NWP SAF

Satellite Application Facility
for Numerical Weather Prediction

Document NWPSAF-TV-002

Version 2.2

29-09-2014

SDP Test Report

KNMI Scatterometer Team

Jur Vogelzang, Anton Verhoef, Jeroen Verspeek, Jos de Kloe and Ad Stoffelen

KNMI, De Bilt, The Netherlands

SDP Test Report

KNMI Scatterometer Team

Jur Vogelzang, Anton Verhoef, Jeroen Verspeek, Jos de Kloe,
and Ad Stoffelen

KNMI, De Bilt, The Netherlands

Change record			
Version	Date	Author / changed by	Remarks
0.0	Oct 2004	Hans Bonekamp	First draft
1.0	May 2005	Hans Bonekamp	Beta release
1.1	21-12-2005	Jur Vogelzang	Beta release
1.2	27-03-2006	Jur Vogelzang	First public release
1.3	06-04-2007	Jur Vogelzang	Revised 2DVAR in genscat
1.4	26-06-2007	Jur Vogelzang	Module *Compiler_Features* added
2.0	27-10-2008	Jur Vogelzang	Extended Single Observation Test
2.2	29-09-2014	Anton Verhoef	Tests repeated for SDP version 2.2.

Contents

Preface

Preface to version 1.0

This document is the test report for the SDP program. It is set up according to the guidelines of the NWPSAF, see the document NWPSAF Development Procedures for Software Deliverables. Parts of the SDP development are in fact genscat developments. The tests for genscat modules are done in separate chapters.

Hans Bonekamp, October 2004

Preface to version 1.1

I made a number of adaptations and extensions to the original text, but left the underlying structure of the document unchanged. The reader is kindly invited to give his comments in order to improve future versions of this document.

Jur Vogelzang, September 2005

Preface to version 1.2

The tests have been extended with a test of routine LBFGS. Again, the reader is kindly invited to give his comments in order to improve future versions of this document.

Jur Vogelzang, March 2006

Preface to version 1.3

The two dimensional variational ambiguity removal (2DVAR) method has been critically revised in 2006/2007. A number of serious bugs has been removed, and two new tests were developed to verify the new 2DVAR: the single observation solution test and the single observation analysis test. Module *CompLinAlg* has been removed, and so has its test. An extra FFT test is included.

Jur Vogelzang, April 2007

Preface to version 1.4

A new genscat support module named *Compiler_Features* has been added in order to use command line arguments under all supported Fortran compilers, since the functions *iargc* and *getarg* are not standard Fortran.

Jur Vogelzang, June 2007

Preface to version 2.0

SDP 2.0 contains several new features:

- Outer swath processing;
- Improved MLE flag handling;
- Flexible 2DVAR batch grid definition

The last point, flexible 2DVAR batch grid definition, is possible by using a new mixed-radix FFT routine originally written by R.C. Singleton and available via www.netlib.org. The FFT test in 3.9 and the single observation tests in 3.5 and 4.1 have been extended in order to include different 2DVAR batch grid definitions.

Finally, the SDP tests in chapter 4 were updated.

Jur Vogelzang, October 2008

Preface to version 2.2

The Test Report was updated for SDP 2.2. SDP versions 2.1 and 2.2 contain several new features:

- Capability to process winds in the outer swath, where only vertically polarized data are available.
- New formulation of the MLE to make wind processing independent of the input data winds.
- Implementation of the Bayesian ice screening algorithm.
- Implementation of the NSCAT-4 geophysical model function.
- Improvement of the Quality Control.

Anton Verhoef, September 2014

Chapter 1

Introduction

1.1 The SDP program

The SeaWinds Data Processor (SDP) is a generator of near real time surface winds based on Normalized Radar Cross Section (σ_0) data from the SeaWinds scatterometer instruments on either the QuikSCAT or the Adeos-II (Midori-II) satellites. The main application of SDP is to form the core of an Observation Operator for SeaWinds Scatterometer Data within an operation Numerical Weather Prediction System. Details on the SDP program can be found in [*Vogelzang et. al*, 2014].

SDP is developed within the NWP SAF project as Fortran 90 code which can be run in an operational setting. The development of the code has followed the procedures specified for the NWP SAF.

Special attention has been paid on robustness and readability of the code. The majority of the code follows the European standards for writing and documenting exchangeable Fortran 90 code.

Table 1.1 contains an updated version of the release list for the SeaWinds Data Processor (SDP). Versions 1.0 and 1.1 were restricted to some selected beta users, but version 1.2 is the first public release.

Version	Date	Code owner	Remarks
1.0	October 2004	Hans Bonekamp	Release to beta testers
1.1	December 2005	Jur Vogelzang	Beta release
1.2	March 2006	Jur Vogelzang	First public release
1.3	September 2006	Jur Vogelzang	Minor update
1.4	March 2007	Jur Vogelzang	Important update (new 2DVAR)
1.5	June 2007	Jur Vogelzang	Important update (improved minimalization strategy)
2.0	November 2008	Jur Vogelzang	Major update (outer swath processing, improved MLE flag handling and flexible 2DVAR batch grid definition)
2.1	November 2010	Jur Vogelzang	Included sea ice screening algorithm
2.2	September 2014	Anton Verhoef and Jur Vogelzang	New default GMF, NWP Ocean Calibration, new Quality Control method

Table 1.1 SDP releases.

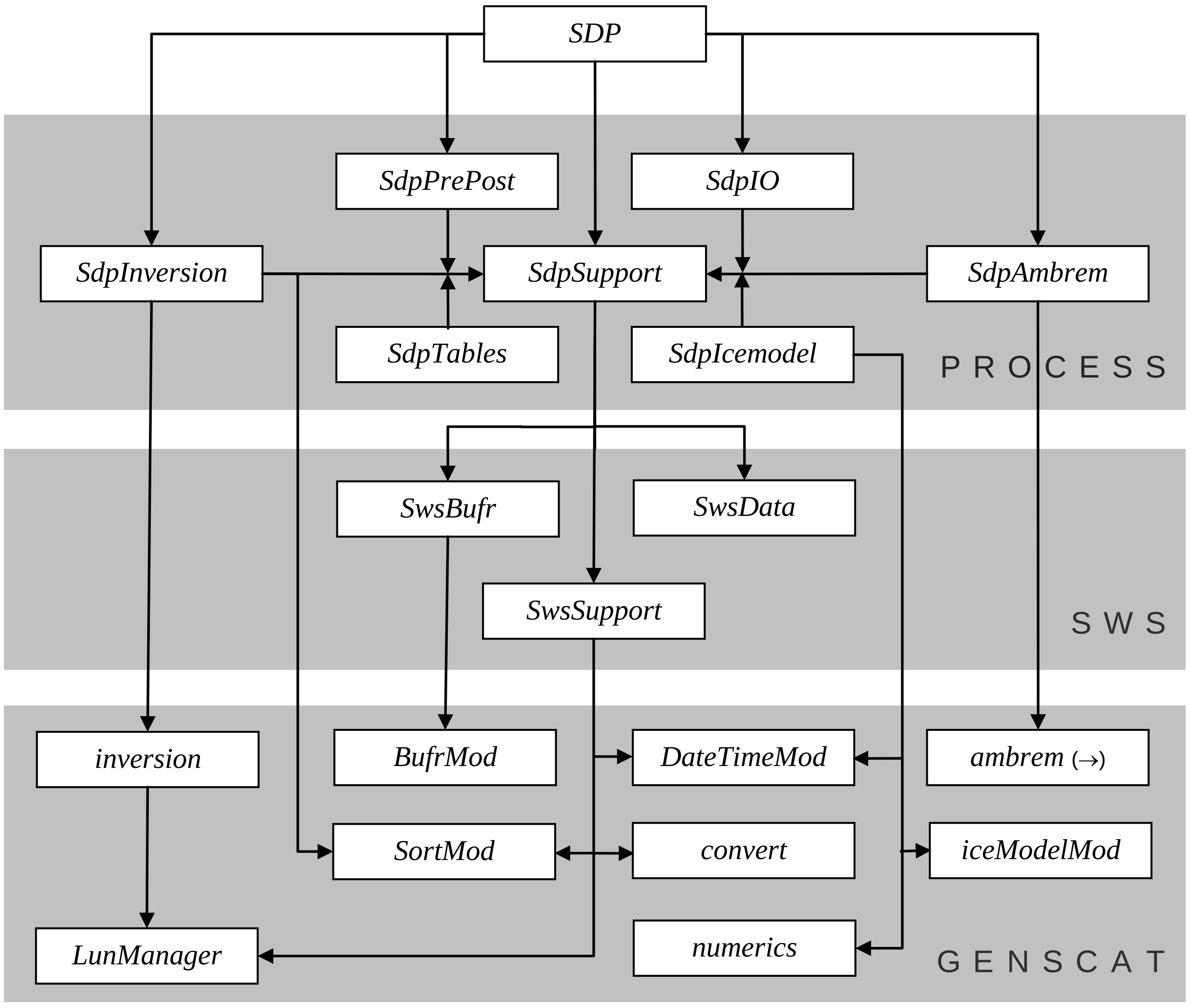

Figure 1.1 SDP Module layers and top level module dependencies. The dependencies for module *ambrem* are continued in figure 1.2.

SDP is a large and complicated program. Figures 1.1 and 1.2 give an overview of the module and layer structure of SDP. It is set up from three layers of software modules which are linked after individual compilation. The bottom layer, the genscat layer, consists of generic modules which are in no way dependent on any type of scatterometer instrument. These modules may therefore be tested independently of the SDP program. The software for Ambiguity Removal is quite complex and involves the modules shown in figure 1.2.

The middle layer, the SeaWinds (SWS) layer acts as an interface translating data from the top layer to generic genscat format and vice versa. Tests on this layer can thus be confined to reading and writing data and checking data integrity.

The top layer consists of the process modules. Tests may be concentrated here on the processing itself. The tests for the process module layer already support the integration tests.

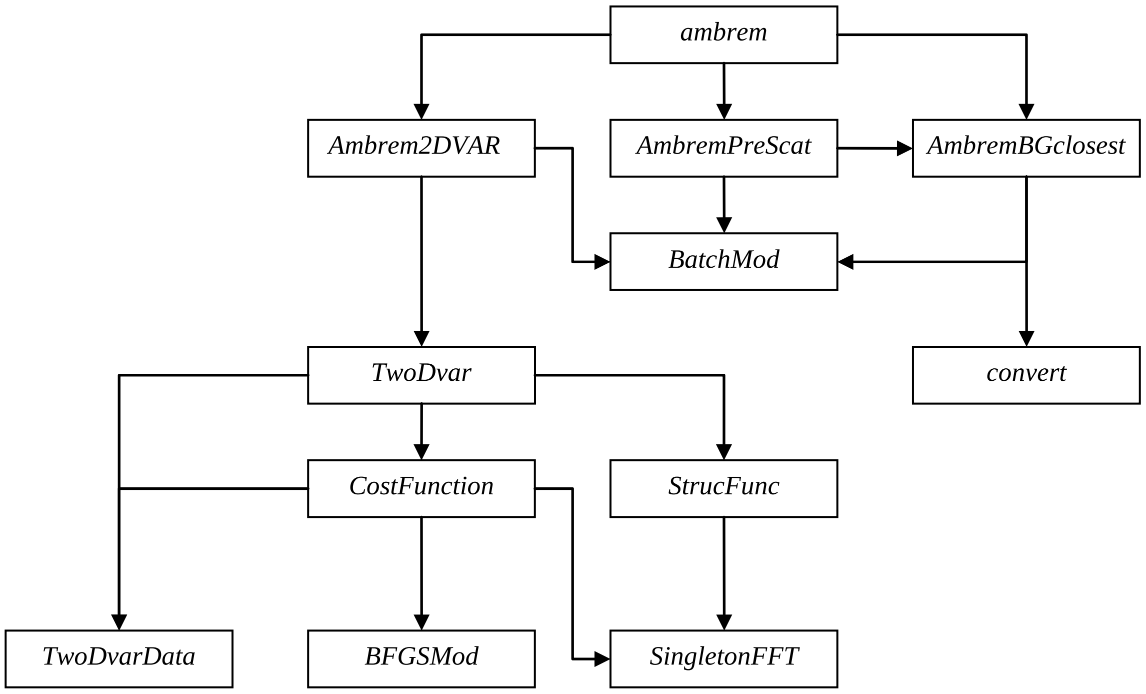

Figure 1.2 Interdependence of the modules for Ambiguity Removal. The direct connection from module *ambrem* to module *BatchMod* is not drawn.

