2_badness`	L2 phase correction badness score
`...%L2_min_SLTA`	Lowest valid L2 SLTA (m)

Table C.2: Elements of `PPDiag` structure, available from `ropp_pp`

C.3 ropp_fm_bg2ro

The extra data which are appended to the ROprof structure if the `ropp_fm` tool `ropp_fm_bg2ro_1d` is called without the '`-f`' option, are described in Table C.3.

ROprof (Absence of '`-f`' option in call to `ropp_fm_bg2ro_1d`, in `ropp_fm`)	
Structure element	Description
`...%gradient_refrac`	$\partial N_i / \partial x_j$ matrix
`...%gradient_bangle`	$\partial \alpha_i / \partial x_j$ matrix

Table C.3: Additional elements of `ROprof` structure, available from `ropp_fm`. See Table C.1 for the detailed structure.

C.4 VarDiag

The extra data which are output to the netCDF file if `config%extended_1dvar_diag` is set to `.TRUE.` in `ropp_1dvar`, are described in Table C.4.

VarDiag (config%extended_1dvar_diag = TRUE in ropp_1dvar)	
Structure element	Description
...%n_data	Number of observation data
...%n_bgqc_reject	Number of data rejected by background QC
...%n_pge_reject	Number of data rejected by PGE QC
...%bg_bangle	Background bending angle
...%bg_refrac	Background refractivity
...%OmB	Observation minus background
...%OmB_sigma	OmB standard deviation
...%pge_gamma	PGE check gamma value
...%pge	Probability of Gross Error along profile
...%pge_weights	PGE weighting values
...%ok	Overall quality flag
...%J	Cost function value at convergence
...%J_scaled	Scaled cost function value ($2J/m$)
...%J_init	Initial cost function value
...%J_bgr	Background cost function profile
...%J_obs	Observation cost function profile
...%B_sigma	Forward modelled bg standard deviation
...%n_iter	Number of iterations to reach convergence
...%n_simul	Number of simulations
...%min_mode	Minimiser exit mode
...%res_bangle	Analysis bending angle
...%res_refrac	Analysis refractivity
...%OmA	Observation minus analysis
...%OmA_sigma	OmA standard deviation

Table C.4: Elements of VarDiag structure, available from ropp_1dvar

D ROPP user documentation

Title	**Reference**	**Description**
ROPP User Licence	SAF/ROM/METO/LIC/ROPP/002	Legal conditions on the use of ROPP software
ROPP Overview	SAF/ROM/METO/UG/ROPP/001	Overview of ROPP and package content and functionality
ROPP_IO User Guide	SAF/ROM/METO/UG/ROPP/002	Description of `ropp_io` module content and functionality
ROPP_PP User Guide.	SAF/ROM/METO/UG/ROPP/004	Description of `ropp_pp` module content and functionality
ROPP_APPS User Guide.	SAF/ROM/METO/UG/ROPP/005	Description of `ropp_apps` module content and functionality
ROPP_FM User Guide.	SAF/ROM/METO/UG/ROPP/006	Description of `ropp_fm` module content and functionality
ROPP_1DVAR User Guide.	SAF/ROM/METO/UG/ROPP/007	Description of `ropp_1dvar` module content and functionality
ROPP UTILS Reference Manual	SAF/ROM/METO/RM/ROPP/001	Reference manual for the `ropp_utils` module
ROPP IO Reference Manual	SAF/ROM/METO/RM/ROPP/002	Reference manual for the `ropp_io` module
ROPP FM Reference Manual	SAF/ROM/METO/RM/ROPP/003	Reference manual for the `ropp_fm` module
ROPP 1D–Var Reference Manual	SAF/ROM/METO/RM/ROPP/004	Reference manual for the `ropp_1dvar` module
ROPP PP Reference Manual	SAF/ROM/METO/RM/ROPP/005	Reference manual for the `ropp_pp` module
ROPP APPS Reference Manual	SAF/ROM/METO/RM/ROPP/006	Reference manual for the `ropp_apps` module
WMO FM94 (BUFR) Specification for Radio Occultation Data	SAF/ROM/METO/FMT/BUFR/001	Description of BUFR template for RO data

