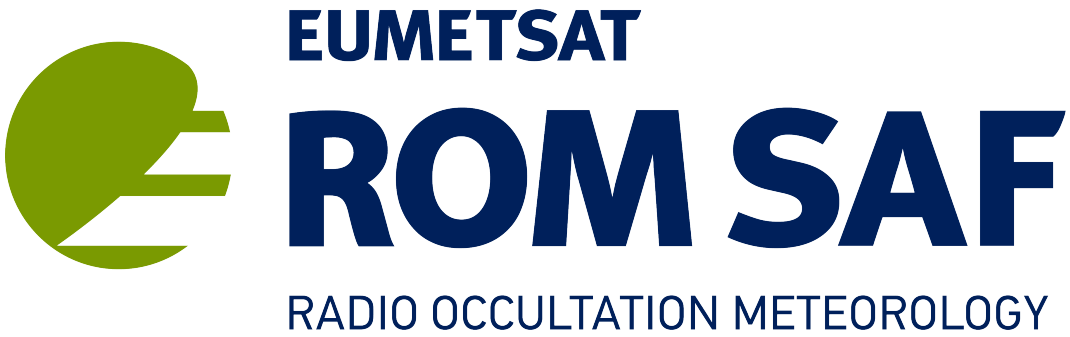

The Radio Occultation Processing Package (ROPP)
Input/Output Module User Guide

Version 11.3

2 May 2024

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)
University of Graz, Wegener Center (UG-WEGC)

Ref: SAF/ROM/METO/UG/ROPP/002

DOCUMENT AUTHOR TABLE

	Author(s)	Function	Date
Prepared by:	M. Schwärz	ROPP Release Manager	2024-05-02
Reviewed by:	O. Lewis	ROM SAF Project Team	2021-12-31
Approved by:	K. B. Lauritsen	ROM SAF Project Manager	2024-05-02

DOCUMENT CHANGE RECORD

Version	Date	By	Description
0.1	01 Nov 2004	CM	Preliminary release for I–RR
0.7	04 Jun 2005	CM	Intermediate release for CPM.
0.8	17 Oct 2005	DO	First beta release (v0.8)
0.9	10 Nov 2006	DO	Second beta release (v0.9)
1.0	01 Mar 2007	DO	First full release (v1.0)
1.1	01 Mar 2008	DO	Updates to first full release (v1.1)
1.2	01 Jul 2008	HL	Updates to first full release (v1.2)
2.0	01 Dec 2008	HL	Second full release (v2.0)
3.0	01 Jun 2009	HL	Third full release (v3.0)
4.0	01 Nov 2009	HL	Fourth full release (v4.0)
4.1	01 Jun 2010	HL	Updates to fourth full release (v4.1)
5.0	14 Jun 2011	IC	Fifth full release (v5.0)
6.0	31 Oct 2011	IC	Sixth full release (v6.0)
6.1	31 Jan 2013	IC	Updates to sixth full release (v6.1)
7.0	31 Jul 2013	IC	Seventh full release (v7.0)
7.1	31 Dec 2013	IC	Updates to seventh full release (v7.1)
8.0	31 Dec 2014	IC	Eighth full release (v8.0)
8.1	31 Dec 2015	IC	Update to eighth full release (v8.1)
9.0	28 Feb 2017	IC	Ninth full release (v9.0)
9.1	30 Jun 2019	IC	Update to ninth full release (v9.1)
10.0	30 Sep 2020	IC	Tenth full release (v10.0)
11.0	31 Dec 2021	IC	Eleventh full release (v11.0).
11.1	28 Feb 2022	IC	Update to eleventh full release (v11.1). Changes from ROPP-11.0 in red — see Change Log (SAF/ROM/METO/SRN/ROPP/019) for details.

continued on next page

continued

Version	Date	By	Description
11.3	2 May 2024	MS	Update version 11.3 to 11th full release.

ROM SAF

The Radio Occultation Meteorology Satellite Application Facility (ROM SAF) is a decentralised processing centre under EUMETSAT which is responsible for operational processing of radio occultation (RO) data from the Metop, Metop-SG and Sentinel-6 satellites and radio occultation data from other missions. The ROM SAF delivers bending angle, refractivity, temperature, pressure, humidity, and other geophysical variables in near real-time for NWP users, as well as reprocessed Climate Data Records (CDRs) and Interim Climate Data Records (ICDRs) for users requiring a higher degree of homogeneity of the RO data sets. The CDRs and ICDRs are further processed into globally gridded monthly-mean data for use in climate monitoring and climate science applications.

The ROM SAF also maintains the Radio Occultation Processing Package (ROPP) which contains software modules that aid users wishing to process, quality-control and assimilate radio occultation data from any radio occultation mission into NWP and other models.

The ROM SAF Leading Entity is the Danish Meteorological Institute (DMI), with Cooperating Entities: i) European Centre for Medium-Range Weather Forecasts (ECMWF) in Reading, United Kingdom, ii) Institut D'Estudis Espacials de Catalunya (IEEC) in Barcelona, Spain, iii) Met Office in Exeter, United Kingdom, and iv) and Wegener Center, University of Graz, in Graz, Austria. To get access to our products or to read more about the ROM SAF please go to: https://rom-saf.eumetsat.int.

Intellectual Property Rights

Contents

1 Introduction

1.1 Purpose of this document

This document ([RD.2b]) provides a User Guide for the Input/Output module of the Radio Occultation Processing Package (ROPP). A generic ROPP data format and software to read and write radio occultation data are provided as part of ROPP. These are described in this document.

For the current version of ROPP please refer to the Release Notes document ([RD.3]).

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] ROM SAF CDOP-4 Product Requirements Document, Ref: SAF/ROM/DMI/MGT/PRD/004

[AD.2] CDOP 4 Proposal: Proposal for the Fourth Continuous Development and Operations Phase (CDOP 4), Ref: SAF/ROM/DMI/MGT/CDOP4/001 (v1.1, 5 April 2021, as approved by the EUMETSAT Council in document reference EUM/C/97/21/DOC/15)

[AD.3] CDOP 4 Cooperation Agreement between EUMETSAT and DMI on the CDOP 4 of the ROM SAF, Ref: EUM/C/97/21/DOC/21, signed on 31 August and 15 September 2021

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

[RD.3] ROPP Release Notes. SAF/ROM/METO/SRN/ROPP/001

[RD.4] WMO FM94 (BUFR) specification for radio occultation data. SAF/ROM/METO/FMT/BUFR/001

[RD.5] Unidata netCDF website: http://www.unidata.ucar.edu/software/netcdf/

[RD.6] HDF Group website: http://www.hdfgroup.org/HDF5/

[RD.7] Development procedures for software deliverables. NWPSAF-MO-SW-002

[RD.8] ECMWF BUFR software website: https://software.ecmwf.int/wiki/display/BUFR

[RD.9] ECMWF GRIB_API software website: https://software.ecmwf.int/wiki/display/GRIB

[RD.10] ZLIB website http://www.zlib.net

[RD.11] EUMETSAT Radio Occultation Level 1 Product Format Specification. EUM/TSS/SPE/16/817861

[RD.12] IAU Standards of Fundamental Astronomy (SOFA) Libraries product. http://www.iausofa.org/

1.3 Acronyms and Abbreviations

AC	Analysis Correction (NWP assimilation technique)
API	Application Programming Interface
ATBD	Algorithm Theoretical Baseline Document
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
CSDP	Climate Service and Development Plan (EUMETSAT)
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
GCOS Global Climate Observing System
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 and the future GALILEO)
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 interim 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)
NCO	Numerically Controlled Oscillator
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
PBLH	Planetary Boundary Layer Height
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. Members are DMI (leader), 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
TPH	Tropopause Height
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
UW5	5^{th} ROM SAF User Workshop
UW6	6^{th} ROM SAF User Workshop
UW7	7^{th} ROM SAF User Workshop
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
WG-DRG	Working Group on Data Record Generation (EUMETSAT)
WGS–84	World Geodetic System, 1984. (US DoD)
WMO	World Meteorological Organization
WWW	World Weather Watch (WMO)

1.4 Definitions

RO data products from current and upcoming EUMETSAT satellites and other missions are grouped in data levels (Level 0, 1, 2, or 3) and product types (NRT, Offline, NTC, 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 variables may be contained in a given data level in a given file.

Data Levels

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, scintillation parameters, and quality information.

Level 1B: Bending angles and impact parameters, tangent point location, total electron content, 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), electron densities (Level 2E), and 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: Data product delivered less than: (i) 3 hours after measurement (ROM SAF Level 2 products for EPS); (ii) 150 min after measurement (ROM SAF Level 2 products for EPS-SG Global Mission); (iii) 125 min after measurement (ROM SAF Level 2 products for EPS-SG Regional Mission).

[1] Note that the level definitions differ partly from the WMO definitions.

Offline, NTC: Data product delivered from about 5 days to up to 6 months after measurement, depending on the applicable 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 an apparently homogeneous data record appropriate for climate usage. The CDR may be referred to as Fundamental (e.g. for Level 1B bending angles) or as Thematic (if related to a specific application area).

ICDR: Interim Climate Data Record which regularly extends in time a 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 defines and describes the `ROprof` derived datatype, which is used throughout ROPP to hold RO data. This Section also gives some tips on how to exploit this data structure: adding data to it, freeing data, range checking, unit conversion, reading and writing data, etc. Section 3.6 describes the various data conversion routines within ROPP

Appendices give brief instructions on how to build ROPP, list the files in the `ropp_io` 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] https://climatemonitoring.info/home/terminology/
[3] 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. It was originally designed to process data from the GRAS instrument on Metop-A and B, but the software should be adaptable enough to handle data from any other GNSS-LEO radio occultation mission.

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 inter-relationships 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.

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, 2022f), input/output (ROM SAF, 2022d), pre-processor (ROM SAF, 2022e), forward model (ROM SAF, 2022c), 1D–Var (ROM SAF, 2022a) and

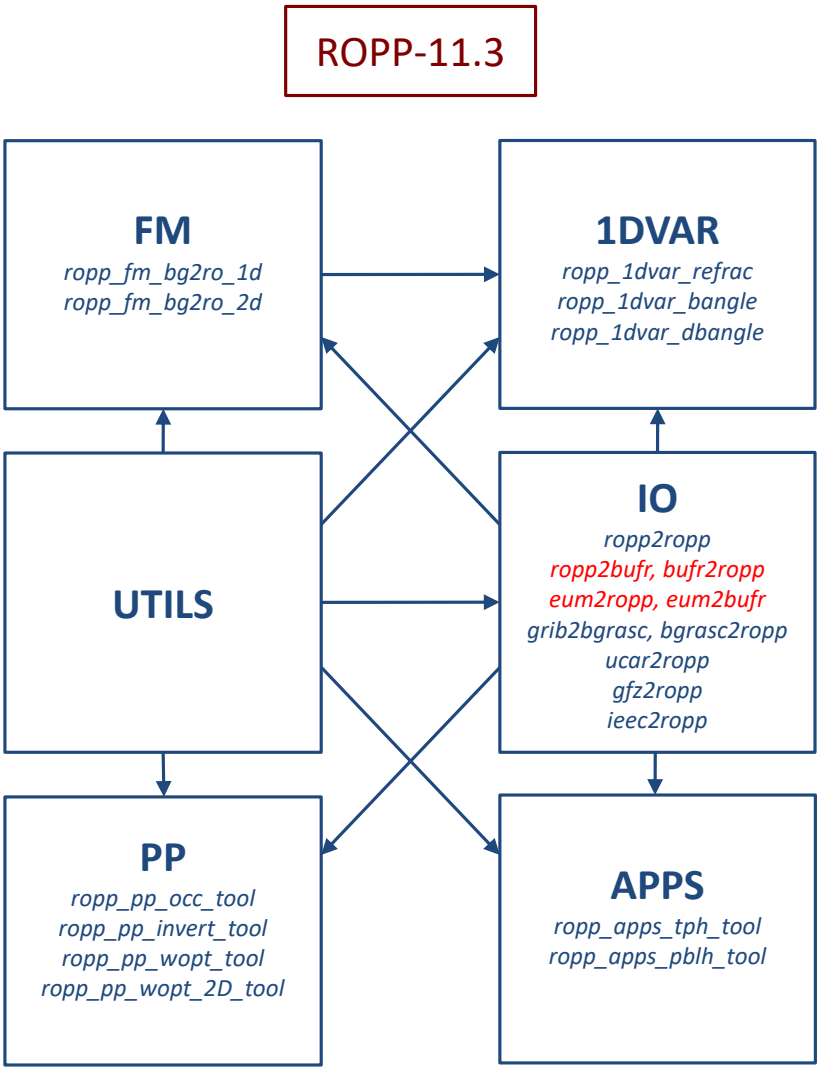

Figure 2.1: The **modules** and *tools* within ROPP-11.3. The module at the head of an arrow depends directly on the module at its tail.

applications (ROM SAF, 2022b) 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.

3 ROPP data format

The ROPP data format is a generic data format for radio occultation and related data beginning with level 1a products, i.e. amplitude and (excess) phase data obtained from GNSS radio occultation measurements. For example, an individual radio occultation measurement (or profile), when stored in ROPP format, may be thought of as consisting of the following parts[1]:

Header: Meta data like date, time and location of the occultation

Level 1: "raw" radio occultation data:

- Level 1a: SNR, (excess) phases and POD data as function of time
- Level 1b: bending angle as function of impact parameter

Level 2: "processed" geophysical data:

- Level 2a: refractivity on "observed" height levels
- Level 2b: temperature, humidity pressure and geopotential height on "model" levels
- Level 2c: surface pressure and surface geopotential height, tropopause and boundary layer heights, Chapman layer parameters
- Level 2d: additional data describing the vertical level structure (e.g., level coefficients for vertical hybrid coordinates)

Apart from the header, all other parts are optional. Thus, a ROPP file may contain only the Level 1a part, i.e. signal–to–noise ratios, excess phases and orbit data; another file (consisting of the header and Levels 1a, 1b and 2a) might contain the raw data, but also bending angle, refractivity and dry temperature profiles calculated from these. Finally, a refractivity-based 1D–Var retrieval, together with the refractivity it was calculated from, would be put into an ROPP data file which contains a header and the Level 2a and 2b parts.

In a similar way, auxiliary data, e.g. vertical profiles of temperature and humidity obtained from NWP analyses or short range forecasts, might be stored in an ROPP file. In this case, only Levels 2b and possibly 2c might be present. If such data serves as background for a variational retrieval as implemented in ROPP, Levels 2b, 2c and 2d (if the background data is given on some sort of model levels instead of fixed pressure or altitude levels) will be present. On the other hand, forward–simulated radio occultation "measurements", e.g. refractivity, bending angles or even forward–simulated amplitudes and phases for a given orbit configuration might also be stored as ROPP file by using the appropriate levels of the data model.

Technically, the ROPP data format is implemented using the netCDF scientific data format (Unidata, -), which is a platform–independent, self–describing format for scientific data. Variables (which may be scalars or arrays with several dimensions) are accessed by name; along with each variable, meta–data such as the data's physical units and valid numerical ranges can be stored as "attributes" of the data.

[1] The 'Level' characterisation used in the description of the ROPP data model is not related to the formal data levels of any specific radio occultation instrument.

The netCDF data model has no way to represent structures. Therefore, all structure elements from the ropp_io data types are mapped to a set of unique variable names in the netCDF data file. The netCDF variable names and their attributes (see below) can be inspected using the **ncdump(1)** utility that is built along with the netCDF interface library — for instance `ncdump -c file.nc`. The mapping from internal netCDF variable names and the ROPP structure names will be obvious as the last part of the structure name is used for the netCDF variable name. For instance, `ROdata%Lev1a%phase_L1` maps simply to `phase_L1`. The exception is the various `ROdata%Lev??%Npoints` structure variables which map to `dim_lev??` in the netCDF file.