1.2 SDP heritage

The SDP code is based on code developed for the ERS scatterometers, NSCAT scatterometer, and the simulations of the ESA Rotating Fan beam Scatterometer (RFSCAT). The common code of these projects is now consolidated in the genscat layer. QDP code has been inherited to form the SWS and process layers. In each development step, following from the heritage, the output of the code developments has been compared to the output of the original code (cf. Chapter 5).

Several developers work with and on SDP at KNMI, and even more with the genscat layer for ERS or ASCAT projects. Improvements to the code follow the test procedures as described in this document. The effort of maintaining a unique reference code greatly improves robustness and reliability of the code, i.e., sharing results and enjoying the benefits.

1.3 SDP test plan

This section describes the Test Plan of the SDP deliverable. Tests have been carried out in all stages of the development of SDP.

The Test Plan distinguishes:

1. method tests, see chapter 2;
2. module tests, see chapter 3;
3. integration tests, see chapter 4;
4. validation tests, see chapter 5;
5. portability tests, see chapter 6; and

6. user documentation tests, see chapter 7.

The inversion module is not tested for the SDP program, because such a test has already been made for the QDP development.

SDP contains several methods for Ambiguity Removal within module *ambrem* and its submodules, see figure 1.2. Only modules needed for the KNMI 2DVAR scheme for Ambiguity Removal are tested within this project.

Several tests are implemented in the Python scripting language (www.python.org). Python is public domain software available free of charge for nearly every platform and operating system. Compilation of SDP is done on several platforms (operating systems) and with different Fortran 90 compilers.

Chapter 2 describes the tests for selecting the minimalization method and for selecting the most precise method for calculating the wind vector cell orientations from the data in the BUFR files.

Chapter 3 contains the tests for a number of individual modules. In general, modules are tested with the associated test programs that are located in the folder containing the module under consideration. The output of the test programs is always the standard output (screen) which may be redirected to any test log file or to some output files which are stored in the associated folders. Some tests require modules that are not necessary for the operation of SDP and are therefore left out of the SDP 1.4 release. An important test is the single observation solution test checking large parts of modules *CostFunctio*n and *StrucFunc* in `genscat/ambrem/twodvar`. This test proved to be of crucial importance in getting 2DVAR correct.

Chapter 4 describes the single observation analysis, another important test for verifying the 2DVAR procedure, and some SDP test runs that are part of the test folder. Some of the resulting wind fields are shown.

Chapter 5 discusses the validation tests. SDP has been run in QDP mode, and the results of both programs have been compared for identical input. SDP gives the same results as QDP where it should, and performs better where expected.

Chapter 6 describes the portability tests. It contains an overview of platform/operating system and Fortran compilers for which SDP is supported.

Chapter 7 is devoted to testing the user documentation.

1.4 Test folder

The Test Folder of the SDP Program is located in subdirectory `sdp/tests`. This subdirectory contains an input file for SDP and three output files that are discussed in more detail in chapter 4. The scripts for executing these tests are located in directory `sdp/execs`. It is recommended to use these scripts (or a modified version) also for normal SDP operation, as the environment variables needed by SDP are set in these scripts.

As stated before, most test programs are located in the same directory as the module to be tested. See chapter 3 for detailed information.

Finally, some tests are available as Python scripts in directory `sdp/python`.

1.5 Conventions

Names of physical quantities (e.g., wind speed components *u* and *v*), modules (e.g. *BufrMod*), subroutines and identifiers are printed italic.

Names of directories and subdirectories (e.g. `genscat/support)`, files (e.g. `sdp.F90`), and commands (e.g. `sdp -f input`) are printed in Courier. When addressing software systems in general, the normal font is used (e.g. SDP, genscat).

Hyperlinks are printed in blue and underlined (e.g. www.knmi.nl/scatterometer).

1.6 Reference documents

Reference is made to the following documents that can be obtained from the NWPSAF web pages http://nwpsaf.eu/deliverables/scatterometer/index.html:

- Vogelzang, J., Verhoef, A., Verspeek, J., de Kloe, J. and Stoffelen, A., 2014,
  SDP User Manual and Reference Guide, version 2.2.
  Report NWPSAF-KN-UD-002, UKMO, UK.
- Vogelzang,, J., 2006,
  The orientation of Seawinds wind vector cells.
  Technical Report NWPSAF-KN-TR-003, UKMO,UK.
- Vogelzang, J., 2007,
  Two dimensional variational ambiguity removal (2DVAR).
  Technical Report NWPSAF-KN-TR-004, UKMO,UK.

Chapter 2

Method tests

This chapter describes tests on methods used in the SDP program. These tests do not pertain to software used in the SDP program, but use separate software to validate general numerical methods. These tests therefore require additional software, which is not part of the SDP release.

The processing chain in SDP contains three main methods: inversion, Quality Control (rain detection), and Ambiguity Removal. Inversion and Quality Control were tested in the framework of projects other than NWP SAF. Therefore this report contains only two method tests: for the selection of a minimization routine in section 2.1, and the single observation solution test in section 2.2. These methods are used in the KNMI 2DVAR Ambiguity Removal scheme.

The program for the single observation solution test, program *Test_SOS*, is included in directory `genscat/ambrem/twodvar`. It also tests some routines from modules *CostFunction* and *StrucFunc*. These tests are discussed in more detail in section 3.4.

2.1 Minimization routine

In the KNMI 2DVAR Ambiguity Removal scheme, the wind field is obtained by minimizing a cost function which contains terms depending on the measured scatterometer winds, their probability, and the NWP model winds contained in the BUFR input file. In QDP the minimization was performed by routine N1QN3 from INRIA, France. However, this routine is copyrighted and therefore not suitable for a public release.

A number of freeware routines employing gradient methods were tested with the simple test function

$$f(x) = \sum_{i=1}^{N} (x - i)^4 \quad . \tag{2.1}$$

This function has a minimum at (1,2,…,*N*) where *f* equals zero. The gradient is easily calculated. The following routines were tested:

- Routine LBFGS, a limited memory BFGS method by J. Nocedal, obtained from http://www.netlib.org/opt/, file `lbfgs_um.shar`.
- Routine L-BFGS-B, also a limited memory BFGS method, by C. Zhu and J. Nocedal, obtained from http://www.netlib.org/opt/, file `lbfgs_bcm.shar`.
- Routine dvmlm, a limited memory quasi-Newton method, by B.M. Averick, R.G. Carter, and J.J. More, obtained from ftp://info.mcs.anl.gov/pub/MINPACK-2/vmlm/.
- Routine PLIPU, a limited memory variable metric method by L. Luksan, C. Matonoha, and J. Vlcek, obtained from http://www.cs.cas.cz/~luksan/subroutines.html.

The routines were tested with the test function (2.1) for N = 2000. The search for the minimum started at x = (0,0, … ,0) with a cost function of $0.64{\cdot}10^{+16}$. Table 2.1 shows the time needed to find the minimum, the cost function at the minimum and the maximum deviation (in one dimension) from the true minimum.

Routine	**Time needed (seconds)**	**Cost function at minimum**	**Precission (worst case)**	**Remarks**
LBFGS	0.016	$0.73{\cdot}10^{-20}$	0.000002	Rank 3
L-BFGS-B	0.047	$0.50{\cdot}10^{-12}$	0.000165	Rank 3
dvmlm	0.016	$0.58{\cdot}10^{-15}$	0.000030	Rank 3
PLIPU	0.078	$0.63{\cdot}10^{-16}$	0.000070	Rank 1 with two point quadratic line search

Table 2.1 Results for the minimization routines.

Routines LBFGS, L-BFGS-B, and dvmlm have a variable rank. The best results in terms of time and accuracy were obtained with rank 3. Routine PLIPU has 32 basic modes of operation. The best results were obtained using a two point quadratic interpolation in the line search with unit correction parameter and rank 1.

Table 2.1 shows that routine LBFGS outperforms the other routines in terms of speed and accuracy. Its code is concise and relatively simple. Therefore this routine is selected for the minimization in SDP and other scatterometer wind processing packages in the NWP SAF.

2.2 Wind vector cell orientation

The first step in the two-dimensional ambiguity removal (2DVAR) procedure in SDP is to transform the zonal and meridional wind speeds (u,v) to transverse and longitudinal (with respect to the satellite orbit) components (t,l). The rotation angle, α, gives the orientation of a particular wind vector cell. It should be obtained from the data available in the BUFR input files.

It is not completely trivial to select the optimum method at forehand, because the coordinates of each wind vector cell is given within 0.01°. At 25 km resolution this may cause considerable errors in the orientation of a single wind vector cell.

Vogelzang [2006] compares four methods for calculating the wind vector cell orientation. The most precise results are obtained with a spherical triangular method using two points on the swath as far apart from each other as possible and the North Pole as third point. This method leads to error of less than 1° in the orientation of a wind vector cell. See *Vogelzang* [2006] for more details.

Chapter 3

Module tests

In this chapter the various tests to individual modules within SDP are presented. The tests are listed alphabetically in the module name. Table 3.1 gives an overview of the modules tested, their location, the name of the associated test programs, and their availability in SDP.

Module tests have been included in SDP if the following conditions were satisfied:
1. The test does not require additional software;
2. The output of the test program is self explanatory enough to judge the outcome of the test.

Module name	**Location**	**Test program**	**Included in SDP**
BFGSMod	genscat/support/BFGS	*Test_BFGS*	yes
BufrMod	genscat/support/bufr	*test_modules*	yes
convert	genscat/support/convert	*test_convert*	yes
CostFunction	genscat/ambrem/twodvar	*Test_SOS*	yes
StrucFunc	genscat/ambrem/twodvar	*Test_SOS*	yes
DateTimeMod	genscat/support/datetime	*TestDateTimeMod*	yes
ErrorHandler	genscat/support/ErrorHandler	*TestErrorHandler*	yes
LunManager	genscat/support/file	*TestLunManager*	yes
SingletonFFT	genscat/support/singletonfft	*TestSingleton*	yes
numerics	genscat/support/num	*test_numerics*	yes
SortMod	genscat/support/sort	*SortModTest*	yes

Table 3.1 Overview of module tests.

