NWP SAF

Satellite Application Facility
for Numerical Weather Prediction

Document NWPSAF-TV-002

Version 1.2

27-03-2006

SDP Test Report

Scat group

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

KNMI, De Bilt, The Netherlands

SDP Test Report
Scat group

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

KNMI, De Bilt, The Netherlands

This documentation was developed within the context of the EUMETSAT Satellite Application Facility on Numerical Weather Prediction (NWP SAF), under the Cooperation Agreement dated 16 December, 2003, between EUMETSAT and the Met Office, UK, by one or more partners within the NWP SAF. The partners in the NWP SAF are the Met Office, ECMWF, KNMI and Météo France.

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

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

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 *SCAT group* [2005].

SDP is developed within the NWP SAF IOP 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

Table 1.1 SDP releases.

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 independent 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 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.

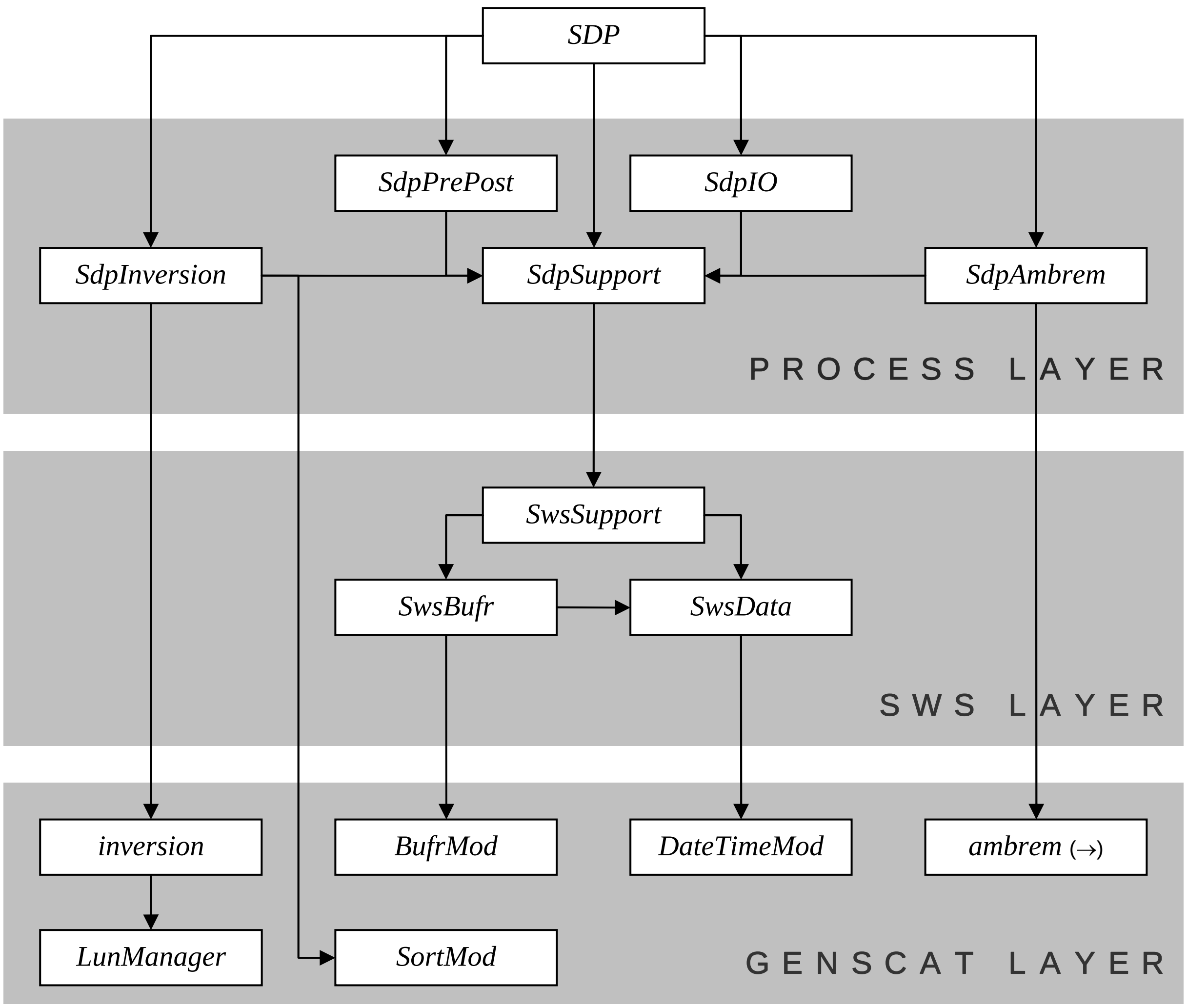

Figure 1.1 SDP Module layers and top level module dependencies. The dependencies for module *ambrem* are continued in figure 1.2.