All variables in a netCDF based ROPP data file have standard attributes according to the CF convention[2]. This includes the `long_name` and `valid_range` attributes giving a longer description of the variable's content and the physical validity range of the variable. The latter should be given in the same units as the variable itself. Physical units of each variable are described by the `units` attribute which facilitates automated unit conversions (Section 3.4.3). CF-compliant "standard names" are not yet included in ROPP data structures.

In order to facilitate the multifile option, all data files, including singlefiles, exhibit an unlimited(or record) netCDF dimension. Thus, scalar structure elements are mapped into 1–dimensional arrays in the netCDF data representation, and 1–dimensional arrays in the `ROprof` structure are technically 2–dimensional variables in the netCDF. In singlefiles, the unlimited dimension has only one element; in multifiles, the number of records is the number of profiles stored in the data file. This needs to be taken into account when reading ropp_io data with means other than the ropp_io interface. But as long as the ropp_io library is used for reading and writing, the actual format for the data files is not relevant for the user as the access is transparent.

The ropp_io module defines a dedicated structure (or 'derived type' in Fortran terminology) `ROprof` to hold data related to a vertical radio occultation profile. It is therefore mandatory to load the ropp_io module at the beginning of each program unit using ropp_io data types. A detailed definition of the elements of an `ROprof` structure is given in Section 3.3.

```
USE ropp_io
TYPE(ROprof) :: ro_data
```

For applications requiring the use of two-dimensional background data (e.g. ropp_fm 2D forward operators), a `ROprof2d` structure is defined. This is equivalent to the profile `ROprof` structure, but Level2b and Level2c (and, from ROPP10.0, Level2a) data may have an additional horizontal dimension to hold a plane of information rather than a profile.

```
USE ropp_io
TYPE(ROprof2d) :: ro_data
```

Note that all ropp_io functionality described below may be applied to either `ROprof` or `ROprof2d` input structures using the same interfaces. Examples in this User Guide are only shown using `ROprof` as input for illustration. Thinning of `ROprof2d` data is not currently provided however.

[2] See http://cf-pcmdi.llnl.gov.

3.1 Reading data

The subroutine `ropp_io_read` reads the contents of an ROPP data file, and puts its data into the structure `ro_data`. The name of the file to be read is given as the second (mandatory) argument:

```
CALL ropp_io_read(ro_data, filename)
```

On return, `ro_data` will contain all data of the profile contained in the data file in its relevant sublevels. Any previous contents of `ro_data` will be lost. The read ROPP data can then be accessed through the respective elements of the `ROprof` data structure.

If certain data levels were not contained in the data file, the returned structure will have zero elements, and the `...%Npoints` element of the respective level will be set to zero. This provides a mechanism to test if the data file contains data for any specific level. For example, if further processing depends on the existence of Level 1b (bending angle) data, an application might be written as

```
USE ropp_io
TYPE(ROprof) :: ro_data
   ...
CALL ropp_io_read(ro_data, filename)
IF (ro_data%Lev1b%Npoints > 0) THEN
   ! ... process the data ...
ELSE
   !      ... or issue an error message.
END IF
```

The read routine automatically recognises the actual technical file format the data has been written in, and calls the appropriate internal read routines to fill the data structure `ro_data`. If the data file represents a 'multifile' containing more than one radio occultation profile, the first profile in the data set will be read by default. There are additional options to the `ropp_io_read` routine which allow reading other than the first profile (see Section 3.4.8).

3.2 Writing data

Writing data to an ROPP file requires the following steps:

- *Declare a structure that holds the content of the ROPP data file*
- *Initialise the `ROprof` structure*

  All array elements of an `ROprof` structure are pointers; they need to be allocated before they can hold data. The subroutine `ropp_io_init` is provided for that purpose:

  ```
  CALL ropp_io_init(ro_data, n_lev_1a, n_lev_1b, &
                            n_lev_2a, n_lev_2b, n_lev_2c, n_lev_2d)
  ```

- *Populate the data structure*

  The elements of the data structure can now be filled by the user application with the data to be written; for details of the elements on the `ROprof` structure see Section 3.3.

- *Write the data*

  The contents of an `ROprof` data structure are finally written to a file with

  ```
  CALL ropp_io_write(ro_data, filename)
  ```

  The second parameter defining the name of the file to be written is optional; if it is not specified, the `ropp_io` library sets the file depending on the occultation id information given in the header. See Section 3.3.6 for details. A ROPP netCDF file is written. Independent of the way the file name is specified, the routine will overwrite any existing file of the same name (but note the `append` keyword for multifiles; see Section 3.4.8).

3.3 Radio occultation profiles

3.3.1 Header

The header contains meta data (static, non-profile) about the occultation, including information about the processing centre and applied algorithms, and general georeferencing data. Table 3.1 shows the elements of the `ROprof` header and their description, along with default units and ranges.

Note that the header contains information about the georeferencing, i.e. the profile's location and time; these may not be available if only Level 1a data are contained in a data file, as the georeferencing information is determined during the Level 1a to Level 1b processing. For the ROPP data model, it is assumed that the profile's header location is formally derived as the longitude and latitude of a single tangent point adopted at some point during the occultation. It is further assumed that the centre and radius of curvature are calculated for the same tangent point. The number `...%GEORef%time_offset` is the time when the georeferencing information was calculated, relative to the start of the occultation. For all applications that interpret a radio occultation measurement as a vertical profile the georeferencing information for the profile should be taken from the header. Note that the georeferencing method — and therefore the longitude and latitude coordinates for the profile — may differ between data providers.

The background meta data contained in the header is used to identify any *a priori* data used for the various processing steps. This may include information on any climatology used within a 'statistical optimisation' of ionospheric corrected bending angles prior to calculating refractivity, or a description of a short range Numerical Weather Prediction forecast used as first guess in a 1D–Var retrieval of geophysical parameters. In the latter case, the forecast lead time as well as the time when the forecast is assumed to be valid should be given. If climatology data is used, the verification time should be used to indicate the month at which the climatology is considered to be valid, with all other time information set to missing values.

Time stamps (given in UTC) denote the start times of the occultation (`...%GEORef%DTocc`) as well as the processing (`...%GEORef%DTpro`). The netCDF file that is produced when the `ROprof` structure is written out with `ropp_io_write_ncdf_put.f90` includes the variable `start_time`, which gives the time in clock seconds, correctly including leap seconds with effect from ROPP9.1, between the start of the occultation and 0Z 01/01/2000. The netCDF file also includes the variable `time`, which is the nominal georeferencing time of the occultation, again expressed in clock seconds since 0Z 01/01/2000. This means that `time = start_time + time_offset`. (Before ROPP9.1, `time` was erroneously set

equal to `start_time`.)

When reading ROPP format netCDF files there is a check on the consistency between `start_time` and the `year/month/day/hour/minute/second/msec` values in the netCDF file. If they differ by more than 30 sec, `year/month`/etc take precedence in the calculation of `...%GEORef%DTocc`, and a warning is issued. If they differ by more than 0.5 msec but less than 30 sec, and if extra diagnostic output has been requested (e.g. by means of the `'-d'` option), then an information message is issued, suggesting that the cause of the discrepancy may be the failure to account for leap seconds. (Prior to ROPP9.1, leap seconds were not included in the calculation of `start_time`, and files generated before this will therefore have a few seconds' difference between the two times.) The netCDF variables `year/month`/etc still take priority in the construction of `...%GEORef%DTocc`, because these numbers are more likely to be correct.

An overall quality parameter (see 3.3.4) and a bit field for Product Confidence Data (see 3.3.5) are also part of the header (and are also written to the netCDF file).

3.3.2 Level 1 profile data

Level 1 data of the ROPP data format consists of time and height dependent 'basic' RO parameters.

Level 1a: SNR, (excess) phases and POD data as function of time

Level 1b: bending angle as function of impact parameter

Level 1a contains the raw GNSS data after (possibly) any clock offsets and differencing is applied to the raw GNSS data as collected by the receiver. It is recommended that the samples shall be in increasing time order; all times are measured with respect to the beginning of the occultation as given in the header.

With respect to the orbit data contained in Level 1a, we recommend that all velocities are given in Earth Centred Inertial (ECI) coordinates in order to make sure that differential rotational effects have not been accidentally included by performing a transformation to the Earth Centred, Earth Fixed (ECF) reference system. Satellite positions, however, may be given in ECF coordinates to allow an easier plotting of the sub satellite points without introducing the need to convert to Earth fixed coordinates. Other choices for the orbital data reference system (e.g., both given with respect to ECI or ECF) are certainly also possible, and might be preferred by the data provider. The data provider should make sure that the corresponding meta data attributes specifying the reference system for each orbit parameter are set properly; see Section 3.4.6 for details. Any processing based on level 1a data should ensure that the coordinate systems are interpreted correctly and, if necessary, converted accordingly.

Level 1b (Table 3.3) data contains bending angles as function of impact parameter. The ROPP profile structure leaves room for four different impact parameter / bending angle pairs. One possible utilisation is to provide separate bending angle and impact parameters as derived from the L1 and L2 frequencies, and store the ionospheric corrected bending angle as the 'generic' bending angle / impact parameter pair. If a data provider chooses to provide 'statistically optimised' ionospheric bending angles, these could also be stored in the 'generic' variables. Note that the statistical optimisation process requires the use of some *a priori* data or assumption. We recommend to use an increasing impact parameter order.

Identifiers

Structure element	Parameter	Description	Range	Units
...%leo_id	LEO ID	LEO ID code (4 characters). The following ID codes are currently envisaged: `META/B/C` = Metop–A/B/C; `Cnnn` = COSMIC–*nnn* `GRAA` = GRACE–A; `GRAB` = GRACE–B `GRAC` = GRACE–C (FO); `GRAD` = GRACE–D (FO) `TSRX` = TerraSAR–X; `TDMX` = TanDEM–X `FY3C` = FY–3C; `FY3D` = FY–3D `CHMP` = CHAMP; `CNOF` = C/NOFS `SACC` = SAC–C; `OERS` = Oersted `SUNS` = SunSat; `KOM5` = KOMPSAT–5 `PAZE` = PAZ; `OSAT` = OceanSat–2 `MGTP` = Megha-Tropiques; `C2En` = COSMIC–2E*n* `SP3U` = Spire; `GOP1/2` = GeoOptics–1/2 `PLIA/B` = PlanetiQ–A/B; `SE6A/B` = Sentinel6–A/B Other LEO codes may be defined in the future.	[A-Z,0-9]	
...%gns_id	GNSS ID	GNSS ID code (1 letter + 3 digit PRN of the occulting GNSS satellite (‘Xnnn’)). The following ID codes are currently envisaged: `Gnnn` = GPS; `Rnnn` = GLONASS `Ennn` = Galileo; `Cnnn` = Beidou(/Compass) Other GNSS codes may be defined in the future.	[A-Z,0-9]	
...%stn_id	Station ID	Ground station ID used for differencing (if any; IGS-style 4–character code)	[A-Z,0-9]	
...%occ_id	Occultation ID	Unique occultation ID; see section 3.3.6	[A-Z,0-9]	

Table 3.1: Contents of an `ROprof` header (to be continued).

Processing

Structure element	Parameter	Description	Range	Units
...%FmtVersion	ROPP format version	Exact text (21 characters)	eg ROPP V9.1	
...%processing_centre	Processing Centre	Text indicating processing centre (40 characters)	[A-Z,0-9]	
...%processing_software	Processing centre software	Text indicating processing centre software (40 chars)	[A-Z,0-9]	
...%software_version	ROPP Version	String indicating the ROPP version	eg V9.1	
...%pod_method	POD algorithm	Text strings (40 characters) indicating algorithms used for deriving precise orbit, excess phase / amplitude, bending angle, refractivity and meteorological data	[A-Z,0-9]	
...%phase_method	Level 1a algorithm		[A-Z,0-9]	
...%bangle_method	Level 1b algorithm		[A-Z,0-9]	
...%refrac_method	Level 2a algorithm		[A-Z,0-9]	
...%meteo_method	Level 2b,c algorithm		[A-Z,0-9]	
...%thin_method	Profile thinning algorithm (version ID)			

Georeferencing

Structure element	Parameter	Description	Range	Units
...%GEOref%time_offset	Time since start	Time since start of occultation to the time when georeferencing data and radius of curvature are determined.	$-10 - 239.999$	s
...%GEOref%lat	Latitude	Position of tangent point as used for georeferencing	$-90 - 90$	deg
...%GEOref%lon	Longitude	Position of tangent point as used for georeferencing	$-180 - 180$	deg
...%GEOref%roc	Radius of curvature	Radius of curvature value	$(6.2 - 6.6) \times 10^6$	m
...%GEOref%r_coc	Centre of curvature	Centre of curvature coordinates (ECF; X, Y, Z)	$\pm 50.0 \times 10^3$	m
...%GEOref%azimuth	Line of sight	GNSS to LEO azimuth direction w.r.t. true North	$0 - 360$	deg_T
...%GEOref%undulation	Geoid undulation	Deviation of geoid (EGM–96) from the ellipsoid (WGS–84)[a]	± 150	m
...%GEOref%leo_pod%pos	Ref. LEO position	Reference LEO coordinates (ECF; X, Y, Z)[b]	$\pm 10.0 \times 10^6$	m
...%GEOref%leo_pod%vel	Ref. LEO velocity	Reference LEO velocities (ECI; X, Y, Z)[b]	$\pm 10.0 \times 10^3$	m/s
...%GEOref%gns_pod%pos	Ref. GNS position	Reference GNSS coordinates (ECF; X, Y, Z)[b]	$\pm 43.0 \times 10^6$	m
...%GEOref%gns_pod%vel	Ref. GNS velocity	Reference GNSS velocities (ECI; X, Y, Z)[b]	$\pm 10.0 \times 10^3$	m/s

[a] The height h_G with respect to the EGM–96 geoid, the height h_E with respect to the WGS–84 ellipsoid, and the undulation u are related by $h_E = h_G + u$.
[b] See footnotes of Table 3.2.

Table 3.1: Further contents of an ROprof header (to be continued).

Background meta data

Structure element	Parameter	Description	Range	Units
...%bg%source	Background source	Source of meteorological or atmospheric data used as background ("ancillary") data	[A-Z,0-9]	
...%bg%year ...%bg%month ...%bg%day ...%bg%hour ...%bg%minute	Verification time	Verification time of background data (if applicable)	1995 01 01 00 00 — 2099 12 31 23 59	
...%bg%fcperiod	F/C period	Forecast period of background data (if applicable)	0 – 24	hours

Time stamps

Structure element	Parameter	Description	Range	Units
...%DTocc%year ...%DTocc%month ...%DTocc%day ...%DTocc%hour ...%DTocc%minute ...%DTocc%second ...%DTocc%msec	Date / time of occultation	Time stamp at **start** of occultation (UTC)	1995 01 01 00 00 00 000 — 2099 12 31 23 59 59 999	
...%DTpro%year ...%DTpro%month ...%DTpro%day ...%DTpro%hour ...%DTpro%minute ...%DTpro%second ...%DTpro%msec	Date / time of processing	Time stamp of processing (UTC)	1995 01 01 00 00 00 000 — 2099 12 31 23 59 59 999	

Table 3.1: Further contents of an `ROprof` header (to be continued).

Quality				
Structure element	Parameter	Description	Range	Units
...%PCD	Product confidence	Product confidence data (see Section 3.3.5)	$0-(2^{15}-1)$	bit flags
...%overall_qual	Data quality	Overall summary data quality	$0-100$	%
Additional Variables				
Structure element	Parameter	Description	Range	Units
...%vlist%VlistD0d	0-D extra variables	Parameters of 0-D extra variables (see Section 3.4.10)		
...%vlist%VlistD1d	1-D extra variables	Parameters of 1-D extra variables (see Section 3.4.10)		
...%vlist%VlistD2d	2-D extra variables	Parameters of 2-D extra variables (see Section 3.4.10)		

Table 3.1: Further contents of an ROprof header (concluded).

