The EUMETSAT Network of Satellite Application Facilities

ROM SAF

Radio Occultation Meteorology

ROM SAF CDOP-2

The Radio Occultation Processing Package (ROPP) User Guide Part II: Forward model and 1D–Var modules

Version 7.0

31 July 2013

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

Ref: SAF/ROM/METO/UG/ROPP/003

Document Author Table

	Name	Function	Date	Comment
Prepared by:	I. Culverwell	ROM SAF Project Team	31 July 2013	
Reviewed by:	D. Offiler	ROM SAF Project Team	31 July 2013	
Approved by:	K.B. Lauritsen	ROM SAF Project Manager	31 July 2013	

Document Change Record

Issue/Revision	Date	By	Description
Version 0.1	01 Nov 2004	CM	ROPP User Guide: Preliminary release for I–RR
Version 0.7	04 Jun 2005	CM	ROPP User Guide: Intermediate release for CPM.
Version 0.8	17 Oct 2005	DO	ROPP User Guide: First Beta release (v0.8)
Version 0.9	10 Nov 2006	DO	ROPP User Guide: Second Beta release (v0.9)
Version 1.0	01 Mar 2007	DO	ROPP User Guide: First Full release (v1.0)
Version 1.1	01 Mar 2008	DO	ROPP User Guide: Updates to First Full release (v1.1)
Version 1.2	01 Jul 2008	HL	ROPP User Guide: Updates to First Full release (v1.2)
Version 2.0	01 Dec 2008	HL	ROPP User Guide: Second Full release (v2.0)
Version 3.0	01 Jun 2009	HL	ROPP User Guide: Third Full release (v3.0)
Version 4.0	01 Nov 2009	HL	ROPP User Guide: Fourth Full release (v4.0)
Version 4.1	01 Jun 2010	HL	ROPP User Guide: Updates to Fourth Full release (v4.1)
Version 5.0	14 Jun 2011	IC	ROPP User Guide: Fifth Full release (v5.0)
Version 6.0	31 Oct 2011	IC	ROPP User Guide: Sixth Full release (v6.0)
Version 6.1	31 Jan 2013	IC	ROPP User Guide: Updates to Sixth Full release (v6.1)
Version 7.0	31 Jul 2013	IC	ROPP User Guide: Seventh Full release (v7.0)

ROM SAF

The ROM SAF is the EUMETSAT Satellite Application Facility responsible for operational processing of radio occultation data from the Metop satellites. It delivers bending angle, refractivity, temperature, pressure, and humidity profiles in near-real time and offline for NWP and climate users. The offline profiles are further processed into climate products consisting of gridded monthly zonal means of bending angle, refractivity, temperature, humidity, and geopotential heights together with error descriptions.

The ROM SAF also maintains the Radio Occultation Processing Package (ROPP) which contains software modules that will 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, and iii) Met Office in Exeter, United Kingdom. To get access to our products or to read more about the project please go to http://www.romsaf.org.

Intellectual Property Rights

Contents

1 Introduction **1**
1.1 Purpose of this document . . . 1
1.2 ROPP . . . 1
1.3 User documentation . . . 2
References . . . 3

2 Data assimilation **4**
2.1 Variational data assimilation . . . 4
2.2 Forward models . . . 5
2.3 Quality control . . . 6
2.4 1D–Var retrievals . . . 6
2.5 Preprocessing and error characteristics . . . 7
References . . . 7

3 Forward models **9**
3.1 Radio occultation geometry . . . 9
3.2 Bending angle . . . 10
3.3 Refractivity . . . 11
3.4 Summary . . . 11
References . . . 12

4 ROPP one-dimensional forward models **13**
4.1 ROPP forward model tool . . . 14
4.1.1 Implementation . . . 15
4.1.2 Code organisation . . . 15
4.2 Data types . . . 16
4.2.1 Observation vector: `Obs1dRefrac`, `Obs1DBangle` . . . 16
4.2.2 State vector: `State1dFM` . . . 17
4.3 Defining default units: `ropp_fm_set_units` . . . 18
4.4 Defining the state vector: `ropp_fm_roprof2state` . . . 19
4.5 Hybrid sigma vertical levels - ECMWF model type: `ropp_fm_state2state_ecmwf` . . . 20
4.5.1 Vertical grid structure . . . 20
4.6 Geopotential-based vertical levels - Met Office model type: `ropp_fm_state2state_meto` . . . 21
4.6.1 Vertical grid structure . . . 22
4.6.2 Input data . . . 23
4.7 Defining the observation vector: `ropp_fm_roprof2obs` . . . 23

4.7.1 Set observation geopotential height levels set_obs_levels_refrac . . . 25
4.7.2 Set observation impact parameter levels set_obs_levels_bangle . . . 25
4.8 Refractivity forward model ropp_fm_refrac_1d . . . 26
4.8.1 Update variables in State1dFM . . . 26
4.8.2 Compute partial water vapour pressure . . . 26
4.8.3 Compute refractivity . . . 27
4.8.4 Interpolate refractivity to measurement geopotential levels ropp_fm_interpol_log 27
4.8.5 Refractivity forward model gradient . . . 27
4.9 Bending angle forward model ropp_fm_bangle_1d . . . 27
4.9.1 Update variables in State1dFM . . . 28
4.9.2 Compute refractivity profile . . . 28
4.9.3 Compute background impact parameter . . . 28
4.9.4 Calculate bending angles ropp_fm_abel . . . 28
4.9.5 Bending angle forward model gradient . . . 30
4.10 Copy simulated observations to RO structure ropp_fm_obs2roprof . . . 30
4.11 Plotting tools . . . 30
4.12 Test software tools . . . 31
4.12.1 t_fascod . . . 31
4.12.2 t_fascod_tl . . . 31
4.12.3 t_fascod_ad . . . 32
References . . . 33

5 ROPP two-dimensional forward model 34
5.1 Background . . . 34
5.2 Theoretical basis of 2D operator . . . 34
5.3 ROPP 2D forward model tool . . . 35
5.3.1 Implementation . . . 37
5.3.2 Code organisation . . . 37
5.4 Data types . . . 38
5.4.1 Observation vector: Obs1DBangle . . . 38
5.4.2 State vector: State2dFM . . . 38
5.5 Defining the state vector: ropp_fm_roprof2state . . . 39
5.6 Defining the observation vector: ropp_fm_roprof2obs . . . 40
5.6.1 Set observation impact parameter levels set_obs_levels_bangle . . . 40
5.7 Bending angle forward model ropp_fm_bangle_2d . . . 40
5.7.1 Compute partial water vapour pressure . . . 43
5.7.2 Compute refractivity . . . 43
5.7.3 Compute geometric heights and radius values . . . 43
5.7.4 Compute background impact parameter . . . 43
5.7.5 Calculate bending angles ropp_fm_alpha2drk . . . 44
5.7.6 Bending angle forward model gradient . . . 46
5.8 Copy simulated observations to RO structure ropp_fm_obs2roprof . . . 46

5.9 Comparing with a 1D bending angle operator . . . 47
5.10 Writing out data . . . 47
5.11 Plotting tools . . . 47
5.12 Test software tools . . . 47
5.12.1 t_twodop . . . 47
5.12.2 t_twodtl . . . 48
5.12.3 t_twodad . . . 48
References . . . 49

6 ROPP 1D–Var: Refractivity **50**
6.1 ROPP 1D–Var refractivity tool . . . 50
6.1.1 Implementation . . . 50
6.1.2 Code organisation . . . 51
6.2 Defining observation and background errors . . . 54
6.3 Input data . . . 56
6.3.1 Configuration options . . . 57
6.4 Observation data . . . 57
6.4.1 Defining the observation vector: ropp_fm_roprof2obs . . . 57
6.4.2 Defining the observation error covariance matrix: ropp_1dvar_covar_refrac . . . 58
6.5 Background data . . . 59
6.5.1 Defining the state vector: ropp_fm_roprof2state . . . 59
6.5.2 Defining the background error covariance matrix: ropp_1dvar_covar_bg . . . 60
6.6 Quality control . . . 63
6.6.1 Valid observation height range: ropp_qc_cutoff() . . . 64
6.6.2 Generic quality control check: ropp_qc_genqc() . . . 64
6.6.3 Observation minus background check: ropp_qc_OmB() . . . 64
6.6.4 Background quality control check: ropp_qc_bgqc() . . . 65
6.6.5 Probability of gross error (PGE): ropp_qc_pge() . . . 65
6.7 Minimise the cost function . . . 66
6.7.1 Preconditioning . . . 66
6.7.2 Compute the cost function . . . 68
6.7.3 Convergence check . . . 68
6.7.4 Minimisation . . . 69
6.8 Output diagnostics . . . 71
6.9 Output data . . . 72
6.10 Plotting tools . . . 73
References . . . 73

7 ROPP 1D–Var: Bending angle **76**
7.1 ROPP 1D–Var bending angle tool . . . 76
7.1.1 Implementation . . . 76
7.1.2 Code organisation . . . 77

7.2 Defining observation and background errors . . . 80
7.3 Input data . . . 82
7.3.1 Configuration options . . . 83
7.4 Observation data . . . 83
7.4.1 Defining the observation vector: `ropp_fm_roprof2obs` . . . 83
7.4.2 Defining the observation error covariance matrix: `ropp_1dvar_covar_bangle` . . . 84
7.5 Background data . . . 85
7.5.1 Defining the state vector: `ropp_fm_roprof2state` . . . 85
7.5.2 Defining the background error covariance matrix: `ropp_1dvar_covar_bg` . . . 86
7.6 Quality control . . . 89
7.6.1 Valid observation height range: `ropp_qc_cutoff()` . . . 90
7.6.2 Generic quality control check: `ropp_qc_genqc()` . . . 90
7.6.3 Observation minus background check: `ropp_qc_OmB()` . . . 90
7.6.4 Background quality control check: `ropp_qc_bgqc()` . . . 91
7.6.5 Probability of gross error (PGE): `ropp_qc_pge()` . . . 91
7.7 Minimise the cost function . . . 92
7.7.1 Preconditioning . . . 93
7.7.2 Compute the cost function . . . 94
7.7.3 Convergence check . . . 94
7.7.4 Minimisation . . . 95
7.8 Output diagnostics . . . 97
7.9 Output data . . . 98
7.10 Plotting tools . . . 99
References . . . 99

8 ROPP FM and 1D–Var: Including non-ideal gas effects 102
8.1 Background . . . 102
8.2 Effects of non-ideal gas compressibility . . . 102
8.3 Code structure . . . 103
8.4 Running ROPP routines with non-ideal effects included . . . 105
References . . . 105

A `ropp_utils` library 107
A.1 `Missing data values` . . . 107
A.2 `ropp_messages` . . . 107
A.3 `Unitconvert` . . . 108
A.4 `Coordinates` . . . 108
A.5 `Datetime` . . . 108
A.6 `Geodesy` . . . 108
A.7 `Arrays` . . . 109
A.8 `Misc` . . . 109
A.8.1 `typeSizes` . . . 109

B Installing and using ROPP 110
B.1 Software requirements . . . 110
B.2 Software release notes . . . 110
B.3 Third-party packages . . . 110
B.3.1 NetCDF . . . 111
B.3.2 BUFR (optional) . . . 111
B.3.3 GRIB_API (optional) . . . 112
B.3.4 netCDF4/HDF5 (optional) . . . 112
B.3.5 RoboDoc (optional) . . . 113
B.3.6 autoconf and automake (optional) . . . 113
B.4 BUILDPACK script . . . 113
B.5 Building and installing ROPP manually . . . 114
B.5.1 Unpacking . . . 115
B.5.2 Configuring . . . 115
B.5.3 Compiling . . . 116
B.5.4 Installing . . . 116
B.5.5 Cleaning up . . . 117
B.6 Linking . . . 117
B.7 Testing . . . 118
B.7.1 `ropp_utils` . . . 118
B.7.2 `ropp_io` . . . 118
B.7.3 `ropp_pp` . . . 119
B.7.4 `ropp_fm` . . . 120
B.7.5 `ropp_1dvar` . . . 120
B.8 Troubleshooting . . . 120

C `ropp_fm` program files 122

D `ropp_1dVar` program files 124

E ROPP extra data 126
E.1 ropp_io_addvar . . . 126
E.2 PPDiag . . . 127
E.3 ropp_fm_bg2ro . . . 127
E.4 VarDiag . . . 127

F ROPP user documentation 129

G Acronyms and abbreviations 131

H Definitions 134

I Copyrights 135

1 Introduction

1.1 Purpose of this document

This document provides a User Guide for the Forward Model and 1D–Var modules of the Radio Occultation Processing Package (ROPP). The Forward Models are designed to compute refractivity and bending angle profiles from background atmospheric profiles of pressure, temperature and humidity data. The Forward Model output can be compared with radio occultation observations and used as part of a 1D–Var retrieval or in a 3D–Var or 4D–Var NWP data assimilation system. The 1D–Var processing is designed to retrieve atmospheric temperature, moisture and pressure from radio occultation bending angle and refractivity profiles.

The ROPP User Guide Part I (2013a) provides an overview of the generic ROPP data format and software to read and write radio occultation data provided as part of ROPP. The ROPP User Guide Part III (2013b) provides details of the pre-processing software included within ROPP to generate refractivity and bending angle profiles from radio occultation data.

An overview of the ROPP modules and the installation, build and test procedures for ROPP are provided in the appendices. Detailed build and install instructions are contained in the release notes of the individual ROPP software modules.

1.2 ROPP

The aim of ROPP is

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

ROPP is a collection of software modules (provided as source code), supporting data files and documentation, which aids users wishing to assimilate radio occultation data into their NWP models. As far as is practical, the software is generic, in that it can handle any GNSS–LEO configuration radio occultation mission (CHAMP, GRACE, SAC–C, GRAS, COSMIC, etc).

The software is distributed in the form of a source code library written in Fortran 90. ROPP is implemented using Fortran modules and derived types, enabling the use of object oriented techniques such as the overloading of routines. The software is split into several modules. Figure 1.1 illustrates the interrelationships between each module. Users may wish to integrate a subset of ROPP code into their own software applications, individually linking modules to their own code. These users may not require the

complete ROPP distribution package. Alternatively, users may wish to use the executable tools provided as part of each module as stand-alone applications for RO data processing. These users should download the complete ROPP release.

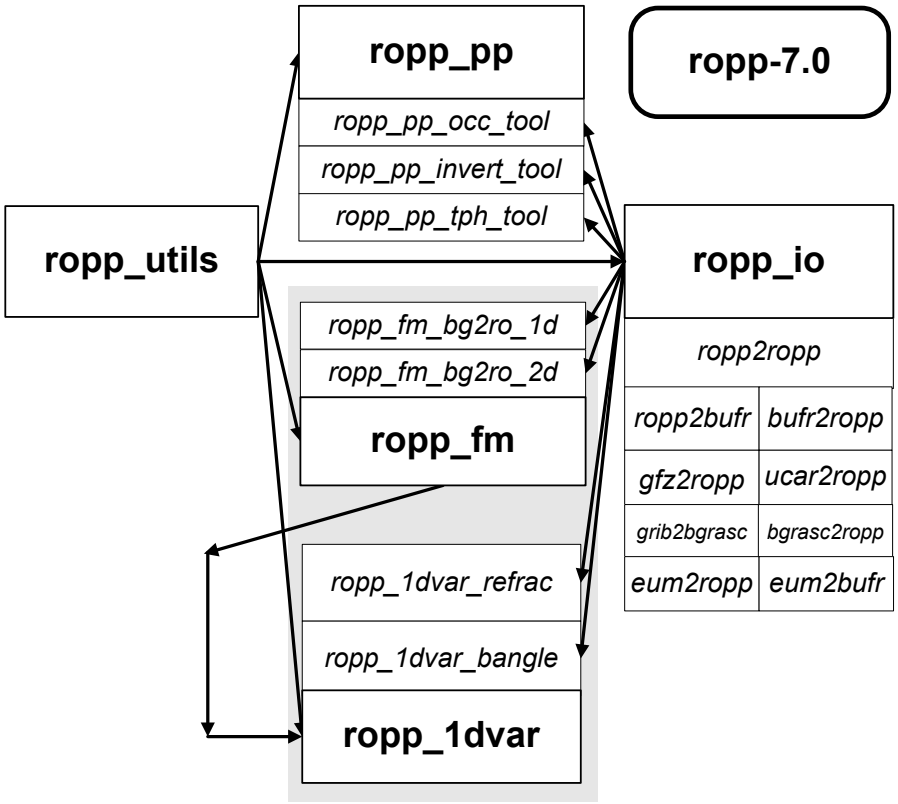

Figure 1.1: ROPP modules and their relationships. Stand-alone applications provided with each module are shown in italics. Connecting lines indicate module dependencies, with arrow heads pointing to the modules or stand-alone tools which require that module.

ROPP contains support for a generic data format for radio occultation data (`ropp_io`), pre-processor routines (`ropp_pp`), one- and two-dimensional forward models (`ropp_fm`), tools for quality control (immersed within the other modules, principally `ropp_1dvar`) and routines for the implementation of 1D–Var retrievals (`ropp_1dvar`). Utility routines used by some or all of the ROPP modules are provided in an additional module (`ropp_utils`). This structure (Figure 1.1) 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 independent, whereas routines in `ropp_1dvar` depend on `ropp_fm`. Sample stand-alone 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.

1.3 User documentation

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

The ROPP distribution website has a Release Notes (html) 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.

This ROPP 1D–Var User Guide provides documentation of the forward model and retrieval software provided with ROPP. An overview on how to install, build and use it, along with a description of the ROPP tools used by other ROPP modules are also provided.

Module-specific user guides for the Input/Output (ROM SAF, 2013a) and Pre-processor (ROM SAF, 2013b) 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 read and write radio occultation data and compute bending angle or refractivity profiles from excess phase.

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 in PDF format from the ROPP section of the ROM SAF web site http://www.romsaf.org. The full user documentation set is listed in Table F.1.

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

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

References

ROM SAF, The Radio Occultation Processing Package (ROPP) User Guide. Part I: Input/Output module, SAF/ROM/METO/UG/ROPP/002, Version 7.0, 2013a.

ROM SAF, The Radio Occultation Processing Package (ROPP) User Guide. Part III: Pre-processor module, SAF/ROM/METO/UG/ROPP/004, Version 7.0, 2013b.

2 Data assimilation

The assimilation of GNSS-RO observations in NWP requires a means to relate the information provided by the measured data to the meteorological variables considered in a NWP model. The ROPP 1D–Var module (`ropp_1dvar`) provided as part of the ROPP software enables the retrieval of temperature, humidity and pressure from profiles of bending angle or refractivity measured by radio occultation.

2.1 Variational data assimilation

The majority of today's operational data assimilation systems are based on variational (Var) methods. For a detailed overview of the theory see Lorenc (1986). Variational techniques provide a framework for the assimilation of meteorological observations of widely different types. By using a forward model (or forward operator) information in the space of model (i.e. NWP forecast) variables can be mapped into that of the observations, and back, in a consistent manner (Eyre, 1997).

The variational approach to data assimilation is usually formulated as a minimisation problem of the cost function

$$J(\mathbf{x}) = \frac{1}{2}(\mathbf{x}-\mathbf{x}_b)^T\mathbf{B}^{-1}(\mathbf{x}-\mathbf{x}_b) + \frac{1}{2}(\mathbf{y}_o-\mathbf{H}[\mathbf{x}])^T(\mathbf{E}+\mathbf{F})^{-1}(\mathbf{y}_o-\mathbf{H}[\mathbf{x}]) \tag{2.1}$$

where $\mathbf{x}$ denotes the (model) atmosphere's state and $\mathbf{y}_o$ denotes the observations. $\mathbf{x}_b$ is a background state of the atmosphere, taken from a short range weather forecast. $\mathbf{H}[\mathbf{x}]$ denotes a (possibly nonlinear) forward operator calculating what a measurement $\mathbf{y}$ would be for any given atmospheric state $\mathbf{x}$. The matrices $\mathbf{B}$, $\mathbf{E}$ and $\mathbf{F}$ are error covariance matrices, describing the assumed uncertainties in the background data, the measurements, and the forward operator respectively. By minimising J with respect to the state vector $\mathbf{x}$, we obtain a solution that minimises the total deviation against background and observational data. If all errors are normally distributed and both background and measurements are unbiased, this solution is also the maximum likelihood solution (see, e.g., Lorenc, 1986).

The expression for $J(\mathbf{x})$ is general regardless of the dimension of $\mathbf{x}$. The analysis method may therefore be applied to retrieve a 1-dimensional vertical profile ("1D–Var"), a surface ("2D–Var") or to analyse the state of a regional or global model in 3 spatial ("3D–Var") or even 3 spatial and the time dimension ("4D–Var"). A schematic illustration of a 3D–Var system is given in Figure 2.1. At each analysis time, NWP fields are interpolated onto the observation's location, and the corresponding forward operator is used to calculate "forecast observations", which are compared with the real observations. Increments to the NWP fields are then calculated so that the difference between forecast and actual observations as well as between analysis and background are reduced, in effect providing a mapping of the observations back to the geophysical variables. Once the minimum of the cost function is reached, the next forecast is produced based on the updated NWP fields.

In the more elaborate 4D–Var, an initial state is forecast in time using a numerical model of the at-

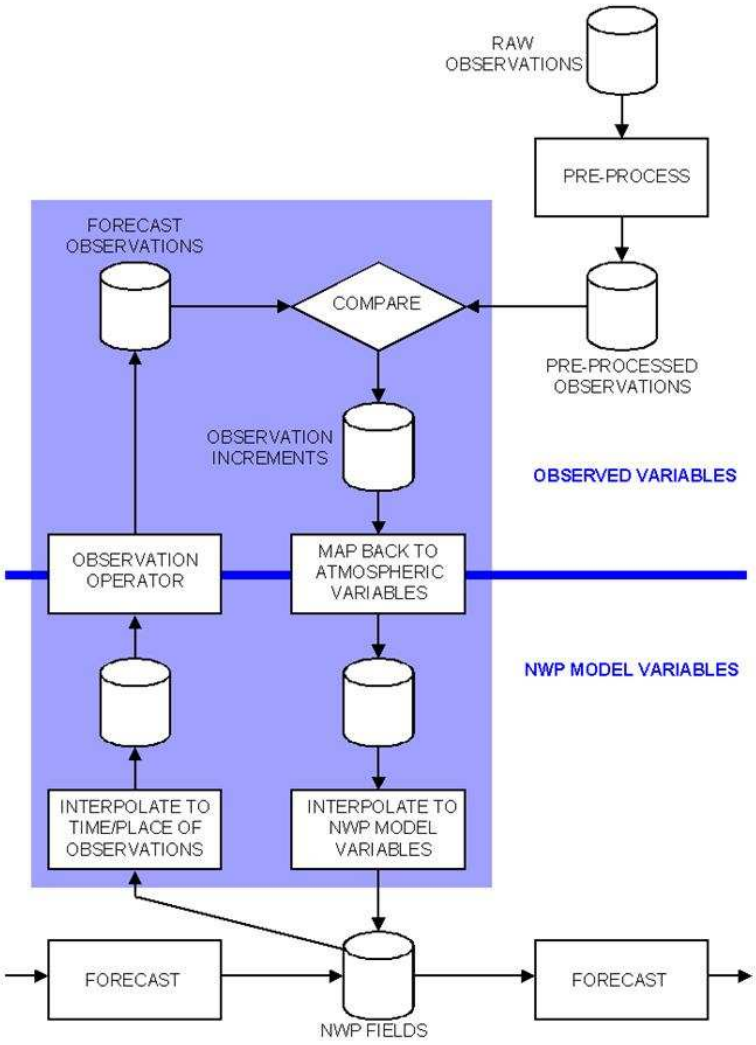

Figure 2.1: Schematic view of a 3D–Var data assimilation system (from Eyre, 1997).

mospheric circulation. The major advantage of 4D–Var is that observations at non–synoptic times can be utilised at the times they were actually taken, as the atmospheric state is forward modelled into the space of observations at the times they occurred. This is especially beneficial for satellite measurements which occur quasi–continuously, but comes at a significantly higher cost in terms of required computational time.

ROPP provides 1D–Var assimilation routines in module `ropp_1dvar` to retrieve atmospheric variables from the solution state vector which minimises the cost function for some observation and background data together with their associated errors. Quality control procedures are provided to ensure that the data have normally distributed errors.

2.2 Forward models

The main task in assimilating a new data type into an existing variational data assimilation system is the development of appropriate forward models for the observations in question. The forward models themselves are independent of the type of data assimilation system they are used in. For example, a forward model

calculating a vertical profile of bending angles can be used in a 1D–Var retrieval as well as in a 3D–Var or 4D–Var NWP data assimilation system. Apart from the actual forward models themselves, their gradient, tangent linear and adjoint codes are also required, as the latter are used in the numerical minimisation of the cost function.

ROPP provides one-dimensional forward models in module `ropp_fm` to calculate both refractivity and bending angle profiles for the neutral atmosphere from temperature, humidity and pressure data. Tangent linear, adjoint and gradient versions of the code along with test routines for the correctness of these are provided. These models, being one-dimensional, implicitly assume spherical symmetry of the atmosphere.

2.3 Quality control

The optimal solution of a variational analysis depends on the observations and background being unbiased (at least against each other), with normally distributed errors. The first condition is usually assured through regular monitoring of the observations and possibly the application of bias correction procedures derived from the monitoring statistics. A Gaussian distribution of errors is ensured by letting the observations undergo several quality control steps. These may include

- comparing observations with forward modelled background data (and rejecting observations that deviate by more than a critical value from the background, based on the assumed error estimates for both background and observations). This is sometimes termed "Background Quality Control (BGQC)".
- calculating "Probability of Gross Error" (Lorenc and Hammon, 1988; Ingleby and Lorenc, 1993) for individual data points (and later down-weighting them during the evaluation of the cost function).
- performing a 1D–Var retrieval as part of the data processing, using the value of the 1D–Var cost function at convergence as quality indicator (rejecting those with values above a certain threshold).

Note that the first two approaches require the ability to propagate background errors into the space of observations. In the linear limit, this can be achieved (for the full error covariance matrix) by calculating

$$\mathbf{O}_{bg} = \mathbf{H}'\mathbf{B}\mathbf{H}'^{T} \ ,$$

where $\mathbf{H}'$ is the gradient of the forward operator $\mathbf{H}$ with respect to the state vector elements. Thus, the gradients available from the forward models double in functionality when it comes to quality control. ROPP also contains other routines and functions required for an implementation of quality control procedures mentioned above.

2.4 1D–Var retrievals

Apart from its use as a quality control tool, one–dimensional variational retrievals provide an alternative to the classic dry temperature retrievals predominantly used in the radio occultation community. 1D–Var or

"statistically optimal" (Rodgers, 1976, 1990, 2000; Palmer et al., 2000) retrievals provide a solution to the well known temperature / humidity ambiguity of tropospheric radio occultation measurements (and many microwave and infra red remote sensing methods). They also provide a more robust upper level initialisation methodology for stratospheric temperatures obtained from radio occultation soundings. Assuming that the error characteristics of both background and observations are realistic, the variational framework also provides a full error characterisation for the retrieved profiles. This information is not available from the more traditional RO retrieval methods.

ROPP provides minimisation and some technical routines (e.g., to perform an efficient pre–conditioning of the minimisation problem) required for the implementation of 1D–Var retrieval systems. Post-processing routines calculating the error covariance matrix of the retrieval are also available. For both testing and illustration purposes, implementations of 1D–Var retrievals using refractivity and bending angle profiles are also contained in ROPP.

2.5 Preprocessing and error characteristics

The solution of a variational data assimilation analysis depends on the error characteristics of both the background and the observations. The ROPP package only contains simple error covariance models for bending angle and refractivity profiles. Some global mean error statistics for operational NWP data are also provided in the form of data files.

It should be kept in mind, though, that the error characteristics of any observation depend on the pre-processing the raw data underwent. In case of RO, the preprocessing chain from the raw GNSS observables is rather elaborate. It usually contains several smoothing steps as well as the use of *a priori* information (e.g., in form of some kind of "statistical optimisation" when refractivity has been calculated from bending angles). Thus, the error characteristics of refractivity or bending angle data depend on details of the pre-processing, which may or may not be known in detail by the user. NWP users may prefer to tune smoothing and other parameters in the preprocessing according to their individual needs. The ROPP module `ropp_pp` contains an implementation of processing from Level 1 (amplitude and excess phase) data to Level 2 bending angle and refractivity profiles. See ROM SAF (2013) for details. Users also likely to thin RO profiles in the vertical according to their needs (and their models' vertical resolution), e.g. in order to reduce or avoid vertically correlated observations. The ROPP module `ropp_io` contains 1D thinning algorithms as detailed by ROM SAF (2009).

References

Eyre, J. R., Variational assimilation of remotely–sensed observations of the atmosphere, *J. Met. Soc. Jap.*, *75*, 331–338, 1997.

Ingleby, N. B. and Lorenc, A. C., Bayesian quality control using multivariate normal distributions, *Quart. J. Roy. Meteorol. Soc.*, *119*, 1195–1225, 1993.

Lorenc, A. C., Analysis methods for numerical weather prediction, *Quart. J. Roy. Meteorol. Soc.*, *112*, 1177–1194, 1986.

Lorenc, A. C. and Hammon, O., Objective quality control of observations using Bayesian methods. Theory and a practical implementation, *Quart. J. Roy. Meteorol. Soc.*, *114*, 515–543, 1988.

Palmer, P. I., Barnett, J. J., Eyre, J. E., and Healy, S. B., A nonlinear optimal estimation inverse method for radio occultation measurements of temperature, humidity, and surface pressure, *J. Geophys. Res.*, *105*, 17.513–17.526, 2000.

Rodgers, C. D., Retrieval of atmospheric temperature and composition from remote sounding measurements of thermal radiation, *Rev. Geophys. Space Phys.*, *14*, 609–624, 1976.

Rodgers, C. D., Characterization and error analysis of profiles retrieved from remote sounding measurements, *J. Geophys. Res.*, *95*, 5587–5595, 1990.

Rodgers, C. D., *Inverse methods for atmospheric sounding: Theory and practice*, World Scientific Publishing, Singapore, New Jersey, London, Hong Kong, 2000.

ROM SAF, ROPP Thinner Algorithm, SAF/GRAS/METO/REP/GSR/008, 2009.

ROM SAF, The Radio Occultation Processing Package (ROPP) User Guide. Part III: Pre-processor module, SAF/ROM/METO/UG/ROPP/004, Version 7.0, 2013.

3 Forward models

The ROPP Forward Model `ropp_fm` module provided as part of the ROPP software enables users to relate the bending angle and refractivity derived from GNSS-RO data to meteorological profiles of pressure, temperature and humidity. By using a forward model (or forward operator) information in the space of model variables can be mapped into that of the measured variables, and back, in a consistent manner (Eyre, 1997). A data assimilation system determines increments to NWP fields by comparing the NWP field, mapped into observation space using a forward model, with real observations and minimising the differences between the model and observations and between analysis and background NWP fields.

3.1 Radio occultation geometry

A detailed review of GNSS-RO measurements was provided by Kursinski et al. (1997). The typical radio occultation geometry is sketched in Figure 3.1. A ray passing from a GNSS to a LEO satellite through the atmosphere is refracted as a result of gradients in the refractive index, n. In general, the equations defining the the two-dimensional ray path, in circular polar co-ordinates (r and θ), are given by (page 149 Rodgers, 2000).

$$\begin{aligned} \frac{dr}{ds} &= \cos\phi \\ \frac{d\theta}{ds} &= \frac{\sin\phi}{r} \\ \frac{d\phi}{ds} &= -\sin\phi\left[\frac{1}{r}+\frac{1}{n}\left(\frac{\partial n}{\partial r}\right)_\theta\right]+\frac{\cos\phi}{nr}\left(\frac{\partial n}{\partial\theta}\right)_r \end{aligned} \tag{3.1}$$

where s is the distance along the ray-path, n is the refractive index, ϕ is angle between the local radius vector and the tangent to the ray-path. Under the assumption of spherical symmetry, where it is assumed $\left(\frac{\partial n}{\partial\theta}\right)_r = 0$, it can be shown that

$$a = nr\sin\phi = \text{constant} \tag{3.2}$$

along the ray-path. This is also known as Bouguer's formula (e.g., Born and Wolf, 1980). The constant, a, is called the impact parameter, and in the processing of GNSS radio occultation measurements it is assumed that the value of a at the GNSS satellite, a_G, and LEO satellite, a_L, are equal. The impact parameter also determines the height of the tangent point of the ray-path. If the tangent point is defined as that point on the ray where the angle ϕ between the position vector and the ray's tangent equals 90°, we obtain $\sin\phi = 1$, and therefore

$$a = n_t r_t \tag{3.3}$$

with r_t being the distance between the centre of curvature and the tangent point, and n_t being the refractive index at that point. This allows the calculation of the impact parameter from given refractivity

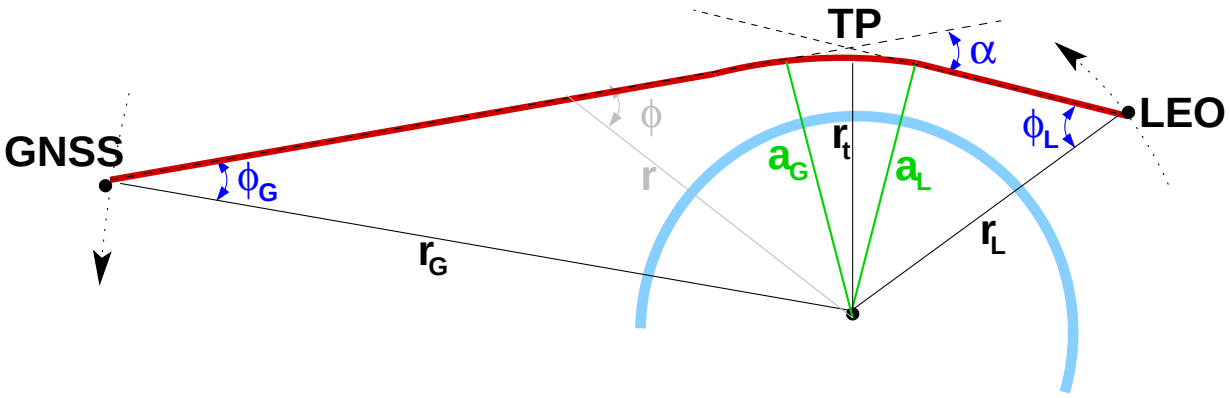

Figure 3.1: Radio occultation geometry; shown are the bending angle α, the GNSS and LEO side impact parameters (a_G and a_L), the GNSS and LEO coordinate vectors ($\mathbf{r}_G$, $\mathbf{r}_L$), the ray path (solid red line) and the satellite side asymptotes of the ray path (dashed). Radius r_t shows the radial distance between the centre of curvature and the ray tangent point.

n and geometric height h:

$$a = (h + R_c) \cdot n(h + R_c) \tag{3.4}$$

where, as indicated, n is evaluated at $(h+R_c)$ and R_c is the radius of curvature of the (assumed ellipsoidal) Earth at the tangent point. (h is the height above this ellipsoid.)

Note that the two satellite side impact parameters a_G and a_L will, in general, not be equal if the spherical symmetry assumption is not met. In addition, the impact parameter a calculated from (3.3) at some hypothetical tangent point is not necessarily the same as satellite side impact parameters in the non–symmetrical case.

