EUMETSAT

ROM SAF

RADIO OCCULTATION METEOROLOGY

The Radio Occultation Processing Package (ROPP) 1D–Var Module User Guide

Version 11.0

31 December 2021

The ROM SAF Consortium

Danish Meteorological Institute (DMI)

European Centre for Medium-Range Weather Forecasts (ECMWF)

Institut d'Estudis Espacials de Catalunya (IEEC)

Met Office (MetO)

Ref: SAF/ROM/METO/UG/ROPP/007

Document Author Table

	Name	Function	Date	Comment
Prepared by:	I. Culverwell	ROM SAF Project Team	31 December 2021	
Reviewed by:	O. Lewis	ROM SAF Project Team	31 December 2021	
Approved by:	K. B. Lauritsen	ROM SAF Project Manager	31 December 2021	

Document Change Record

Issue/Revision	Date	By	Description
Version 0.1	01 Nov 2004	CM	Preliminary release for I–RR
Version 0.7	04 Jun 2005	CM	Intermediate release for CPM.
Version 0.8	17 Oct 2005	DO	First beta release (v0.8)
Version 0.9	10 Nov 2006	DO	Second Beta release (v0.9)
Version 1.0	01 Mar 2007	DO	First full release (v1.0)
Version 1.1	01 Mar 2008	DO	Updates to first full release (v1.1)
Version 1.2	01 Jul 2008	HL	Updates to first full release (v1.2)
Version 2.0	01 Dec 2008	HL	Second full release (v2.0)
Version 3.0	01 Jun 2009	HL	Third full release (v3.0)
Version 4.0	01 Nov 2009	HL	Fourth full release (v4.0)
Version 4.1	01 Jun 2010	HL	Updates to fourth full release (v4.1)
Version 5.0	14 Jun 2011	IC	Fifth full release (v5.0)
Version 6.0	31 Oct 2011	IC	Sixth full release (v6.0)
Version 6.1	31 Jan 2013	IC	Updates to sixth full release (v6.1)
Version 7.0	31 Jul 2013	IC	Seventh full release (v7.0)
Version 7.1	31 Dec 2013	IC	Updates to seventh full release (v7.1)
Version 8.0	31 Dec 2014	IC	Eighth full release (v8.0)
Version 8.1	31 Dec 2015	IC	Update to eighth full release (v8.1); split from FM guide
Version 9.0	28 Feb 2017	IC	Ninth full release (v9.0)
Version 9.1	30 Jun 2019	IC	Update to ninth full release (v9.1)
Version 10.0	30 Sep 2020	IC	Tenth full release (v10.0)
Version 11.0	31 Dec 2021	IC	Eleventh full release (v11.0). Changes from ROPP-10.0 in red — see Change Log (SAF/ROM/METO/SRN/ROPP/018) for details.

ROM SAF

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

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

The ROM SAF Leading Entity is the Danish Meteorological Institute (DMI), with Cooperating Entities: i) European Centre for Medium-Range Weather Forecasts (ECMWF) in Reading, United Kingdom, ii) Institut D'Estudis Espacials de Catalunya (IEEC) in Barcelona, Spain, and iii) Met Office in Exeter, United Kingdom. To get access to our products or to read more about the ROM SAF please go to: http://www.romsaf.org.

Intellectual Property Rights

Contents

1 Introduction

1.1 Purpose of this document

This document, the ROPP_1DVAR User Guide ([RD.2f]), describes the 1D–Var module of the Radio Occultation Processing Package (ROPP). The routines in this module are designed to retrieve atmospheric temperature, moisture and pressure from radio occultation bending angle and refractivity profiles. They rely heavily on routines in the ROPP Forward Model module, which are described in ROPP_FM User Guide ([RD.2e]). The current document occasionally refers to equations and figures in the FM User Guide.

1.2 Applicable and reference documents

1.2.1 Applicable documents

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

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

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

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

1.2.2 Reference documents

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

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

[RD.2] The ROPP User Guides:

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

1.3 Acronyms and abbreviations

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

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

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

1.4 Definitions, levels and types

RO data products from the Metop, Metop-SG and Sentinel-6 satellites and RO data from other missions are grouped in *data levels* (Level 0, 1, 2, or 3) and *product types* (NRT, Offline, NTC, CDR, or ICDR). The data levels and product types are defined below[1]. The lists of variables should not be considered as the complete contents of a given data level, and not all data may be contained in a given data level.

Data levels:

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

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

Product types:

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

1.5 Structure of this document

Section 2 briefly describes ROPP and its documentation. Section 3 describes the theoretical basis of variational data assimilation. Section 4 describes the ROPP 1D–Var retrieval tool `ropp_1dvar_refrac`, which adjusts a given background profile so that its forward-modelled refractivities are a closer fit to a given refractivity profile. Backgrounds, errors, quality control, minimisation routines, diagnostics and user-controllable settings are described fully. Naturally, there is a strong connection with the ROPP forward model, whose User Guide ([RD.2e]) should be read alongside this document. Section 5 describes the ROPP 1D–Var retrieval tool `ropp_1dvar_bangle`, which does the same minimisation but against bending angle observations. Section 6 describes how retrievals can be made using the difference between bending angles

[2] (i) GCOS 2016 Implementation Plan; (ii) http://climatemonitoring.info/home/terminology/.
[3] http://climatemonitoring.info/home/terminology (the ICDR definition was endorsed at the 9th session of the joint CEOS/CGMS Working Group Climate Meeting on 29 March 2018 (http://ceos.org/meetings/wgclimate-9)).

at two different frequencies, by means of a multiple 'VaryChap'-layered model ionosphere in a new forward model. Section 7 describes how non-ideal gas effects can be included in the 1D–Var retrieval routines (really, in the forwards models that they call) by means of 'compressibility factors'. Finally, Section 8 describes how retrievals can be made using non-ionospherically corrected L1 and L2 bending angles, by means of a simple model ionosphere in the forward model.

Appendices give brief instructions on how to build ROPP, list the files in the `ropp_1dvar` module, list the 'extra diagnostic data' that is produced by the various ROPP tools (usually by means of a '`-d`' option), record useful ROPP and other ROM SAF documentation, list the principal authors of ROPP, and state the copyright information that applies to various parts of the code.

2 ROPP

2.1 ROPP introduction

The aim of ROPP is

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

ROPP is a collection of software modules (provided as source code), supporting data files and documentation, which aids users wishing to assimilate radio occultation data into their NWP models. It was originally designed to process data from the GRAS instrument on Metop-A and B, but the software should be adaptable enough to handle data from any other GNSS-LEO radio occultation mission.

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

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

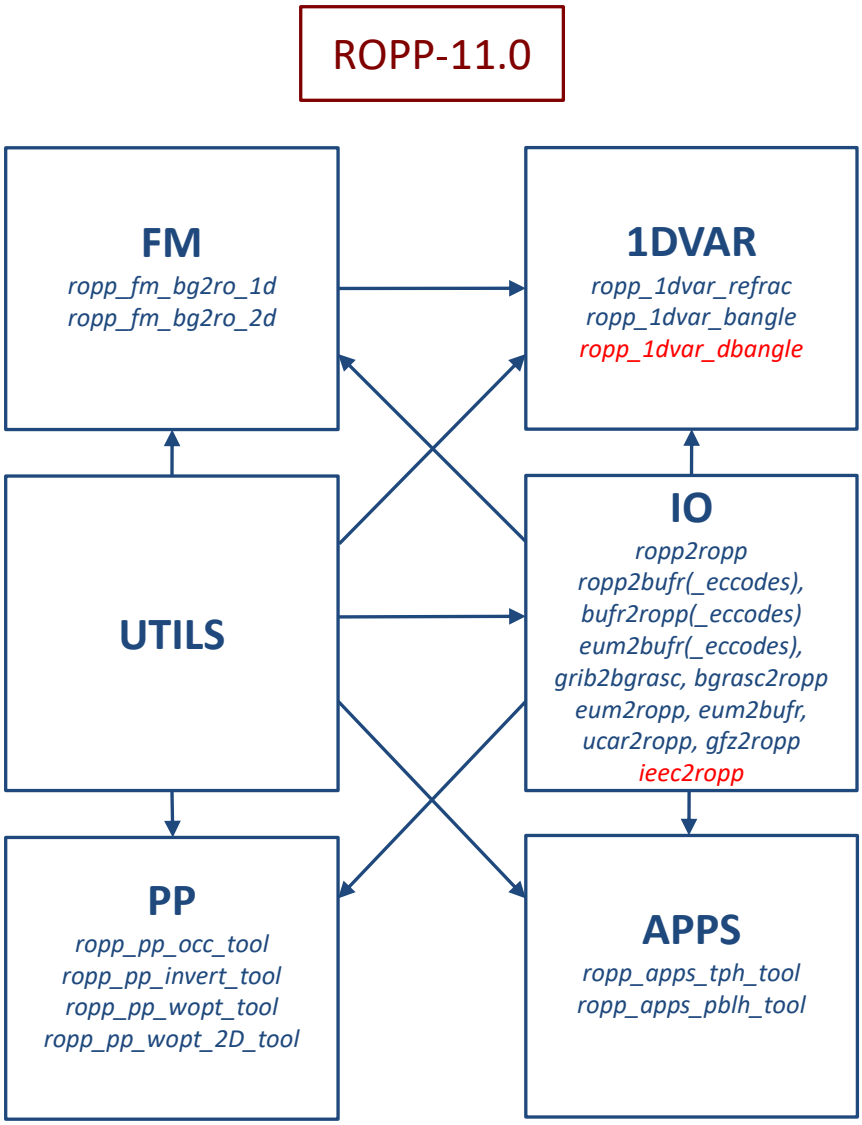

Figure 2.1: The **modules** and *tools* within ROPP-11.0. The module at the head of an arrow depends directly on the module at its tail.

2.2 User documentation

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

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

Module-specific user guides for the utilities (ROM SAF, 2021f), input/output (ROM SAF, 2021d), pre-processor (ROM SAF, 2021e), forward model (ROM SAF, 2021c), 1D–Var (ROM SAF, 2021a) and applications (ROM SAF, 2021b) modules describe the algorithms and routines used in those modules. These provide the necessary background and descriptions of the ROPP software for users to process radio

occultation data from excess phase to bending angle or refractivity, to forward model background fields to refractivity and bending angle profiles, to simulate the propagation of GNSS radio waves through idealised atmospheric refractivity structures, and to perform 1D–Var retrievals of radio occultation data, as well as advice on how to implement ROPP in their own applications.

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

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

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

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

References

ROM SAF, The Radio Occultation Processing Package (ROPP) 1D–Var module User Guide, SAF/ROM/METO/UG/ROPP/007, Version 11.0, 2021a.

ROM SAF, The Radio Occultation Processing Package (ROPP) Applications module User Guide, SAF/ROM/METO/UG/ROPP/005, Version 11.0, 2021b.

ROM SAF, The Radio Occultation Processing Package (ROPP) Forward model module User Guide, SAF/ROM/METO/UG/ROPP/006, Version 11.0, 2021c.

ROM SAF, The Radio Occultation Processing Package (ROPP) Input/Output module User Guide, SAF/ROM/METO/UG/ROPP/002, Version 11.0, 2021d.

ROM SAF, The Radio Occultation Processing Package (ROPP) Pre-processor module User Guide, SAF/ROM/METO/UG/ROPP/004, Version 11.0, 2021e.

ROM SAF, The Radio Occultation Processing Package (ROPP) Utilities module User Guide, SAF/ROM/METO/UG/ROPP/008, Version 11.0, 2021f.

3 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.

3.1 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

$$J(\mathbf{x}) = \frac{1}{2}(\mathbf{x}-\mathbf{x}_b)^T\mathbf{B}^{-1}(\mathbf{x}-\mathbf{x}_b) + \frac{1}{2}(\mathbf{y}_o-\mathbf{H}[\mathbf{x}])^T(\mathbf{E}+\mathbf{F})^{-1}(\mathbf{y}_o-\mathbf{H}[\mathbf{x}]) \tag{3.1}$$

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 3.1. 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-

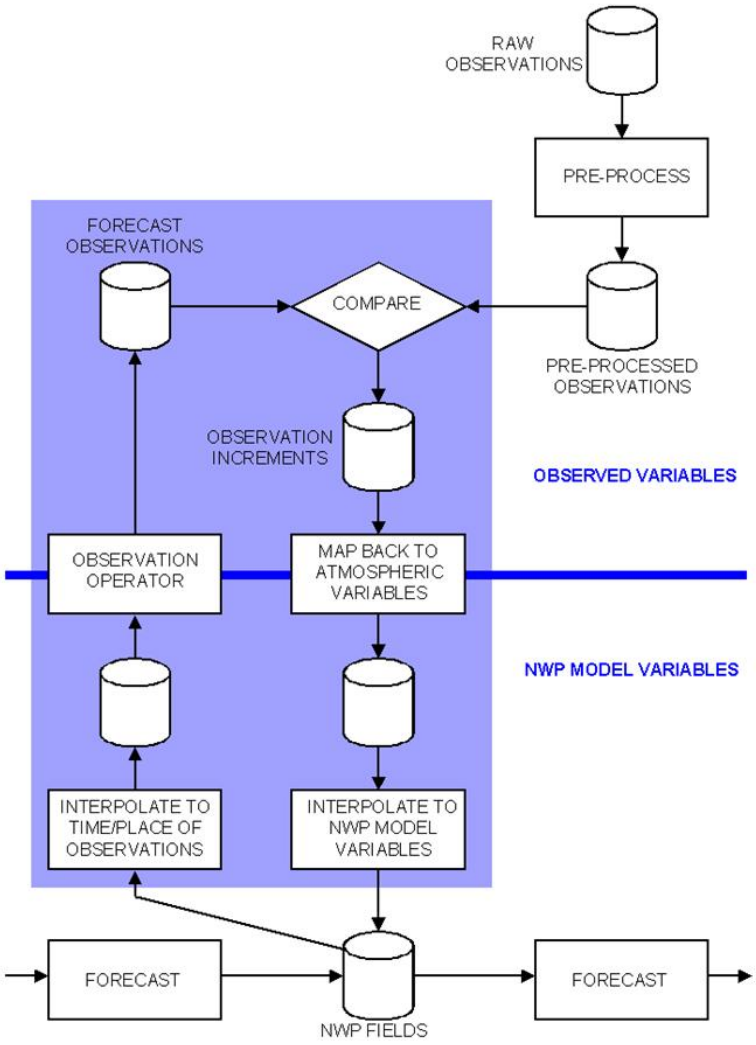

Figure 3.1: Schematic view of a 3D–Var data assimilation system (from Eyre, 1997).

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 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.

3.2 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 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.

With effect from ROPP-11.0, `ropp_fm` also contains a forward model for the difference between the bending angles at two frequencies, which depends only on the electron density in the ionosphere.

3.3 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.

3.4 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.

3.5 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 preprocessing 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 preprocessing, 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 (2021) 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) Pre-processor module User Guide, SAF/ROM/METO/UG/ROPP/004, Version 11.0, 2021.

4 ROPP 1D–Var: Refractivity

Note that this Section is, apart from the references to refractivity rather than bending angle, and some associated minor differences, identical to Sec 5. There is little to be gained from reading both.

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.

$$J(\mathbf{x}) = \frac{1}{2}(\mathbf{x}-\mathbf{x}_b)^T\mathbf{B}^{-1}(\mathbf{x}-\mathbf{x}_b) + \frac{1}{2}(\mathbf{y}_o-\mathbf{H}[\mathbf{x}])^T\mathbf{O}^{-1}(\mathbf{y}_o-\mathbf{H}[\mathbf{x}]) \tag{4.1}$$

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 (see Sec 4.8 of ROPP FM User Guide (2021a)) for a given atmospheric state.

Figure 4.1 shows example pressure, temperature and humidity profiles retrieved from background model data and colocated GNSS-RO refractivity observations.

4.1 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 4.2 shows how the `ropp_1dvar` routines are integrated in the `ropp_1dvar_refrac` code.

4.1.1 Implementation

The `ropp_1dvar_refrac` tool is run using the command

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

where `<outputfile>` is a netCDF file in ROPP format (ROM SAF, 2021b), which will contain the retrieved temperature, humidity and pressure profiles on model background levels.

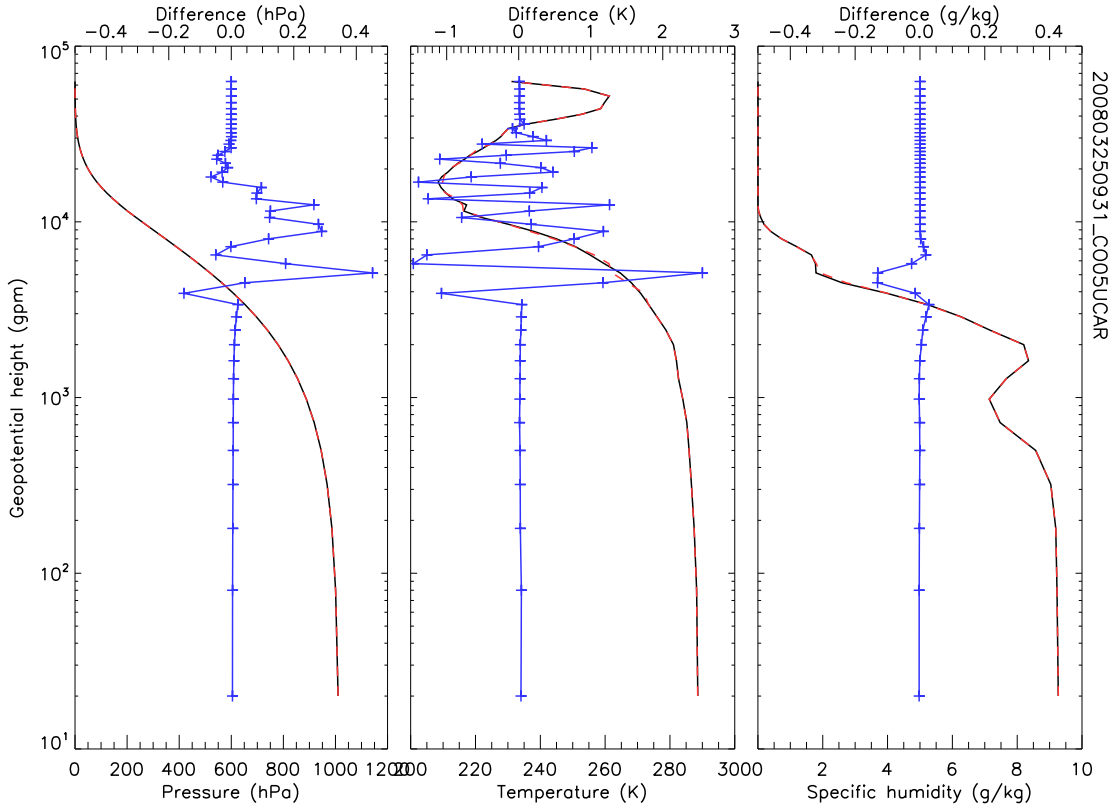

Figure 4.1: Example pressure, temperature and humidity profile retrievals (red) computed using background profiles (plotted in black) and colocated GNSS-RO refractivity observations. The difference between the two profiles are plotted in blue.

The executable has the following options.

`-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
`-new_op`	Use alternative refractivity forward model
`-check_qsat`	Include check against saturation in forward model
`-nocheck_qmin`	Do not include check against dryness 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.

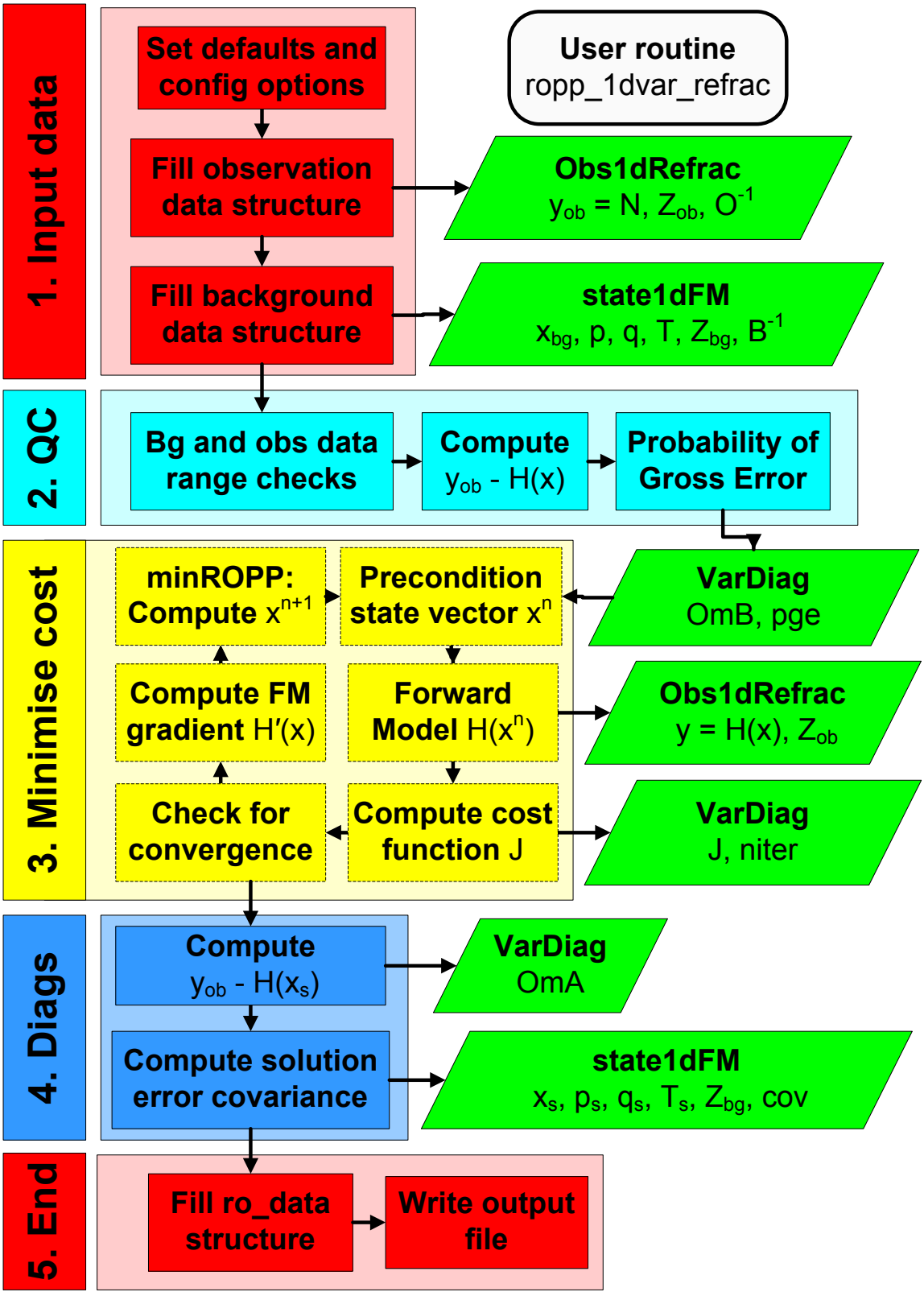

Figure 4.2: Flow chart illustrating the calling tree of the ROPP 1D–Var to retrieve atmospheric profiles from observed refractivity profiles and input background model data.

4.1.2 Code organisation

Figure 4.2 shows how the `ropp_1dvar_refrac` tool is composed of the following stages:

- **Input data access and transformation to a generic state vector** (Section 4.3)

  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 4.6)

  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 4.7)

  `ropp_1dvar_solve` and `ropp_1dvar_levmarq` are the main routines within `ropp_1dvar` for finding the atmospheric state vector $\mathbf{x}$ that minimises the cost function for given backgrounds, observations and their associated errors. The second is used if `config%minROPP%method` equals `'LEVMARQ'`; otherwise (for a 'minROPP' procedure) the first is used. Both undertake the same three stages on each iteration towards a solution:

  - Compute the cost function (Equation 4.1) and its gradient. 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 a minimiser to update the state vector $\mathbf{x}$ towards a solution. In `ropp_1dvar_solve` this is effected by a call to `ropp_1dvar_minropp`; in `ropp_1dvar_levmarq` it is part of the subroutine.
  - 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 4.8)

  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 4.9)

4.2 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 4.4.2 and 4.5.2). 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

$$\text{corr}_{nm} = \exp\left(-|z_n - z_m|/H\right) \tag{4.2}$$

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. It is generally recommended that `ropp_1dvar_refrac` is run with configuration options `obs_covar_method` = 'VSFC' and `bg_covar_method` = 'VSFC'. Background error covariance files generated by the 'Scaled Ensemble Standard Deviation' (SES) method of Holm and Kral (2012) need to be run with `bg_covar_method = RSFC`. (See Secs 4.4.2 and 4.5.2 for the definitions of these terms.)