Level 1a				
Structure element	Parameter	Description	Range	Units
...%Lev1a%dtime	Time since start	Time offset from time in header	$-1.0 - 539.999$	s
...%Lev1a%snr_L1ca	Signal to noise ratio L1 (ca–code)	Relative signal amplitude for L1 (ca–code)	$0 - 50 \times 10^3$	V/V
...%Lev1a%snr_L1p	Signal to noise ratio L1 (p–code)	Relative signal amplitude for L1 (p–code)	$0 - 50 \times 10^3$	V/V
...%Lev1a%snr_L2p	Signal to noise ratio L2 (p–code)	Relative signal amplitude for L2 (p–code)	$0 - 50 \times 10^3$	V/V
...%Lev1a%phase_L1	Excess phase L1	L1 phase corrected for geometry	$\pm 10 \times 10^3$	m
...%Lev1a%phase_L2	Excess phase L2	L2 phase corrected for geometry	$\pm 10 \times 10^3$	m
...%Lev1a%r_gns	Transmitter position	Earth centred Earth fixed,[a] phase centre (X, Y, Z)[b]	$\pm 43 \times 10^6$	m
...%Lev1a%v_gns	Transmitter velocity	Earth centred inertial,[a] phase centre (X, Y, Z)[b]	$\pm 10 \times 10^3$	m/s
...%Lev1a%r_leo	Receiver position	Earth centred earth fixed,[a] phase centre (X, Y, Z)[b]	$\pm 10 \times 10^6$	m
...%Lev1a%v_leo	Receiver velocity	Earth centred inertial,[a] phase centre (X, Y, Z)[b]	$\pm 10 \times 10^3$	m/s
...%Lev1a%phase_qual	Quality	Percentage confidence value	$0 - 100$	%

[a] Using the Earth Centred Fixed (ECF) and Earth Centred Inertial (ECI) reference frames for satellite positions and velocities, respectively, are the *default* settings; these can be changed, e.g. to use ECF for both positions and velocities.
[b] Position and velocity variables are 3–dimensional arrays with dimension `(/n,3/)` in Fortran.

Table 3.2: Contents of the `ROprof` Level 1a data.

Level 1b				
Structure element	Parameter	Description	Range	Units
...%Lev1b%lat_tp	Latitude	Latitude and longitude wrt the WGS 84 ellipsoid of the tangent points of the occultation's bending angles	± 90	deg
...%Lev1b%lon_tp	Longitude		± 180	deg
...%Lev1b%azimuth_tp	Azimuth	GNSS-to-LEO bearing wrt true North at tangent pts	$0 - 360$	deg_T
...%Lev1b%impact_L1	Impact parameter (L1)	Impact parameter derived from L1 signal	$(6.2 - 6.6) \times 10^6$	m
...%Lev1b%impact_L2	Impact parameter (L2)	Impact parameter derived from L2	$(6.2 - 6.6) \times 10^6$	m
...%Lev1b%impact	Impact parameter	Impact parameter (generic)	$(6.2 - 6.6) \times 10^6$	m
...%Lev1b%impact_Opt	Impact parameter (Opt)	Impact parameter for optimised Bending Angles	$(6.2 - 6.6) \times 10^6$	m
...%Lev1b%bangle_L1	Bending angle (L1)	Bending angle derived from L1	$-0.001 - 0.1$	rad
...%Lev1b%bangle_L2	Bending angle (L2)	Bending angle derived from L2	$-0.001 - 0.1$	rad
...%Lev1b%bangle	Bending angle	Bending angle (generic)[a]	$-0.001 - 0.1$	rad
...%Lev1b%bangle_Opt	Bending angle (Opt)	Bending angle optimised (usually smoothed) prior to performing the Abel Transform	$-0.001 - 0.1$	rad
...%Lev1b%bangle_L2_sigma	Bending angle errors (L1)	Estimated errors (one σ) of L1 bending angle values	$0 - 0.01$	rad
...%Lev1b%bangle_L2_sigma	Bending angle errors (L2)	Estimated errors (one σ) of L2 bending angle values	$0 - 0.01$	rad
...%Lev1b%bangle_sigma	Bending angle errors	Estimated errors (one σ) of bending angle values	$0 - 0.01$	rad
...%Lev1b%bangle_Opt_sigma	Bending angle errors (Opt)	Estimated errors (one σ) of optimised bending angle values	$0 - 0.01$	rad
...%Lev1b%bangle_L1_qual	Bending angle quality	Percentage confidence values for L1 bending angles	$0 - 100$	%
...%Lev1b%bangle_L2_qual	Bending angle quality	Percentage confidence values for L2 bending angles	$0 - 100$	%
...%Lev1b%bangle_qual	Bending angle quality	Percentage confidence values for bending angles	$0 - 100$	%
...%Lev1b%bangle_Opt_qual	Bending angle quality	Percentage confidence values for opt. bending angles	$0 - 100$	%

[a] In the output of `ropp_1dvar_dbangle` (ROM SAF, 2022a), `Lev1b%bangle` actually contains (forward modelled) `Lev1b%bangle_L2` – `Lev1b%bangle_L1`.

Table 3.3: Contents of the `ROprof` Level 1b (bending angle) data.

3.3.3 Level 2 profile data

Level 2 data in the `ROprof` structure consist of vertical profiles of refractivity (level 2a; Table 3.4), meteorological quantities (level 2b , 2c and 2d; Tables 3.5, 3.6 and 3.7), and ionospheric quantities (see Table 3.8).

Refractivity data are assumed to be in the usual N–units, so that the refractive index n of air is given by

$$n = 1 + N \times 10^{-6} \ .$$

The altitude referencing for refractivity is given in two ways, one using the geometric altitude above the reference geoid (EGM–96), and the other one being geopotential height. Note that geopotential height is related to potential energy within the gravity field of the Earth. It is therefore related to a particular model of the gravity field of the Earth, which we take to be the EGM–96 model. The SI unit of geopotential is m^2/s^2. For convenience, geopotentials are usually stored as heights in 'geopotential metres', which by WMO convention are defined as the geopotential divided by the standard value of gravity, namely 9.80665 m/s.

Level 2a data of the ROPP format does not include a latitude / longitude referencing for the refractivity profile. However, this information can easily be reconstructed from Level 1b data. The impact parameter p is first calculated from altitude above the geoid h using

$$p = rn(r) \quad \text{where } r = r_c + h$$

in which r_c is the radius of curvature (available from the header), and $n(r)$ is the refractive index at $r_c + h$. Latitude and longitude coordinates of individual refractivity data points can then be calculated by interpolating the impact parameter geolocation information contained in the level 1b data.

Temperature, geopotential height, pressure, and moisture data are contained in the level 2b part (Table 3.5). These may consist of the results of a 1D–Var retrieval of atmospheric parameters from level 1b or level 2a data. Other data-providers might supply dry temperatures and pressures, or specific humidity and the temperature used to derive the latter from refractivity, or some similar combination.

1D–Var retrievals may also provide some information on surface (or another reference) pressure which, together with data on the geopotential height of the (model) surface, is contained in the Level 2c data (see Table 3.6). The surface geopotential height is also defined as the rescaled surface geopotential. There may be inconsistencies in the way surface geopotential heights have been calculated from Digital Elevation Model data; for example, surface orography (i.e. elevation of the surface above sea level) may have been used instead of a properly calculated geopotential. Information on the details of the corresponding conversions should be made available by the data provider.

If the vertical level structure of retrieved or background meteorological profiles is based on NWP forecast model levels, the surface pressure and geopotential height will also be required to calculate information on the actual model levels. An example are the terrain following hybrid vertical coordinates as used by ECMWF, or the terrain following altitude coordinates as used in the Met Office system. The additional information needed to do this for hybrid coordinates is defined by the A- and B-level coefficients, which

Level 2a				
Structure element	Parameter	Description	Range	Units
...%Lev2a%alt_refrac	Height	Geometric height above geoid (EGM–96)	$(-1-150)\times10^3$	m
...%Lev2a%geop_refrac	Geopotential height	Geopotential height above geoid (EGM–96)	$(-1-150)\times10^3$	gpm
...%Lev2a%refrac	Refractivity	Derived refractivity	$0-500$	N–units
...%Lev2a%refrac_sigma	Refractivity error	Estimated errors (one σ) of refractivity values	$0-10$	N–units
...%Lev2a%refrac_qual	Refractivity quality	Percentage confidence value	$0-100$	%
...%Lev2a%dry_temp	Dry temperature	Derived dry temperature	$150-350$	K
...%Lev2a%dry_temp_sigma	Dry temperature error	Estimated errors (one σ) of dry temperature values	$0-50$	K
...%Lev2a%dry_temp_qual	Dry temperature quality	Percentage confidence value	$0-100$	%

Table 3.4: Contents of the ROprof Level 2a (refractivity) data. ROprof2d%Lev2a substructure extended to 2D at ROPP-10.0.

are held in the Level 2d substructure (see Table 3.7).

Ionospheric variables are held in the Level 2e substructure (see Table 3.8). Three elements — `n_e`, `n_e_sigma` and `r_iono` — define an electron density profile. These are held on `dim_lev2e` levels in an ROPP format netCDF file. The remaining elements — `ne_peak`, `ne_peak_sigma`, `r_peak`, `r_peak_sigma`, `h_zero`, `h_zero_sigma`, `h_grad` and `h_grad_sigma` — specify the parameters of model 'VaryChap' electron density distributions. These are held on `dim_layer` levels in an ROPP format netCDF file: one set of parameters for each VaryChap layer. VaryChap layers define electron density profiles $n_e(r)$ by:

$$n_e(r) = n_e^{\max} \exp\left((1-u-e^{-u})/2\right) \sqrt{H_0/H} \quad \text{where} \tag{3.1}$$

$$u = \log\left(H/H_0\right)/k \quad \text{and} \tag{3.2}$$

$$H = H_0 + k(r-r_0) \tag{3.3}$$

where $n_e^{\max}$ = `ne_peak`, r_0 = `r_peak`, H_0 = `h_zero` and k = `h_grad`. Eqns 3.1 – 3.3 apply if $r > r_0$. If $r \leq r_0$, or if $k \leq 10^{-3}$, the 'standard' Chapman layer approximation, which is given by the limit as $k \to 0$ of the above, applies — namely:

$$n_e(r) = n_e^{\max} \exp\left((1-u-e^{-u})/2\right) \quad \text{where} \tag{3.4}$$

$$u = (r-r_0)/H_0 \tag{3.5}$$

See (ROM SAF, 2019) for more details.

It is recommended to store profiles in ascending height order.

Level 2b				
Structure element	Parameter	Description	Range	Units
...%Lev2b%geop	Geopotential height	Geopotential height above geoid (EGM–96)	$(-1-100)\times10^3$	gpm
...%Lev2b%geop_sigma	Geopotential height error	Estimated error (one σ) of geopotential heights	$0-500$	gpm
...%Lev2b%press	Pressure	Retrieved pressure	$0.0001-1100$	hPa
...%Lev2b%press_sigma	Pressure error	Estimated error (one σ) of retrieved pressure	$0-5$	hPa
...%Lev2b%temp	Temperature	Retrieved temperature	$150-350$	K
...%Lev2b%temp_sigma	Temperature error	Estimated error (one σ) of retrieved temperature	$0-5$	K
...%Lev2b%shum	Specific humidity	Retrieved specific humidity	$0-50^{\dagger}$	g / kg
...%Lev2b%shum_sigma	Specific humidity error	Estimated error (one σ) of retrieved specific humidity	$0-50$	g / kg
...%Lev2b%meteo_qual	Quality	Overall percentage confidence value	$0-100$	%

Table 3.5: Contents of the ROprof Level 2b (background) data. †If the -nocheck_qmin option is activated in ropp_1dvar_refrac or ropp_1dvar_bangle the valid range of Lev2b%shum becomes $-20-50$ g / kg.

Level 2c				
Structure element	Parameter	Description	Range	Units
...%*L*ev2c%lat_2d	Latitude position	Latitude position (*R*Oprof2d structure only)	$-90-90$	deg
...%*L*ev2c%lon_2d	Longitude position	Longitude position (*R*Oprof2d structure only)	$-180-180$	deg
...%*L*ev2c%dtheta	Angle	Angle between profiles (*R*Oprof2d structure only)	$0-\pi$	rad
...%Lev2c%geop_sfc	Geopotential height	Geopotential height of surface above geoid (EGM–96)	$(-1-10)\times10^3$	gpm
...%Lev2c%press_sfc	Surface pressure	Retrieved surface (or reference) pressure	$250-1100$	hPa
...%Lev2c%press_sfc_sigma	Surface pressure error	Estimated error (one σ) of retrieved surface pressure	$0-10$	hPa
...%Lev2c%press_sfc_qual	Quality	Percentage confidence value	$0-100$	%
...%Lev2c%Ne_max	Electron density	Peak electron density	$-10^{15}-10^{15}$	m^{-3}
...%Lev2c%Ne_max_sigma	Electron density error	Error in peak electron density	$0-10^{15}$	m^{-3}
...%Lev2c%H_peak	Height	Height of peak electron density	$0-10^6$	m
...%Lev2c%H_peak_sigma	Height error	Error in height of peak electron density	$0-10^6$	m
...%Lev2c%H_width	Width	Width of electron density distribution	$0-10^6$	m
...%Lev2c%H_width_sigma	Width error	Error in width of electron density distribution	$0-10^6$	m
...%Lev2c%tph_bangle[a]	Impact parameter	Tropopause height derived from bending angle	$(6.2-6.6)\times10^6$	m
...%Lev2c%tpa_bangle[a]	Bending angle	Bending angle at TPH derived from bending angle	$-0.001-0.1$	rad
...%Lev2c%tph_bangle_flag[a]	Flag	QC flag for TPH derived from bending angle	$0-65535$	
...%Lev2c%pblh_bangle[a]	Height	1st planetary BL height derived from bending angle	-10^3-10^5	m
...%Lev2c%pblh_bangle2[a]	Height	2nd planetary BL height derived from bending angle	-10^3-10^5	m
...%Lev2c%pbla_bangle[a]	Bending angle	Bending angle at 1st PBLH derived from bend. angle	$-0.001-0.1$	rad
...%Lev2c%pbla_bangle2[a]	Bending angle	Bending angle at 2nd PBLH derived from bend. angle	$-0.001-0.1$	rad
...%Lev2c%pblh_bangle_flag[a]	Flag	QC flag for PBLH derived from bending angle	$0-65535$	

[a] Analagous tropopause height and planetary boundary layer height diagnostics, derived from other variables (refractivity, temperature, etc), exist in the Level 2c structure. They are fully discussed in the ROPP APPS User Guide (ROM SAF, 2022b).

Table 3.6: Contents of the ROprof (*and ROprof2d*) Level 2c (surface) data.

Level 2d				
Structure element	Parameter	Description	Range	Units
...%Lev2d%level_type	Level type	Level type. Only HYBRID ECMWF, ECMWF HYBRID, HYBRID, ECMWF and METOFFICE are currently supported.		
...%Lev2d%level_coeff_a	A-coefficients	Level A-coefficients (hybrid vertical levels only)	$0-2000$	hPa
...%Lev2d%level_coeff_b	B-coefficients	Level B-coefficients (hybrid vertical levels only)	$0-2$	n/a

Table 3.7: Contents of the ROprof Level 2d (level coefficients) data.