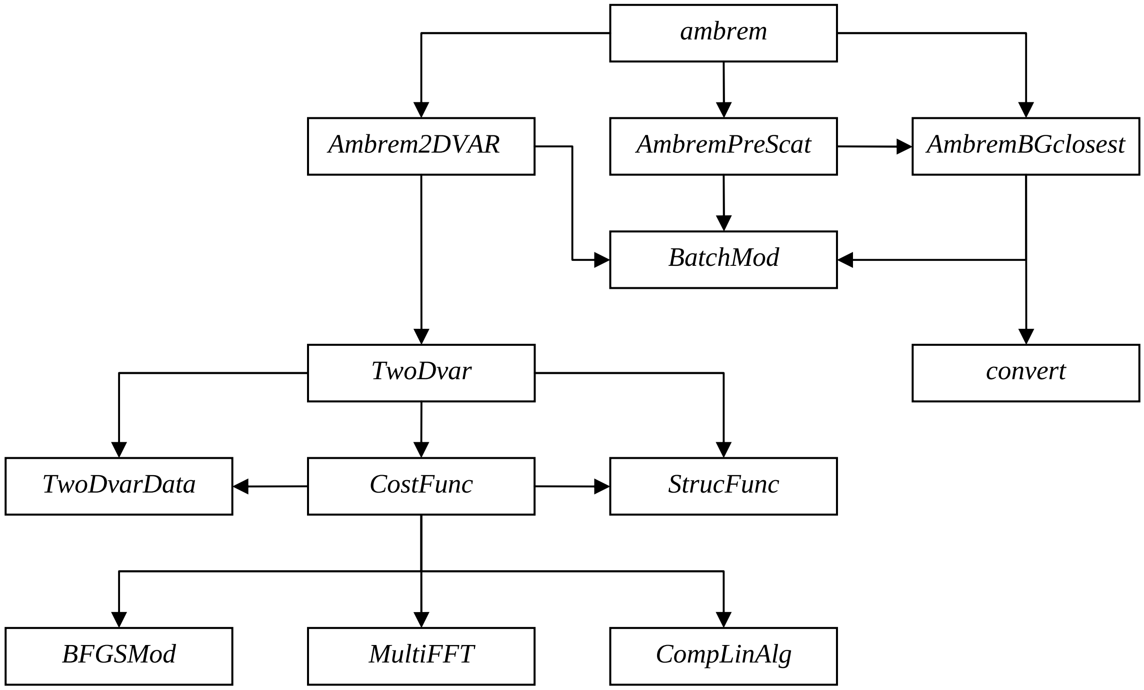

Figure 1.2 Interdependence of the modules for Ambiguity Removal. The direct connection from module *ambrem* to module *BatchMod* is not drawn.

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). Moreover, KNMI runs an experimental suite in the framework of the OSI SAF, where SDP, in different modes, is routinely compared to the operational OSI SAF suite at www.knmi.nl/scatterometer . This comparison is both field-wise and statistical.

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 distinghuishes:

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 is done on several platforms (operating systems) and with different Fortran 90 compilers. The integration and validation tests were done on both a LINUX work station and a SUN machine (SunOS 5.9).

Chapter 2 contains the test of the adjoint method for calculating the gradient of an objective function. Note that a numerical method is tested rather than its software implementation. This test requires software which is not necessary for the operation of SDP, and is therefore left out of the SDP 1.2 release.

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 1.2 release.

Chapter 4 describes 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. As SDP 1.2 is the first public release, it is not possible yet to test how the SDP User Manual and Reference Guide 1.2 is appreciated by others than the beta testers.

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:

- SCAT group, *SDP User Manual and Reference Guide*, *version 1.2*, KNMI, March 2006.

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 1.2 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 adjoint method for calculating the gradient of a function in section 2.1, and for selection of a minimization routine in section 2.2. These methods are used in the KNMI 2DVAR Ambiguity Removal scheme.