- **Background error correlation matrices.** These files are suitable arguments to the '–bg-corr' option to `ropp_1dvar_refrac`.
  - **ropp_bg_meto_error_corr_L50.nc** — Background error correlation matrix in packed form for the previous Met Office model structure (50 levels).
  - **ropp_bg_meto_error_corr_L70.nc** — Background error correlation matrix in packed form for the latest Met Office model structure (70 levels).
  - **ropp_bg_ecmwf_error_corr_L60.nc** — Background error correlation matrix in packed form for the previous ECMWF model structure (60 levels).
  - **ropp_bg_ecmwf_error_corr_L91.nc** — Background error correlation matrix in packed form for the previous ECMWF model structure (91 levels).
  - **ropp_bg_ses_ecmwf_error_corr_L60.nc** — Background error correlation matrix in packed form for 60L ECMWF model, generated by the SES method.
  - **ropp_bg_ses_ecmwf_error_corr_L91.nc** — Background error correlation matrix in packed form for 91L ECMWF model, generated by the SES method.
  - **ropp_bg_ses_ecmwf_error_corr_L137.nc** — Background error correlation matrix in packed form for 137L ECMWF model, generated by the SES method.
- **Observation error correlation matrices.** These files are suitable arguments to the '–ob-corr' option to `ropp_1dvar_refrac`.
  - **ropp_ob_refrac_error_corr_247L.nc** — Refractivity observation correlation matrix in packed form. For use with 'standard' 247 geopotential levels generated by using the '-247L' option to `ropp_fm_bg2ro_1d`.

4.3 Input data

Figure 4.3 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.

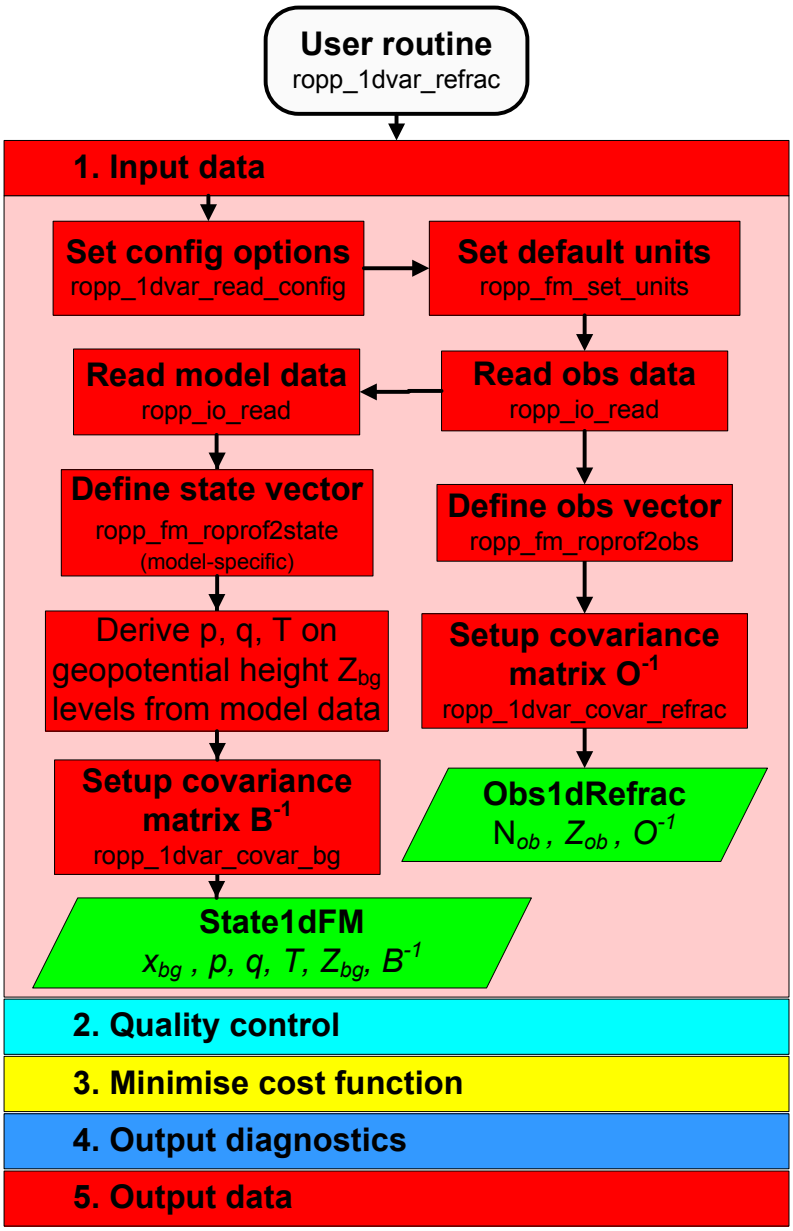

Figure 4.3: Flow chart illustrating the calling tree of the input data step of ROPP 1D–Var to retrieve atmospheric profiles from observed refractivity profiles and input background model data.

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 Sec 4.4 of the ROPP FM User Guide). If required for 1D–Var, users must ensure that the error correlations used are appropriate for background and observation data in ascending height order.

4.3.1 Configuration options

A number of configuration options can be defined by the user in order to tune the performance of the ropp_1dvar retrieval. Tables 4.1 and 4.2 list 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.

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. allowed obs height
...%max_1dvar_height	60 km	Max. allowed obs height
...%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_obheight	20 km	Min. required obs height
...%genqc_{min,max}_temperature	150, 350 K	Min./Max. bg data values
...%genqc_{min,max}_spec_humidity	0, 50 g/kg	
...%genqc_{min,max}_impact	6.2e6, 6.6e6 m	Min./Max. obs data values
...%genqc_{min,max}_bangle	-1.0e-4, 0.1 rad	
...%genqc_{min,max}_geop_refrac	-1.e3, 1.e5 m	
...%genqc_{min,max}_refractivity	0.0, 500 N-units	
...%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	Width of gross error plateau
...%minROPP%method	LEVMARQ	Minimisation method
...%minROPP%log_file	screen	minROPP output device
...%minROPP%impres	0	minROPP output mode
...%minROPP%n_iter	1500	minROPP storage parameter
...%minROPP%n_updates	50	Maximum number of iterations
...%minROPP%eps_grad	1.0e-8	minROPP convergence parameter
...%minROPP%dx_min	1.0e-16	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

Table 4.1: Configuration options held as elements of the VarConfig structure which are used by ropp_1dvar routines. The default values are assumed unless overwritten by configuration options that are read from an input configuration file.

VarConfig		
Structure element	Default	Description
...%j_s_limit	5.0	Maximum value of 2J/m in QC
...%n_iter_limit	50	Maximum number of iterations in QC
...%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
...%compress	.FALSE.	Use non-ideal compressibility factors
...%season_amp	0.0	Amplitude of seasonal ob error scaling
...%season_offset	0.0	Constant offset for seasonal scaling
...%season_phase	0.0	Phase of seasonal scaling factor
...%nlayer	1	Number of VaryChapman layers
...%f1	1.57542E9	First frequency (Hz)
...%f2	1.22760E9	Second frequency (Hz)
...%r_gns	2.67E7	Nominal GNSS to CoC distance (m)
...%r_leo	7.19E6	Nominal LEO to CoC distance (m)
...%genqc_min_ne_peak	0.0	Minimum ne_peak (m-3)
...%genqc_max_ne_peak	1.0E15	Maximum ne_peak (m-3)
...%genqc_min_r_peak	6.2E6	Minimum r_peak (m)
...%genqc_max_r_peak	7.4E6	Maximum r_peak (m)
...%genqc_min_h_zero	0.0	Minimum h_zero (m)
...%genqc_max_h_zero	1.0E6	Maximum h_zero (m)
...%genqc_min_h_grad	0.0	Minimum h_grad
...%genqc_max_h_grad	1.0	Maximum h_grad

Table 4.2: Configuration options held as elements of the VarConfig structure (continued).

4.4 Observation data

For refractivities, the routines `ropp_1dvar` and `ropp_fm` use observation data defined as elements of a structure of type `Obs1dRefrac`. (See Sec 4.2.1 of ROPP FM UG.)

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, 2021b).

```
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)
```

4.4.1 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`. (See Sec 4.7 of ROPP FM UG.)

4.4.2 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,

$$\mathbf{O} = \sigma.\mathbf{Corr}.\sigma^T \tag{4.3}$$

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`.

ROPP has an option to vary the standard deviations σ according to the time of the year — see Sec 4.4.3 for details.

Observation error covariance options

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 4.1).

- **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 4.2).
- **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 4.2).
- **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 4.2).

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.

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

4.4.3 Seasonal scaling of observation errors

Configuration options exist to apply a seasonal dependence to the observation errors *before* they 'sandwich' the correlations in Eqn 4.3. The input error values are scaled according to the time of year, based on a sinusoidal function that takes three parameters, labelled here as Δ, A and ϕ but appearing in Table 4.1 as `season_offset`, `season_amp` and `season_phase` respectively. Here, the time t is the fraction of the way through the year, i.e. between 0 (Jan. 1) and 1 (Dec. 31).

$$\sigma_{scaled} = \sigma_{orig}\left[1+\Delta+A\cos\left(2\pi(t+\phi)\right)\right] \tag{4.4}$$

where Δ is a constant offset, on which the seasonal variation is applied, A is the amplitude of the sinusoidal scaling factor and ϕ is the 'phase shift' of the sinusoid, i.e. a value of 0.1 will shift the maximum of the sinusoid back one tenth of a year.

This option should be used with care to ensure that realistic values are produced. Recommended usage is to take the read-in error values as the minimum errors for the annual cycle and set $\Delta = A > 0$ to ensure that the scaling will only produce increased sigma values.

A full specification of the seasonal dependence of observation errors would require additional dependence on latitude and height (Scherllin-Pirscher et al. (2011)), but the functionality provided here should provide a starting point for further developments.

Note that this option offers the user a simple way to rescale the observation errors without having to regenerate input files.

4.5 Background data

4.5.1 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` (see Sec 4.4 of the ROPP FM User Guide (2021a)).

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 Secs 4.5 and 4.6 of ROPP FM UG 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.

4.5.2 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,

$$\mathbf{B} = \sigma.\mathbf{Corr}.\sigma^T \tag{4.5}$$

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.

Background error covariance options

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 4.1).

- **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 4.2).
- **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 4.2).

- **RSFC — Relative Sigmas, Fixed Correlations**

  Relative specific humidity (q) errors and relative surface pressure ($p*$) errors, and (absolute) temperature (T) errors, are read from a background error correlation file, as are all the error correlations. (All fields must come from the same file.) The relative q ($p*$) errors are multiplied by the profile q ($p*$) values to give the profile errors. (As before, error values are automatically adjusted if the user sets either `config%use_logp` or `config%use_logq`.) In this case the error correlation file must contain `temp_sigma` (length n), `shum_rel_sigma` (length n), `press_sfc_rel_sigma` (length 1) and `corr` (length $2n+1$). Further details are available in (ROM SAF, 2014). Example files are included in the ROPP distribution — see Section 4.2. RSFC is therefore a hybrid of FSFC and VSFC. *This method can only be applied to background fields on ECMWF levels.*

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. (See Secs 4.5 and 4.6 of ROPP UG.)

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 = 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 = 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 $j =$ 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 $j =$ bg%n_lev $+ i$

4.6 Quality control

Figure 4.4 illustrates the implementation of `ropp_1dvar` module routines to perform preliminary quality control on the input data.

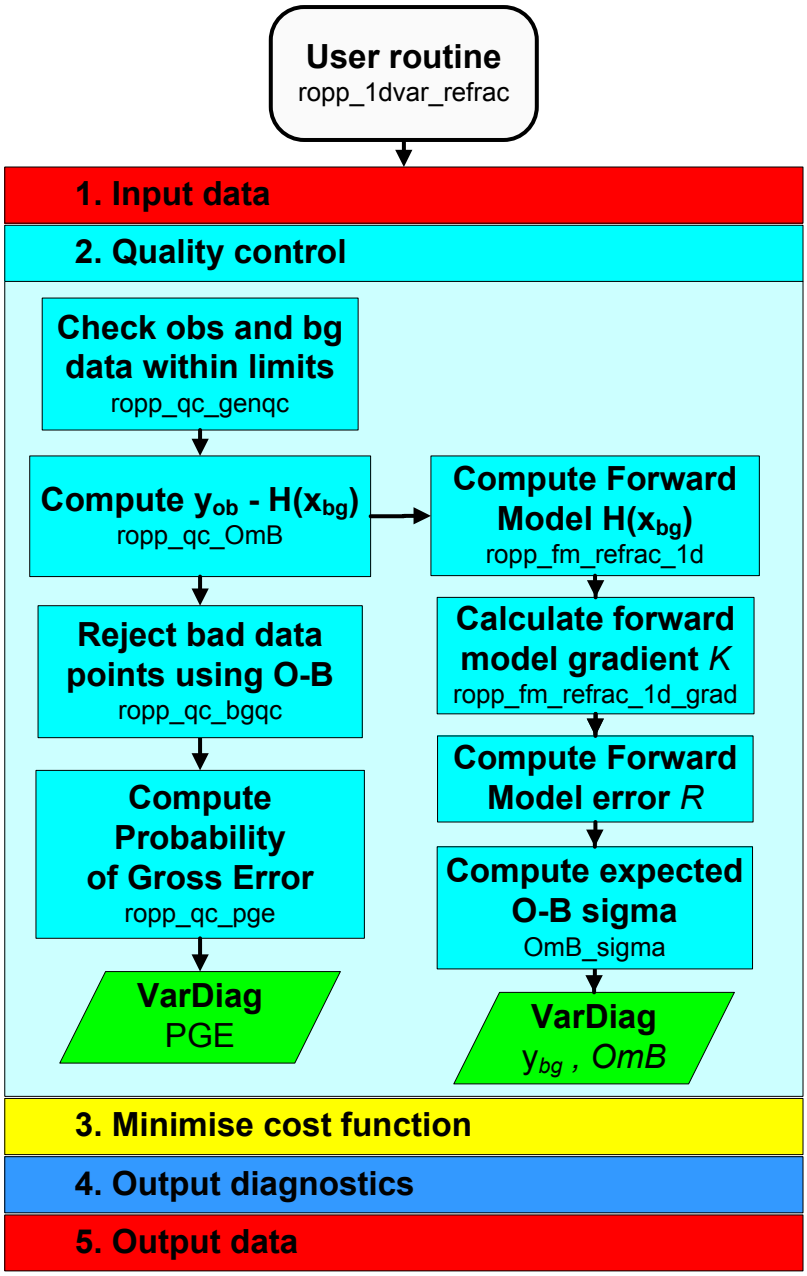

Figure 4.4: Flow chart illustrating the calling tree of the quality control step of ROPP 1D–Var to retrieve atmospheric profiles from observed refractivity profiles and input background model data.

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, bg_data%bg%fcperiod)
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)
```

4.6.1 Valid observation height range: `ropp_qc_cutoff()`

Configuration options `config%min_1dvar_height` and `config%max_1dvar_height` (Table 4.1) 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.

4.6.2 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 4.1). 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`.

4.6.3 Observation minus background check: `ropp_qc_OmB()`

The difference between the observations and the background data forward modelled into observation space (`diag%OmB` $= \mathbf{y}_{ob} - \mathbf{H}[\mathbf{x}_{bg}]$) and the standard deviation of this difference (`diag%OmB_sigma`) 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 observations (see Sec 4.8 of ROPP FM UG).

The error in the observation minus background calculation is given by

$$\texttt{diag\%OmB\_sigma(:)} = (\mathbf{O} + \mathbf{K}.\mathbf{B}.\mathbf{K}^T)^{1/2} \tag{4.6}$$

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`.

4.6.4 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.

4.6.5 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 at each observation point is computed based on the $(\mathbf{y}_{ob} - \mathbf{H}(\mathbf{x}_{bg}))$ 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`.

$$\texttt{diag\%pge(i)} = \left[1 + \gamma^{-1} \exp(-u_i^2/2)\right]^{-1} \tag{4.7}$$

where

$$\mathbf{u} = \frac{\mathbf{y}_{ob} - \mathbf{H}(\mathbf{x}_{bg})}{\sigma(\mathbf{y}_{ob} - \mathbf{H}(\mathbf{x}_{bg}))}. \tag{4.8}$$

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

$$\gamma = \frac{A\sqrt{2\pi}}{(1-A)2d} \tag{4.9}$$

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

4.7 Minimise the cost function

Figure 4.5 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 ($\mathbf{x}$) that minimises the cost function (Equation 4.1) is either `ropp_1dvar_solve`, if `config%minROPP%method = MINROPP`, or `ropp_1dvar_levmarq` otherwise. 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 $\mathbf{x}_{bg}$. On exit $\mathbf{x}$ contains the 1D–Var solution. Both solvers take the same input variables and produce the same output variables (with different *values*, of course, although these should be very similar if both solvers have converged successfully.)

```
USE ropp_fm
USE ropp_1dvar
TYPE(State1dFM)   :: bg
TYPE(State1dFM)   :: state
TYPE(Obs1dRefrac) :: obs
...
state = bg    ! initial guess - set state equal to background state
IF (config%minROPP%method == 'MINROPP') THEN
  CALL ropp_1dvar_solve(obs, bg, state, config, diag)
ELSE
  CALL ropp_1dvar_levmarq(obs, bg, state, config, diag)
ENDIF
```

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. By default, the state vector is updated using a Levenberg-Marquardt minimisation algorithm (Press et al., 1992). See Sec 4.7.2 and ROM SAF (2008) for details. Alternatively, if the parameter `config%minROPP%method` equals `MINROPP` then the ROPP-specific minimiser minROPP will be used to minimise the cost function. See Sec 4.7.1 and ROM SAF (2007) for details.

4.7.1 Minimisation with the minROPP scheme

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 transform the initial state variable $\mathbf{x}$ to a control variable $\mathbf{c}$, given a preconditioner $\mathbf{L}$ defined such that $\mathbf{B} = \mathbf{L}\mathbf{L}^T$, thus:

$$\mathbf{c} = \mathbf{L}^{-1}\mathbf{x} \tag{4.10}$$

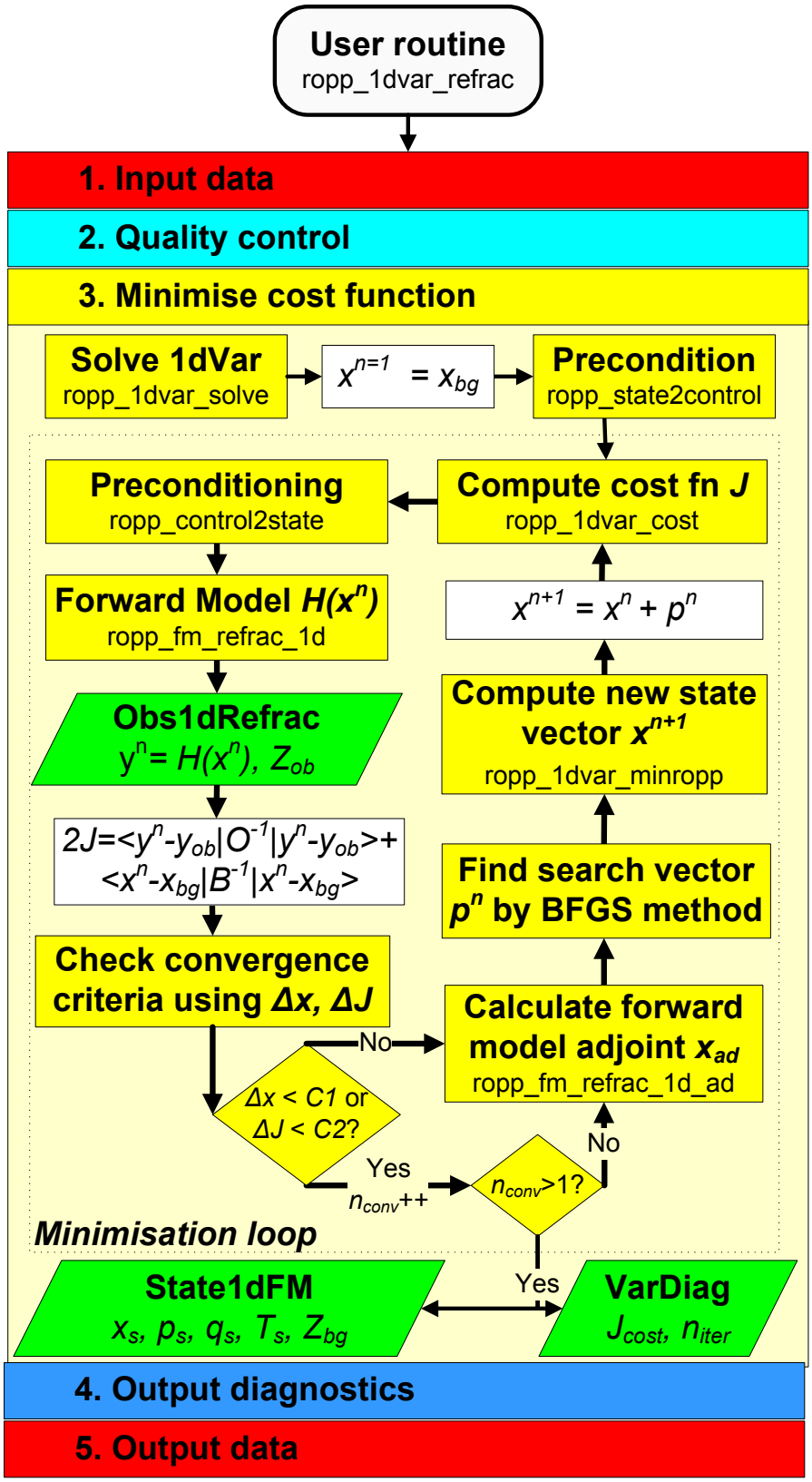

Figure 4.5: Flow chart illustrating the calling tree of the minROPP minimisation step of ROPP 1D–Var to retrieve atmospheric profiles from observed refractivity profiles and input background model data.

No other elements of the State1dFM structure are affected 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 $\mathbf{c}$ by calling the routine ropp_control2state. If config%use_precond is not set, $\mathbf{c}$ is set equal to $\mathbf{x}$.

Compute the cost function

Equation (4.1) 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 $\mathbf{y}_{ob} - \mathbf{H}[\mathbf{x}]$ with the weighting factors determined from the quality control processing.

The matrix multiplication of the differences $(\mathbf{x} - \mathbf{x}_b)$ and $(\mathbf{y}_{ob} - \mathbf{H}[\mathbf{x}])$ 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:

$$\nabla J(\mathbf{x}) = \mathbf{B}^{-1}(\mathbf{x} - \mathbf{x}_b) - (\mathbf{H}'[\mathbf{x}])^T \mathbf{O}^{-1}(\mathbf{y}_{ob} - \mathbf{H}[\mathbf{x}]) \tag{4.11}$$

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. On exit, ropp_1dvar_cost returns the variable J_grad, the cost function gradient with respect to the control variable $\mathbf{c}$, which is calculated by premultiplying $\nabla J(\mathbf{x})$ in Eqn 4.11 by $\mathbf{L}^T$ if preconditioning is being applied.

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 the following two criteria needs to be met on `config%conv_check_n_previous` consecutive iterations for convergence to be achieved.

1. Maximum relative change in state

The state vector has not changed by more than a set value between iterations:

$$\max\left[\frac{|\mathbf{x}_n - \mathbf{x}_{n-1}|}{\sqrt{\mathbf{B}}}\right] < \texttt{config\%conv\_check\_max\_delta\_state}, \tag{4.12}$$

where $\sqrt{\mathbf{B}}$ is a vector comprising the square roots of the diagonal elements of the background covariance matrix $\mathbf{B}$.

2. Maximum change in cost function

The cost function has not changed by more than a set value between iterations:

$$|J_n - J_{n-1}| < \texttt{config\%conv\_check\_max\_delta\_J}. \tag{4.13}$$

When either of the convergence criteria is 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` (Sec 4.5 of ROPP FM UG) or `ropp_fm_state2state_meto` (see Sec 4.6 of ROPP FM UG). Diagnostic parameters `diag%J`, `diag%J_scaled` and `diag%n_iter` are set.

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 from `ropp_1dvar_solve` thus:

```
CALL ropp_1dvar_minropp(control%state, J_grad, J_dir, dJ, gconv, &
                        n_iter, m_indic, config%minropp%n_updates, &
                        config%minropp%n_iter)
```

where control%state 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. n_iter is an iteration counter, which is incremented each time minROPP updates the state vector. m_indic is a convergence indication flag, config%minropp%n_updates is the maximum number of iterations that are allowed, and config%minropp%n_iter defines the size of some workspace.

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.

$$||\nabla J_k|| < \text{eps}||\nabla J_1|| \tag{4.14}$$

Parameter eps equals the configuration element config%minROPP%eps_grad, and the right hand side of the above equation is the variable gconv, which is input to ropp_1dvar_minropp. The routine exits if this condition is met, and the minimisation loop is terminated.

Figure 4.5 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_updates iterations have been completed, then no convergence can be achieved and the 1D–Var retrieval fails.

4.7.2 Minimisation with the Levenberg-Marquardt scheme

The theory behind the Levenberg-Marquardt minimisation scheme is clearly explained in Press et al. (1992). Loosely speaking, it amounts to an inverse Hessian (or 'Newton-Raphson') increment of the state variable, unless this causes the cost function to increase, in which case the minimisation procedure becomes more like a steepest descent algorithm. This trend towards steepest descent continues, ultimately with a small step size, until the cost function starts to decrease, at which point the algorithm begins to recover its inverse Hessian character.

Figure 4.6 illustrates the implementation of ropp_1dvar module routines to minimise the cost function using the Levenberg-Marquardt method and obtain the solution state vector for a given 1D–Var refractivity retrieval. The user might find it useful to compare it to the minROPP equivalent, Figure 4.5.

Preconditioning

Unlike the minROPP minimiser, no preconditioning is applied in the Levenberg-Marquardt scheme used in ROPP.

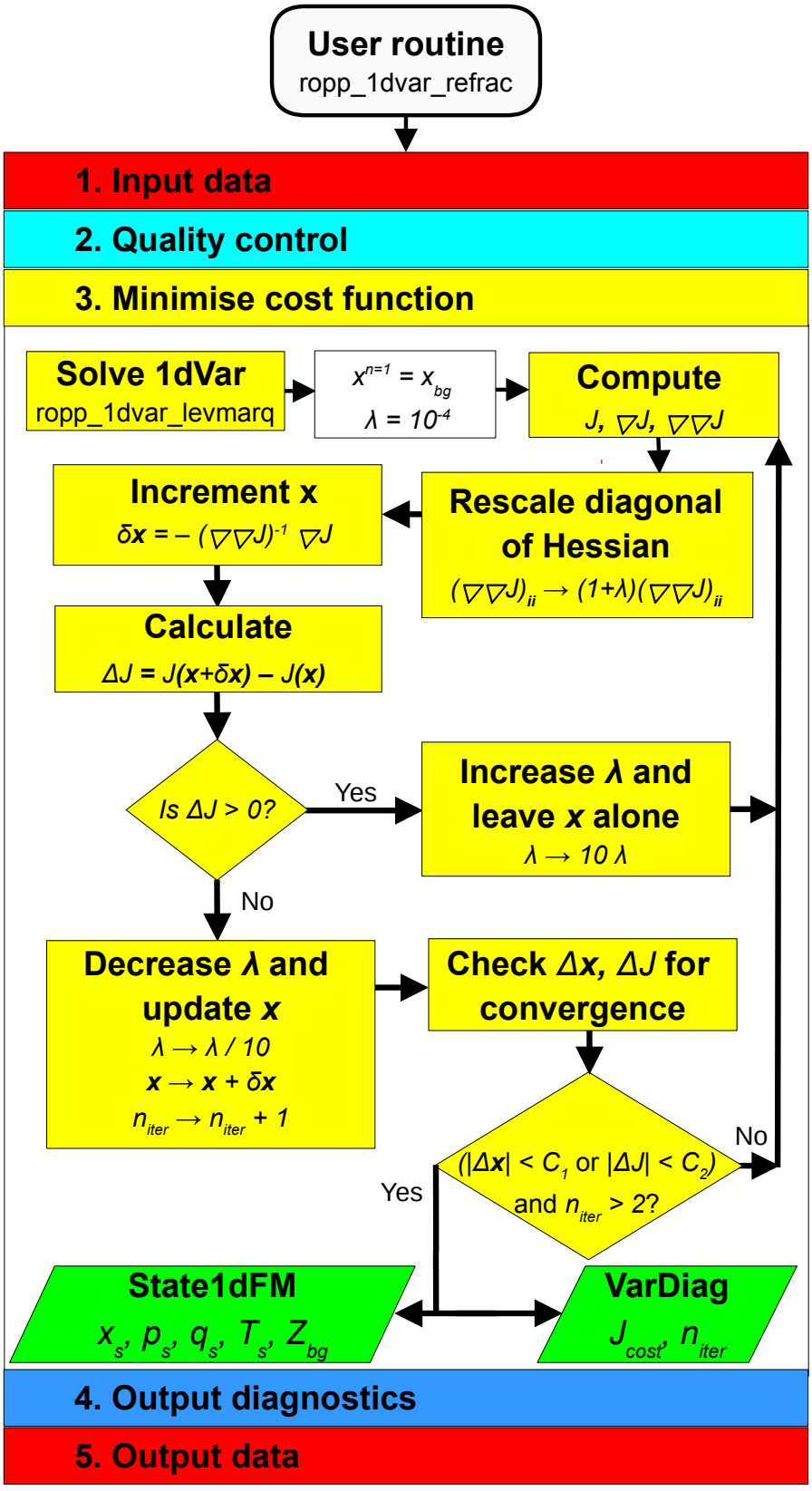

Figure 4.6: Flow chart illustrating the calling tree of the LevMarq minimisation step of ROPP 1D–Var to retrieve atmospheric profiles from observed refractivity profiles and input background model data.

Computation of cost function and its first two derivatives

The cost function J and its first derivative $\nabla J(\mathbf{x})$ are computed directly from Eqns 4.1 and 4.11. ($\mathbf{H}[\mathbf{x}]$ is calculated by means of a call to `ropp_fm_refrac_1d`, and $\mathbf{H}'[\mathbf{x}]$ is calculated by means of a call to `ropp_fm_refrac_1d_grad`.) The second derivative, or Hessian, $\nabla\nabla J(\mathbf{x})$, is calculated from

$$\nabla\nabla J(\mathbf{x}) = \mathbf{B}^{-1} + (\mathbf{H}'[\mathbf{x}])^T \mathbf{O}^{-1} \mathbf{H}'[\mathbf{x}] \tag{4.15}$$

The initial value of J is stored in the diagnostic variable `diag%J_init`.

Minimisation

The *diagonal* elements of $\nabla\nabla J(\mathbf{x})$ are multiplied by $(1+\lambda)$, where the dimensionless number λ, the key feature of the Levenberg-Marquardt scheme, controls the relative sizes of the steepest descent and the inverse Hessian characteristics of the minimisation algorithm. Since λ is initialised to 10^{-4}, this multiplication has little effect on the first iteration, which is almost completely inverse Hessian. This means that the increment to the state vector $\mathbf{x}$ is given by

$$\delta\mathbf{x} = -(\nabla\nabla J(\mathbf{x}))^{-1} \nabla J(\mathbf{x}), \tag{4.16}$$

which follows (approximately) from making Eqn 4.11 stationary with respect to small changes in $\mathbf{x}$.

As λ gets larger, the matrix $\nabla\nabla J(\mathbf{x})$ becomes increasingly dominated by its diagonal, so that the increments given by Eqn 4.16 become increasingly parallel to the direction of steepest descent, $-\nabla J(\mathbf{x})$, suitably rescaled, for dimensional propriety, by the covariances that constitute $\nabla\nabla J(\mathbf{x})$ (see Eqn 4.15).

The state vector $\mathbf{x}$ is stored before it is incremented by $\delta\mathbf{x}$: $\mathbf{x} \mapsto \mathbf{x} + \delta\mathbf{x}$.

The cost function is recalculated. If it has increased by more than `config%conv_check_max_delta_J`, then $\mathbf{x}$ reverts to the stored value, λ is multiplied by 10 (making it more of a steepest descents algorithm), and the iteration is repeated. If this procedure results in λ exceeding 10^{10} then, because the scheme is evidently not converging, a message is issued and the algorithm stops.

If, on the other hand, the cost function has not increased by at least `config%conv_check_max_delta_J`, then the updated value of $\mathbf{x}$ is used, and λ is divided by 10 (making it more of an inverse Hessian algorithm). The minimisation undergoes another iteration unless the algorithm is considered to have converged.

Convergence checks

If the absolute value of the change in the cost function J is less than `config%conv_check_max_delta_J` (see Eqn 4.13), and this has been the case for the last `config%conv_check_n_previous` iterations, then the algorithm is deemed to have converged, and the minimisation stops.

Similarly, if the maximum value of the magnitude of the change in state $\delta\mathbf{x}$ divided by $\sqrt{\mathbf{B}}$ is less than `config%conv_check_max_delta_state` (see Eqn 4.12), and this has been the case for the last

`config%conv_check_n_previous` iterations, then the algorithm is deemed to have converged, and the minimisation stops.

Finally, if λ is found to be greater than 10^{10} (as a result of successive increases in the cost function), then a warning message is issued and the minimisation stops.

Before leaving subroutine `ropp_1dvar_levmarq_refrac`, the minimum cost function J is copied to the diagnostic variable `diag%J`, the scaled cost function $2J/m$ (where m is the number of non-zero weighted observations) is copied to `diag%J_scaled`, and the level-by-level state vector 'half' of the final cost function, $(1/2)(\mathbf{x}-\mathbf{b})_i \left(\mathbf{B}^{-1}(\mathbf{x}-\mathbf{b})\right)_i$ (no summation over i), is copied to `diag%J_bgr`.

4.8 Output diagnostics

Figure 4.7 illustrates the implementation of `ropp_1dvar` and `ropp_fm` module routines to compute final output diagnostics associated with a 1D–Var retrieval solution. Table 4.3 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).

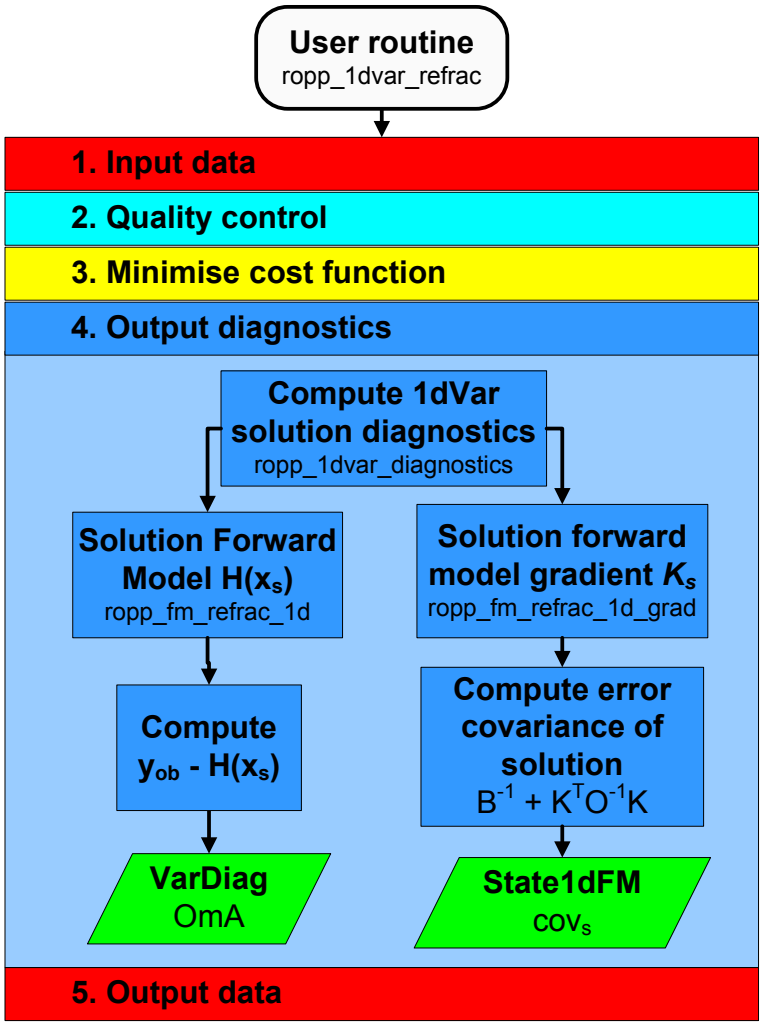

Figure 4.7: Flow chart illustrating the calling tree of the output diagnostics step of ROPP 1D–Var to retrieve atmospheric profiles from observed refractivity profiles and input background model data.

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
```

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
...%bg_ne	Background electron density
...%bg_ne_sigma	Error in background electron density
...%res_ne	Analysis electron density
...%res_ne_sigma	Error in analysis electron density
...%VTEC_bg	VTEC of background electron density
...%VTEC_an	VTEC of analysis electron density

Table 4.3: Elements of VarDiag structure output using the extended_1dvar_diag configuration flag.

```
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 (4.6.3). The diagnostic variable diag%OmA is computed as the difference between the observations ($\mathbf{y}_{ob}$) 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 (Chap. 5 Rodgers, 2000)

$$\mathrm{x\%cov} = \left(\mathbf{B}^{-1} + \mathbf{K}^T\mathbf{O}^{-1}\mathbf{K}\right)^{-1} \tag{4.17}$$

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`.

4.9 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, 2021b).

```
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 (see Sec 4.2.2 of ROPP FM UG).

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.

4.10 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.

Holm, E. V. and Kral, T., Flow-dependent, geographically varying background error covariances for 1d-var applications in MTG-IRS l2 processing, Tech memo 680, ECMWF, http://old.ecmwf.int/publications/library/do/references/list/14, 2012.

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, Algorithm Theoretical Baseline Document: NRT and Offline 1D–Var products, SAF/ROM/DMI/ALG/1DV/002, Version 2.4, 2014.

ROM SAF, The Radio Occultation Processing Package (ROPP) Forward model module User Guide, SAF/ROM/METO/UG/ROPP/006, Version 11.0, 2021a.

ROM SAF, The Radio Occultation Processing Package (ROPP) Input/Output module User Guide, SAF/ROM/METO/UG/ROPP/002, Version 11.0, 2021b.

Scherllin-Pirscher, B., Steiner, A. K., Kirchengast, G., Kuo, Y.-H., and Foelsche, U., Empirical analysis and modeling of errors of atmospheric profiles from GPS radio occultation, *Atmospheric Measurement Techniques*, *4*, 1875–1890, 2011.

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.

5 ROPP 1D–Var: Bending angle

Note that this Section is, apart from the references to bending angle rather than refractivity, and some associated minor differences, identical to Sec 4. There is little to be gained from reading both.

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.

$$J(\mathbf{x}) = \frac{1}{2}(\mathbf{x}-\mathbf{x}_b)^T\mathbf{B}^{-1}(\mathbf{x}-\mathbf{x}_b) + \frac{1}{2}(\mathbf{y}_o-\mathbf{H}[\mathbf{x}])^T\mathbf{O}^{-1}(\mathbf{y}_o-\mathbf{H}[\mathbf{x}]) \tag{5.1}$$

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 (see Sec 4.10 of ROPP FM User Guide (2021a)) for a given atmospheric state.

Figure 5.1 shows example pressure, temperature and humidity profiles retrieved from background model data and colocated GNSS-RO bending angle observations.

5.1 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 5.2 shows how the `ropp_1dvar` routines are integrated in the `ropp_1dvar_bangle` code.

5.1.1 Implementation

The `ropp_1dvar_bangle` tool is run using the command

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

where `<outputfile>` is a netCDF file in ROPP format (ROM SAF, 2021b), which will contain the retrieved temperature, humidity and pressure profiles on model background levels.

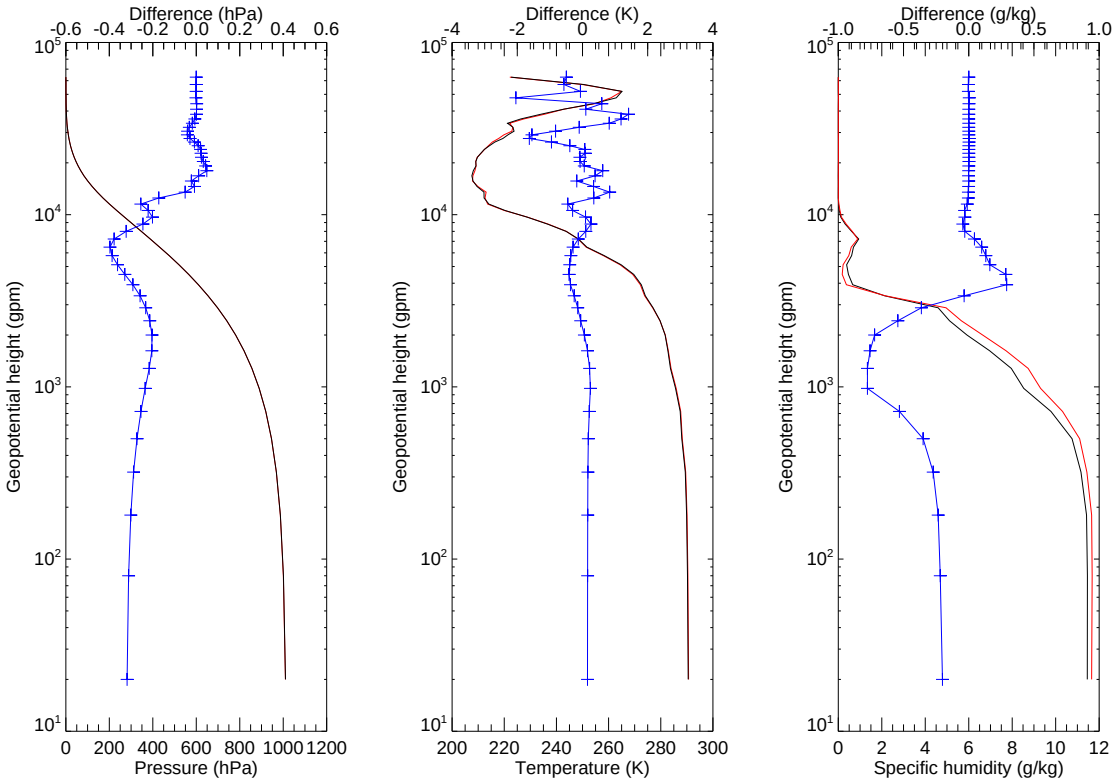

Figure 5.1: Example pressure, temperature and humidity profile retrievals (red) computed using background profiles (plotted in black) and colocated GNSS-RO bending angle observations. The difference between the two profiles are plotted in blue.

The executable has the following options.

`-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
`-nocheck_qmin`	Do not include check against dryness in forward model
`-new_op`	Use new refractivity interpolation between model levels
`-direct_ion`	Use L1 and L2 bending angles
`-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.

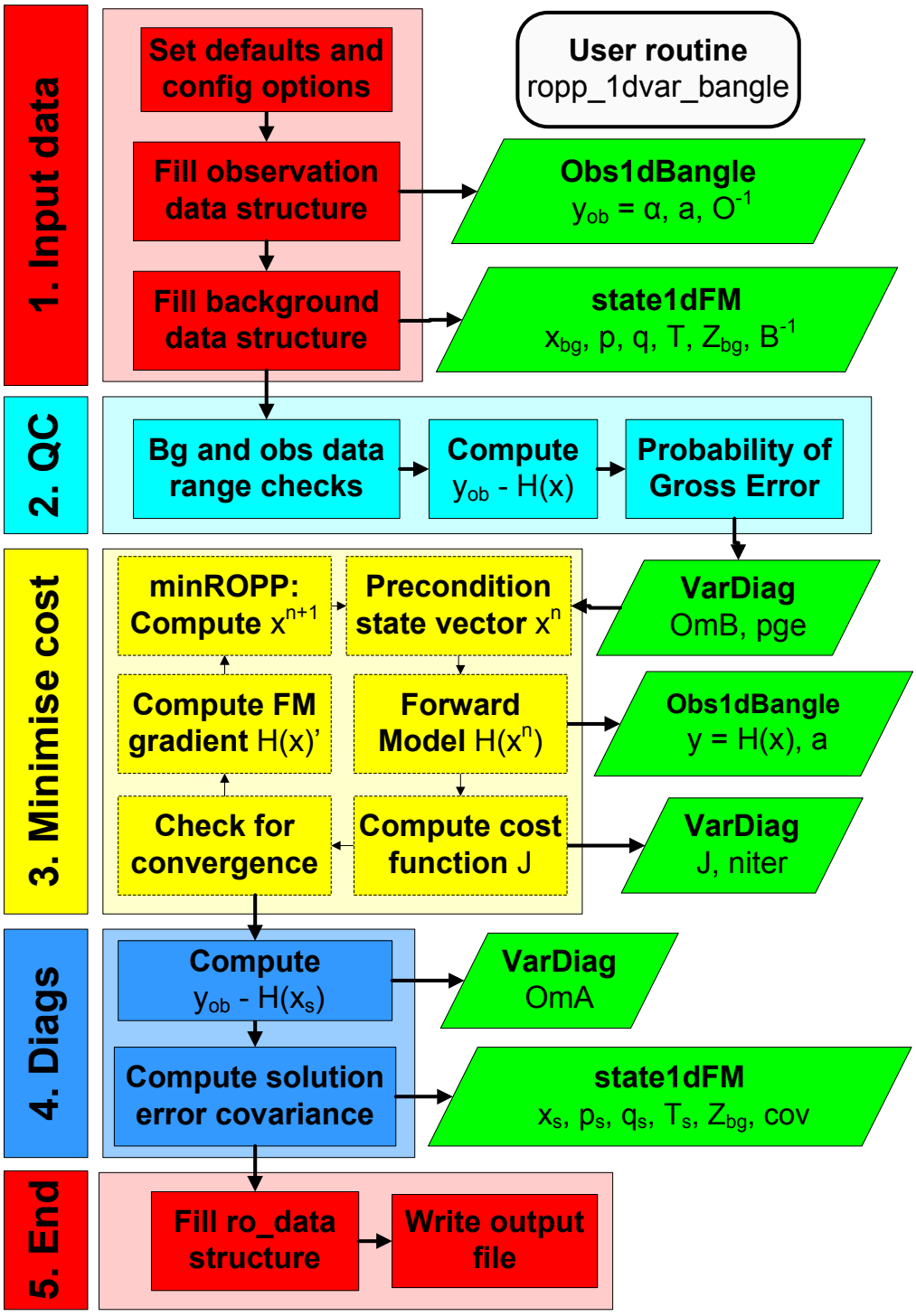

Figure 5.2: Flow chart illustrating the calling tree of the ROPP 1D–Var to retrieve atmospheric profiles from observed bending angle profiles and input background model data.