Level 2e				
Structure element	Parameter	Description	Range	Units
...%Lev2e%n_e	Electron density (n_e)	Profile of n_e	$0-10^{15}$	m^{-3}
...%Lev2e%n_e_sigma	Error in n_e	Profile of estimated error in n_e	$0-10^{15}$	m^{-3}
...%Lev2e%r_iono	Height of n_e	Impact parameter of n_e profile	$(6.2-7.4)\times10^6$	m
...%Lev2e%n_e_peak	Peak n_e	Peak n_e in each layer	$0-10^{15}$	m^{-3}
...%Lev2e%n_e_peak_sigma	Error in peak n_e	Peak n_e error in each layer	$0-10^{15}$	m^{-3}
...%Lev2e%r_peak	Height of peak n_e	Impact height of peak n_e in each layer	$0-10^6$	m
...%Lev2e%r_peak_sigma	Error in height of peak n_e	Error in impact height of peak n_e in each layer	$0-10^6$	m
...%Lev2e%h_zero	Scale height of n_e	Scale height of n_e in each layer	$0-10^6$	m
...%Lev2e%h_zero_sigma	Error in scale height of n_e	Error in scale height of n_e in each layer	$0-10^6$	m
...%Lev2e%h_grad	Gradient of scale height of n_e	Gradient of scale height of n_e in each layer	$0-1$	
...%Lev2e%h_grad_sigma	Error in gradient of scale height of n_e	Error in gradient of scale height of n_e in each layer	$0-1$	

Table 3.8: Contents of the ROprof Level 2e (ionospheric) data. The first three contain profile data, the rest are 'VaryChap' parameters.

3.3.4 Quality

The ROPP–format uses the concept of 'Quality' as a parameter in all of the sublevels and a 'Summary Quality' value in the header (see Table 3.1). In `BUFR` (ROM SAF, 2021), the term 'Percentage Confidence' is equivalent. The meaning of this parameter depends on the context and to a large degree on the specific processing algorithms used (and hence its derivation has to be defined by the data processing centre), but we may give some hints as to the intent.

At a user-level, Quality (or 'Percentage Confidence') can be used as a first indicator of whether a whole profile or a particular datum point is likely to be useful. In NWP, the concept of 'probability of gross error' is often used for quality control in variational assimilation systems. The objective is to reject observations with extreme errors, particularly if they do not lie within the normal (assumed) Gaussian error distribution.

In the simplest case, Quality (Q) could be derived from the estimated error(s) in the associated 'observed' parameter. Assuming an estimated observed error variance O and an *a priori* variance E, then

$$Q = 100 \exp(-O/E)$$

would be a suitable value. Errors for several observables may be combined to form an overall probability value.

Other Quality values might be derived from other QC information — for instance the Quality for bending angles might be nominally derived as above, but assigned a lower value (even zero) if loss–of–lock (fly wheeling) condition is detected. In this case, the observables may have apparently sensible values but there is little or no confidence in them. The end–user can then make the decision on whether to use such data or not.

3.3.5 Product Confidence Data (PCD)

Product confidence data (PCD) (in BUFR: Quality flags for RO) are held (see Table 3.1) as sets of bit flags within a single 32-bit / 4-byte integer value and indicate various product quality states or tests. Bits are numbered from 1. This format allows for 15 "information" bits (values 0–32727); setting bit 16 (values 32768–65535) indicates missing PCD data (i.e., none of the bit settings are valid). If some items are valid and others are not, invalid items should have their assigned bit clear (zero). The module `ropp_io` defines several integer parameters that are intended to be used with Fortran's internal `BTEST`, `IBCLR` and `IBSET` bit functions (see below). The defined bits and their Fortran parameter names are shown in Table 3.9. These bit flags are designed to be generic in that they can be applied to any mission and any processing scheme. Certain processing dependence is inevitable (e.g., meteorological quantities might be derived directly from bending angles, but it is likely that refractivity will be derived anyway, and the flagging can be mapped to the actual processing). The Software ID parameter and Processing Centre ID together can be used to trace back the detail of the particular processing algorithms in order to interpret the specific meaning of these flags.

The Fortran parameters shown in Table 3.9 become available after loading the `ropp_io_types` module in the variable declaration part of a Fortran subroutine or function. Testing the PCD flags can be

Bit	Variable	Description	Meaning if	
			unset (0)	set (1)
1	PCD_summary	Quality	nominal	non–nominal
2	PCD_offline	Product type	NRT	off line
3	PCD_rising	Occultation type	setting	rising
4	PCD_phase	Excess phase processing	nominal	non–nominal
5	PCD_bangle	Bending angle processing	nominal	non–nominal
6	PCD_refrac	Refractivity processing	nominal	non–nominal
7	PCD_met	Meteorological processing	nominal	non–nominal
8	PCD_open_loop	Open Loop	not used	used
9	PCD_reflection	Surface reflections detected	no	yes
10	PCD_l2_signal	L2P or L2C GNSS signal used	L2P	L2C
11	PCD_reserved_11	Reserved		
12	PCD_reserved_12	Reserved		
13	PCD_reserved_13	Reserved		
14	PCD_bg	Background profile	nominal	non–nominal
15	PCD_occultation	Profile type	observed	background
16	PCD_missing	PCD missing; bits 1–15...	valid	invalid

Table 3.9: Product confidence data definition. PCD_summary is a summary bit which shall be set if any of Bits 4,5,6,7 or 14 is set. Note that the PCD_* variables become available by USEing the module ropp_io_types.

accomplished with Fortran's BTEST function. To test if the overall data quality is nominal, for example, a construction like

```
IF (BTEST(ro_data%PCD, PCD_summary)) THEN
   ! ... do what needs to be done for non-nominal ...
ELSE
   !       ... and for nominal data quality.
END IF
```

allows the user to take appropriate action in a Fortran program, where ro_data is a structure of type ROprof. If using the bit number directly, rather than the ROPP parameter name, remember that Fortran BTEST numbers bits from zero.

When writing data to a ROPP data file, the user needs to set the PCD bits explicitly. This is required because initialising an ROprof structure sets its PCD value to 65535, indicating that the PCD is currently invalid. If only a few bits need to be set, the quickest way is to set the PCD to zero (all bits unset), and then to set the required bits. As an example, a profile retrieved off–line from a rising radio occultation measurement for which the overall quality as well as processing steps are nominal requires both the PCD_offline and the PCD_rising flags to be set. This can be achieved by

```
ro_data%PCD = 0
ro_data%PCD = IBSET(ro_data%PCD, PCD_offline)
ro_data%PCD = IBSET(ro_data%PCD, PCD_rising)
```

When storing simulated or background data in a ROPP file, the PCD_occultation flag must be set to

indicate that the file does not contain an actual occultation measurement:

```
ro_data%PCD = IBSET(ro_data%PCD, PCD_occultation) ! Background data, NOT an occultation
 ...                                              !       measurement!
```

This is particularly important as the setting of the Occultation ID (and therefore the generation of automated file names during `ropp_io_write`, see the following section) relies on proper setting of the `PCD_occultation` bit.

The summary bit `PCD_summary` shall be set if any of the other 'nominal' bits are set:

```
IF (BTEST(ro_data%PCD, PCD_phase)  .OR.
    BTEST(ro_data%PCD, PCD_bangle) .OR.
    BTEST(ro_data%PCD, PCD_refrac) .OR.
    BTEST(ro_data%PCD, PCD_met)    .OR.
    BTEST(ro_data%PCD, PCD_bg)) THEN
  ro_data%PCD = IBSET(ro_data%PCD, PCD_summary)
```

3.3.6 Occultation IDs and file names

The occultation ID in the header of a `ROprof` structure should be a unique identifier for a particular occultation event that could be used for cataloguing, filtering etc. This identifier could also be used as a file naming convention. In fact, writing data using the `ropp_write` interface without specifying a filename will result in the creation of a file with a name derived from the occultation ID.

It is recommended that the following format be used for the occultation ID field in the header of ROPP-format files, and that this also forms the base for the file names:

`tt_yyyymmddhhmmss_llll_gggg_pppp`

where

- `tt` denotes the type of the profile data (e.g., '`OC`' or '`BG`' for occultation and background data, respectively);
- `yyyymmdd` is the date (year, month, day) of the occultation event;
- `hhmmss` is the time (hour, minute, second) of the occultation event;
- `llll`, gggg and pppp denote LEO, GNSS and Processing Centre ID's.

File names, regardless whether generated from an occultation ID or otherwise, should have an extension indicating their technical file type. It is recommended to be '`.nc`' for the netCDF–based version of ROPP data files.

For convenience, the routine `ropp_io_occid` generates an occultation ID from the header information, and sets the `occ_id` parameter in the header to a value consistent with the above recommendations. This requires that all other elements of the header are set up properly:

```
CALL ropp_io_occid(ro_data) !  Sets ro_data%occ_id
```

The write routine `ropp_io_write` will auto–generate a file name from the occultation ID if no explicit

file is given by appending the occultation ID in the header with an extension appropriate for the file type. If the occultation ID is not available (i.e., if `...%occ_id` is set to '`UNKNOWN`' or empty), a suitable ID will be generated using the above convention.

3.4 Advanced use

3.4.1 Initialisation

Initialisation of `ropp_io` data types includes allocating pointers as well as setting the relevant integer elements of (sub–)structures to the number of elements in these arrays. The latter is of particular importance, as the software uses the counters to determine if a subsection contains data or not; for example, if `...%Npoints` equals zero, no data for the respective level is written. As already mentioned in Section 3.2, the easiest way to initialise a data structure for an RO profile is to call

```
CALL ropp_io_init(ro_data, n_lev_1a, n_lev_1b, &
                           n_lev_2a, n_lev_2b, n_lev_2c, n_lev_2d, n_lev_2e)
```

where the parameters `n_lev_1a`, `n_lev_1b`, `n_lev_2a`, `n_lev_2b`, `n_lev_2c`, `n_lev_2d` and `n_lev_2e` denote the number of elements in the arrays for each sublevel. The optional argument `n_lev_2e` has two elements: `n_lev_2e` = (/`Npoints`, `Nlayers`/), where `Npoints` defines the number of levels in the electron density profile and `Nlayers` defines the number of VaryChap layers that comprise it. See Table 3.8. Setting one or more of these numbers to zero or a negative number indicates that the corresponding sub–level does not include any data; this will also free all memory associated with these sub–parts should they have been used earlier.

Depending on circumstances, it might be more appropriate to initialise the relevant parts of an `ROprof` separately. This is possible by calling `ropp_io_init` with that part of the structure, and a single mandatory integer number defining the number of elements for that subpart. Examples are

```
CALL ropp_io_init(ro_data%Lev1a, n_lev_1a)
CALL ropp_io_init(ro_data%Lev1b, n_lev_1b)
CALL ropp_io_init(ro_data%Lev2a, n_lev_2a)
CALL ropp_io_init(ro_data%Lev2b, n_lev_2b)
CALL ropp_io_init(ro_data%Lev2c, n_lev_2c)
CALL ropp_io_init(ro_data%Lev2d, n_lev_2d)
CALL ropp_io_init(ro_data%Lev2e, n_lev_2e)
```

The above subroutine calls initialise the `Npoints` elements of each subpart to the appropriate number, allocates all arrays within the `ROprof` structure, and fills the allocated arrays with predefined "missing values". Note that the scalar surface part (Level 2c) does not require a specific initialisation of arrays. However, for surface data the number of Level 2c elements must be set to one (the only allowed value apart from 0, indicating that no surface pressure data is contained in the profile data). While all allocations could alternatively been undertaken by the user, calling the provided subroutines for this task will ensure consistency of the data.

3.4.2 Freeing memory

Occasionally, it will be required to free (deallocate) all memory in a given data structure. The call

```
CALL ropp_io_free(ro_data)
```

will deallocate all arrays from all sublevels, and set the corresponding `...%Npoints` variables to zero.

It will also set all scalar variables (including all information in the header) to default missing values.

The individual sublevels of an `ROprof` structure can be freed with one of

```
CALL ropp_io_free(ro_data%Lev1a)
CALL ropp_io_free(ro_data%Lev1b)
CALL ropp_io_free(ro_data%Lev2a)
CALL ropp_io_free(ro_data%Lev2b)
CALL ropp_io_free(ro_data%Lev2c)
CALL ropp_io_free(ro_data%Lev2d)
CALL ropp_io_free(ro_data%Lev2e)
```

Note, however, that the above sequence is *not* equivalent to freeing the same data structure at once, as the individual calls to `ropp_io_free` will still leave the header untouched.

The main intended use of the subroutine is to allow the re–use of the same variable for different profiles, possibly with a different number of data values. Therefore, meta–data describing units and valid ranges are *not* reset, in order to keep settings the user might have changed from the default. Also note that the current interface only allows deallocating scalar data structures; if an array of structures is to be freed, a loop construction as in

```
do i = 1, size(ro_profiles)
   CALL ropp_io_free(ro_profiles(i))
end do
```

is required.

3.4.3 Unit conversion

Data files written by the `ropp_io` library contain data in standard physical units which are documented in the `units` attribute for each variable. These standard units are listed in Tables 3.1–3.8.

A user application may require the data in different (compatible) physical units. Therefore, the `ROprof` structure contains variables specifying the units of its elements. For details, see the reference documentation. After having been declared, the unit specifying variables contain the standard set of units. By changing the standard values of the unit specification to the desired ones *before* reading data from an ROPP data file with `ropp_io_read`, a user can request the conversion of the data in the file to the desired physical units. For example, to specify the altitude for refractivity data in units of kilometres, and the geopotential in units of m^2/s^2,

```
ro_data%Lev2a%units%alt_refrac  = 'km'
ro_data%Lev2a%units%geop_refrac = 'm^2/s^2'
 ...
CALL ropp_io_read(ro_data, filename)
```

The ranges (maximum and minimum valid data values for each parameter) are also converted to the new units, so range–checking in the new units remains consistent. The mechanism also works when the ROPP data file contains units different from the standard units, as long as the `units` and `range` attributes in the netCDF based ROPP data file are consistent with the actual physical units of the data.

All unit conversions within ropp_io are performed using the unitconvert utility library provided in the ROPP UTILS module — see its User Guide (ROM SAF, 2022f).

Similarly, a program might internally produce variables in physical units different from ROPP's standard units. The ropp_io library will take care of the necessary unit conversions if the respective unit specifying variables of the ROprof structure are set to the ones used by the program *before* the data is written by ropp_io_write. For example, consider a program which produces a retrieval of meteorological quantities with temperature units of degree Celsius, pressure units of Pa, specific humidity units of kg/kg, and geopotential height units of geopotential kilometre. The same units are used for the estimated uncertainties. Exploiting the conversion capability of the ropp_io library would require

```
ro_data%Lev2b%units%press            = 'Pa'
ro_data%Lev2b%units%press_sigma      = 'Pa'
ro_data%Lev2b%units%temp             = 'degreeC'
ro_data%Lev2b%units%temp_sigma       = 'degreeC'
ro_data%Lev2b%units%shum             = 'kg/kg'
ro_data%Lev2b%units%shum_sigma       = 'kg/kg'
ro_data%Lev2b%units%geop             = 'geopotential km'
ro_data%Lev2b%units%geop_sigma       = 'geopotential km'
ro_data%Lev2c%units%press_sfc        = 'Pa'
ro_data%Lev2c%units%press_sfc_sigma  = 'Pa'
ro_data%Lev2c%units%geop_sfc         = 'geopotential km'
 ...
CALL ropp_io_write(ro_data, filename)
```

3.4.4 Range checking

The ropp_io module contains a range checking routine which checks every parameter in the ROprof structure — including header parameters and strings as well as numeric values — for validity. Numeric parameters have attributes of range (minimum and maximum valid limits) against which data values are tested. Invalid (out-of-range) points are set to 'missing'. String parameters are tested against a valid character set or set of valid options. Further, profile basic coordinates (time for Level 1a, impact parameter, geometric or geopotential heights for Levels 1b, 2a and 2b, respectively) are tested for validity; invalid points are removed from the profile, and the Npoints value set to the number of valid levels in the profile. Note that this procedure does *not* remove levels with invalid observed data points (bending angles, etc), only those with invalid basic coordinates (ie, independent variables).

This validity checking is accomplished by calling:

```
CALL ropp_io_rangecheck(ro_data)
```

This operation is performed within the ROPP thinner routine ropp_io_thin() and may be performed on read or write by setting the optional ranchk keyword argument in the call to ropp_io_read() or ropp_io_write().