2.1 Adjoint method

The common manner to test the adjoint method for the calculation of the gradient of an objective function, f, is to compare the calculated gradient, g, with a finite difference approximation as given by

$$\delta f = f(x+\delta x) - f(x) \approx g(x)\delta x + O(\delta x)^2 \quad . \tag{2.1}$$

Parameter	**Type**	**Description**
lbl	character	Label (e.g. function name)
fge	subroutine	Function gradient evaluation
xlo	real array	Lower bounds of multidimensional box)
xhi	real array	Upper bounds of multidimensional box
maxx	integer	Number of start points
maxopt	integer	Number of iterations for steepest descend method

Table 2.1 Arguments of routine *GradTest*.

This test is implemented in the routine *GradTest* of the module *GradientTtest* in directory `genscat/support/adjoint` (not included in SDP 1.2 release). This routine has six arguments, listed in table 2.1.

The routine *GradTest* tests the gradient of a objective function in a specified range, the multi-dimensional box (*xlo*,*xhi*). Randomly, a point within the box is selected. This point is the start of the steepest descent method to minimise the objective function. For every gradient evaluation during the steepest descent method, the implemented gradient calulation and the finite difference approximation are compared.

The validity of the test itself is checked by the program *TestGradienTest*. This test program applies *GradTest* for a simple test function with a known gradient and yields the output of table 2.2. The third to last columns contain, respectively, the relative error $[\delta f - g(x)\delta x]/\delta f$, the absolute error $\delta f - g(x)\delta x$, the first order approximation of the objective function difference $g(x)\delta x$, the objective function difference δf, the function value $f_0 = f(x)$, the norm of x, the norm of g, and finally the norm of δx.

```
grad test:   rel err    df-g*dx       g*dx       df         f0  normx  normg  normh
grad test:  0.0136691  0.0012813    0.09246    0.09374   20.2956  0.052  0.174  0.001
grad test:  0.0006029 -0.0000546    0.09059    0.09053    9.7149  0.048  0.086  0.001
grad test:  0.0005131  0.0000193    0.03762    0.03764    4.7605  0.044  0.045  0.000
grad test:  0.0000248  0.0000003    0.01155    0.01155    2.3718  0.040  0.024  0.000
grad test:  0.0000968  0.0000003    0.00360    0.00360    1.1968  0.036  0.013  0.000
grad test:  0.0000764  0.0000001    0.00099    0.00099    0.6100  0.033  0.007  0.000
grad test:  0.0008429  0.0000001    0.00009    0.00009    0.3134  0.030  0.004  0.000
grad test:  0.0034976  0.0000000   -0.00001   -0.00001    0.1621  0.027  0.002  0.000
grad test:  0.0003228  0.0000000    0.00003    0.00003    0.0843  0.024  0.001  0.000
grad test:  0.0008172  0.0000000    0.00002    0.00002    0.0441  0.022  0.001  0.000

grad test:   rel err    df-g*dx       g*dx       df         f0  normx  normg  normh
grad test:  0.0187458 -0.0023816   -0.12466   -0.12704   14.9839  0.050  0.136  0.001
grad test:  0.0009779  0.0003007    0.30721    0.30751    7.1201  0.046  0.065  0.001
grad test:  0.0003392  0.0000103    0.03033    0.03034    3.5045  0.042  0.034  0.000
grad test:  0.0009384 -0.0000027    0.00286    0.00286    1.7629  0.039  0.018  0.000
grad test:  0.0001735  0.0000001    0.00055    0.00055    0.8976  0.035  0.010  0.000
grad test:  0.0046713 -0.0000001   -0.00002   -0.00002    0.4602  0.032  0.006  0.000
grad test:  0.0002819  0.0000000    0.00013    0.00013    0.2371  0.029  0.003  0.000
grad test:  0.0000725  0.0000000    0.00009    0.00009    0.1226  0.026  0.002  0.000
grad test:  0.0008287  0.0000000    0.00002    0.00002    0.0637  0.024  0.001  0.000
grad test:  0.0003175  0.0000000    0.00001    0.00001    0.0332  0.021  0.001  0.000
```

Table 2.2 Output of program *TestGradientTest.*

2.2 Minimisation 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 NCEP 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 2.3 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	Precisison (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 2.3 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 2.3 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.

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 1.2.

Module tests have been included in SDP 1.2 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.