Table D.1: ROPP user documentation

Title	Reference	Description
Mono-dimensional thinning for GPS Radio Occultations	SAF/GRAS/METO/REP/GSR/001	Technical report on profile thinning algorithm implemented in ROPP
Geodesy calculations in ROPP	SAF/GRAS/METO/REP/GSR/002	Summary of geodetic calculations to relate geometric and geopotential height scales
ROPP minimiser - minROPP	SAF/GRAS/METO/REP/GSR/003	Description of ROPP-specific minimiser, minROPP
Error function calculation in ROPP	SAF/GRAS/METO/REP/GSR/004	Discussion of impact of approximating erf in ROPP
Refractivity calculations in ROPP	SAF/GRAS/METO/REP/GSR/005	Summary of expressions for calculating refractivity profiles
Levenberg-Marquardt minimisation in ROPP	SAF/GRAS/METO/REP/GSR/006	Comparison of Levenberg-Marquardt and minROPP minimisers
Abel integral calculations in ROPP	SAF/GRAS/METO/REP/GSR/007	Comparison of ‘Gorbunov’ and ‘ROM SAF’ Abel transform algorithms
ROPP thinner algorithm	SAF/GRAS/METO/REP/GSR/008	Detailed review of the ROPP thinner algorithm
Refractivity coefficients used in the assimilation of GPS radio occultation measurements	SAF/GRAS/METO/REP/GSR/009	Investigation of sensitivity of ECMWF analyses to empirical refractivity coefficients and non-ideal gas effects
Latitudinal Binning and Area-Weighted Averaging of Irregularly Distributed RO Data	SAF/GRAS/METO/REP/GSR/010	Discussion of alternative spatial averaging method for RO climate data
ROPP 1D–Var validation	SAF/GRAS/METO/REP/GSR/011	Illustration of ROPP 1D–Var functionality and output diagnostics
Assimilation of GPSRO Data in the ECMWF ERA-Interim Re-analysis	SAF/GRAS/METO/REP/GSR/012	Assimilation of GPSRO Data in the ECMWF ERA-Interim Re-analysis
ROPP_PP validation	SAF/GRAS/METO/REP/GSR/013	Illustration of ROPP_PP functionality and output diagnostics

Table D.2: GRAS SAF Reports

Title	Reference	Description
A review of the geodesy calculations in ROPP	SAF/ROM/METO/REP/RSR/014	Comparison of various potential geodesy calculations
Improvements to the ROPP refractivity and bending angle operators	SAF/ROM/METO/REP/RSR/015	Improved interpolation in ROPP forward models
Simplifying EGM96 undulation calculations in ROPP	SAF/ROM/METO/REP/RSR/016	Simplifying ROPP undulation calculations
Simulation of L1 and L2 bending angles with a model ionosphere	SAF/ROM/METO/REP/RSR/017	Simulating L1 and L2 bending angles in ROPP
Single Frequency Radio Occultation Retrievals: Impact on Numerical Weather Prediction	SAF/ROM/METO/REP/RSR/018	Potential impact of loss of L2 bending angle on NWP
Implementation of the ROPP two-dimensional bending angle observation operator in an NWP system	SAF/ROM/METO/REP/RSR/019	Implementation of ROPP 2D forward model at ECMWF
Interpolation artefact in ECMWF monthly standard deviation plots	SAF/ROM/METO/REP/RSR/020	Investigation into plot anomaly
5th ROM SAF User Workshop on Applications of GPS radio occultation measurements	SAF/ROM/METO/REP/RSR/021	Report on 5th ROM SAF User Workshop
The use of the GPS radio occultation reflection flag for NWP applications	SAF/ROM/METO/REP/RSR/022	Impact of reflected occultations at ECMWF
Assessment of a potential reflection flag product	SAF/ROM/METO/REP/RSR/023	Assessment of flagged COSMIC occultations
The calculation of planetary boundary layer heights in ROPP	SAF/ROM/METO/REP/RSR/024	Description of ROPP PBLH diagnostics
Survey on user requirements for potential ionospheric products from EPS-SG radio occultation measurements	SAF/ROM/METO/REP/RSR/025	Results of a ROM SAF survey of the interest in possible EPS-SG ionospheric products
Estimates of GNSS radio occultation bending angle and refractivity error statistics	SAF/ROM/METO/REP/RSR/026	RO error statistics as derived by forward modelling ECMWF model errors
Recent forecast impact experiments with GPS radio occultation measurements	SAF/ROM/METO/REP/RSR/027	Impacts in NWP of 2014–2015 RO data
Description of wave optics modelling in ROPP-9 and suggested improvements for ROPP-9.1	SAF/ROM/METO/REP/RSR/028	Wave optics propagator in ROPP-9.0 and 9.1