3.4.5 Missing data

Special values are reserved to indicate that data are 'missing' or invalid; for most parameters this is set to the 'missing data flag value' defined by `ropp_MDFV` (some parameters are better set to zero (= `ropp_ZERO`)). Currently, this special value is `-99999000.0`, but this may change at a future release.

Having done a range check operation, the parameter 'missing data test value' `ropp_MDTV` can be used to test for missing data thus:

```
IF ( ro_data%georef%undulation < ropp_MDTV ) THEN
    ! ... can't apply geoid/ellipsoid conversion ...
ELSE
    ! ... apply geoid/ellipsoid conversion ...
END IF
```

These three missing data related parameters are defined in the `ropp_utils` module. The `ropp_MDFV` value is written to output files as the global attribute `_FillValue`. Note that no checking of data is performed against user-defined `_FillValue` attributes on reading an input file in the current release.

Parameters set to MDTV are exempt from units conversion, so will always retain this special value for consistent testing.

3.4.6 Reference systems for POD data

Precise Orbit Determination (POD) data are given with respect to reference terrestrial or stellar reference systems; typical ones are "Earth Centred, Earth Fixed" (ECF) or "Earth Centred Inertial" (ECI) coordinates. Another coordinate system sometimes used in satellite Geodesy is the "True of Date System" (TDS). The `ROprof` data structure provides a means for data producers to specify which reference system is used for the provided POD data; the variables

```
ro_data%Lev1a%reference_frame%r_gns  = 'ECF' | 'ECI' | ...
ro_data%Lev1a%reference_frame%v_gns  =          ...
ro_data%Lev1a%reference_frame%r_leo  =          ...
ro_data%Lev1a%reference_frame%v_leo  =          ...
```

are intended to be used by data producers to specify which reference systems is used. Note that the default settings are

```
ro_data%Lev1a%reference_frame%r_gns  = 'ECF'
ro_data%Lev1a%reference_frame%v_gns  = 'ECI'
ro_data%Lev1a%reference_frame%r_leo  = 'ECF'
ro_data%Lev1a%reference_frame%v_leo  = 'ECI'
```

for the reasons discussed in Section 3.3.

3.4.7 Output precision

ROPP data files contain most floating point variables as single precision. The exception are variables holding internal time, bending angle, heights and POD data which are stored as `double` to cover the possible physical range of the data to sufficient numerical precision. The default behaviour can be

changed by setting the optional `output_precision` argument to the `ropp_io_write` routine, as in

```
CALL ropp_io_write(ro_data, filename, output_precision='double')
```

This will cause all floating point variables to be written as double precision netCDF variables. Note that double precision output may increase the size of data files considerably.

3.4.8 Multi–profile data files

The examples presented so far all assumed that an individual ROPP data file contains only a single profile. It is also possible to store multiple profiles in one data file. In the following, we will call such 'multi–profile' data file a *multifile*, in contrast to *singlefiles* holding exactly one profile.

A major (but only) restriction for multifiles is that the number of data points in any given level must be equal for all profiles. Thus, it is possible to combine retrievals (say, levels 2b, c, and d) of atmospheric parameters on a fixed set of altitude or pressure levels into a multifile. Level 1a data from several profiles, each having a different number of raw data samples, can only be merged into a multifile if the individual data arrays are padded with missing values to exhibit the same number of elements. This means that the first profile must be the (joint) largest. The latter comes at the expense of increased memory requirements for the data.

There are several ways to read and write multifiles, all described in the following subsections. An example for the use of the multifile interface as described above can be found in the source code of the `ropp2ropp` program (see Section 3.6 on page 55), which is part of `ropp_io` and fully supports copying, merging and splitting multifiles.

Sequential read

In a database–like terminology, the profiles in a multifile correspond to one of many records in the netCDF file; a singlefile is simply a data file with a single record. Assuming the following variable declarations,

```
USE ropp_io
TYPE(ROprof)         :: ro_data
INTEGER              :: n_prof, i
CHARACTER (LEN=...) :: filename = '...'
```

the number of records (i.e., profiles) is available through

```
n_prof = ropp_io_nrec(filename)
```

The *i*–th profile can be read by

```
CALL ropp_io_read(ro_data, filename, rec=i)
```

and the entire data set could be processed with

```
n_prof = ropp_io_nrec(filename)
   ...
DO i = 1, n_prof
   CALL ropp_io_read(ro_data, filename, rec=i)
   ! Process the data as required...
```

```
END DO
```

If `rec` is not specified, the first record present in the data file is read.

Read–at–once

Data contained in a multifile can also be ingested in a single `ropp_io_read` call. This requires that the variable holding the set of profiles is declared as a 1–dimensional *pointer* of type `ROprof`, i.e.

```
TYPE(ROprof), DIMENSION(:), POINTER :: ro_data_set => NULL()
```

where `ro_data_set` is explicitly nullified in the declaration. Reading the entire data set at once then only requires

```
CALL ropp_io_read(ro_data_set, filename)
```

The subroutine `ropp_io_read` will allocate sufficient amounts of memory to hold all profiles in `filename`. The elements of the `ROdata` structure for individual profiles can be accessed via

```
ro_data_set(i)%...
```

and the number of profiles read is given by

```
n_prof = SIZE(ro_data_set)
```

Sequential write

The default write mode is to overwrite already existing data files. Instead, data can be appended to an already existing data file with

```
CALL ropp_io_write(ro_data, filename, append=.true.)
```

Note that a new file will still be created if the file `filename` does not exist. In effect, this allows the user to sequentially add profiles to an existing data file. Similar to `ropp_io_read`, the optional parameter `rec` is also supported: adding profile data as i–th record in an existing data file can be achieved with

```
CALL ropp_io_write(ro_data, filename, rec=i, append=.true.)
```

Note that the `append` argument needs to be given as well.

Write–at–once

Similar to read–at–once, an entire data set of profiles can be written into an ROPP multifile in one go by declaring

```
TYPE(ROprof), DIMENSION(:), POINTER :: ro_data_set => NULL()
```

After `ro_data_set` has been allocated and filled with data, the entire data set can be written with

```
CALL ropp_io_write(ro_data_set, filename)
```

The `append` and `rec` arguments can be used in a similar way as in sequential writing. For example,

```
CALL ropp_io_write(ro_data_set, filename, append=.true.)
```

will append the `ro_data_set` to an already existing data file (instead of overwriting it), while

```
CALL ropp_io_write(ro_data_set, filename, rec=i, append=.true.)
```

will append the data set to an already existing file, *starting* at the *i*–th record.

NetCDF operators (nco)

The netCDF operators (nco)[3] are a versatile set of Unix–style command line tools to manipulate netCDF data files. While ropp_io does not require them to be available, we highly recommend nco as generic tool set for netCDF data files. The operator ncrcat can be used to merge several ROPP data file into a single one (i.e., to create a multifile) with

```
ncrcat <file_1.nc> <file_2.nc> ... <file_n.nc> -o <multifile.nc>
```

As an example, consider that a directory daily contains ROPP data files from a single day. Combining all of them into a single data file today.nc would require

```
ncrcat daily/* -o today.nc
```

Individual profiles (or ranges of profiles) can be extracted from a multifile using the ncks operator exploiting its -d option:

```
ncks -d dim_unlim,<start>[,<end>] <multifile.nc> -o <resultfile.nc>
```

where start and end denote the start and end indexes of the range of profiles to be extracted. Note that, by default, dimension indexes for ncks are zero based as in C, i.e. 0 corresponds to the first profile, 1 to the second etc. (Fortran-like indexing — 1, 2, etc — can be invoked by using the -F option in most routines.) To extract the second profile from a file named mropp_test.nc, therefore, the appropriate ncks command would be

```
ncks -d dim_unlim,1 mropp_test.nc -o profile_2.nc
```

Note that ncks also supports an index convention similar to Fortran with the -F option. Thus, the command

```
ncks -F -d dim_unlim,2 mropp_test.nc -o profile_2.nc
```

yields the same result as the previous example.

In general, the use of ncrcat and ncks for merging and splitting ROPP files provides the same data content as manually created single- and multifiles. Differences will occur due to different entries in the global history attribute of the generated data set. For the nco operators, adding information to the history attribute can be disabled with -h; the results should then be identical to those obtained with, e.g., ropp2ropp.

3.4.9 Arrays of structures

The previous section introduced arrays of structures of type ROprof, e.g. variables declared like

```
TYPE(ROprof), DIMENSION(:), POINTER :: ro_data_set => NULL()
```

It is important to remember that two allocation or deallocation steps have to be undertaken when dealing with structure arrays: First, an array of structures needs to be allocated; then, the elements of

[3] See http://nco.sourceforge.net.

the individual structure elements need to be initialised. Similarly, freeing the memory taken up by an array of structures such as `ro_data_set` in the above examples requires that the allocated memory of the individual structures is free before the array of structures itself is finally deallocated. A complete memory allocation and deallocation cycle would look like the following example:

```
! 1. Allocate an array of structures
! ----------------------------------
  ALLOCATE(ro_data_set(n_prof))

! 2. Initialise individual structures
! ----------------------------------
  DO i = 1, n_prof
     CALL ropp_io_init(ro_data_set(i), ...)
  END DO

! 3. Fill individual structures
! -----------------------------
! Individual code goes here...

! 4. Free individual structures
! -----------------------------
  DO i = 1, n_prof
     CALL ropp_io_free(ro_data_set(i))
  END DO

! 5. Deallocate array of structures
! ---------------------------------
  DEALLOCATE(ro_data_set)
```

3.4.10 Extending the `ropp_io` data type

NetCDF data is ultimately referenced by its variable name; additional data items can be added at any time as long as the added data uses variable names different from the already stored data. The presence of such additional data does not compromise the ability of existing readers to ingest the variables they know of. Thus, netCDF (and therefore ROPP) data files can easily be extended beyond the core set of variables as defined in the original `ROprof` structure, simply by adding additional variables to the data file. The only restriction is that the netCDF names of the additional variables may not coincide with the netCDF names used for the ROPP core variables.

In order to facilitate a simple way to write such additional (or non–core) variables, `ropp_io` offers the ability to extend the `ROprof` data type: prior to writing, additional variables can be "appended" to an existing profile structure by specifying the data to be written and attributes required for the later storage in the netCDF data file. The required functionality is provided by the `ropp_io_addvar` routine. The following example shows how to add the value of the cost function of a 1D-Var retrieval and the

Probability of Gross Error.

```
USE typesizes, only: dp => EightByteReal
USE ropp_io
TYPE(ROprof) :: ro_data
REAL(wp)     :: cost
REAL(wp)     :: pge
   ...
CALL ropp_io_addvar(ro_data,                                        &
                    name      = "J",                                &
                    long_name = "Cost function value at convergence", &
                    units     = "",                                 &
                    range     = (/ 0.0_dp, 9999.0_dp/),             &
                    data      = cost)
CALL ropp_io_addvar(ro_data,                                  &
                    name      = "pge",                        &
                    long_name = "Probability of gross error", &
                    units     = "%",                         &
                    range     = (/ 0.0_dp, 100.0_dp/),        &
                    data      = pge)
```

where `cost` and `pge` are the program's Fortran variables holding the respective values. If `ro_data` is later written to a netCDF data file with `ropp_io_write`, additional variables named `J` and `pge` will be created in the netCDF and filled with the contents of the corresponding `data`. The netCDF standard attributes for `long_name`, `units` and `valid_range` are taken from the `long_name`, `units` and `range` arguments to the `ropp_io_addvar` routine, respectively.

The shape of the `data` arguments defines the shape of the variables written to the netCDF data file: if the argument to `data` is scalar (like in the above example), it is assumed that each profile within the data file obtains an additional scalar value. Similarly, one and two dimensional `data` arguments will create vector or matrix shaped additional variables. Note that the current implementation only supports double precision scalar, one or two–dimensional floating point data.

If non-core variables are present in a netCDF file to be read in with `ropp_io_read` the `ROprof` data type is similarly extended, so that all variables in a netCDF file are preserved on read/write using `ropp_io`.

Additional variables are stored as a linked list in the `ROprof%vlist` substructure. The idea is shown in Table 3.10.

3.4.11 Alternative ways to read ROPP files

Since ROPP data files are regular netCDF files, they can be read with any software or application that is able to read netCDF data files; all that is required is the knowledge about the variables in the netCDF data file and their meaning. Apart from the native netCDF interface supporting C, C++, Fortran 77 and Fortran 90/95, graphics systems like IDL, PV Wave, Matlab, R or NCAR Graphic's ncl can read (and write) netCDF data. There are also APIs for other programming languages like Perl, Python or

Additional variables

Structure element	Parameter	Description	Range	Units
...%vlist%VlistD0d%name	Name	Name of 1st 0D extra variable	[A-Z,0-9]	
...%vlist%VlistD0d%long_name	Long name	Long name of 1st 0D extra variable	[A-Z,0-9]	
...%vlist%VlistD0d%units	Units	Units of 1st 0D extra variable	[A-Z,0-9]	
...%vlist%VlistD0d%range	Range	Range of 1st 0D extra variable		
...%vlist%VlistD0d%DATA	Data	Value of 1st 0D extra variable		
...%vlist%VlistD0d%next%name (etc)	Name (etc)	Name (etc) of 2nd 0D extra variable	[A-Z,0-9]	
...%vlist%VlistD0d%next%next%name (etc)	Name (etc)	Name (etc) of 3rd 0D extra variable	[A-Z,0-9]	
...%vlist%VlistD1d%name	Name	Name of 1st 1D extra variable	[A-Z,0-9]	
...%vlist%VlistD1d%long_name	Long name	Long name of 1st 1D extra variable	[A-Z,0-9]	
...%vlist%VlistD1d%units	Units	Units of 1st 1D extra variable	[A-Z,0-9]	
...%vlist%VlistD1d%range	Range	Range of 1st 1D extra variable		
...%vlist%VlistD1d%DATA	Data	Value of 1st 1D extra variable		
...%vlist%VlistD1d%next%name (etc)	Name (etc)	Name (etc) of 2nd 1D extra variable	[A-Z,0-9]	
...%vlist%VlistD1d%next%next%name (etc)	Name (etc)	Name (etc) of 3rd 1D extra variable	[A-Z,0-9]	
...%vlist%VlistD2d%name	Name	Name of 1st 2D extra variable	[A-Z,0-9]	
...%vlist%VlistD2d%long_name	Long name	Long name of 1st 2D extra variable	[A-Z,0-9]	
...%vlist%VlistD2d%units	Units	Units of 1st 2D extra variable	[A-Z,0-9]	
...%vlist%VlistD2d%range	Range	Range of 1st 2D extra variable		
...%vlist%VlistD2d%DATA	Data	Value of 1st 2D extra variable		
...%vlist%VlistD2d%next%name (etc)	Name (etc)	Name (etc) of 2nd 2D extra variable	[A-Z,0-9]	
...%vlist%VlistD2d%next%next%name (etc)	Name (etc)	Name (etc) of 3rd 2D extra variable	[A-Z,0-9]	

Table 3.10: Contents of the `ROprof` additional variables.

Java. For an exhaustive list on programming environment and graphical applications that do support netCDF, see the netCDF website (Unidata, -).