5.1.2 Code organisation

Figure 5.2 shows how the `ropp_1dvar_bangle` tool is composed of the following stages:

- **Input data access and transformation to a generic state vector** (Section 5.3)

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 5.6)

  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 5.7)

  `ropp_1dvar_solve` and `ropp_1dvar_levmarq` are the main routines within `ropp_1dvar` for finding the atmospheric state vector $\mathbf{x}$ that minimises the cost function for given backgrounds, observations and their associated errors. The second is used if `config%minROPP%method` equals `'LEVMARQ'`; otherwise (for a 'minROPP' procedure) the first is used. Both undertake the same three stages on each iteration towards a solution:

  - Compute the cost function (Equation 4.1) and its gradient. 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 a minimiser to update the state vector $\mathbf{x}$ towards a solution. In `ropp_1dvar_solve` this is effected by a call to `ropp_1dvar_minropp`; in `ropp_1dvar_levmarq` it is part of the subroutine.
  - 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 5.8)

  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 5.9)

5.2 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 5.4.2 and 5.5.2). 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. It is generally recommended that `ropp_1dvar_bangle` is run with configuration options `obs_covar_method` = 'VSDC' (because bending angle errors are usually assumed to be uncorrelated) and `bg_covar_method` = 'VSFC'. If `obs_covar_method` = 'VSDC' then no observation correlation matrix need be supplied. (Those in the example files are in fact just the identity matrices, for

completeness.) Background error covariance files generated by the 'Scaled Ensemble Standard Deviation' (SES) method of Holm and Kral (2012) need to be run with `bg_covar_method = RSFC`. (See Secs 5.4.2 and 5.5.2 for the definitions of these terms.)

- **Background error correlation matrices.** These files are suitable arguments to the '–bg-corr' option to `ropp_1dvar_bangle`.
  - **ropp_bg_meto_error_corr_L50.nc** — Background error correlation matrix in packed form for the previous Met Office model structure (50 levels).
  - **ropp_bg_meto_error_corr_L70.nc** — Background error correlation matrix in packed form for the latest Met Office model structure (70 levels).
  - **ropp_bg_ecmwf_error_corr_L60.nc** — Background error correlation matrix in packed form for the previous ECMWF model structure (60 levels).
  - **ropp_bg_ecmwf_error_corr_L91.nc** — Background error correlation matrix in packed form for the previous ECMWF model structure (91 levels).
  - **ropp_bg_ses_ecmwf_error_corr_L60.nc** — Background error correlation matrix in packed form for 60L ECMWF model, generated by the SES method.
  - **ropp_bg_ses_ecmwf_error_corr_L91.nc** — Background error correlation matrix in packed form for 91L ECMWF model, generated by the SES method.
  - **ropp_bg_ses_ecmwf_error_corr_L137.nc** — Background error correlation matrix in packed form for 137L ECMWF model, generated by the SES method.
- **Observation error correlation matrices.** These files are suitable arguments to the '–ob-corr' option to `ropp_1dvar_bangle`.
  - **ropp_ob_bangle_error_corr_300L.nc** — Bending angle observation correlation matrix in packed form. For use with 300 impact heights generated by default by `ropp_fm_bg2ro_1d`.
  - **ropp_ob_bangle_error_corr_247L.nc** — Bending angle observation correlation matrix in packed form. For use with 'standard' 247 impact heights generated by using the '-247L' option to `ropp_fm_bg2ro_1d`.

5.3 Input data

Figure 5.3 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.

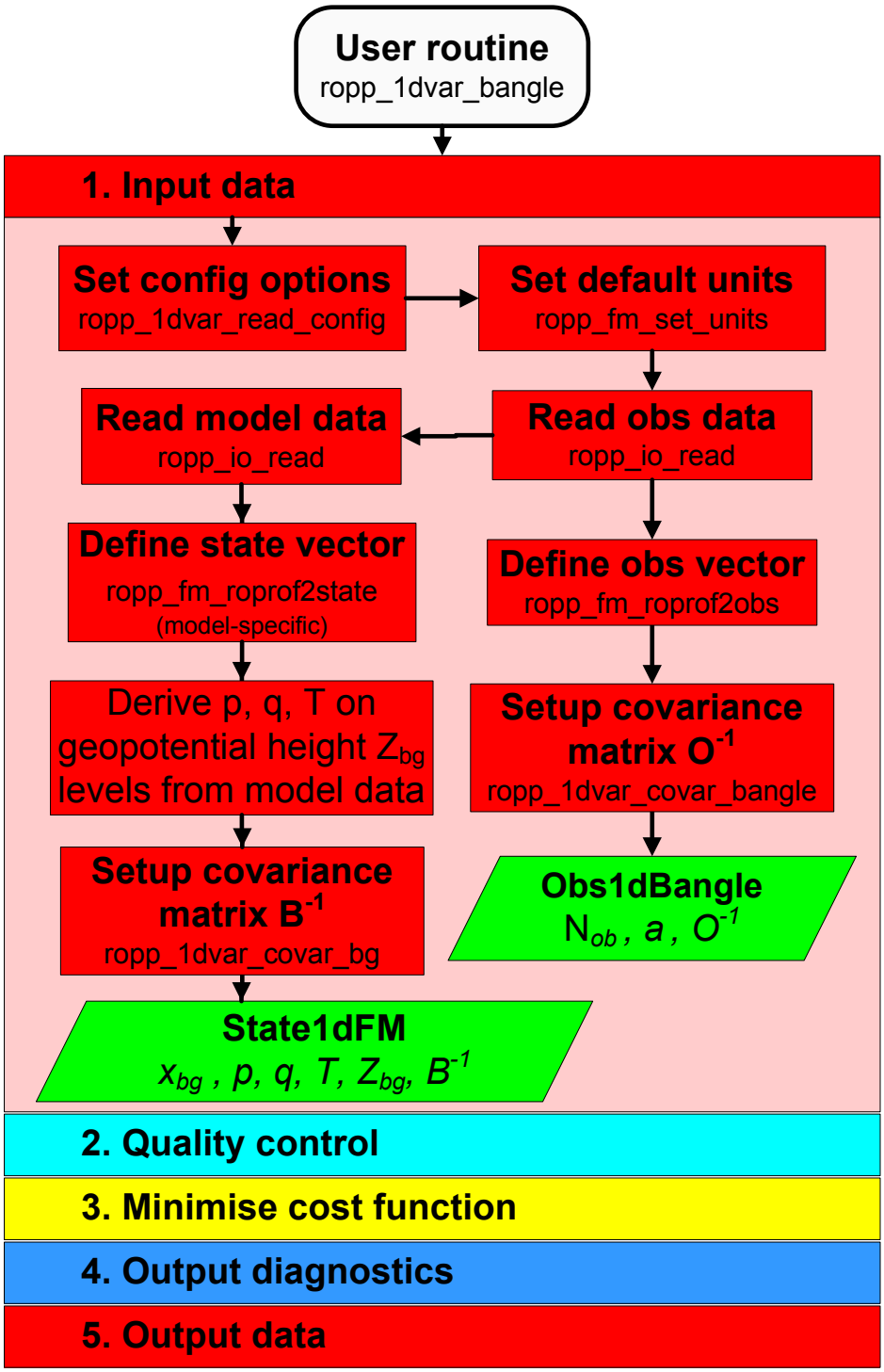

Figure 5.3: Flow chart illustrating the calling tree of the input data step of ROPP 1D–Var to retrieve atmospheric profiles from observed bending angle profiles and input background model data.

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 in the ROPP FM user guide). If required for 1D–Var, users must ensure that the error correlations used are appropriate for background and observation data in ascending height order.

5.3.1 Configuration options

A number of configuration options can be defined by the user in order to tune the performance of the `ropp_1dvar` retrieval. Tables 5.1 and 5.2 list 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.

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. allowed obs height
...%max_1dvar_height	60 km	Max. allowed obs height
...%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_obheight	20 km	Min. required obs height
...%genqc_{min,max}_temperature	150, 350 K	Min./Max. bg data values
...%genqc_{min,max}_spec_humidity	0, 50 g/kg	
...%genqc_{min,max}_impact	6.2e6, 6.6e6 m	Min./Max. obs data values
...%genqc_{min,max}_bangle	-1.0e-4, 0.1 rad	
...%genqc_{min,max}_geop_refrac	-1.e3, 1.e5 m	
...%genqc_{min,max}_refractivity	0.0, 500 N-units	
...%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	Width of gross error plateau
...%minROPP%method	LEVMARQ	Minimisation method
...%minROPP%log_file	screen	minROPP output device
...%minROPP%impres	0	minROPP output mode
...%minROPP%n_iter	1500	minROPP storage parameter
...%minROPP%n_updates	50	Maximum number of iterations
...%minROPP%eps_grad	1.0e-8	minROPP convergence parameter
...%minROPP%dx_min	1.0e-16	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

Table 5.1: Configuration options held as elements of the VarConfig structure which are used by ropp_1dvar routines. The default values are assumed unless overwritten by configuration options that are read from an input configuration file.

VarConfig		
Structure element	Default	Description
...%j_s_limit	5.0	Maximum value of 2J/m in QC
...%n_iter_limit	50	Maximum number of iterations in QC
...%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
...%compress	.FALSE.	Use non-ideal compressibility factors
...%season_amp	0.0	Amplitude of seasonal ob error scaling
...%season_offset	0.0	Constant offset for seasonal scaling
...%season_phase	0.0	Phase of seasonal scaling factor
...%nlayer	1	Number of VaryChapman layers
...%f1	1.57542E9	First frequency (Hz)
...%f2	1.22760E9	Second frequency (Hz)
...%r_gns	2.67E7	Nominal GNSS to CoC distance (m)
...%r_leo	7.19E6	Nominal LEO to CoC distance (m)
...%genqc_min_ne_peak	0.0	Minimum ne_peak (m-3)
...%genqc_max_ne_peak	1.0E15	Maximum ne_peak (m-3)
...%genqc_min_r_peak	6.2E6	Minimum r_peak (m)
...%genqc_max_r_peak	7.4E6	Maximum r_peak (m)
...%genqc_min_h_zero	0.0	Minimum h_zero (m)
...%genqc_max_h_zero	1.0E6	Maximum h_zero (m)
...%genqc_min_h_grad	0.0	Minimum h_grad
...%genqc_max_h_grad	1.0	Maximum h_grad

Table 5.2: Configuration options held as elements of the VarConfig structure (continued).

5.4 Observation data

For bending angles, the routines `ropp_1dvar` and `ropp_fm` use observation data defined as elements of a structure of type `Obs1dBangle`. (See Sec 4.2.1 of ROPP FM UG.)

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, 2021b).

```
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)
```

5.4.1 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`. (See Sec 4.7 of ROPP FM UG.)

5.4.2 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,

$$\mathbf{O} = \sigma.\mathbf{Corr}.\sigma^T \tag{5.2}$$

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`.

ROPP has an option to vary the standard deviations σ according to the time of the year — see Sec 5.4.3 for details.

Observation error covariance options

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 5.1).

- **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 5.2).
- **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 5.2).
- **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 5.2).

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.

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

5.4.3 Seasonal scaling of observation errors

Configuration options exist to apply an optional seasonal dependence to the observation errors read in from the files described above. The error values that are read in are scaled according to the time of year, based on a sinusoidal funtion that takes three parameters, labelled here as Δ, A and ϕ but appearing in Table 5.1 as `season_offset`, `season_amp` and `season_phase` respectively. Here, the time t is the fraction of the way through the year, i.e. between 0 (Jan. 1) and 1 (Dec. 31).

$$\sigma_{scaled} = \sigma_{orig}\left[1+\Delta+A\cos\left(2\pi(t+\phi)\right)\right] \tag{5.3}$$

where Δ is a constant offset, on which the seasonal variation is applied, A is the amplitude of the sinusoidal scaling factor and ϕ is the 'phase' of the sinusoid, i.e. a value of 0.1 will shift the maximum of the sinusoid back one tenth of a year.

This option should be used with care to ensure that realistic values are produced. Recommended usage is to take the read-in error values as the minimum errors for the annual cycle and set $\Delta = A > 0$ to ensure that the scaling will only produce increased sigma values.

A full specification of the seasonal dependence of observation errors would require additional dependence on latitude and height (Scherllin-Pirscher et al. (2011)), but the functionality provided here should provide a starting point for further developments.

Note that this option offers the user a simple way to rescale the observation errors without having to regenerate input files.

5.5 Background data

5.5.1 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` (see Sec 4.4 of the ROPP FM User Guide (2021a)).

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 Secs 4.5 and 4.6 of ROPP FM UG 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.

5.5.2 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,

$$\mathbf{B} = \sigma.\mathbf{Corr}.\sigma^T \tag{5.4}$$

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.

Background error covariance options

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 5.1).

- **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 5.2).
- **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 5.2).

- **RSFC — Relative Sigmas, Fixed Correlations**

  Relative specific humidity (q) errors and relative surface pressure ($p*$) errors, and (absolute) temperature (T) errors, are read from a background error correlation file, as are all the error correlations. (All fields must come from the same file.) The relative q ($p*$) errors are multiplied by the profile q ($p*$) values to give the profile errors. (As before, error values are automatically adjusted if the user sets either `config%use_logp` or `config%use_logq`.) In this case the error correlation file must contain `temp_sigma` (length n), `shum_rel_sigma` (length n), `press_sfc_rel_sigma` (length 1) and `corr` (length $2n+1$). Further details are available in (ROM SAF, 2014). Example files are included in the ROPP distribution — see Section 4.2. RSFC is therefore a hybrid of FSFC and VSFC. *This method can only be applied to background fields on ECMWF levels.*

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. (See Secs 4.5 and 4.6 of ROPP FM UG.)

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 = 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 = 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 $j =$ 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 $j =$ bg%n_lev $+ i$

5.6 Quality control

Figure 5.4 illustrates the implementation of `ropp_1dvar` module routines to perform preliminary quality control on the input data.

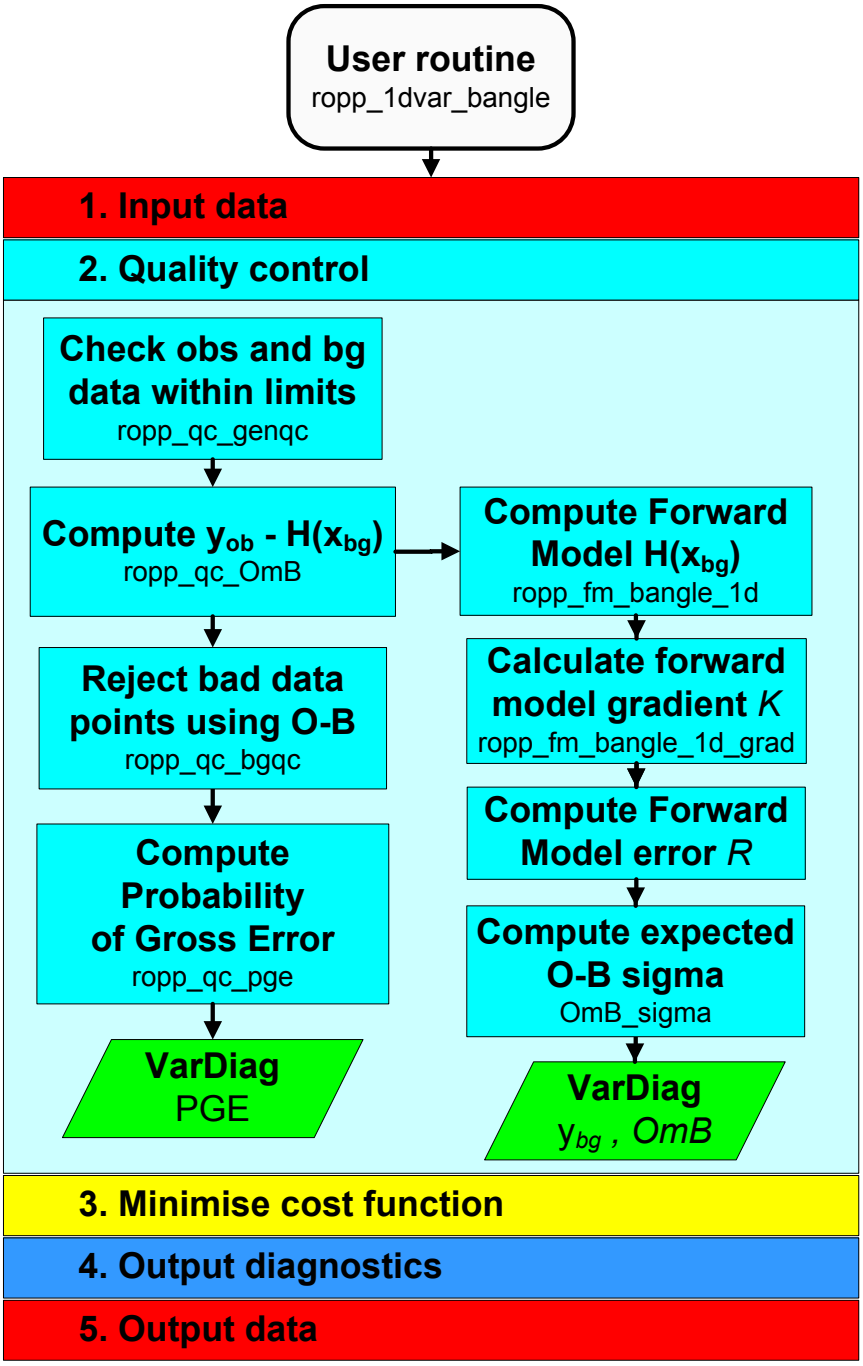

Figure 5.4: Flow chart illustrating the calling tree of the quality control step of ROPP 1D–Var to retrieve atmospheric profiles from observed bending angle profiles and input background model data.

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, bg_data%bg%fcperiod))
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)
```

5.6.1 Valid observation height range: ropp_qc_cutoff()

Configuration options config%min_1dvar_height and config%max_1dvar_height (Table 5.1) 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.

5.6.2 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 5.1). 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.

5.6.3 Observation minus background check: ropp_qc_OmB()

The difference between the observations and the background data forward modelled into observation space (diag%OmB $= \mathbf{y}_{ob} - \mathbf{H}[\mathbf{x}_{bg}]$) and the standard deviation of this difference (diag%OmB_sigma) 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 (see Sec 4.10 of ROPP FM UG) observations.

The error in the observation minus background calculation is given by

$$\texttt{diag\%OmB\_sigma(:)} = (\mathbf{O} + \mathbf{K}.\mathbf{B}.\mathbf{K}^T)^{1/2} \tag{5.5}$$

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_bangle_1d_grad` for bending angle.

5.6.4 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.

5.6.5 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 at each observation point is computed based on the $(\mathbf{y}_{ob} - \mathbf{H}(\mathbf{x}_{bg}))$ 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`.

$$\texttt{diag\%pge(i)} = \left[1 + \gamma^{-1} \exp(-u_i^2/2)\right]^{-1} \tag{5.6}$$

where

$$\mathbf{u} = \frac{\mathbf{y}_{ob} - \mathbf{H}(\mathbf{x}_{bg})}{\sigma(\mathbf{y}_{ob} - \mathbf{H}(\mathbf{x}_{bg}))}. \tag{5.7}$$

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

$$\gamma = \frac{A\sqrt{2\pi}}{(1-A)2d} \tag{5.8}$$

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

5.7 Minimise the cost function

Figure 5.5 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 ($\mathbf{x}$) that minimises the cost function (Equation 5.1) is either `ropp_1dvar_solve`, if `config%minROPP%method = MINROPP`, or `ropp_1dvar_levmarq` otherwise. 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 $\mathbf{x}_{bg}$. On exit $\mathbf{x}$ contains the 1D–Var solution. Both solvers take the same input variables and produce the same output variables (with different *values*, of course, although these should be very similar if both solvers have converged successfully.)

```
USE ropp_fm
USE ropp_1dvar
TYPE(State1dFM)   :: bg
TYPE(State1dFM)   :: state
TYPE(Obs1dBangle) :: obs
...
state = bg    ! initial guess - set state equal to background state
IF (config%minROPP%method == 'MINROPP') THEN
  CALL ropp_1dvar_solve(obs, bg, state, config, diag)
ELSE
  CALL ropp_1dvar_levmarq(obs, bg, state, config, diag)
ENDIF
```

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. By default, the state vector is updated using a Levenberg-Marquardt minimisation algorithm (Press et al., 1992). See Sec 5.7.2 and ROM SAF (2008) for details. Alternatively, if the parameter `config%minROPP%method` equals `MINROPP` then the ROPP-specific minimiser minROPP will be used to minimise the cost function. See Sec 5.7.1 and ROM SAF (2007) for details.

5.7.1 Minimisation with the minROPP scheme

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 transform the initial state variable $\mathbf{x}$ to a control variable $\mathbf{c}$, given a preconditioner $\mathbf{L}$ defined such that $\mathbf{B} = \mathbf{L}\mathbf{L}^T$, thus:

$$\mathbf{c} = \mathbf{L}^{-1}\mathbf{x} \tag{5.9}$$

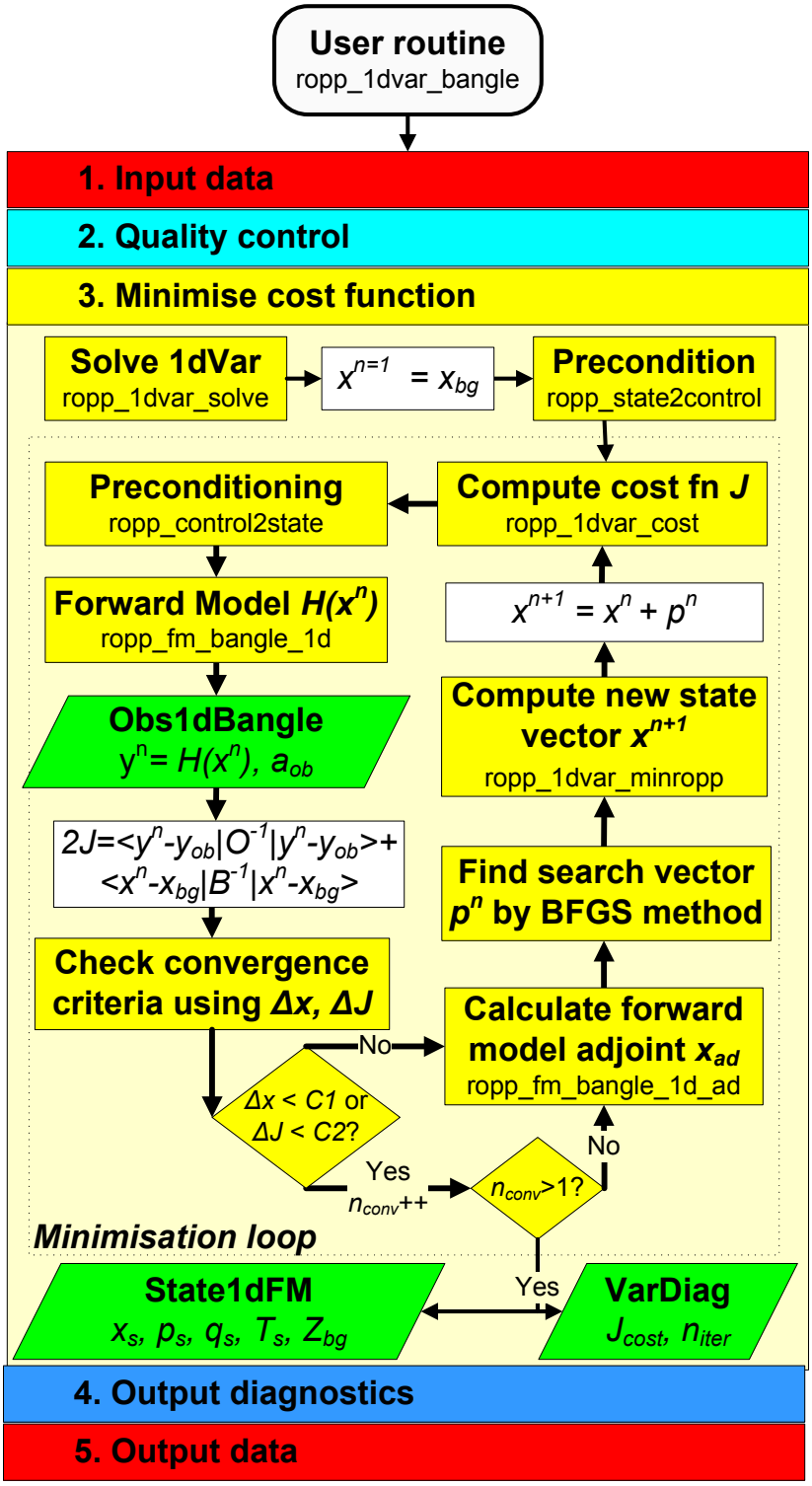

Figure 5.5: Flow chart illustrating the calling tree of the minROPP minimisation step of ROPP 1D–Var to retrieve atmospheric profiles from observed bending angle profiles and input background model data.

No other elements of the `State1dFM` structure are affected 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 $\mathbf{c}$ by calling the routine `ropp_control2state`. If `config%use_precond` is not set, $\mathbf{c}$ is set equal to $\mathbf{x}$.

Compute the cost function

Equation (5.1) 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 $\mathbf{y}_{ob} - \mathbf{H}[\mathbf{x}]$ with the weighting factors determined from the quality control processing.

The matrix multiplication of the differences $(\mathbf{x} - \mathbf{x}_b)$ and $(\mathbf{y}_{ob} - \mathbf{H}[\mathbf{x}])$ 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:

$$\nabla J(\mathbf{x}) = \mathbf{B}^{-1}(\mathbf{x} - \mathbf{x}_b) - (\mathbf{H}[\mathbf{x}]')^T \mathbf{O}^{-1}(\mathbf{y}_{ob} - \mathbf{H}[\mathbf{x}]) \tag{5.10}$$

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_bangle_1d_ad`. On exit, `ropp_1dvar_cost` returns the variable `J_grad`, the cost function gradient with respect to the control variable $\mathbf{c}$, which is calculated by premultiplying $\nabla J(\mathbf{x})$ in Eqn 5.10 by $\mathbf{L}^T$ if preconditioning is being applied.

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 the following two criteria needs to be met on `config%conv_check_n_previous` consecutive iterations for convergence to be achieved.

1. Maximum relative change in state

The state vector has not changed by more than a set value between iterations:

$$\max\left[\frac{|\mathbf{x}_n - \mathbf{x}_{n-1}|}{\sqrt{\mathbf{B}}}\right] < \texttt{config\%conv\_check\_max\_delta\_state}, \tag{5.11}$$

where $\sqrt{\mathbf{B}}$ is a vector comprising the square roots of the diagonal elements of the background covariance matrix $\mathbf{B}$.

2. Maximum change in cost function

The cost function has not changed by more than a set value between iterations:

$$|J_n - J_{n-1}| < \texttt{config\%conv\_check\_max\_delta\_J}. \tag{5.12}$$

When either of the convergence criteria is 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` (Sec 4.5 of ROPP FM UG) or `ropp_fm_state2state_meto` (Sec 4.6 of ROPP FM UG). Diagnostic parameters `diag%J`, `diag%J_scaled` and `diag%n_iter` are set.

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 from `ropp_1dvar_solve` thus:

```
CALL ropp_1dvar_minropp(control%state, J_grad, J_dir, dJ, gconv, &
                        n_iter, m_indic, config%minropp%n_updates, &
                        config%minropp%n_iter)
```

where `control%state` 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`. `n_iter` is an iteration counter, which is incremented each time `minROPP` updates the state vector. `m_indic` is a convergence indication flag, `config%minropp%n_updates` is the maximum number of iterations that are allowed, and `config%minropp%n_iter` defines the size of some workspace.

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.

$$||\nabla J_k|| < \text{eps}||\nabla J_1|| \tag{5.13}$$

Parameter eps equals the configuration element `config%minROPP%eps_grad`, and the right hand side of the above equation is the variable `gconv`, which is input to `ropp_1dvar_minropp`. The routine exits if this condition is met, and the minimisation loop is terminated.

Figure 4.5 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_updates` iterations have been completed, then no convergence can be achieved and the 1D–Var retrieval fails.

5.7.2 Minimisation with the Levenberg-Marquardt scheme

The theory behind the Levenberg-Marquardt minimisation scheme is clearly explained in Press et al. (1992). Loosely speaking, it amounts to an inverse Hessian (or 'Newton-Raphson') increment of the state variable, unless this causes the cost function to increase, in which case the minimisation procedure becomes more like a steepest descent algorithm. This trend towards steepest descent continues, ultimately with a small step size, until the cost function starts to decrease, at which point the algorithm begins to recover its inverse Hessian character.

Figure 5.6 illustrates the implementation of `ropp_1dvar` module routines to minimise the cost function using the Levenberg-Marquardt method and obtain the solution state vector for a given 1D–Var bending angle retrieval. The user might find it useful to compare it to the minROPP equivalent, Figure 5.5.

Preconditioning

Unlike the `minROPP` minimiser, no preconditioning is applied in the Levenberg-Marquardt scheme used in ROPP.

Computation of cost function and its first two derivatives

The cost function J and its first derivative $\nabla J(\mathbf{x})$ are computed directly from Eqns 5.1 and 5.10. ($\mathbf{H}[\mathbf{x}]$ is calculated by means of a call to `ropp_fm_bangle_1d`, and $\mathbf{H}'[\mathbf{x}]$ is calculated by means of a call to `ropp_fm_bangle_1d_grad`.) The second derivative, or Hessian, $\nabla\nabla J(\mathbf{x})$, is calculated from

$$\nabla\nabla J(\mathbf{x}) = \mathbf{B}^{-1} + (\mathbf{H}'[\mathbf{x}])^T \mathbf{O}^{-1} \mathbf{H}'[\mathbf{x}] \tag{5.14}$$

The initial value of J is stored in the diagnostic variable `diag%J_init`.

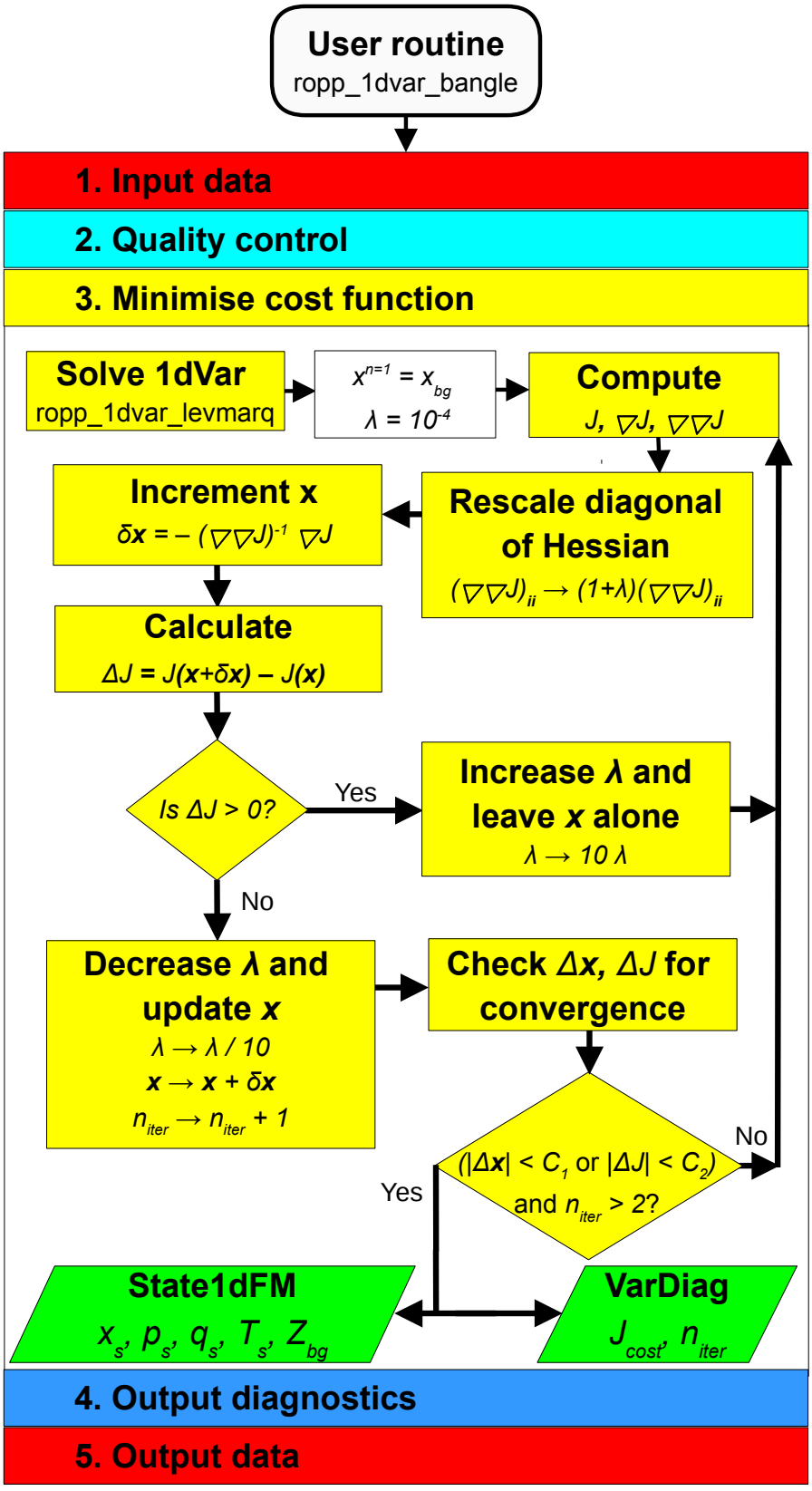

Figure 5.6: Flow chart illustrating the calling tree of the LevMarq minimisation step of ROPP 1D–Var to retrieve atmospheric profiles from observed bending angle profiles and input background model data.

Minimisation

The *diagonal* elements of $\nabla\nabla J(\mathbf{x})$ are multiplied by $(1+\lambda)$, where the dimensionless number λ, the key feature of the Levenberg-Marquardt scheme, controls the relative sizes of the steepest descent and the inverse Hessian characteristics of the minimisation algorithm. Since λ is initialised to 10^{-4}, this multiplication has little effect on the first iteration, which is almost completely inverse Hessian. This means that the increment to the state vector $\mathbf{x}$ is given by

$$\delta\mathbf{x} = -(\nabla\nabla J(\mathbf{x}))^{-1}\nabla J(\mathbf{x}), \tag{5.15}$$

which follows (approximately) from making Eqn 5.10 stationary with respect to small changes in $\mathbf{x}$.

As λ gets larger, the matrix $\nabla\nabla J(\mathbf{x})$ becomes increasingly dominated by its diagonal, so that the increments given by Eqn 5.15 become increasingly parallel to the direction of steepest descent, $-\nabla J(\mathbf{x})$, suitably rescaled, for dimensional propriety, by the covariances that constitute $\nabla\nabla J(\mathbf{x})$ (see Eqn 5.14).

The state vector $\mathbf{x}$ is stored before it is incremented by $\delta\mathbf{x}$: $\mathbf{x} \mapsto \mathbf{x} + \delta\mathbf{x}$.

(Some warning messages are issued if the resulting ionospheric parameters $N_e^{\max}$, H_{peak} and H_{width} take unphysical values. This only happens if the `direct_ion` option to the bending angle retrieval is in force — see Sec 8 for details.)

The cost function is recalculated. If it has increased by more than `config%conv_check_max_delta_J`, then $\mathbf{x}$ reverts to the stored value, λ is multiplied by 10 (making it more of a steepest descents algorithm), and the iteration is repeated. If this procedure results in λ exceeding 10^{10} then, because the scheme is evidently not converging, a message is issued and the algorithm stops.

If, on the other hand, the cost function has not increased by at least `config%conv_check_max_delta_J`, then the updated value of $\mathbf{x}$ is used, and λ is divided by 10 (making it more of an inverse Hessian algorithm). The minimisation undergoes another iteration unless the algorithm is considered to have converged.

Convergence checks

If the absolute value of the change in the cost function J is less than `config%conv_check_max_delta_J` (see Eqn 5.12), and this has been the case for the last `config%conv_check_n_previous` iterations, then the algorithm is deemed to have converged, and the minimisation stops.

Similarly, if the maximum value of the magnitude of the change in state $\delta\mathbf{x}$ divided by $\sqrt{\mathbf{B}}$ is less than `config%conv_check_max_delta_state` (see Eqn 5.11), and this has been the case for the last `config%conv_check_n_previous` iterations, then the algorithm is deemed to have converged, and the minimisation stops.

Finally, if λ is found to be greater than 10^{10} (as a result of successive increases in the cost function), then a warning message is issued and the minimisation stops.

Before leaving subroutine `ropp_1dvar_levmarq_bangle`, the minimum cost function J is copied to the diagnostic variable `diag%J`, the scaled cost function $2J/m$ (where m is the number of non-zero weighted

observations) is copied to `diag%J_scaled`, and the level-by-level state vector 'half' of the final cost function, $(1/2)(\mathbf{x}-\mathbf{b})_i \left(\mathbf{B}^{-1}(\mathbf{x}-\mathbf{b})\right)_i$ (no summation over i), is copied to `diag%J_bgr`.

5.8 Output diagnostics

Figure 5.7 illustrates the implementation of `ropp_1dvar` and `ropp_fm` module routines to compute final output diagnostics associated with a 1D–Var retrieval solution. Table 5.3 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).

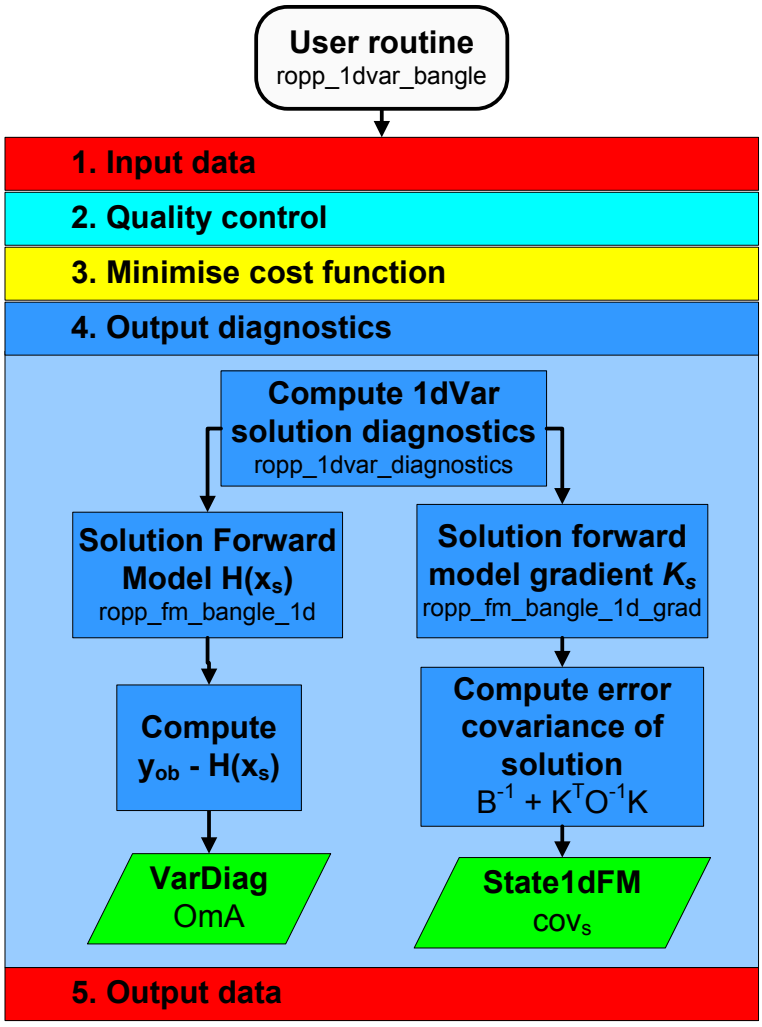

Figure 5.7: Flow chart illustrating the calling tree of the output diagnostics step of ROPP 1D–Var to retrieve atmospheric profiles from observed bending angle profiles and input background model data.

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
```

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
...%bg_ne	Background electron density
...%bg_ne_sigma	Error in background electron density
...%res_ne	Analysis electron density
...%res_ne_sigma	Error in analysis electron density
...%VTEC_bg	VTEC of background electron density
...%VTEC_an	VTEC of analysis electron density

Table 5.3: Elements of VarDiag structure output using the extended_1dvar_diag configuration flag.

```
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 (5.6.3). The diagnostic variable diag%OmA is computed as the difference between the observations ($\mathbf{y}_{ob}$) 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 (Chap. 5 Rodgers, 2000)

$$\mathrm{x\%cov} = \left(\mathbf{B}^{-1} + \mathbf{K}^T\mathbf{O}^{-1}\mathbf{K}\right)^{-1} \tag{5.16}$$

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`.

5.9 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, 2021b).

```
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 (see Sec 4.2.2 of ROPP FM UG).

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 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.

5.10 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.

Holm, E. V. and Kral, T., Flow-dependent, geographically varying background error covariances for 1d-var applications in MTG-IRS l2 processing, Tech memo 680, ECMWF, http://old.ecmwf.int/publications/library/do/references/list/14, 2012.

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, Algorithm Theoretical Baseline Document: NRT and Offline 1D–Var products, SAF/ROM/DMI/ALG/1DV/002, Version 2.4, 2014.

ROM SAF, The Radio Occultation Processing Package (ROPP) Forward model module User Guide, SAF/ROM/METO/UG/ROPP/006, Version 11.0, 2021a.

ROM SAF, The Radio Occultation Processing Package (ROPP) Input/Output module User Guide, SAF/ROM/METO/UG/ROPP/002, Version 11.0, 2021b.

Scherllin-Pirscher, B., Steiner, A. K., Kirchengast, G., Kuo, Y.-H., and Foelsche, U., Empirical analysis and modeling of errors of atmospheric profiles from GPS radio occultation, *Atmospheric Measurement Techniques*, *4*, 1875–1890, 2011.

6 ROPP 1D–Var: Differenced bending angle

Note that this Section is similar to that of Sec 5, apart from the references to differenced bending angle rather than bending angle, and some associated minor differences. There is little to be gained from reading both.

The ROPP 1D–Var module (`ropp_1dvar`) includes routines to retrieve the parameters that define a model ionospheric electron density profile that consists of multiple 'VaryChap' layers. The retrieval uses the difference between bending angles at two frequencies (f2 and f1), the *a priori* knowledge of the state of the ionosphere (i.e. background profiles), and the associated errors of each. (Note that f2 and f1 are not necessarily equal to L2 and L1.) This is achieved in the `ropp_1dvar_levmarq` 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 `State0dFM` containing four ionospheric parameters per VaryChap layer, the number of which can be chosen by the user. Naturally, the number of layers defines the size of the background state $\mathbf{x}_b$ and its covariance $\mathbf{B}$. It could also affect the height range over which the retrieval is to be made. $\mathbf{x}$ contains the retrieved ionospheric parameters. $\mathbf{x}_b$ is the initial background state of type `State0dFM` defined by input background ionospheric parameters. 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 differenced bending angle then $\mathbf{y}_o$ is an observation vector of type `Obs1dBangle` which contains the f1 and f2 bending angles 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 the differenced bending angle (see Sec 8 of the ROPP FM User Guide (2021a)) for a given ionospheric state.

Figure 6.1 shows example differenced bending angle and electron density profiles retrieved from (global) background model data and GNSS-RO differenced bending angle observations.

6.1 ROPP 1D–Var differenced bending angle tool

The stand-alone tool `ropp_1dvar_dbangle` is provided in `ropp_1dvar` as an illustration of how the `ropp_1dvar` routines can be implemented to retrieve ionospheric parameters from differenced bending angle observations. Figure 6.2 shows how the `ropp_1dvar` routines are integrated in the `ropp_1dvar_dbangle` code.

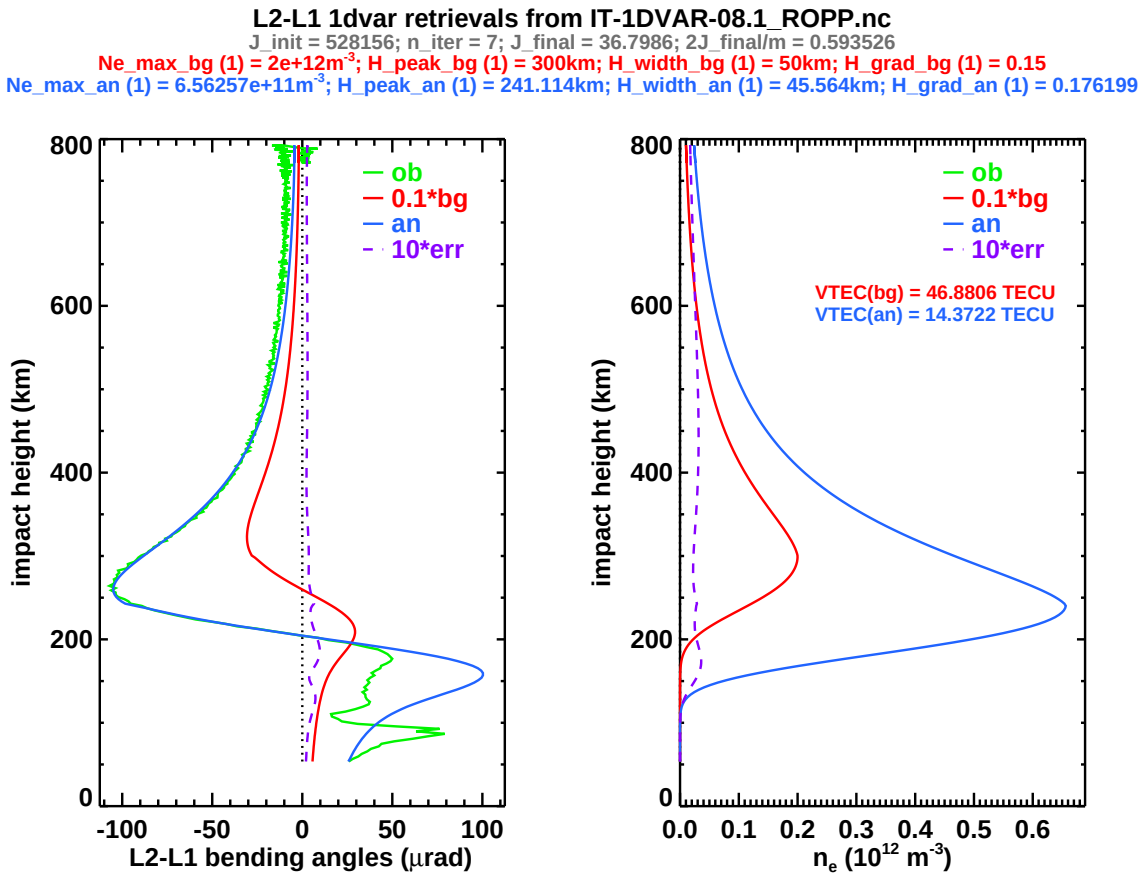

Figure 6.1: Example single layer ropp_1dvar_dbangle output. Left hand plot: $\alpha_{f2} - \alpha_{f1}$ from observations (green), forward modelled background (red), forward modelled analysis (blue), and analysis error estimate (the square root of the diagonal elements of the analysis error covariance matrix $\mathbf{A}$) (purple). Right hand plot: electron density profiles from forward modelled background (red), forward modelled analysis (blue) and analysis error estimate (purple). Text: convergence parameters (grey), background VaryChap parameters (red) and analysis VaryChap parameters (blue). Background and analysis VTECs also shown.

6.1.1 Implementation

The `ropp_1dvar_dbangle` tool is run using the command

```
ropp_1dvar_dbangle [options] --no-ranchk -o <outputfile>
```

where `<outputfile>` is a netCDF file in ROPP format (ROM SAF, 2021b), which will hold the retrieved ionospheric parameters (four per layer) as well as the forward modelled retrieved solution. The latter includes the reconstructed electron density profile (`n_e`) and its height variable (`r_iono`), as well as the difference between the bending angles at the f2 and f1 frequencies, which is stored in the `bangle` variable and its corresponding height variable (`impact`). For ionospheric variables the impact parameters can extend far beyond the default impact parameter valid range (ROM SAF, 2021b), and therefore would be nullified by range-checking unless this is bypassed by means of the mandatory `--no-ranchk` option.