3.1 Module *BFGSMod* (genscat support)

Directory `genscat/support/BFGS` contains program `Test_BFGS`. This program tests the minimization routine LBFGS and its associated routines in module *BFGSMod*. The routines in *BFGSMod* are a slightly modified versions of the freeware routine `LBFGS` and its subroutines. `LBFGS` was written by J. Nocedal.

Program `Test_BFGS` finds the minimum of the function

$$f(x) = \sum_{i=1}^{100000} (x-i)^4 \quad . \tag{3.1}$$

The minimum is the point (1,2,…,100000). The search starts at the origin. The typical output is shown in table 3.2.

```
Program Test_BFGS testing routine LBFGS

 Behavour of cost function:
 Iter         Cost
 -----------------
    0  0.20001E+25
    1  0.19527E+25
...
   85  0.95608E-16
   86  0.30995E-16

 Routine LBFGS completed succesfully
   Number of iterations                :      87
   Dimension of problem                :  100000
   Number of corrections in BFGS update  :       5
   Cost function at start              :  0.20001D+25
   Cost function at end                :  0.30995D-16
   Precision required                  :  0.10D-19
   Norm of final X                     :  0.18258D+08
   Norm of final G                     :  0.97625D-13
   Minimum and Maximum error in solution :  0.000003  0.000005
   Time needed                         :  0.277  seconds
 Program Test_BFGS completed succesfully.
```

Table 3.2 Output of program Test_BFGS.

3.2 Module *BufrMod* (genscat support)

Directory `genscat/support/bufr` contains program *test_modules*. This program is generated and called automatically by the genscat make system, since it is needed to translate the ASCII BUFR tables to binary form.

Moreover, the program can be used to check the BUFR library and its interface to SDP. Table 3.3 shows the output generated by *test_modules*. The first part of the path to the BUFR tables depends, of course, on where SDP is actually installed in the user's system. The last part of the path, `/genscat/support/bufr/bufr_tables`, is fixed.

```
 nr of BUFR messages in this file is:      1
                   ECMWF

       BUFR DECODING SOFTWARE VERSION -  400
```

```
  Your path for bufr tables is :
  ./bufr_tables/
 BUFR TABLES TO BE LOADED  B0000000000210000001.TXT,D0000000000210000001.TXT
  tbd%nelements =            44
  pos_lat =           25
  pos_lon =           26
  latitude  range:    -3.630000        1.260000
  longitude range:     2.850000        7.690000
                    ECMWF

       BUFR ENCODING SOFTWARE VERSION -  7.2
             1 April  2007.



  Your path for bufr tables is :
  ./bufr_tables/
 BUFR TABLES TO BE LOADED  B0000000000210000001.TXT,D0000000000210000001.TXT
```

Table 3.3 Output of program *test_modules.*

3.3 Module *Compiler_Features* (genscat support)

Directory `genscat/support/Compiler_Features` contains *Compiler_Features.F90*, a mudule with alternative routines for *iargc* and *getarg*. The routines in `Compiler_Features` run under all supported Fortran compilers, while *iargc* and *getargc* are an extension to Fortran 90, and therefore not fully supported by all compilers.

TestCompiler_Features is a test program. It can be called with any number of arguments, which will be printed by the program. Its output without command line arguments is shown in table 3.4.

```
  ---------------------------
  testing special characters:
  ---------------------------
  ichar(tabchar) =            9
  [tabchar] = [   ]
  ichar(retchar) =           13
 ][retchar] = [
  ichar(newline) =           10
  [newline] = [
 ]
  ichar(bs) =           92
  [bs] = [\]
  ---------------------------
  testing argument handling:
  ---------------------------
  nr. of arguments present =            0
```

Table 3.4 Output of program *TestCompiler_Features.*

3.4 Module *convert* (genscat support)

Directory `genscat/support/convert` contains module *convert.F90* which contains a number of routines for the conversion of meteorological and geographical quantities. Its associated test program is *test_convert*, and part of its output is listed in table 3.5. Program test_convert produces quite a lot of output.

It starts with checking some conversions between different wind vector representations and transformations between different geographical coordinate systems, followed by a check of the transformation from orbit angles (*p*,*a*,rot(*z*)) to three-dimensional position (*x*,*y*,*z*).

Only the results for *p* = 0° and 90° are shown in table 3.5; those for *p* = 10°, 45°, and 70° are omitted. Program *test_convert* ends with some trigoniometric calculations on a sphere.

```
  ===========================================
  u =     5.000000      v =    -7.000000
  uv_to_speed, uv_to_dir ====> sp =     8.602325      dir =     324.4623
  ===========================================
  sp =     8.602325      dir =     324.4623
  speeddir_to_u, speeddir_to_v ====> u =     5.000001      v =    -7.000000
  ===========================================
  met2uv: sp =     10.00000      dir =     135.0000
  met2uv: ====> u =    -7.071068      v =     7.071068
  uv2met: u =    -7.071068      v =     7.071068
  uv2met: ====> sp =     10.00000      dir =     135.0000
  ===========================================
  lat,lon =     55.00000        5.000000
  latlon2xyz: ====> x,y,z =    0.5713938       4.9990479E-02   0.8191521
  x,y,z =    0.5713938       4.9990479E-02   0.8191521
  xyz2latlon: ====>lat,lon =     54.99999        4.999999
  ===========================================
          p           a      rot_z          x           y           z          a1     rot_z1          a2     rot_z2
    0.00000  -90.00000    0.00000    0.00000    0.00000   -1.00000  -90.00000    0.00000  270.00000  180.00000
    0.00000  -90.00000   15.00000    0.00000    0.00000   -1.00000  -90.00000   15.00000  270.00000  180.00000
    0.00000  -90.00000   30.00000    0.00000    0.00000   -1.00000  -90.00000   30.00000  270.00000  180.00000
...
   90.00000   90.00000   30.00000   -0.50000    0.86603    0.00000   60.00000    0.00000  120.00000    0.00000
   90.00000   90.00000   45.00000   -0.70711    0.70711    0.00000   44.99999    0.00000  135.00000    0.00000
   90.00000   90.00000   60.00000   -0.86603    0.50000    0.00000   30.00000    0.00000  150.00000    0.00000
  ===========================================
  latlon1 =     5.000000        5.000000     latlon2 =     6.000000        5.000000
  angle distance =     1.000000
  km distance    =     111.3188
  latlon1 =     55.00000        5.000000     latlon2 =     56.00000        5.000000
  angle distance =     1.000000
  km distance    =     111.3188
  latlon1 =     85.00000        5.000000     latlon2 =     86.00000        5.000000
  angle distance =     1.000000
  km distance    =     111.3188
  ===========================================
  latlon1 =     5.000000        5.000000     latlon2 =     5.000000        6.000000
  angle distance =    0.9961947
  km distance    =     110.8952
  latlon1 =     55.00000        5.000000     latlon2 =     55.00000        6.000000
  angle distance =    0.5735765
  km distance    =     63.84987
  latlon1 =     85.00000        5.000000     latlon2 =     85.00000        6.000000
  angle distance =    8.7155804E-02
  km distance    =     9.702084
  ===========================================

  Test WVC_Orientation
    WVC1 coordinates (Lam1,Phi1) =   -115.2000       -18.61000
    WVC2 coordinates (Lam2,Phi2) =   -123.6500       -17.52000
    WVC1 orientation Alfa1 =    173.5995        (Should equal 173.5994720)
    WVC2 orientation Alfa2 =    170.9748        (Should equal 170.9747467)
  ===========================================
```

Table 3.5 Output of program *test_convert*

3.5 Modules *CostFunction* and *StrucFunc* (genscat ambrem)

Module *CostFunc.F90* in directory `genscat/ambrem/twodvar` contains the cost function definition of the 2DVAR method. Module *StrucFunc* in the same directory contains the error covariance model of the background filed. Large parts of these modules are tested in the single observation solution test implemented in program *Test_SOS*. Table 3.6 lists its output.

This single observation test proved of crucial importance for getting the normalizations in 2DVAR right. The main idea behind it is that the 2DVAR analysis increment can be calculated analytically in case of one single observation with unit probability. Starting with zero background increment and an observation increment (t_o, l_o) on the 2DVAR grid at the position with indices (1,1), the initial total cost function equals

$$J_t^{init} = \frac{t_o^2 + l_o^2}{\varepsilon^2} \quad , \tag{3.1}$$

where ε stands for the standard deviation of the observation error, which is set to 1.8 in *Test_SOS*. The 2DVAR problem now reduces to a simple optimal interpolation problem. If the standard deviation of the background error is set to the same value as that of the observation error, the final solution has $J_t^{fin} = J_o^{fin} + J_b^{fin} = \frac{1}{2} J_t^{init}$ with $J_b^{fin} = J_o^{fin}$. This allows construction of the final solution and its gradient, see *Vogelzang* [2007] for more detailed information and a complete description of the 2DVAR method.

Program *Test_SOS* reads the observation increment and the structure function parameters from an input file with default name *Test_SOS.inp*, see below. The Helmholz transformation coefficients are set according to option JV, which is the default option standing for sampled continuum (the other option is for periodic boundary conditions but these do not reproduce the correct scaling, see *Vogelzang* [2007] for more details). The program copies the structure function parameters into the *SF*-struct, and the observation increments in the *TwoDvarObs*-struct. The structure function parameters are printed by routine *PrnStrucFuncPars*.

```
=================================================================
PROGRAM Test_SOS - Single Observation Soluton Check
=================================================================


Input read from file        : Test_SOS.inp
Helmholz coefficients type  : JV
2DVAR:
2DVAR: Parameters inside the StructFunc module:
2DVAR:  Grid size in position domain      :    100000.0       m
2DVAR:  Grid dimensions                   :          32  by          32
2DVAR:  Free edge size                    :           5  points
2DVAR:  Structure function type           : Gaus
2DVAR:  Northern hemisphere:
2DVAR:    Error standard deviation in psi :    2.000000      m/s
2DVAR:    Error standard deviation in chi :    2.000000      m/s
2DVAR:    Rotation/divergence ratio       :   0.1000000
2DVAR:    Range parameter for psi         :    300000.0
2DVAR:    Range parameter for chi         :    300000.0
2DVAR:  Tropics:
2DVAR:    Error standard deviation in psi :    1.800000      m/s
2DVAR:    Error standard deviation in chi :    1.800000      m/s
2DVAR:    Rotation/divergence ratio       :    1.000000
2DVAR:    Range parameter for psi         :    300000.0
2DVAR:    Range parameter for chi         :    300000.0
2DVAR:  Southern hemisphere:
2DVAR:    Error standard deviation in psi :    2.000000      m/s
2DVAR:    Error standard deviation in chi :    2.000000      m/s
2DVAR:    Rotation/divergence ratio       :   0.1000000
2DVAR:    Range parameter for psi         :    300000.0
2DVAR:    Range parameter for chi         :    300000.0

CheckCovMat - checking precision of Covariances
  Relative precision in covariances of psi:    0.000000
  Relative precision in covariances of chi:   3.1197767E-04
```

```
Number of observations      :           1
Number of control variables :        2046

Obs2dvar after initialization:
 i  j  Namb    u    v          Jo          gu          gv
------------------------------------------------------------
 1  1     1  1.0  0.0  0.77160E-01 -0.30864E+00  0.00000E+00

The gradient velocity fields duo and dvo (nonzero components only):
 i  j         duo         dvo
------------------------------
 1  1 -0.30864E+00  0.00000E+00

The cost function of the solution:
  Observation part :   7.7160493E-02
  Background part  :   7.7160530E-02        precision   3.7252903E-08

The background velocity field:
  u(1,1)         :   0.5000000
  Expected value :   0.5000000                precision   2.9802322E-08
  v(1,1)         :   3.1561567E-20
  Expected value :    0.000000                precision   3.1561567E-20

Check background cost function
  Direct calculation from psi and chi   :   7.7160530E-02
  Calculation by Jb from control vector :   7.7160537E-02        precision
  7.4505806E-09

Check observation cost function
  Expected value                        :   7.7160493E-02
  Calculation by Jo from control vector :   7.7160463E-02        precision
  2.9802322E-08
  Precision in gradients better than    1.9753901E-10

Check packing/unpacking:
  Precision in packing/unpacking of xi    0.000000
  Precision in packing/unpacking of psi    0.000000
  Precision in packing/unpacking of chi    0.000000

Program Test_SOS completed.
================================================================
```

Table 3.6 Output of the single observation solution test.