This may change in future updates of SDP.

Module name	**Location**	**Test program**	**Included in SDP 1.2**
BFGSMod	`genscat/support/BFGS`	*Test_BFGS*	yes
BufrMod	`genscat/support/bufr`	*test_modules*	yes
CompLinAlg	`genscat/ambrem/twodvar`	*CompLinAlgTest*	no
convert	`genscat/support/convert`	*test_convert*	yes
CostFunct	`genscat/ambrem/twodvar`	*CostFuncTest*	no
DateTimeMod	`genscat/support/datetime`	*TestDateTimeMod*	yes
ErrorHandler	`genscat/support/ErrorHandler`	*TestErrorHandler*	yes
LunManager	`genscat/support/file`	*TestLunManager*	yes
MultiFFT	`genscat/support/multifft`	*MultiFFTtest*	no
numerics	`genscat/support/num`	*test_numerics*	yes
SortMod	`genscat/support/sort`	*SortModTest*	yes
SwsData	`sdp/sws`	*SwsDataTest*	no

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

Routine LBFGS completed succesfully
  Number of iterations                 :      95
  Dimension of problem                 :  100000
  Number of corrections in BFGS update :       5
  Cost function at start               :  0.20001D+25
  Cost function at end                 :  0.33346D-16
  Precision required                   :  0.10D-19
  Minimum and Maximum error in solution :  0.000004  0.000006
  Time needed                          :  9.687  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 -  6.1
            07 JULy 2003.


 Your path for bufr tables is :
 /usr/people/vogelzan/Software/genscat/support/bufr/bufr_tables/
BUFR TABLES TO BE LOADED  B002100001 ,D002100001
 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 -  6.1
            07 JULY 2003.


 Your path for bufr tables is :
 /usr/people/vogelzan/Software/genscat/support/bufr/bufr_tables/
 BUFR TABLES TO BE LOADED B002100001 ,D002100001
```

Table 3.3 Output of program *test_modules.*

3.3 Module *CompLinAlg* (genscat ambrem)

Directory `genscat/ambrem/twodvar` contains module *CompLinAlg.F90* which contains several routines to apply elementary linear algebra on complex fields. Program *CompLinAlgTest* applies routine *gradtest*, see section 2.1, to check the adjoint calculations for these routines. Table 3.4 lists the typical output of this program. For *Cprod2D* the objective function is the squared norm of the output field. The quantities listed in table 3.4 are already described in section 2.1.

Note from the '`F0`' column that minimisation by the steepest descent method is not monotonic for the *Cdot2d* routine. However, the 2DVAR cost function uses only *Cprod2D* and *Cnorm2d*. For these routines the minimisation is monotonic.

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 3.4; those for p = 10°, 45°, and 70° are omitted. Program *test_convert* ends with some trigoniometric calculations on a sphere.

```
IMAG PART ZERO
Cprod2DTest : grad test:   rel err    df-g*dx      g*dx        df          f0   normx  normg  normh
Cprod2DTest : grad test:  0.0002863  0.0004226  -1.47637   -1.47595  1218.8365  0.282  0.190  0.001
Cprod2DTest : grad test:  0.0000996 -0.0002375  -2.38312   -2.38336   846.8969  0.274  0.142  0.001
Cprod2DTest : grad test:  0.0002508 -0.0001563   0.62320    0.62305   576.7486  0.267  0.106  0.001
Cprod2DTest : grad test:  0.0004334 -0.0000486  -0.11207   -0.11212   381.8311  0.260  0.078  0.000
Cprod2DTest : grad test:  0.0005004  0.0000656  -0.13123   -0.13116   243.0038  0.254  0.056  0.000

Cdot2dTest : grad test:   rel err    df-g*dx      g*dx        df          f0   normx  normg  normh
Cdot2dTest : grad test:  0.0000043  0.0000000   0.01150    0.01150     0.6270  0.143  0.016  0.000
Cdot2dTest : grad test:  0.0002437 -0.0004441   1.82288    1.82243   107.7788  0.143  0.204  0.001
Cdot2dTest : grad test:  0.0001900 -0.0000012   0.00654    0.00653     0.5965  0.142  0.015  0.000
Cdot2dTest : grad test:  0.0001727  0.0005352  -3.10000   -3.09946   106.3849  0.142  0.201  0.001
Cdot2dTest : grad test:  0.0000416  0.0000006   0.01466    0.01466     0.5669  0.142  0.015  0.000