A particularly simple interface for reading (and also writing) netCDF data from Fortran is the freely available `ncdf90` high level interface to netCDF. In fact, the `ropp_io` library is built on top of `ncdf90`, and contains most of its source code. Thus, all variables contained in an ROPP data file can also be read (from Fortran) using the `ncdf` interface which is part of `ropp_io`. Compared to the standard read method using `ropp_io_read()` this can yield performance improvements if only a small subset of the data is required by the application. For example, to read refractivity and its geopotential height from the Level 2a part, along with its coordinates and the date and time of the occultation, the following would suffice (the example assumes that the Fortran variables `refrac` and `geop` are one dimensional real pointers):

```
USE ncdf
CALL ncdf_open(filename)
   ...
CALL ncdf_getvar('lon', lon,  rec=i)
CALL ncdf_getvar('lat', lat,  rec=i)
CALL ncdf_getvar('js',  time, rec=i, units='seconds since 2004-10-01 12:00')
   ...
CALL ncdf_getvar_alloc('refrac',      refrac, rec=i)
CALL ncdf_getvar_alloc('geop_refrac', geop,   rec=i , units='m^2/s^2')
   ...
CALL ncdf_close()
```

In a similar way, the `J` and `pge` data added to a ROPP data file (as described in the previous section) might be read using

```
USE ncdf
CALL ncdf_open(filename)
   ...
CALL ncdf_getvar('J', J, rec=i)
   ...
CALL ncdf_getvar_alloc('pge', pge, rec=i)
   ...
CALL ncdf_close()
```

where `J` is a scalar, and `pge` a one–dimensional double precision pointer variable. Note that the specification of the `rec` argument is mandatory in the above example, as precisely one profile is to be read. The variable `i` denotes the record number, and might be 1 for singlefiles.

For details on the various `ncdf` routines, see the ROPP I/O Reference Manual.

3.4.12 NcView and NcBrowse

NcView[4] is a freely available visual browser for netCDF format files. Users may find this package useful for a quick and easy way to look at the contents of a netCDF file. NcView runs on unix/Linux under the X-windows system. NcBrowse[5] is an alternative graphical netCDF file browser, and being written in Java is a cross-platform application, so is particularly useful on MS Windows and Mac OS-X.

3.5 Plotting ROPP data

In the following sections, we give a few hints on how to read and plot data in the ROPP format into various graphics programs that support direct input of netCDF data. For an exhaustive list on graphical applications that support netCDF, see the netCDF website (Unidata, -). The `ropp_io/contrib` directory of the source code distribution contains some tools for data ingestion for `IDL` and `PV-Wave` (which are not fully supported, however). Note that the plots generated by the code in the following examples are not intended to be publication quality figures. They are provided for illustration purposes only.

3.5.1 IDL

The commercial graphics language IDL[6] (Interactive Data Language) provides native read and write access to netCDF data files. The directory `ropp_io/contrib/idl` contains the IDL procedure `ncdf_getvar.pro` to read data from generic netCDF data files. The following sequence of commands would create a simple plot of a retrieved temperature profile (note that the variable `ncdf` is assumed to be set to the file name of the ROPP data file):

```
ncdf_getvar, ncdf, 'press', press ; Read pressure.
ncdf_getvar, ncdf, 'temp', temp   ; Read temperature.

temp = temp - 273.15              ; Convert to degree C.

plot, temp, press, xtitle = "Temperature (K)", xrange = [-90., 30.], $
                   ytitle = "Pressure (hPa)", yrange = [1000., 0.1], /ylog
```

Note that `ncdf_getvar` requires the netCDF variable names; it has no knowledge of the `ROprof` data structure. For the full documentation of `ncdf_getvar`, run

```
ncdf_getvar, /help
```

at the IDL prompt or read the full documentation in the routine's header. In order to run `ncdf_getvar` from the IDL command line prompt, the file `ncdf_getvar.pro` must have been installed somewhere in IDL's search path, or the current directory; see the IDL documentation for details. Note that the routines in the `contrib` directory are provided for convenience only, and are not fully supported by the ROM SAF.

[4] See http://meteora.ucsd.edu/~pierce/ncview_home_page.html.
[5] See http://www.epic.noaa.gov/java/ncBrowse.
[6] See http://www.ittvis.com/language/en-US/ProductsServices/IDL.aspx.

3.5.2 PV-Wave

PV–Wave[7] is another proprietary graphics system, originally derived from an early version of IDL. Reading netCDF data files is supported as part of the HDF interface, although the latter is only poorly documented. The `ropp_io/contrib/wave` directory contains some routines that mimic IDL's read interface to netCDF data files, and also the routine `ncdf_getvar`. Creating a simple plot is very similar to IDL; the commands corresponding to the above example would be

```
ncdf_getvar, ncdf, 'press', press ; Read pressure.
ncdf_getvar, ncdf, 'temp', temp   ; Read temperature.

temp = temp - 273.15              ; Convert to degree C.

plot_io, temp, press, xtitle = "Temperature (K)", xrange = [-90., 30.], $
                      ytitle = "Pressure (hPa)", yrange = [1000., 0.1]
```

In order to run `ncdf_getvar` from the PV–Wave command line prompt, the file `ncdf_getvar.pro` as well as the other support files must have been installed somewhere in Wave's search path, or in the current directory; see the Wave documentation for details. Again, note that all routines in the directory `contrib` have only been provided as a matter of convenience, and are not fully supported by the ROM SAF.

3.5.3 Python

After importing the Python modules `matplotlib`[8] and `netCDF4`[9] as follows,

```
import matplotlib.pyplot as plt
import netCDF4 as nc,
```

a NetCDF file can be opened with:

```
ds = nc.Dataset('/path/to/file/example_file.nc', mode='r')
```

Temperature and pressure data could then be read with the following commands:

```
x = ds.variables['temp'][:]
x = x - 273.15 # Convert to degrees C
y = ds.variables['press'][:]
```

The temperature profile could then be plotted as follows:

```
plt.plot(x[0],y[0])
plt.xlabel('Temperature (K)')
plt.ylabel('Pressure (hPa)')
plt.yscale('log')
plt.show() # Display plot in a window
```

[7] See http://www.roguewave.com/products/pv-wave-family.aspx.
[8] Pyplot documentation available here: https://matplotlib.org/api/pyplot_summary.html
[9] netCDF4 API documentation available here: https://unidata.github.io/netcdf4-python/netCDF4/index.html

To find out the other variables in the NetCDF file using Python, one could use

```
print(ds.variables)
```

Note that 'full-fat' netCDF-4 files have a 'directory' structure, and so the lines to read the variables would look more like:

```
x=ds.groups['data']['level_1b']['thinned']['bangle'][:]
y=ds.groups['data']['level_1b']['thinned']['impact_height'][:]
```

The variable names in such files could be discovered with something like:

```
print(ds.groups['data']['level_1b'])
```

3.5.4 R

R[10] is a freely available environment for statistical computing and graphics. There are various packages available from on the Comprehensive R Archive Network[11] which facilitate the reading and writing of netCDF data; a good choice is `RNetCDF`. A plot similar to the ones discussed for the previous graphics packages can be generated by (assuming that the `RNetCDF` library has been installed):

```
library(RNetCDF)                          # Load the RNetCDF library.
nc    <- open.nc(ncdf)                    # Read the data.
temp  <- var.get.nc(nc, "temp") - 273.15
press <- var.get.nc(nc, "press")
close.nc(nc)

plot(temp, press, log = "y", type = "l", ylim=c(1000,0.1))  # Make the plot.
```

3.5.5 ncl

The NCAR Graphics Command Language[12] (ncl) is a freely available graphics system which supports netCDF as one of its main data formats. Creating a similar plot as in the IDL and PV–Wave examples before would require (the variable `ncdf` contains the file name of an ROPP singlefile, and the file `gsn_code.ncl` must have been loaded):

```
ro_profile = addfile(ncdf, "r")       # Read the data.
press = ro_profile->press
temp  = ro_profile->temp

temp = temp - 273.15                  # Convert to degree C.

wks = gsn_open_wks("x11", "example")

res             = True                # Set resources.
```

[10] See http://www.r-project.org.
[11] See http://cran.r-project.org.
[12] See http://www.ncl.ucar.edu.

```
res@trYLog     = True
res@trYReverse = True
res@trXMinF    =  -90.
res@trXMaxF    =   30.
res@trXMinF    =    0.1
res@trXMaxF    = 1000.

plot = gsn_xy(wks, temp, press, res) # Make the plot.
```

3.6 Data handling and conversion tools

3.6.1 ropp2ropp

The program ropp2ropp provides a management tool for ROPP netCDF files. The program supports the merging of several ropp files into a single multifile, as well as the splitting of multifiles into several files, each holding a single individual profile. The program may therefore be used to handle simple data organisation tasks. Note that individual files which are to be merged must fulfil the requirements for data in multifiles, i.e. the various data levels must exhibit the same number of elements across all data files, or an error will occur. ropp2ropp can also be used to thin datasets — see Sec 3.7 for details.

Summary help information on all of the options available to ropp2ropp can be found by typing:

```
> ropp2ropp -h
```

or for more detailed information consult the Unix manual page:

```
man ropp2ropp
```

or the ropp_io Reference Manual.

3.6.2 bufr2ropp, bufr2ropp_mobufr, bufr2ropp_ecbufr, bufr2ropp_eccodes, ropp2bufr, ropp2bufr_mobufr, ropp2bufr_ecbufr and ropp2bufr_eccodes

If the MetDB, ECMWF or ecCodes BUFR libraries are available on the user's computer system (see Appendix A), the executables (bufr2ropp_mobufr and ropp2bufr_mobufr), (bufr2ropp_ecbufr and ropp2bufr_ecbufr) or (bufr2ropp_eccodes and ropp2bufr_eccodes), respectively, will be built.

bufr2ropp is now a soft link to (in order of preference) bufr2ropp_eccodes, bufr2ropp_ecbufr or bufr2ropp_mobufr. Similarly, ropp2bufr is now a soft link to (in order of preference) ropp2bufr_eccodes, ropp2bufr_ecbufr or ropp2bufr_mobufr. Any particular tool can be called with the -v switch to check which library it was built on.

These programs convert between the ROPP netCDF and WMO BUFR formats and vice versa. For details on their usage, call them with the -h option, consult the respective Unix manual pages or see the ropp_io Reference Manual.

For example, to encode and decode RO data:

```
> ropp2bufr <input_filename.nc> -o <output_filename.bufr>
> bufr2ropp <input_filename.bufr> -o <output_filename.nc>
```

Or, if the user prefers to specify the executable to be used:

```
> ropp2bufr_eccodes <input_filename.nc> -o <output_filename.bufr>
> bufr2ropp_eccodes <input_filename.bufr> -o <output_filename.nc>
```

Likewise for the other executables.

Users should also note that the environment variable ECCODES_BUFR_SET_TO_MISSING_IF_OUT_OF_RANGE needs to be set to 1 for ecCodes BUFR encoding. This is done for the user from within the eum2bufr_eccodes and ropp2bufr_eccodes tools, but users may need to set it themselves if using other tools with ecCodes.

Note that the BUFR format, while saving disk space and transmissison bandwidth because of its in-built compression, has certain limitations compared to the netCDF binary format. In particular, orbit data cannot be stored in BUFR with sufficient numerical precision or vertical resolution for most RO applications. Also, being a specialist format designed for operational use by NWP centres, it contains only a sub-set of all the possible parameters defined in the ROPP netCDF files; low–level 'raw' data like excess phases, signal amplitudes (or Signal–to–Noise ratios), for instance, are not present in BUFR.

Users of the ecCodes version of these programs wishing to write Abbreviated Routing Headers (ARHs) to BUFR files will need to use the script `ropp_io/tools/gtsheaders_bufr.pl`, because the ecCodes library cannot currently encode these headers. It is called as follows:

```
> perl gtsheaders_bufr.pl [--iph] infile outfile T1T2A1A2ii CCCC [YYGGgg]
```

where

- `T1T2A1A2` is the product identifier (excluding the sequence number) (for RO data `T1=I`, `T2=U` or `E`, `A1=T` or `G`, `A2=[A-L]` or `X`, and `ii=14` or `01`);
- `CCCC` is the originating centre (ICAO location indicator);
- `YYGGgg` is optional day-of-month/hour/minute (otherwise script uses Section 1 time);
- `--iph` includes an Internet Protocol Header in the BUFR file.

See Sec 5 of (ROM SAF, 2021) for details. If `ropp2bufr_eccodes` (or `eum2bufr_eccodes`) is called with the `-g` or `-gi` options, which request these GTS data headers (the latter with an IPH), the code will echo the call to `gtsheaders_bufr.pl` that the user must make to wrap the BUFR message in them.

3.6.3 ucar2ropp

This tool converts the UCAR '`atmPhs`', '`atmPrf`', '`sonPrf`', '`ecmPrf`', '`ncpPrf`' or '`gfsPrf`' netCDF files to standard ROPP netCDF format. Summary help information can be displayed by:

```
> ucar2ropp -h
```

For example, to convert a UCAR format `atmPrf` file containing Level1b and Level2 data to ROPP netCDF (e.g. for use as input to a 1D-Var routine):

```
> ucar2ropp <atmPrf_filename.nc> -o <outputROPP_filename.nc>
```

Similarly, to convert a UCAR format `atmPhs` file containing Level1a data to ROPP netCDF (e.g. for use as input to the `ropp_pp` module routines):

```
> ucar2ropp <atmPhs_filename.nc> -o <outputROPP_filename.nc>
```

See the program's summary help, man pages and I/O Reference Manuals for more information on usage.

3.6.4 gfz2ropp

This tool converts GFZ native text files — the '`.dat`' and '`.dsc`' pair to standard ROPP netCDF format. Summary help information can be displayed by:

```
> gfz2ropp -h
```

For example, to convert a GFZ format pair of files containing Level1b and Level2 data to ROPP netCDF (e.g. for use as input to a 1D-Var routine):

```
> gfz2ropp <inputGFZ_filename.dat> -o <outputROPP_filename.nc>
```

It is assumed that a companion '.dsc' file with the same name exists in the same directory as the '.dat' file specified on the command line. See the program's summary help, man pages and I/O Reference Manuals for more information on usage.

3.6.5 grib2bgrasc

If a supported ECMWF GRIB_API library, or the ECMWF ecCodes library, is available on the user's computer system (see Appendix A), the program grib2bgrasc will be available as part of ropp_io. This program reads gridded data in ECMWF GRIB format (edition 1 or 2, on 60, 91 or 137 levels), and outputs a profile of background fields, horizontally interpolated to the user-specified latitude and longitude, as a Fortran namelist and as an ROPP-format netCDF file. The namelist can be converted to ROPP format by the ropp_io sister program bgrasc2ropp, which may be useful for users wishing to generate background profiles 'by hand'. Thus, for example, to extract a background profile of RO data at (lon, lat):

```
> grib2bgrasc <ifile.grib> -lat <lat> -lon <lon>  -o <ofile.nml>
```

In this case the ROPP-format netCDF file would be called <ofile.nc>.

At present the code can only extract a single profile at a time. Users wishing to extract a 2D slice of background data would probably find it easier to concatenate the netCDF output from multiple calls to grib2bgrasc, using, for example, the NCO tool ncecat.

The GRIB files must be gridded at a uniform longitudinal and and latitudinal grid, and must contain {temperature (K), specific humidity (kg/kg), ln(surface pressure (Pa)), surface geopotential (m^2/s^2)}. Regularly gridded GRIB files may be downloaded from the ECMWF data portal at http://www.ecmwf.int/products/d
The surface geopotential is not routinely provided with analysis or forecast data, but it can be found in reanalysis products, for example. To handle this complication, grib2bgrasc allows the user to specify an auxiliary file which contains it, thus:

```
> grib2bgrasc <ifile.grib> -z <ifilez.grib> -lat <lat> -lon <lon> -o <ofile.nml>
```

Very coarse (4°) resolution GRIB datasets are provided with the ROPP distribution, for testing purposes. Their contents and structure may be investigated by such tools as grib_dump and grib_ls, which are built as part of the standard GRIB_API, and which may be found in $ROPP_ROOT/<compiler>/bin. The program has been succesfully tested on datasets as fine as 1/4° resolution.

grib2bgrasc also attempts to calculate the undulation (see ROPP_1DVAR User Guide (2022)) provided that the environment variables GEOPOT_COEF and GEOPOT_CORR, which specify files that define the shape of the EGM-96 geoid, are specified. If the ropp_pp module has been downloaded, suitable files are ropp_pp/data/egm96.dat and ropp_pp/data/corrcoef.dat for GEOPOT_COEF and GEOPOT_CORR respectively. Alternatively, the user may specify the undulation directly by using the -u option. The user

may also choose to specify the radius of curvature of the tangent plane,the (nominal) azimuth and the centre of curvature at the command line. Alternatively, these quantities can be read from an auxiliary ROPP file.