The error covariances are calculated numerically in module *StrucFunc*. For Gaussian structure functions, they can also be calculated analytically. The two methods are compared and the relative precision is printed. In table 3.6 it is 0.0 for the stream function ψ (since it is identically zero in this example) and 0.0003 for the velocity potential χ.

The total cost function and its gradient is evaluated by routine *JoScat* in module *CostFunction*. From this the cost function components and gradients at the final solution are calculated and checked against their analytical value. The (absolute) precision is printed. Finally, *Test_SOS* checks the packing and unpacking routines of the control vector in both directions.

As stated before, program *Test_SOS* reads its input from an input file. The name (and path) of that file must be given as command line argument of *Test_SOS*. When omitted, the program assumes *Test_SOS.inp* as input file. Table 3.7 gives the structure and contents of the input file. It is in free format.

Record	Item nr.	Name	Meaning
1	1	u0_ini	Initial observation increment in transversal direction (m/s)
1	2	v0_ini	Initial observation increment in longitudinal direction (m/s)
2	1	lparameter	Read 2DVAR parameters from file?
3	1	TDVParameterFile	Name and path of files with 2DVAR parameters.

Table 3.7 Input file for *Test_SOS*.

The default input file *Test_SOS.inp* refers to file *SOT01.2DV* as input file for the 2DVAR parameters. This file contains the default 2DVAR parameter settings, but without gross error probabilities (GEP's) and an equatorial background error correlation length of 300 km. Files *SOT02.2DV* and *SOT03.2DV* contain alternative 2DVAR settings, see table 3.8, and are called by input files *Test_SOS.in2* and *Test_SOS.in3*, respectively. Script *SOS.sh* runs *Test_SOS* with all three input files.

File	2DVAR grid	Free edge	Grid Size
SOT01.2DV	32 × 32	5	100 km
SOT02.2DV	48 × 48	18	50 km
SOT03.2DV	64 × 80	24	25 km

Table 3.8 2DVAR grid definitions for *Test_SOS*.

3.6 Module *DateTimeMod* (genscat support)

Module *DateTimeMod.F90* in directory `genscat/support/datetime` contains general purpose date and time help functions. These are tested by program *TestDateTimeMod*, the output of which is listed in table 3.9.

```
time-tests
time: 14:22:03.70
time_real        = 51723.70
time_real + 77.2 = 51800.90
time: 14:23:20.90
 time2 is valid
 time1 =
time: 14:22:03.70
 time2 =
time: 14:23:20.90
 time 1 .ne. time2
 date-tests
date: 15-12-1999
 date_int =      19991215
 date_int + 1 =      19991216
date: 16-12-1999
 date2 is valid
 date1 =
date: 15-12-1999
 date2 =
date: 16-12-1999
 date 1 .ne. date2
 date-stepping-tests
 ERROR: The date      21000101  is outside the
range
 19000101...20991231, this is not implemented
at this time
 ERROR: Julian routines differ from my own
routines
date: 31-12-2099
 next_date_int =    2147483647
date: 01-01-2100
 next_julian_date_int =      21000101
 all OK
 before:
time: 23:59:57.70
date: 31-12-1999
after incrementing by:  5.22 seconds
time: 00:00:02.92
date: 01-01-2000
```

```
  valid time
  test of function date2string: 19991231
  test of function date2string_sep: 1999-12-31
  test of function time2string: 235957
  test of function time2string_sep: 23:59:57
  before convert_to_derived_datetime:
 date: 28-02-2005
 time: 52:00:00.00
  after convert_to_derived_datetime:
 date: 02-03-2005
 time: 04:00:00.00
  Current date and time:
 date: 08-09-2014
 time: 07:00:35.52
```

Table 3.9 Output of program *TestDateTimeMod*.

3.7 Module *ErrorHandler* (genscat support)

Module *ErrorHandler.F90* in directory `genscat/support/ErrorHandler` contains routines for handling errors during program execution. The module is tested by program *TestErrorHandler*, the output of which is listed in table 3.10.

```
 The Error Handler program_abort routine is set to
 return after each error,
 in order to try and resume the program...
 testing: report_error
 an error was reported from within subroutine: dummy_module_name1
 error while allocating memory
 testing: program_abort (with abort_on_error = .false.)
 an error was reported from within subroutine: dummy_module_name2
 error while allocating memory
 ==> trying to resume the program ...
 The Error Handler program_abort routine is set to
 abort on first error...
 testing: program_abort (with abort_on_error = .true.)
 an error was reported from within subroutine: dummy_module_name2
 error while allocating memory
```

Table 3.10 Output of program *TestErrorHandler*.

3.8 Module *LunManager* (genscat support)

Module *LunManager.F90* in directory `genscat/support/file` contains routines for file unit management. It is tested by program *TestLunManager*, the output of which is listed in table 3.11.

```
 Starting fileunit test program
  ===== lun_manager ======
  fileunit:           31  was not in use !!!
  free_lun returns without freeing any fileunit
  fileunit:           88  was not in the range that is handled
  by this module ! (          30  -           39 )
  free_lun returns without freeing any fileunit
  fileunit:           88  was not in the range that is handled
  by this module ! (          30  -           39 )
  enable_lun returns without enabling any fileunit
  fileunit:           88  was not in the range that is handled
  by this module ! (          30  -           39 )
  disable_lun returns without disabling any fileunit
  fileunit:           21  was not in the range that is handled
  by this module ! (          30  -           39 )
  disable_lun returns without disabling any fileunit
```

```
unit:            31  is used?:   F
unit:            31  is used?:   T
start of inspect_luns
 lun              0  is open
 lun              0  has a name: stderr
 lun              5  is open
 lun              5  has a name: stdin
 lun              6  is open
 lun              6  has a name: stdout
 lun             31  is open
 lun             31  has a name: TestLunManager.F90
end of inspect_luns
fileunit:            31  is still in use !
disabling it is only possible if it is not used !
disable_lun returns without disabling any fileunit
fileunit:            30  is in use
fileunit:            31  is in use
fileunit:            32  is still available
fileunit:            33  is still available
fileunit:            34  is still available
fileunit:            35  is still available
fileunit:            36  is still available
fileunit:            37  is still available
fileunit:            38  is still available
fileunit:            39  is still available
fileunit:            21  was not in the range that is handled
by this module ! (           30   -            39 )
enable_lun returns without enabling any fileunit
fileunit:            22  was not in the range that is handled
by this module ! (           30   -            39 )
enable_lun returns without enabling any fileunit
```

Table 3.11 Output of program *TestLunManager*.

3.9 Module *SingletonFFT_Module* (genscat support)

Module *SingletoniFFT_Module.F90* in directory `genscat/support/singletonfft` contains routines for mixed-radix multivariate Fast Fourier Transformation (FFT). The module is based on routine `fft` originally written by R.C. Singleton and available at www.netlib.org/go/fft. The original program does not compile with Portland under all optimization options, so it was splitted in separate subroutines and translated to Fortran 90.

Program *TestSingleton* contains tests for the forward and backward FFT in one and two dimensions for various even grid sizes between 32 and 50. The test function is a Gaussian. Table 3.12 shows part of the output of *TestSingleton*.

Since for a Gaussian function the Fourier transform can be calculated analytically, it is possible to find the (absolute) accuracy of the FFT method. The worst case accuracies are listed at the end of the output of *TestSingleton*. These figures should be of the order of 10^{-5} at most as in table 3.12.

```
=========================================================================
PROGRAM TestSingleton
Test of SingletonFFT routines by comparing with analytical FT
=========================================================================

Spreading times grid size in dimension 1:   0.1000000      (should be ~ 0.1)
Spreading times grid size in dimension 2:   0.1000000      (should be ~ 0.1)
=========================================================================

 1D             F O R W A R D           B A C K W A R D
            P r e c i s i o n         P r e c i s i o n
 N1         Real         Imag         Real         Imag
--------------------------------------------------------
 32  0.83631E-06  0.10286E-04  0.11921E-06  0.69247E-07
 34  0.61329E-06  0.78932E-05  0.11921E-06  0.11285E-07
 36  0.94782E-06  0.12215E-04  0.11921E-06  0.11036E-06
 38  0.27877E-06  0.20358E-05  0.17881E-06  0.22604E-07
 40  0.83631E-06  0.12143E-04  0.11921E-06  0.54017E-07
 42  0.44603E-06  0.56252E-05  0.77824E-07  0.92940E-07
 44  0.12900E-06  0.27819E-06  0.17881E-06  0.14948E-06
 46  0.94782E-06  0.13554E-04  0.35763E-06  0.34905E-07
 48  0.94782E-06  0.14143E-04  0.23842E-06  0.12666E-06
 50  0.50178E-06  0.66967E-05  0.17881E-06  0.10431E-06
==========================================================================

 2D                 F O R W A R D   F F T         B A C K W A R D   F F T
                 P r e c i s i o n                P r e c i s i o n
 N1  N2          Real         Imag   Time         Real         Imag   Time
--------------------------------------------------------------------------
 32  32  0.11995E-05  0.20572E-04 0.0000  0.17881E-06  0.10663E-06 0.0000
 32  34  0.10952E-05  0.18179E-04 0.0000  0.11921E-06  0.63061E-07 0.0000
 32  36  0.12516E-05  0.22501E-04 0.0000  0.11921E-06  0.11339E-06 0.0000
 32  38  0.88658E-06  0.82503E-05 0.0000  0.17881E-06  0.66826E-07 0.0000

 50  46  0.99089E-06  0.20251E-04 0.0001  0.23842E-06  0.10202E-06 0.0001
 50  48  0.10430E-05  0.20840E-04 0.0001  0.29802E-06  0.15117E-06 0.0001
 50  50  0.57367E-06  0.13393E-04 0.0001  0.35763E-06  0.11255E-06 0.0001
==========================================================================

Program TestSingleton: Resume
Worst case accuracies

                F O R W A R D          B A C K W A R D
             Real         Imag         Real         Imag
---------------------------------------------------------
1D     0.94782E-06  0.14143E-04  0.35763E-06  0.14948E-06
2D     0.13559E-05  0.28287E-04  0.77486E-06  0.28650E-06

Program TestSingleton: Normal termination.
==========================================================================
```