The executable has the following options.

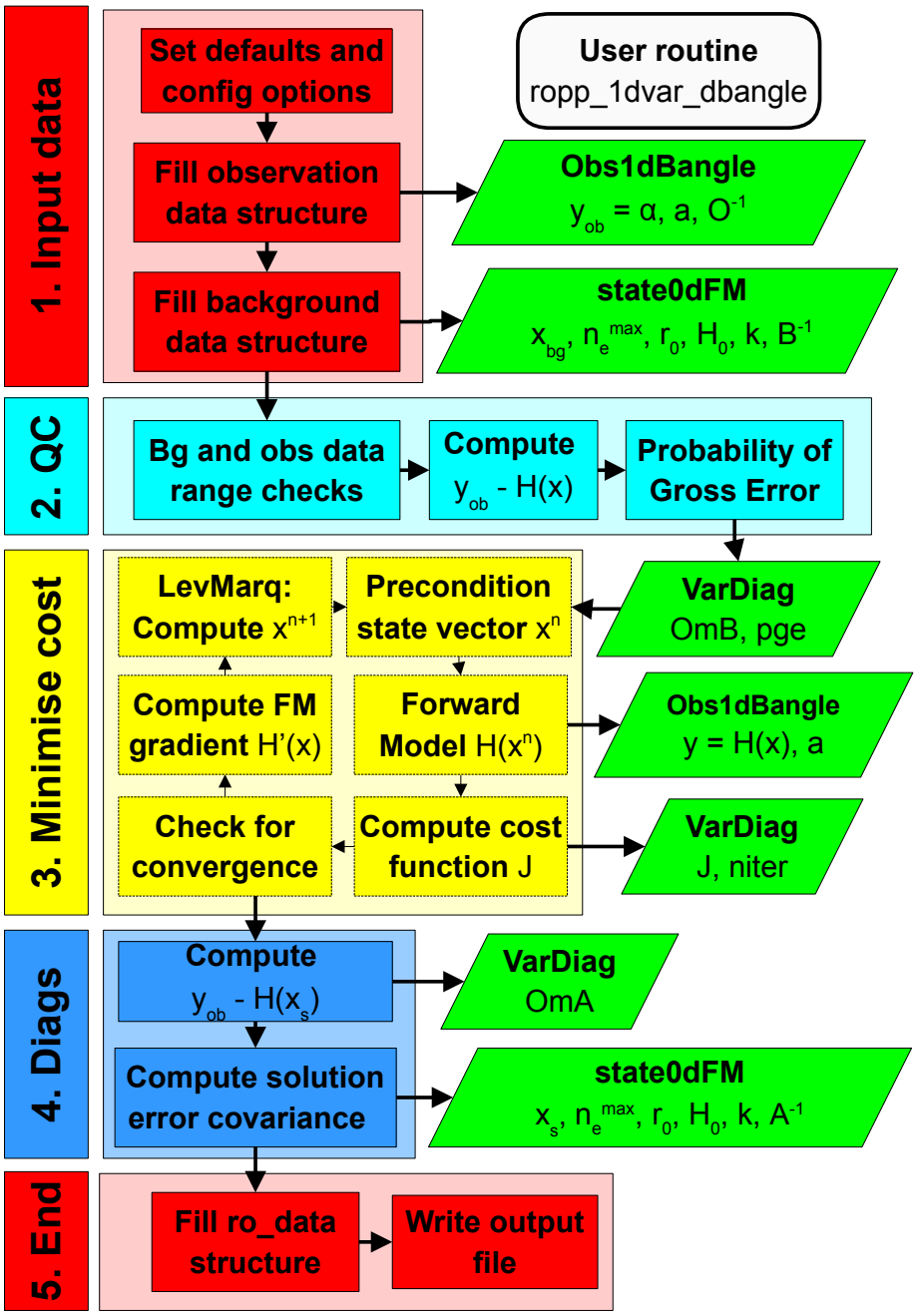

Figure 6.2: Flow chart illustrating the calling tree of the ROPP 1D–Var to retrieve ionospheric parameters from observed differenced bending angle profiles and input background model data.

`-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
`-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_dbangle` 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_dbangle` tool is composed of the following stages:

- **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 `State0dFM` containing the four defining parameters $\{n_e^{\max}, r_0, H_0, k\}$ of each VaryChap layer. (See Eqns (8.1)–Eqns (8.3) of the ROPP FM user guide (ROM SAF, 2021a) for the equation that defines the electron density in terms of these parameters.) Define an observation vector structure containing the observed f1 and f2 bending angles α_{f1} and α_{f2} 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 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 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 forward modelled background state 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_levmarq` is the routine within `ropp_1dvar` that finds the ionospheric state vector $\mathbf{x}$ that minimises the cost function for given backgrounds, observations and their associated errors. Its use requires the `VarConfig` parameter `config%minROPP%method` to be set to `'LEVMARQ'`; if it isn't, the retrieval will bail out. It undertakes the same three steps on each iteration towards a solution:

  - Compute the cost function (Equation 4.1) and its gradient. 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 a minimiser to update the state vector $\mathbf{x}$ towards a solution. This is part of the `ropp_1dvar_levmarq` subroutine.
  - 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. Users are encouraged to adapt the provided routines to meet their own applications.

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. It is generally recommended that `ropp_1dvar_dbangle` is run with configuration options `obs_covar_method` = 'VSDC' (if the number of observations is unknown) or 'FSFC' (if the number of observation levels is known), and `bg_covar_method` = 'VSDC'. Although `obs_covar_method` = 'FSFC' allows for correlations between differenced bending angles at different heights, the example observation correlation files in fact just contain the identity matrix.)

- **Background error correlation matrices.** These files are suitable arguments to the '–bg-corr' option to `ropp_1dvar_dbangle`.
  - **ropp_bg_iono_error_corr_1L.nc** — Background error correlation matrix suitable for one VaryChap layer.
  - **ropp_bg_iono_error_corr_2L.nc** — Background error correlation matrix suitable for two VaryChap layers.
- **Observation error correlation matrices.** These files are suitable arguments to the '–ob-corr' option to `ropp_1dvar_dbangle`.
  - **ropp_ob_dbangle_error_corr_201L.nc** — Bending angle observation correlation matrix in packed form. For use with 201 impact heights, uniformly spaced every 1 km between 80 km and 280 km.
  - **ropp_ob_dbangle_error_corr_511L.nc** — Bending angle observation correlation matrix in packed form. For use with 511 impact heights, uniformly spaced every 1 km between 80 km and 590 km.

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.

Note that the ROPP forward model assumes observational data are in ascending height order. The input data are checked and reordered as part of the stand-alone tool processing (see 4.4 in the ROPP FM user guide). If required for 1D–Var, users must ensure that the error correlations used are appropriate for 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. Tables 6.1 and 6.2 list 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_dbangle` 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)
```

ROPP-11 can handle multiple VaryChap layers, by setting up the background data and covariance files appropriately, and adapting the configuration file. A number of sample configuration files are included with the ROPP distribution in the `ropp_1dvar/config` directory. In practice, retrieving one layer is robust and fast, but can (obviously) only model the main F-layer of the ionosphere. Something like `default_dbangle_onelayer_1dvar.cf` will allow the user to do this. Two layers can generate some useful and apparently genuine structure between 100 km and 150 km, but requires more iterations and fails to converge more often. A configuration file like `default_dbangle_twolayer_1dvar.cf` will allow the user to do this. The benefits of using more layers are yet to be proven. Indeed, there is some evidence that adding more layers makes it harder for the solver to converge. Two layers is therefore probably a good place for the new user to start.

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. allowed obs height
...%max_1dvar_height	60 km	Max. allowed obs height
...%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_obheight	20 km	Min. required obs height
...%genqc_{min,max}_temperature	150, 350 K	Min./Max. bg data values
...%genqc_{min,max}_spec_humidity	0, 50 g/kg	
...%genqc_{min,max}_impact	6.2e6, 6.6e6 m	Min./Max. obs data values
...%genqc_{min,max}_bangle	-1.0e-4, 0.1 rad	
...%genqc_{min,max}_geop_refrac	-1.e3, 1.e5 m	
...%genqc_{min,max}_refractivity	0.0, 500 N-units	
...%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	Width of gross error plateau
...%minROPP%method	LEVMARQ	Minimisation method
...%minROPP%log_file	screen	minROPP output device
...%minROPP%impres	0	minROPP output mode
...%minROPP%n_iter	1500	minROPP storage parameter
...%minROPP%n_updates	50	Maximum number of iterations
...%minROPP%eps_grad	1.0e-8	minROPP convergence parameter
...%minROPP%dx_min	1.0e-16	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

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 configuration options that are read from an input configuration file.

VarConfig		
Structure element	Default	Description
...%j_s_limit	5.0	Maximum value of 2J/m in QC
...%n_iter_limit	50	Maximum number of iterations in QC
...%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
...%compress	.FALSE.	Use non-ideal compressibility factors
...%season_amp	0.0	Amplitude of seasonal ob error scaling
...%season_offset	0.0	Constant offset for seasonal scaling
...%season_phase	0.0	Phase of seasonal scaling factor
...%nlayer	1	Number of VaryChapman layers
...%f1	1.57542E9	First frequency (Hz)
...%f2	1.22760E9	Second frequency (Hz)
...%r_gns	2.67E7	Nominal GNSS to CoC distance (m)
...%r_leo	7.19E6	Nominal LEO to CoC distance (m)
...%genqc_min_ne_peak	0.0	Minimum ne_peak (m-3)
...%genqc_max_ne_peak	1.0E15	Maximum ne_peak (m-3)
...%genqc_min_r_peak	6.2E6	Minimum r_peak (m)
...%genqc_max_r_peak	7.4E6	Maximum r_peak (m)
...%genqc_min_h_zero	0.0	Minimum h_zero (m)
...%genqc_max_h_zero	1.0E6	Maximum h_zero (m)
...%genqc_min_h_grad	0.0	Minimum h_grad
...%genqc_max_h_grad	1.0	Maximum h_grad

Table 6.2: Configuration options held as elements of the VarConfig structure (continued).

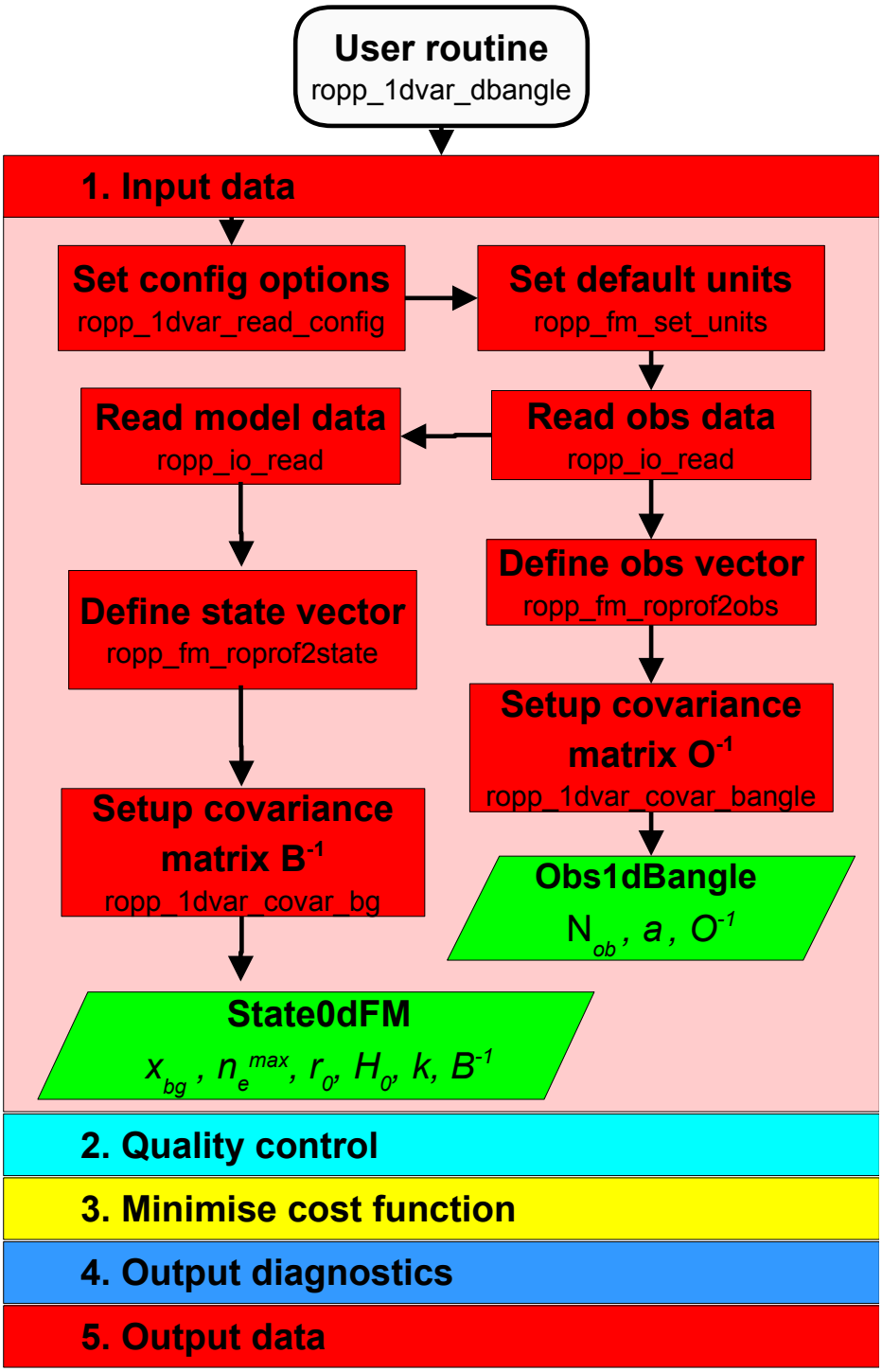

Figure 6.3: Flow chart illustrating the calling tree of the input data step of ROPP 1D–Var to retrieve ionospheric profiles from observed bending angle profiles and input background model data.

6.4 Observation data

For bending angles, the routines `ropp_1dvar` and `ropp_fm` use observation data defined as elements of a structure of type `Obs1dBangle`. (See Sec 4.2.1 of ROPP FM UG.)

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, 2021b).

```
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 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. (See Sec 4.7 of ROPP FM UG.)

6.4.2 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,

$$\mathbf{O} = \sigma.\mathbf{Corr}.\sigma^T \tag{6.2}$$

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.

ROPP has an option to vary the standard deviations σ according to the time of the year — see Sec 6.4.3 for details.

Observation error covariance options

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 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.

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

6.4.3 Seasonal scaling of observation errors

Configuration options exist to apply an optional seasonal dependence to the observation errors read in from the files described above. The error values that are read in are scaled according to the time of year, based

on a sinusoidal funtion that takes three parameters, labelled here as Δ, A and ϕ but appearing in Table 5.1 as `season_offset`, `season_amp` and `season_phase` respectively. Here, the time t is the fraction of the way through the year, i.e. between 0 (Jan. 1) and 1 (Dec. 31).

$$\sigma_{scaled} = \sigma_{orig}\left[1 + \Delta + A\cos\left(2\pi(t+\phi)\right)\right] \tag{6.3}$$

where Δ is a constant offset, on which the seasonal variation is applied, A is the amplitude of the sinusoidal scaling factor and ϕ is the 'phase' of the sinusoid, i.e. a value of 0.1 will shift the maximum of the sinusoid back one tenth of a year.

This option should be used with care to ensure that realistic values are produced. Recommended usage is to take the read-in error values as the minimum errors for the annual cycle and set $\Delta = A > 0$ to ensure that the scaling will only produce increased sigma values.

A full specification of the seasonal dependence of observation errors would require additional dependence on latitude and height (Scherllin-Pirscher et al. (2011)), but the functionality provided here should provide a starting point for further developments.

Note that this option offers the user a simple way to rescale the observation errors without having to regenerate input files.

6.5 Background data

6.5.1 Defining the state vector: `ropp_fm_roprof2state`

Background ionospheric parameters are represented in `ropp_fm` and `ropp_1dvar` by the `State0dFM` data structure. The relevant background data are copied to elements of `State0dFM` by calling `ropp_fm_roprof2state` (see Sec 4.4 of the ROPP FM User Guide (2021a)).

The `State0dFM` structure used by `ropp_fm` routines is required to contain the four VaryChap parameters $n_e^{\max}$, r_0, H_0 and k for every layer. The elements of the state vector are extracted from the `bg_data%Lev2e` background data provided by the user.

State vector

The elements of the state vector `bg%state` used in the `ropp_1dvar` analysis are calculated by `ropp_fm_roprof2state` and converted back to ROProf structure by `ropp_fm_state2roprof`.

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(State0dFM)   :: 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 $\mathbf{B}$ are held in the `State0dFM` structure for use in the `ropp_1dvar` processing as element `bg%cov`.

Background error covariance options

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).
- **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 background data file. Note that the input background file must contain valid error estimates for all banckground vector elements. In this case, no background 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).

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.

6.6 Quality control

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

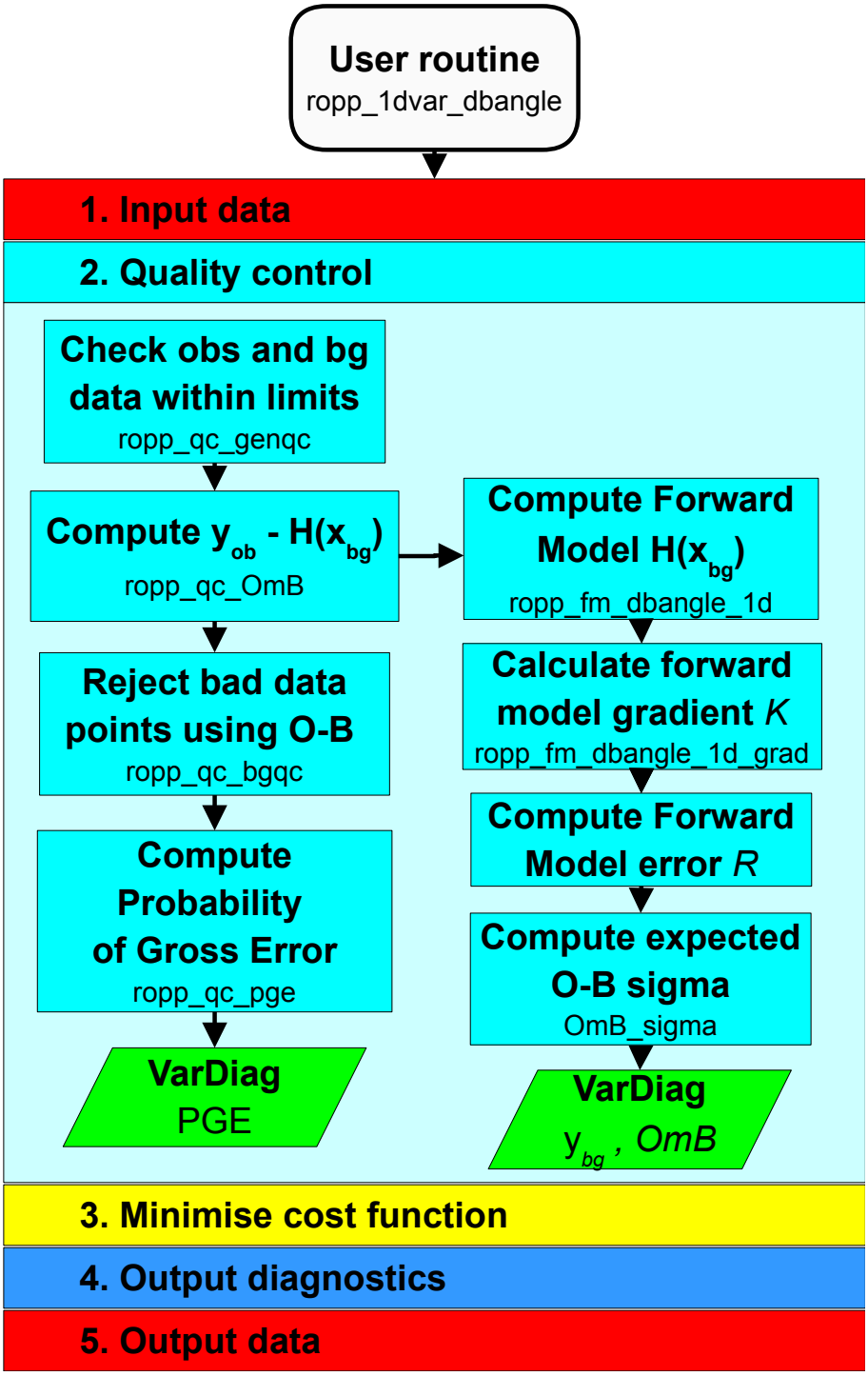

Figure 6.4: Flow chart illustrating the calling tree of the quality control step of ROPP 1D–Var to retrieve ionospheric profiles from observed bending angle profiles and input background model data.

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(State0dFM)   :: bg
TYPE(Obs1dBangle) :: obs
TYPE(VarConfig)   :: config
TYPE(VarDiag)     :: diag
diag % ok = .true.
IF (diag % ok) CALL ropp_qc_cutoff(obs, config)
IF (diag % ok) CALL ropp_qc_genqc(obs, bg, config, diag, bg_data%bg%fcperiod)
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. A suitable range for ionospheric profiles might be something like

```
min_1dvar_height = 200.0  # minimum observation height (km)
max_1dvar_height = 500.0  # maximum observation height (km)
```

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 if config%genqc_colocation_apply is .TRUE., which ensures that the great circle distance between the observation and background coordinates is within a pre-defined limit config%genqc_max_distance. Similarly, this option also checks that the temporal separation of background and observation data are within config%genqc_max_time_sep, and config%ok set to false if they are not. This option is not recommended for differenced bending angles, however, because the background fields are usually global ones, for which such proximity tests are not appropriate. Users should therefore ensure that config%genqc_colocation_apply is set to .FALSE. in their configuration files, as they are in the example ionospheric configuration files in ropp_1dvar/config.

For ionospheric retrievals, the minimum and maximum allowable impact parameters might perhaps be something like

```
genqc_min_impact = 6.2e6  # m
genqc_max_impact = 7.4e6  # m
```

Similarly, for ionospheric retrievals `config%genqc_min_obheight` — the height beneath which the profile must penetrate for any processing to take place — should probably be set to something like

```
genqc_min_obheight = 250.0e3  # m
```

rather than the default 20 km (see Table 6.1).

Again, for ionospheric retrievals, it is also necessary to extend the range of allowable bending angles, particularly to allow more negative ones. Users might therefore like to specify something like

```
genqc_min_bangle = -1.0e-3  # rad
genqc_max_bangle =  1.0e-3  # rad
```

in their configuration files.

See the example ionosphere retrieval configuration files in `ropp_1dvar/config` for more examples of possible settings.

6.6.3 Observation minus background check: `ropp_qc_OmB()`

The difference between the observations and the background data forward modelled into observation space (`diag%OmB` $= \mathbf{y}_{ob} - \mathbf{H}[\mathbf{x}_{bg}]$) and the standard deviation of this difference (`diag%OmB_sigma`) 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 (see Sec 4.10 of ROPP FM UG) 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_dbangle_1d_grad` for bending angle.

Forward modelled background electron density profiles, as well as their implied errors, are also added to the `bg_ne` and `bg_ne_sigma` elements of the `VarDiag` structure.

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.

In view of the global nature of the background fields for ionospheric retrievals, users are strongly recommended to switch off the background quality control check in such cases, by setting

```
bgqc_apply = .false.
```

in their configuration files.