3.2 Bending angle

The total amount of refraction is characterised by the bending angle α. This is the angle between the two satellite side ray asymptotes (Figure 3.1). Two-dimensional bending angle forward models in ROPP solve Equations (3.1). However, the forward modelling can be simplified considerably by assuming spherical symmetry so that a is a constant along the ray path, In this case, it can be shown that $\alpha(a)$ is given by the integral (Fjeldbo et al., 1971; Melbourne et al., 1994; Kursinski et al., 1997)

$$\begin{aligned} \alpha(a) = -2\int_{r_t}^{\infty} d\alpha &= -2a\int_{r_t}^{\infty} \frac{1}{\sqrt{r^2n^2 - a^2}} \frac{d\ln(n)}{dr} dr \\ &= -2a\int_{a}^{\infty} \frac{1}{\sqrt{x^2 - a^2}} \frac{d\ln(n)}{dx} dx \ . \end{aligned} \tag{3.5}$$

The second expression is obtained by substituting $x = nr$. The negative sign in Equation (3.5) follows the convention that bending towards the Earth's surface is positive.

The one-dimensional (1D) forward models provided in ROPP solve Equation (3.5) using the algorithm is described by Healy and Thepaut (2006) and outlined in Chapter 4.

3.3 Refractivity

The amount of bending a ray of radio waves experiences on its way through the atmosphere depends on the local refractive index n of air along the ray path. For convenience, refractivity N is often used instead of the refractive index n. These are related as

$$N = (n-1) \times 10^6 \ . \tag{3.6}$$

At microwave frequencies in the Earth's atmosphere, N varies due to contributions from the dry neutral atmosphere, water vapour, free electrons in the ionosphere and particulates, thus Kursinski et al. (1997):

$$N = \kappa_1 \frac{P_d}{T} + \kappa_2 \frac{e}{T} + \kappa_3 \frac{e}{T^2} + \kappa_4 \frac{n_e}{f^2} + \kappa_5 W \ . \tag{3.7}$$

The first term is the dry neutral atmosphere contribution where P_d is the pressure of the dry air and T is the temperature. The second and third term is the water vapour contribution, where e is the partial water vapour pressure. The fourth ionospheric contribution results from free electrons in the ionosphere where n_e is the electron number density and f is the transmitter frequency of the radio signal. The final term results from scattering, mainly due to cloud droplets where W is the liquid water content.

For radio occultation soundings it is usually assumed that the scattering term is negligible, and that ionospheric effects have been removed during the pre-processing, such as by an ionospheric correction of the bending angles (Vorob'ev and Krasil'nikova, 1994). The forward models in ROPP therefore only deal with the refractivity of the neutral atmosphere.

Neglecting non-ideal gas effects, a 3-term expression relates refractivity to atmospheric parameters as

$$N = k_1 \frac{P_d}{T} + k_2 \frac{e}{T^2} + k_3 \frac{e}{T} \tag{3.8}$$

(Smith and Weintraub, 1953; Bevis et al., 1994) (Note that κ_1 (Equation (3.7)) $= k_1$ (Equation (3.8)), but that $\kappa_2 = k_3$ and $\kappa_3 = k_2$.) Further discussion of the refractivity formulation is provided by Rueger (2002), ROM SAF (2008) and ROM SAF (2009).

In the default ROPP settings we use a three term expression, but set $k_1 = k_3 = 77.60$ N-unit K hPa^{-1}, so it is consistent with the two term expression given in Smith and Weintraub (1953). However, it is easier to introduce non-ideal gas effects into the three term expression (see Chapter 8).

3.4 Summary

The `ropp_fm` module contains a one-dimensional operator to calculate refractivity as a function of geopotential height $N(Z)$, and both one- and two-dimensional operators to calculate bending angle as function of impact parameter, $\alpha(a)$. The one-dimensional refractivity operator evaluates Equation (3.8) at the observation heights. The one- and two-dimensional bending angles operators solve equations (3.5) and (3.1), respectively.

These forward models all include the following components:

- Mapping background information to pressure, temperature and humidity profiles.
- Integration of the hydrostatic equation to determine the geopotential heights of the model levels

The details of these tasks will depend on the format of the background data, and are they discussed in Chapter 4 and 5.

References

Bevis, M., Businger, S., Chiswell, S., Herring, T. A., Anthes, R. A., Rocken, C., and Ware, R. H., Mapping zenith wet delays onto precipitable water, *J. Appl. Meteor.*, *33*, 379–386, 1994.

Born, M. and Wolf, E., *Principles of Optics*, Pergamon Press, Oxford, 1980.

Eyre, J. R., Variational assimilation of remotely–sensed observations of the atmosphere, *J. Met. Soc. Jap.*, *75*, 331–338, 1997.

Fjeldbo, G., Kliore, G. A., and Eshleman, V. R., The neutral atmosphere of Venus as studied with the Mariner V radio occultation experiments, *Astron. J.*, *76*, 123–140, 1971.

Healy, S. B. and Thepaut, J.-N., Assimilation experiments with CHAMP GPS radio occultation measurements, *Quart. J. Roy. Meteorol. Soc.*, *132*, 605–623, 2006.

Kursinski, E. R., Hajj, G. A., Schofield, J. T., Linfield, R. P., and Hardy, K. R., Observing earth's atmosphere with radio occultation measurements using the Global Positioning System, *J. Geophys. Res.*, *102*, 23.429–23.465, 1997.

Melbourne, W. G., Davis, E. S., Duncan, C. B., Hajj, G. A., Hardy, K. R., Kursinski, E. R., Meehan, T. K., and Young, L. E., The application of spaceborne GPS to atmospheric limb sounding and global change monitoring, Publication 94–18, Jet Propulsion Laboratory, Pasadena, Calif., 1994.

Rodgers, C. D., *Inverse methods for atmospheric sounding: Theory and practice*, World Scientific Publishing, Singapore, New Jersey, London, Hong Kong, 2000.

Rueger, J. M., Refractive index formulae for electronic distance measurement with radio and millimetre waves, Unisurv Report S-68. School of Surveying and Spatial Information Systems, University of New South Wales, [Summary at http://www.fig.net/pub/fig_2002/Js28/JS28_rueger.pdf], 2002.

ROM SAF, Refractivity calculations in ROPP, SAF/GRAS/METO/REP/GSR/005, 2008.

ROM SAF, Refractivity coefficients used in the assimilation of GPS radio occultation measurements, SAF/GRAS/METO/REP/GSR/009, 2009.

Smith, E. K. and Weintraub, S., The constants in the equation for atmospheric refractivity index at radio frequencies, in *Proc. IRE*, vol. 41, pp. 1035–1037, 1953.

Vorob'ev, V. V. and Krasil'nikova, T. G., Estimation of the accuracy of the atmospheric refractive index recovery from doppler shift measurements at frequencies used in the NAVSTAR system, *USSR Phys. Atmos. Ocean, Engl. Transl.*, *29*, 602–609, 1994.

4 ROPP one-dimensional forward models

The ROPP forward model module (`ropp_fm`) includes routines to compute one-dimensional profiles of refractivity (`ropp_fm_refrac_1d`) and bending angle (`ropp_fm_bangle_1d`) from one-dimensional background pressure, temperature and humidity data. This is necessary for profile retrievals and the assimilation of radio occultation observations in NWP.

Figure 4.1 shows example refractivity and bending angle profiles computed from model background profiles of pressure, temperature and specific humidity. The results are computed for specified observation heights, enabling direct comparison with refractivity or bending angle data obtained by radio occultation in a data assimilation system.

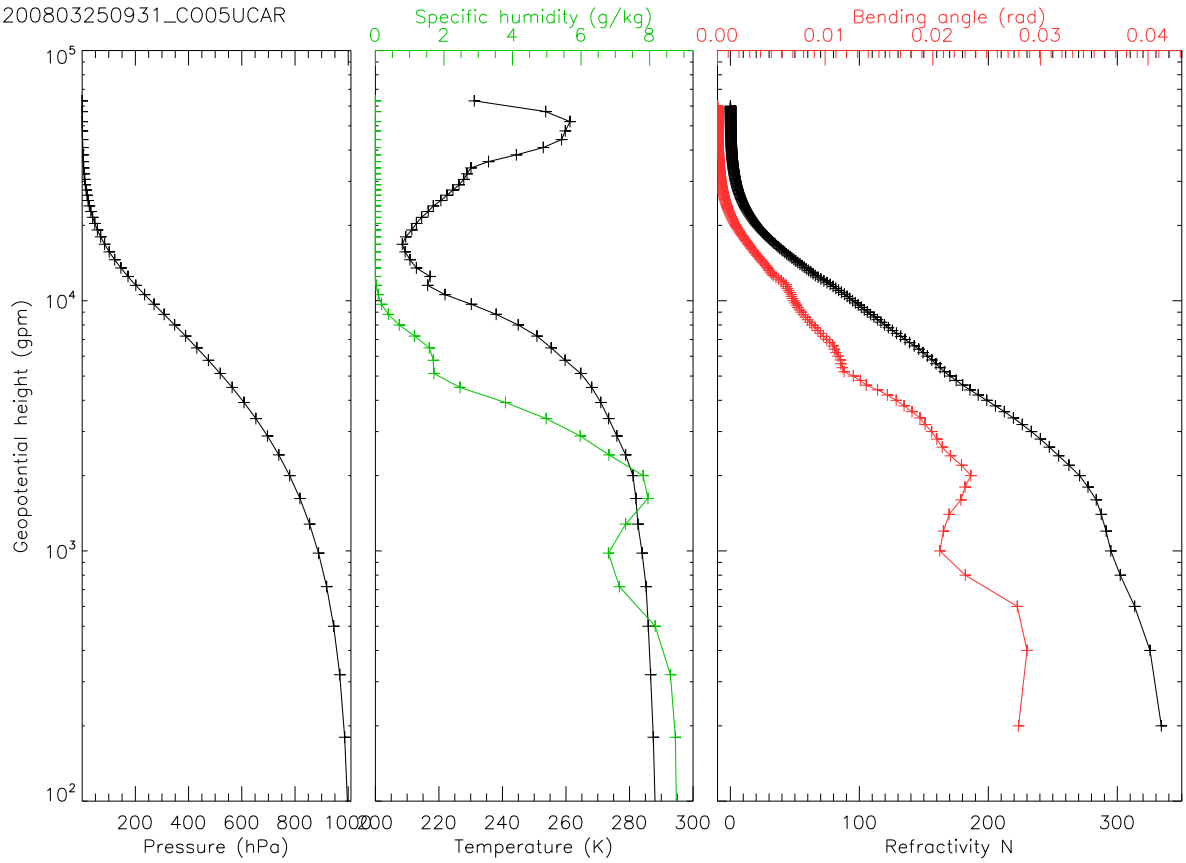

Figure 4.1: Example refractivity and bending angle profiles computed using the model pressure, temperature and specific humidity profiles plotted. Symbols show the vertical resolution of the input model data and the output observation heights.

4.1 ROPP forward model tool

A stand-alone tool `ropp_fm_bg2ro_1d` is provided in `ropp_fm` as an illustration of how the `ropp_fm` routines can be implemented to derive profiles of refractivity and bending angle from background meteorological data. Figure 4.2 shows how the `ropp_fm` routines are integrated in the `ropp_fm_bg2ro_1d` code.

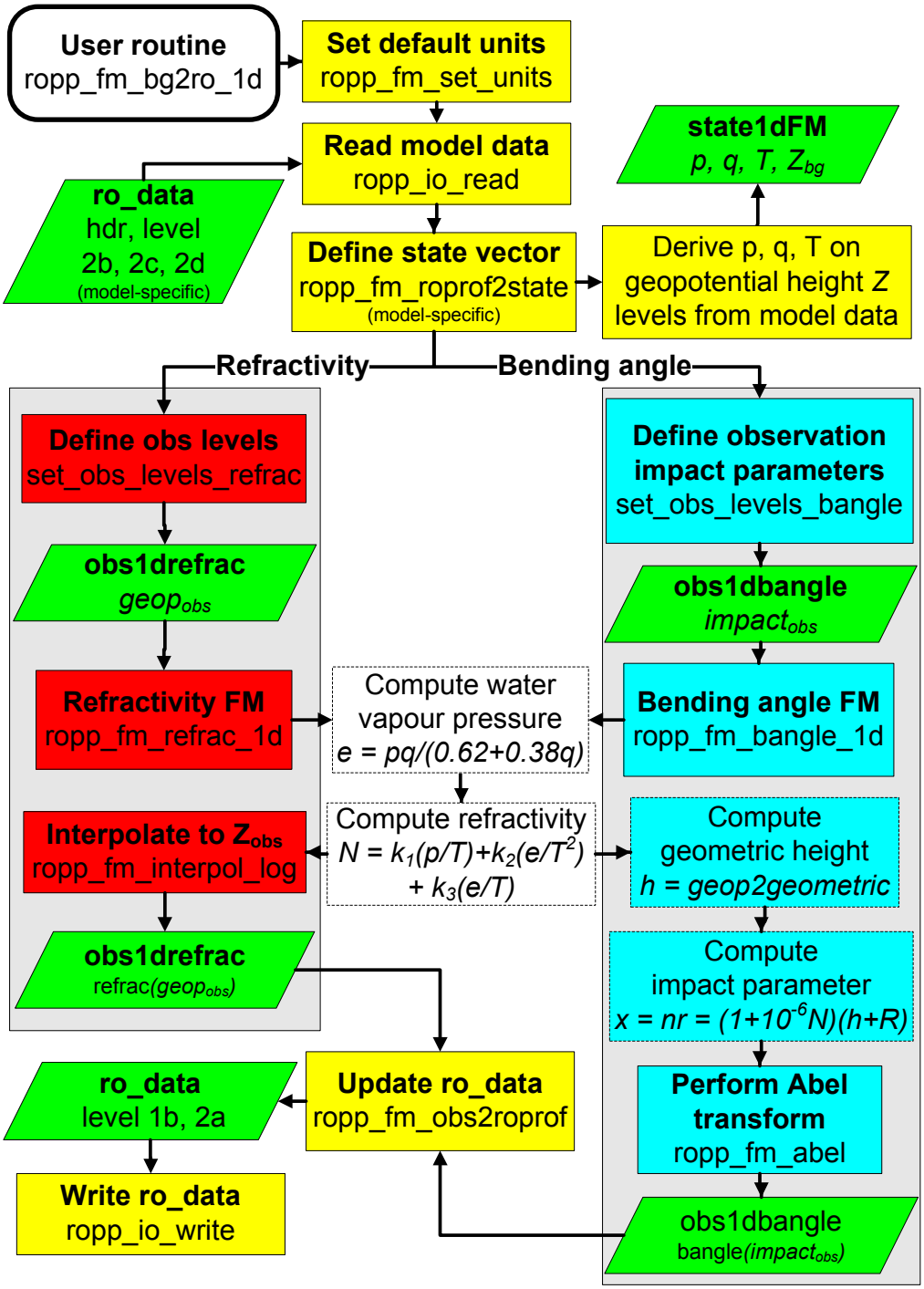

Figure 4.2: Flow chart illustrating calling tree of the ROPP forward model to compute refractivity and bending angle profiles from input background model data.

4.1.1 Implementation

The `ropp_fm_bg2ro_1d` tool is run using the command

```
ropp_fm_bg2ro_1d <inputdatafile> -o <outputfile>
```

where `<inputdatafile>` is a ROPP netCDF file (ROM SAF, 2013) containing the input model data and `<outputfile>` will contain the forward modelled refractivity and bending angle profiles on pre-defined observation levels, also as a ROPP netCDF file.

If the input file is a multi-file containing more than one model profile, or more than one input file is specified, `ropp_fm_bg2ro_1d` computes the forward model for each profile in turn and an output file is generated containing all the output profiles.

The following command line options can be used with the `ropp_fm_bg2ro_1d` tool:

`-o <outputfile>`	ROPP netCDF file for output retrieved profiles
`-f`	forward model only, do not calculate gradients
`-use_logp`	use log(pressure) in forward model
`-use_logq`	use log(spec humidity) in forward model
`-comp`	use non-ideal gas compressibility options in forward model
`-check_qsat`	include check against saturation in forward model
`-d`	output additional diagnostic information (VerboseMode)
`-h`	give help menu
`-v`	output version information

4.1.2 Code organisation

Figure 4.2 shows how the `ropp_fm_bg2ro_1d` tool is composed of the following stages:

- **Input data access and transformation to a generic state vector**

  Setup the input data arrays, read the input data and define a state vector structure `State1dFM` containing the background pressure p, temperature T and humidity q data as a function of geopotential height Z. If `check_qsat` is in force, q is limited by the saturation value at that temperature and pressure; otherwise, by default, supersaturation is allowed. See Section 4.4.

- **Define the observation levels for which refractivity and bending angles are to be calculated**

  This stage follows from the requirement of the forward model to compute results on levels corresponding to the observation heights for a given occultation. The observation vector structure `Obs1dRefrac` and `Obs1dBangle` is either defined using the observation heights specified in the input file, if these exist, or with default levels specified using subroutines `set_obs_levels_refrac` and `set_obs_levels_bangle` respectively.

- **Compute refractivity profile**

  `ropp_fm_refrac_1d` is the main refractivity forward model routine to compute refractivity N as a function of geopotential height Z. See Section 4.8.

- **Compute bending angle profile**

  `ropp_fm_bangle_1d` is the main bending angle forward model routine to compute bending angle α as a function of impact parameter a. See Section 4.9.

- **Write results to generic RO data structure and output file**

4.2 Data types

The observation state vector `y` and background state vector `x` are represented within ROPP by Fortran 90 derived types (or structures). These data structures are defined in `ropp_fm_types`.

Note that the ROPP forward model assumes data are in ascending height order. The input data are checked and reordered as part of the stand-alone tool processing (see Section 4.4). If required for 1D–Var, users must ensure that the error correlations used are appropriate for background and observation data in ascending height order.

4.2.1 Observation vector: `Obs1dRefrac`, `Obs1DBangle`

Refractivity and bending angle profiles are represented by the `Obs1dRefrac` and `Obs1dBangle` data structures respectively. Their elements are listed in Table 4.1.

For example, a structure holding data of an observed (or modelled) refractivity profile can be declared by

```
USE ropp_fm
TYPE(Obs1dRefrac) :: y
```

Refractivity observations are stored in the `y%refrac` element along with the corresponding geopotential heights `y%geop`. Similarly, a bending angle profile can be declared as `type(Obs1dBangle)` with bending angles stored in element `y%bangle` and the corresponding impact parameter values in `y%impact`.

The structure element `y%weights` is used by 1D–Var quality control routines provided by ROPP. Values range between 0 (data point rejected by QC) and 1 (data point passed by QC). This information can be used to downweight particular observation data points during the cost function minimisation of a variational analysis. The error covariance element `y%cov` is also required by the `ropp_1dvar` module (Section 7.4).

The observation vector for bending angles also contains information on the surface gravity `y%g_sfc` and effective radius of Earth `y%r_earth`. The local radius of curvature `y%r_curve` is also included together with the undulation `y%undulation`, which is defined as the height of the EGM96 (NASA/NIMA) geoid above the WGS84 (NGA) reference ellipsoid at the occultation point. If a height `h_Geoid` is expressed with respect to the geoid, then height `h_Ellipsoid` with respect to the ellipsoid is given by

$$\texttt{h\_Ellipsoid} = \texttt{h\_Geoid} + \texttt{y\%undulation} \tag{4.1}$$

Obs1drefrac			
Structure element	Description	Range	Units
...%refrac	Refractivity values	[0 500]	N-units
...%geop	Geopotential height	[-1000 10000]	gpm
...%weights	Observation weights	[0 1]	
...%lon	Longitude	[-180 180]	° E
...%lat	Latitude	[-90 90]	° N
...%time	Time	[]	Jul sec
...%cov	Error covariance for refractivity	[0 10]	N-units
...%obs_ok	Flag to specify observations ok	[0 1]	
...%cov_ok	Flag to specify error covariance ok	[0 1]	
Obs1dbangle			
Structure element	Description	Range	Units
...%bangle	Bending angle	[0.0001 0.01]	rad
...%impact	Impact parameter	[6200000 6600000]	m
...%weights	Observation weights	[0 1]	
...%rtan	Tangent point radius	[6200000 6600000]	
...%a_path	$nr\sin\phi$ at ray endpoints	[6200000 6600000]	
...%lon	Longitude	[-180 180]	° E
...%lat	Latitude	[-90 90]	° N
...%time	Time	[]	Jul sec
...%g_sfc	Gravity at the surface	[]	ms^{-2}
...%r_earth	Effective earth radius	[6200000 6600000]	m
...%r_curve	Radius of curvature	[6200000 6600000]	m
...%undulation	Undulation	[-150 150]	m
...%azimuth	Azimthuthal angle	[-180 180]	° N
...%cov	Error covariance for bending angle	[0 0.01]	rad
...%obs_ok	Flag to specify observations ok	[0 1]	
...%cov_ok	Flag to specify error covariance ok	[0 1]	

Table 4.1: Elements of the Obs1dBangle and the Obs1dRefrac observation Vector structure

4.2.2 State vector: `State1dFM`

Background meteorological data are represented by the `State1dFM` data structure. Its elements are listed in Table 4.2.2. Vertical profiles of temperature `x%temp`, pressure `x%pres` and specific humidity `x%shum` are stored together with the corresponding geopotential height levels `x%geop`. The number of background vertical levels is stored in the `x%n_lev` element. A structure holding background data can then be declared as

```
USE ropp_fm
TYPE(State1dFM) :: x
```

The state vector `x%state` is required for the `ropp_1dvar` processing. Its contents depends on the details of the background data. Although it is not directly used in `ropp_fm` processing its elements are updated using background-specific mapping between the state vector and conventional meteorological variables (as

State1dFM			
Structure element	Description	Range	Units
...%state	State vector data		
...%temp	Temperature	[150 350]	K
...%shum	Specific humidity	[0 0.05]	kg/kg
...%pres	Pressure	[10 110000]	Pa[a]
...%geop	Geopotential height	[-1000 100000]	gpm
...%ak	ECMWF-specific level coefficient	[0 200000]	Pa[a]
...%bk	ECMWF-specific level coefficient	[0 2]	
...%n_lev	Number of model levels	[1 100]	
...%lon	Longitude	[-180 180]	° E
...%lat	Latitude	[-90 90]	° N
...%time	Time	[]	Jul sec
...%cov	Error covariance for state vector	[0 2500]	
...%state_ok	Flag to specify state vector ok	[0 1]	
...%cov_ok	Flag to specify error covariance ok	[0 1]	
...%use_logp	Flag to specify use log(p) in state	[0 1]	
...%use_logq	Flag to specify use log(q) in state	[0 1]	
...%non_ideal	Non-ideal gas flag	[0 1]	

[a] Note that pressure variables in State1dFM are in units of Pa while standard units in the ROprof structure is hPa. See ROM SAF (2013) for details. Users must specify pressure variables in Pa or ensure unit conversion is carried out before performing ropp_fm calculations if using ropp_io for reading input data. Routine ropp_fm_set_units is provided for this task.

Table 4.2: Elements of the State1dFM vector structure

required if ropp_fm were implemented as part of ropp_1dvar). Further details are provided in Section 4.4 and Chapter 7. Elements x%ak and x%bk are required for this mapping if State1dFM is holding a state vector using hybrid vertical levels (e.g. ECMWF) as described in Section 4.5. The error covariance element x%cov is required by the ropp_1dvar module (Section 7.5).

4.3 Defining default units: ropp_fm_set_units

The units listed in Tables 4.1 and 4.2.2 are the assumed default units for each variable in the ropp_fm module. Note these may differ from the units specified in a ROPP-format input file. Unit conversion together with transformation of the corresponding valid range parameters is performed before any other ropp_fm processing by calling subroutine ropp_fm_set_units. This utilises the ropp_io unitconvert library functions.

```
USE ropp_io
USE ropp_fm
TYPE(ROprof) :: ro_data
CALL ropp_fm_set_units(ro_data)
```

4.4 Defining the state vector: ropp_fm_roprof2state

The ropp_io module routine ropp_io_read reads a single profile of model background data from a netCDF ROPP format input file and fills the elements of the generic ROPP data structure type ROprof (ROM SAF, 2013).

```
USE ropp_io
TYPE(ROprof) :: ro_data
CALL ropp_io_read(ro_data, inputfilename, rec=iprofile)
```

The relevant background model data are copied to the state vector x of data type State1dFM by calling the routine ropp_fm_roprof2state.

```
USE ropp_io
USE ropp_fm
TYPE(ROprof)    :: ro_data
TYPE(State1dFM) :: x
CALL ropp_fm_roprof2state(ro_data, x)
```

A number of checks on the input data are also conducted at this stage. The state vector x%state_ok flag is set to false if any test fails.

- Check longitude and latitude for background data are within range,
  set x%lon = ro_data%georef%lon
  set x%lat = ro_data%georef%lat
- Check that data arrays are in ascending order. Note that ropp_fm calculations assume that data array index 1 is closest to the surface. The input data are checked and reordered as required using
  call ropp_io_ascend(ro_data)
- Check that ro_data contains free atmosphere data (level 2b data)
  set x%n_lev = ro_data%Lev2b%Npoints

The state vector is required to contain temperature, specific humidity and pressure data as a function of geopotential height (Table 4.2.2). These data might be read in directly from the input file if the user provides the background data in this form. It is more likely however that elements of the state vector need to be calculated from the available background data provided by a user and the exact specification of the vertical level representation adopted in the NWP model. The type of model data contained in the input file is specified by the input variable ro_data%Lev2d%level_type. ropp_fm contains routines to derive the required generic state vector using background data from a NWP model where the vertical coordinate is based on pressure levels (e.g. ECMWF, Section 4.5) or on geopotential height (e.g. Met Office, Section 4.6). Users may prefer to implement their own functions to cater for the specific background vertical level type and available data provided.

Elements of the error covariance matrix x%cov used in ropp_1dvar are initialised in ropp_fm_roprof2state by setting its diagonal elements to the standard deviation associated with each corresponding meteorological variable. These values should be specified in the input RO file (Chapter 7).

4.5 Hybrid sigma vertical levels - ECMWF model type: `ropp_fm_state2state_ecmwf`

For `ropp_fm` applications using background data from a model where the vertical coordinate is pressure-based (e.g. ECMWF), it is necessary to compute the pressure p on each model level. The geopotential height Z of each model level can then be calculated using the hydrostatic equation

$$Z(p) = Z(p_{\rm sfc}) - \frac{R_{\rm dry}}{g_{\rm wmo}} \int_{p_{\rm sfc}}^{p} T_v d\ln p \tag{4.2}$$

where $Z(p_{\rm sfc})$ is the surface geopotential height, $R_{\rm dry}$ is the gas constant for dry air, $g_{\rm wmo}$ is the reference gravitational acceleration and T_v is the virtual temperature. This conversion from model-specific to generic `ropp_fm` variables is performed using a method which is consistent with the model dynamics in `ropp_fm_state2state_ecmwf`.

```
USE ropp_fm
TYPE(State1dFM) :: x
CALL ropp_fm_state2state_ecmwf(x)
```

4.5.1 Vertical grid structure

Figure 4.3 shows a schematic illustration of the vertical model level and data structure of the ECMWF Integrated Forecasting System (IFS) (ECMWF, 2007). The atmosphere is divided into N layers, on which values of the temperature `x%temp`$=T$ and humidity `x%shum`$=q$ are defined. The levels are defined by the pressure at the interfaces (half-levels) between them. The pressure on each half-level is not stored directly, but is defined using the surface pressure $p_{\rm sfc}$ and level coefficients `x%ak`$=a_{k+1/2}$, `x%bk`$=b_{k+1/2}$ stored on each half level as

$$p_{k+1/2} = a_{k+1/2} + b_{k+1/2} p_{\rm sfc} \tag{4.3}$$

for $0 \leq k \leq N$. The pressure on each full-level is then calculated as

$$\texttt{x\%pres} = p_k = \frac{1}{2}(p_{k-1/2} + p_{k+1/2}) \tag{4.4}$$

for $1 \leq k \leq N$. The level coefficients are provided if available in the `ROprof` data structure as Level2d data (ROM SAF, 2013).

The geopotential height of each full level `x%geop`$=Z_k$ is computed following the ECMWF (2007) algorithms (Simmons and Burridge, 1981). The discrete version of the hydrostatic equation (Equation 4.2) enables the geopotential height on a half-level to be computed as

$$Z_{k+1/2} = Z_{\rm sfc} + \sum_{j=1}^{k} \frac{R_{\rm dry}(T_v)_j}{g_{\rm wmo}} \ln\left(\frac{p_{k-1/2}}{p_{k+1/2}}\right) \tag{4.5}$$

for $1 \leq k \leq N$. Full-level values of the geopotential height are then given by interpolation as

$$Z_k = Z_{k-1/2} + \alpha_k \frac{R_{\rm dry}(T_v)_k}{g_{\rm wmo}} \tag{4.6}$$

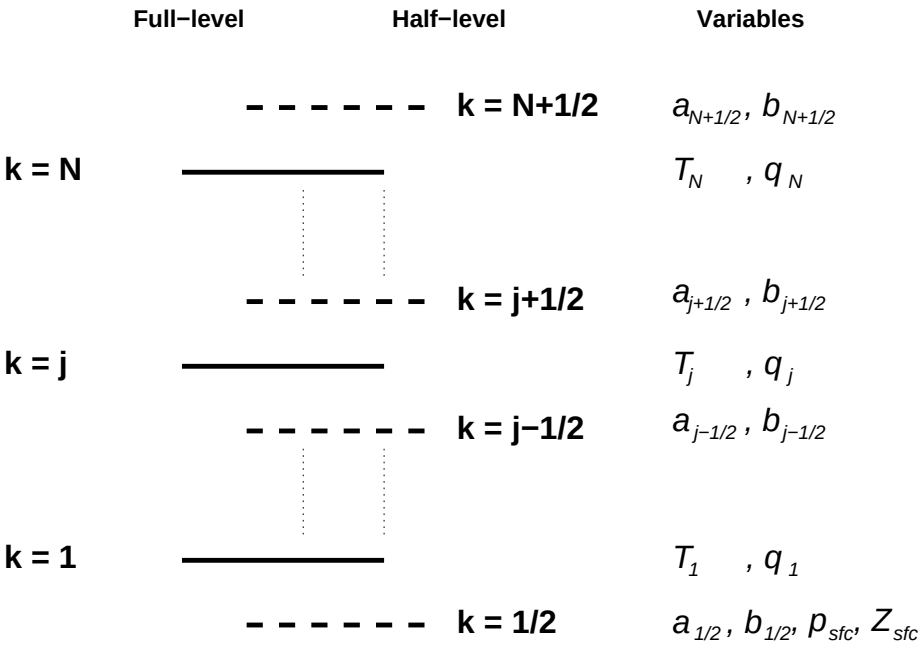

Figure 4.3: Schematic illustration of model level and data structure for a hybrid sigma vertical level model (e.g. ECMWF). The atmosphere is divided into N layers or full levels, defined by the pressures at the $N+1$ interfaces (half-levels) between them.

where the interpolation coefficient α_k is

$$\alpha_k = 1 - \frac{p_{k+1/2}}{\Delta p_k} \ln\left(\frac{p_{k-1/2}}{p_{k+1/2}}\right) \tag{4.7}$$

for $1 \leq k < N$ (and $\alpha_N = \ln 2$), and the pressure difference between half-levels is given by

$$\Delta p_k = p_{k-1/2} - p_{k+1/2} \tag{4.8}$$

The virtual temperature T_v on each full level is derived from the background temperature `x%temp`$=T$ and specific humidity `x%shum`$=q$ as

$$T_v = T\left[1 + \left(\frac{R_{\text{vap}}}{R_{\text{dry}}} - 1\right) q\right] \tag{4.9}$$

where R_{vap} is the gas constant for water vapour. Note that all physical constants used in `ropp_fm` are defined in `ropp_fm_constants`.

4.6 Geopotential-based vertical levels - Met Office model type: `ropp_fm_state2state_meto`

For `ropp_fm` applications using background data from a model where the vertical coordinate is geopotential height-based (e.g. Met Office), it is necessary to know the pressure p on model levels where humidity information is stored. The temperature T at those locations can also be recovered using the hydrostatic

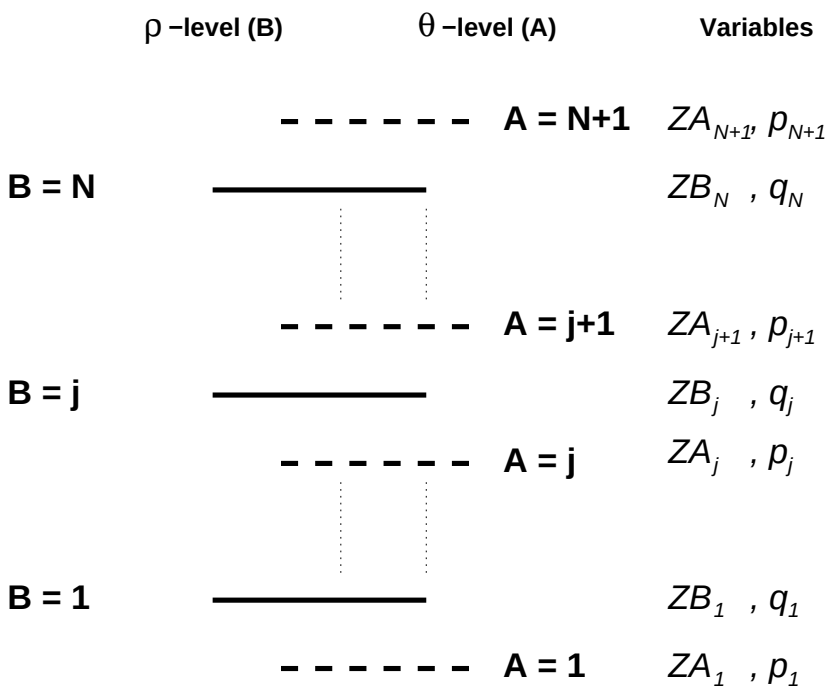

Figure 4.4: Schematic illustration of model level and data structure for a geopotential height based vertical level model (e.g. Met Office). Information is stored on a staggered height grid, with pressure and density on $N+1$ 'ρ' (or 'A') levels and potential temperature and humidity information on the N intermediate 'θ' (or 'B') levels.

equation (Equation 4.2) assuming the temperature is constant across each model layer. These conversions from model-specific to generic `ropp_fm` variables are performed using a method which is consistent with the model dynamics in `ropp_fm_state2state_meto`.

```
USE ropp_fm
TYPE(State1dFM) :: x
CALL ropp_fm_state2state_meto(x)
```

4.6.1 Vertical grid structure

Figure 4.4 shows a schematic illustration of the vertical model level and data structure of the Met Office Unified Model (UM). Information is provided on a staggered vertical grid. Pressure and density are defined on 'ρ' or 'A' levels while humidity and potential temperature are defined on the intermediate 'θ' or 'B' levels.