Table 3.12 Part of the output of program *TestSingleton.*

3.10 Module *numerics* (genscat support)

Module *numerics.F90* in directory `genscat/support/num` contains routines for checking and handling numerical issues like variable sizes and ranges. These are tested by program *test_numerics*, the output of which is listed in Table 3.13.

```
Starting numerics test program
 ===== representation tests ======
 REALACC(6)
 r4: digits                24
 r4: epsilon       1.1920929E-07
 r4: huge          3.4028235E+38
 r4: minexponent         -125
 r4: maxexponent          128
 r4: precision              6
 r4: radix                  2
 r4: range                 37
 r4: tiny          1.1754944E-38
 ENDREALACC
 REALACC(12)
 r8: digits                53
 r8: epsilon       2.2204460492503131E-016
 r8: huge          1.7976931348623157E+308
 r8: minexponent        -1021
```

```
  r8: maxexponent         1024
  r8: precision             15
  r8: radix                  2
  r8: range                307
  r8: tiny          2.2250738585072014E-308
  ENDREALACC
  ===== numerics tests ======
  int1 =    127
  int2 =    32767
  int4 =    2147483647
  int8 =       9223372036854775807
  huge(int1) =          127
  huge(int2) =        32767
  huge(int4) =    2147483647
  huge(int8) =       9223372036854775807
  REALACC(6)  r4   =    1.7000000E+38  ENDREALACC
  REALACC(12) r8   =    1.7000000000000000E+038  ENDREALACC
  ===== check variable sizes  ======
  Variable sizes are as expected
  ===== detect and print variable sizes ======
 var_type nr_of_words range precision
        i          4     9
      i1_          1     2
      i2_          2     4
      i4_          4     9
      i8_          8    18
       dr          4    37         6
       s_          4    37         6
       l_          4    37         6
       r_          4    37         6
      r4_          4    37         6
      r8_          8   307        15
  ===== dB conversion test ======
  REALACC(6)
  input test number:           1.2300001E-04
  converted to dB:             -39.10095
  converted back to a real:    1.2299998E-04
  ENDREALACC
  ===== done ======
```

Table 3.13 Output of program *test_mumerics*.

3.11 Module *SortMod*

Module *SortMod* in directory `genscat/support/sort` contains two routines for sorting the wind vector solutions found in the inversion step to their probability. The associated test program is *SortModTest*. Its output is shown in table 3.14.

```
 Test program for the SortMod module
  Unsorted array
  10.0  9.0  8.0  7.0  6.0  5.0  4.0  3.0  2.0  1.0
  After GetSortIndex
   1.0  2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0 10.0
  Sorted array, after SortWithIndex
   1.0  2.0  3.0  4.0  5.0  6.0  7.0  8.0  9.0 10.0
```

Table 3.14 Output of program *SortModTest*

Chapter 4

Integration tests

The 2DVAR method is tested in the test program *Test_SOA.* This program is very similar to program *Test_SOS* described in section 3.4, but in *Test_SOA* the final solution is found by numerical minimization of the cost function, thus also testing module *BFGSMod.*

The performance of the SDP program is tested by judging the graphical representations of the wind field of the SDP processor itself (integration test) and by numerical comparison of the SDP wind field with that of the QDP program (validation test). The latter test is the subject for the next chapter.

4.1 2DVAR

Directory `genscat/ambrem/twodvar` contains program *Test_SOA* which tests the 2DVAR method on a single observation. *Test_SOA* is similar to *Test_SOS* in the same directory. The difference is that *Test_SOA* finds the minimum of the cost function by numerical minimization of the cost function with routine LBFGS. Table 4.1 shows the output of *Test_SOA*.

The most important item is the final (analysis increment) velocity field at the position of the observation. For the given parameter values (in particular equal values of the standard deviations of both observation error and background error), the analysis increment velocity should be exactly half of the input observation increment at the final solution point. As can be seen from table 4.1 this is achieved within machine precision. More information on the 2DVAR method and on the single observation analysis, including plots of the resulting analysis increment fields, can be found in *Vogelzang* [2007].

Like program *Test_SOS* in section 3.4, program *Test_SOA* reads some input on the initial observation and the structure function parameters from an input file. The name (and path) of this input file is a command line of *Test_SOA*. Its default name is *Test_SOA.inp*. Table 4.2 lists the parameters in the input file.

```
================================================================
Program Test_SOA - Single Observation Analysis
================================================================

2DVAR:
2DVAR: Parameters inside the StructFunc module:
2DVAR:  Grid size in position domain      :    100000.0       m
2DVAR:  Grid dimensions                   :           32  by           32
2DVAR:  Free edge size                    :            5  points
2DVAR:  Structure function type           : Gaus
2DVAR:  Northern hemisphere:
2DVAR:    Error standard deviation in psi :    2.000000      m/s
2DVAR:    Error standard deviation in chi :    2.000000      m/s
2DVAR:    Rotation/divergence ratio       :   0.1000000
2DVAR:    Range parameter for psi         :    300000.0
2DVAR:    Range parameter for chi         :    300000.0
2DVAR:  Tropics:
2DVAR:    Error standard deviation in psi :    1.800000      m/s
2DVAR:    Error standard deviation in chi :    1.800000      m/s
2DVAR:    Rotation/divergence ratio       :    1.000000
2DVAR:    Range parameter for psi         :    300000.0
2DVAR:    Range parameter for chi         :    300000.0
2DVAR:  Southern hemisphere:
2DVAR:    Error standard deviation in psi :    2.000000      m/s
2DVAR:    Error standard deviation in chi :    2.000000      m/s
2DVAR:    Rotation/divergence ratio       :   0.1000000
2DVAR:    Range parameter for psi         :    300000.0
2DVAR:    Range parameter for chi         :    300000.0

Number of observations      :           1
Number of control variables :        2046

SOA: Obs2dvar after initialization:
   i  j  Namb    u    v
  ---------------------
  65 65     1  0.0  1.0

The cost function before minimization
  Observation part :   0.3086420
  Background part  :    0.000000
  Total            :   0.3086420

The cost function after minimization
  Observation part :   7.7160463E-02
  Background part  :   7.7160433E-02
  Total            :   0.1543209

Check norm of control vector and gradient
  Norm of gradient       :   8.1465704E-13
  Norm of control vector :    628539.1
  Their ratio            :   1.2961120E-18
  Maximum allowed ratio  :   1.0000000E-18

The final velocity field at point (iobs,jobs)=          17           17
  u=   8.7142391E-11      precision   8.7142391E-11
  v=   0.5000000          precision    0.000000

Program Test_SOA completed
================================================================
```

Table 4.1 Output of program *Test_SOA*.

Record	Item nr.	Name	Meaning
1	1	iobs	First position index of observation on 2DVAR grid
1	2	jobs	Second position index of observation on 2DVAR grid
1	3	uobs	Initial observation increment in transversal direction (m/s)
1	4	vobs	Initial observation increment in longitudinal direction (m/s)
2	1	lparameter	Read 2DVAR parameters from file?
3	1	TDVParameterFile	Name and path of file with 2DVAR parameters

Table 4.2 Input file for *Test_SOA*.

Note that *Test_SOA.inp* reads its 2DVAR parameters from file *SOT01.2DV*. The input files *TEST_SOA.in2* and *Test_SOA.in3* read their 2DVAR data from files *SOT02.2DV* and *SOT03.2DV*, respectively. See section 3.5 for more information on these files. Script *SOA.sh* performs three runs of Test_SOA with the three input files mentioned above.

4.2 SDP

Directory `SDP/tests` contains four BUFR files for testing the SDP executable. File `QS_D08001_S0006_E0052_B4444343` is the first full orbit file of 2008 and acts as input file for SDP test run scripts `RunSDP_Test_1`, `RunSDP_Test_2`, and `RunSDP_Test_3`. Files `SDP_Test_1.bufr`, `SDP_Test_2.bufr`, and `SDP_Test_3.bufr` are their output files. See also table 4.3.

Running one of the scripts in table 2.7 will yield a BUFR output file with the default name `QS_D08001_S0006_E0052_B4444343_025` for the first and second command, or `QS_D08001_S0006_E0052_B4444343_100` for the third command. These files should contain the same results as the corresponding `SDP_Test_i.bufr` files.

Script	Output from script	Output identical with
RunSDP_Test_1	QS_D08001_S0006_E0052_B4444343_025	SDP_Test_1.bufr
RunSDP_Test_2	QS_D08001_S0006_E0052_B4444343_025	SDP_Test_2.bufr
RunSDP_Test_3	QS_D08001_S0006_E0052_B4444343_100	SDP_Test_3.bufr

Table 4.3 SDP test runs.

Directory `sdp/execs` contains a number of shell scripts of which `sdp_025` and `sdp_qdp` are invoked by the test scripts of table 4.3

The shell scripts in `sdp/execs` automatically set the values for the environment variables needed by SDP. The first run is a standard SDP run at 25 km resolution. The second run is the same as the first run, but now with the Multi Solution Scheme for Ambiguity Removal switched on. The third run is at 100 km resolution in QDP mode.

Figure 4.1 shows the global coverage of the test run. SeaWinds covered small parts of the Hudson Bay, the Great Lakes, and the Gulf of Mexico, and a large strip in the Pacific west of South America. The colors indicate the magnitude of the wind speed as indicated by the color bar.

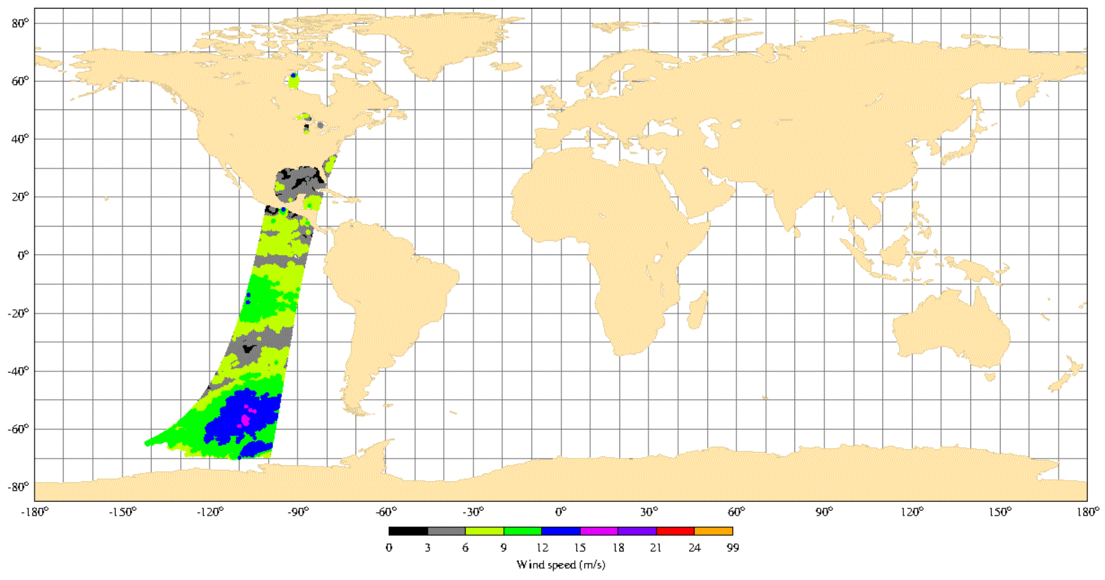

Figure 4.1 Global coverage of the test runs. Wind speed results for test run 2 are shown.