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 at each observation point is computed based on the $(\mathbf{y}_{ob} - \mathbf{H}(\mathbf{x}_{bg}))$ 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`.

$$\texttt{diag\%pge(i)} = \left[1 + \gamma^{-1} \exp(-u_i^2/2)\right]^{-1} \tag{6.6}$$

where

$$\mathbf{u} = \frac{\mathbf{y}_{ob} - \mathbf{H}(\mathbf{x}_{bg})}{\sigma(\mathbf{y}_{ob} - \mathbf{H}(\mathbf{x}_{bg}))}. \tag{6.7}$$

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

$$\gamma = \frac{A\sqrt{2\pi}}{(1-A)2d} \tag{6.8}$$

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 ropp_1dvar routine for retrieving the solution state vector ($\mathbf{x}$) that minimises the cost function (Equation 6.1) is ropp_1dvar_levmarq. On entry the solution state vector $\mathbf{x}$ of type State0dFM is set to a first guess solution of $\mathbf{x}_{bg}$. On exit $\mathbf{x}$ contains the 1D–Var solution.)

```
USE ropp_fm
USE ropp_1dvar
TYPE(State0dFM)   :: bg
TYPE(State0dFM)   :: state
TYPE(Obs1dBangle) :: obs
...
state = bg    ! initial guess - set state equal to background state
...
IF (config%minROPP%method == 'MINROPP') THEN
  CALL message(msg_fatal, &
      "Only Levenberg-Marquardt solver enabled for differenced bending " // &
      "angle 1Dvar ... set config%minROPP%method to 'LEVMARQ' in " // &
      "configuration file and retry.")
ELSE
  CALL ropp_1dvar_levmarq(obs, bg, state, config, diag)
END IF
```

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 Levenberg-Marquardt minimisation algorithm (Press et al., 1992) will be used to minimise the cost function. See Sec 6.7.1 and ROM SAF (2008) for details.

6.7.1 Minimisation with the Levenberg-Marquardt scheme

The theory behind the Levenberg-Marquardt minimisation scheme is clearly explained in Press et al. (1992). Loosely speaking, it amounts to an inverse Hessian (or 'Newton-Raphson') increment of the state variable, unless this causes the cost function to increase, in which case the minimisation procedure becomes more like a steepest descent algorithm. This trend towards steepest descent continues, ultimately with a small step size, until the cost function starts to decrease, at which point the algorithm begins to recover its inverse Hessian character.

Figure 6.5 illustrates the implementation of ropp_1dvar module routines to minimise the cost function using the Levenberg-Marquardt method and obtain the solution state vector for a given 1D–Var differenced bending angle retrieval.

Preconditioning

Preconditioning — multiplication of the state vector by $\mathbf{B}^{-1/2}$, which normalises the state elements — is applied in the Levenberg-Marquardt scheme if `config%use_precond = .TRUE.`.

Computation of cost function and its first two derivatives

The cost function J is computed directly from Eqn 6.1. Its first derivative $\nabla J(\mathbf{x})$ is calculated from

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

($\mathbf{H}[\mathbf{x}]$ is calculated by means of a call to `ropp_fm_dbangle_1d`, and $\mathbf{H}'[\mathbf{x}]$ is calculated by means of a call to `ropp_fm_dbangle_1d_grad`, which populates the matrix by simply calling the tangent linear code `n_state` times, with a different basis vector for $\delta\mathbf{x}$ each time.) The second derivative, or Hessian, $\nabla\nabla J(\mathbf{x})$, is calculated from

$$\nabla\nabla J(\mathbf{x}) = \mathbf{B}^{-1} + (\mathbf{H}'[\mathbf{x}])^T \mathbf{O}^{-1} \mathbf{H}'[\mathbf{x}]. \tag{6.10}$$

The initial value of J is stored in the diagnostic variable `diag%J_init`.

Minimisation

The *diagonal* elements of $\nabla\nabla J(\mathbf{x})$ are multiplied by $(1+\lambda)$, where the dimensionless number λ, the key feature of the Levenberg-Marquardt scheme, controls the relative sizes of the steepest descent and the inverse Hessian characteristics of the minimisation algorithm. Since λ is initialised to 10^{-5}, this multiplication has little effect on the first iteration, which is almost completely inverse Hessian. This means that the increment to the state vector $\mathbf{x}$ is given by

$$\delta\mathbf{x} = -(\nabla\nabla J(\mathbf{x}))^{-1} \nabla J(\mathbf{x}), \tag{6.11}$$

which follows (approximately) from making Eqn 6.9 stationary with respect to small changes in $\mathbf{x}$.

As λ gets larger, the matrix $\nabla\nabla J(\mathbf{x})$ becomes increasingly dominated by its diagonal, so that the increments given by Eqn 6.11 become increasingly parallel to the direction of steepest descent, $-\nabla J(\mathbf{x})$, suitably rescaled, for dimensional propriety, by the covariances that constitute $\nabla\nabla J(\mathbf{x})$ (see Eqn 6.10).

The state vector $\mathbf{x}$ is stored before it is incremented according to

```
x%state(i) = x%state(i) + SIGN(MIN(ABS(delta_x(i)), state_sigma(i)), delta_x(i))
```

In other words, $x_i \mapsto x_i + \delta x_i$ if $|\delta x_i| < \sigma_i$, or $x_i \mapsto x_i + \sigma_i \mathrm{sgn}(\delta x_i)$ if $|\delta x_i| \geq \sigma_i$.

The elements of the updated state vector are examined and altered further, if necessary, as follows:

- If $n_e^{\mathrm{max}} < 0$, then $n_e^{\mathrm{max}} \mapsto 0.01\sigma_{n_e}$
- If $r_{\mathrm{peak}} < 0.1\sigma_{r_{\mathrm{peak}}}$, then $r_{\mathrm{peak}} \mapsto 0.1\sigma_{r_{\mathrm{peak}}}$

- If $H_0 < 0.1\sigma_{H_0}$, then $H_0 \mapsto 0.1\sigma_{H_0}$
- If $k < 10^{-10}\sigma_k$, then $k \mapsto 10^{-10}\sigma_k$

Users should be aware that the above choices are preliminary ones, which have been found to give acceptable results during initial testing. But other choices are possible, and may well be adopted if subsequent testing suggests it would be prudent to do so. For instance, at least one user has had success with

```
x%state(i) = x%state(i) + SIGN(MIN(ABS(delta_x(i)), 0.75*ABS(x%state(i)), delta_x(i))
```

Users should therefore treat the above choices as subject to change. They are encouraged to experiment with this and other options.

The cost function is recalculated. If it has increased by more than `config%conv_check_max_delta_J`, then $\mathbf{x}$ reverts to the stored value, λ is multiplied by 100 (making it more of a steepest descents algorithm), and the iteration is repeated. If this procedure results in λ exceeding 10^{10} then, because the scheme is evidently not converging, a message is issued and the algorithm stops.

If, on the other hand, the cost function has not increased by at least `config%conv_check_max_delta_J`, then the updated value of $\mathbf{x}$ is used, and λ is divided by 10 (making it more of an inverse Hessian algorithm). The minimisation undergoes another iteration unless the algorithm is judged to have converged.

Convergence checks

If the absolute value of the change in the cost function J is less than `config%conv_check_max_delta_J`, and this has been the case for the last `config%conv_check_n_previous` iterations, then the algorithm is deemed to have converged, and the minimisation stops.

Similarly, if the maximum value of the magnitude of the change in state $\delta\mathbf{x}$ divided by $\sqrt{\mathbf{B}}$ is less than `config%conv_check_max_delta_state`, and this has been the case for the last `config%conv_check_n_previous` iterations, then the algorithm is deemed to have converged, and the minimisation stops.

Finally, if λ is found to be greater than 10^{10} (as a result of successive increases in the cost function), then a warning message is issued and the minimisation stops.

Before leaving subroutine `ropp_1dvar_levmarq_dbangle`, the minimum cost function J is copied to the diagnostic variable `diag%J`, the scaled cost function $2J/m$ (where m is the number of non-zero weighted observations) is copied to `diag%J_scaled`, and the level-by-level state vector 'half' of the final cost function, $(1/2)(\mathbf{x}-\mathbf{b})_i \left(\mathbf{B}^{-1}(\mathbf{x}-\mathbf{b})\right)_i$ (no summation over i), is copied to `diag%J_bgr`.

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 6.3 lists the diagnostics which may be optionally output if the config%extended_1dvar_diag configuration flag is set.

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
...%bg_ne	Background electron density
...%bg_ne_sigma	Error in background electron density
...%res_ne	Analysis electron density
...%res_ne_sigma	Error in analysis electron density
...%VTEC_bg	VTEC of background electron density
...%VTEC_an	VTEC of analysis electron density

Table 6.3: Elements of VarDiag structure output using the extended_1dvar_diag configuration flag.

The ropp_1dvar diagnostic routine is ropp_1dvar_diagnostics which is called as,

```
USE ropp_fm
USE ropp_1dvar
TYPE(State0dFM)   :: 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 (6.6.3). The diagnostic variable `diag%OmA` is computed as the difference between the observations ($\mathbf{y}_{ob}$) 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 (Chap. 5 Rodgers, 2000)

$$\mathrm{x\%cov} = \left(\mathbf{B}^{-1} + \mathbf{K}^T\mathbf{O}^{-1}\mathbf{K}\right)^{-1} \tag{6.12}$$

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_dbangle_1d_grad`.

Forward modelled analysis electron density profiles, as well as their implied errors, are also added to the `res_ne` and `bes_ne_sigma` elements of the `VarDiag` structure.

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, 2021b).

```
USE ropp_io
USE ropp_fm
USE ropp_1dvar
TYPE(ROprof)       :: res_data
TYPE(State0dFM)    :: 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 Level 2e ionospheric variables by `ropp_fm_state2roprof`.

Level 1b (differenced 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 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.

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.

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

ROM SAF, The Radio Occultation Processing Package (ROPP) Forward model module User Guide, SAF/ROM/METO/UG/ROPP/006, Version 11.0, 2021a.

ROM SAF, The Radio Occultation Processing Package (ROPP) Input/Output module User Guide, SAF/ROM/METO/UG/ROPP/002, Version 11.0, 2021b.

Scherllin-Pirscher, B., Steiner, A. K., Kirchengast, G., Kuo, Y.-H., and Foelsche, U., Empirical analysis and modeling of errors of atmospheric profiles from GPS radio occultation, *Atmospheric Measurement Techniques*, *4*, 1875–1890, 2011.

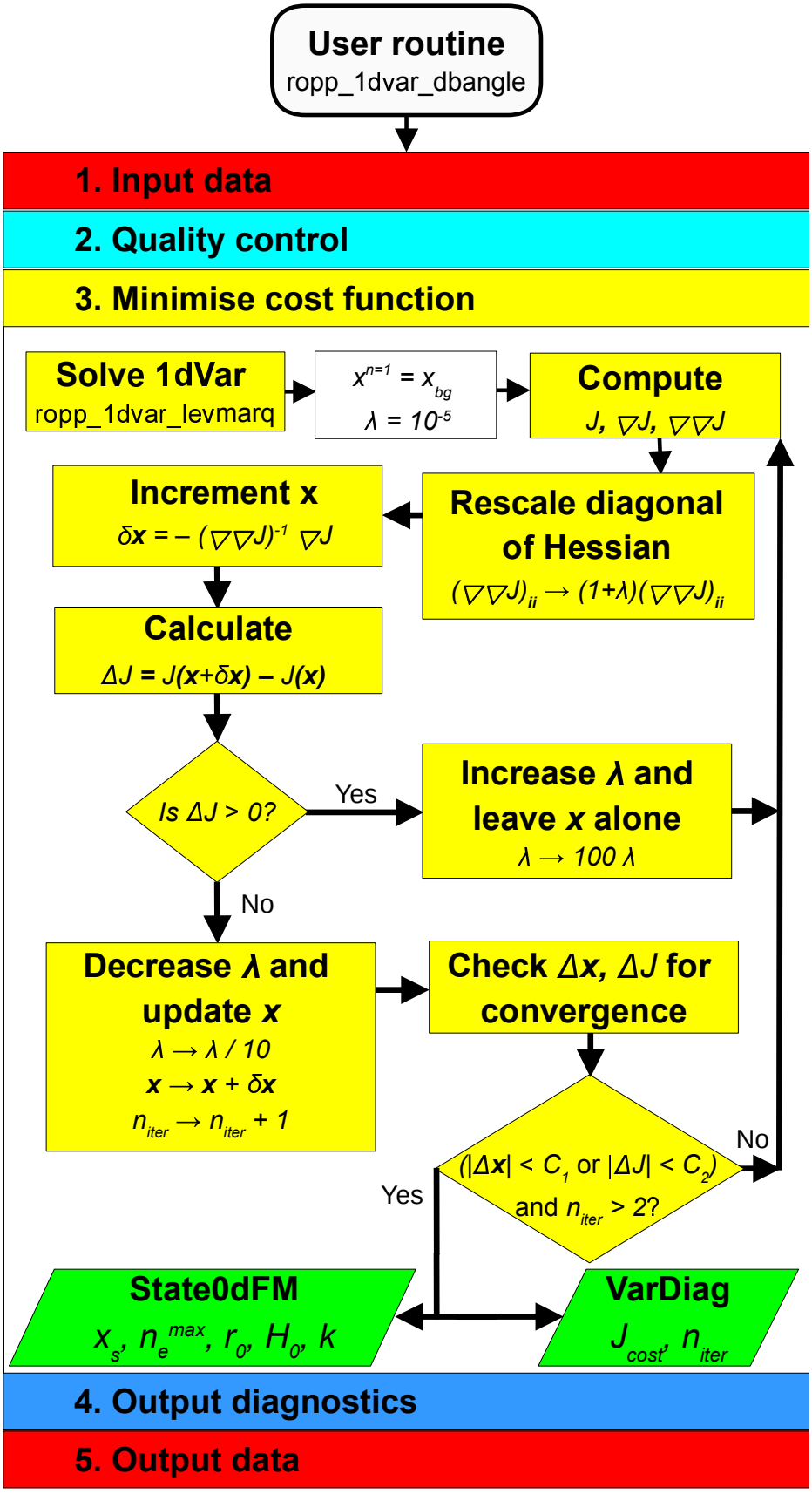

Figure 6.5: Flow chart illustrating the calling tree of the LevMarq minimisation step of ROPP 1D–Var to retrieve ionospheric profiles from observed differenced bending angle profiles and input background model data.

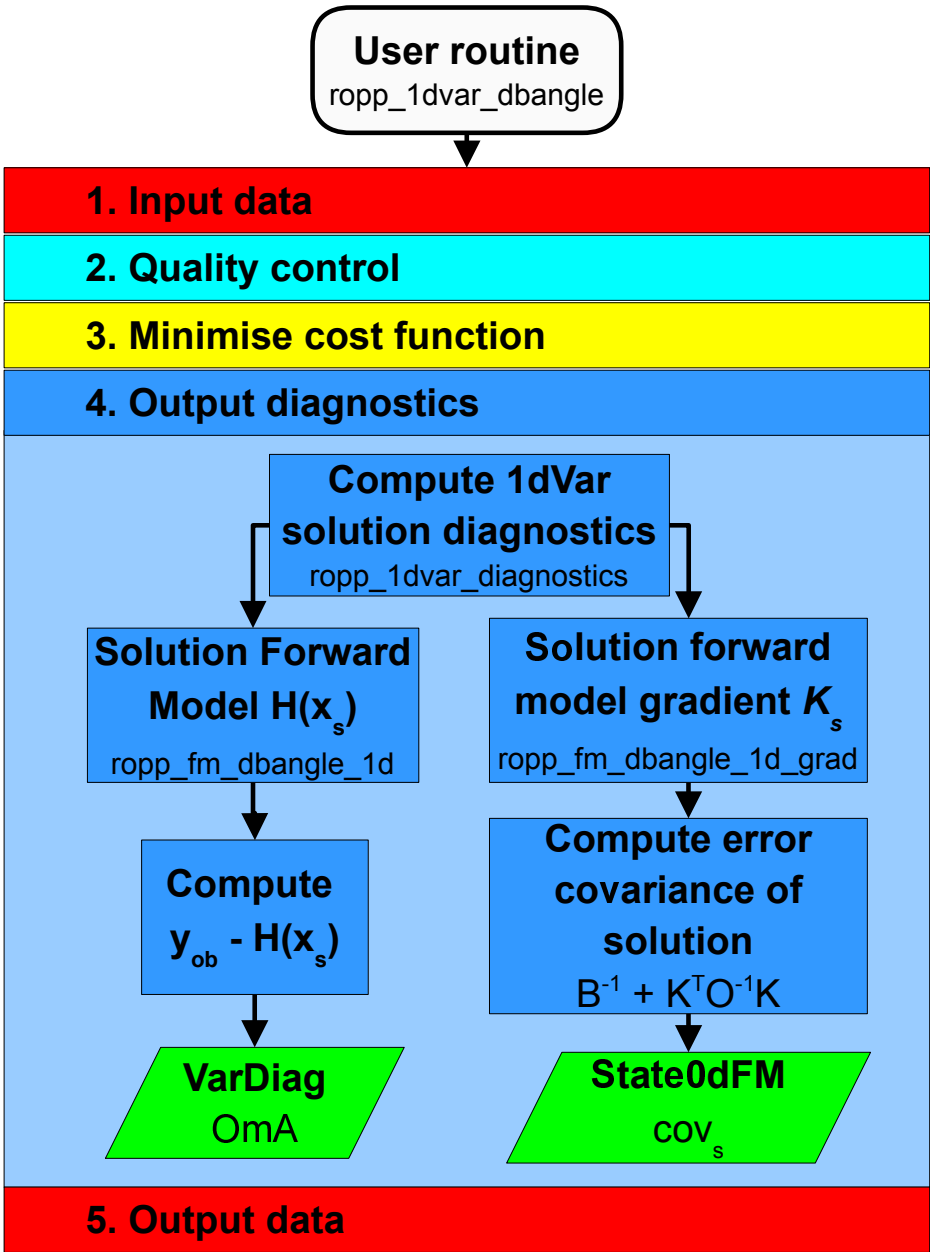

Figure 6.6: Flow chart illustrating the calling tree of the output diagnostics step of ROPP 1D–Var to retrieve ionospheric parameters from observed bending angle profiles and input background model data.

7 Including non-ideal gas effects

7.1 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).

Details of the forward model calculation can be found in Sec 6 of the ROPP FM User Guide (2021).

7.2 Running ROPP 1D–Var routines with non-ideal effects

The ROPP 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 1D–Var refractivity retrieval tool `ropp_1dvar_refrac` can be run with non-ideal gas compressibility factors in the forward model by using the command:

```
ropp_1dvar_refrac <inputdatafiles> -comp -o <outputfile>
```

where `<inputdatafiles>` is the set of input files described in Secs 4 and 5, and `<outputfile>` will contain the retrieved state vector variables.

The test scripts

- `ropp_1dvar/test/test_1dvar.sh`
- `ropp_1dvar/test/test_1dvar_COSMIC_refrac.sh`
- `ropp_1dvar/test/test_1dvar_COSMIC_bangle.sh`
- `ropp_1dvar/test/test_1dvar_GRAS_refrac.sh`
- `ropp_1dvar/test/test_1dvar_GRAS_bangle.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.

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) Forward model module User Guide, SAF/ROM/METO/UG/ROPP/006, Version 11.0, 2021.

Thayer, G. D., An improved equation for the refractive index of air, *Radio Sci.*, *9*, 803–807, 1974.

8 Modelling L1 and L2 bending angles

8.1 Background

ROPP has the facility to forward model L1 and L2 bending angles, instead of the ionospherically corrected neutral bending angle which is used elsewhere throughout the `ropp_fm` and `ropp_1dvar` modules. It does this by assuming a simple model ionosphere comprising a single Chapman layer (Chapman, 1931).

This facility allows the user to:

- Interpolate or extrapolate L1 and L2 using a simple physical model of the difference between them;
- Examine the sensitivity of L1 and L2 bending angles to variations in ionospheric parameters;
- Make 1D–Var retrievals using the less heavily processed L1 and L2 bending angles directly, rather than the ionospherically corrected neutral bending angle.

Details of the forward model calculation can be found in Sec 7 of the ROPP FM User Guide (2021).

8.2 ropp_1dvar module

8.2.1 Implementation

The `roppp_1dvar_bangle` 1D–Var retrieval tool can carry out retrievals using the L1 and L2 bending angles by calling it with the `-direct_ion` option, thus:

```
ropp_1dvar_bangle  -direct_ion  -y obs_input_file.nc  -b bgr_input_file.nc \
                   --bg-corr bgr_cov_file.nc  --obs-corr obs_cov_file.nc   \
                   -c config_file.cf  -o anl_output_file.nc
```

The same extension of the state vector and concatenation of the observation vector as described in Sec 7.3 of ROPP FM UG apply, and again these manipulations are (largely) hidden from the user. Thus, in the above:

- **obs_input_file.nc** obviously needs to include (valid) `bangle_L1` and `bangle_L2` bending angles, as well their errors `bangle_L1_sigma` and `bangle_L2_sigma`, if `obs_covar_method` (which is set in `config_file.cf`) equals 'VSFC' or 'VSDC'. If, on the other hand, `obs_covar_method` equals 'FSFC', so that the observation errors are to be taken from the observation covariance file `obs_cov_file.nc`, then these errors need to be the *neutral* bending angle errors (as found in the standard observation

error covarance files). In this case, the L1 and L2 errors are set equal to the neutral errors, 'floored'[1] by the parameters `sigma_L1_min` and `sigma_L2_min`, which are specified in `ropp_1dvar_iono_repack_bangle.f90`, and are currently set at:

```
REAL(wp), PARAMETER  :: sigma_L1_min=10.0e-6_wp ! rad
REAL(wp), PARAMETER  :: sigma_L2_min=30.0e-6_wp ! rad
```

This 'flooring' is simply intended to specify reasonable errors on the L1 and L2 bending angles, which are of course much larger than the corresponding neutral bending angle at height (eg see Fig 7.2 of ROPP FM UG). A simple percentage error would probably work just as well. The 3:1 ratio in the L2:L1 errors has some justification for the GRAS instrument (Marquardt et al., 2013), for which L2 is certainly noisier than L1. *The L1 and L2 bending angle error specifications in the ROPP 1D–Var module is very rudimentary, and is subject to change in later releases.*

L1 and L2 bending angles have generally been found to be less 'clean' than their neutral counterparts. As a result, it has been found necessary to introduce some extra quality control for these bending angles, which is carried out by `ropp_qc_ion_bangle.f90`. At present, this extra checking amounts to omitting (by setting their weight to zero) all bending angles for which $|\mathrm{d}\log\alpha/\mathrm{d}a|$ exceeds 10^{-3} m^{-1} at both adjacent points; this device therefore removes isolated 'spikes' in the data. L1 and L2 bending angles are treated separately. L1 and L2 are of course also subjected to the usual bending angle quality control checks, as described in Sec 5.6.

- **bgr_input_file.nc** should hold the background Chapman layer parameters `Ne_max`, `H_peak` and `H_width` and their (1-sigma) errors `Ne_max_sigma`, `H_peak_sigma` and `H_width_sigma`; otherwise they will default to the values specified in `ropp_fm/iono/ropp_fm_iono_set_default.f90`, as described in Sec 7.3 of ROPP FM UG.

  In practice, however, it has proved beneficial (in the sense of reducing the number of iterations and improving the fit of the final analysis) to initialise n_e^{max} by using the (averaged) L2 – L1 data directly. It follows from Eqn (7.8) of ROPP FM UG that, approximately, as H tends to zero,

$$n_e^{max} \propto \overline{\alpha_2 - \alpha_1} \tag{8.1}$$

  where, it can be shown, the proportionality constant is approximately 1.6×10^{16} m^{-3} if we assume $r_0 - a \sim 250$ km. This calculation is carried out in `ropp_qc_iono_bg.f90`, and the modified background n_e^{max} is written to stdout. Note that the average difference in the bending angles in Eqn (8.1) of ROPP FM UG is carried out over all impact heights greater than a threshold `min_fit_height`, which is currently set at 30 km in the routine.

- **bgr_cov_file.nc** should be the usual neutral background error covariance file for the model background field being used. `ropp_1dvar_iono_repack_bg.f90` extends the covariance structures (`bg%cov%d` and `bg%cov%f`) to include the three ionospheric parameters. We therefore end up with a $\mathbf{B}$-matrix of size $(2N_{\mathrm{model}}+1)+3 = 2N_{\mathrm{model}}+4$. It is assumed that n_e^{max}, r_0 and H are uncorrelated

[1] ie, $\sigma_{L1} = \max(\sigma_{L1}, \sigma_{L1}^{\mathrm{min}})$, and similarly for σ_{L2}.

with each other and with all other background fields; this is why Eqn (7.8) of ROPP FM UG is expressed in terms of n_e^{max} rather than TEC, which is more likely to be correlated with H.