The pressure on each 'B'-level `x%pres` is calculated from the available background data by first computing the Exner pressure Π on each 'A'-level as

$$\Pi_A = \left(\frac{p_A}{p_{\mathrm{ref}}} \right)^{\kappa} \tag{4.10}$$

where $\kappa = R_{\mathrm{dry}}/C_p$ and p_{ref} is the standard surface pressure (1000 hPa). The Exner value on the i^{th} 'B'-level

is then calculated assuming a linear variation with geopotential height as

$$\Pi_B(i) = \left(\frac{Z_A(i+1) - Z_B(i)}{Z_A(i+1) - Z_A(i)}\right)\Pi_A(i) + \left(\frac{Z_B(i) - Z_A(i)}{Z_A(i+1) - Z_A(i)}\right)\Pi_A(i+1) \tag{4.11}$$

The pressure on each 'B'-level `x%pres`$=p_B$ is then recovered as

$$p_B = p_{\text{ref}}(\Pi_B)^{1/\kappa} \tag{4.12}$$

It is also possible to calculate the layer mean virtual temperature across level i given the pressure on each 'B'-level by using the hydrostatic equation to give

$$T_{vB}(i) = \frac{g_{\text{wmo}}}{C_p}\Pi_B(i)\left(\frac{Z_A(i+1) - Z_A(i)}{\Pi_A(i) - \Pi_A(i+1)}\right) \tag{4.13}$$

and the layer mean temperature `x%temp`$=T_B$ is calculated using the specific humidity defined on 'B'-levels `x%shum`$=q$ as

$$T_B = \frac{T_{vB}}{1 + (\varepsilon_w^{-1} - 1)q} \tag{4.14}$$

where ε_w is the ratio of molecular weight of water to that of dry air (0.622).

4.6.2 Input data

The `ropp_fm_roprof2state` and `ropp_fm_state2state_meto` processing assume that the following variables are held in an input RO data structure containing background data from a geopotential height-based model.

- `ro_data%Lev2b%shum` - specific humidity values on each model 'B'-level.
- `ro_data%Lev2b%geop` - geopotential height values on each model 'B'-level.
- `ro_data%Lev2c%geop_sfc` - first element of Z_A, the geopotential height on 'A'-levels. Values of Z_A on other vertical levels are interpolated from Z_B in `ropp_fm_state2state_meto`.
- `ro_data%Lev2c%press_sfc` - first element of p_A, the pressure on 'A'-levels.
- `ro_data%Lev2b%press` - elements 2:`x%n_lev`+1 of p_A, the pressure on 'A'-levels.

Following the call to `ropp_fm_state2state_meto` elements `x%temp`, `x%shum`, `x%pres` and `x%geop` (those of the type `State1dFM` structure used by `ropp_fm`) are all co-located and correspond to data on the model 'B'-levels.

4.7 Defining the observation vector: `ropp_fm_roprof2obs`

When implemented as part of a data assimilation system, `ropp_fm` is required to compute refractivity or bending angle profiles from the background data on the observed height levels. These are unlikely to match the height levels on which the background data are defined.

If observations are read from a file into the ROprof structure, subroutines ropp_fm_roprof2obs1drefrac and ropp_fm_roprof2obs1dbangle can be used to define the observation variables of type Obs1dRefrac and Obs1dBangle respectively.

```
USE ropp_io
USE ropp_fm
TYPE(ROprof)      :: ro_data
TYPE(Obs1dRefrac) :: y
CALL ropp_fm_roprof2obs(ro_data, y)
```

A number of checks on the input data are conducted.

- Check longitude and latitude for background data are within range,
  set y%lon = ro_data%georef%lon
  set y%lat = ro_data%georef%lat
- Check that data arrays are in ascending order. Note that ropp_fm calculations assume that data array index 1 is closest to the surface. The input data are checked and reordered as required using
  call ropp_io_ascend(ro_data)
- Copy observation geopotential heights
  set y%geop = ro_data%Lev2a%geop_refrac
- Copy observed refractivity values
  set y%refrac = ro_data%Lev2a%refrac
- Set default observation weights values
  set y%weights = 1.0

Alternatively, for bending angle observations,

- Check longitude and latitude for background data are within range,
  set y%lon = ro_data%georef%lon
  set y%lat = ro_data%georef%lat
  call ropp_io_ascend(ro_data)
- Copy the radius of curvature
  set y%rcurve = ro_data%georef%roc
- Copy the undulation
  set y%undulation = ro_data%georef%undulation
- Copy the azimuth
  set y%azimuth = ro_data%georef%azimuth
- Check that data arrays are in ascending order. Note that ropp_fmcalculations assume that data array index 1 is closest to the surface. The input data are checked and reordered as required using
- Copy observation impact parameters
  set y%impact = ro_data%Lev1b%impact
- Copy observed bending angle values
  set y%bangle = ro_data%Lev1b%bangle

Elements y%g_sfc and y%r_earth are also set using latitude-dependent routines based on Somagliana's equation Mahoney (2001) provided as part of ropp_utils (ROM SAF, 2007).

The diagonal elements of the observation error covariance (y%cov%d) may also be specified in ropp_fm_roprof2obs by the estimated error associated with the bending angle or refractivity observations.

$$\texttt{y\%cov\%d(i+i*(i-1)/2)} = \texttt{ro\_data\%Lev2a\%refrac\_sigma(i)}^2 \quad \text{for each level } i$$

4.7.1 Set observation geopotential height levels set_obs_levels_refrac

The set_obs_levels_refrac subroutine provided as part of the stand-alone ropp_fm_bg2ro tool provides a simple way of defining the observation geopotential height levels for which the forward modelled refractivity profiles are to be computed. The observation vector geopotential height element y%geop is defined by 300 evenly spaced vertical levels between 200 and 60000 gpm. The refractivity element y%refrac is initialised to zero and the weights y%weights are initialised to unity.

4.7.2 Set observation impact parameter levels set_obs_levels_bangle

Similarly, the impact parameters for the bending angle observation vector y%impact are set to coincide with the same 300 geopotential levels defined for the refractivity observation vector. The geopotential heights Z are converted into geometric heights h using the latitude-dependent conversion

$$h(Z,\phi) = \frac{R_{\text{eff}} Z}{(g/g_{\text{wmo}})R_{\text{eff}} - Z} \tag{4.15}$$

where R_{eff} is the Earth's effective radius and g is the surface gravity, both at latitide ϕ (Mahoney, 2001). This is implemented in the ropp_utils function geopotential2geometric.

Using Equation (3.4) the impact parameter y%impact=a is then defined as

$$a = nr = (1 + 10^{-6}N)(h + R_c) \tag{4.16}$$

where R_c is the radius of curvature of Earth, corrected for the difference between the EGM-96 geoid (NASA/NIMA) and the WGS-84 ellipsoid (NGA), and N is the refractivity at each observation height, computed by a prior call to the refractivity forward model ropp_fm_refrac_1d.

For convenience, other geodetic parameters are also defined in the bending angle observation vector (Table 4.1). These latitude-dependent variables are computed with reference to the WGS-84 ellipsoid using expressions derived from Somagliana's equation (see Mahoney (2001) for more details). The following functions provided as part of the ropp_utils module are used.

Gravity at surface:	obs_bangle%g_sfc	=	gravity(ro_data%GEOref%lat)
Effective Earth radius:	obs_bangle%r_earth	=	r_eff(ro_data%GEOref%lat)
Corrected radius of curvature:	obs_bangle%r_curve_corr	=	ro_data%GEOref%roc

(Users should be aware of the important physical difference between the effective radius, which is used in expressions involving the variation of gravity with height such as (4.15), and the radius of curvature, which is used in expressions involving the ray path geometry such as (4.16). The effective radius is computed within ROPP from the latitude (see App A). It is not part of the `ROprof` data structure and is therefore never output from ROPP. The radius of curvature, on the other hand, can be read in or calculated if necessary, and will be written out (as the variable '`roc`'), because it is held as part of the `ROprof%GEOtype` data substructure.)

4.8 Refractivity forward model `ropp_fm_refrac_1d`

`ropp_fm` computes the refractivity N on observation geopotential height levels by applying Equation (3.8) to background profiles of pressure, humidity and temperature and interpolating the results onto observation heights. Constants k_1, k_2 and k_3 are defined in `ropp_fm_constants`. The input background data are contained in the state vector `x` of type `State1dFM` (Section 4.4) and the output refractivity values are contained in an element of an observation vector `y` of type `Obs1dRefrac` (Section 4.2.1).

```
USE ropp_fm
TYPE(State1dFM)   :: x
TYPE(Obs1dRefrac) :: y
CALL ropp_fm_refrac_1d(x, y)
```

4.8.1 Update variables in State1dFM

Elements of the `State1dFM` structure are computed from state vector `x%state` every time `ropp_fm_refrac_1d` is called to ensure that the temperature, pressure and humidity variables `x%temp`, `x%pres` and `x%shum` are consistent with `x%state`. This becomes necessary when `ropp_fm_refrac_1d` is implemented within `ropp_1dvar` (Chapter 7). The computation depends on the details of the state vector and the background data. Subroutine `ropp_fm_state2state_ecmwf` is used if the background data are on pressure-based vertical levels (Section 4.5) while `ropp_fm_state2state_meto` should be used if the data are on geopotential height-based levels (Section 4.6).

4.8.2 Compute partial water vapour pressure

The water vapour pressure e on each background level is derived from the definition of specific humidity q as

$$q = \frac{r}{1+r}$$

where r is the dry mixing ratio of water vapour defined as

$$r = \varepsilon_w \left(\frac{e}{p-e} \right)$$

This leads to a relationship for e in terms of the background specific humidity x%shum=q and pressure x%pres=p as

$$e = p\left(\frac{q}{\varepsilon_w + (1-\varepsilon_w)q}\right) \tag{4.17}$$

4.8.3 Compute refractivity

Equation (3.8) is applied to compute a refractivity profile $N(Z)$ on background vertical levels from background profiles of pressure x%pres, temperature x%temp and partial water vapour pressure.

4.8.4 Interpolate refractivity to measurement geopotential levels ropp_fm_interpol_log

The refractivity values N computed on the background geopotential height levels Z are interpolated to the required observation heights y%geop to enable direct comparison with the observations. This interpolation is calculated in ropp_fm_interpol_log assuming that refractivity varies exponentially with geopotential height between the background levels.

```
USE ropp_fm
TYPE(State1dFM)   :: x
TYPE(Obs1dRefrac) :: y
CALL ropp_fm_interpol_log (x%geop, y%geop, N, y%refrac)
```

At each observation height k, the output refractivity y%refrac is calculated as

$$\ln(\mathrm{y\%refrac}(k)) = \ln(N_j) + \frac{\mathrm{y\%geop}(k) - \mathrm{x\%geop}(j)}{\mathrm{x\%geop}(j-1) - \mathrm{x\%geop}(j)}(\ln(N_{j-1}) - \ln(N_j)) \tag{4.18}$$

where observation level k lies between background model levels $j-1$ and j.

4.8.5 Refractivity forward model gradient

The ropp_fm module includes routines to compute the gradient (tangent linear) and adjoint of the refractivity forward model with respect to the state vector for use in ropp_1dvar. Corresponding tangent linear and adjoint codes exist for each ropp_fm routine. Further details are provided in Chapter 7.

4.9 Bending angle forward model ropp_fm_bangle_1d

ropp_fm computes a profile of bending angles α as a function of impact parameter a corresponding to the observation geopotential heights used in the refractivity forward model. Bending angles at a given impact parameter are derived by evaluating Equation (3.5) using the refractivity profile computed on background vertical levels using Equation (3.8). The input background data are contained in the state vector x of type State1dFM (Section 4.4) and the output bending angle values are contained in an element of an observation vector y of type Obs1dBangle (Section 4.2.1).

```
USE ropp_fm
TYPE(State1dFM)   :: x
TYPE(Obs1dBangle) :: y
CALL ropp_fm_bangle_1d(x, y)
```

4.9.1 Update variables in State1dFM

As in `ropp_fm_refrac_1d` either `ropp_fm_state2state_ecmwf` or `ropp_fm_state2state_meto` is called to ensure than temperature, pressure and humidity variables `x%temp`, `x%pres` and `x%shum` are consistent with `x%state` before computing the forward model.

4.9.2 Compute refractivity profile

The refractivity N on each background level is calculated following the process described in Section 4.8. This involves the following stages:

- compute partial water vapour pressure on background geopotential height levels using Equation (5.4)
- compute refractivity `refrac` on background geopotential height levels using Equation (3.8)

4.9.3 Compute background impact parameter

The impact parameter corresponding to each background level is computed using the refractivity values N following Equation (3.4)

$$\texttt{impact} = nr = (1 + 1 \times 10^{-6} N)(h + R_c) \tag{4.19}$$

The background geopotential height levels `x%geop` are converted to geometric altitude h using the `ropp_utils` function `geopotential2geometric`.

4.9.4 Calculate bending angles `ropp_fm_abel`

The evaluation of Equation (3.5) in the bending angle forward model is performed in subroutine `ropp_fm_abel`. It computes bending angle `y%bangle` as a function of observation impact parameter `y%impact` from the profile of refractivity values on background impact parameter levels (Healy and Thepaut, 2006). *We restrict the calculation to non-super-refracting conditions.*

```
USE ropp_fm
TYPE(Obs1dBangle) :: y
CALL ropp_fm_abel(impact, refrac, y%impact, y%bangle)
```

Equation (3.5) is simplified by approximating

$$\frac{d\ln(n)}{dx} \approx 10^{-6}\frac{dN}{dx}$$

which is valid because the refractivity is small, and

$$\sqrt{x^2 - a^2} \approx \sqrt{2a(x-a)}$$

This is valid because the refractivity scale height is small compared to the radius of the Earth. This gives

$$\alpha(a) = -\sqrt{2a}10^{-6} \int_a^{\infty} \frac{dN/dx}{\sqrt{x-a}} dx \tag{4.20}$$

Assuming that refractivity varies exponentially with $x = nr$ between background levels leads to an estimate for the refractivity gradient between the j and $j+1$ levels,

$$\frac{dN}{dx} = -k_j N_j \exp(-k_j(x - x_j)) \tag{4.21}$$

where

$$k_j = \frac{\ln(N_j/N_{j+1})}{(x_{j+1} - x_j)}. \tag{4.22}$$

The value of k_j is required to be positive, so a minimum positive value of $k_j^{\min} = 10^{-6}$ m^{-1} is assumed. We also require $x_{j+1} - x_j$ to be at least 10m to avoid problems with large refractivity gradients near super-refracting regions. A further restriction is used that k_j can be no larger than $0.157/N_j$ (i.e. the maximum dN/dx is set to 0.157 N-units/m (the curvature of the earth), so that dN/dr is about half the critical reflection value). This avoids problems encountered in a 4D-Var assimilation system with non-linearity when the atmosphere approaches super-refracting conditions.

Substituting Equation (4.21) into Equation (4.20) leads to an expression for the bending between the j and $j+1$ levels,

$$\Delta\alpha_j = 10^{-6} k_j N_j \exp(k_j(x_j - a))\sqrt{2a} \int_{x_j}^{x_{j+1}} \frac{\exp(-k_j(x-a))}{\sqrt{x-a}} dx \tag{4.23}$$

The lowest usable level to compute bending angle is defined as that level where the impact parameter first decreases with decreasing height towards the surface. This will occur in conditions of super-refraction.

By change of variables,

$$\Delta\alpha_j = 10^{-6}\sqrt{2\pi a k_j} N_j \exp(k_j(x_j - a)) \left[\mathrm{erf}\left(\sqrt{k_j(x_{j+1} - a)}\right) - \mathrm{erf}\left(\sqrt{k_j(x_j - a)}\right) \right] \tag{4.24}$$

where the error function `erf` is defined as

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2) dt \tag{4.25}$$

The error function terms are computed in `ropp_fm_abel` using the following polynomial approximation to `erf` (Abramowitz and Stegun, 1965):

$$\mathrm{erf}(x) \approx 1 - (a_0 + a_1 t + a_2 t^2) t \exp(-x^2) \tag{4.26}$$

where $t = 1/(1+bx)$ and

$$a_0 = 0.3480242; \quad a_1 = -0.0958798; \quad a_2 = 0.7478556; \quad b = 0.47047.$$

The total bending angle y%bangle at a certain impact parameter y%impact=a is then found in ropp_fm_abel by summing contributions calculated using Equation (4.24) between a and the top of the background profile. Ray bending above the model top is accounted for by extrapolating when $j+1$=x%n_lev and evaluating

$$\Delta\alpha_{top} = 10^{-6}\sqrt{2\pi a k_j} N_j \exp(k_j(x_j - a)) \left[1 - \mathrm{erf}\left(\sqrt{k_j(x_j - a)}\right)\right] \tag{4.27}$$

4.9.5 Bending angle forward model gradient

The ropp_fm module includes routines to compute the gradient (tangent linear) and adjoint of the bending angle forward model with respect to the state vector for use in the ropp_1dvar 1D–Var module. Corresponding tangent linear and adjoint code exist for each ropp_fm routine. Further details on this code is provided in Chapter 7. If ropp_fm_bg2ro_1d is called without the -f option, it outputs the forward model gradients $\partial N_i/\partial x_j$ and $\partial \alpha_i/\partial x_j$, where $\mathbf{x}$ is the state vector (see Chapter 7). (These are large matrices which take a long time to generate, so most users will probably want to call ropp_fm_bg2ro_1d -f.)

4.10 Copy simulated observations to RO structure ropp_fm_obs2roprof

The computed refractivity and bending angle observation vectors are copied to the Level2a and Level1b parts of the ROprof data structure respectively for writing to the output netCDF data file using ropp_fm_obs2roprof.

```
USE ropp_io
USE ropp_fm
TYPE(State1dFM)  :: ro_data
TYPE(Obs1dRefrac) :: obs_refrac
TYPE(Obs1dBangle) :: obs_bangle
CALL ropp_fm_obs2roprof(obs_refrac, ro_data)
CALL ropp_fm_obs2roprof(obs_bangle, ro_data)
```

The results are written to the specified output file using the ropp_io module routine ropp_io_write.

4.11 Plotting tools

The directory tests/ contains the IDL procedure plot_fm.pro which gives an example of how to read and plot refractivity and bending angle profiles computed by running the forward model.

An example of its implementation, using archive data available from the ROM SAF archive (http://www.romsaf.org), is provided by running the test script test_fm_GRAS.sh.

4.12 Test software tools

In addition to the main stand-alone simulation tool `ropp_fm_bg2ro_1d`, we provide three additional simple test programs that the user may find instructive. These are in the directory `tests/` and are called:

- `t_fascod`: a simple call to the 1D refractivity and bending angle operators and comparison with pre-calculated profiles.
- `t_fascod_tl`: a test of the 1D operator tangent linear code.
- `t_fascod_ad`: a test of the 1D operator adjoint code.

Note these programs use the `ropp_io` module for reading the test data.

4.12.1 `t_fascod`

This simple program reads meteorological and observation data from the file `data/FASCOD_Scenarios.nc`, simulates refractivity and bending angle profiles with the 1D operators, `ropp_fm_refrac_1d` and `ropp_fm_bangle_1d`, and compares the results with pre-calculated profiles. If the fractional differences are greater than 0.0001 a failure message is written to screen.

4.12.2 `t_fascod_tl`

This program provides an example of how to call and test the tangent linear of the 1D refractivity and bending angle codes, `ropp_fm_refrac_1d_tl` and `ropp_fm_bangle_1d_tl`. For example, the 1D operator is called to compute simulated observations. A set of random perturbations between 0 and 1 are stored in a `x_tl` variable structure of type `Obs1dBangle` (the geopotential perturbation is increased a factor of 100). The code then loops through the following procedures 15 times. Firstly, the perturbations are added to the original state:

```
x_new%temp(:) = x%temp(:) + x_tl%temp(:)
x_new%pres(:) = x%pres(:) + x_tl%pres(:)
x_new%geop(:) = x%geop(:) + x_tl%geop(:)
x_new%shum(:) = x%shum(:) + x_tl%shum(:)
```

and the 1D operator is called with the perturbed state vector. The tangent linear routine is then called with the original vector and the perturbation

```
CALL ropp_fm_bangle_1d(x_new,y_new)
CALL ropp_fm_bangle_1d_tl(x,x_tl,y,y_tl)
```

The code then compares the change in bending angle produced by the tangent linear routine, `y_tl%bangle` with the finite difference approximation `y_new%bangle(:) - y_save%bangle(:)`, calculating the cosine

of the angle between the vectors (the dot product divided by the magnitude of the vectors) and evaluating the relative error. These are then output to screen.

The perturbations are then reduced by a factor of 10 and the procedure is repeated. Ideally, the cosine of the angle between the vectors should tend to unity as the size of the perturbation is reduced, but it does not reach unity because of numerical error. However, how close it gets to unity is a good test of the operator/tangent linear consistency. A failure message is written to screen if the tangent linear test does not converge towards unity.

4.12.3 `t_fascod_ad`

The `t_fascod_ad` program tests the consistency of the tangent linear and adjoint code, routines `ropp_fm_refrac_1d_tl` and `ropp_fm_refrac_1d_ad`, and routines `ropp_fm_bangle_1d_tl` and `ropp_fm_bangle_1d_ad`, respectively. For example, the 1D operator is called to compute simulated bending angles.

```
CALL ropp_fm_bangle_1d(x,y)
```

A set of random perturbations between 0 and 1 are stored in a `x_tl` variable structure of type `Obs1dBangle` (the geopotential perturbation is increased a factor of 100) and the tangent linear routine is called.

```
CALL ropp_fm_bangle_1d_tl(x,x_tl,y,y_tl)
```

We compute the matrix expression $\Delta\mathbf{y}^{\mathbf{T}}\Delta\mathbf{y}$, where $\Delta\mathbf{y} = \mathbf{H}\Delta\mathbf{x}$, given $\mathbf{H}$ is the matrix representing the linearised operator and $\Delta\mathbf{x}$ is the perturbation to the state.

```
norm1 = DOT_PRODUCT(y_tl%bangle(:),y_tl%bangle(:))
```

An adjoint of the state variables `x_ad` of type `State1dFM` is defined and initialised to 0. An adjoint of the bending angle value `y_ad` is initialised to the output of the tangent linear code. The adjoint of the 1D operator routine is then called.

```
y_ad%bangle(:) = y_tl%bangle(:)
CALL ropp_fm_bangle_1d_ad(x,x_ad,y,y_ad)
```

We then define `norm2` in terms of `x_ad` and the original perturbation `x_tl`. Mathematically, `norm2` is equivalent to the matrix expression $\mathbf{x}_{ad}^{\mathbf{T}}\Delta\mathbf{x}$, where $\mathbf{x}_{ad} = \mathbf{H}^{\mathbf{T}}\Delta\mathbf{y}$ and is the output of the adjoint routine. If the adjoint and tangent linear routines are consistent, by definition, `norm1=norm2`. The routine outputs `norm1, norm2` and `norm1/norm2` to screen. A failure message is written to screen if the ratio of norms is not close to unity.

References

Abramowitz, M. and Stegun, I. A., eds., *Handbook of mathematical functions*, Dover, 1965.

ECMWF, IFS documentation - Cy31r1. Part III: Dynamics and numerical procedures, IFS documentation, ECMWF,
http://ecmwf.int/research/ifsdocs/CY31r1/index.html, 2007.

Healy, S. B. and Thepaut, J.-N., Assimilation experiments with CHAMP GPS radio occultation measurements, *Quart. J. Roy. Meteorol. Soc.*, *132*, 605–623, 2006.

Mahoney, M. J., A discussion of various measures of altitudes,
http://mtp.jpl.nasa.gov/notes/altitude/altitude.html, 2001.

NASA/NIMA, NASA and NIMA joint geopotential model EGM96 website,
http://cddis.nasa.gov/926/egm96/egm96.html.

NGA, National Geospatial-Intelligence Agency WGS84 website,
http://earth-info.nga.mil/GandG/wgs84/index.html.

ROM SAF, Geodesy calculations in ROPP, SAF/GRAS/METO/REP/GSR/002, 2007.

ROM SAF, The Radio Occultation Processing Package (ROPP) User Guide. Part I: Input/Output module, SAF/ROM/METO/UG/ROPP/002, Version 7.0, 2013.

Simmons, A. J. and Burridge, D. M., An energy and angular-momentum conserving vertical finite-difference scheme and hybrid vertical coordinates, *Mon. Wea. Rev.*, *109*, 758–766, 1981.

5 ROPP two-dimensional forward model

5.1 Background

The bending angle and refractivity forward models described in Chapter 4 are one-dimensional. This means that the simulated observations are computed using NWP information at a single location and this can introduce "horizontal gradient errors". The use of two-dimensional operators can reduce these forward model errors, particularly in the lower troposphere. The ROPP forward model module, `ropp_fm`, contains a two-dimensional (2D) bending angle forward model, (`ropp_fm_bangle_2d`). This model simulates bending angles from co-located NWP profiles of pressure, temperature, humidity and geopotential height, extracted at a series of points within a "2D occultation plane". The 2D occultation plane is defined in terms of the latitude and longitude of the occultation point, and an azimuthal angle relative to north, which is provided with the observation. The geometry of the measurement within the 2D occultation plane is illustrated in Figure 5.1.

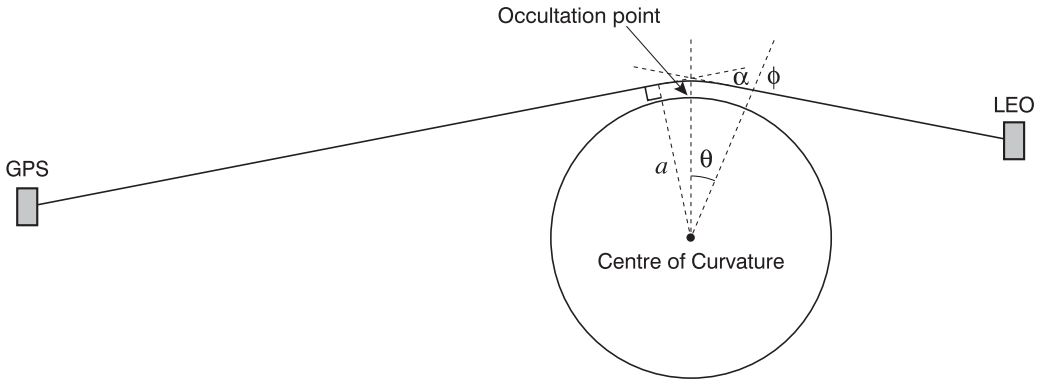

Figure 5.1: Illustrating the geometry of the GNSS-RO measurement.

The 2D forward model has been tested in extended forecast impact experiments with the ECMWF NWP assimilation system. Results at ECMWF suggest that the standard deviations of observed minus background bending angle departures in the lower troposphere can be reduced by around 7 % with a 2D operator. Further details are provided by Healy et al. (2007).

5.2 Theoretical basis of 2D operator

The 2D bending angle operator is more complex than the 1D version because the radial refractivity gradients, $\frac{\partial n}{\partial r}$, vary as a function of horizontal position, θ, within the 2D occultation plane. In this more general case, the ordinary differential equations defining the two-dimensional ray-path in circular polar co-ordinates (r and θ) are given by (page 149, Rodgers, 2000) by

$$\frac{dr}{ds} = \cos\phi \tag{5.1}$$

$$\frac{d\theta}{ds} = \frac{\sin\phi}{r} \tag{5.2}$$

$$\frac{d\phi}{ds} = -\sin\phi\left[\frac{1}{r} + \frac{1}{n}\left(\frac{\partial n}{\partial r}\right)_\theta\right] + \frac{\cos\phi}{nr}\left(\frac{\partial n}{\partial \theta}\right)_r \tag{5.3}$$

where s is the distance along the ray-path, n is the refractive index, ϕ is angle between the local radius vector and the tangent to the ray-path. The ray path equations are integrated numerically.

The ROPP 2D bending angle operator applies a 4th order Runge-Kutta approach to solve the ray path equations where the horizontal gradients are expected to be large in the lower and mid-troposphere, but reverts to a 1D calculation from the upper-troposphere upwards. This "hybrid" approach reduces the computational costs, but note that the user is able to perform a fully 2D calculation if required.

The integration of the ray path equations starts from the assumed tangent point location, rather than one of the satellites. Furthermore, the user should be aware of two approximations currently used in the 2D bending angle calculation:

1. Tangent point drift is neglected, meaning the tangent points of all the bending angles are assumed to have the same horizontal location.
2. The observed impact parameter is used to determine the tangent point height.

These will be addressed in future releases of the 2D operator.

5.3 ROPP 2D forward model tool

A stand-alone tool `ropp_fm_bg2ro_2d` is provided in `ropp_fm` as an illustration of how the `ropp_fm` routines can be implemented to derive profiles of bending angle with the 2D operator from background meteorological data. This is based on the `ropp_fm_bg2ro_1d` stand-alone tool and it provides a convenient template for describing the 2D operator code. Figure 5.2 shows how the `ropp_fm` routines are integrated in the `ropp_fm_bg2ro_2d` code.

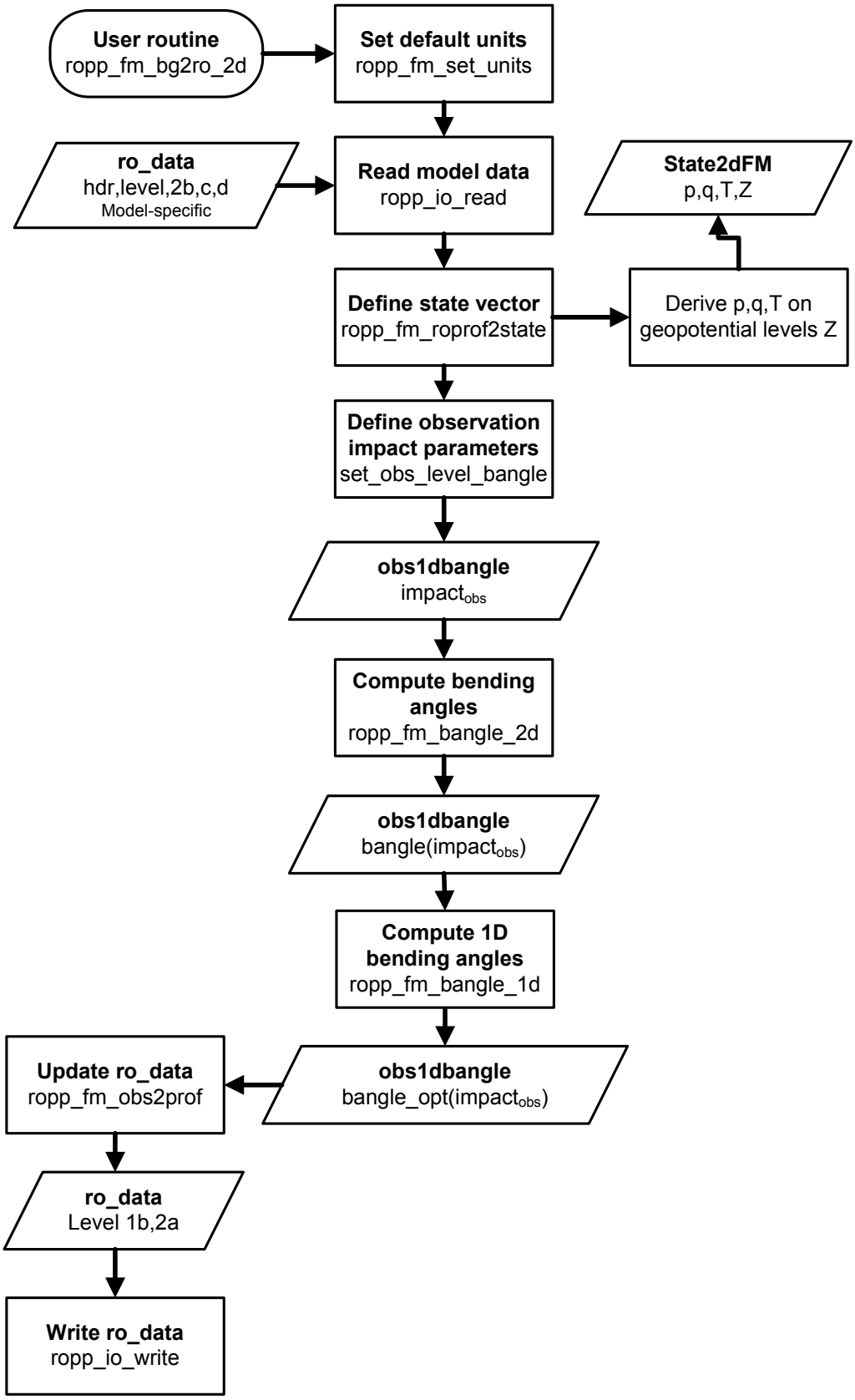

Figure 5.2: Flow chart illustrating calling tree of the ROPP forward model to compute bending angle profiles from input background model data using the 2D operator.

5.3.1 Implementation

The ropp_fm_bg2ro_2d tool is run using the command

```
ropp_fm_bg2ro_2d <inputdatafile> -o <outputfile>
```

where <inputdatafile> is a ROPP netCDF file (ROM SAF, 2013) containing the input model data and <outputfile> will contain the forward modelled bending angle profiles on pre-defined observation levels.

If the input file is a multi-file containing more than one model profile, or more than one input file is specified, ropp_fm_bg2ro_2d computes the forward model for each profile in turn and an output file is generated containing all the output profiles.

The following command line options can be used with the ropp_fm_bg2ro_2d tool:

-o <outputfile>	ROPP netCDF file for output retrieved profiles
-comp	use non-ideal gas compressibility options in forward model
-check_qsat	include check against saturation in forward model
-d	output additional diagnostic information (VerboseMode)
-h	give help menu
-v	output version information

5.3.2 Code organisation

Figure 5.2 shows how the ropp_fm_bg2ro_2d tool is composed of the following stages:

- **Input data access and transformation to a generic state vector**

  Setup the input data arrays, read the input data and define a 2D state vector structure State2dFM containing the background pressure p, temperature T and humidity q data as a function of geopotential height Z. If check_qsat is in force, q is limited by the saturation value at that temperature and pressure; otherwise, by default, supersaturation is allowed. See Section 5.5.

- **Define the observation levels for which the bending angles are to be calculated**

  This stage follows from the requirement of the forward model to compute results on levels corresponding to the observation heights for a given occultation. The observation vector Obs1dBangle is either defined using the observation heights specified in the input file, if these exist, or with default levels specified using subroutine set_obs_levels_bangle.

- **Compute bending angle profile**

  ropp_fm_bangle_2d is 2D forward model routine to compute bending angle α as a function of impact parameter a. See Section 5.7.

- **Write results to generic RO data structure and output file**

5.4 Data types

The observation state vector `y` and background state vector `x` are represented within ROPP by Fortran 90 derived types (or structures). These data structures are defined in `ropp_fm_types`.

5.4.1 Observation vector: `Obs1DBangle`

Bending angle profiles are represented by the `Obs1DBangle` data structure. This structure has been appended to include new variables required for the 2D code. The elements are listed in Table 4.1 (Section 4.2.1).

5.4.2 State vector: `State2dFM`

Background meteorological data are represented by the `State2dFM` data structure. Its elements are listed in Table 5.1. Note that `x%lat` and `x%lon` are arrays (rather than single values as in the 1D case) and they contain the locations of `x%n_horiz` profiles in the 2D occultation plane. The `x%n_horiz` profiles are equally spaced in the plane, with an angular spacing of `x%dtheta` radians. `x%n_horiz` should be an odd number, with the central location corresponding to the location that would be used in a 1D calculation. `ropp_fm` includes a tool `ropp_2d_plane` for calculating the latitude and longitude of a series of equally spaced points in a plane which has been defined by `y%lat`, `y%lon` and `y%azimuth`. The tool can be called with

```
USE ropp_fm
INTEGER :: ierror
TYPE(State2dFM) :: x
TYPE(Obs1DBangle) :: y
CALL ropp_2d_plane(y%lat,y%lon,y%azimuth,x%dtheta,x%n_horiz,x%lat,x%lon,ierror)
```

`ierror` is a returned error flag, with a non-zero value denoting a failure in the routine.

Two-dimensional arrays of temperature `x%temp(:,:)`, pressure `x%pres(:,:)` and humidity `x%shum(:,:)` are stored together with the corresponding geopotential height levels `x%geop(:,:)`. The number of background vertical levels is stored in the `x%n_lev` element and, as noted, the number of horizontal locations in the plane is `x%n_horiz`. So, for example the temperature information in the occultation plane is stored as a 2D array of size `x%temp(n_lev,n_horiz)`.