Figure 4.1 shows the results of test run number 2, but the two other test runs yield very similar results for the magnitude of the wind speed.

Figures 4.2 and 4.3 show detailed wind fields at 25 km resolution over the Pacific, west of South America (lower left part in figure 4.1). Figure 4.2 is obtained with the test run 1 data. Note that the wind field is quite noisy, especially near the centre of the SeaWinds swath. Here the angle between the fore and aft measurements of the scatterometer are almost 180° apart, which is quite unfavorable. The gaps in the wind field are due to measurements that were rejected during quality control, probably due to rain.

Figure 4.3 is obtained with test run 2 data, i.e., with the Multiple Solution Scheme switched on. Now the wind field is much smoother. Note that a sharp front in the upper right corner appears clearly in figure 4.3 while it is obscured by noise in figure 4.2.

Due to round-off differences, a simple file comparison is not appropriate to test the SDP BUFR output. It is necessary to decode the BUFR files and compare the retrieved wind field with the one in the `SDP_Test_i.bufr` file. Such software is not part of the SDP package, but is available at KNMI and may be obtained upon request.

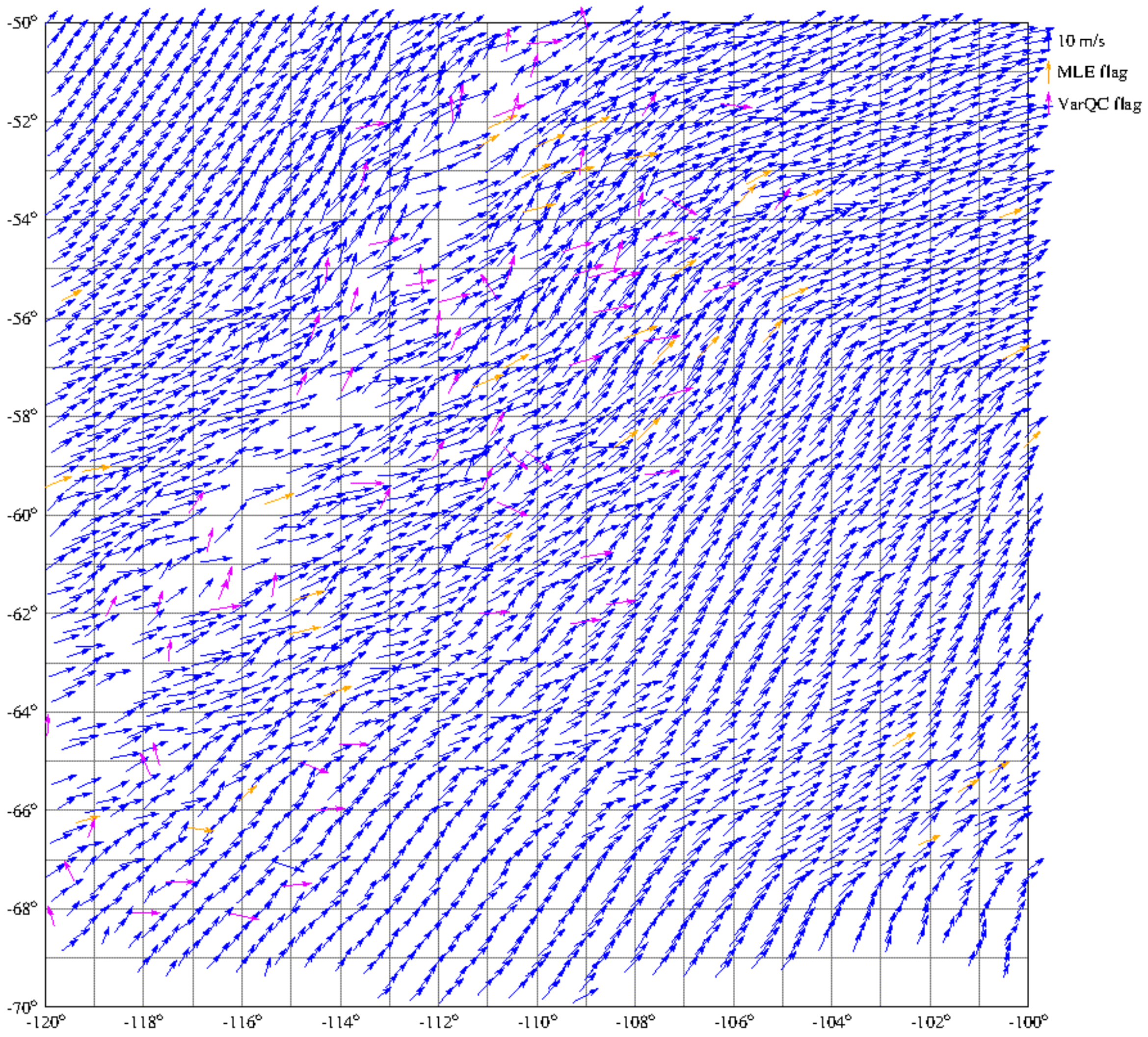

Figure 4.2 Part of test run 1 (Pacific west of South America) obtained with SDP standard processing at 25 km resolution. Purple arrows represent cells with the variational quality control flag set; orange arrows cells with the KNMI MLE flag set.

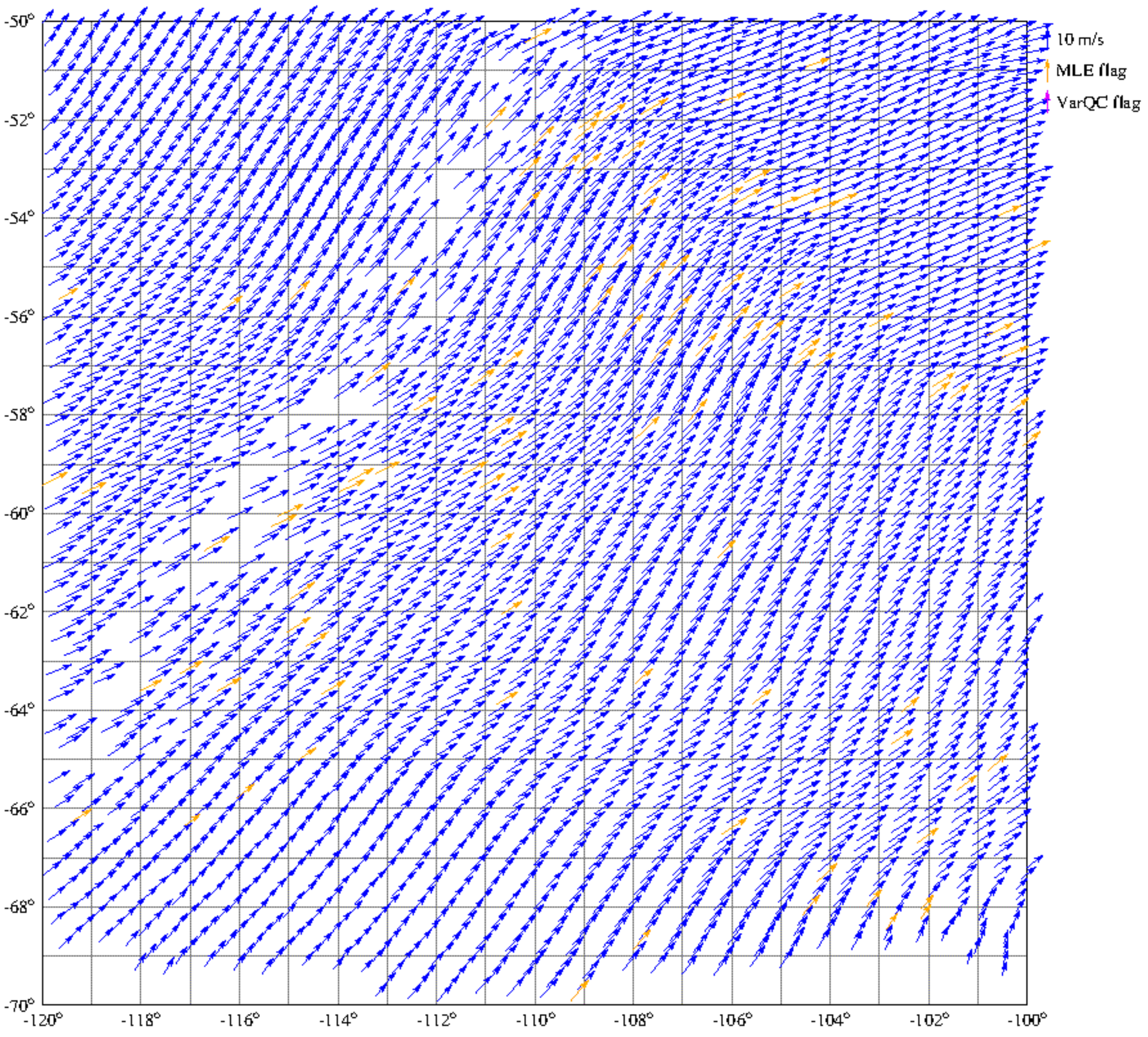

Figure 4.3 As figure 4.2, but now with the Multiple Solution Scheme switched on.

Chapter 5

Validation tests

The SDP program should be backward compatible with the QDP program. In other words, running SDP in QDP mode should give the same results (except, of course, for the improvements in SDP).

5.1 Description

Python script `QdpCompare.py` in directory `SDP/tests` runs the QDP comparisons tests. The scripts sets its own environment variables such as `BUFR_TABLES`. This script collects an input BUFR file from directory `SDP/data`. The input BUFR file is processed by both the QDP and the SDP program. The SAP program for conversion of BUFR files to ASCII format produces several ASCII tables for the QDP and the SDP output BUFR file. These ASCII tables have the same name as the input BUFR file preceded by the prefix `sdp\_` or `qdp\_` and a suffix based on the SAP masknumber and the extension `.asc`. Next, the ASCII tables are compared using for example the grep program to extract specific nodes or rows. Results are produced in the folder `SDP/tests/scratch/QdpCompare`. Here follows some snapshots of the results. Of course, other rows and nodes have occasionally been selected.

Note that the SAP program is not part of the SDP release. The tests in this chapter have not been repeated for SDP release 2.2.