Cnorm2dTest : grad test:   rel err    df-g*dx      g*dx        df          f0   normx  normg  normh
Cnorm2dTest : grad test:  0.0000096 -0.0000014  -0.14374   -0.14374    32.8828  0.040  0.057  0.000
Cnorm2dTest : grad test:  0.0003107  0.0000040  -0.01301   -0.01301    28.4107  0.039  0.053  0.000
Cnorm2dTest : grad test:  0.0001086  0.0000077  -0.07109   -0.07108    24.4042  0.038  0.049  0.000
Cnorm2dTest : grad test:  0.0007074 -0.0000104  -0.01470   -0.01471    20.8211  0.036  0.046  0.000
Cnorm2dTest : grad test:  0.0004213 -0.0000072  -0.01697   -0.01697    17.6235  0.035  0.042  0.000

REAL PART ONLY
Cprod2DTest : grad test:   rel err    df-g*dx      g*dx        df          f0   normx  normg  normh
Cprod2DTest : grad test:  0.0006843 -0.0007869   1.15069    1.14990  1953.9215  0.296  0.304  0.001
Cprod2DTest : grad test:  0.0009051 -0.0006673   0.73797    0.73730  1324.9117  0.288  0.230  0.001
Cprod2DTest : grad test:  0.0018441  0.0004174   0.22590    0.22632   863.0936  0.282  0.172  0.001
Cprod2DTest : grad test:  0.0000135 -0.0000014  -0.10718   -0.10718   529.5201  0.276  0.125  0.001
Cprod2DTest : grad test:  0.0000854  0.0000141  -0.16478   -0.16476   296.9749  0.271  0.087  0.000

Cdot2dTest : grad test:   rel err    df-g*dx      g*dx        df          f0   normx  normg  normh
Cdot2dTest : grad test:  0.0002947 -0.0000175  -0.05922   -0.05924     5.4027  0.145  0.067  0.000
Cdot2dTest : grad test:  0.0016343  0.0005801  -0.35553   -0.35495   209.0368  0.145  0.419  0.002
Cdot2dTest : grad test:  0.0002157  0.0000338   0.15658    0.15662     5.3489  0.145  0.067  0.000
Cdot2dTest : grad test:  0.0014463  0.0095515  -6.61357   -6.60402   206.9541  0.145  0.416  0.002
Cdot2dTest : grad test:  0.0008179 -0.0000054   0.00664    0.00664     5.2956  0.144  0.066  0.000

Cnorm2dTest : grad test:   rel err    df-g*dx      g*dx        df          f0   normx  normg  normh
Cnorm2dTest : grad test:  0.0000450 -0.0000134   0.29655    0.29653    71.1709  0.042  0.084  0.001
Cnorm2dTest : grad test:  0.0000190  0.0000027  -0.14467   -0.14467    64.2317  0.040  0.080  0.001
Cnorm2dTest : grad test:  0.0001275 -0.0000191   0.15003    0.15001    57.9691  0.038  0.076  0.001
Cnorm2dTest : grad test:  0.0001184 -0.0000074   0.06249    0.06248    52.3171  0.036  0.072  0.000
Cnorm2dTest : grad test:  0.0001668 -0.0000012  -0.00693   -0.00694    47.2162  0.034  0.069  0.000

FULLY COMPLEX
Cprod2DTest : grad test:   rel err    df-g*dx      g*dx        df          f0   normx  normg  normh
Cprod2DTest : grad test:  0.0005581 -0.0010246  -1.83491   -1.83594  4105.1465  0.286  0.434  0.002
Cprod2DTest : grad test:  0.0001649 -0.0012112  -7.34181   -7.34302  3204.6743  0.275  0.355  0.002
Cprod2DTest : grad test:  0.0001202  0.0002220  -1.84641   -1.84619  2494.7507  0.263  0.293  0.001
Cprod2DTest : grad test:  0.0002643  0.0004582   1.73306    1.73352  1932.9487  0.253  0.242  0.001
Cprod2DTest : grad test:  0.0000454 -0.0001168   2.57604    2.57593  1487.5677  0.243  0.200  0.001