A structure holding the 2D background data can be declared as

```
USE ropp_fm
TYPE(State2dFM) :: x
```

Further details are provided in Section 4.4. Elements `x%ak` and `x%bk` are required for this mapping if `State2dFM` is holding a state vector using hybrid vertical levels (e.g. ECMWF) as described in Section 4.5.

State2dFM			
Structure element	Description	Range	Units
...%temp	Temperature	[150 350]	K
...%shum	Specific humidity	[0 0.05]	kg/kg
...%pres	Pressure	[10 110000]	Pa[a]
...%geop	Geopotential height	[-1000 100000]	gpm
...%ak	ECMWF-specific level coefficient	[0 200000]	Pa[a]
...%bk	ECMWF-specific level coefficient	[0 2]	
...%n_lev	Number of vertical model levels	[1 100]	
...%n_horiz	Number of horizontal locations	[1 101]	
...%dtheta	Angular separation of profiles	[0 1.5707]	Radians
...%lon	Longitude	[-180 180]	° E
...%lat	Latitude	[-90 90]	° N
...%time	Time	[]	Jul sec
...%state_ok	Flag to specify state vector ok	[0 1]	
...%non_ideal	Non-ideal gas flag	[0 1]	

[a] Note that pressure variables in State2dFM are in units of Pa while standard units in the ROprof structure is hPa. See ROM SAF (2013) for details. Users must specify pressure variables in Pa or ensure unit conversion is carried out before performing ropp_fm calculations if using ropp_io for reading input data. Routine ropp_fm_set_units is provided for this task.

Table 5.1: Elements of the State2dFM vector structure

5.5 Defining the state vector: ropp_fm_roprof2state

The ropp_io module routine ropp_io_read reads a single profile of model background data from a netCDF ROPP netCDF input file and fills the elements of the generic ROPP data structure type ROprof2d (ROM SAF, 2013).

```
USE ropp_io
TYPE(ROprof2d) :: ro_data
CALL ropp_io_read(ro_data, inputfilename, rec=iprofile)
```

The relevant background model data are copied to the state vector x of data type State2dFM by calling the routine ropp_fm_roprof2state.

```
USE ropp_io
USE ropp_fm
TYPE(ROprof2d)  :: ro_data
TYPE(State2dFM) :: x
CALL ropp_fm_roprof2state(ro_data, x)
```

A number of checks on the input data are also conducted at this stage. The state vector x%state_ok flag is set to false if any test fails. See Section 4.4 for further details.

The state vector is required to contain temperature, specific humidity and pressure data as a function of geopotential height (Table 5.1). These data might be read in directly from the input file if the user

provides the background data in this form. It is more likely however that elements of the state vector need to be calculated from the available background data provided by a user and the exact specification of the vertical level representation adopted in the NWP model. The type of model data contained in the input file is specified by the input variable `ro_data%Lev2d%level_type`. `ropp_fm` contains 2D routines to derive the required generic state vector using background data from a NWP model where the vertical coordinate is based on pressure levels (e.g. ECMWF, Section 4.5). Users may prefer to implement their own functions to cater for the specific background vertical level type and available data provided, as these routines are usually part of a data assimilation system.

5.6 Defining the observation vector: `ropp_fm_roprof2obs`

When implemented as part of a data assimilation system, `ropp_fm` is required to compute bending angle profiles from the background data on the observed impact parameter levels.

If observations are read from a file into the `ROprof2d` structure, the routine `ropp_fm_roprof2obs` can be used to define the observation variables of type `Obs1dBangle`. See Section 4.7 for further details.

5.6.1 Set observation impact parameter levels `set_obs_levels_bangle`

The `set_obs_levels_bangle` subroutine provided as part of the stand-alone `ropp_fm_bg2ro` tool provides a simple way of defining the observation impact parameters for which the forward modelled bending angle parameters profiles are to be computed. The observation vector impact parameter element `y%impact` is defined by 250 evenly spaced vertical levels. The lowest impact height is 2700 m and the spacing is 200 m. The bending angle values `y%bangle` are initialised to zero.

5.7 Bending angle forward model `ropp_fm_bangle_2d`

The `ropp_fm` module computes a profile of bending angles α as a function of impact parameter a using a 2D observation operator (Healy et al., 2007).

The input background data are contained in the state vector `x` of type `State2dFM` (Section 4.4) and the output bending angle values are contained in an element of an observation vector `y` of type `Obs1dBangle` (Section 4.2.1).

```
USE ropp_fm
TYPE(State2dFM)   :: x
TYPE(Obs1dBangle) :: y
CALL ropp_fm_bangle_2d(x, y)
```

The main steps of 2D the operator are illustrated in Figure 5.3 and they can be summarised as follows:

1. Compute the water-vapour partial pressure on the (2D grid) model levels.

2. Compute the refractivity, N, on the model levels.

3. Compute the geometric height, h, and radius, r, values the model levels.

4. Compute the product of refractive index and radius, nr, on the model levels.

5. Compute the bending angles, α, as a function of observed impact parameter, a.

Note that steps 1 to 4 are a straightforward generalisation of the 1D code. Essentially, the 1D profile calculations are performed at the `n_horiz` horizontal locations in the plane. However, the details of the calculations will be repeated here for clarity.

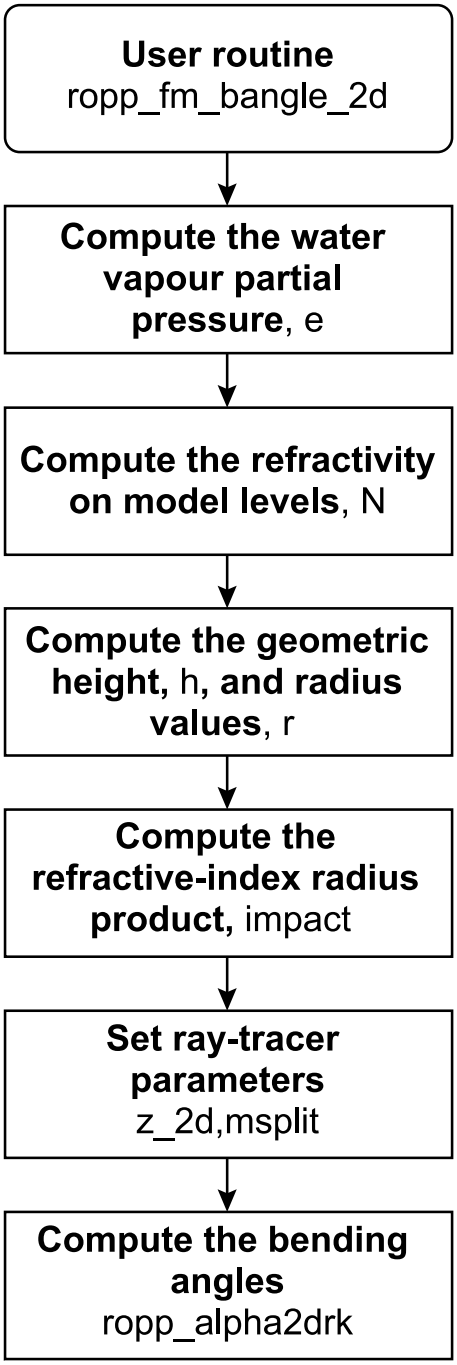

Figure 5.3: Flow chart illustrating the main components of the 2D bending angle operator.

5.7.1 Compute partial water vapour pressure

The water vapour pressure e on each model level in the 2D plane is derived from the definition of specific humidity q as

$$q = \frac{r}{1+r}$$

where r is the dry mixing ratio of water vapour defined as

$$r = \varepsilon_w \left(\frac{e}{p-e} \right)$$

This leads to a relationship for e in terms of the background specific humidity `x%shum`$=q$ and pressure `x%pres`$=p$ as

$$e = p \left(\frac{q}{\varepsilon_w + (1-\varepsilon_w)q} \right) \tag{5.4}$$

5.7.2 Compute refractivity

The refractivity N on the model levels from background profiles of pressure `x%pres`, temperature `x%temp` and partial water vapour pressure using

$$N = k_1 \frac{P}{T} + k_2 \frac{e}{T^2} + k_3 \frac{e}{T} \tag{5.5}$$

The refractivity constants k_1, k_2 and k_3 are defined in `ropp_fm_constants`.

5.7.3 Compute geometric heights and radius values

The geopotential heights Z are converted into geometric heights h using the latitude-dependent conversion

$$h(Z,\phi) = \frac{R_{\text{eff}} Z}{(g/g_{\text{wmo}})R_{\text{eff}} - Z} \tag{5.6}$$

where R_{eff} is the Earth's effective radius and g is the surface gravity, both at latitide ϕ (Mahoney, 2001). The radius values are given by

$$r = h + u + R_c \tag{5.7}$$

where u is the undulation and R_c is the radius of curvature.

5.7.4 Compute background impact parameter

The impact parameter (or refractive-index radius product) corresponding to each model level in the 2D plane is computed using the refractivity values N

$$\texttt{impact} = nr = (1 + 1 \times 10^{-6} N) r \tag{5.8}$$

5.7.5 Calculate bending angles `ropp_fm_alpha2drk`

Two parameters are currently "hardwired" in the `ropp_fm_bangle_2d` code. The integer `msplit=4` is used to govern the step-size used in the ray-tracer. Increasing `msplit` reduces the step size and increases the cost of the operator. The current setting appears to reasonable with ECMWF and Met Office vertical grids. The real `z_2d` defines the height above which the bending is calculated with a 1D approximation. This is currently set to 10 km.

The solution of Equations (5.1)–(5.3) in the bending angle forward model is performed in subroutine `ropp_fm_alpha2drk`. It computes bending angle `y%bangle` as a function of observation impact parameter `y%impact` from the 2D planar refractivity values. This is called within `ropp_fm_bangle_2d`

```
USE ropp_fm
TYPE(State2DFM) :: x
TYPE(Obs1dBangle) :: y
CALL ropp_fm_alpha2drk(y%nobs, x%n_lev, x%n_horiz, msplit, x%dtheta, y%impact, &
                       refrac, rad, impact, y%r_curve, z_2d, a_path, y%bangle, y%rtan)
```

The variables `a_path` and `y%rtan` are the simulated values of $nr\sin\phi$ values at the ray end-points and the tangent radius values, respectively.

As noted earlier, the user should be aware of two approximations currently used in the 2D bending angle calculation:

1. Tangent point drift is neglected, meaning the tangent points of all the bending angles are assumed to have the same horizontal location.
2. The observed impact parameter is used to determine the tangent point height.

The computation of the bending angles within `ropp_fm_alpha2drk` proceeds as follows. The horizontal location of the tangent point is initialised to the middle profile of the 2D plane (`ikcen`). The radius of the ray tangent point is derived from the observed impact parameter, a, and the background (model) refractivity values at the central profile using $r_t = a/(1+10^{-6}N)$. Note that if the tangent point of the ray is above user-prescribed `z_2d`, the 2D ray tracer defaults to a pure 1D calculation of the bending angle. However, if the tangent point of the ray is below the user-prescribed `z_2d`, we estimate the ray bending for the section of path from the tangent point up to a height of `z_2d` with a numerical, raytracing solution of the of the differential equations that define the ray path. The ray tracer starts the calculation at the assumed tangent point, rather than one of the satellites. To evaluate the total bending, we have to compute the bending associated with the ray path on both sides of the tangent point and then add the two values together.

The bending along either side of the tangent point is calculated as follows (Figure 5.4). A four element

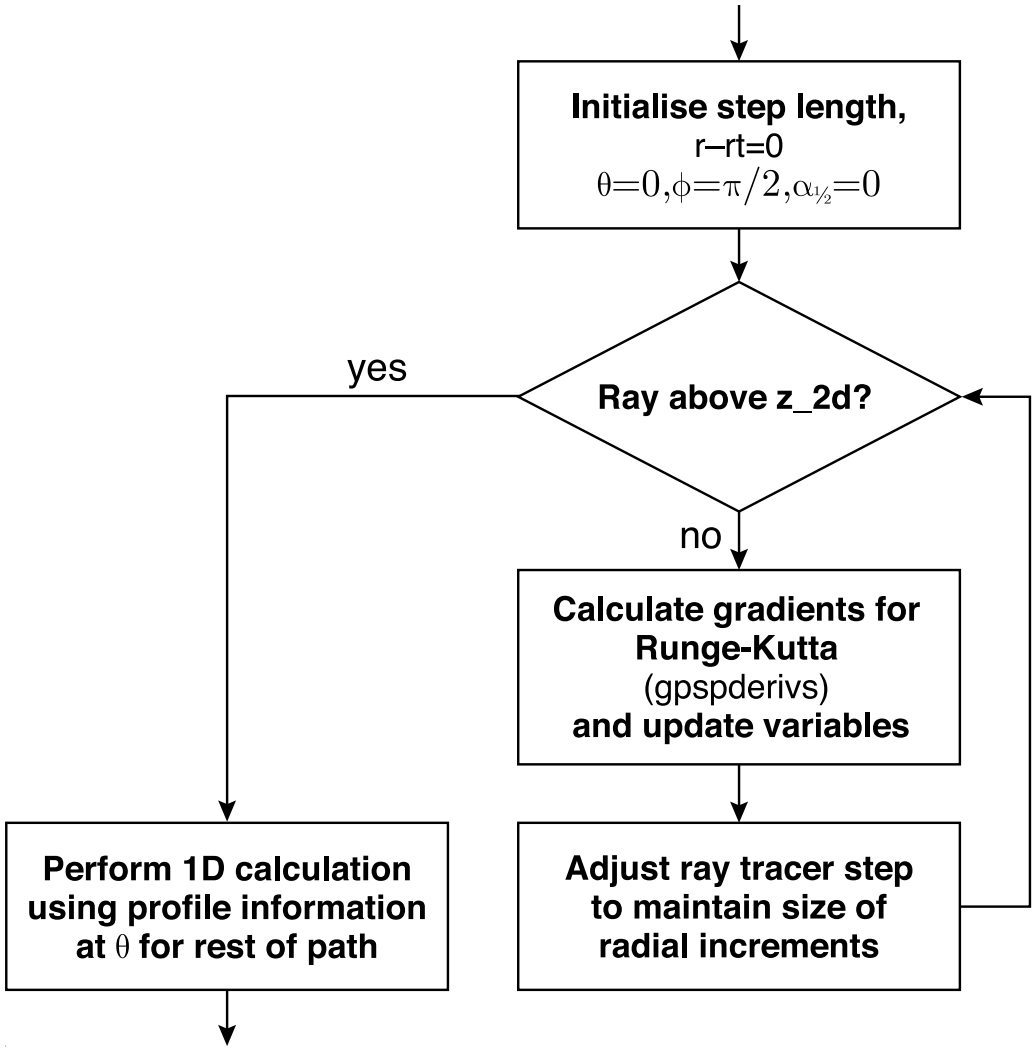

Figure 5.4: Flow chart illustrating the calculation of the bending angles with the Runge-Kutta.

vector $\mathbf{zy}(:)$ containing the following variables $(r - r_t, \theta, \phi, \alpha_{1/2})$ is initialised as

$$\begin{aligned} r - r_t &= 0 \\ \theta &= 0 \\ \phi &= \frac{\pi}{2} \\ \alpha_{1/2} &= 0 \end{aligned} \tag{5.9}$$

where $r - r_t$ is the height above the tangent point, θ is the angular position in the plane, ϕ is the angle between the local radius vector and the ray-path and $\alpha_{1/2}$ is the bending angle along the section of ray path. The subscript "1/2" indicates that this is bending from just one side of the path. Note that we adopt the convention $\theta = 0$ at the horizontal location of the tangent point.

The numerical solution of the ray path equations is performed with a 4th order Runge-Kutta routine (RK4) (e.g., section 16.1, Press et al., 1992). The derivatives of the four variables, at an arbitrary point in the plane (r, θ) are evaluated in `gpspderivs`. This routine calculates the following gradients:

$$\begin{aligned}
\frac{d(r-r_t)}{ds} &= \cos\phi \\
\frac{d\theta}{ds} &= \frac{\sin\phi}{r} \\
\frac{d\phi}{ds} &\simeq -\sin\phi\left[\frac{1}{r}+\left(\frac{\partial n}{\partial r}\right)_\theta\right] \\
\frac{d\alpha_{1/2}}{ds} &= -\sin\phi\left(\frac{\partial n}{\partial r}\right)_\theta
\end{aligned} \tag{5.10}$$

where s is the distance along the ray-path and n is the refractive index. In `gpspderivs` we linearly interpolate the profile information to θ and then assume that refractivity varies exponentially between the model levels to calculate the radial gradient. A maximum refractivity gradient of 0.15 N units/m is also imposed.

As is standard in RK4, there are four calls to `gpspderivs` per integration step (eq. 16.1.3, Press et al., 1992), and then a weighted average of the gradient is calculated. This weighted average gradient is used to project the `zy(:)` vector forward one step. The code is set up so that there are `msplit=4` steps between vertical model levels, and the step lengths `zh` are adjusted to achieve roughly equal radial increments per step between model levels. This entire procedure is repeated until the ray height exceeds the `z_2d`, and then a 1D bending angle calculation is performed for the remainder of the path.

The bending above `z_2d` is estimated by integrating the familiar 1D bending angle equation,

$$\Delta\alpha_{1d}(a) = -a\int_{R_c+z_{2d}}^{\infty}\frac{d\ln n/dx}{(x^2-a^2)^{1/2}}dx \tag{5.11}$$

The integral is evaluated using the nearest model profile to θ, the angular position where the ray height equals `z_2d`. The solution is given in terms of the Gaussian error function (Section 4.9).

5.7.6 Bending angle forward model gradient

The `ropp_fm` module includes routines to compute tangent linear and adjoint of the 2D bending angle forward model with respect to the state vector. These tangent linear and adjoint routines are called `ropp_fm_bangle_2d_tl` and `ropp_fm_bangle_2d_ad`, respectively. Some examples of their implementation are outlined in section 5.12.

5.8 Copy simulated observations to RO structure `ropp_fm_obs2roprof`

The computed bending angle observation vectors are copied to the Level2a and Level1b parts of the `ROprof` data structure respectively for writing to the output netCDF data file using `ropp_fm_obs2roprof` (Section 4.10).

```
USE ropp_fm
TYPE(State2dFM)  :: ro_data
TYPE(Obs1dBangle) :: obs_bangle
```

```
CALL ropp_fm_obs2roprof(obs_bangle, ro_data)
```

5.9 Comparing with a 1D bending angle operator

The ropp_fm_bg2ro_2d tool also calculates bending angles with the 1D bending angle operator to enable comparison between the two forward models. ropp_fm_roprof2state is used to copy the level2a data into a state1dFM structure. The 1D bending angles can then be calculated with ropp_fm_bangle_1d (Section 4.9).

5.10 Writing out data

The results are written to the specified output file using the ropp_io module routine ropp_io_write.

5.11 Plotting tools

The directory tests/ contains the IDL procedure plot_fm_twod.pro which gives an example of how to read and plot refractivity and bending angle profiles computed by running the forward model.

An example of its implementation is provided by running the test script test_fm_2D.sh.

5.12 Test software tools

In addition to the main stand alone simulation tool ropp_fm_bg2ro_2d, we provide three additional simple test programs that the user may find instructive. These are in the directory tests/ and are called:

- t_twodop: a simple call to the 2D operator and comparison with bending angles computed at ECMWF.
- t_twodtl: a test of the 2D operator tangent linear code.
- t_twodad: a test of the 2D operator adjoint code.

These programs do not use the ropp_io modules.

5.12.1 t_twodop

This simple program reads meteorological and observation data from a text file, simulates bending angles with the 2D operator, ropp_fm_bangle_2d, and compares the results with bending angles simulated at ECMWF with the same data and 2D operator. Test data are provided in the text file data/ECMWF_2D_DATA.DAT. If the fractional differences are greater than 0.0001 a failure message is written to screen, along with the simulated bending angle values that have failed the test.

5.12.2 t_twodtl

This program provides an example of how to call and test the tangent linear of the 2D bending angle code, ropp_fm_bangle_2d_tl. The 2D operator is called to compute simulated bending angles, which are saved to a new structure.

```
CALL ropp_fm_bangle_2d(x,y)
y_save%bangle(:) = y%bangle(:)
```

A set of random perturbations between 0 and 1 are stored in a x_tl variable structure of type Obs1dBangle (the geopotential perturbation is increased a factor of 100). The code then loops through the following procedures 15 times. Firstly, the perturbations are added to the original state:

```
x_new%temp(:,:) = x%temp(:,:) + x_tl%temp(:,:)
x_new%pres(:,:) = x%pres(:,:) + x_tl%pres(:,:)
x_new%geop(:,:) = x%geop(:,:) + x_tl%geop(:,:)
x_new%shum(:,:) = x%shum(:,:) + x_tl%shum(:,:)
```

and the 2D operator is called with the perturbed state vector. The tangent linear routine is then called with the original vector and the perturbation

```
CALL ropp_fm_bangle_2d(x_new,y_new)
CALL ropp_fm_bangle_2d_tl(x,x_tl,y,y_tl)
```

The code then compares the change in bending angle produced with of the tangent linear routine, y_tl%bangle with the finite difference approximation y_new%bangle(:) - y_save%bangle(:), calculating the cosine of the angle between the vectors (the dot product divided by the magnitude of the vectors) and evaluating the relative error. These are then output to screen.

The perturbations are then reduced by a factor of 10 and the procedure is repeated. Ideally, the cosine of the angle between the vectors should tend to unity as the size of the perturbation is reduced, but it does not reach unity because of numerical error. However, how close it gets to unity is a good test of the operator/tangent linear consistency. A failure message is written to screen if the tangent linear test does not converge towards unity.

5.12.3 t_twodad

The t_twodad program tests the consistency of the tangent linear and adjoint code, routines ropp_fm_bangle_2d_tl and ropp_fm_bangle_2d_ad, respectively. The 2D operator is called to compute simulated bending angles.

```
CALL ropp_fm_bangle_2d(x,y)
```

A set of random perturbations between 0 and 1 are stored in a `x_tl` variable structure of type `Obs1dBangle` (the geopotential perturbation is increased a factor of 100) and the tangent linear routine is called.

```
CALL ropp_fm_bangle_2d_tl(x,x_tl,y,y_tl)
```

We compute the matrix expression $\Delta\mathbf{y}^{\mathrm{T}}\Delta\mathbf{y}$, where $\Delta\mathbf{y} = \mathbf{H}\Delta\mathbf{x}$, given $\mathbf{H}$ is the matrix representing the linearised operator and $\Delta\mathbf{x}$ is the perturbation to the state.

```
norm1 = DOT_PRODUCT(y_tl%bangle(:),y_tl%bangle(:))
```

An adjoint of the state variables `x_ad` of type `State2dFM` is defined and initialised to 0. An adjoint of the bending angle value `y_ad` is initialised to the output of the tangent linear code. The adjoint of the 2D operator routine is then called.

```
y_ad%bangle(:) = y_tl%bangle(:)
CALL ropp_fm_bangle_2d_ad(x,x_ad,y,y_ad)
```

We then define `norm2` in terms of `x_ad` and the original perturbation `x_tl`. Mathematically, `norm2` is equivalent to the matrix expression $\mathbf{x}_{ad}^{\mathrm{T}}\Delta\mathbf{x}$, where $\mathbf{x}_{ad} = \mathbf{H}^{\mathrm{T}}\Delta\mathbf{y}$ and is the output of the adjoint routine. If the adjoint and tangent linear routines are consistent, by definition, `norm1=norm2`. The routine outputs `norm1, norm2` and `norm1/norm2` to screen. A failure message is written to screen if the ratio of norms is not close to unity.

References

Healy, S. B., Eyre, J. R., Hamrud, M., and Thepaut, J.-N., Assimilating GPS radio occultation measurements with two-dimensional bending angle observation operators, *Quart. J. Roy. Meteorol. Soc.*, *133*, 1213–1227, 2007.

Mahoney, M. J., A discussion of various measures of altitudes, http://mtp.jpl.nasa.gov/notes/altitude/altitude.html, 2001.

Press, W., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical recipes in C – The Art of Scientific Computing*, Cambridge University Press, Cambridge, New York, 2nd edn., 1992.

Rodgers, C. D., *Inverse methods for atmospheric sounding: Theory and practice*, World Scientific Publishing, Singapore, New Jersey, London, Hong Kong, 2000.

ROM SAF, The Radio Occultation Processing Package (ROPP) User Guide. Part I: Input/Output module, SAF/ROM/METO/UG/ROPP/002, Version 7.0, 2013.

6 ROPP 1D–Var: Refractivity

The ROPP 1D–Var module (`ropp_1dvar`) includes routines to retrieve profiles of pressure, temperature and humidity using a measured refractivity profile, the *a priori* knowledge of the state of the atmosphere (i.e. background profiles) and their associated errors. This is achieved in the `ropp_1dvar_solve` subroutine through the minimisation of a quadratic cost function.

$$J(\mathbf{x}) = \frac{1}{2}(\mathbf{x}-\mathbf{x}_b)^T\mathbf{B}^{-1}(\mathbf{x}-\mathbf{x}_b) + \frac{1}{2}(\mathbf{y}_o-\mathbf{H}[\mathbf{x}])^T\mathbf{O}^{-1}(\mathbf{y}_o-\mathbf{H}[\mathbf{x}]) \tag{6.1}$$

In ROPP the state vector $\mathbf{x}$ is part of a Fortran 90 derived structure of type `State1dFM` containing pressure, temperature and humidity data on geopotential height levels. This contains the retrieved atmospheric state. $\mathbf{x}_b$ is the initial background state of type `State1dFM` defined by input background model profiles. Matrix $\mathbf{B}$ defines the error covariance of the background data. $\mathbf{y}_o$ is the observation vector. If the 1D–Var is performed using measured refractivity then $\mathbf{y}_o$ is an observation vector of derived type `Obs1dRefrac` which contains refractivity as a function of geopotential height. The forward modelled observation $\mathbf{H}[\mathbf{x}]$ is also held in an observation vector, and is given by the output of `ropp_fm` routines to compute refractivity (Section 4.8) for a given atmospheric state.

Figure 6.1 shows example pressure, temperature and humidity profiles retrieved from background model data and colocated GNSS-RO refractivity observations.

6.1 ROPP 1D–Var refractivity tool

The stand-alone tool `ropp_1dvar_refrac` is provided in `ropp_1dvar` as an illustration of how the `ropp_1dvar` routines can be implemented to retrieve pressure, temperature and humidity profiles from refractivity observations respectively. Figure 6.2 shows how the `ropp_1dvar` routines are integrated in the `ropp_1dvar_refrac` code.

6.1.1 Implementation

The `ropp_1dvar_refrac` tool is run using the command

```
ropp_1dvar_refrac [options] -o <outputfile>
```

where `<outputfile>` is a ROPP netCDF file (ROM SAF, 2013) which will contain the retrieved temperature, humidity and pressure profiles on model background levels.

The following options can be defined on running the command.

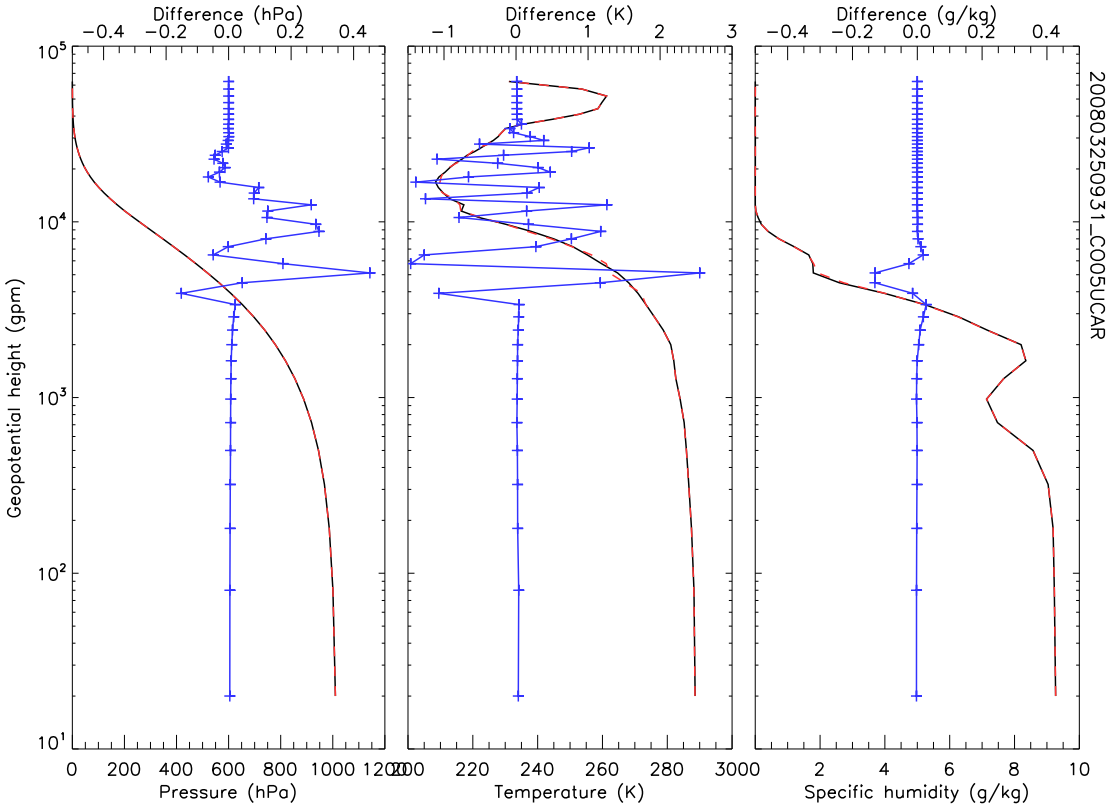

Figure 6.1: Example pressure, temperature and humidity profile retrievals (red) computed using background profiles (plotted in black) and colocated GNSS-RO refractivity observations. The difference between the two profiles are plotted in blue.

`-c <config_file>`	text file specifying configuration options
`-y <obs_file>`	ROPP netCDF observation file
`--obs-corr <obs_corr_file>`	file with observation error covariance parameters
`-b <bg_file>`	ROPP netCDF background data file
`--bg-corr <bg_corr_file>`	netCDF file with background error covariance parameters
`-o <outputfile>`	ROPP netCDF file for output retrieved profiles
`-comp`	use non-ideal gas compressibility options in forward model
`-check_qsat`	include check against saturation in forward model
`-d`	output additional diagnostic information (VerboseMode)
`-h`	give help menu
`-v`	output version information

If the input observation and background data files are multi-files containing more than one profile, the routine `ropp_1dvar_refrac` computes a 1D–Var retrieval for each profile in turn and the output file generated contains all the output profiles.

6.1.2 Code organisation

Figure 6.2 shows how the `ropp_1dvar_refrac` tool is composed of the following stages:

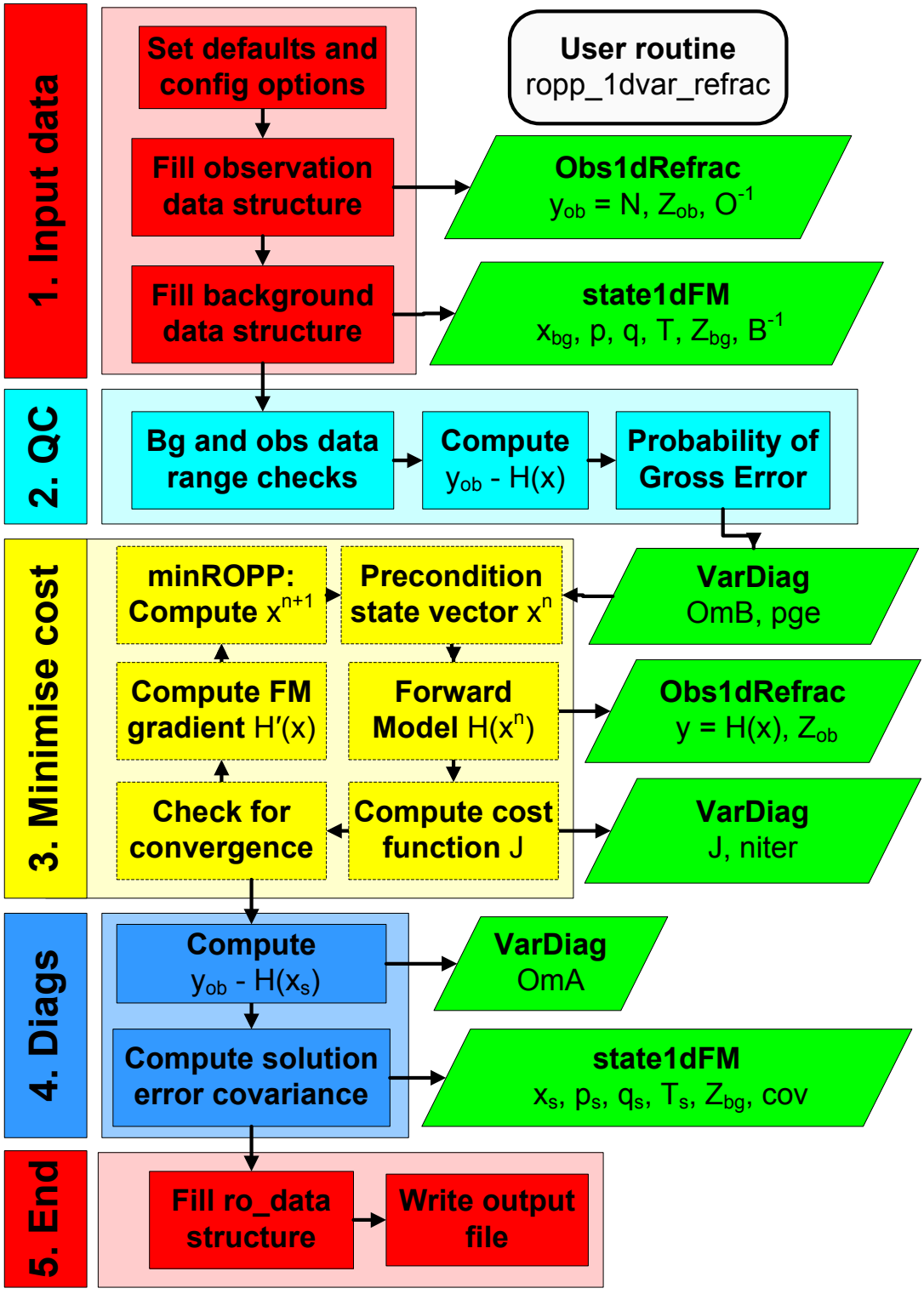

Figure 6.2: Flow chart illustrating the calling tree of the ROPP 1D–Var to retrieve atmospheric profiles from observed refractivity profiles and input background model data.

- **Input data access and transformation to a generic state vector.** (Section 6.3)

  Setup the input data arrays and read the input data into the RO data structure of type `ROprof`. Define a state vector structure `State1dFM` containing the background pressure p, temperature T and humidity q data as a function of geopotential height Z_{bg}. Define an observation vector structure

`Obs1dRefrac` containing the observed refractivity N as a function of geopotential height Z_{ob}. Define the observation error covariance matrix $\mathbf{O}$ and background error covariance matrix $\mathbf{B}$.