5.2 Orbit data comparison

ASCII tables were produced with only orbit data. Table 5.1 presents the results for revolution number 22043 and row number 101. The columns represent revolution number, row number, node number, date, time, latitude, and longitude, respectively. The SDP data are shown in the upper half of table 5.1; the QDP data in the lower half. Note that everything is spot on except for the seconds (fifth column). The reason is that SDP has a just as accurate but more robust processing of the time data during the averaging.

```
> grep "101 " sdp_D03256_S0040_E0154_B2204243_orbit.asc

  22043   101    1  20030913  010838    -2.0700    63.8500
  22043   101    2  20030913  010838    -1.9600    64.6100
  22043   101    3  20030913  010838    -1.8200    65.5200
  22043   101    4  20030913  010838    -1.6900    66.3900
  22043   101    5  20030913  010838    -1.5600    67.2900
  22043   101    6  20030913  010838    -1.4200    68.1700
  22043   101    7  20030913  010838    -1.2900    69.0600
  22043   101    8  20030913  010838    -1.1500    69.9500
  22043   101    9  20030913  010838    -1.0100    70.8400
  22043   101   10  20030913  010838    -0.8800    71.7200
  22043   101   11  20030913  010838    -0.7500    72.6100
  22043   101   12  20030913  010838    -0.6100    73.5000
  22043   101   13  20030913  010838    -0.4700    74.3900
  22043   101   14  20030913  010838    -0.3400    75.2700
  22043   101   15  20030913  010838    -0.2000    76.1700
  22043   101   16  20030913  010838    -0.0700    77.0600
  22043   101   17  20030913  010838     0.0700    77.9400
  22043   101   18  20030913  010838     0.2000    78.8400
  22043   101   19  20030913  010838     0.3200    79.6000

> grep "101 " qdp_D03256_S0040_E0154_B2204243_orbit.asc

  22043   101    1  20030913  010827    -2.0700    63.8500
  22043   101    2  20030913  010827    -1.9600    64.6100
  22043   101    3  20030913  010827    -1.8200    65.5200
  22043   101    4  20030913  010827    -1.6900    66.3900
  22043   101    5  20030913  010827    -1.5600    67.2900
  22043   101    6  20030913  010827    -1.4200    68.1700
  22043   101    7  20030913  010827    -1.2900    69.0600
  22043   101    8  20030913  010827    -1.1500    69.9500
  22043   101    9  20030913  010827    -1.0100    70.8400
  22043   101   10  20030913  010827    -0.8800    71.7300
  22043   101   11  20030913  010827    -0.7500    72.6100
  22043   101   12  20030913  010827    -0.6100    73.5000
  22043   101   13  20030913  010827    -0.4700    74.3900
  22043   101   14  20030913  010827    -0.3400    75.2700
  22043   101   15  20030913  010827    -0.2000    76.1700
  22043   101   16  20030913  010827    -0.0700    77.0600
  22043   101   17  20030913  010827     0.0700    77.9400
  22043   101   18  20030913  010827     0.2000    78.8400
  22043   101   19  20030913  010827     0.3200    79.6000
```

Table 5.1 Orbit data comparison for SDP (upper part) and QDP (lower part).

5.3 Radar cross section comparison

Several ASCII tables are produced with σ_0 data. Table 5.2 presents the data for beam 2, rev olution number 22043, and row number 106. The columns represent revolution number, row number, node number, σ_0, azimuth angle, and incidence angle, respectively. Note that everything is spot on. The values for σ_0 are somewhat odd , because in QDP (and in SDP in QDP mode) linear values of σ_0 are stored as if they were in dB.

```
grep " 106 " qdp_D03256_S0040_E0154_B2204243_sigma0_2.asc
  22043   106    3    1.0000000000   297.6000    54.0000
  22043   106    4    1.0000000000   305.5900    54.0100
  22043   106    5    1.0000000000   313.6700    54.0000
  22043   106    6    1.0000000000   321.0500    54.0200
  22043   106    7    1.0000000000   327.9900    54.0400
  22043   106    8    1.0000000000   334.6200    54.0100
  22043   106    9    1.0000000000   340.9900    54.0400
  22043   106   10    1.0000000000   347.4600    54.0300
  22043   106   11    1.0000000000   353.7900    54.0100
  22043   106   12    1.0000000000     0.2400    54.0400
  22043   106   14    1.0000000000    13.8400    54.0100
  22043   106   15    1.0023052692    21.2400    54.0000
  22043   106   16    1.0023052692    29.3500    54.0000
  22043   106   17    1.0023052692    37.3900    53.9900

grep " 106 " sdp_D03256_S0040_E0154_B2204243_sigma0_2.asc

  22043   106    3    1.0000000000   297.6000    54.0000
  22043   106    4    1.0000000000   305.5900    54.0100
  22043   106    5    1.0000000000   313.6700    54.0000
  22043   106    6    1.0000000000   321.0500    54.0200
  22043   106    7    1.0000000000   327.9900    54.0400
  22043   106    8    1.0000000000   334.6200    54.0100
  22043   106    9    1.0000000000   340.9900    54.0400
  22043   106   10    1.0000000000   347.4600    54.0300
  22043   106   11    1.0000000000   353.7900    54.0100
  22043   106   12    1.0000000000     0.2400    54.0400
  22043   106   14    1.0000000000    13.8400    54.0100
  22043   106   15    1.0023052692    21.2400    54.0000
  22043   106   16    1.0023052692    29.3500    54.0000
  22043   106   17    1.0023052692    37.3900    53.9900
```

Table 5.2 Radar cross section data comparison for QDP (upper part) and SDP (lower part).

5.4 Ambiguity comparison

Table 5.3 presents the data for revolution number 22043 and row number 80, The spacing is adjusted and revolution numbers are left out. The columns represent row number, node number, the number of ambiguities, and the ambiguities (*u*,*v*) themselves, respectively. Again everything is spot on. In contrast to QDP, SDP orders the output ambiguities on the basis of their probability. This makes the comparison a bit more difficult. The entries in table 5.3 where the order of the ambiguities differ are shaded.

```
grep " 80 " qdp_D03256_S0040_E0154_B2204243_ambies.asc
80  3  4  -10.22096   0.89422    8.20000   0.00000  -7.29296    9.50437   3.86483  -10.61853
80  4  4  -10.54897   1.38880   -7.67918   7.67918   8.67174   -0.37862   4.16702   -8.93619
80  5  4  -10.14126   0.88725   -6.94515   6.36406   8.58000    0.00000   3.45702   -7.41360
80  6  3   -9.82154   2.17738    8.97145  -0.39170   4.06199   -6.37604       NaN  -Infinity
80  7  3   -9.32175   3.39284    9.82000   0.00000   4.05124   -6.35917       NaN  -Infinity
80  8  3  -10.83769   2.90395   11.32230   1.49061   0.34547   -7.91246       NaN  -Infinity
80  9  3  -10.85330   3.42203   -8.27225  -7.58012  11.50184    4.18633       NaN  -Infinity
80 10  2    8.88556   6.81813  -10.87594   2.41114       NaN  -Infinity       NaN  -Infinity
80 11  3  -10.25111   2.27262    9.59556   5.54000   9.99727   -2.21634       NaN  -Infinity
80 12  2  -10.51775   1.85456   10.04563  -2.69172       NaN  -Infinity       NaN  -Infinity
80 13  2  -10.60257   2.35053   10.62518  -2.84701       NaN  -Infinity       NaN  -Infinity
80 14  2  -10.66115   2.36352   10.69370  -3.89219       NaN  -Infinity       NaN  -Infinity
80 15  2  -10.16154   2.72278   10.86285  -3.95375       NaN  -Infinity       NaN  -Infinity
80 16  2  -10.07029   4.17125   11.34524  -4.69935       NaN  -Infinity       NaN  -Infinity
80 17  2  -10.34745   4.28605   12.23217  -5.06673       NaN  -Infinity       NaN  -Infinity

grep " 80 " sdp_D03256_S0040_E0154_B2204243_ambies.asc
80  3  4  -10.22096   0.89422    8.20000   0.00000  -7.29296    9.50437   3.86483  -10.61853
80  4  4  -10.54897   1.38880   -7.67918   7.67918   8.67174   -0.37862   4.16702   -8.93619
80  5  4  -10.14126   0.88725   -6.94515   6.36406   8.58000    0.00000   3.45702   -7.41360
80  6  3   -9.82154   2.17738    8.97145  -0.39170   4.06199   -6.37604       NaN  -Infinity
80  7  3   -9.32175   3.39284    9.82000   0.00000   4.05124   -6.35917       NaN  -Infinity
80  8  3  -10.83769   2.90395   11.32230   1.49061   0.34547   -7.91246       NaN  -Infinity
80  9  3  -10.85330   3.42203   -8.27225  -7.58012  11.50184    4.18633       NaN  -Infinity
80 10  2    8.88556   6.81813  -10.87594   2.41114       NaN  -Infinity       NaN  -Infinity
80 11  3  -10.25111   2.27262    9.99727  -2.21634   9.59556    5.54000       NaN  -Infinity
80 12  2  -10.51775   1.85456   10.04563  -2.69172       NaN  -Infinity       NaN  -Infinity
80 13  2  -10.60257   2.35053   10.62518  -2.84701       NaN  -Infinity       NaN  -Infinity
80 14  2  -10.66115   2.36352   10.69370  -3.89219       NaN  -Infinity       NaN  -Infinity
80 15  2  -10.16154   2.72278   10.86285  -3.95375       NaN  -Infinity       NaN  -Infinity
```

```
 80 16  2  -10.07029   4.17125   11.34524  -4.69935       NaN  -Infinity       NaN  -Infinity
 80 17  2  -10.34745   4.28605   12.23217  -5.06673       NaN  -Infinity       NaN  -Infinity
```

Table 5.3 Ambiguity comparison for QDP (upper part) and SDP (lower part). The order of the ambiguities differs in the two shaded entries.