- `obs_cov_file.nc` is the usual neutral observation error covariance matrix. The routine `ropp_1dvar_iono_repack_bangle.f90` manipulates the covariance structures `obs%cov%d` and `obs%cov%f` so that they represent a $\mathbf{O}$-matrix of size $2N_{\rm obs}$. This matrix is assumed to be diagonal, as usual for bending angle errors. *The correlation between α_1 and α_2, directly precluded by this assumption, may be introduced in a future release of ROPP.*

- **config_file.cf** is the usual configuration file. The usual 1D–Var choices should work, *although* `ropp_1dvar_bangle` *has only been tested thoroughly with the options coded in the configuration file* `ecmwf_bangle_1dvar_iono.cf` *(which is included in the ROPP distribution)*. This file differs from the 'standard' ECMWF configuration file, `ecmwf_bangle_1dvar.cf`, in the following ways:

  - `bg_covar_method` is 'FSFC' rather than 'VSFC', which prevents users from having to add errors on n_e^{max}, r_0 and H to their background files (instead, the code uses the default values specified above);
  - `obs_covar_method` is 'FSFC' rather than 'VSDC', which prevents users from having to add errors on α_1, α_2 to their observation files (instead, the code derives these from the neutral errors as described above);
  - `max_1dvar_height` is set to 80 km rather than 50 km, since the L1 and L2 bending angles are often very similar up to 40–50 km (see Fig 7.2 of ROPP FM UG for example), and the minimiser will struggle to converge if the observations are insensitive to elements (in this case, the ionospheric components) of the state vector;
  - `minropp_method` is set to 'LevMarq' rather than 'minROPP', since the former has been found to be more robust (but slower) in a range of tests of the `-direct_ion` option. ('minROPP' is available for those users who prefer it, following the introduction of some rescaling of the $\mathbf{B}$-matrix (which now has elements varying by as much as 10^{28} as a result of the inclusion of n_e^{max} in the state vector) prior to preconditioning. Again, this manipulation is hidden from the user.)

- **anl_output_file.nc** is the output file containing the retrieved ionospheric parameters (as well as the usual temperatures etc), and the forward modelled L1 and L2 (and neutral) bending angles. If extra diagnostics have been requested (ie `ropp_1dvar_bangle` has been called with the '-d' option), then the weights given to the α_1 and α_2 observations will be recorded. This can be useful in discovering 'rogue' observations that should have been excluded from the retrieval but still crept in.)

Once the revised observation and state vectors and their covariance matrices have been set up in this way, the minimisation of the cost function proceeds as before (see Sec 5.7).

8.2.2 Testing

The 'make test' step, as applied when building `ropp_1dvar` with `buildpack`, has been extended to test the `-direct_ion` functionality by means of one new test:

- **test_1dvar_iono.sh**, which compares the retrieved ionospheric parameters n_e^{max}, r_0 and H from a single GRAS/ECMWF profile, with those held in a reference file, which is included in the ROPP distribution. A 'PASS/FAIL' message is written to stdout saying whether these numbers agree or differ significantly.

8.2.3 Results

As an example, Fig 8.1 shows the observed and retrieved neutral, L1, L2 and L2 – L1 bending angles for a GRAS occultation and co-located ECMWF background fields from May 2009. The retrieved ionospheric parameters are: $n_e^{max} = 113 \times 10^9$ m^{-3} (cf background $= 300 \times 10^9$ m^{-3}); $r_0 = 267$ km (cf background $=$ 300 km); and $H = 67$ km (cf background $= 75$ km). Convergence occurs after 6 iterations (cf 4 for the usual neutral retrieval in this case). Not unusally, L2 is absent below about 15km. In addition, L1 is very noisy at the top and bottom of the profile. Despite these problems, and partly as a result of the extra quality control applied to the individual signals, reasonable L1 and L2 retrievals are obtained, which have a well defined difference over the whole profile. (The small 'shoulder' in the bending angles at about 5km is also present in the neutral retrieval, and may be associated with the deep boundary layer in this profile. The forward modelled background also has it.)

Fig 8.2 shows the background and retrieved temperature, specific humidty, pressure and (implied) electron density for the same occultation and background. The first three are comparable to their neutral retrieval equivalents (not shown), while the last shows that the model electron density is something like 50% smaller than in the background. This large change is no doubt a result of the generous default uncertainties on the ionospheric parameters (see Sec 7.3 of ROPP FM UG). The retrieved n_e^{max}, r_0 and H are of course not likely to be reliable measures of the true ionosphere, and nor are they intended to be. It is better to view them as coefficients in a physically based fitting procedure — namely, the minimisation of the cost function.

8.2.4 Summary

ROPP can carry out 1D–Var retrievals using L1 and L2 bending angles directly, by assuming a single Chapman layer model ionosphere in the forward model. This allows information from one of the signals to be used where the other one is missing or invalid. This is in contrast to traditional bending angle retrievals, based on neutral bending angles, which require both L1 and L2 to be present.

References

Chapman, S., The absorption and dissociative or ionizing effect of monochromatic radiation of an atmosphere on a rotating earth, *Proc. Phys. Soc.*, *43*, 483–501, 1931.

Marquardt, C., von Engeln, A., Andres, Y., and Yoon, Y., Single frequency radio occultation retrievals, EUM/TSS/TEN/13/707742, Issue 2, 2013.

ROM SAF, The Radio Occultation Processing Package (ROPP) Forward model module User Guide, SAF/ROM/METO/UG/ROPP/006, Version 11.0, 2021.

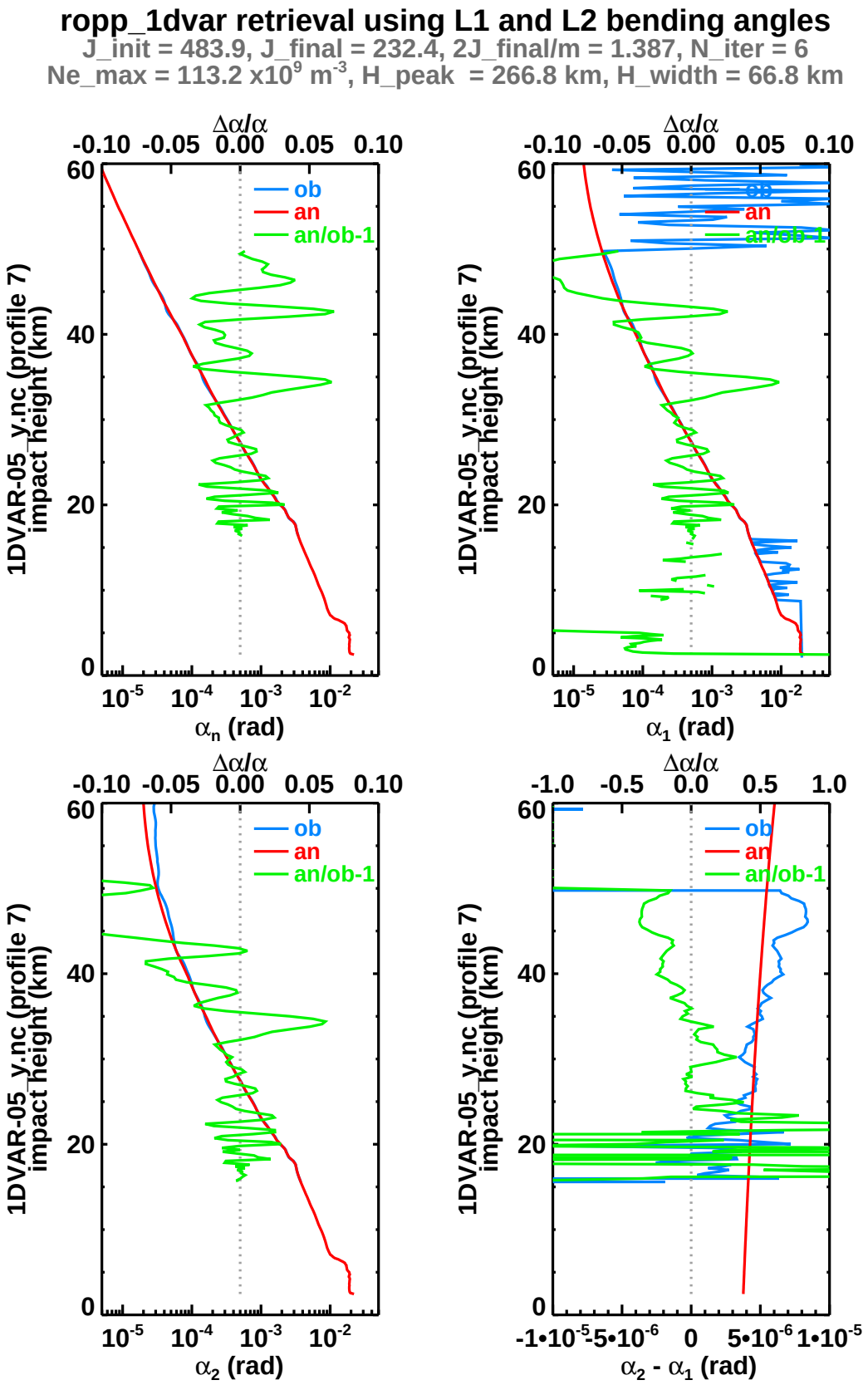

Figure 8.1: Example observed (blue) and retrieved (red) neutral, L1, L2 and L2 – L1 bending angles, and the fractional differences between them (green), for a GRAS/ECMWF observation/background pair from May 2009.

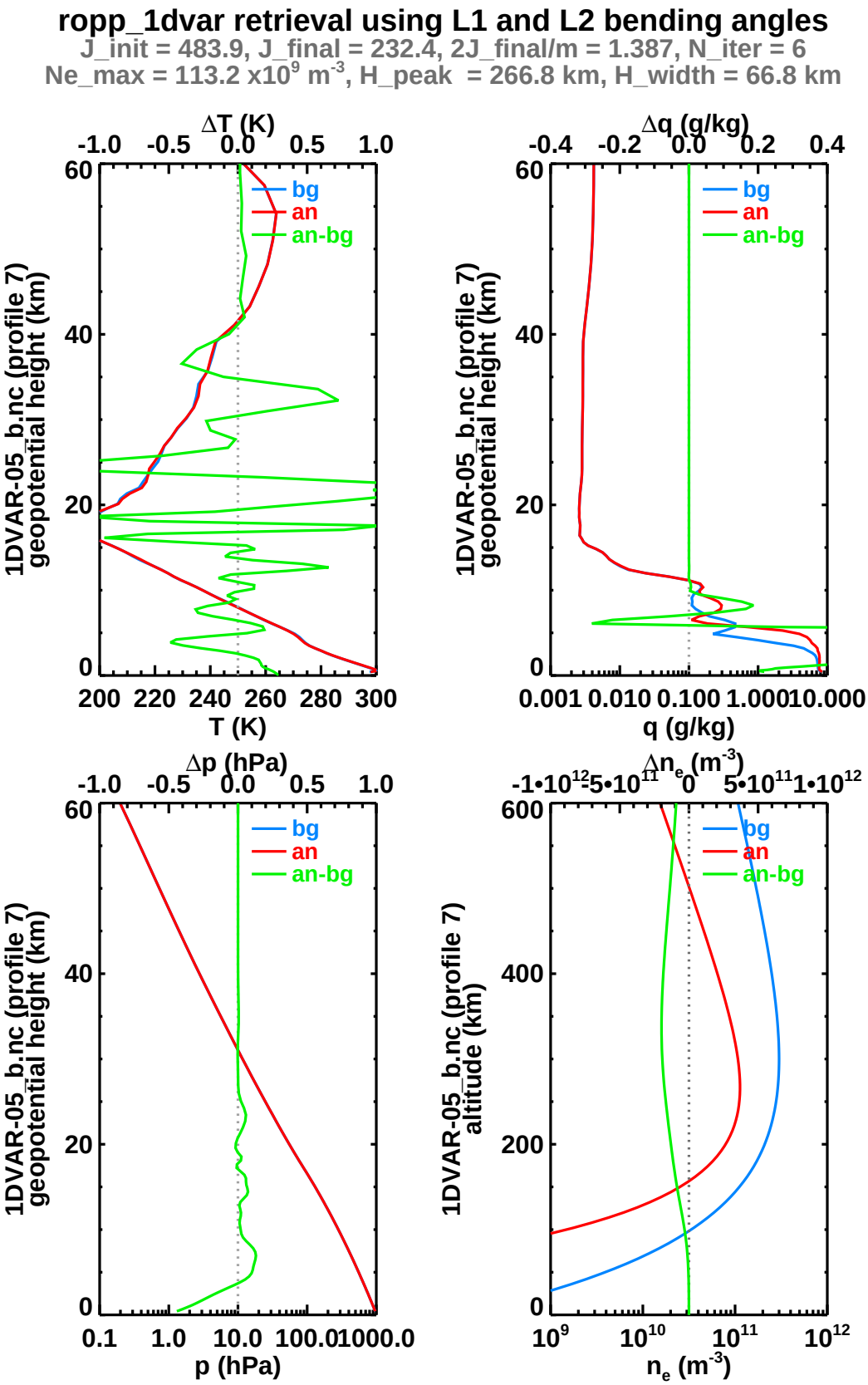

Figure 8.2: Example background (blue) and retrieved (red) temperature, specific humidity, pressure and (implied) electron density, and the differences between them (green), for a GRAS/ECMWF observation/background pair from May 2009.

A Installing and using ROPP

A.1 Software requirements

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

A.2 Software release notes

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

A.3 Third-party packages

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

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

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

A.3.1 NetCDF (optional in principle)

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

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

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

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

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

and

http://www.zlib.net/

respectively.

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

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

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

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

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

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

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

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

A.3.2 BUFR (optional)

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

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

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

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

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

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

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

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

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

or

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

while the ECMWF BUFR library would be be built with

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

and the ECMWF ecCodes library would be be built with

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

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

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

or for the ECMWF BUFR library or ecCodes library:

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

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

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

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

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

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

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

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

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

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

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

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

A.3.4 SOFA (optional)

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

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

http://www.iausofa.org/

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

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

A.3.5 RoboDoc (optional)

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

A.3.6 autoconf and automake (optional)

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

A.4 BUILDPACK script

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

```
> buildpack -h
```

In general, to build and install a package,

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

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

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

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

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

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

Or, even more quickly:

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

A.5 Building and installing ROPP manually

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

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

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

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

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

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

A.5.1 Unpacking

Once the required third-party software packages have been installed successfully, the ROPP packages can be installed. The complete ROPP package and individual modules are distributed as gzipped tar (`.tar.gz`) files. The complete package file name consists of the version name (e.g. `ropp-11.0.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. 11.0), as in `ropp_utils-11.0.tar.gz`. If GNU tar is available (as on Linux systems), gzipped tar files can be unzipped with

```
> tar -xvzf ropp-11.0.tar.gz
```

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

```
> gunzip -c ropp-11.0.tar.gz | tar -xv
```

In all cases, a new subdirectory named (in the above example) `ropp-11.0` will be created which contains the source code of the complete package.

A.5.2 Configuring

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

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

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

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

For most compilers, this means that the two paths to the header and module files need to be specified via the proper compiler options — usually via the `-I` option. The linker also needs to know where libraries are

located; on most Unix systems, this can be achieved by specifying the -L option at link time. Users are referred to the examples provided in the configure package for further details.

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

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

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

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

A.5.3 Compiling

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

```
> make
```

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

Currently supported Fortran compilers include (on Linux unless otherwise stated): Intel's ifort (v16 and v17); NAG's nagfor (v6.1); Portland Group's pgf95 (v16); GNU gfortran (v4.8.5); Cray's ftn (v8.3.4). For the authoritative list please refer to the ROPP Release Notes and README files in each sub-package.

A.5.4 Installing

After building the software successfully, the command

```
> make install
```

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

A.5.5 Cleaning up

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

```
> make clean
```

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

```
> make distclean
```

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

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

```
> make uninstall
```

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

A.6 Linking

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

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

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

A.7 Testing

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

A.7.1 ropp_utils

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

A.7.2 ropp_io

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

```
> make test_netcdf
```

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

A second test can be run with

```
> make test_ropp
```

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

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

```
> make test_bufr
```

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

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

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

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

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

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

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

Finally, the command

```
> make test
```

will run all of the above described tests.

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

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

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

A.7.3 ropp_pp

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

```
> make test
```

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

A.7.4 ropp_apps

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

```
> make test
```

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

A.7.5 ropp_fm

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

```
> make test
```

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

A.7.6 ropp_1dvar

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

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

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

```
> make test
```

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

A.8 Troubleshooting

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

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

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

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

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

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

B ropp_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.

As Fig 2.1 shows, ropp_utils, ropp_io and ropp_fm are required for all ropp_1dvar tools. So is the netCDF library.

tools/

- **ropp_1dvar_bangle**
- **ropp_1dvar_refrac**
- **ropp_1dvar_dbangle**

tests/

- **test_1dvar.sh**
- **test_1dvar_COSMIC_bangle.sh**
- **test_1dvar_COSMIC_refrac.sh**
- **test_1dvar_GRAS_bangle.sh**
- **test_1dvar_GRAS_refrac.sh**
- **test_1dvar_iono.sh**
- **test_1dvar_IEEC_dbangle.sh**
- ropp_1dvar_compare.f90
- ropp_1dvar_summary.f90

- 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_1dvar_covar_bg.f90
- ropp_1dvar_covar_bangle.f90
- ropp_1dvar_covar_refrac.f90
- ropp_1dvar_diag2roprof.f90

qc/

- ropp_qc.f90
- ropp_qc_OmB.f90
- ropp_qc_bgqc.f90
- ropp_qc_cutoff.f90
- ropp_qc_genqc.f90
- ropp_qc_pge.f90
- ropp_qc_ion_bangle.f90
- ropp_qc_ion_bg.f90

math/precon/

- ropp_control2state.f90
- ropp_control2state_ad.f90
- ropp_state2control.f90

math/matrix/

- matrix.f90
- matrix_assign.f90
- matrix_bm2full.f90

```
matrix_bm2full_alloc.f90
matrix_delete.f90
matrix_full2bm.f90
matrix_full2bm_alloc.f90
matrix_full2pp.f90
matrix_full2pp_alloc.f90
matrix_invert.f90
matrix_operators.f90
matrix_pp2full.f90
matrix_pp2full_alloc.f90
matrix_pp2full_subset.f90
matrix_solve.f90
matrix_sqrt.f90
matrix_svd.f90
matrix_toast.f90
matrix_types.f90
iono/
    ropp_1dvar_iono.f90
    ropp_1dvar_iono_repack_bangle.f90
    ropp_1dvar_iono_repack_bg.f90
    ropp_1dvar_iono_unpack_bangle.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 4.2 and 5.2 for details.

config/

A collection of 1D–Var configuration files (text format), which users may find helpful for defining input to `ropp_1dvar` tools. See Sections 4.2, 5.2 and 8.2 for details.

C ROPP extra diagnostic data

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

C.1 ropp_io_addvar

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

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

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

C.2 PPDiag

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

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

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

C.3 ropp_fm_bg2ro

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

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

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

C.4 VarDiag

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

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

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

D ROPP user documentation

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

Table D.1: ROPP user documentation

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

Table D.2: GRAS SAF Reports

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

Table D.3: ROM SAF Reports

Title	Reference	Description
Testing reprocessed GPS radio occultation datasets in a reanalysis system	SAF/ROM/METO/REP/RSR/029	Impact of reprocessed RO data on reanalyses
A first look at the feasibility of assimilating single and dual frequency bending angles	SAF/ROM/METO/REP/RSR/030	Single and dual frequency assimilation
Sensitivity of some RO measurements to the shape of the ionospheric electron density profile	SAF/ROM/METO/REP/RSR/031	Ionospheric shape sensitivity
An initial assessment of the quality of RO data from KOMPSAT-5	SAF/ROM/METO/REP/RSR/032	KOMPSAT-5 quality assessment
Some science changes in ROPP-9.1	SAF/ROM/METO/REP/RSR/033	ROPP-9.1 science
An initial assessment of the quality of RO data from Metop-C	SAF/ROM/METO/REP/RSR/034	Metop-C quality assessment
An initial assessment of the quality of RO data from FY-3D	SAF/ROM/METO/REP/RSR/035	FY-3D quality assessment
An initial assessment of the quality of RO data from PAZ	SAF/ROM/METO/REP/RSR/036	PAZ quality assessment
6th ROM SAF User Workshop	SAF/ROM/METO/REP/RSR/037	ROM SAF–IROWG 2019 report
An initial assessment of the quality of RO data from COSMIC-2	SAF/ROM/METO/REP/RSR/038	COSMIC-2 quality assessment
Impacts of RO mission differences on trends in multi-mission data records	SAF/ROM/METO/REP/RSR/039	RO mission CDR differences
Anomalous GRAS radio occultations	SAF/ROM/METO/REP/RSR/040	Anomalous occultations
Assessment of sensitivity of the ROM SAF 1D-Var solutions to various error covariance choices	SAF/ROM/METO/REP/RSR/041	Sensitivity to error covariances
A one-dimensional variational ionospheric retrieval for truncated GNSS Radio Occultation measurements	SAF/ROM/METO/REP/RSR/042	Ionospheric 1dvar

Table D.4: ROM SAF Reports (continued)

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

Table D.5: Applicable documents

E Authors

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

ROPP Authors

Name	Current institute	Contribution
Carlo Buontempo	Met Office	Savitzky-Golay thinner code.
Chris Burrows	ECMWF	2nd ROPP Test Manager. Test folder developments, improved FM vertical interpolation scheme.
Ian Culverwell	Met Office	2nd ROPP Development Manager. Documentation, testing, consolidation, IO development, GRIB2 reader, implementation of tropopause height diagnostics and planetary boundary layer height diagnostics, forward modelling of L1 and L2 bending angles, implementation of VaryChap f2–f2 FM and 1DVAR code
Axel von Engeln	EUMETSAT	Author of original Test Folder system and of EUMETSAT-formatted RO data reader.
Hans Gleisner	DMI	Elements of ropp_pp, prototype GRIB2 reader, ec{i/f}2ec{i/f} code.
Michael Gorbunov	Russian Academy of Sciences	Original pre-processor code.
Sean Healy	ECMWF	Original 1D FM code, 2D FM operator code, introduction of compressibility factors, improved FM vertical interpolation scheme, forward modelling of L1 and L2 bending angles, 1D and 2D wave optics propagators, prototype VaryChap f2–f2 FM and 1DVAR code.
Helge Jønch-Sørensen	DMI	BAROCLIM code.
Kjartan Kinch	DMI	Elements of ropp_pp.
Kent Bækgaard Lauritsen	DMI	Code reviews; liaison with EUMETSAT (licences, beta tester contracts).
Huw Lewis	Met Office	1st ROPP Development Manager, FM and 1D–VAR extensions. PP module.
Owen Lewis	Met Office	BUFR developments.
Christian Marquardt	EUMETSAT	Author of majority of ROPP-1 code in UTILS, IO, FM and 1DVAR modules, and much personal, pre-existing software.
Dave Offiler	Met Office	ROPP Project Manager, IO application code and IO extensions, BUFR format/template.
Michael Rennie	ECMWF	1st ROPP Test Manager. Test folder developments.
Barbara Scherllin-Pirscher	Wegener Center	BAROCLIM (3) dataset for statistical optimisation.
Torsten Schmidt	GFZ	Guidance on tropopause height diagnostics.
Stig Syndergaard	DMI	Original spectral version of MSIS model (expansion in spherical harmonics and Chebychev polynomials), PP module developments.
Francis Warrick	Met Office	Implementation of ecCodes lib; ROPP devt and testing.
Feiqin Xie	Texas A & M	Suggested boundary layer height diagnostic algorithms.

Table E.1: Contributors to ROPP

F Copyrights

The majority of ROPP code is

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

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

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.

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