Cdot2dTest : grad test:   rel err    df-g*dx      g*dx        df          f0   normx  normg  normh
Cdot2dTest : grad test:  0.0003320  0.0022798  -6.86805   -6.86577   203.0999  0.143  0.407  0.002
Cdot2dTest : grad test:  0.0002575 -0.0000888  -0.34494   -0.34503     4.2470  0.143  0.059  0.000
Cdot2dTest : grad test:  0.0023979 -0.0120592  -5.01712   -5.02917   201.0766  0.143  0.404  0.002
Cdot2dTest : grad test:  0.0001542 -0.0000222   0.14418    0.14416     4.2047  0.142  0.058  0.000
Cdot2dTest : grad test:  0.0005100  0.0048532  -9.52065   -9.51579   199.0735  0.142  0.401  0.002

Cnorm2dTest : grad test:   rel err    df-g*dx      g*dx        df          f0   normx  normg  normh
Cnorm2dTest : grad test:  0.0000221 -0.0000103  -0.46621   -0.46622    67.2394  0.041  0.082  0.001
Cnorm2dTest : grad test:  0.0000871  0.0000123  -0.14158   -0.14156    60.6836  0.039  0.078  0.000
Cnorm2dTest : grad test:  0.0000474  0.0000097  -0.20376   -0.20375    54.7669  0.037  0.074  0.001
Cnorm2dTest : grad test:  0.0000112  0.0000049  -0.43531   -0.43531    49.4272  0.035  0.070  0.001
Cnorm2dTest : grad test:  0.0000375  0.0000100  -0.26786   -0.26785    44.6080  0.033  0.067  0.000
```

Table 3.4 Output of program *CompLinAlgTest*.

```
===========================================
 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 =     5.000002      v =    -6.999999
 ===========================================
 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 =     55.00000        5.000000
 ===========================================
         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  106.16298  270.00000    0.00000
   0.00000  -90.00000   15.00000    0.00000    0.00000   -1.00000  -90.00000  105.59795  270.00000    9.72975
   0.00000  -90.00000   30.00000    0.00000    0.00000   -1.00000  -90.00000  103.95005  270.00000   27.91061
   0.00000  -90.00000   45.00000    0.00000    0.00000   -1.00000  -90.00000  101.35209  270.00000   43.81981
   0.00000  -90.00000   60.00000    0.00000    0.00000   -1.00000  -90.00000   98.00070  270.00000   59.32336
   0.00000  -10.00000    0.00000    0.98481    0.00000   -0.17365  -10.00000    0.00000  190.00000  180.00000
   0.00000  -10.00000   15.00000    0.95125    0.25489   -0.17365  -10.00000   15.00000  190.00000 -164.99998
   0.00000  -10.00000   30.00000    0.85287    0.49240   -0.17365  -10.00000   30.00000  190.00000 -149.99998
   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
  90.00000   45.00000   30.00000    0.25882    0.96593    0.00000   74.99999    0.00000  105.00000    0.00000
  90.00000   45.00000   45.00000    0.00000    1.00000    0.00000   90.00000    0.00000   90.00000    0.00000
  90.00000   45.00000   60.00000   -0.25882    0.96593    0.00000   74.99999    0.00000  105.00000    0.00000
  90.00000   90.00000    0.00000    0.00000    1.00000    0.00000   90.00000    0.00000   90.00000    0.00000
  90.00000   90.00000   15.00000   -0.25882    0.96593    0.00000   74.99999    0.00000  105.00000    0.00000
  90.00000   90.00000   30.00000   -0.50000    0.86603    0.00000   59.99999    0.00000  120.00000    0.00000
  90.00000   90.00000   45.00000   -0.70711    0.70711    0.00000   45.00000    0.00000  135.00000    0.00000
  90.00000   90.00000   60.00000   -0.86603    0.50000    0.00000   30.00000    0.00000  149.99998    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
 ===========================================
```

Table 3.5 Output of program *test_convert*

3.5 Module *CostFunc* (genscat ambrem)

Module *CostFunc.F90* in directory `genscat/ambrem/twodvar` contains the cost function definition of the 2DVAR method. Program *CostFunctTest* applies routine *gradtest* introduced in section 2.1 to check the adjoint calculations for these routines. The cost function consists of several operations. To test these operations individually, several test routines are developed within the program *CostFunctTest*. These are listed in table 3.6. The tests are currently confined to gradient testing.

Routine	Description
SetBogusObs	Create bogus wind observations
Createzlozhi	Create bogus *z* field
Createyloyhi	Create bogus *y* field
testfrq2grd	Test FFT
Testpc2uv	Test (ψ,χ) conversion
Testz2y	Test *z* to *y* transformation
TestConvolute	Test convolution with covariances (structure functions)
TestCost	Test total cost function

Table 3.6 Routines in program *CostFuncTest*.

As program *CostFucTest* generates quite a lot of output, it is splitted over tables 3.7 - 3.11. Table 3.7 lists the output of program *CostFunctTest* for the tests of the individual transformations by routines *Testfrq2grd*, *Testpc2uv*, and *TestConvolute*. The header of the output has been removed in order to save space. Table 3.8 gives the test results for routine *Testz2y*

In many cases the objective function is the squared norm of the output field. In the '`F0`' column one observes that minimisation by the steepest descent method is problematic for the *convolute_test* This is a matter of further investigation. Maybe the steepest decent method is not robust enough or maybe the squared norm of the output field is a too difficult objective function. The main routine in the convolution is the complex product *Cprod*. The adjoint method works fine for this routine, see section 2.1, so maybe the very definition of the structure functions makes this operation (or the test of it) 'ill conditioned'. This point needs further investigation.

Using simplified forms for the cost function does not seem to help the problem (no results shown), so probably the main problem here is the 'condition' of the convolution. Further investigations are needed to solve this issue.

The gradients of J_b, J_o, and J_t are tested for a set bogus wind observations created by a random generator. The results are shown in tables 3.9, 3.10, and 3.11, respectively. Convergence is quite slow, especially for J_o. Program *CostFuncTest* takes quite a lot of time and produces a lot of output, so only the first and last lines of the output are shown.