Table D.3: ROM SAF Reports

Title	Reference	Description
Testing reprocessed GPS radio occultation datasets in a reanalysis system	SAF/ROM/METO/REP/RSR/029	Impact of reprocessed RO data on reanalyses
A first look at the feasibility of assimilating single and dual frequency bending angles	SAF/ROM/METO/REP/RSR/030	Single and dual frequency assimilation
Sensitivity of some RO measurements to the shape of the ionospheric electron density profile	SAF/ROM/METO/REP/RSR/031	Ionospheric shape sensitivity
An initial assessment of the quality of RO data from KOMPSAT-5	SAF/ROM/METO/REP/RSR/032	KOMPSAT-5 quality assessment
Some science changes in ROPP-9.1	SAF/ROM/METO/REP/RSR/033	ROPP-9.1 science
An initial assessment of the quality of RO data from Metop-C	SAF/ROM/METO/REP/RSR/034	Metop-C quality assessment
An initial assessment of the quality of RO data from FY-3D	SAF/ROM/METO/REP/RSR/035	FY-3D quality assessment
6th ROM SAF User Workshop	SAF/ROM/METO/REP/RSR/037	ROM SAF–IROWG 2019 report

Table D.4: ROM SAF Reports (continued)

Title	Reference	Description
CDOP-3 Proposal	SAF/ROM/DMI/MGT/CDOP3/001	Proposal for the Third Continuous Development and Operations Phase (CDOP-3) March 2017 – February 2022
Co-operation Agreement	EUM/C/85/16/DOC/19	C/A between EUMETSAT and DMI, Lead Entity for the CDOP-3 of the ROM SAF, signed at the 86th Council meeting on 7th December 2016
Product Requirements Document (PRD)	SAF/ROM/DMI/MGT/PRD/001	Detailed specification of the products of the ROM SAF
System Requirements Document (SRD)	SAF/ROM/DMI/RQ/SRD/001	Detailed specification of the system and software requirements of the ROM SAF

Table D.5: Applicable documents

E Authors

Many people, inside and outside the ROM SAF, have contributed to the development of ROPP. The principal authors are listed in Table E.1. The ROM SAF extends its sincere gratitude for their efforts.

ROPP Authors

Name	Current institute	Contribution
Christian Marquardt	EUMETSAT	Author of majority of ROPP-1 code in UTILS, IO, FM and 1DVAR modules, and much personal, pre-existing software.
Huw Lewis	Met Office	1st ROPP Development Manager, FM and 1D–VAR extensions. PP module.
Dave Offiler	Met Office	ROPP Project Manager, IO application code and IO extensions, BUFR format/template.
Sean Healy	ECMWF	Original 1D FM code, 2D FM operator code, introduction of compressibility factors, improved FM vertical interpolation scheme, forward modelling of L1 and L2 bending angles, 1D and 2D wave optics propagators.
Michael Gorbunov	Russian Academy of Sciences	Original pre-processor code.
Axel von Engeln	EUMETSAT	Author of original Test Folder system and of EUMETSAT-formatted RO data reader.
Stig Syndergaard	DMI	Original spectral version of MSIS model (expansion in spherical harmonics and Chebychev polynomials), PP module developments.
Ian Culverwell	Met Office	2nd ROPP Development Manager. Documentation, testing, consolidation, IO development, GRIB2 reader, implementation of tropopause height diagnostics and planetary boundary layer height diagnostics, forward modelling of L1 and L2 bending angles.
Chris Burrows	ECMWF	2nd ROPP Test Manager. Test folder developments, improved FM vertical interpolation scheme.
Francis Warrick	Met Office	Implementation of ecCodes library; ROPP development and testing.
Michael Rennie	ECMWF	1st ROPP Test Manager. Test folder developments.
Kjartan Kinch	DMI	Elements of ropp_pp.
Hans Gleisner	DMI	Elements of ropp_pp, prototype GRIB2 reader, ec{i/f}2ec{i/f} code.
Carlo Buontempo	Met Office	Savitzky-Golay thinner code.
Torsten Schmidt	GFZ	Guidance on tropopause height diagnostics.
Feiqin Xie	Texas A & M	Suggested boundary layer height diagnostic algorithms.
Barbara Scherllin-Pirscher	Wegener Center	BAROCLIM (3) dataset for statistical optimisation.
Helge Jønch-Sørensen	DMI	BAROCLIM code.
Kent Bækgaard Lauritsen	DMI	Code reviews; liaison with EUMETSAT (licences, beta tester contracts).

Table E.1: Contributors to ROPP

F Copyrights

The majority of ROPP code is

Some parts of the source code within this distribution were developed within the Met Office outside the context of the ROM SAF and represents pre-existing software (PES); this portion is

This ROPP package also contains open source code libraries available through its author, Christian Marquardt. This is also PES, and is

This ROPP package may also contain open source code libraries available through its author, Michael Gorbunov. This is also PES, and is

This ROPP package may also contain open source code libraries available through its author, Stig Syndergaard. This is also PES, and is

This ROPP package may also contain a dataset available through its author, Barbara Scherllin-Pirscher, and is