- **Perform quality control checks.** (Section 6.6)

  A number of preliminary data quality control checks are performed. This includes setting the required valid height range of the observations required in the 1D–Var analysis. General checks are made that observation and background data are co-located in space and time and that background and observation data are within pre-defined ranges. Diagnostic parameters, which may be utilised as part of a data assimilation system are also computed and stored as part of a structure of type `VarDiag`. The difference between observations and the background state forward modelled into observation space is computed using the `ropp_fm` routines. The probability of gross error (PGE) is also computed.

- **Minimise the cost function.** (Section 6.7)

  `ropp_1dvar_solve` is the main routine for solving the 1D–Var to find the atmospheric state vector $\mathbf{x}$ which minimises the cost function for a given background and observed data and their associated errors. This consists of three stages on each iteration towards a solution:

  - Compute the cost function (Equation 6.1) and its gradient in routine `ropp_1dvar_cost`. This requires re-computing the forward model $\mathbf{H}[\mathbf{x}]$ using `ropp_fm` routines on each iteration using the current state vector $\mathbf{x}$.
  - Call minimiser `ropp_1dvar_minropp` to update the state vector $\mathbf{x}$ towards a solution.
  - Check against pre-defined convergence criteria to identify whether the cost function has been minimised and the optimal solution has been obtained.

- **Compute final diagnostics.** (Section 6.8)

  The deviation between observations and the solution state vector forward modelled into observation space (using `ropp_fm` routines) is computed and stored as an element in the structure of type `VarDiag`. The error covariance of the solution is also computed.

- **Write results to a generic RO data structure and output file.** (Section 6.9)

6.2 Defining observation and background errors

The variational approach requires estimates of the background and observation errors. The `ropp_1dvar` module includes a collection of observation and background error correlation and standard deviation files and tools which users may find helpful for setting up input to `ropp_1dvar` tools. These are available in the `ropp_1dvar/errors/` subdirectory.

ROM SAF (2010) provides an illustration of the available error structures. Users are encouraged to adapt the provided routines to meet their own applications.

A number of tools are provided to add profile-by-profile background or observation error values to a ROPP format file, which may then be used within the 1D–Var routines (see Sections 6.4.2 and 6.5.2). These are summarised below.

Background errors: `ropp_1dvar_add_bgr_error`

```
ropp_1dvar_add_bgr_error <bg_file> -c <cov_file> [-o <out_file>]
```

This tool adds background pressure, temperature and humidity error values to a ROPP format netCDF file `<bg_file>`. The errors are read from the 'sigma' variable in an input background covariance matrix file `<cov_file>` (e.g. `errors/ropp_bg_ecmwf_error_corr_L91.nc`).

Refractivity observation errors: `ropp_1dvar_add_refrac_error`

```
ropp_1dvar_add_refrac_error <obs_file> -Omod <Omodel>
                            -o <out_file> [-b <bg_file>] [-c <corr_file>]
```

This tool adds refractivity observation error values to a ROPP format netCDF file `<obs_file>`. A number of different observational error types may be chosen by the user, as specified using the `-Omod` command line option.

- **1%** : errors 1% at 0 km, 0.1% from 12 km, min($\sigma(N)$)=0.02
- **2%** : errors 2% at 0 km, 0.2% from 12 km, min($\sigma(N)$)=0.02
- **3%** : errors 3% at 0 km, 0.3% from 12 km, min($\sigma(N)$)=0.02
- **TP** : ROM SAF operational implementation. As '2%' model, with a dynamic tropopause height calculation from background data (in the `<bg_file>` file) rather than 12 km.
- **MO** : Met Office operational implementation using pre-defined latitudinally varying percentage errors.
- **SK** : Error model based on Steiner and Kirchengast (2005).

If an output correlation file `<corr_file>` is specified on the command line, the resulting observation correlation and standard deviation values are written to that file. For refractivity observations, the $(n,m)^{\text{th}}$ element of the observation correlation matrix is defined as

$$\text{corr}_{nm} = e^{-|z_n - z_m|/H} \tag{6.2}$$

where z_n and z_m are geopotential heights at points n and m along the profile and H is a characteristic scale height of 3 km. Further details are provided by ROM SAF (2010).

Error correlation files

A number of error correlation data files are also provided in the `ropp_1dvar/errors/` subdirectory for reference and use with the 1D–Var tools.

- **ropp_bg_ecmwf_error_corr_L91.nc** - Background error correlation matrix in packed form for the latest ECMWF model structure (91 levels). This file is suitable for input to the `ropp_1dvar` tools using '–bg-corr' command line option with ECMWF backgrounds. It is recommended that 1D–Var tools are run with bg_covar_method = 'VSFC' config option.
- **ropp_bg_meto_error_corr_L70.nc** - Background error correlation matrix in packed form for the latest Met Office model structure (70 levels). This file is suitable for input to the `ropp_1dvar` tools using '–bg-corr' command line option with Met Office backgrounds. It is recommended that 1D–Var tools are run with bg_covar_method = 'VSFC' config option.
- **ropp_bg_ecmwf_error_corr_L60.nc** - Background error correlation matrix in packed form for the previous ECMWF model structure (60 levels). It is recommended that 1D–Var tools are run with bg_covar_method = 'VSFC' config option.
- **ropp_bg_meto_error_corr_L50.nc** - Background error correlation matrix in packed form for the previous Met Office model structure (50 levels). This file is suitable for input to the `ropp_1dvar` tools using '–bg-corr' command line option with Met Office backgrounds. It is recommended that 1D–Var tools are run with bg_covar_method = 'VSFC' config option.
- **ropp_ob_refrac_error_corr.nc** - Refractivity observation correlation matrix in packed form. For use with 'standard' 247 levels. This file is suitable for input to the `ropp_1dvar_refrac` tool using '–obs-corr' command line option. It is recommended that 1D–Var tools are run with either obs_covar_method = 'VSDC' or 'VSFC' config options.

6.3 Input data

Figure 6.3 illustrates the implementation of `ropp_1dvar` and `ropp_fm` module routines to input background and observation data and associated error covariance matrices into the data structures required by the subsequent 1D–Var processing.

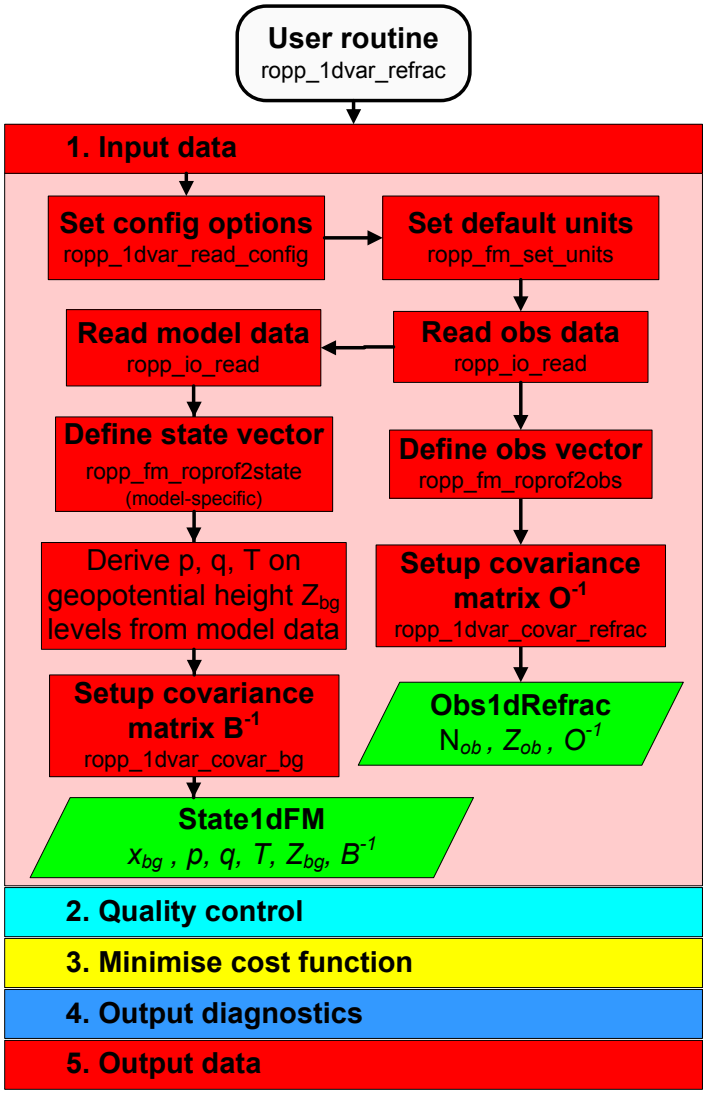

Figure 6.3: Flow chart illustrating the calling tree of the input data step of ROPP 1D–Var to retrieve atmospheric profiles from observed refractivity profiles and input background model data.

Note that the ROPP forward model assumes data are in ascending height order. The input data are checked and reordered as part of the stand-alone tool processing (see 4.4). If required for 1D–Var, users must ensure that the error correlations used are appropriate for background and observation data in ascending height order.

6.3.1 Configuration options

A number of configuration options can be defined by the user in order to tune the performance of the ropp_1dvar retrieval. Table 6.1 lists the configuration options and their default values held in a structure of derived type VarConfig. The use of these parameters within ropp_1dvar are described within this User Guide. A user can specify configuration parameters at run-time by setting their values in a configuration file and including the '-c <config_file>' command line option when running the ropp_1dvar_refrac tool.

The configuration file is read, if specified, and the elements of a variable of type VarConfig are overwritten by calling subroutine ropp_1dvar_read_config.

```
USE ropp_1dvar
TYPE(VarConfig)  :: config
CALL ropp_1dvar_read_config(config_file, config)
```

A number of sample configuration files are included with the ROPP distribution in the ropp_1dvar/config directory.

6.4 Observation data

For refractivities, the routines ropp_1dvar and ropp_fm use observation data defined as elements of a structure of type Obs1dRefrac. (Section 4.2.1.)

The ropp_fm subroutine ropp_fm_set_units is first called to ensure that all variables are specified in the default forward model units before any other ropp_1dvar processing. This utilises the ropp_utils unitconvert library functions. The ropp_io module routine ropp_io_read then reads a single profile of observation data from a netCDF ROPP format input file and fills the elements of the generic ROPP data structure of type ROprof (ROM SAF, 2013).

```
USE ropp_io
USE ropp_fm
USE ropp_1dvar
TYPE(ROprof) :: obs_data
CALL ropp_fm_set_units(obs_data)
CALL ropp_io_read(obs_data, config%obs_file, rec=iprofile)
```

6.4.1 Defining the observation vector: ropp_fm_roprof2obs

The relevant refractivity observation data can be copied to elements of the observation vector $\mathbf{y}$ of type Obs1dRefrac by calling the routine ropp_fm_roprof2obs (Section 4.7.)

6.4.2 Defining the observation error covariance matrix: `ropp_1dvar_covar_refrac`

The subroutine `ropp_1dvar_covar_refrac` is used to set up the observation error covariance matrix for a vector of refractivity observations.

```
USE ropp_fm
USE ropp_1dvar
TYPE(Obs1dRefrac) :: obs
TYPE(VarConfig)   :: config
CALL ropp_1dvar_covar(obs, config)
```

The error covariance matrix $\mathbf{O}$ is constructed by computing the matrix product,

$$\mathbf{O} = \sigma.\mathbf{Corr}.\sigma^T \tag{6.3}$$

where $\mathbf{Corr}$ is a matrix containing the correlation between elements in the observation vector and σ is a diagonal matrix where the diagonal elements contain the error standard deviations for each element in the observation vector. The elements of $\mathbf{O}$ are held in the `Obs1dRefrac` structure for use in the `ropp_1dvar` processing as element `obs%cov`.

The observation error covariances can be constructed using the following methods in ROPP. The method to be used is specified as a configuration option (Table 6.1).

- **FSFC - Fixed Sigmas, Fixed correlations**
  Both error correlations and error standard deviations are read from an observation error correlation file. The error correlation file is specified by the '`--obs-corr <obs_corr_file>`' command-line option or can be set as a default configuration file option. The error correlation file must contain both the error correlation matrix as well as the standard deviations (errors) for all observation vector elements. Sample files containing observation sigma and correlation values are provided in the `errors/` sub-directory of the `ropp_1dvar` distribution (see Section 6.2).
- **VSDC - Variable Sigmas, Diagonal Correlations**
  A diagonal error correlation structure (i.e., no error correlations) is assumed. Error estimates are obtained separately for each input profile by using standard deviation values specified in the input observation data file. Note that the input observation file must contain valid error estimates for all observation vector elements, even if the observation value at a given level is invalid. In this case, no observation error correlation data file is required. Tools for defining variable sigma values in an input profile are provided in the `errors/` sub-directory of the `ropp_1dvar` distribution (see Section 6.2).
- **VSFC - Variable Sigmas, Fixed Correlations**
  Error correlations are read from an error correlation file, while error estimates are obtained separately for each input profile from the standard deviations contained in the input observation data file. Note that the input observation file must contain valid standard deviations for all observation vector elements, even if the observation value at a given level is invalid. The error correlation file is specified by the '`--obs-corr <obs_corr_file>`' command-line option or can be set as a default configuration file option. In this case the error correlation data files only need to contain the error

correlations. Tools for defining variable sigma values in an input profile and sample observation error files are provided in the `errors/` sub-directory of the `ropp_1dvar` distribution (see Section 6.2).

Note that error correlation files may contain latitudinally binned error correlations and standard deviations, allowing for latitudinally varying error correlation structures and standard deviations in the `FSFC` and `VSFC` methods.

When standard deviations are read from the input observation data file using the `VSFC` or `VSDC` methods the diagonal elements of σ are specified in `ropp_fm_roprof2obs` by estimates of the error associated with the refractivity observations.

`obs%cov%d(i+i*(i-1)/2) = ro_data%Lev2a%refrac_sigma(i)`^{2} for each level i

6.5 Background data

6.5.1 Defining the state vector: `ropp_fm_roprof2state`

Background meteorological data are represented in `ropp_fm` and `ropp_1dvar` by the `State1dFM` data structure. The relevant background data are copied to elements of `State1dFM` by calling `ropp_fm_roprof2state` (Section 4.4).

The `State1dFM` structure used by `ropp_fm` routines is required to contain temperature, specific humidity and pressure data as a function of geopotential height. Typically the elements of the state vector need to be calculated from the available background data provided by a user and the exact specification of the vertical level representation adopted in the NWP model from which the data are taken. The type of model data contained in the input file is specified by the input variable `bg_data%Lev2d%level_type`. The `ropp_fm` module contains routines to derive the required generic elements of `State1dFM` using background data from a NWP model where the vertical levels is based on pressure levels (e.g. ECMWF, `ropp_fm_state2state_ecmwf`) or geopotential height (e.g. Met Office, `ropp_fm_state2state_meto`). See Sections 4.5 and 4.6 for further details.

State vector

The elements of the state vector `bg%state` used in the `ropp_1dvar` analysis also depend on the exact details of the source of the background data. Note that if the configuration option `config%use_logp` is set to `true` then pressure variables in the state vector and its covariance are expressed as $\ln(p)$. Similarly, it the configuration option `config%use_logq` is set to `true` then specific humidity variables in the state vector and its covariance are expressed as $\ln(q)$. The appropriate conversions are applied by routines `ropp_fm_roprof2state` and `ropp_fm_state2roprof`.

ECMWF

For background data with hybrid vertical levels (e.g. ECMWF), the elements of the state vector are given by vertical profiles of temperature and specific humidity on the background's vertical levels and an

additional surface pressure element. bg%n_lev is the number of background vertical levels.

For config%use_logp and config%use_logq set to false,

```
bg%temp(:) = bg%state(1 : bgn_lev)
bg%shum(:) = bg%state(bg%n_lev + 1 : 2*bg%n_lev)
psfc       = bg%state(2*bg%n_lev +1)
```

For config%use_logp and config%use_logq set to true,

```
bg%temp(:) = bg%state(1 : bgn_lev)
bg%shum(:) = exp[bg%state(bg%n_lev + 1 : 2*bg%n_lev)]
psfc       = exp[bg%state(2*bg%n_lev +1)]
```

Met Office

For background data with geopotential height-based vertical levels (e.g. Met Office), the elements of the state vector are given by vertical profiles of pressure and humidity on the original background's vertical levels. Note that pressure and humidity variables are stored on different levels of a staggered vertical grid in the Met Office Unified Model, so bg%state contains one more pressure element than specific humidity. bg%n_lev is the number of background vertical levels for humidity data.

For config%use_logp and config%use_logq set to false,

```
pressA(:)  = bg%state(1 : bg%n_lev + 1)
bg%shum(:) = bg%state(bg%n_lev + 2 : 2*bg%n_lev + 1)
```

For config%use_logp and config%use_logq set to true,

```
pressA(:)  = exp[bg%state(1 : bg%n_lev + 1)]
bg%shum(:) = exp[bg%state(bg%n_lev + 2 : 2*bg%n_lev + 1)]
```

The ropp_fm_state2state_ecmwf and ropp_fm_state2state_meto routines allow for consistent mapping between the elements of the state vector and the variables required by ropp_fm. It is therefore called each time the state vector is updated in minimising the cost function for example.

6.5.2 Defining the background error covariance matrix: ropp_1dvar_covar_bg

The subroutine ropp_1dvar_covar_bg is used to set up the background error covariance matrix for a background state vector.

```
USE ropp_fm
USE ropp_1dvar
TYPE(State1dFM)   :: bg
TYPE(VarConfig)   :: config
CALL ropp_1dvar_covar(bg, config)
```

The error covariance matrix $\mathbf{B}$ is constructed by computing the matrix product,

$$\mathbf{B} = \sigma.\mathbf{Corr}.\sigma^T \tag{6.4}$$

where $\mathbf{Corr}$ is a matrix containing the correlation between elements in the state vector and σ is a diagonal matrix where the diagonal elements contain the error standard deviations for each element in the state vector. The elements of $\mathbf{O}$ are held in the `State1dFM` structure for use in the `ropp_1dvar` processing as element `bg%cov`.

The background error covariance matrix can be constructed using the following methods in ROPP. The method to be used is specified as a configuration option (Table 6.1).

- **FSFC - Fixed Sigmas, Fixed correlations**

  Both error correlations and error standard deviations are read from a background error correlation file. The error correlation file is specified by the '`--bg-corr <bg_corr_file>`' command-line option or can be set as a default configuration file option. The error correlation file must contain both the error correlation matrix as well as the standard deviations (errors) for all background state vector elements. Note it is assumed that the standard deviations are input in the required format for the user's choice of `config%use_logp` and `config%use_logq` options. Sample files containing background sigma and correlation values are provided in the `errors/` sub-directory of the `ropp_1dvar` distribution (see Section 6.2).
- **VSFC - Variable Sigmas, Fixed Correlations**

  Error correlations are read from an error correlation file, while error estimates are obtained separately for each input profile from the standard deviations contained in the input background data file (and values are automatically adjusted if the user sets either `config%use_logp` or `config%use_logq`). The error correlation file is specified by the '`--bg-corr <bg_corr_file>`' command-line option or can be set as a default configuration file option. In this case the error correlation data files only need to contain the error correlations. Tools for defining variable sigma values in an input profile and sample background error files are provided in the `errors/` sub-directory of the `ropp_1dvar` distribution (see Section 6.2).

Error correlation files may contain latitudinally binned error correlations and standard deviations, allowing for latitudinally varying error correlation structures and standard deviations even in the `FSFC` scenario.

Note that the error standard deviations are dependent on which variables are used to define each element of the background state vector, so that σ is specific to a particular background model type (Sections 4.5, 4.6).

ECMWF

The diagonal elements of σ for the state vector defined for ECMWF background data are specified by the estimated error associated with each meteorological variable.

For config%use_logp and config%use_logq set to false,

bg%cov%d(i+i*(i-1)/2) = ro_data%Lev2b%temp_sigma(i)^2 for each level i

bg%cov%d(j+j*(j-1)/2) = ro_data%Lev2b%shum_sigma(i)^2 for each level $j = \text{bg\%n\_lev} + i$

bg%cov%d(2*bg%n_lev + 1) = ro_data%Lev2b%press_sfc_sigma

For config%use_logp and config%use_logq set to true,

bg%cov%d(i+i*(i-1)/2) = ro_data%Lev2b%temp_sigma(i)^2 for each level i

bg%cov%d(j+j*(j-1)/2) = (...%shum_sigma(i)/...%shum(i))^2 for each level $j = \text{bg\%n\_lev} + i$

bg%cov%d(2*bg%n_lev + 1) = (...%press_sfc_sigma/...%press_sfc)^2

Met Office

The diagonal elements of σ for the state vector defined for Met Office background data are specified by the estimated error associated with each meteorological variable.

For config%use_logp and config%use_logq set to false,

bg%cov%d(i+i*(i-1)/2) = ro_data%Lev2b%press_sigma(i)^2 for each level i

bg%cov%d(j+j*(j-1)/2) = ro_data%Lev2b%shum_sigma(i)^2 for each level $j = \text{bg\%n\_lev} + i$

For config%use_logp and config%use_logq set to true,

bg%cov%d(i+i*(i-1)/2) = (...%press_sigma(i)/...%press(i))^2 for each level i

bg%cov%d(j+j*(j-1)/2) = (...%shum_sigma(i)/...%shum(i))^2 for each level $j = \text{bg\%n\_lev} + i$

6.6 Quality control

Figure 6.4 illustrates the implementation of `ropp_1dvar` module routines to perform preliminary quality control on the input data.

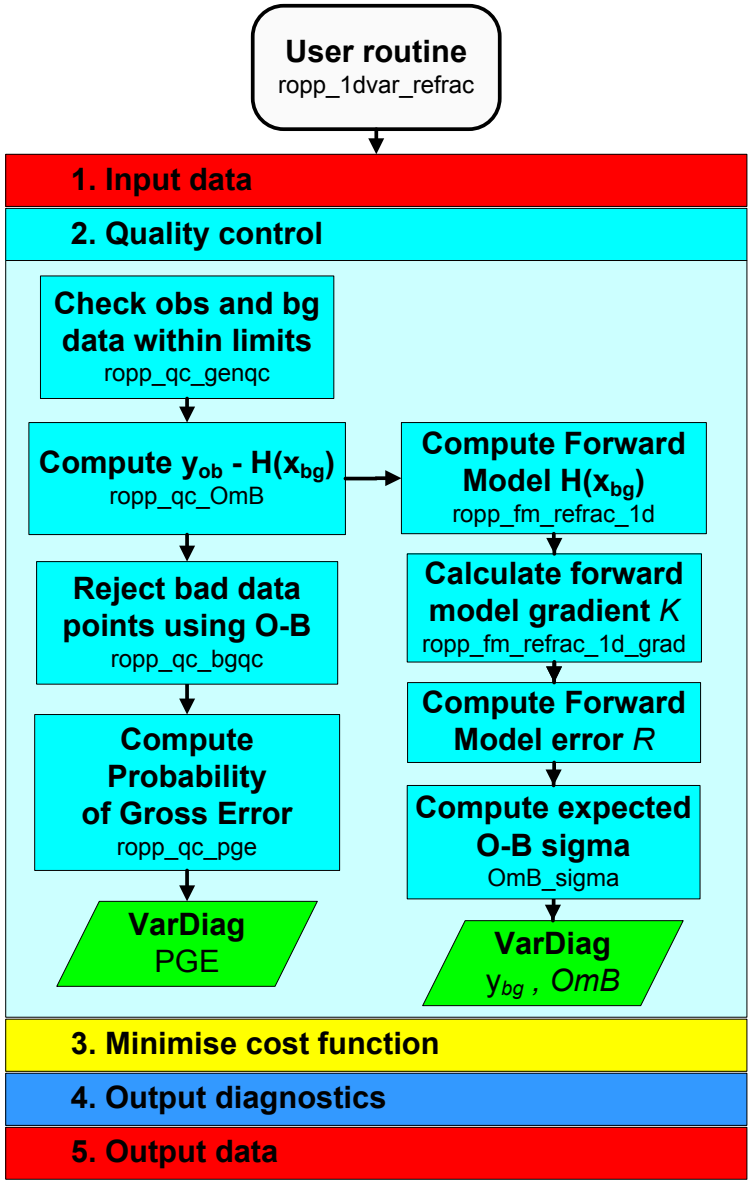

Figure 6.4: Flow chart illustrating the calling tree of the quality control step of ROPP 1D–Var to retrieve atmospheric profiles from observed refractivity profiles and input background model data.

The `ropp_qc` routines fill elements of a structure of type `VarDiag` as defined in `ropp_1dvar_types()`. These parameters may be written to the output file on completion of the `ropp_1dvar` processing. The `ropp_qc` routines determine the status of an overall quality flag `...%ok`. If set to false during the quality control stage the 1D–Var retrieval is not attempted.

```
USE ropp_fm
USE ropp_1dvar
```

```
TYPE(State1dFM)   :: bg
TYPE(Obs1dRefrac) :: obs
TYPE(VarConfig)   :: config
TYPE(VarDiag)     :: diag
diag % ok = .true.
IF (diag % ok) CALL ropp_qc_cutoff(obs, onfig)
IF (diag % ok) CALL ropp_qc_genqc(obs, bg, config, diag)
IF (diag % ok) CALL ropp_qc_OmB(obs, bg, config, diag)
IF (diag % ok) CALL ropp_qc_bgqc(obs, config, diag)
IF (diag % ok) CALL ropp_qc_pge(obs, config, diag)
```

6.6.1 Valid observation height range: `ropp_qc_cutoff()`

Configuration options `config%min_1dvar_height` and `config%max_1dvar_height` (Table 6.1) may be set to specify the height range within which observations are used in the 1D–Var analysis. Data outside this height range are given zero weighting and therefore do not contribute to the cost function. Users may wish to set the range of valid observation heights to, for example, exclude biased lower-tropospheric observations or upper-stratospheric climatological information from the 1D–Var analysis.

6.6.2 Generic quality control check: `ropp_qc_genqc()`

Generic quality control checks may be conducted by calling the subroutine `ropp_qc_genqc`. This routine checks that the background and observation data are within the physical ranges specified in the `VarConfig` structure (Table 6.1). The co-location of background and observation profiles is also tested, ensuring that the great circle distance between the observation and background coordinates is within a pre-defined limit `config%genqc_max_distance`. Similarly, the temporal separation of background and observation data can be tested and `config%ok` set to false if the data are not within `config%genqc_max_time_sep`.

6.6.3 Observation minus background check: `ropp_qc_OmB()`

The difference between the observations and the background data forward modelled into observation space ($\texttt{diag\%OmB} = \mathbf{y}_o - \mathbf{H}[\mathbf{x}_{bg}]$) and the expected uncertainty of this difference ($\texttt{diag\%OmB\_sigma} = \sigma_{(\mathbf{y}_o - \mathbf{H}[\mathbf{x}_{bg}])}$) are computed in `ropp_qc_OmB`. This routine calls the relevant `ropp_fm` module forward model to enable comparison between the background data and refractivity (Section 4.8) observations.

The error in the observation minus background calculation is given by

$$\texttt{diag\%OmB\_sigma(:)} = (\mathbf{O} + \mathbf{K}.\mathbf{B}.\mathbf{K}^T)^{1/2} \tag{6.5}$$

where $\mathbf{O}$ and $\mathbf{B}$ are the observation and background data error covariance matrices respectively and $\mathbf{K}$ gives the gradient of the forward model with respect to each element of the state vector. This is computed by calling the `ropp_fm` routine `ropp_fm_refrac_1d_grad`.

6.6.4 Background quality control check: `ropp_qc_bgqc()`

The `ropp_1dvar` 1D–Var retrieval is terminated (`diag%ok` set to false) if `config%bgqc_reject_max_percent` or more percent of the observed data are rejected in `ropp_qc_bgqc`. Data points are rejected where

$$\texttt{diag\%OmB} > \texttt{config\%bgqc\_reject\_factor} * \texttt{diag\%OmB\_sigma}$$

The weights of the rejected observation data `obs%weights` are set to zero and do not contribute to the computation of the cost function.

The number of data points used and rejected in the 1D–Var retrieval are stored in the `VarDiag` structure as `diag%n_data` and `diag%n_bgqc_reject` respectively.

6.6.5 Probability of gross error (PGE): `ropp_qc_pge()`

It is possible to screen out observations from the 1D–Var analysis which have gross errors that are inconsistent with the assumed observation errors $\mathbf{O}$. An estimate of the Probability of Gross Error (PGE) can be computed in routine `ropp_qc_pge`. Weights of $(1-PGE)$ can then be applied to elements of the observation vector by setting the configuration option `config%pge_apply`.

The PGE is computed based on the $(\mathbf{y}_o - \mathbf{H}[\mathbf{x}_{bg}])$ difference following the approach outlined by Ingleby and Lorenc (1993) and Andersson and Järvinen (1999). This is stored in the `VarDiag` structure as `diag%pge`.

$$\texttt{diag\%pge} = \frac{\gamma}{\gamma + \exp\left[-\frac{1}{2}\left(\frac{\mathbf{y}_o - \mathbf{H}[\mathbf{x}_{bg}]}{\sigma_{(\mathbf{y}_o - \mathbf{H}[\mathbf{x}_{bg}])}}\right)^2\right]} \tag{6.6}$$

The exponential argument is the contribution to the observation cost function for uncorrelated observation errors. The parameter γ is stored as element `diag%pge_gamma` and computed as

$$\gamma = \frac{A\sqrt{2\pi}}{(1-A)2d} \tag{6.7}$$

where A is the first guess PGE (`config%pge_fg`) and d is the width of the gross error (`config%pge_d`).

6.7 Minimise the cost function

Figure 6.5 illustrates the implementation of `ropp_1dvar` module routines to minimise the cost function and obtain the solution state vector for a given 1D–Var retrieval.

The main `ropp_1dvar` routine for retrieving the solution state vector ($\mathbf{x}$) which minimises the cost function (Equation 6.1) is `ropp_1dvar_solve`. Given input variables of the correct format, `ropp_1dvar_solve` could be implemented by users as a 'black box' to perform 1D–Var retrievals using either refractivity or bending angle observations. On entry the solution state vector $\mathbf{x}$ of type `State1dFM` is set to a first guess solution of x_{bg}. On exit $\mathbf{x}$ contains the 1D–Var solution.

```
USE ropp_fm
USE ropp_1dvar
TYPE(State1dFM)   :: bg
TYPE(State1dFM)   :: x
TYPE(Obs1dRefrac) :: obs
...
x = bg    ! initial guess - set x equal to background state
CALL ropp_1dvar_solve(obs, bg, x, config, diag)
```

Minimisation of the cost function is achieved by an iterative method which computes the cost function value and updates the state vector on each of $n = 1,..,$`n_iter` iterations until convergence to a solution is achieved. The state vector is updated using the ROPP-specific minimiser minROPP (`ropp_1dvar_minropp`). This implements a quasi-Newton method (Nocedal, 1980) using a BFGS algorithm (Press et al., 1992). Further details are provided in ROM SAF (2007).

An alternative Levenberg-Marquardt minimisation algorithm (Press et al., 1992) is also provided with ROPP. The `ropp_1dvar_levmarq` routine may be used in place of `ropp_1dvar_solve`. This can be implemented in the `ropp_1dvar` tools by specifying the `minROPP%method` configuration option (Table 6.1) as `LEVMARQ`. Further details on the algorithm and its performance are provided in ROM SAF (2008). Users should note that the default `minROPP` minimiser provided with ROPP, as described in ROM SAF (2007), is more efficient than the Levenberg-Marquardt algorithm.

6.7.1 Preconditioning

Convergence to a solution during the minimisation step can be accelerated by a change of variable from the state vector to a control variable. If configuration option `config%use_precond` is set, routine `ropp_state2control` is used to perform the initial variable transform from the state variable to a control variable , given a preconditioner $\mathbf{L}$ such that $\mathbf{B} = \mathbf{L}\mathbf{L}^T$.

$$\mathbf{control} = \mathbf{L}^{-1}\mathbf{x} \tag{6.8}$$

All other elements of the `State1dFM` structure are unaffected by this transformation. Note that preconditioning is only applied to the state vector for the minimisation step. The cost function is computed using

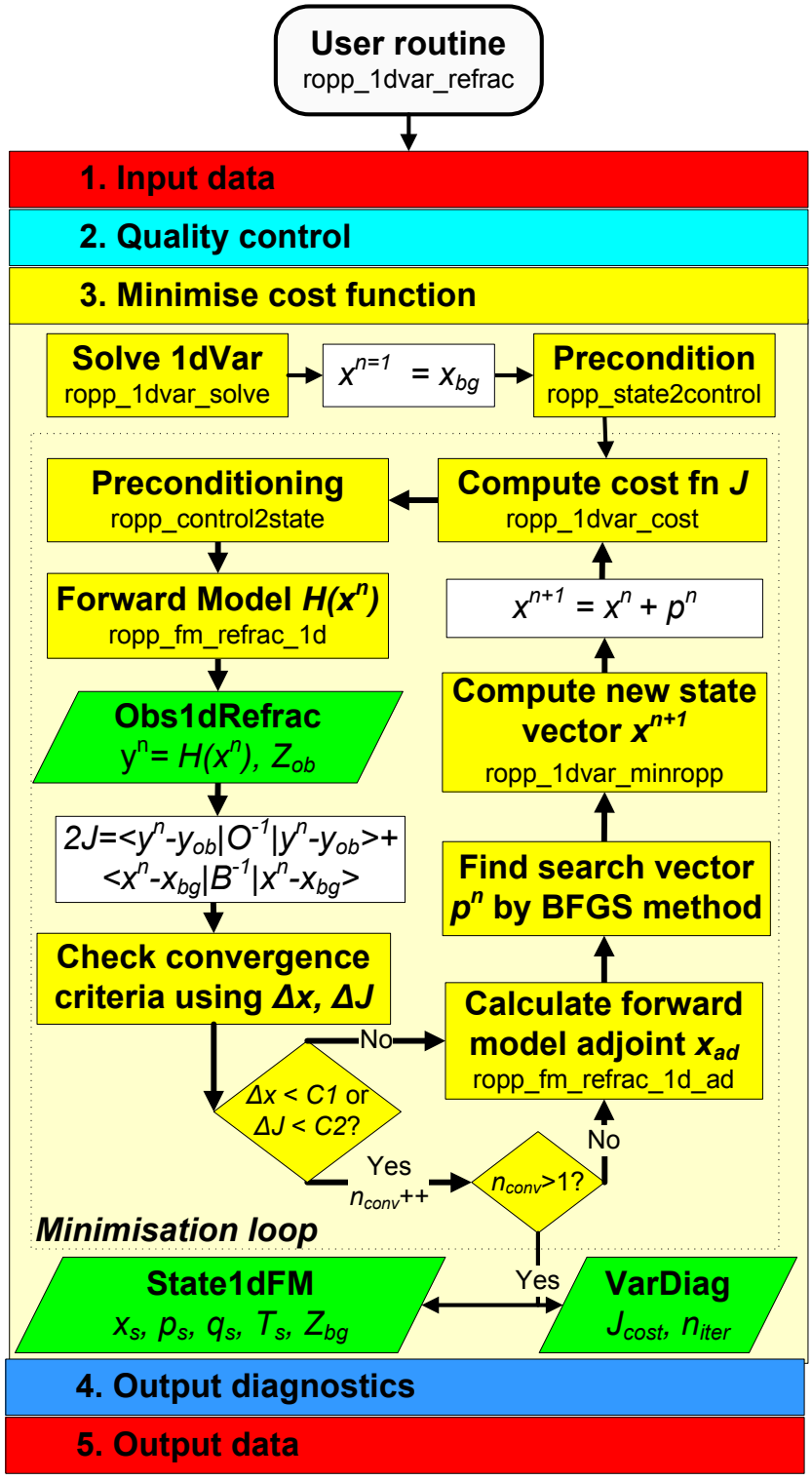

Figure 6.5: Flow chart illustrating the calling tree of the minimisation step of ROPP 1D–Var to retrieve atmospheric profiles from observed refractivity profiles and input background model data.

the state vector $\mathbf{x}$, which can be recovered from **control** by calling the routine `ropp_control2state`.