5.5 Selection comparison

Table 5.4 presents a snapshot of the selection comparison. The spacing is adjusted, and date and time are left out. The columns represent revolution number, row number, node number, latitude, longitude, selection number, selected wind (u,v), corresponding probability, and the corresponding model wind (u,v), respectively. In this case, the new KNMI 2DVAR scheme seems to have a better performance in the nadir swath. However, for node 4 the selection is poor. Of course this only a single row. The KNMI 2DVAR scheme requires a proper calibration and validation.

```
grep " 100 " qdp_D03256_S0040_E0154_B2204243_selection.asc
 22043 100 3   -2.7000 65.7100  1  -3.41849  3.73062  0.976000  -2.19302  1.28150
 22043 100 4   -2.5700 66.5900  1  -2.75097  4.31816  0.951000  -1.28981  1.46249
 22043 100 5   -2.4400 67.4800  1  -1.32311  4.19635  0.961000  -0.75782  1.59957
 22043 100 6   -2.3000 68.3700  1   0.71196  4.03771  1.000000  -0.65922  1.53438
 22043 100 7   -2.1700 69.2500  1   0.55343  4.20373  1.000000  -0.87103  1.51783

 22043 100 8   -2.0300 70.1400  1  -0.17186  3.93625  1.000000  -1.34873  1.81343
 22043 100 9   -1.9000 71.0300  1   0.00000  3.72000  1.000000  -1.85392  2.20394
 22043 100 10  -1.7600 71.9200  2   1.93000  3.34286  0.497000  -2.23444  2.40118
 22043 100 11  -1.6300 72.8100  2   5.85351  1.56844  0.031000  -1.90931  2.35275
 22043 100 12  -1.4900 73.7000  2   6.44713  2.03277  0.000000  -0.92908  2.18030

 22043 100 13  -1.3600 74.5800  2   4.66219  0.40789  0.459000   0.44006  1.98173
 22043 100 14  -1.2200 75.4700  1   4.51570  0.19716  0.602000   1.51566  1.60838
 22043 100 15  -1.0800 76.3600  1   5.43482  0.23729  0.897000   2.14849  1.77991
 22043 100 16  -0.9400 77.2500  1   7.24000  0.00000  1.000000   2.44265  2.52062
 22043 100 17  -0.8100 78.1300  1   9.31958  1.22695  1.000000   2.50212  3.53388

grep " 100 " sdp_D03256_S0040_E0154_B2204243_selection.asc

 22043 100 3   -2.7000 65.7100  1  -3.41849  3.73062  0.976000  -2.19302  1.28150
 22043 100 4   -2.5700 66.5900  2  -5.21503 -0.22769  0.044000  -1.28981  1.46249
 22043 100 5   -2.4400 67.4800  1  -1.32311  4.19635  0.961000  -0.75782  1.59957
 22043 100 6   -2.3000 68.3700  1   0.71196  4.03771  1.000000  -0.65922  1.53438
 22043 100 7   -2.1700 69.2500  1   0.55343  4.20373  1.000000  -0.87103  1.51783

 22043 100 8   -2.0300 70.1400  1  -0.17186  3.93625  1.000000  -1.34873  1.81343
 22043 100 9   -1.9000 71.0300  1   0.00000  3.72000  1.000000  -1.85392  2.20394
 22043 100 10  -1.7600 71.9200  1  -3.16193  2.21400  0.503000  -2.23444  2.40118
 22043 100 11  -1.6300 72.8100  1  -5.62347 -1.24669  0.969000  -1.90931  2.35275
 22043 100 12  -1.4900 73.7000  1  -5.87490 -1.85235  1.000000  -0.92908  2.18030

 22043 100 13  -1.3600 74.5800  2   4.66219  0.40789  0.459000   0.44006  1.98173
 22043 100 14  -1.2200 75.4700  1   4.51570  0.19716  0.602000   1.51566  1.60838
 22043 100 15  -1.0800 76.3600  1   5.43482  0.23729  0.897000   2.14849  1.77991
 22043 100 16  -0.9400 77.2500  1   7.24000  0.00000  1.000000   2.44265  2.52062
 22043 100 17  -0.8100 78.1300  1   9.31958  1.22695  1.000000   2.50212  3.53388
```

Table 5.4 Selection comparison for QDP (upper part) and SDP (lower part).

5.6 SDP 2.1 and SDP 2.2 winds versus ECMWF winds

In the scope of this Test Report, the wind statistics of QuikSCAT winds versus ECMWF model forecast winds have been compared.

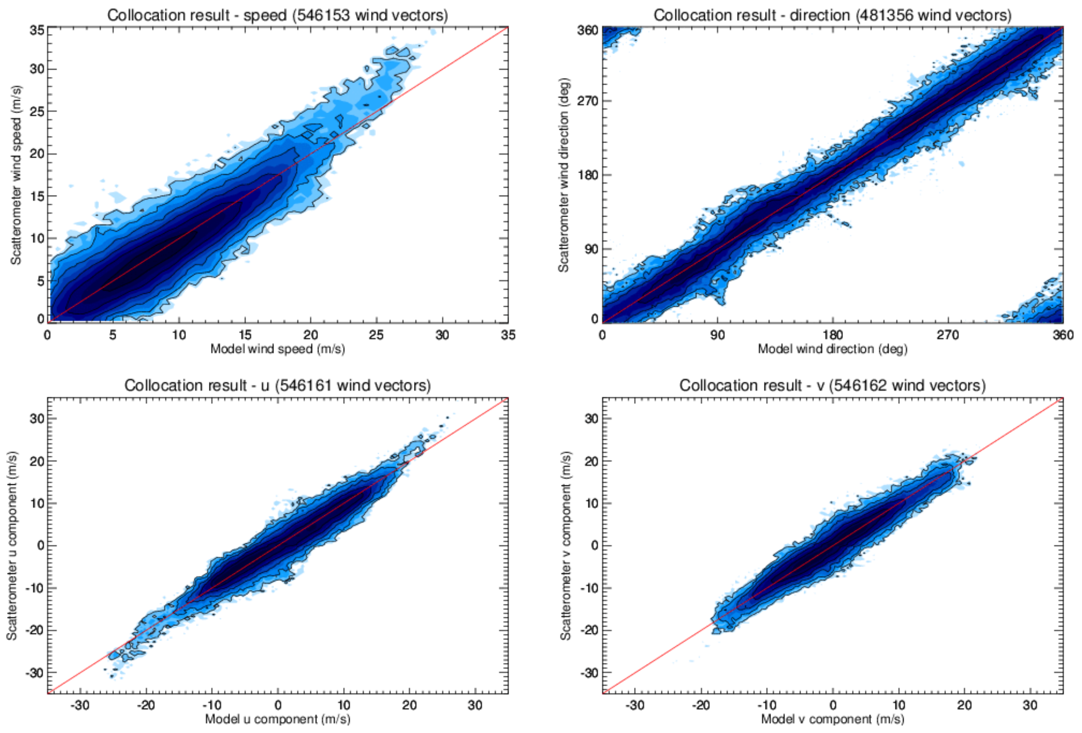

Figure 5.1 Comparison of QuikSCAT winds from SDP 2.1 with ECMWF model forecasts for wind speed, wind direction, u and v wind components.

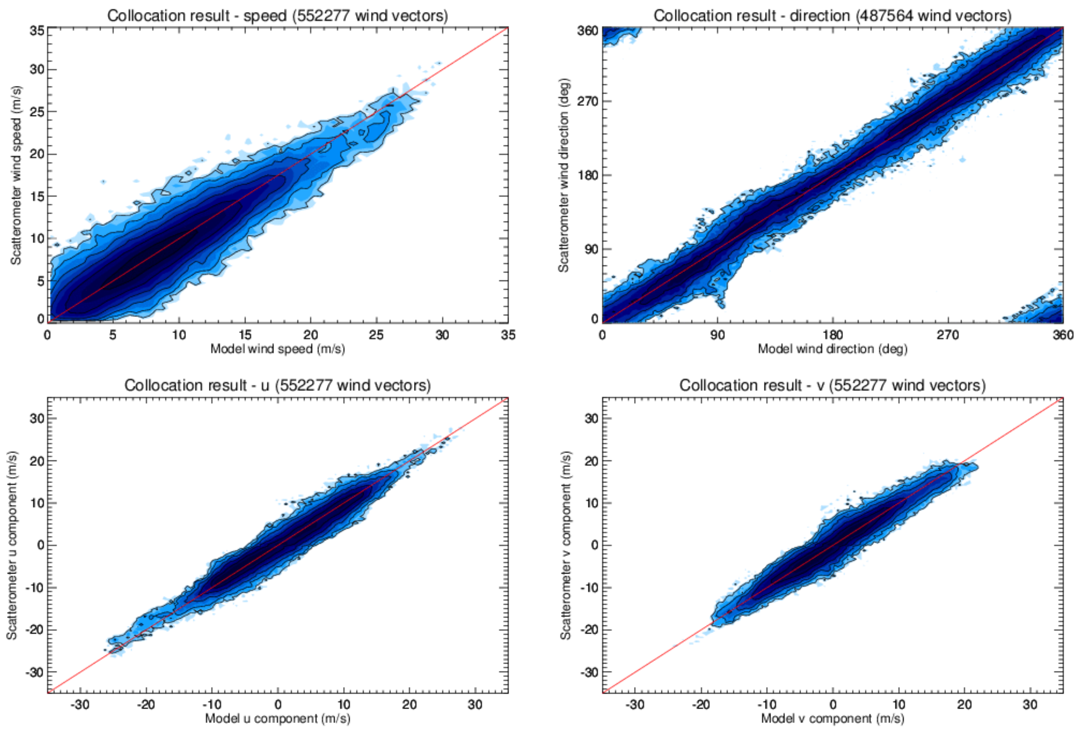

Figure 5.2 Comparison of QuikSCAT winds from SDP 2.2 with ECMWF model forecasts for wind speed, wind direction, u and v wind components.

This was done using all QuikSCAT data of 1st January 2008, both for SDP 2.1 and SDP 2.2. We compared the QuikSCAT winds from SDP with ECMWF forecast winds from the operational model (+3 to +21 hours forecasts from the 00 UTC and 12 UTC runs). Figure 5.1 and 5.2 show the collocations of the QuikSCAT and ECMWF winds from SDP 2.1 and SDP 2.2, respectively. Contoured histograms are shown for wind speed, wind direction and u and v wind components and after rejection of Quality Controlled (KNMI QC flagged) wind vectors. Note that the ECMWF winds are real 10m winds, whereas the scatterometer winds are equivalent neutral 10m winds, which are on average 0.2 m/s higher. In the wind direction plots, only those wind vectors where the model wind speed is at least 4 m/s are taken into account. The bin sizes for the histograms are 0.5 m/s for wind speed, u and v, and 2.5° for wind direction.

From the contour plots it is clear that wind speed biases are generally low. For SDP 2.1, we see an increasing bias for winds above 15 m/s, which is reduced in SDP 2.2, due to the use of the new NSCAT-4 Geophysical Model Function. We obtain wind component standard deviations of 1.42 in u and 1.40 in v directions for SDP 2.1. For SDP 2.2, these numbers are reduced to 1.35 in u and 1.37 in v directions. Moreover, the number of accepted wind vectors in SDP 2.2 is increased by approximately 1% from 546162 to 552277. So SDP 2.2 not only shows lower standard deviations, but also yields more accepted wind vectors from the same data set. This is due to the improvements in Quality Control and wind retrieval at low wind speeds. The overall wind speed bias was reduced from -0.25 m/s to -0.09 m/s due to the application of NWP Ocean Calibration in SDP 2.2.

Chapter 6

Portability tests

The SDP program inherits its portability by using strict Fortran 90 code (with a few low level routines for reading and writing binary in C). SDP is delivered with a complete make system. The Makeoptions include file of genscat takes care of the different settings needed under various systems. This Makeoptions file is also tested for the other NWP SAF scatterometer processors.

The default platform for development is a LINUX work station. Different Fortran 90 compilers were used to compile both genscat and SDP. Table 6.1 provides an overview of the platforms and compilers on which SDP has been tested successfully.

Platform	**Operating system**	**Fortran compiler**
HP workstation	Fedora LINUX	GNU g95, GCC gfortran, Portland f90
SUN workstation	SUN OS UNIX	Sun Fortran
Compaq workstation	Tru64 UNIX	Compaq Fortran
SGI workstation	IRIX64 UNIX	MIPSpro

Table 6.1 Supported platforms and compilers for SDP.

In principle, SDP can also run under Windows provided a suitable Linux environment like Wubi is used, so that the environmental variables can be set and the BUFR library properly installed.

Chapter 7

User documentation tests

The SDP package versions 1.0 and 2.0 and their user documentation have been provided to beta testers for review. More specifically, SDP 2.0 was tested by F. van Geffen and by the OSI SAF (A. Verhoef). Only minor issues in the documentation were found which have been implemented in the user documentation. Beta test reports have been delivered for review with SDP version 2.0.

Since SDP version 2.2 is a minor release with no significant changes in functionality, no beta testing was done for this version.

The C shell scripts in the genscat make system were translated to Bourne shell which is supported on all UNIX and LINUX systems and emulators. After some adjustments, installation following the instructions ran smoothly.