```
frq2grd_test : grad test:   rel err    df-g*dx       g*dx        df          f0  normx  normg  normh
frq2grd_test : grad test:  0.0193613 -0.0035452    0.18665    0.18311  3382.6235  0.014  0.028  0.000
frq2grd_test : grad test:  0.0015492  0.0002421   -0.15649   -0.15625  3052.8152  0.013  0.027  0.000
frq2grd_test : grad test:  0.1576377  0.0003849    0.00206    0.00244  2755.1680  0.013  0.026  0.000
frq2grd_test : grad test:  0.0021031 -0.0002788    0.13285    0.13257  2486.5396  0.012  0.024  0.000
frq2grd_test : grad test:  0.0003149  0.0001428    0.45323    0.45337  2244.1018  0.012  0.023  0.000
frq2grd_test : grad test:  0.0055563  0.0008865    0.15866    0.15955  2025.3018  0.011  0.022  0.000
frq2grd_test : grad test:  0.0010655 -0.0001145    0.10754    0.10742  1827.8340  0.010  0.021  0.000
frq2grd_test : grad test:  0.0018569 -0.0000753    0.04060    0.04053  1649.6205  0.010  0.020  0.000
frq2grd_test : grad test:  0.0050523 -0.0006383   -0.12570   -0.12634  1488.7828  0.009  0.019  0.000
frq2grd_test : grad test:  0.0042125 -0.0007364   -0.17407   -0.17480  1343.6262  0.009  0.018  0.000
frq2grd_test : grad test:  0.0206779  0.0004594   -0.02268   -0.02222  1212.6228  0.009  0.017  0.000
frq2grd_test : grad test:  0.0001206  0.0000158    0.13109    0.13110  1094.3920  0.008  0.016  0.000
frq2grd_test : grad test:  0.0181253 -0.0008961    0.05033    0.04944   987.6889  0.008  0.015  0.000
frq2grd_test : grad test:  0.0001693 -0.0000242    0.14303    0.14301   891.3892  0.007  0.015  0.000
frq2grd_test : grad test:  0.0518942 -0.0002977    0.00604    0.00574   804.4791  0.007  0.014  0.000
frq2grd_test : grad test:  0.0017067 -0.0001341    0.07869    0.07855   726.0420  0.007  0.013  0.000
frq2grd_test : grad test:  0.0021839  0.0002121    0.09689    0.09711   655.2530  0.006  0.012  0.000
frq2grd_test : grad test:  0.0184951  0.0005531   -0.03046   -0.02991   591.3658  0.006  0.012  0.000
frq2grd_test : grad test:  0.0038559 -0.0000532   -0.01374   -0.01379   533.7076  0.006  0.011  0.000
frq2grd_test : grad test:  0.0008648 -0.0000313    0.03626    0.03622   481.6713  0.005  0.011  0.000