6.7.2 Compute the cost function

Equation (6.1) is computed on each iteration in routine `ropp_1dvar_cost`. The cost function `J` is computed for the current state vector $\mathbf{x}$ together with the gradient of `J` with respect to the state vector elements (`J_grad`).

```
USE ropp_fm
USE ropp_1dvar
USE matrix
TYPE(State1dFM)   :: bg
TYPE(State1dFM)   :: control
TYPE(Obs1dRefrac) :: obs
TYPE(VarConfig)   :: config
TYPE(matrix_sq)   :: precon
CALL ropp_1dvar_cost(obs, bg, control, precon, J, J_grad, config, indic)
```

To process refractivities, the `ropp_fm` routine `ropp_fm_refrac_1d` is called each time the cost function is evaluated to forward model the current state vector into the observation space. The relative weighting applied to each observation is applied at the cost function stage by scaling the difference $\mathbf{y}_o - \mathbf{H}[\mathbf{x}]$ with the weighting factors determined from the quality control processing.

The matrix multiplication of the differences $(\mathbf{x} - \mathbf{x}_b)$ and $(\mathbf{y}_o - \mathbf{H}[\mathbf{x}])$ with the inverse of the background and observation error covariance matrices respectively is performed using the `matrix_solve` routine. This solves a linear matrix equation of the form $\mathbf{Ax} = \mathbf{b}$ using a Cholesky decomposition (Press et al., 1992).

To minimise the cost function it is necessary to evaluate the gradient of the cost function with respect to the state vector:

$$\nabla J(\mathbf{x}) = \mathbf{B}^{-1}(\mathbf{x} - \mathbf{x}_b) - (\mathbf{H}[\mathbf{x}]')^T \mathbf{O}^{-1}(\mathbf{y}_o - \mathbf{H}[\mathbf{x}]) = \mathbf{0} \tag{6.9}$$

where $\mathbf{H}'$ is the gradient (or tangent linear) of the forward operator with respect to the state vector and $\mathbf{H}'^T$ is termed the adjoint of the forward operator. Tangent linear and adjoint code for the ROPP forward models are provided with the `ropp_fm` module. The adjoint is computed as part of `ropp_1dvar_cost` by calling routine `ropp_fm_refrac_1d_ad` as required. On exit `ropp_1dvar_cost` returns the variable `J_grad` which contains the cost function gradient.

6.7.3 Convergence check

If the configuration option `config%conv_check_apply` is set then `ropp_1dvar_cost` performs a check to identify when convergence to a satisfactory solution has been obtained. One of two criteria need to be met on `config%conv_check_n_previous` consecutive iterations for convergence to be achieved.

Maximum relative change in state

Convergence is assumed if the state vector does not change by more than a set value between iterations

$$\max\left[\frac{|\mathbf{x}_n - \mathbf{x}_{n-1}|}{\sqrt{\mathbf{B}}}\right] < \texttt{config\%conv\_max\_delta\_state} \tag{6.10}$$

for at least `config%conv_check_n_previous` consecutive iterations.

Maximum change in cost function

Convergence is assumed if the cost function does not change by more than a set value between iterations

$$|J_n - J_{n-1}| < \texttt{config\%max\_delta\_J} \tag{6.11}$$

for at least `config%conv_check_n_previous` consecutive iterations.

When either of the convergence criteria are met, flag `indic` is returned with a value of zero which terminates the minimisation loop in routine `ropp_1dvar_solve`. If preconditioning was used, the solution control vector is transformed to the state vector using `ropp_control2state`. Pressure, temperature and humidity profiles corresponding to the solution state vector are recovered using a background model-dependent conversion routine `ropp_fm_state2state_ecmwf` (Section 4.5) or `ropp_fm_state2state_meto` (Section 4.6). Diagnostic parameters `diag%J`, `diag%J_scaled` and `diag%n_iter` are set.

6.7.4 Minimisation

If the convergence criteria tested in `ropp_1dvar_cost` are not met, flag `indic` is returned with a value of 4, indicating that further iteration is required. The state vector is incremented towards a solution by the ROPP-specific minimiser `minROPP`. This is a quasi-Newton minimisation routine which finds a search direction vector to determine the updated state vector using a BFGS algorithm (e.g. Press et al. (1992), Nocedal (1980)). A technical summary of this algorithm is provided by ROM SAF (2007).

The `minROPP` routine `ropp_1dvar_minropp` is called as

```
call ropp_1dvar_minropp(control, J_grad, J_dir, dJ, gconv, n_iter,
                        indic, config%minropp%n_iter, maxstore)
```

where **control** is the control state vector, which is updated on exit. `J_grad` is the gradient of the cost function with respect to the control vector, determined by `ropp_1dvar_cost`. `J_dir` is the quasi-Newton search direction vector which is updated by `minROPP`. This determines the updated state vector. `dJ` is the expected decrease of the cost function, which is used in `minROPP` to calculate the initial value of search direction vector `J_dir`. The `n_iter` variable is an iteration counter, which is incremented each time `minROPP` updates the state vector.

An additional convergence check is provided in `minROPP`. Convergence is assumed at iteration n if the gradient of the cost function at $\mathbf{x}_n$ differs from its initial value when $n = 1$ by more than a pre-defined

factor.

$$||\nabla J_k|| < \text{eps}||\nabla J_1|| \tag{6.12}$$

Parameter eps is set as a configuration option as element `config%minROPP%eps_grad`. The routine exits if this condition is met, and the minimisation loop is terminated.

Figure 6.5 illustrates how the minimisation loop continues in `ropp_1dvar_solve` to update the state vector and compute new values for J and ∇J for `n_iter` iterations until convergence is achieved. If a maximum of `config%minROPP%n_iter` iterations have been completed, then no convergence can be achieved and the 1D–Var retrieval fails.

6.8 Output diagnostics

Figure 6.6 illustrates the implementation of `ropp_1dvar` and `ropp_fm` module routines to compute final output diagnostics associated with a 1D–Var retrieval solution. Table 6.2 lists the diagnostics which may be optionally output if the `config%extended_1dvar_diag` configuration flag is set. Further information and examples of these diagnostics are provided by ROM SAF (2010).

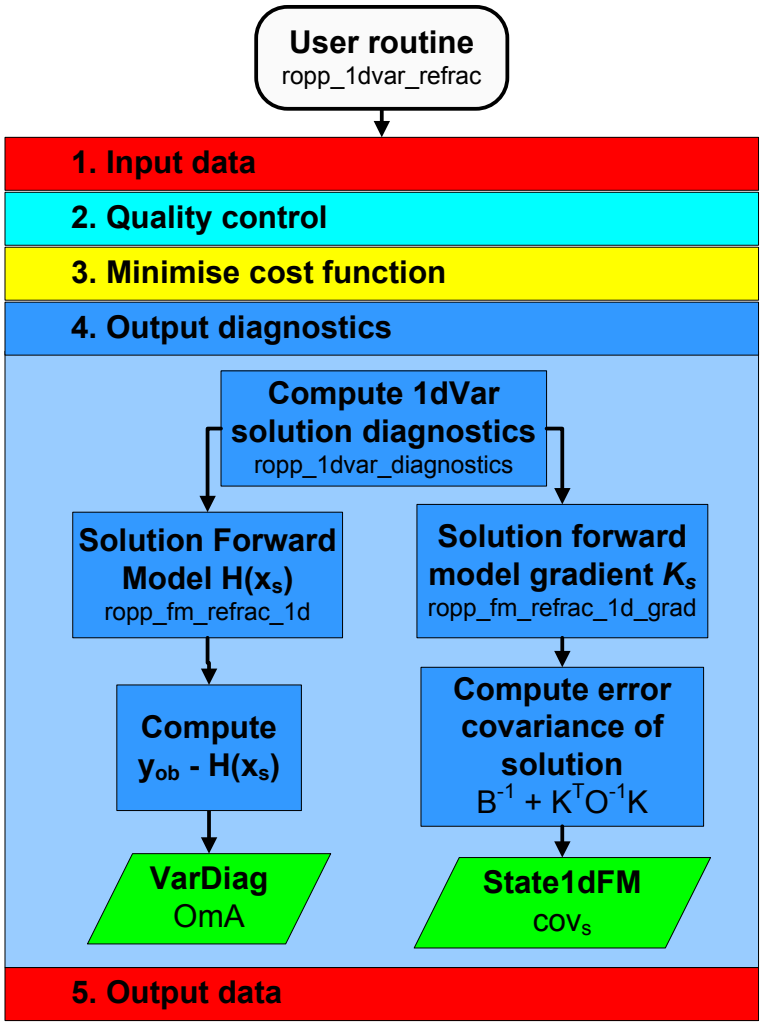

Figure 6.6: Flow chart illustrating the calling tree of the output diagnostics step of ROPP 1D–Var to retrieve atmospheric profiles from observed refractivity profiles and input background model data.

The `ropp_1dvar` diagnostic routine is `ropp_1dvar_diagnostics` which is called as,

```
USE ropp_fm
USE ropp_1dvar
TYPE(State1dFM)   :: x
TYPE(Obs1dRefrac) :: obs
TYPE(VarConfig)   :: config
```

```
TYPE(VarDiag)     :: diag
CALL ropp_1dvar_diag2roprof(obs, x, config, diag)
```

The diagnostic processing applied to the solution state vector is similar to the initial quality control checks applied to compare the observation and background data (6.6.3). The diagnostic variable `diag%OmA` is computed as the difference between the observations ($\mathbf{y}_o$) and solution state vector forward modelled into observation space ($\mathbf{H}[\mathbf{x}_s]$). The forward modelled solution $\mathbf{H}[\mathbf{x}_s]$ is saved as element `diag%res_refrac`.

An estimate of the solution error covariance matrix is obtained using the observation and background error covariance matrices and forward model gradient $\mathbf{K}$ as (Ch 5 Rodgers, 2000)

$$\mathrm{x\%cov} = \left(\mathbf{B}^{-1} + \mathbf{K}^T\mathbf{O}^{-1}\mathbf{K}\right)^{-1} \tag{6.13}$$

The gradient matrix $\mathbf{K}$ gives the gradient of the forward model with respect to each element in the solution state vector. This is computed by calling the routine `ropp_fm_refrac_1d_grad`.

6.9 Output data

The final stage in the `ropp_1dvar` processing is to fill the elements of the generic ROPP structure of type `ROprof` with the 1D–Var solution for writing to an output file using the `ropp_io` module routine `ropp_io_write` (ROM SAF, 2013).

```
USE ropp_io
USE ropp_fm
USE ropp_1dvar
TYPE(ROprof)      :: res_data
TYPE(State1dFM)   :: x
TYPE(VarDiag)     :: diag
TYPE(VarConfig)   :: config
CALL ropp_fm_state2roprof(x, res_data)
CALL ropp_fm_obs2roprof(diag%res_refrac, res_data)
CALL ropp_1dvar_diag2roprof(obs, diag, res_data, config)
```

The solution state vector elements are copied to meteorological variables as Level 2b and Level 2c data in `ropp_fm_state2roprof`. The translation between state vector elements and `ROprof` variables depends on the exact details of the state vector and structure of the background model (Section 4.2.2).

Level 2a (refractivity) data in the `ROprof` structure are filled with the solution state vector forward modelled into observation space. These profiles are given by `diag%res_refrac` returned by `ropp_1dvar_diagnostics`.

Diagnostic parameters gathered during a 1D–Var retrieval are added to the `ROprof` data structure in routine `ropp_1dvar_diag2roprof`. All elements of the `VarDiag` structure may be output if the configuration option `config%extended_1dvar_diag` is set. Otherwise, only the cost function value at convergence and the cost function scaled by the number of observations are output.

6.10 Plotting tools

The directory `tests/` contains the IDL procedure `plot_1dvar.pro` which gives an example of how to read and plot pressure, temperature and humidity increments computed by running the refractivity 1D–Var.

An example of its implementation, using archive data available from the ROM SAF archive http://www.romsaf.org, is provided by running the test script `test_1dvar_GRAS.sh`.

References

Andersson, E. and Järvinen, H., Variational quality control, *Quart. J. Roy. Meteorol. Soc.*, *125*, 697–722, 1999.

Ingleby, N. B. and Lorenc, A. C., Bayesian quality control using multivariate normal distributions, *Quart. J. Roy. Meteorol. Soc.*, *119*, 1195–1225, 1993.

Nocedal, J., Updating quasi-Newton matrices with limited storage, *Mathematics of Computation*, *35*, 773–782, 1980.

Press, W., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical recipes in C – The Art of Scientific Computing*, Cambridge University Press, Cambridge, New York, 2nd edn., 1992.

Rodgers, C. D., *Inverse methods for atmospheric sounding: Theory and practice*, World Scientific Publishing, Singapore, New Jersey, London, Hong Kong, 2000.

ROM SAF, ROPP minimiser - minROPP, SAF/GRAS/METO/REP/GSR/003, 2007.

ROM SAF, Levenberg-Marquardt minimisation in ROPP, SAF/GRAS/METO/REP/GSR/006, 2008.

ROM SAF, ROPP 1dVar Validation, SAF/GRAS/METO/REP/GSR/011, 2010.

ROM SAF, The Radio Occultation Processing Package (ROPP) User Guide. Part I: Input/Output module, SAF/ROM/METO/UG/ROPP/002, Version 7.0, 2013.

Steiner, A. and Kirchengast, G., Error analysis for GNSS radio occultation data based on ensembles of profiles from end-to-end simulations, *J. Geophys. Res.*, *110*, D15 307, 2005.

VarConfig		
Structure element	Default	Description
...%obs_file	ropp_obs.nc	Input observation data file
...%obs_covar_method	VSDC	Observation error covariance method
...%obs_corr_file	ropp_obs_corr.nc	Observation error correlation file
...%bg_file	ropp_bg.nc	Input background data file
...%out_file	ropp_out.nc	Output file
...%bg_covar_method	VSFC	Background error covariance method
...%bg_corr_file	ropp_bg_corr.nc	Background error correlation file
...%min_1dvar_height	-10 km	Min. obs height to include in 1D–Var
...%max_1dvar_height	60 km	Max. obs height to include in 1D–Var
...%genqc_colocation_apply	.true.	Apply obs and bg colocation check?
...%genqc_max_distance	300 km	Max. obs to bg great circle distance
...%genqc_max_time_sep	300 sec	Max. obs to bg temporal separation
...%genqc_min_temperature	150 K	Min./Max. bg data values
...%genqc_max_temperature	350 K	
...%genqc_min_spec_humidity	0 g/kg	
...%genqc_max_spec_humidity	50 g/kg	
...%genqc_min_impact	6.2e6 m	Min./Max. obs data values
...%genqc_max_impact	6.6e6 m	
...%genqc_min_bangle	-1.0e-4 rad	
...%genqc_max_bangle	0.1 rad	
...%genqc_min_geop_refrac	-1.e3 m	
...%genqc_max_geop_refrac	1.e5 m	
...%genqc_min_refractivity	0.0	
...%genqc_max_refractivity	500	
...%bgqc_apply	.true.	Apply background quality control?
...%bgqc_reject_factor	10	Data rejected if O-B > factor * sigma
...%bgqc_reject_max_percent	50	Maximum %age data rejected
...%pge_apply	.false.	Apply PGE for quality control?
...%pge_fg	0.001	First guess PGE
...%pge_d	10.0	Width of gross error plateau
...%minROPP%method	MINROPP	minimisation method
...%minROPP%log_file	screen	minROPP output device
...%minROPP%impres	0	minROPP output mode
...%minROPP%n_iter	1500	maximum number of iterations
...%minROPP%eps_grad	1.0e-8	minROPP convergence parameter
...%use_precond	.true.	Use preconditioning?
...%conv_check_apply	.true.	Apply additional convergence checks?
...%conv_check_n_previous	2	No. of previous iterations to check
...%conv_check_max_delta_state	0.1	Maximum change in state vector
...%conv_check_max_delta_J	0.1	Maximum change in cost function
...%extended_1dvar_diag	.false.	Output additional diagnostics?
...%use_logp	.false.	Use log(pres) in 1D–Var
...%use_logq	.false.	Use log(shum) in 1D–Var

Table 6.1: Configuration options held as elements of the `VarConfig` structure which are used by `ropp_1dvar` routines. The default values are assumed unless overwritten by reading configuration options from an input file.

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

Table 6.2: Elements of `VarDiag` structure output using the `extended_1dvar_diag` configuration flag.

7 ROPP 1D–Var: Bending angle

The ROPP 1D–Var module (`ropp_1dvar`) includes routines to retrieve profiles of pressure, temperature and humidity using a measured bending angle profile, the *a priori* knowledge of the state of the atmosphere (i.e. background profiles) and their associated errors. This is achieved in the `ropp_1dvar_solve` subroutine through the minimisation of a quadratic cost function.

$$J(\mathbf{x}) = \frac{1}{2}(\mathbf{x}-\mathbf{x}_b)^T\mathbf{B}^{-1}(\mathbf{x}-\mathbf{x}_b) + \frac{1}{2}(\mathbf{y}_o-\mathbf{H}[\mathbf{x}])^T\mathbf{O}^{-1}(\mathbf{y}_o-\mathbf{H}[\mathbf{x}]) \tag{7.1}$$

In ROPP the state vector $\mathbf{x}$ is part of a Fortran 90 derived structure of type `State1dFM` containing pressure, temperature and humidity data on geopotential height levels. This contains the retrieved atmospheric state. $\mathbf{x}_b$ is the initial background state of type `State1dFM` defined by input background model profiles. Matrix $\mathbf{B}$ defines the error covariance of the background data. $\mathbf{y}_o$ is the observation vector. If the 1D–Var is performed using measured bending angle then $\mathbf{y}_o$ is an observation vector of type `Obs1dBangle` which contains bending angle as a function of impact parameter. The forward modelled observation $\mathbf{H}[\mathbf{x}]$ is also held in an observation vector, and is given by the output of `ropp_fm` routines to compute bending angle (Section 4.9) for a given atmospheric state.

Figure 7.1 shows example pressure, temperature and humidity profiles retrieved from background model data and colocated GNSS-RO bending angle observations.

7.1 ROPP 1D–Var bending angle tool

The stand-alone tool `ropp_1dvar_bangle` is provided in `ropp_1dvar` as an illustration of how the `ropp_1dvar` routines can be implemented to retrieve pressure, temperature and humidity profiles from bending angle observations respectively. Figure 7.2 shows how the `ropp_1dvar` routines are integrated in the `ropp_1dvar_bangle` code.

7.1.1 Implementation

The `ropp_1dvar_bangle` tool is run using the command

```
ropp_1dvar_bangle [options] -o <outputfile>
```

where `<outputfile>` is a ROPP netCDF file (ROM SAF, 2013) which will contain the retrieved temperature, humidity and pressure profiles on model background levels.

The following options can be defined on running the command.

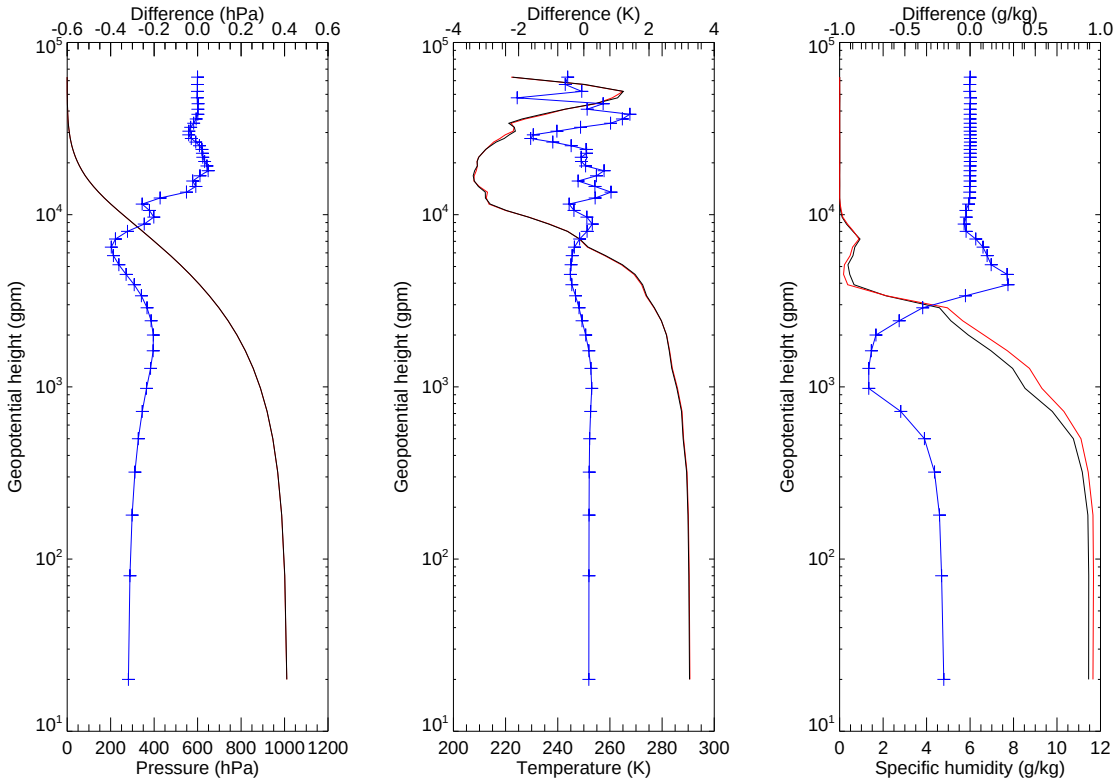

Figure 7.1: Example pressure, temperature and humidity profile retrievals (red) computed using background profiles (plotted in black) and colocated GNSS-RO bending angle observations. The difference between the two profiles are plotted in blue.

`-c <config_file>`	text file specifying configuration options
`-y <obs_file>`	ROPP netCDF observation file
`--obs-corr <obs_corr_file>`	file with observation error covariance parameters
`-b <bg_file>`	ROPP netCDF background data file
`--bg-corr <bg_corr_file>`	netCDF file with background error covariance parameters
`-o <outputfile>`	ROPP netCDF file for output retrieved profiles
`-comp`	use non-ideal gas compressibility options in forward model
`-check_qsat`	include check against saturation in forward model
`-d`	output additional diagnostic information (VerboseMode)
`-h`	give help menu
`-v`	output version information

If the input observation and background data files are multi-files containing more than one profile, the routine `ropp_1dvar_bangle` computes a 1D–Var retrieval for each profile in turn and the output file generated contains all the output profiles.

7.1.2 Code organisation

Figure 7.2 shows how the `ropp_1dvar_bangle` tool is composed of the following stages:

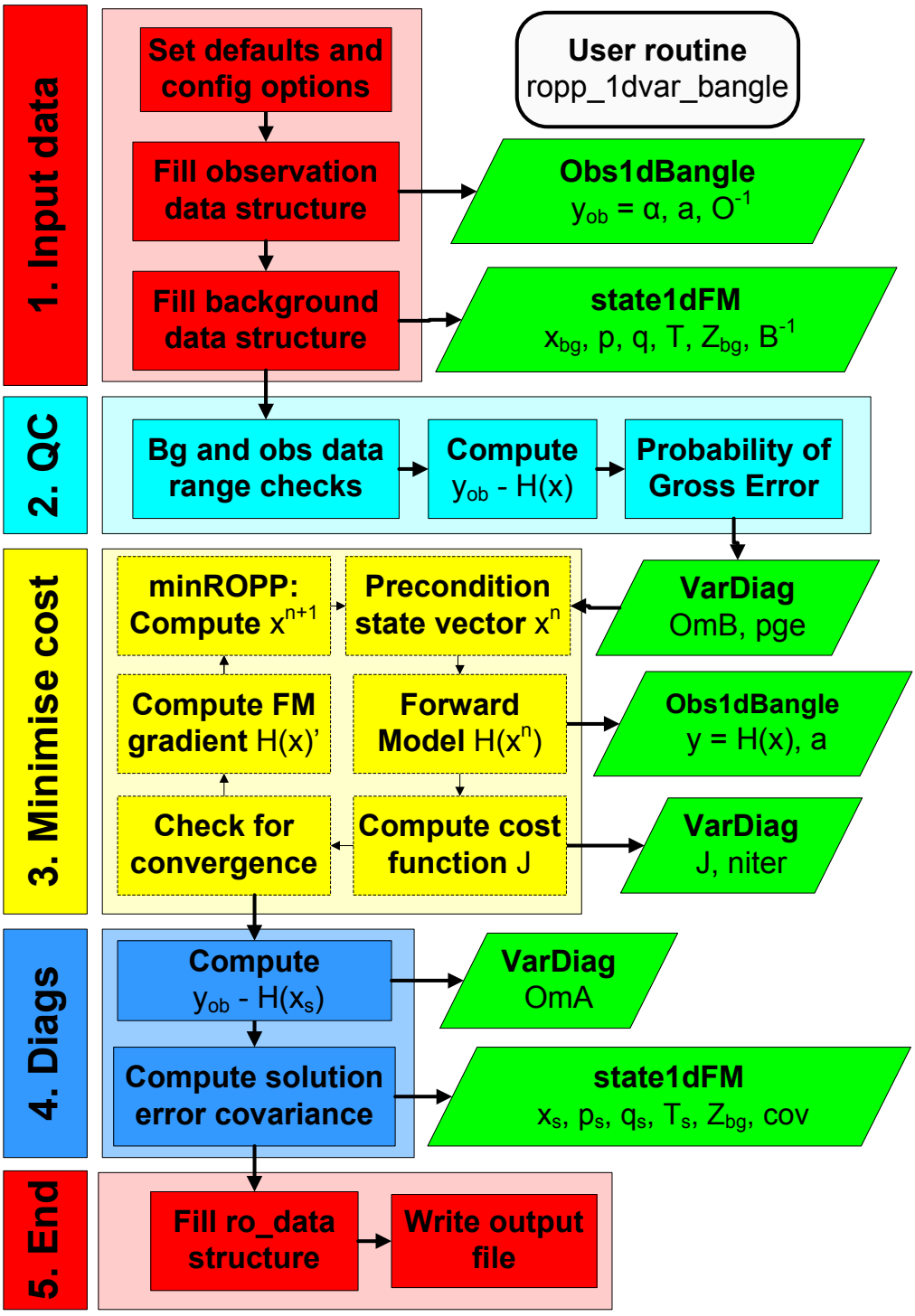

Figure 7.2: Flow chart illustrating the calling tree of the ROPP 1D–Var to retrieve atmospheric profiles from observed bending angle profiles and input background model data.

- **Input data access and transformation to a generic state vector.** (Section 7.3)

  Setup the input data arrays and read the input data into the RO data structure of type `ROprof`. Define a state vector structure `State1dFM` containing the background pressure p, temperature T and humidity q data as a function of geopotential height Z_{bg}. Define an observation vector structure containing the observed bending angles α as a function of impact parameter a. Define the observation

error covariance matrix $\mathbf{O}$ and background error covariance matrix $\mathbf{B}$.

- **Perform quality control checks.** (Section 7.6)

  A number of preliminary data quality control checks are performed. This includes setting the required valid height range of the observations required in the 1D–Var analysis. General checks are made that observation and background data are co-located in space and time and that background and observation data are within pre-defined ranges. Diagnostic parameters, which may be utilised as part of a data assimilation system are also computed and stored as part of a structure of type `VarDiag`. The difference between observations and the background state forward modelled into observation space is computed using the `ropp_fm` routines. The probability of gross error (PGE) is also computed.

- **Minimise the cost function.** (Section 7.7)

  `ropp_1dvar_solve` is the main routine for solving the 1D–Var to find the atmospheric state vector $\mathbf{x}$ which minimises the cost function for a given background and observed data and their associated errors. This consists of three stages on each iteration towards a solution:

  - Compute the cost function (Equation 7.1) and its gradient in routine `ropp_1dvar_cost`. This requires re-computing the forward model $\mathbf{H}[\mathbf{x}]$ using `ropp_fm` routines on each iteration using the current state vector $\mathbf{x}$.
  - Call minimiser `ropp_1dvar_minropp` to update the state vector $\mathbf{x}$ towards a solution.
  - Check against pre-defined convergence criteria to identify whether the cost function has been minimised and the optimal solution has been obtained.

- **Compute final diagnostics.** (Section 7.8)

  The deviation between observations and the solution state vector forward modelled into observation space (using `ropp_fm` routines) is computed and stored as an element in the structure of type `VarDiag`. The error covariance of the solution is also computed.

- **Write results to a generic RO data structure and output file.** (Section 7.9)

7.2 Defining observation and background errors

The variational approach requires estimates of the background and observation errors. The `ropp_1dvar` module includes a collection of observation and background error correlation and standard deviation files and tools which users may find helpful for setting up input to `ropp_1dvar` tools. These are available in the `ropp_1dvar/errors/` subdirectory.

ROM SAF (2010) provides an illustration of the available error structures. Users are encouraged to adapt the provided routines to meet their own applications.

A number of tools are provided to add profile-by-profile background or observation error values to a ROPP format file, which may then be used within the 1D–Var routines (see Sections 7.4.2 and 7.5.2). These are summarised below.

Background errors: `ropp_1dvar_add_bgr_error`

```
ropp_1dvar_add_bgr_error <bg_file> -c <cov_file> [-o <out_file>]
```

This tool adds background pressure, temperature and humidity error values to a ROPP format netCDF file `<bg_file>`. The errors are read from the 'sigma' variable in an input background covariance matrix file `<cov_file>` (e.g. `errors/ropp_bg_ecmwf_error_corr_L91.nc`).

Bending angle observation errors: `ropp_1dvar_add_bangle_error`

```
ropp_1dvar_add_bangle_error <obs_file> -Omod <Omodel> -o <out_file>
```

This tool adds bending angle observation error values to a ROPP format netCDF file `<obs_file>`. A number of different observational error types may be chosen by the user, as specified using the `-Omod` command line option.

- **1%** : errors 1% at 0 km, 0.1% from 12 km, min($\sigma(BA)$)=6 μrad
- **2%** : errors 2% at 0 km, 0.2% from 12 km, min($\sigma(BA)$)=6 μrad
- **3%** : errors 3% at 0 km, 0.3% from 12 km, min($\sigma(BA)$)=6 μrad
- **MO** : Met Office operational implementation using pre-defined latitudinally varying percentage errors.
- **EC** : ECMWF operational implementation using pre-defined percentage errors.

Error correlation files

A number of error correlation data files are also provided in the `ropp_1dvar/errors/` subdirectory for reference and use with the 1D–Var tools.

- **ropp_bg_ecmwf_error_corr_L91.nc** - Background error correlation matrix in packed form for the latest ECMWF model structure (91 levels). This file is suitable for input to the `ropp_1dvar` tools

using '–bg-corr' command line option with ECMWF backgrounds. It is recommended that 1D–Var tools are run with bg_covar_method = 'VSFC' config option.

- **ropp_bg_meto_error_corr_L70.nc** - Background error correlation matrix in packed form for the latest Met Office model structure (70 levels). This file is suitable for input to the `ropp_1dvar` tools using '–bg-corr' command line option with Met Office backgrounds. It is recommended that 1D–Var tools are run with bg_covar_method = 'VSFC' config option.
- **ropp_bg_ecmwf_error_corr_L60.nc** - Background error correlation matrix in packed form for the previous ECMWF model structure (60 levels). It is recommended that 1D–Var tools are run with bg_covar_method = 'VSFC' config option.
- **ropp_bg_meto_error_corr_L50.nc** - Background error correlation matrix in packed form for the previous Met Office model structure (50 levels). This file is suitable for input to the `ropp_1dvar` tools using '–bg-corr' command line option with Met Office backgrounds. It is recommended that 1D–Var tools are run with bg_covar_method = 'VSFC' config option.
- **ropp_ob_bangle_error_corr.nc** - Bending angle observation correlation matrix in packed form. For use with 'standard' 300 levels. This file is suitable for input to the `ropp_1dvar_bangle` tool using '–obs-corr' command line option.

Note that bending angle observations are usually taken to be uncorrelated, so it is generally recommended that the `ropp_1dvar_bangle` tool is run with errors specified with the `VSDC` configuration option, and without specifying an 'obs-corr' matrix. (The example bending angle correlation matrix is in fact the identity matrix, and is included only for completeness.)

7.3 Input data

Figure 7.3 illustrates the implementation of `ropp_1dvar` and `ropp_fm` module routines to input background and observation data and associated error covariance matrices into the data structures required by the subsequent 1D–Var processing.

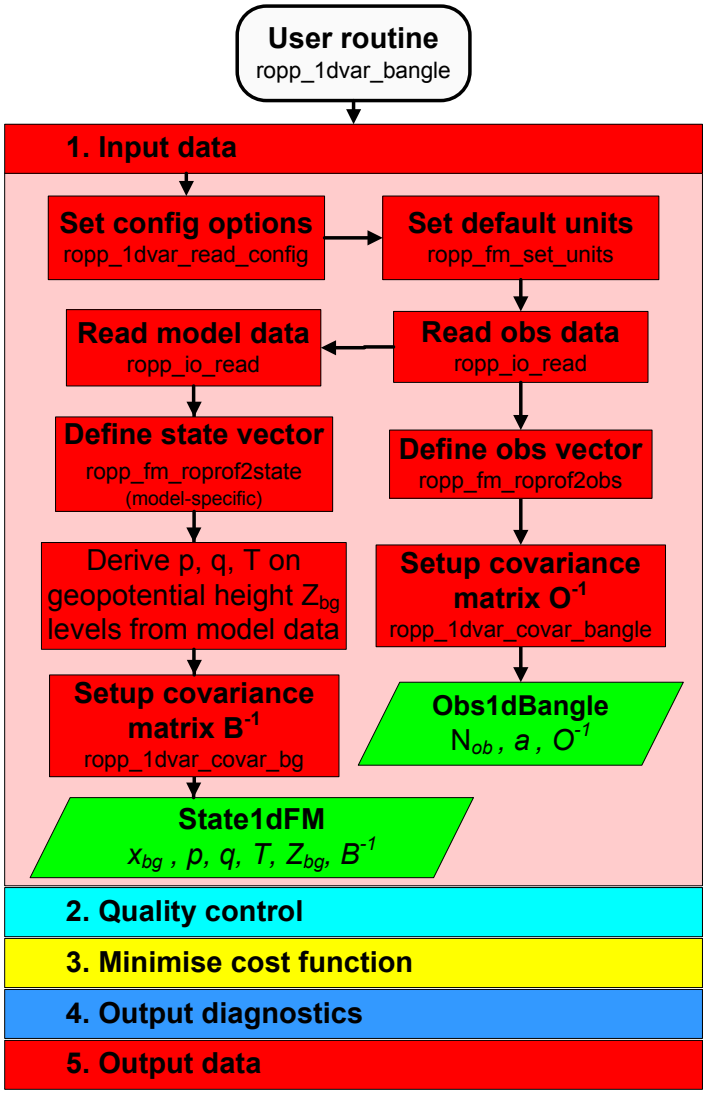

Figure 7.3: Flow chart illustrating the calling tree of the input data step of ROPP 1D–Var to retrieve atmospheric profiles from observed bending angle profiles and input background model data.

Note that the ROPP forward model assumes data are in ascending height order. The input data are checked and reordered as part of the stand-alone tool processing (see 4.4). If required for 1D–Var, users must ensure that the error correlations used are appropriate for background and observation data in ascending height order.