Linear time interpolation between the fields in two GRIB files is also supported. In this case the desired date and time need to be specified, thus:

```
> grib2bgrasc <ifile1.grib> <ifile2.grib> -lat <lat> -lon <lon> \
>             -date <date> -time <time> -o <ofile.nml>
```

For details on the usage of `grib2bgrasc`, consult the respective Unix manual pages or the `ropp_io` Reference Manual.

3.6.6 bgrasc2ropp

The program `bgrasc2ropp` converts ascii data (in the form of a Fortran namelist) into an ROPP-formatted netCDF file. This data might, for instance, be generated by the `ropp_io` sister program `grib2bgrasc`. The test file `ropp_io/data/hy_20121001000000_T+0.nml_ref`, which is included in the ROPP distribution, provides an example. For reference, its contents are:

```
&BGR_PROFILE
YEAR    =        2012,
MON     =          10,
DAY     =           1,
HOUR    =           0,
MIN     =           0,
SEC     =           0,
TLEAD   =  -99999000.0000000     ,
LON     =  -3.53000000000000     ,
LAT     =   50.7200000000000     ,
UND     =   51.9751003650887     ,
ROC     =  -99999000.0000000     ,
COC     = 3*-99999000.0000000       ,
AZI     =  -99999000.0000000     ,
Z0      =   28.6550605568034     ,
P0      =   99925.9737444862     ,
NLEVS   =          91,
P       =   99807.5690073690     ,   99521.2795208724, ... 109*-99999000.0000000       ,
T       =   284.218981518555     ,   284.194775048828, ... 109*-99999000.0000000       ,
Q       =   8.27091598888801     ,   8.18340651465114, ... 109*-99999000.0000000       ,
Z       =   38.5707811373220     ,   62.5898839668868, ... 109*-99999000.0000000       ,
AK      =  0.000000000000000E+000,  3.160000080242753E-003, ...
           9.999999999999999E-033, 109*-99999000.0000000       ,
BK      =   1.00000000000000     ,  0.997630119323730     , ...
           34*0.000000000000000E+000   , 109*-99999000.0000000
```

/

As can be seen, the number of levels is currently fixed at 200, but dim_lev2b in the output netCDF file is truncated to NLEVS, which equals 91 here. Note the use of the usual ROPP missing data value of ropp_MDFV = -99999000.0000000 in this namelist.

Thus, to convert a background profile of RO data in namelist format to ROPP netCDF format:

```
> bgrasc2ropp <ifile.nml>  -o <ofile.nc>
```

For details on the usage of bgrasc2ropp, consult the respective Unix manual pages or the ropp_io Reference Manual.

3.6.7 ieec2ropp

The program ieec2ropp converts slantwise TEC data produced by IEEC into an ROPP-formatted netCDF file suitable for use as an observational input file by the 1dvar retrieval tool ropp_1dvar_dbangle. The ascii data have the format:

```
       iyyobs idoyobs tsecdayobs/3600.d0 rec iprn sele0 rrec/1.d+3  rarec . . .
(e.g.)   11     261         10.37       1220  13 -26.80   7184.51  199.27 . . .
          .      .            .           .    .    .         .        .   . . .
```

The full list of columns of variables is:

```
iyyobs = last two digits of year
idoyobs = day of year
tsecdayobs/3600.d0 = GPS time in hours
rec = Receiver Id
iprn = Transmitter Id
sele0 = LOS elevation above a local spherical Earth in deg
rrec/1.d+3 = LEO radius in km
rarec = Receiver right ascension in deg
xlatrec = Receiver latitude in deg
rsat/1.d+3 = Transmitter geocentric distance in km
rasat = Transmitter right ascension in deg
xlatsat = Transmitter latitude in deg
xli = Geometry-free carrier phase combination LI=L1-L2 in m
```

If the -e option is employed, ieec2ropp can process IEEC datafiles that have been post-processed by ECMWF to generate an ascii file in a similar format to the above, but with the following columns:

```
iyyobs = last two digits of year
idoyobs = day of year
tsecdayobs/3600.d0 = GPS time in hours
sele0 = LOS elevation above a local spherical Earth in deg
rrec/1.d+3 = LEO radius in km
rarec = Receiver right ascension in deg
```

```
xlatrec = Receiver latitude in deg
rsat/1.d+3 = Transmitter geocentric distance in km
rasat = Transmitter right ascension in deg
xlatsat = Transmitter latitude in deg
dbangle = L2 bending angle - L1 bending angle in rad
impact = impact parameter in km (based on satellite positions)
impact2 = an unused estimate of impact parameter in km (based on sele0)
```

dbangle in the second type of file is given in terms of the variables in the first file by dxli/dimpact, where impact (also present in the second type of file) is the shortest distance between the centre of the earth and the straight line between the two satellites. impact2 is another estimate of impact parameter, given by rrec*COS(sele0 in rad).

Thus, to convert a background profile of IEEC data in original ascii format to ROPP netCDF format:

```
> ieec2ropp <ifile.dat>  --no-ranchk  -o <ofile.nc>
```

For details on the usage of ieec2ropp, consult the respective Unix manual page or the ropp_io Reference Manual.

3.6.8 eum2ropp, eum2bufr_ecbufr and eum2bufr_eccodes

If the netCDF4/HDF5 library is available on the user's computer system (see Appendix A), the program eum2ropp will be available as part of ropp_io. This program reads 'EUMETSAT-formatted' RO data, which exploit the netCDF4 feature of data 'groups', and converts them into standard ROPP format. An example 'EUMETSAT-formatted' netCDF4 data file is provided with the distribution. (It is used to test these tools.) For example, to convert EUMETSAT-formatted RO data to standard ROPP format:

```
> eum2ropp <input_filename.n4>  -o <output_filename.nc>
```

If the -q option is used with eum2ropp or eum2ropp_eccodes, ionospheric data (only) is extracted from post-2020 EUMETSAT Level 1 data files. In this case, the --no-ranchk option to eum2ropp(_ecodes) is needed to bypass the range-checking of impact parameters which, for ionospheric data, extend far beyond the default range of $(6.2 - 6.6) \times 10^6$ m (see Table 3.3).

For details on the usage of eum2ropp, consult the respective Unix manual pages or the ropp_io Reference Manual.

If, in addition, either the ECMWF BUFR library or the ECMWF ecCodes library are available (see Appendix A again), the executables eum2bufr_eccodes or eum2bufr_eccodes, respectively, will be built. eum2bufr is now a soft link to (in order of preference) eum2bufr_eccodes or eum2bufr_ecbufr. eum2bufr can be called with the -v switch to check which library it was built on.

Both programs read EUMETSAT-formatted RO data and convert it to BUFR format, like this:

```
> eum2bufr <input_filename.n4>  -o <output_filename.bufr>
```

or

```
> eum2bufr_eccodes <input_filename.n4>  -o <output_filename.bufr>
```

(etc) if the user wishes to specify the executable directly.

Users should also note that the environment variable ECCODES_BUFR_SET_TO_MISSING_IF_OUT_OF_RANGE needs to be set to 1 for ecCodes BUFR encoding. This is done for the user from within the eum2bufr_eccodes and ropp2bufr_eccodes tools, but users may need to set it themselves if using other tools with ecCodes.

For details on the usage of eum2bufr consult the Unix manual page or the ropp_io Reference Manual.

RO Level 1 Product Format and BUFR

The mapping between EPS RO Level 1 data variables and WMO's BUFR format for RO measurements is given below. Full specifications of the EUMETSAT's EPS RO Level 1 Product Format appear in (Marquardt, 2016). The full description of the RO BUFR format is outside the scope of this document; it is assumed that the reader is familiar with the details of the RO BUFR format as defined in (ROM SAF, 2021).

BUFR Section 1 (Identification)

BUFR section 1 is filled with metadata as described in the RO-BUFR specification (ROM SAF, 2021) using Edition 4 messages. The time information "most typical for [the] BUFR message content" contained in octet numbers 16–17 (year) and 18–22 (month, day, hour, minute and second) are derived from the georeferencing time, i.e. from the variables utc_georef_absdate and utc_georef_absdate in the /data/occultation group.

BUFR Section 2 (Optional Data)

There is no Section 2 in RO BUFR products.

BUFR Section 3 (Data Description)

Section 3 is to be set dynamically from the number of profiles (usually 1 in a single BUFR message) and the message size.

BUFR Section 4 (Data Template)

Quality information is stored in a single 16-bit data field (octet number 13), where the detailed meaning of each flag is defined in Table 8 of (ROM SAF, 2021). The mapping between BUFR and EPS-SG RO Level 1 product quality flags is described in Table 3.11 below.

Note that values stored in BUFR products match the BUFR conventions, although in some cases this may require a translation from the logical values used in the EPS-SG RO Level 1 data format. For

Bit	Description	Variable
1	Nominal / non-nominal quality	/quality/overall_quality_ok
2	NRT / Offline product	/environment
3	Descending / ascending occultation	/data/occultation/occultation_type
4	Excess phase processing (non-) nominal	/quality/cl_snr_[ca|p1|p2]_ok
5	Bending angle processing (non-) nominal	/quality/thinned_ok
6	Refractivity processing (non-) nominal	*Unused*
7	Meteorological processing (non-) nominal	*Unused*
8	Closed / open loop data only / included	/quality/ol_data_used
9	(No) Surface reflections detected	*Unused*
10	L2P / L2C GPS signals used	*Unused*
11–13	Reserved	*Unused*
14	Background profile (non-) nominal	*Unused*
15	Retrieved / background profile	*Unused*

Table 3.11: Mapping between BUFR Section 4 quality flags for RO and EPS-SG RO L1 data format quality flags.

bit 2, for example, the global attribute `environment` may exhibit values of "Operational", "Validation", "Development", "Offline","Integration & Verification" or "Support". Values of "Operational" and "Validation" will be mapped to "NRT product", while the remaining ones will be mapped to "Offline product". On the other hand, the excess phase processing flag is calculated as the logical 'and' of the /quality_snr_ca_ok, /quality_snr_p1_ok, and /quality_snr_p2_ok flags in the EPS-SG RO Level 1 product.

Table 3.12 links data fields as used in BUFR Section 4 entries (see Table 5 of (ROM SAF, 2021)) to the corresponding variables in the netCDF profile data format. We finally note that BUFR products generated by EUMETSAT do not contain any "Step 2a", "Level2a" or "Level2b" data.

Octet	Variable(s)	Remarks
Nominal Reporting Time		
7–12	`/data/occultation/utc_georef_absdate` `/data/occultation/utc_georef_abstime`	Assumes (IROWG, 2015) is implemented
RO Summary Quality Information		
13	*various*	See Table 3.11
14	`/quality/overall_quality_ok`	100% if true, 0% otherwise
LEO & GNSS POD – Location of Platform		
15–17	`/data/occultation/position_rec_fixed`	Antenna phase centre positions...
18–20	`/data/occultation/velocity_rec`	...and velocities at georeferencing time
21	`/data/transmitter/satellite/satellite_prn`	First letter, e.g. `Gxx` → 401 (for GPS)
22	`/data/transmitter/satellite/satellite_prn`	Integer part, e.g. 1 → 1 (for PRN G01)
23–25	`/data/occultation/position_gns_fixed`	Positions...
26–28	`/data/occultation/velocity_gns`	...and velocities at georeferencing time
Local Earth Parameters		
29	*always zero*	Assumes (IROWG, 2015) is implemented
30	`/data/occultation/latitude`	Valid at georeferencing time
31	`/data/occultation/longitude`	*ditto*
32–34	`/data/occultation/r_curve_centre_fixed`	*ditto*
35	`/data/occultation/r_curve`	*ditto*
36	`/data/occultation/azimuth_north`	*ditto*
37	`/data/occultation/undulation`	*ditto*
RO Step 1b Data		
38	`len(/data/level_1b/thinned/impact)`	Fixed number of levels; same for all products
39	`/data/level_1b/thinned/lat_tp`	
40	`/data/level_1b/thinned/lon_tp`	
41	`/data/level_1b/thinned/azimuth_tp`	
42	3	For ionospheric corrected, L1, and L2 data
43	0 or nominal carrier frequencies (Hz)	0 for ionospheric corrected bending angles
44	`/data/level_1b/thinned/impact`	

45	`/data/level_1b/thinned/bangle`
	`/data/level_1b/thinned/bangle_ca`
	`/data/level_1b/thinned/bangle_p2`
		L1/L2 based, TBD for L1/L5

46–48	*Currently unused*	

Table 3.12: Mapping between BUFR Section 4 data fields and EPS-SG RO Level 1 data format variables.

3.7 ROPP Thinner

The ropp_io module contains several routines which can be used to thin Level 1b, Level2a and Level2b data. Thinning aims to reduce the amount of data without reducing the information content. Details on the ROPP thinning algorithms are provided by (GRAS SAF, 2009) and (GRAS SAF, 2007).

ROPP supports several different thinning algorithms. For details see GRAS SAF, 2007.

NONE	: no thinning.
SAMPLE	: Uniform sampling (select or reject 1-in-N), up to a specified maximum number of thinned levels.
LIN	: Linear interpolation to fixed levels (no smoothing).
LOG	: Logarithmic interpolation to fixed levels (no smoothing).
SGLIN	: Savitzky-Golay smoothing filter with linear interpolation to fixed number of thinned levels.
SGLOG	: Savitzky-Golay smoothing filter with log interpolation to fixed number of thinned levels.
ASGLIN	: Adaptive S-G smoothing filter with linear interpolation to fixed number of thinned levels.
ASGLOG	: Adaptive S-G smoothing filter with log interpolation to fixed number of thinned levels.

The thinner may be called from user programs as

```
USE ropp_io
TYPE(ROprof) :: ro_data
  ...
CALL ropp_io_thin(ro_data, ThinFile)
```

where ThinFile is the name of a thinning control file. This is a plain-text file which should be formatted as follows.

```
Title=<title>       ! free-text description
Method=<method>     ! Thinning method selected from options above
Nlevels=<nlev>      ! Maximum number of levels for output
Hlevels=            ! Required set of fixed levels as impact heights
  <level1>
  <level2>
  ...
  <levelnlev>
```

The impact heights (Hlevels) on which data are to be thinned are not used by the SAMPLE method. Otherwise, the height values should increase monotonically.

The ROPP thinner may be optionally called by the ropp2ropp, ucar2ropp and ropp2bufr tools. This is achieved by specifying the ThinFile with the '-p' command line argument.

```
> ropp2ropp input_data.nc -o thinned_data.nc -p ThinFile.dat
```

Note that when using one of the log-based thinners, only those variable which vary roughly exponentially

with height (bending angle, refractivity, humidity and pressure) are interpolated this way. All other variables are interpolated linearly.

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.

As a courtesy to users, all the third-party dependency packages used by ROPP can be downloaded from the ROM SAF ROPP download page (https://www.romsaf.org/ropp/files.php). The versions provided there are known to work with the associated version of ROPP.

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` 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`).

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

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

Note that a small change has been made to the ecCodes tarball supplied with ROPP to suppress the warning message that is produced each time a missing data indicator is set. This change can be made to a user's own copy of the ecCodes library by using the patch provided at `ropp_io/tools/eccodes_patch`.

All libraries generate essentially identical data when decoded (there may be insignificant round-off differences due to use of single– vs. double–precision interfaces). The MetDB library can be built with

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

the ECMWF BUFR library can be be built with

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

and the ECMWF ecCodes library can 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 BUFR library or ecCodes 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 the 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.