7.3.1 Configuration options

A number of configuration options can be defined by the user in order to tune the performance of the ropp_1dvar retrieval. Table 7.1 lists the configuration options and their default values held in a structure of derived type VarConfig. The use of these parameters within ropp_1dvar are described within this User Guide. A user can specify configuration parameters at run-time by setting their values in a configuration file and including the '-c <config_file>' command line option when running the ropp_1dvar_bangle tool.

The configuration file is read, if specified, and the elements of a variable of type VarConfig are overwritten by calling subroutine ropp_1dvar_read_config.

```
USE ropp_1dvar
TYPE(VarConfig)  :: config
CALL ropp_1dvar_read_config(config_file, config)
```

A number of sample configuration files are included with the ROPP distribution in the ropp_1dvar/config directory.

7.4 Observation data

For bending angles, the routines ropp_1dvar and ropp_fm use observation data defined as elements of a structure of type Obs1dBangle. (Section 4.2.1).

The ropp_fm subroutine ropp_fm_set_units is first called to ensure that all variables are specified in the default forward model units before any other ropp_1dvar processing. This utilises the ropp_utils unitconvert library functions. The ropp_io module routine ropp_io_read then reads a single profile of observation data from a netCDF ROPP format input file and fills the elements of the generic ROPP data structure of type ROprof (ROM SAF, 2013).

```
USE ropp_io
USE ropp_fm
USE ropp_1dvar
TYPE(ROprof) :: obs_data
CALL ropp_fm_set_units(obs_data)
CALL ropp_io_read(obs_data, config%obs_file, rec=iprofile)
```

7.4.1 Defining the observation vector: ropp_fm_roprof2obs

The relevant bending angle observation data can be copied to elements of the observation vector $\mathbf{y}$ of type Obs1dBangle by calling the routine ropp_fm_roprof2obs (Section 4.7).

7.4.2 Defining the observation error covariance matrix: `ropp_1dvar_covar_bangle`

The subroutine `ropp_1dvar_covar_bangle` is used to set up the observation error covariance matrix for a vector of bending angle observations.

```
USE ropp_fm
USE ropp_1dvar
TYPE(Obs1dBangle) :: obs
TYPE(VarConfig)   :: config
CALL ropp_1dvar_covar(obs, config)
```

The error covariance matrix $\mathbf{O}$ is constructed by computing the matrix product,

$$\mathbf{O} = \sigma.\mathbf{Corr}.\sigma^T \tag{7.2}$$

where $\mathbf{Corr}$ is a matrix containing the correlation between elements in the observation vector and σ is a diagonal matrix where the diagonal elements contain the error standard deviations for each element in the observation vector. The elements of $\mathbf{O}$ are held in the `Obs1dBangle` structure for use in the `ropp_1dvar` processing aselement `obs%cov`.

The observation error covariances can be constructed using the following methods in ROPP. The method to be used is specified as a configuration option (Table 7.1).

- **FSFC - Fixed Sigmas, Fixed correlations**
  Both error correlations and error standard deviations are read from an observation error correlation file. The error correlation file is specified by the '`--obs-corr <obs_corr_file>`' command-line option or can be set as a default configuration file option. The error correlation file must contain both the error correlation matrix as well as the standard deviations (errors) for all observation vector elements. Sample files containing observation sigma and correlation values are provided in the `errors/` sub-directory of the `ropp_1dvar` distribution (see Section 7.2).
- **VSDC - Variable Sigmas, Diagonal Correlations**
  A diagonal error correlation structure (i.e., no error correlations) is assumed. Error estimates are obtained separately for each input profile by using standard deviation values specified in the input observation data file. Note that the input observation file must contain valid error estimates for all observation vector elements, even if the observation value at a given level is invalid. In this case, no observation error correlation data file is required. Tools for defining variable sigma values in an input profile are provided in the `errors/` sub-directory of the `ropp_1dvar` distribution (see Section 7.2).
- **VSFC - Variable Sigmas, Fixed Correlations**
  Error correlations are read from an error correlation file, while error estimates are obtained separately for each input profile from the standard deviations contained in the input observation data file. Note that the input observation file must contain valid standard deviations for all observation vector elements, even if the observation value at a given level is invalid. The error correlation file is specified by the '`--obs-corr <obs_corr_file>`' command-line option or can be set as a default configuration file option. In this case the error correlation data files only need to contain the error

correlations. Tools for defining variable sigma values in an input profile and sample observation error files are provided in the `errors/` sub-directory of the `ropp_1dvar` distribution (see Section 7.2).

Note that error correlation files may contain latitudinally binned error correlations and standard deviations, allowing for latitudinally varying error correlation structures and standard deviations in the `FSFC` and `VSFC` methods.

When standard deviations for bending angles are read from the input observation data file using the `VSFC` or `VSDC` methods the diagonal elements of σ are specified in `ropp_fm_roprof2obs` by estimates of the error associated with the bending angle observations.

$$\texttt{obs\%cov\%d(i+i*(i-1)/2)} = \texttt{ro\_data\%Lev2a\%bangle\_sigma(i)}^2 \quad \text{for each level } i$$

7.5 Background data

7.5.1 Defining the state vector: `ropp_fm_roprof2state`

Background meteorological data are represented in `ropp_fm` and `ropp_1dvar` by the `State1dFM` data structure. The relevant background data are copied to elements of `State1dFM` by calling `ropp_fm_roprof2state` (Section 4.4).

The `State1dFM` structure used by `ropp_fm` routines is required to contain temperature, specific humidity and pressure data as a function of geopotential height. Typically the elements of the state vector need to be calculated from the available background data provided by a user and the exact specification of the vertical level representation adopted in the NWP model from which the data are taken. The type of model data contained in the input file is specified by the input variable `bg_data%Lev2d%level_type`. The `ropp_fm` module contains routines to derive the required generic elements of `State1dFM` using background data from a NWP model where the vertical levels is based on pressure levels (e.g. ECMWF, `ropp_fm_state2state_ecmwf`) or geopotential height (e.g. Met Office, `ropp_fm_state2state_meto`). See Sections 4.5 and 4.6 for further details.

State vector

The elements of the state vector `bg%state` used in the `ropp_1dvar` analysis also depend on the exact details of the source of the background data. Note that if the configuration option `config%use_logp` is set to `true` then pressure variables in the state vector and its covariance are expressed as $\ln(p)$. Similarly, it the configuration option `config%use_logq` is set to `true` then specific humidity variables in the state vector and its covariance are expressed as $\ln(q)$. The appropriate conversions are applied by routines `ropp_fm_roprof2state` and `ropp_fm_state2roprof`.

ECMWF

For background data with hybrid vertical levels (e.g. ECMWF), the elements of the state vector are given by vertical profiles of temperature and specific humidity on the background's vertical levels and an

additional surface pressure element. bg%n_lev is the number of background vertical levels.

For config%use_logp and config%use_logq set to false,

```
bg%temp(:) = bg%state(1 : bgn_lev)
bg%shum(:) = bg%state(bg%n_lev + 1 : 2*bg%n_lev)
psfc       = bg%state(2*bg%n_lev +1)
```

For config%use_logp and config%use_logq set to true,

```
bg%temp(:) = bg%state(1 : bgn_lev)
bg%shum(:) = exp[bg%state(bg%n_lev + 1 : 2*bg%n_lev)]
psfc       = exp[bg%state(2*bg%n_lev +1)]
```

Met Office

For background data with geopotential height-based vertical levels (e.g. Met Office), the elements of the state vector are given by vertical profiles of pressure and humidity on the original background's vertical levels. Note that pressure and humidity variables are stored on different levels of a staggered vertical grid in the Met Office Unified Model, so bg%state contains one more pressure element than specific humidity. bg%n_lev is the number of background vertical levels for humidity data.

For config%use_logp and config%use_logq set to false,

```
pressA(:)  = bg%state(1 : bg%n_lev + 1)
bg%shum(:) = bg%state(bg%n_lev + 2 : 2*bg%n_lev + 1)
```

For config%use_logp and config%use_logq set to true,

```
pressA(:)  = exp[bg%state(1 : bg%n_lev + 1)]
bg%shum(:) = exp[bg%state(bg%n_lev + 2 : 2*bg%n_lev + 1)]
```

The ropp_fm_state2state_ecmwf and ropp_fm_state2state_meto routines allow for consistent mapping between the elements of the state vector and the variables required by ropp_fm. It is therefore called each time the state vector is updated in minimising the cost function for example.

7.5.2 Defining the background error covariance matrix: ropp_1dvar_covar_bg

The subroutine ropp_1dvar_covar_bg is used to set up the background error covariance matrix for a background state vector.

```
USE ropp_fm
USE ropp_1dvar
TYPE(State1dFM)   :: bg
TYPE(VarConfig)   :: config
CALL ropp_1dvar_covar(bg, config)
```

The error covariance matrix $\mathbf{B}$ is constructed by computing the matrix product,

$$\mathbf{B} = \sigma.\mathbf{Corr}.\sigma^T \tag{7.3}$$

where $\mathbf{Corr}$ is a matrix containing the correlation between elements in the state vector and σ is a diagonal matrix where the diagonal elements contain the error standard deviations for each element in the state vector. The elements of $\mathbf{O}$ are held in the `State1dFM` structure for use in the `ropp_1dvar` processing as element `bg%cov`.

The background error covariance matrix can be constructed using the following methods in ROPP. The method to be used is specified as a configuration option (Table 7.1).

- **FSFC - Fixed Sigmas, Fixed correlations**

  Both error correlations and error standard deviations are read from a background error correlation file. The error correlation file is specified by the '`--bg-corr <bg_corr_file>`' command-line option or can be set as a default configuration file option. The error correlation file must contain both the error correlation matrix as well as the standard deviations (errors) for all background state vector elements. Note it is assumed that the standard deviations are input in the required format for the user's choice of `config%use_logp` and `config%use_logq` options. Sample files containing background sigma and correlation values are provided in the `errors/` sub-directory of the `ropp_1dvar` distribution (see Section 7.2).
- **VSFC - Variable Sigmas, Fixed Correlations**

  Error correlations are read from an error correlation file, while error estimates are obtained separately for each input profile from the standard deviations contained in the input background data file (and values are automatically adjusted if the user sets either `config%use_logp` or `config%use_logq`). The error correlation file is specified by the '`--bg-corr <bg_corr_file>`' command-line option or can be set as a default configuration file option. In this case the error correlation data files only need to contain the error correlations. Tools for defining variable sigma values in an input profile and sample background error files are provided in the `errors/` sub-directory of the `ropp_1dvar` distribution (see Section 7.2).

Error correlation files may contain latitudinally binned error correlations and standard deviations, allowing for latitudinally varying error correlation structures and standard deviations even in the `FSFC` scenario.

Note that the error standard deviations are dependent on which variables are used to define each element of the background state vector, so that σ is specific to a particular background model type (Sections 4.5, 4.6).

ECMWF

The diagonal elements of σ for the state vector defined for ECMWF background data are specified by the estimated error associated with each meteorological variable.

For config%use_logp and config%use_logq set to false,

bg%cov%d(i+i*(i-1)/2) = ro_data%Lev2b%temp_sigma(i)^2 for each level i

bg%cov%d(j+j*(j-1)/2) = ro_data%Lev2b%shum_sigma(i)^2 for each level $j = \text{bg\%n\_lev} + i$

bg%cov%d(2*bg%n_lev + 1) = ro_data%Lev2b%press_sfc_sigma

For config%use_logp and config%use_logq set to true,

bg%cov%d(i+i*(i-1)/2) = ro_data%Lev2b%temp_sigma(i)^2 for each level i

bg%cov%d(j+j*(j-1)/2) = (...%shum_sigma(i)/...%shum(i))^2 for each level $j = \text{bg\%n\_lev} + i$

bg%cov%d(2*bg%n_lev + 1) = (...%press_sfc_sigma/...%press_sfc)^2

Met Office

The diagonal elements of σ for the state vector defined for Met Office background data are specified by the estimated error associated with each meteorological variable.

For config%use_logp and config%use_logq set to false,

bg%cov%d(i+i*(i-1)/2) = ro_data%Lev2b%press_sigma(i)^2 for each level i

bg%cov%d(j+j*(j-1)/2) = ro_data%Lev2b%shum_sigma(i)^2 for each level $j = \text{bg\%n\_lev} + i$

For config%use_logp and config%use_logq set to true,

bg%cov%d(i+i*(i-1)/2) = (...%press_sigma(i)/...%press(i))^2 for each level i

bg%cov%d(j+j*(j-1)/2) = (...%shum_sigma(i)/...%shum(i))^2 for each level $j = \text{bg\%n\_lev} + i$

7.6 Quality control

Figure 7.4 illustrates the implementation of `ropp_1dvar` module routines to perform preliminary quality control on the input data.

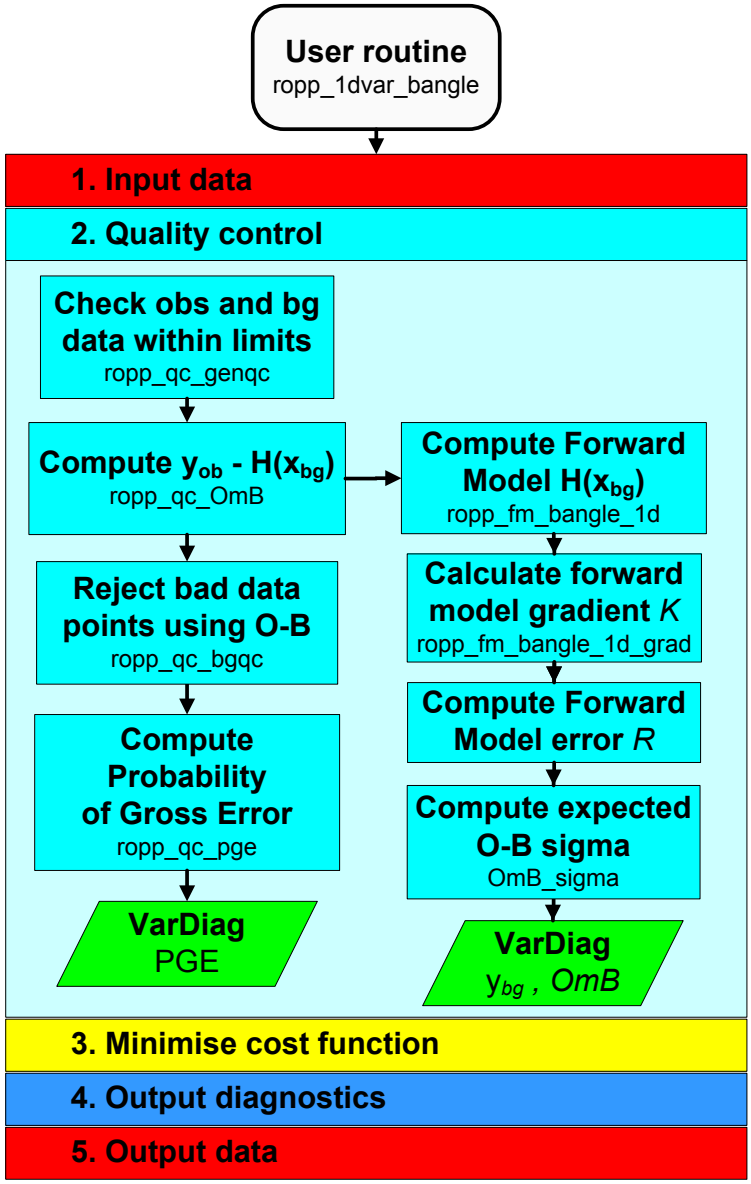

Figure 7.4: Flow chart illustrating the calling tree of the quality control step of ROPP 1D–Var to retrieve atmospheric profiles from observed bending angle profiles and input background model data.

The `ropp_qc` routines fill elements of a structure of type `VarDiag` as defined in `ropp_1dvar_types()`. These parameters may be written to the output file on completion of the `ropp_1dvar` processing. The `ropp_qc` routines determine the status of an overall quality flag `...%ok`. If set to false during the quality control stage the 1D–Var retrieval is not attempted.

```
USE ropp_fm
USE ropp_1dvar
```

```
TYPE(State1dFM)   :: bg
TYPE(Obs1dBangle) :: obs
TYPE(VarConfig)   :: config
TYPE(VarDiag)     :: diag
diag % ok = .true.
IF (diag % ok) CALL ropp_qc_cutoff(obs, onfig)
IF (diag % ok) CALL ropp_qc_genqc(obs, bg, config, diag)
IF (diag % ok) CALL ropp_qc_OmB(obs, bg, config, diag)
IF (diag % ok) CALL ropp_qc_bgqc(obs, config, diag)
IF (diag % ok) CALL ropp_qc_pge(obs, config, diag)
```

7.6.1 Valid observation height range: `ropp_qc_cutoff()`

Configuration options `config%min_1dvar_height` and `config%max_1dvar_height` (Table 7.1) may be set to specify the height range within which observations are used in the 1D–Var analysis. Data outside this height range are given zero weighting and therefore do not contribute to the cost function. Users may wish to set the range of valid observation heights to, for example, exclude biased lower-tropospheric observations or upper-stratospheric climatological information from the 1D–Var analysis.

7.6.2 Generic quality control check: `ropp_qc_genqc()`

Generic quality control checks may be conducted by calling the subroutine `ropp_qc_genqc`. This routine checks that the background and observation data are within the physical ranges specified in the `VarConfig` structure (Table 7.1). The co-location of background and observation profiles is also tested, ensuring that the great circle distance between the observation and background coordinates is within a pre-defined limit `config%genqc_max_distance`. Similarly, the temporal separation of background and observation data can be tested and `config%ok` set to false if the data are not within `config%genqc_max_time_sep`.

7.6.3 Observation minus background check: `ropp_qc_OmB()`

The difference between the observations and the background data forward modelled into observation space ($\texttt{diag\%OmB} = \mathbf{y}_o - \mathbf{H}[\mathbf{x}_{bg}]$) and the expected uncertainty of this difference ($\texttt{diag\%OmB\_sigma} = \sigma_{(\mathbf{y}_o - \mathbf{H}[\mathbf{x}_{bg}])}$) are computed in `ropp_qc_OmB`. This routine calls the relevant `ropp_fm` module forward model to enable comparison between the background data and bending angle (Section 4.9) observations.

The error in the observation minus background calculation is given by

$$\texttt{diag\%OmB\_sigma(:)} = (\mathbf{O} + \mathbf{K}.\mathbf{B}.\mathbf{K}^T)^{1/2} \tag{7.4}$$

where $\mathbf{O}$ and $\mathbf{B}$ are the observation and background data error covariance matrices respectively and $\mathbf{K}$ gives the gradient of the forward model with respect to each element of the state vector. This is computed by calling the `ropp_fm` routine `ropp_fm_bangle_1d_grad` for bending angle.

7.6.4 Background quality control check: `ropp_qc_bgqc()`

The `ropp_1dvar` 1D–Var retrieval is terminated (`diag%ok` set to false) if `config%bgqc_reject_max_percent` or more percent of the observed data are rejected in `ropp_qc_bgqc`. Data points are rejected where

$$\texttt{diag\%OmB} > \texttt{config\%bgqc\_reject\_factor} * \texttt{diag\%OmB\_sigma}$$

The weights of the rejected observation data `obs%weights` are set to zero and do not contribute to the computation of the cost function.

The number of data points used and rejected in the 1D–Var retrieval are stored in the `VarDiag` structure as `diag%n_data` and `diag%n_bgqc_reject` respectively.

7.6.5 Probability of gross error (PGE): `ropp_qc_pge()`

It is possible to screen out observations from the 1D–Var analysis which have gross errors that are inconsistent with the assumed observation errors $\mathbf{O}$. An estimate of the Probability of Gross Error (PGE) can be computed in routine `ropp_qc_pge`. Weights of $(1-PGE)$ can then be applied to elements of the observation vector by setting the configuration option `config%pge_apply`.

The PGE is computed based on the $(\mathbf{y}_o - \mathbf{H}[\mathbf{x}_{bg}])$ difference following the approach outlined by Ingleby and Lorenc (1993) and Andersson and Järvinen (1999). This is stored in the `VarDiag` structure as `diag%pge`.

$$\texttt{diag\%pge} = \frac{\gamma}{\gamma + \exp\left[-\frac{1}{2}\left(\frac{\mathbf{y}_o - \mathbf{H}[\mathbf{x}_{bg}]}{\sigma_{(\mathbf{y}_o - \mathbf{H}[\mathbf{x}_{bg}])}}\right)^2\right]} \tag{7.5}$$

The exponential argument is the contribution to the observation cost function for uncorrelated observation errors. The parameter γ is stored as element `diag%pge_gamma` and computed as

$$\gamma = \frac{A\sqrt{2\pi}}{(1-A)2d} \tag{7.6}$$

where A is the first guess PGE (`config%pge_fg`) and d is the width of the gross error (`config%pge_d`).

7.7 Minimise the cost function

Figure 7.5 illustrates the implementation of `ropp_1dvar` module routines to minimise the cost function and obtain the solution state vector for a given 1D–Var retrieval.

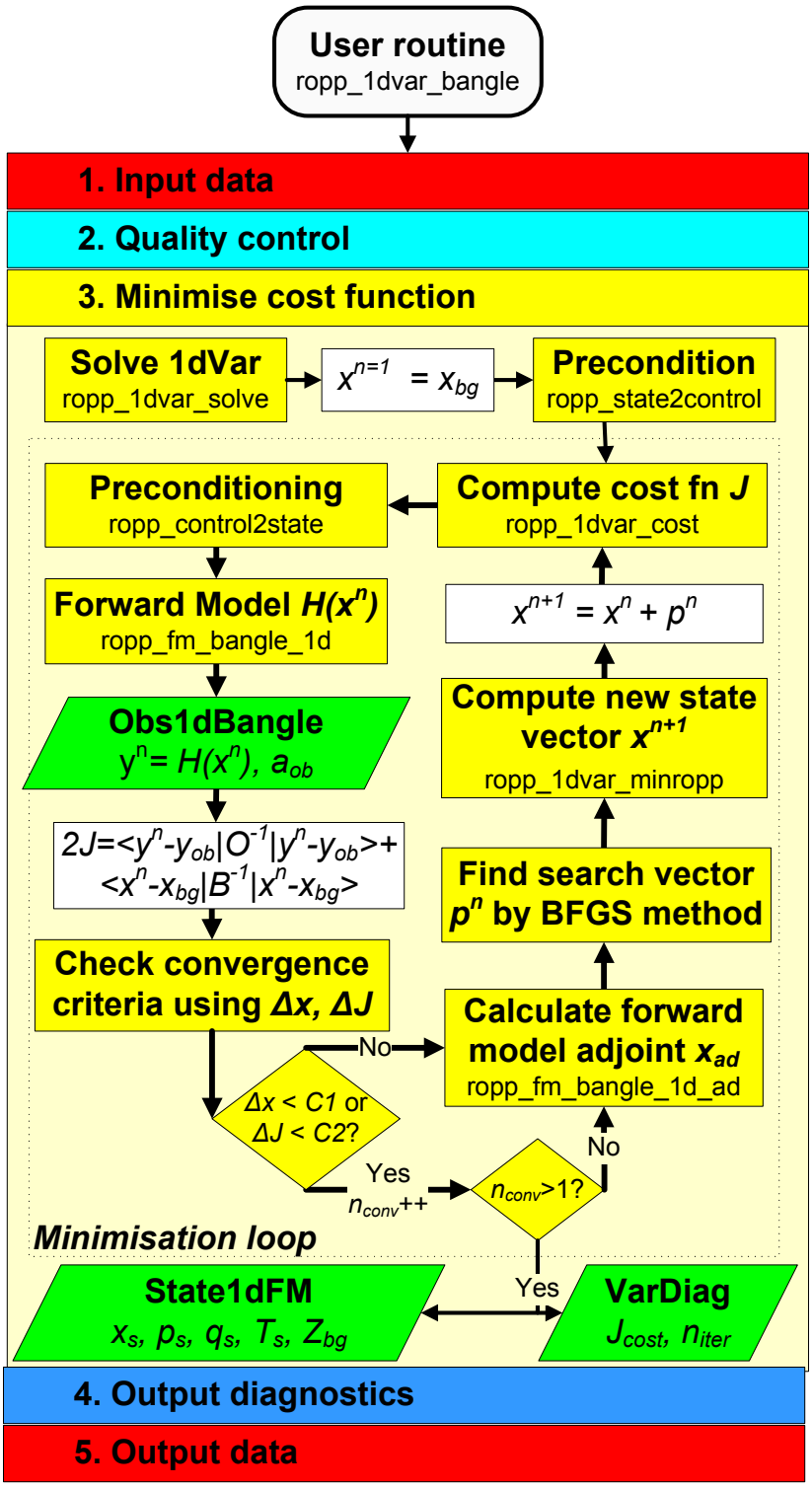

Figure 7.5: Flow chart illustrating the calling tree of the minimisation step of ROPP 1D–Var to retrieve atmospheric profiles from observed bending angle profiles and input background model data.

The main ropp_1dvar routine for retrieving the solution state vector ($\mathbf{x}$) which minimises the cost function (Equation 7.1) is ropp_1dvar_solve. Given input variables of the correct format, ropp_1dvar_solve could be implemented by users as a 'black box' to perform 1D–Var retrievals using either refractivity or bending angle observations. On entry the solution state vector $\mathbf{x}$ of type State1dFM is set to a first guess solution of x_{bg}. On exit $\mathbf{x}$ contains the 1D–Var solution.

```
USE ropp_fm
USE ropp_1dvar
TYPE(State1dFM)   :: bg
TYPE(State1dFM)   :: x
TYPE(Obs1dBangle) :: obs
...
x = bg    ! initial guess - set x equal to background state
CALL ropp_1dvar_solve(obs, bg, x, config, diag)
```

Minimisation of the cost function is achieved by an iterative method which computes the cost function value and updates the state vector on each of $n = 1, .., \mathtt{n\_iter}$ iterations until convergence to a solution is achieved. The state vector is updated using the ROPP-specific minimiser minROPP (ropp_1dvar_minropp). This implements a quasi-Newton method (Nocedal, 1980) using a BFGS algorithm (Press et al., 1992). Further details are provided in ROM SAF (2007).

An alternative Levenberg-Marquardt minimisation algorithm (Press et al., 1992) is also provided with ROPP. The ropp_1dvar_levmarq routine may be used in place of ropp_1dvar_solve. This can be implemented in the ropp_1dvar tools by specifying the minROPP%method configuration option (Table 7.1) as LEVMARQ. Further details on the algorithm and its performance are provided in ROM SAF (2008). Users should note that the default minROPP minimiser provided with ROPP, as described in ROM SAF (2007), is more efficient than the Levenberg-Marquardt algorithm.

7.7.1 Preconditioning

Convergence to a solution during the minimisation step can be accelerated by a change of variable from the state vector to a control variable. If configuration option config%use_precond is set, routine ropp_state2control is used to perform the initial variable transform from the state variable to a control variable , given a preconditioner $\mathbf{L}$ such that $\mathbf{B} = \mathbf{L}\mathbf{L}^T$.

$$\mathbf{control} = \mathbf{L}^{-1}\mathbf{x} \tag{7.7}$$

All other elements of the State1dFM structure are unaffected by this transformation. Note that preconditioning is only applied to the state vector for the minimisation step. The cost function is computed using the state vector $\mathbf{x}$, which can be recovered from **control** by calling the routine ropp_control2state.

7.7.2 Compute the cost function

Equation (7.1) is computed on each iteration in routine `ropp_1dvar_cost`. The cost function `J` is computed for the current state vector $\mathbf{x}$ together with the gradient of `J` with respect to the state vector elements (`J_grad`).

```
USE ropp_fm
USE ropp_1dvar
USE matrix
TYPE(State1dFM)   :: bg
TYPE(State1dFM)   :: control
TYPE(Obs1dBangle) :: obs
TYPE(VarConfig)   :: config
TYPE(matrix_sq)   :: precon
CALL ropp_1dvar_cost(obs, bg, control, precon, J, J_grad, config, indic)
```

To process bending angles, the `ropp_fm` routine `ropp_fm_bangle_1d` is called each time the cost function is evaluated to forward model the current state vector into the observation space. The relative weighting applied to each observation is applied at the cost function stage by scaling the difference $\mathbf{y}_o - \mathbf{H}[\mathbf{x}]$ with the weighting factors determined from the quality control processing.

The matrix multiplication of the differences $(\mathbf{x} - \mathbf{x}_b)$ and $(\mathbf{y}_o - \mathbf{H}[\mathbf{x}])$ with the inverse of the background and observation error covariance matrices respectively is performed using the `matrix_solve` routine. This solves a linear matrix equation of the form $\mathbf{Ax} = \mathbf{b}$ using a Cholesky decomposition (Press et al., 1992).

To minimise the cost function it is necessary to evaluate the gradient of the cost function with respect to the state vector:

$$\nabla J(\mathbf{x}) = \mathbf{B}^{-1}(\mathbf{x} - \mathbf{x}_b) - (\mathbf{H}[\mathbf{x}]')^T \mathbf{O}^{-1}(\mathbf{y}_o - \mathbf{H}[\mathbf{x}]) = \mathbf{0} \tag{7.8}$$

where $\mathbf{H}'$ is the gradient (or tangent linear) of the forward operator with respect to the state vector and $\mathbf{H}'^T$ is termed the adjoint of the forward operator. Tangent linear and adjoint code for the ROPP forward models are provided with the `ropp_fm` module. The adjoint is computed as part of `ropp_1dvar_cost` by calling the routine `ropp_fm_bangle_1d_ad`. On exit `ropp_1dvar_cost` returns the variable `J_grad` which contains the cost function gradient.

7.7.3 Convergence check

If the configuration option `config%conv_check_apply` is set then `ropp_1dvar_cost` performs a check to identify when convergence to a satisfactory solution has been obtained. One of two criteria need to be met on `config%conv_check_n_previous` consecutive iterations for convergence to be achieved.

Maximum relative change in state

Convergence is assumed if the state vector does not change by more than a set value between iterations

$$\max\left[\frac{|\mathbf{x}_n - \mathbf{x}_{n-1}|}{\sqrt{\mathbf{B}}}\right] < \texttt{config\%conv\_max\_delta\_state} \tag{7.9}$$

for at least `config%conv_check_n_previous` consecutive iterations.

Maximum change in cost function

Convergence is assumed if the cost function does not change by more than a set value between iterations

$$|J_n - J_{n-1}| < \texttt{config\%max\_delta\_J} \tag{7.10}$$

for at least `config%conv_check_n_previous` consecutive iterations.

When either of the convergence criteria are met, flag `indic` is returned with a value of zero which terminates the minimisation loop in routine `ropp_1dvar_solve`. If preconditioning was used, the solution control vector is transformed to the state vector using `ropp_control2state`. Pressure, temperature and humidity profiles corresponding to the solution state vector are recovered using a background model-dependent conversion routine `ropp_fm_state2state_ecmwf` (Section 4.5) or `ropp_fm_state2state_meto` (Section 4.6). Diagnostic parameters `diag%J`, `diag%J_scaled` and `diag%n_iter` are set.

7.7.4 Minimisation

If the convergence criteria tested in `ropp_1dvar_cost` are not met, flag `indic` is returned with a value of 4, indicating that further iteration is required. The state vector is incremented towards a solution by the ROPP-specific minimiser `minROPP`. This is a quasi-Newton minimisation routine which finds a search direction vector to determine the updated state vector using a BFGS algorithm (e.g. Press et al. (1992), Nocedal (1980)). A technical summary of this algorithm is provided by ROM SAF (2007).

The `minROPP` routine `ropp_1dvar_minropp` is called as

```
call ropp_1dvar_minropp(control, J_grad, J_dir, dJ, gconv, n_iter,
                        indic, config%minropp%n_iter, maxstore)
```

where **control** is the control state vector, which is updated on exit. `J_grad` is the gradient of the cost function with respect to the control vector, determined by `ropp_1dvar_cost`. `J_dir` is the quasi-Newton search direction vector which is updated by `minROPP`. This determines the updated state vector. `dJ` is the expected decrease of the cost function, which is used in `minROPP` to calculate the initial value of search direction vector `J_dir`. The `n_iter` variable is an iteration counter, which is incremented each time `minROPP` updates the state vector.

An additional convergence check is provided in `minROPP`. Convergence is assumed at iteration n if the gradient of the cost function at $\mathbf{x}_n$ differs from its initial value when $n = 1$ by more than a pre-defined

factor.

$$||\nabla J_k|| < \text{eps}||\nabla J_1|| \tag{7.11}$$

Parameter eps is set as a configuration option as element `config%minROPP%eps_grad`. The routine exits if this condition is met, and the minimisation loop is terminated.

Figure 7.5 illustrates how the minimisation loop continues in `ropp_1dvar_solve` to update the state vector and compute new values for J and ∇J for `n_iter` iterations until convergence is achieved. If a maximum of `config%minROPP%n_iter` iterations have been completed, then no convergence can be achieved and the 1D–Var retrieval fails.

7.8 Output diagnostics

Figure 7.6 illustrates the implementation of `ropp_1dvar` and `ropp_fm` module routines to compute final output diagnostics associated with a 1D–Var retrieval solution. Table 7.2 lists the diagnostics which may be optionally output if the `config%extended_1dvar_diag` configuration flag is set. Further information and examples of these diagnostics are provided by ROM SAF (2010).

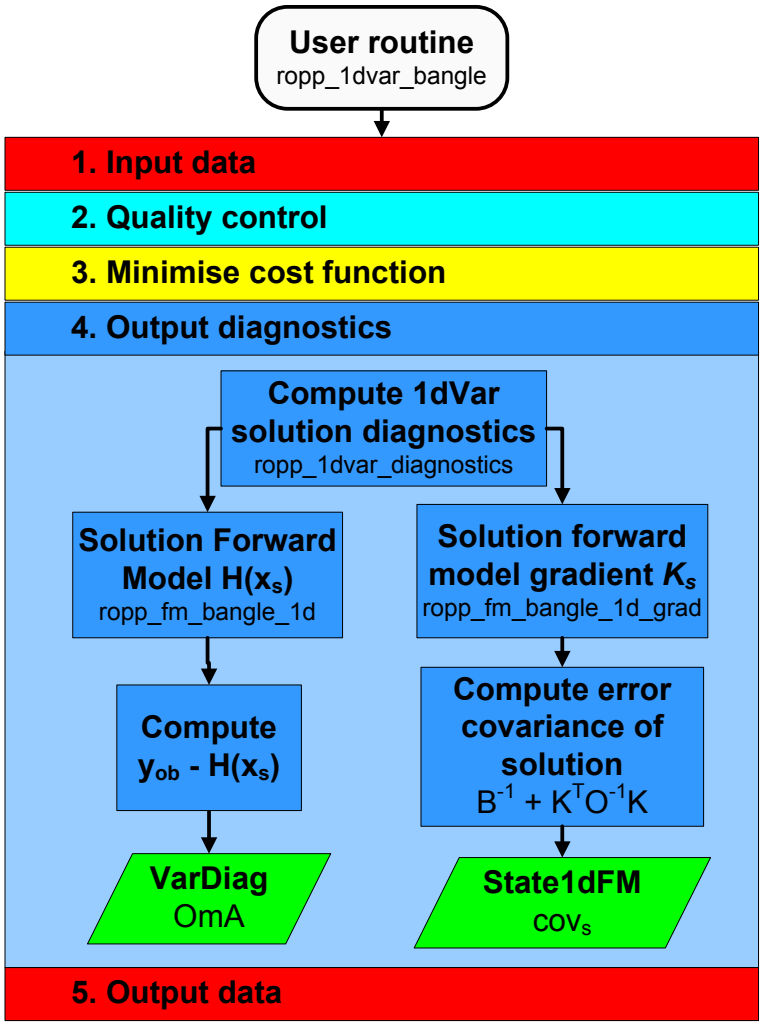

Figure 7.6: Flow chart illustrating the calling tree of the output diagnostics step of ROPP 1D–Var to retrieve atmospheric profiles from observed bending angle profiles and input background model data.

The `ropp_1dvar` diagnostic routine is `ropp_1dvar_diagnostics` which is called as,

```
USE ropp_fm
USE ropp_1dvar
TYPE(State1dFM)    :: x
TYPE(Obs1dBangle) :: obs
TYPE(VarConfig)    :: config
```

```
TYPE(VarDiag)      :: diag
CALL ropp_1dvar_diag2roprof(obs, x, config, diag)
```

The diagnostic processing applied to the solution state vector is similar to the initial quality control checks applied to compare the observation and background data (7.6.3). The diagnostic variable `diag%OmA` is computed as the difference between the observations ($\mathbf{y}_o$) and solution state vector forward modelled into observation space ($\mathbf{H}[\mathbf{x}_s]$). The forward modelled bending angle solution $\mathbf{H}[\mathbf{x}_s]$ is saved as element `diag%res_bangle`.

An estimate of the solution error covariance matrix is obtained using the observation and background error covariance matrices and forward model gradient $\mathbf{K}$ as (Ch 5 Rodgers, 2000)

$$\mathtt{x\%cov} = \left(\mathbf{B}^{-1} + \mathbf{K}^T\mathbf{O}^{-1}\mathbf{K}\right)^{-1} \tag{7.12}$$

The gradient matrix $\mathbf{K}$ gives the gradient of the forward model with respect to each element in the solution state vector. This is computed by calling the routine `ropp_fm_bangle_1d_grad`.

7.9 Output data

The final stage in the `ropp_1dvar` processing is to fill the elements of the generic ROPP structure of type `ROprof` with the 1D–Var solution for writing to an output file using the `ropp_io` module routine `ropp_io_write` (ROM SAF, 2013).

```
USE ropp_io
USE ropp_fm
USE ropp_1dvar
TYPE(ROprof)       :: res_data
TYPE(State1dFM)    :: x
TYPE(VarDiag)      :: diag
TYPE(VarConfig)    :: config
CALL ropp_fm_state2roprof(x, res_data)
CALL ropp_fm_obs2roprof(diag%res_bangle, res_data)
CALL ropp_1dvar_diag2roprof(obs, diag, res_data, config)
```

The solution state vector elements are copied to meteorological variables as Level 2b and Level 2c data in `ropp_fm_state2roprof`. The translation between state vector elements and `ROprof` variables depends on the exact details of the state vector and structure of the background model (Section 4.2.2).

Level 1b (bending angle) data in the `ROprof` structure are filled with the solution state vector forward modelled into observation space. These profiles are given by `diag%res_bangle` returned by `ropp_1dvar_diagnostics`.

Diagnostic parameters gathered during a 1D–Var retrieval are added to the `ROprof` data structure in routine `ropp_1dvar_diag2roprof`. All elements of the `VarDiag` structure may be output if the configura-

tion option `config%extended_1dvar_diag` is set. Otherwise, only the cost function value at convergence and the cost function scaled by the number of observations are output.

7.10 Plotting tools

The directory `tests/` contains the IDL procedure `plot_1dvar.pro` which gives an example of how to read and plot pressure, temperature and humidity increments computed by running the bending angle 1D–Var.

An example of its implementation, using archive data available from the ROM SAF archive http://www.romsaf.org, is provided by running the test script `test_1dvar_GRAS.sh`.

References

Andersson, E. and Järvinen, H., Variational quality control, *Quart. J. Roy. Meteorol. Soc.*, *125*, 697–722, 1999.

Ingleby, N. B. and Lorenc, A. C., Bayesian quality control using multivariate normal distributions, *Quart. J. Roy. Meteorol. Soc.*, *119*, 1195–1225, 1993.

Nocedal, J., Updating quasi-Newton matrices with limited storage, *Mathematics of Computation*, *35*, 773–782, 1980.

Press, W., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical recipes in C – The Art of Scientific Computing*, Cambridge University Press, Cambridge, New York, 2nd edn., 1992.

Rodgers, C. D., *Inverse methods for atmospheric sounding: Theory and practice*, World Scientific Publishing, Singapore, New Jersey, London, Hong Kong, 2000.

ROM SAF, ROPP minimiser - minROPP, SAF/GRAS/METO/REP/GSR/003, 2007.

ROM SAF, Levenberg-Marquardt minimisation in ROPP, SAF/GRAS/METO/REP/GSR/006, 2008.

ROM SAF, ROPP 1dVar Validation, SAF/GRAS/METO/REP/GSR/011, 2010.

ROM SAF, The Radio Occultation Processing Package (ROPP) User Guide. Part I: Input/Output module, SAF/ROM/METO/UG/ROPP/002, Version 7.0, 2013.

VarConfig		
Structure element	Default	Description
...%obs_file	ropp_obs.nc	Input observation data file
...%obs_covar_method	VSDC	Observation error covariance method
...%obs_corr_file	ropp_obs_corr.nc	Observation error correlation file
...%bg_file	ropp_bg.nc	Input background data file
...%out_file	ropp_out.nc	Output file
...%bg_covar_method	VSFC	Background error covariance method
...%bg_corr_file	ropp_bg_corr.nc	Background error correlation file
...%min_1dvar_height	-10 km	Min. obs height to include in 1D–Var
...%max_1dvar_height	60 km	Max. obs height to include in 1D–Var
...%genqc_colocation_apply	.true.	Apply obs and bg colocation check?
...%genqc_max_distance	300 km	Max. obs to bg great circle distance
...%genqc_max_time_sep	300 sec	Max. obs to bg temporal separation
...%genqc_min_temperature	150 K	Min./Max. bg data values
...%genqc_max_temperature	350 K	
...%genqc_min_spec_humidity	0 g/kg	
...%genqc_max_spec_humidity	50 g/kg	
...%genqc_min_impact	6.2e6 m	Min./Max. obs data values
...%genqc_max_impact	6.6e6 m	
...%genqc_min_bangle	-1.0e-4 rad	
...%genqc_max_bangle	0.1 rad	
...%genqc_min_geop_refrac	-1.e3 m	
...%genqc_max_geop_refrac	1.e5 m	
...%genqc_min_refractivity	0.0	
...%genqc_max_refractivity	500	
...%bgqc_apply	.true.	Apply background quality control?
...%bgqc_reject_factor	10	Data rejected if O-B > factor * sigma
...%bgqc_reject_max_percent	50	Maximum %age data rejected
...%pge_apply	.false.	Apply PGE for quality control?
...%pge_fg	0.001	First guess PGE
...%pge_d	10.0	Width of gross error plateau
...%minROPP%method	MINROPP	minimisation method
...%minROPP%log_file	screen	minROPP output device
...%minROPP%impres	0	minROPP output mode
...%minROPP%n_iter	1500	maximum number of iterations
...%minROPP%eps_grad	1.0e-8	minROPP convergence parameter
...%use_precond	.true.	Use preconditioning?
...%conv_check_apply	.true.	Apply additional convergence checks?
...%conv_check_n_previous	2	No. of previous iterations to check
...%conv_check_max_delta_state	0.1	Maximum change in state vector
...%conv_check_max_delta_J	0.1	Maximum change in cost function
...%extended_1dvar_diag	.false.	Output additional diagnostics?
...%use_logp	.false.	Use log(pres) in 1D–Var
...%use_logq	.false.	Use log(shum) in 1D–Var

Table 7.1: Configuration options held as elements of the `VarConfig` structure which are used by `ropp_1dvar` routines. The default values are assumed unless overwritten by reading configuration options from an input file.

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

Table 7.2: Elements of `VarDiag` structure output using the `extended_1dvar_diag` configuration flag.

8 ROPP FM and 1D–Var: Including non-ideal gas effects

8.1 Background

In their default configuration, the ROPP 1D and 2D forward operators assume an ideal gas when computing the hydrostatic integration and the refractivity values from pressure, temperature and humidity. However, Aparicio et al. (2009) have recently demonstrated that neglecting the non-ideal gas effects can be a source of forward model error.

We have introduced an non-ideal gas option in all of the 1D and 2D forward models. The inclusion of non-ideal gas effects is inextricably linked to the choice of refractivity coefficients used in the forward calculation. We have chosen the "best average" refractivity coefficients provided by Rueger (2002), but with a k_1 coefficient adjusted for use in a refractivity expression that includes non-ideal gas compressibility (Thayer, 1974). A discussion on the uncertainty of the coefficients can be found in Cucurull (2010) and Healy (2011).

8.2 Effects of non-ideal gas compressibility

The ROPP refractivity and bending angle observation operator contains essentially three major components. The first is the integration of the hydrostatic equation, to compute the height of the model levels. The second is the evaluation of refractivity on the model levels, and the third is either interpolating the model refractivity to the observed refractivity height, or the evaluation of the bending angle integral, for the refractivity and bending angle operators, respectively.

Non-ideal gas effects should be included in both the integration of the hydrostatic equation and the evaluation of the refractivity on the model levels. Note that Aparicio et al. (2009) do not include the impact of non-ideal gas effects on the evaluation of the refractivity on the model levels.

Following Aparicio et al. (2009) the non-ideal gas equation of state can be written as,

$$P = \rho R T_v Z \tag{8.1}$$

where P is the total pressure, ρ is the density, R is the gas constant for dry air, T_v is the virtual temperature and Z is the compressibility of moist air. The compressibility accounts for non-ideal gas effects, such as the finite size of the molecules and mutual attraction, and it is a function of pressure, temperature and water vapour pressure. For an ideal gas $Z = 1$, but for air in the troposphere typically $Z \sim 0.9995$, so the departure is 0.05 %. Picard et al. (2008) provide a polynomial expansion for Z, which is straightforward to implement in the ROPP observation operators.

If we include the compressibility in the hydrostatic integral, the geopotential height, h, becomes

$$h = -\int \left(\frac{ZRT_v}{g_o P} \right) dp \tag{8.2}$$

where $g_o = 9.80665 \mathrm{ms}^{-2}$. Given that $Z < 1$, it is clear that including compressibility reduces the height of model levels. In the ROPP operators, the thickness between the ith and $(i+1)th$ model levels initially calculated neglecting non-ideal gas effects, denoted by $\Delta h'_{i,i+1}$, is scaled by the average of the compressibility at ith and $(i+1)th$ levels, to give $\Delta h_{i,i+1} = 0.5 \times (Z_i + Z_{i+1}) \times \Delta h'_{i,i+1}$. This approach is equally valid for ECMWF and Met Office height based vertical grids.

To put these adjustments in some context, in simulations we have found that the height of model levels near 100 hPa can be reduced by $dh \sim 7\mathrm{m}$. Assuming a refractivity scale height of 7 km, $\Delta N/N = -dh/7\mathrm{km}$, this translates into a reduction in the forward modelled refractivity or bending angle value of -0.1%.

The second step where non-ideal effects should be included is the computation of refractivity on the model levels. Using a general expression that includes non-ideal gas compressibility (Thayer, 1974)

$$N = \frac{k_1 P_d}{Z_d T} + \frac{k_2 e}{Z_w T^2} + \frac{k_3 e}{Z_w T} \tag{8.3}$$

where Z_d and Z_w are the compressibility of the dry air and water vapour, respectively. Again we use the Picard et al. (2008) polynomial to determine the dry and wet compressibilities. Note that the introduction of compressibility in this expression increases the refractivity value, whereas introducing compressibility in the hydrostaic integration reduces the value.

When running the ROPP observation operators with non-ideal gas effects included, we use the "best average" coefficients given by Rueger (2002). However, we adjust the best average k_1 value to account for its use in a refractivity expression that includes compressibility (Healy, 2011). This reduces the value by around 0.05 % to $k_1 = 77.643\ \mathrm{KhPa}^{-1}$.

8.3 Code structure

The introduction of non-ideal gas effects has required some modification of the forward operators. The 1D and 2D statevectors `State1dFM` and `State2dFM` contain a logical switch `x%non_ideal`. (These are set to .FALSE. in `ropp_fm_types.f90` by default.)

We can use sections of the `ropp_fm_refrac_1d` code to demonstrate the structure, since the same approach is used in both `ropp_fm_bangle_1d` and `ropp_fm_bangle_2d`:

```
! set inverse of compressibilities

  zcomp_dry_inv(:) = 1.0_wp
  zcomp_wet_inv(:) = 1.0_wp

! initialise geopotential heights
```

```
  z_geop(:) = x%geop(:)

  IF (x%non_ideal) THEN

! if non ideal gas calculation, use adjusted coefficients

     kap1 = kappa1_comp
     kap2 = kappa2_comp
     kap3 = kappa3_comp

!    calculate compressibility and adjust geopotential heights in z_geop

     CALL ropp_fm_compress(x,z_geop,zcomp_dry_inv,zcomp_wet_inv)

  ELSE

     kap1 = kappa1
     kap2 = kappa2
     kap3 = kappa3

  ENDIF
```

To summarize the above, the inverse of the dry and wet compressibility values are initialised to unity, and we copy the model level geopotential height values into `z_geop(:)`. The code then tests `x%non_ideal` and if `.TRUE.` selects the (modified) Rueger (2002) coefficients and calls `ropp_fm_compress`. This routine reduces the model geopotential height values in `z_geop(:)`, and calculates the inverse dry and wet compressibilities.

Since the operators use a three term refractivity expression, we then calculate the dry air and vapour pressure,

```
  pwvp = x%pres * x%shum / (epsilon_water + (1.0_wp - epsilon_water)*x%shum)

  pdry = x%pres - pwvp
```

The refractivity on the model levels is then calculated with the three term expression

```
  refrac = kap1 * pdry * zcomp_dry_inv/ x%temp +  &
           kap2 * pwvp * zcomp_wet_inv/ x%temp**2 + &
           kap3 * pwvp * zcomp_wet_inv/ x%temp
```

As noted, this expression defaults to the standard two-term Smith and Weintraub (1953) formula when `x%non_ideal=.FALSE.` since `kap1=kap3` and the inverse compressibility factors are set to unity.

Note that the corresponding tangent linear and adjoint routines have been modified to be consistent with these changes in the forward model.

8.4 Running ROPP routines with non-ideal effects included

The ROPP forward model tools, `ropp_fm_bg2ro_1d` and `ropp_fm_bg2ro_2d`, and the 1D–Var tools, `ropp_1dvar_refrac` and `ropp_1dvar_bangle`, have been modified to include a new command line argument `-comp`. So, for example, the 2D bending angle operator tool, `ropp_fm_bg2ro_2d`, can be run using the command:

```
ropp_fm_bg2ro_2d <inputdatafile> -comp -o <outputfile>
```

where `<inputdatafile>` is a ROPP netCDF file (ROM SAF, 2013) containing the input model data and `<outputfile>` will contain the forward modelled bending angle profiles on pre-defined observation levels, computed with non-ideal gas effects included in the calculation.

The following test routines also run with a `-comp` command-line option:

- `t_fascod`
- `t_fascod_tl`
- `t_fascod_ad`
- `t_twodop`
- `t_twodad`
- `t_twodtl`

The test scripts

- `ropp_fm/test/test_fm_GRAS.sh`
- `ropp_fm/test/test_fm_2D.sh`
- `ropp_1dvar/test/test_1dvar.sh`
- `ropp_1dvar/test/test_1dvar_MO.sh`
- `ropp_1dvar/test/test_1dvar_GRAS.sh`

have been extended to test this functionality.

References

Aparicio, J., Deblonde, G., Garand, L., and Laroche, S., The signature of the atmospheric compressibility factor in COSMIC, CHAMP and GRACE radio occultation data, *J. Geophys. Res.*, p. doi:10.1029/2008JD011156, 2009.

Cucurull, L., Improvement in the use of an operational constellation of GPS radio-occultation receivers in weather forecasting, *Weather and Forecasting*, *25*, 749–767, 2010.

Healy, S., Refractivity coefficients used in the assimilation of GPS radio occultation measurements, *J. Geophys. Res.*, *116*, D01 106, doi:10.1029/2010JD014 013, 2011.

Picard, A., Davis, R., Glaser, M., and Fujii, K., Revised formula for the density of moist air (CIPM-2007), *Metrologia*, *25*, 149–155, 2008.

Rueger, J. M., Refractive index formulae for electronic distance measurement with radio and millimetre waves, Unisurv Report S-68. School of Surveying and Spatial Information Systems, University of New South Wales, [Summary at http://www.fig.net/pub/fig_2002/Js28/JS28_rueger.pdf], 2002.

ROM SAF, The Radio Occultation Processing Package (ROPP) User Guide. Part I: Input/Output module, SAF/ROM/METO/UG/ROPP/002, Version 7.0, 2013.

Smith, E. K. and Weintraub, S., The constants in the equation for atmospheric refractivity index at radio frequencies, in *Proc. IRE*, vol. 41, pp. 1035–1037, 1953.

Thayer, G. D., An improved equation for the refractive index of air, *Radio Sci.*, *9*, 803–807, 1974.

A ropp_utils library

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

A.1 Missing data values

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

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

A.2 ropp_messages

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

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

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

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

```
USE messages
msg_MODE = VerboseMode           ! Enable all messages
```

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

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

A.3 Unitconvert

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

A.4 Coordinates

A collection of low-level F90 coordinate manipulation routines. Functionality includes routines to convert cartesian position vectors between Earth Centred Fixed and Earth Centred Inertial reference frames, convert between cartesian and geodetic position description, compute impact parameter, occultation point, radius of curvature and undulation (ie, difference between the EGM96[1] geoid and the WGS84[2] ellipsoid). Vector manipulation routines (vector product, vector angle, rotation) are also included.

A.5 Datetime

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

A.6 Geodesy

A collection of low-level F90 geodetic conversion routines to convert to/from geometric/geopotential heights and compute Earth's effective radius and gravity. These routines are based on Somigliana's equation to compute height scales relative to the WGS-84 reference ellipsoid.

[1] See http://cddis.nasa.gov/926/egm96/egm96.html.
[2] See http://earth-info.nga.mil/GandG/wgs84/index.html.

A.7 Arrays

A collection of low-level F90 array manipulation routines supporting all data types. The utilities were written by Christian Marquardt as personal code, independent of the ROM SAF. The author grants a free-use licence for all of this code.

A.8 Misc

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

A.8.1 typeSizes

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

B Installing and using ROPP

B.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.

B.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 are 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.

B.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_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 B.4 for further details.

B.3.1 NetCDF

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 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/

Note that the `tests` subdirectory of the `ropp_io` distribution contains a simple test to check if the `netCDF` installation works; see Section B.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_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_fm.a` and `libropp_1dvar.a` can be compiled and installed without `ropp_io` and thus `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`).

B.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. If neither BUFR library is detected by the installation configure script, then these tools will not be built.

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

http://www.ecmwf.int/products/data/software/bufr.html

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

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

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

or

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

while the ECMWF library would be be built with

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

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

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

or for the ECMWF library:

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

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

B.3.3 GRIB_API (optional)

The GRIB background reading tool `grib2bgrasc` in `ropp_io` requires the ECMWF GRIB_API library to be pre-installed. If it fails to be detected by the installation configure script, then this tools will not be built.

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

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

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

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

B.3.4 netCDF4/HDF5 (optional)

The 'EUMETSAT-formatted' RO reading tools `eum2ropp` and `eum2bufr` in `ropp_io` require the installation of a netCDF4 library, with its attendant HDF5 and zlib libraries.

These can be found from

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

http://www.hdfgroup.org/ftp/HDF5/releases

and

http://www.zlib.net/

respectively.

The ROPP `buildpack` script allows installation of these libraries as follows:

```
> buildpack zlib <compiler>
> buildpack hdf5 <compiler>
> buildpack netcdf4 <compiler>
```

(They would need to be installed in this order, since each depends on the one before.) Note that the `ropp_io` tool `eum2bufr` has only been interfaced to the ECMWF BUFR library. Note too that the `zlib`, and possibly also the `HDF5` libraries may already be installed as part of a standard Linux distribution, in which case the user need not of course build a local version.

B.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.

B.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.

B.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]
```

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

where <package> is on of the supported package names (e.g. ropp_fm, ropp_io, netcdf, mobufr, ecbufr, etc.) and <comp> is the required compiler (e.g. ifort, nag, xlf95, 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.

Other shell scripts build* are provided which can be used to further automate the build process by calling buildpack with a pre-determined sequence of packages or compilers. Users should review and edit these to suit their requirements.

B.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
ii) ropp_utils
iii) ropp_io (if required)
iv) ropp_pp (if required)
v) ropp_fm (if required)
vi) 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.

B.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-7.0.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. 7.0), as in ropp_utils-7.0.tar.gz. If GNU tar is available (as on Linux systems), gzipped tar files can be unzipped with

```
> tar -xvzf ropp-7.0.tar.gz
```

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

```
> gunzip -c ropp-7.0.tar.gz | tar -xv
```

In all cases, a new subdirectory named (in the above example) ropp-7.0 will be created which contains the source code of the complete package.

B.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, nag, pgf, g95...) and <os> is the operating system ID, as output by the uname(1) command but entirely in lower case (linux, cygwin, hp-ux...). 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 are all installed under /usr/local, i.e. the libraries can be found in the directory /usr/local/lib, and header and module files in /usr/local/include. It is further assumed that the individual ROPP packages are to be installed under $ROPP_ROOT, i.e. libraries are intended to reside in $ROPP_ROOT/lib, programs in $ROPP_ROOT/bin, and headers 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 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 can nevertheless proceed – the parts related to the missing library are simply not built. Should the build process fail in correctly finding the, e.g., `BUFR` libraries and therefore not build the `BUFR` tools, `config.log` should again provide evidence on what went wrong.

B.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 B.7.

Currently supported Fortran compilers include (on Linux unless otherwise stated): Intel's `ifort` (v9.x, v10.x and v11.x, XE (v12.x)); NAG's `f95` (v5.1), `nagfor` (v5.2); Portland Group's `pgf95` (v7, v11); SUN's `sunf95` (v8 with SunStudio 12); GNU's `g95` and `gfortran` on Linux and under Cygwin on MS Windows; IBM's (xlf95) on AIX. This list may be contracted or extended in the future. For a full list please refer to the ROPP Release Notes and README files in each sub-package.

B.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. In the above example, this would be the home directory. If no --prefix is given, the installation root directory defaults to /usr/local which would normally require root (sudo) privileges.

B.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.

B.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 \
                    -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.)

B.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 verify the output and display test results using standard viewer utilities (e.g. xv). (Future releases may use Python instead.) Tests that generate graphical output automatically display a corresponding reference figure (part of the distribution), against which the test result can be compared. By setting the environment variable $ROPP_PAUSE to TRUE, the user can examine the two figures at leisure, before allowing the test script to move on to the next figure by hitting any key.

B.7.1 ropp_utils

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

B.7.2 ropp_io

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

```
> make test_netcdf
```

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

A second test can be run with

```
> make test_ropp
```

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

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

```
> make test_bufr
```

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

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

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

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

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

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

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

Finally, the command

```
> make test
```

will run all of the above described tests.

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

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

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

B.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 compare the ionospheric correction processing with known output. It also checks the tropopause height tool. Run

```
> make test
```

to check if solutions agree to within expected limits.

B.7.4 `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 consistency checks.

B.7.5 `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).

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

```
> make test
```

to check if everything is working correctly. The test runs the 1D–Var application to generate the result file, then runs an IDL script to read the background and results files, perform mean difference calculations and generate a plot file. Finally, XV is run to display the plot. If all is well, IDL-reported mean percentage differences should be, for all practical purposes, zero and the plots should show a series of vertical lines (each line is one profile, and each is offset from zero by a different amount for clarity) showing the bias in retrieved temperature, specific humidity and pressure with height.

B.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), although 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 compiling is fine, 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.

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` 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 a option in a configuration file; the third by means of a command line option in (one 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.

C `ropp_fm` program files

The `ropp_fm` module provides one-dimensional forward operators to compute vertical refractivity and bending angle profiles from background data on height-based and hybrid NWP model vertical grids. A two-dimensional forward operator to compute a plane bending angle profile from a two-dimensional plane of background data is also included. Tangent linear, adjoint and gradient codes to the forward operators are provided for use in assimilation processing.

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`
- Optional packages: `ropp_io`, `netcdf`

- Stand-alone tools and test programs (*optional)*

tools/

ropp_fm_bg2ro_1d.f90 (requires `ropp_io`)
ropp_fm_bg2ro_2d.f90 (requires `ropp_io`)

tests/

test_fm_GRAS.sh (requires `ropp_io`)
test_fm_2D.sh (requires `ropp_io`)
t_fascod.f90 (requires `ropp_io`)
t_fascod_tl.f90 (requires `ropp_io`)
t_fascod_ad.f90 (requires `ropp_io`)
t_twoop.f90 (requires `ropp_io`)
t_twotl.f90 (requires `ropp_io`)
t_twoad.f90 (requires `ropp_io`)

- Integrated code

`bangle_1d/`

`ropp_fm_bangle_1d.f90`
`ropp_fm_bangle_1d_grad.f90`
`ropp_fm_bangle_1d_tl.f90`
`ropp_fm_bangle_1d_ad.f90`
`ropp_fm_abel.f90`
`ropp_fm_abel_tl.f90`
`ropp_fm_abel_ad.f90`

`bangle_2d/`

`ropp_fm_bangle_2d.f90`
`ropp_fm_bangle_2d_tl.f90`
`ropp_fm_bangle_2d_ad.f90`
`ropp_fm_alpha2drk.f90`
`ropp_fm_alpha2drk_ad.f90`
`ropp_fm_alpha2drk_td.f90`
`ropp_fm_gpsderivs.f90`

```
      ropp_fm_gpsderivs_ad.f90
      ropp_fm_gpsderivs_tl.f90

refrac_1d/
      ropp_fm_refrac_1d.f90
      ropp_fm_refrac_1d_grad.f90
      ropp_fm_refrac_1d_tl.f90
      ropp_fm_refrac_1d_ad.f90

math/matrix/
      matrix_types.f90

model_ecmwf/
      ropp_fm_state2state_ecmwf.f90
      ropp_fm_state2state_ecmwf_tl.f90
      ropp_fm_state2state_ecmwf_ad.f90

model_meto/
      ropp_fm_state2state_meto.f90
      ropp_fm_state2state_meto_tl.f90
      ropp_fm_state2state_meto_ad.f90
```

```
common/
      ropp_fm.f90
      ropp_fm_types.f90
      ropp_fm_constants.f90
      ropp_fm_free.f90
      ropp_fm_interpol.f90
      ropp_fm_interpol_tl.f90
      ropp_fm_interpol_ad.f90
      ropp_fm_interpol_log.f90
      ropp_fm_interpol_log_tl.f90
      ropp_fm_interpol_log_ad.f90
      ropp_fm_obs2obs.f90
      ropp_fm_state2state.f90
      ropp_fm_copy.f90 (requires ropp_io)
      ropp_fm_set_units.f90 (ropp_io)
      ropp_fm_roprof2obs.f90 (ropp_io)
      ropp_fm_roprof2state.f90 (ropp_io)
      ropp_fm_obs2roprof.f90 (ropp_io)
      ropp_fm_state2roprof.f90 (ropp_io)
```

D ropp_1dVar program files

The ropp_1dvar module provides 1D–Var and minimiser routines for retrieval of pressure/height, temperature and humidity from a refractivity or bending angle profile, given NWP background data and observation and background covariance matrices.

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_fm, ropp_utils
- Optional packages: ropp_io, netcdf

- Stand-alone tools and test programs (*optional)*

tools/

ropp_1dvar_bangle.f90 (requires ropp_io)
ropp_1dvar_refrac.f90 (requires ropp_io)

tests/

test_1dvar.sh (requires ropp_io)
test_1dvar_GRAS.sh (requires ropp_io)
test_1dvar_MO.sh (requires ropp_io)

- Integrated code

```
common/
    ropp_1dvar.f90
    ropp_1dvar_types.f90
    ropp_1dvar_cost.f90
    ropp_1dvar_solve.f90
    ropp_1dvar_diagnostics.f90
    ropp_1dvar_levmarq.f90
    ropp_1dvar_minropp.f90
    ropp_1dvar_read_config.f90
    ropp_1dvar_copy.f90 (ropp_io)
    ropp_1dvar_covar_bg.f90 (ropp_io)
    ropp_1dvar_covar_bangle.f90 (ropp_io)
    ropp_1dvar_covar_refrac.f90 (ropp_io)
    ropp_1dvar_diag2roprof.f90 (ropp_io)

qc/
    ropp_qc.f90
    ropp_qc_OmB.f90
    ropp_qc_bgqc.f90
    ropp_qc_cutoff.f90
    ropp_qc_genqc.f90
    ropp_qc_pge.f90

math/precon/
    ropp_control2state.f90
    ropp_control2state_ad.f90
    ropp_state2control.f90
```

```
                                                matrix_operators.f90
    math/matrix/                                matrix_pp2full.f90
         matrix.f90                             matrix_pp2full_alloc.f90
         matrix_assign.f90                      matrix_pp2full_subset.f90
         matrix_bm2full.f90                     matrix_solve.f90
         matrix_bm2full_alloc.f90               matrix_sqrt.f90
         matrix_delete.f90                      matrix_svd.f90
         matrix_full2bm.f90                     matrix_toast.f90
         matrix_full2bm_alloc.f90               matrix_types.f90
         matrix_full2pp.f90
         matrix_full2pp_alloc.f90
         matrix_invert.f90
```

- Support data:

errors/

A collection of observation and background error correlation and standard deviation files and tools which users may find helpful for setting up input to `ropp_1dvar` tools. See `ropp_1dvar/errors/README` and Sections 6.2 and 7.2 for details.

E ROPP extra data

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

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

E.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 E.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`$^{1/2}/\alpha_N$].100%
`...%L2_badness`	L2 phase correction badness score

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

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

E.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 E.4.

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

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

F ROPP user documentation

Title	Reference	Description
ROPP Overview	SAF/ROM/METO/UG/ROPP/001	Overview of ROPP and package content and functionality
ROPP User Guide. Part I: Input/output module	SAF/ROM/METO/UG/ROPP/002	Description of `ropp_io` module content and functionality
ROPP User Guide. Part II: Forward model and 1D–Var modules	SAF/ROM/METO/1DVAR/ROPP/003	Description of `ropp_fm` and `ropp_1dvar` module content and functionality
ROPP User Guide. Part III: Pre-processor module	SAF/ROM/METO/PP/ROPP/004	Description of `ropp_pp` 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
WMO FM94 (BUFR) Specification for Radio Occultation Data	SAF/ROM/METO/FMT/BUFR/001	Description of BUFR template for RO data

Table F.1: ROPP user documentation

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

Table F.2: ROM SAF Reports

Title	Reference	Description
Product Requirements Document (PRD)	SAF/GRAS/METO/MGT/PRD/001	

Table F.3: Applicable documents

G Acronyms and abbreviations

AC	Analysis Correction (NWP assimilation technique)
API	Application Programming Interface
BG	Background
CASE	Computer Aided Software Engineering
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)
CODE	Centre for Orbit Determination in Europe
COSMIC	Constellation Observing System for Meteorology, Ionosphere & Climate
DMI	Danish Meteorological Institute
DoD	US Department of Defense
EC	European Community
ECF	Earth–centred, Fixed coordinate system
ECI	Earth–centred, Inertial coordinate system
ECMWF	The European Centre for Medium-Range Weather Forecasts
EGM–96	Earth Gravity Model, 1996. (US DoD)
EOP	Earth Orientation Parameters
EPS	EUMETSAT Polar System
ESA	European Space Agency
ESTEC	European Space Research and Technology Centre (ESA)
EU	European Union
EUMETSAT	European Organisation for the Exploitation of Meteorological Satellites
GALILEO	European GNSS constellation project (EU)
GCM	General Circulation Model
GFZ	GFZ Helmholtz Centre (Germany)
GLONASS	Global Navigation Satellite System (Russia)
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)
GRAS	GNSS Receiver for Atmospheric Sounding (onboard Metop)
GUI	Graphical User Interface
GTS	Global Telecommunications System

HIRLAM	High Resolution Limited Area Model
IERS	International Earth Rotation Service
ITRF	International Terrestrial Reference Frame
ITRS	International Terrestrial Reference System
IGS	International GPS Service
JPL	Jet Propulsion Laboratory (NASA)
LAM	Local Area Model (NWP concept)
LEO	Low Earth Orbited
LGPL	Lesser GPL (*q.v.*)
LOS	Line Of Sight
METOP	Meteorological Operational polar satellites (EUMETSAT)
MKS	Meter, Kilogram, Second
MPEF	Meteorological Products Extraction Facility (EUMETSAT)
MSL	Mean Sea Level
N/A	Not Applicable or Not Available
NASA	National Aeronautics and Space Administration (US)
NMS	National Meteorological Service
NOAA	National Oceanic and Atmospheric Administration (US)
NPOESS	National Polar-orbiting Operational Environmental Satellite System (US)
NRT	Near Real Time
NWP	Numerical Weather Prediction
OI	Optimal Interpolation (NWP assimilation technique)
Operational ROM SAF	Team responsible for the handling of GRAS data and the delivery of meteorological products during the operational life of the instrument
PAZ	Spanish Earth Observation Satellite, carrying a Radio Occultation Sounder
PFS	Product Format Specifications
PMSL	Pressure at Mean Sea Level
POD	Precise Orbit Determination
Q/C	Quality Control
RO	Radio Occultation
ROC	Radius Of Curvature
ROM SAF	The EUMETSAT Satellite Application Facility responsible for operational processing of radio occultation data from the Metop satellites. 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
TP	Tangent Point
UKMO	United Kingdom Meteorological Office
UML	Unified Modelling Language
UT1	Universal Time-1 (proportional to the rotation angle of the Earth)
UTC	Universal Time Coordinated
VAR	Variational analysis; 1D, 2D, 3D or 4D versions (NWP data assimilation technique)
VT	Valid or Verification Time
WGS–84	World Geodetic System, 1984. (US DoD)
WMO	World Meteorological Organization
WWW	World Weather Watch (WMO)

H Definitions

RO data products from the GRAS instrument onboard Metop and RO data from other data providers are grouped in levels and are either NRT or offline products.

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, SNRs, amplitudes, orbit information, I, Q, and NCO values, and navigation bits;
- Level 1b: Bending angles and impact parameters, Earth location, metadata and quality information;
- Level 2: Refractivity profiles (level 2a), and pressure, temperature, and specific humidity profiles (level 2b and 2c), Earth location, metadata, and quality information;
- Level 3: Gridded level 1 and 2 offline profile products in the form of, e.g., monthly and seasonal zonal means, metadata, and quality information.

Product types:

- NRT product: data product delivered less than 3 hours after measurement;
- Offline product: data product delivered less than 30 days after measurement (the timeliness for some offline level 3 products may be up to 6 months).

I 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

IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

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

© Copyright 1998-2010 Michael Gorbunov <michael.gorbunov@zmaw.de>

All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software with the rights to use, copy, modify, merge copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software as well as in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE. HOWEVER, ALL EFFORTS ARE BEING MADE BY THE AUTHOR IN ORDER TO FIND AND ELIMINATE ALL POSSIBLE ERRORS AND PROBLEMS. IN THIS CASE THE AUTHOR MAY PROVIDE AN UPDATE.

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

© Copyright 1998 Stig Syndergaard <ssy@dmi.dk>

All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sublicense the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software as well as in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.

IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.