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

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):

```
> 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 <compiler>
> 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/eccodes grib/eccodes 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-11.3.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. 11.3), as in `ropp_utils-11.3.tar.gz`. If GNU tar is available (as on Linux systems), gzipped tar files can be unzipped with

```
> tar -xvzf ropp-11.3.tar.gz
```

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

```
> gunzip -c ropp-11.3.tar.gz | tar -xv
```

In all cases, a new subdirectory named (in the above example) `ropp-11.3` 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 (v16 and v17); NAG's nagfor (v6.1); Portland Group's pgf95 (v16); GNU gfortran (v4.8.5); Cray's ftn (v8.3.4). 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. The eum2ropp tool, which converts 'EUMETSAT-format' data into standard ROPP format netCDF files, can be tested on a Metop-A dataset with the command

```
> make test_eum
```

and on a Sentinel-6 dataset with the command

```
> make test_sx6
```

If a supported BUFR library is available, the tests subdirectory will also contain a test script for the programs ropp2bufr, bufr2ropp and eum2bufr, which convert ROPP data files (and EUMETSAT data files) to and from BUFR format data files. (With effect from ROPP-11.3, these programs are softlinks to library-dependent executables, e.g. ropp2bufr -> ropp2bufr_eccodes, if the ecCodes BUFR library is available. See Section 3 of this document for details.) Issuing the command

```
> make test_bufr
```

will run a number of file conversions, and numerically compare the results to reference files which are included in the ROPP distribution. For BUFR encoder tests, the resulting BUFR files are passed through a standalone (i.e., independent of any BUFR library) BUFR-to-ROPP encoder, ropp_io/tests/robufr2ropp, the output of which is compared to a reference netCDF file. 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
```

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.

For each test, the fields in the resulting file are compared to those in reference files, and if they agree within certain tolerances then the test is deemed to have passed. This is handled by the pro-

grams `ropp_io_compare` and `ropp_io_summary`, with the detailed comparisons being carried out by `ropp_io/ropp/ropp_io_fields_compare.f90`. If all tests are made (by means of `make test`), a summary table of results is written to the screen. If all the BUFR libraries are available, and can be built on successfully, then the table should look like this:

```
*************** SUMMARY OF ROPP_IO TEST RESULTS ***************************
-----------------------------------------------------------------------------
|        Test name    |              Description       |    Run? |  PASS? |
-----------------------------------------------------------------------------
|            t_netcdf |              IO (netCDF basic) |     Run |  PASS  |
|       t_ropp2ropp_i |      IO (missing/invalid data) |     Run |  PASS  |
|       t_ropp2ropp_v |                IO (valid data) |     Run |  PASS  |
|       t_ropp2ropp_n |               IO (netCDF data) |     Run |  PASS  |
|      t_ropp2ropp_m1 |           IO (single -> multi) |     Run |  PASS  |
|      t_ropp2ropp_m2 |           IO (multi -> single) |     Run |  PASS  |
|      t_ropp2ropp_2d |                   IO (2D data) |     Run |  PASS  |
|  t_ropp2bufr_mobufr |          IO (ropp2bufr_mobufr) |     Run |  PASS  |
|  t_bufr2ropp_mobufr |          IO (bufr2ropp_mobufr) |     Run |  PASS  |
|  t_ropp2bufr_ecbufr |          IO (ropp2bufr_ecbufr) |     Run |  PASS  |
|  t_bufr2ropp_ecbufr |          IO (bufr2ropp_ecbufr) |     Run |  PASS  |
|   t_eum2bufr_ecbufr |           IO (eum2bufr_ecbufr) |     Run |  PASS  |
| t_ropp2bufr_eccodes |         IO (ropp2bufr_eccodes) |     Run |  PASS  |
| t_bufr2ropp_eccodes |         IO (bufr2ropp_eccodes) |     Run |  PASS  |
|  t_eum2bufr_eccodes |          IO (eum2bufr_eccodes) |     Run |  PASS  |
|        t_gfz2ropp_1 |              IO (GFZ NRT data) |     Run |  PASS  |
|        t_gfz2ropp_2 |               IO (GFZ PD data) |     Run |  PASS  |
|       t_ucar2ropp_1 |          IO (UCAR atmPrf data) |     Run |  PASS  |
|       t_ucar2ropp_2 |          IO (UCAR atmPhs data) |     Run |  PASS  |
|          t_eum2ropp |         IO (EUM Metop -> ROPP) |     Run |  PASS  |
|          t_sx62ropp |           IO (EUM SX6 -> ROPP) |     Run |  PASS  |
|       t_grib2bgrasc |             IO (GRIB -> ASCII) |     Run |  PASS  |
|       t_bgrasc2ropp |             IO (ASCII -> ROPP) |     Run |  PASS  |
-----------------------------------------------------------------------------
```

Each test produces its own log file, which should help track down problems for tests that `FAIL`. Similar procedures (and results) apply to the automatic testing that is undertaken when `make test` is run in the `test` directory of the other modules, as described below.

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. As for the `ropp_io` module, 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. As for the `ropp_io` module, 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 pre-calculated 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. As for the `ropp_io` module, 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. As for the `ropp_io` module, 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

The `ropp_io` module provides support for a generic data format for radio occultation data. Routines are provided for flexible netCDF I/O of RO data via simple interfaces with a file management conversion tool, plus data thinning. Application tools include a BUFR encoder and decoder and conversion from UCAR and GFZ format data files to ROPP netCDF and a test data generator. A tool is provided to extract background profiles from ECMWF GRIB format gridded datasets; another tool converts the ascii output of this into ROPP format netCDF files.

Files listed in bold correspond to executable stand-alone tools. These call lower-level routines. In order to build this module the required packages must be first installed. Routines having additional dependencies on other packages or ROPP modules are listed with the required modules given in brackets. If the additional (optional) packages are not recognised by the configure script, only the core functions will be compiled and installed.

- Required packages: `ropp_utils`, `netcdf`.
- Optional packages: `BUFR` (MetDB, ECMWF BUFR, ECMWF ecCodes); `GRIB_API or ecCodes` (ECMWF); `netCDF4/HDF5`.
- Stand-alone tools and test programs († — requires MetDB `BUFR`, ECMWF `BUFR` or ECMWF `ecCodes`; ‡ — requires ECMWF `GRIB_API` or `ecCodes` ; § — requires `netCDF4/HDF5`):

tools/

gfz2ropp.f90
ropp2ropp.f90
test2ropp.f90
ucar2ropp.f90
bufr2ropp_mo.f90 †
bufr2ropp_ec.f90 †
bufr2ropp_eccodes.f90 †
ropp2bufr_mo.f90 †
ropp2bufr_ec.f90 †
ropp2bufr_eccodes.f90 †
grib2bgrasc.f90 ‡
bgrasc2ropp.f90
eum2ropp.f90 §
eum2bufr_ec.f90 §†
eum2bufr_eccodes.f90 §†
ieec2ropp.f90

tests/

robufr2ropp.f90
ropp_io_compare.f90
ropp_io_summary.f90
t_netcdf.f90
nml_diff.f90
t_ropp2ropp.sh
t_roppthin.sh
t_gfz2ropp_1.sh
t_gfz2ropp_2.sh
t_ucar2ropp_1.sh
t_ucar2ropp_2.sh
t_ropp2bufr_mobufr.sh †
t_ropp2bufr_ecbufr.sh †
t_ropp2bufr_eccodes.sh †
t_bufr2ropp_mobufr.sh †
t_bufr2ropp_ecbufr.sh †
t_bufr2ropp_eccodes.sh †
t_grib2bgrasc.sh ‡
t_bgrasc2ropp.sh
t_eum2ropp.sh §
t_sx62ropp.sh §
t_eum2bufr_ecbufr.sh §†
t_eum2bufr_eccodes.sh §†

- Integrated code

```
bufr/
      bufr2ropp_mod.f90
      bufrutils.f90
      convertcodes.f90
      gtshdrs.f90
      ropp2bufr_mod.f90

ropp/
      ropp_io.f90
      ropp_io_addvar.f90
      ropp_io_ascend.f90
      ropp_io_assign.f90
      ropp_io_fields_compare.f90
      ropp_io_free.f90
      ropp_io_init.f90
      ropp_io_isinrange.f90
      ropp_io_nrec.f90
      ropp_io_occid.f90
      ropp_io_rangecheck.f90
      ropp_io_read.f90
      ropp_io_read_ncdf_get.f90
      ropp_io_shrink.f90
      ropp_io_success.f90
      ropp_io_test.f90
      ropp_io_types.f90
      ropp_io_vlist_print.f90
      ropp_io_vlist_read.f90
      ropp_io_vlist_size.f90
      ropp_io_write.f90
      ropp_io_write_ncdf_def.f90
      ropp_io_write_ncdf_put.f90

ncdf/
      is_netcdf.f90
      ncdf.f90
      ncdf_close.f90
      ncdf_create.f90
      ncdf_date_and_time.f90
      ncdf_datmode.f90
      ncdf_defdim.f90
      ncdf_defmode.f90
      ncdf_defvar.f90
      ncdf_error_handler.f90
      ncdf_getatt.f90
      ncdf_getatt_alloc.f90
      ncdf_getgroupid_n3.f90
      ncdf_getgroupid_n4.f90
      ncdf_getnrec.f90
      ncdf_getshape.f90
      ncdf_getsize.f90
      ncdf_getvar.f90
      ncdf_getvar_alloc.f90
      ncdf_isatt.f90
      ncdf_isvar.f90
      ncdf_open.f90
      ncdf_putatt.f90
      ncdf_putvar.f90
      ncdf_renvar.f90
      ncdf_save.f90
      ncdf_sync.f90
      nf90_get_att_string.f90

ecmwf/
      ropp_io_ecmwf.f90

thin/
      ropp_io_thin.f90
      ropp_io_thin_fixed.f90
      ropp_io_thin_select.f90
      ropp_io_thin_sg.f90
      ropp_io_thin_skip.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
. . .%bg_ne	Background electron density
. . .%bg_ne_sigma	Error in background electron density
. . .%res_ne	Analysis electron density
. . .%res_ne_sigma	Error in analysis electron density
. . .%VTEC_bg	VTEC of background electron density
. . .%VTEC_an	VTEC of analysis electron density

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: General 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/DMI/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/DMI/REP/GSR/012	Assimilation of GPSRO Data in the ECMWF ERA-Interim Reanalysis
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

(To be continued)

Title	Reference	Description
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/DMI/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/DMI/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

(To be continued)

Title	Reference	Description
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/ECMWF/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
An initial assessment of the quality of RO data from PAZ	SAF/ROM/METO/REP/RSR/036	PAZ quality assessment
6th ROM SAF User Workshop	SAF/ROM/METO/REP/RSR/037	ROM SAF–IROWG 2019 report
An initial assessment of the quality of RO data from COSMIC-2	SAF/ROM/METO/REP/RSR/038	COSMIC-2 quality assessment
Impacts of RO mission differences on trends in multi-mission data records	SAF/ROM/METO/REP/RSR/039	RO mission CDR differences
Anomalous GRAS radio occultations	SAF/ROM/METO/REP/RSR/040	Anomalous occultations
Assessment of sensitivity of the ROM SAF 1D-Var solutions to various error covariance choices	SAF/ROM/DMI/REP/RSR/041	Sensitivity to error covariances
A one-dimensional variational ionospheric retrieval for truncated GNSS Radio Occultation measurements	SAF/ROM/METO/REP/RSR/042	Ionospheric 1dvar

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 alphabetically in Table E.1. The ROM SAF extends its sincere gratitude for their efforts.

ROPP Authors

Name	Current institute	Contribution
Carlo Buontempo	Met Office	Savitzky-Golay thinner code.
Chris Burrows	ECMWF	2nd ROPP Test Manager. Test folder developments, improved FM vertical interpolation scheme.
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, implementation of VaryChap f2–f2 FM and 1DVAR code
Axel von Engeln	EUMETSAT	Author of original Test Folder system and of EUMETSAT-formatted RO data reader.
Hans Gleisner	DMI	Elements of ropp_pp, prototype GRIB2 reader, ec{i/f}2ec{i/f} code.
Michael Gorbunov	Russian Academy of Sciences	Original pre-processor code.
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, prototype VaryChap f2–f2 FM and 1DVAR code.
Helge Jønch-Sørensen	DMI	BAROCLIM code.
Kjartan Kinch	DMI	Elements of ropp_pp.
Kent Bækgaard Lauritsen	DMI	Code reviews; liaison with EUMETSAT (licences, beta tester contracts).
Huw Lewis	Met Office	1st ROPP Development Manager, FM and 1D–VAR extensions. PP module.
Owen Lewis	Met Office	BUFR developments.
Christian Marquardt	EUMETSAT	Author of majority of ROPP-1 code in UTILS, IO, FM and 1DVAR modules, and much personal, pre-existing software.
Dave Offiler	Met Office	ROPP Project Manager, IO application code and IO extensions, BUFR format/template.
Michael Rennie	ECMWF	1st ROPP Test Manager. Test folder developments.
Barbara Scherllin-Pirscher	Wegener Center	BAROCLIM (3) dataset for statistical optimisation.
Torsten Schmidt	GFZ	Guidance on tropopause height diagnostics.
Stig Syndergaard	DMI	Original spectral version of MSIS model (expansion in spherical harmonics and Chebychev polynomials), PP module developments.
Francis Warrick	Met Office	Implementation of ecCodes lib; ROPP devt and testing.
Feiqin Xie	Texas A & M	Suggested boundary layer height diagnostic algorithms.

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

References

GRAS SAF (2007). *Mono-dimensional data thinning for GPS Radio Occultations*. SAF/GRAS/METO/REP/GSR/001.

GRAS SAF (2009). *ROPP Thinner Algorithm*. SAF/GRAS/METO/REP/GSR/008.

IROWG (June 3, 2015). *Recommendations of the IROWG-4 action group on the homogenization and evolution of the BUFR file specification for GNSS Radio Occultations*. IROWG/MM/2015, Version 1.4. URL: http://irowg.org/wpcms/wp-content/uploads/2015/07/IROWG4-BUFR_action_group_20150603_summary_final.doc.

Marquardt, C. (Dec. 13, 2016). *Radio Occultation Level 1 Product Format Specification*. EUM/TSS/SPE/16/817861, Issue v1A. Technical Note.

ROM SAF (2019). *Sensitivity of some RO measurements to the shape of the ionospheric electron density profile*. SAF/ROM/METO/REP/RSR/031.

ROM SAF (Dec. 31, 2021). *WMO FM94 (BUFR) Specification for Radio Occultation Data*. SAF/ROM/METO/FMT/BUFR/001, Version 2.7.

ROM SAF (Feb. 28, 2022a). *The Radio Occultation Processing Package (ROPP) 1D–Var module User Guide*. SAF/ROM/METO/UG/ROPP/007, Version 11.1.

ROM SAF (Feb. 28, 2022b). *The Radio Occultation Processing Package (ROPP) Applications module User Guide*. SAF/ROM/METO/UG/ROPP/005, Version 11.1.

ROM SAF (Feb. 28, 2022c). *The Radio Occultation Processing Package (ROPP) Forward model module User Guide*. SAF/ROM/METO/UG/ROPP/006, Version 11.1.

ROM SAF (Feb. 28, 2022d). *The Radio Occultation Processing Package (ROPP) Input/Output module User Guide*. SAF/ROM/METO/UG/ROPP/002, Version 11.1.

ROM SAF (Feb. 28, 2022e). *The Radio Occultation Processing Package (ROPP) Pre-processor module User Guide*. SAF/ROM/METO/UG/ROPP/004, Version 11.1.

ROM SAF (Feb. 28, 2022f). *The Radio Occultation Processing Package (ROPP) Utilities module User Guide*. SAF/ROM/METO/UG/ROPP/008, Version 11.1.

Unidata (-). *netCDF website*. URL: http://www.unidata.ucar.edu/software/netcdf/index.html.