pc2uv_test : grad test:   rel err    df-g*dx       g*dx        df          f0  normx  normg  normh
pc2uv_test : grad test:  0.0018457  0.0025361   -1.37656   -1.37402  3667.7791  0.014  0.033  0.000
pc2uv_test : grad test:  0.0025290 -0.0012695    0.50322    0.50195  3296.0659  0.013  0.031  0.000
pc2uv_test : grad test:  0.0006426  0.0005324   -0.82915   -0.82861  2961.7024  0.013  0.029  0.000
pc2uv_test : grad test:  0.0033304 -0.0011619    0.35004    0.34888  2660.9727  0.012  0.028  0.000
pc2uv_test : grad test:  0.0196969  0.0009521   -0.04929   -0.04834  2390.5325  0.011  0.026  0.000
pc2uv_test : grad test:  0.0009160 -0.0004743   -0.51735   -0.51782  2147.3613  0.011  0.025  0.000
pc2uv_test : grad test:  0.0022340  0.0008566   -0.38428   -0.38342  1928.7371  0.010  0.023  0.000
pc2uv_test : grad test:  0.0023766 -0.0004416   -0.18535   -0.18579  1732.2119  0.010  0.022  0.000
pc2uv_test : grad test:  0.0010114 -0.0003732   -0.36865   -0.36902  1555.5776  0.009  0.021  0.000
pc2uv_test : grad test:  0.0141871 -0.0003273    0.02340    0.02307  1396.8420  0.009  0.020  0.000
pc2uv_test : grad test:  0.0443345  0.0004492    0.00968    0.01013  1254.2107  0.009  0.018  0.000
pc2uv_test : grad test:  0.0010642  0.0001277   -0.12012   -0.12000  1126.0687  0.008  0.017  0.000
pc2uv_test : grad test:  0.0053895 -0.0005303   -0.09786   -0.09839  1010.9600  0.008  0.016  0.000
pc2uv_test : grad test:  0.0018416 -0.0002962   -0.16053   -0.16083   907.5718  0.007  0.015  0.000
pc2uv_test : grad test:  0.0024522 -0.0002471   -0.10052   -0.10077   814.7239  0.007  0.015  0.000
pc2uv_test : grad test:  0.0005276 -0.0000161    0.03053    0.03052   731.3524  0.007  0.014  0.000
pc2uv_test : grad test:  0.0033893 -0.0005575   -0.16393   -0.16449   656.4987  0.006  0.013  0.000
pc2uv_test : grad test:  0.0041906 -0.0002857    0.06846    0.06818   589.3011  0.006  0.012  0.000
pc2uv_test : grad test:  0.0032683 -0.0001117   -0.03407   -0.03418   528.9824  0.006  0.011  0.000
pc2uv_test : grad test:  0.0054335  0.0003520   -0.06514   -0.06479   474.8441  0.006  0.011  0.000

convolute_test : grad test:   rel err    df-g*dx       g*dx        df          f0  normx  normg  normh
convolute_test : grad test:  0.0053609 -0.0006479    0.12150    0.12085  1220.2201  0.013  0.020  0.000
convolute_test : grad test:  0.0003667  0.0000944   -0.25742   -0.25732  1027.3759  0.012  0.018  0.000
convolute_test : grad test:  0.0000552 -0.0000445    0.80669    0.80664   872.5594  0.012  0.020  0.000
convolute_test : grad test:  0.0001460 -0.0003996   -2.73617   -2.73657   857.8073  0.012  0.047  0.000
convolute_test : grad test:  0.0000570 -0.0002010    3.52852    3.52832   805.6158  0.012  0.046  0.000
convolute_test : grad test:  0.0000006 -0.0000024   -4.11920   -4.11920   780.3425  0.011  0.050  0.000
convolute_test : grad test:  0.0000138 -0.0000463    3.35716    3.35712   734.6864  0.011  0.048  0.000
convolute_test : grad test:  0.0002790 -0.0003188   -1.14232   -1.14264   710.2282  0.011  0.050  0.000
convolute_test : grad test:  0.0000013 -0.0000057    4.49097    4.49097   671.2653  0.011  0.048  0.000
convolute_test : grad test:  0.0000493 -0.0002398   -4.86780   -4.86804   648.4548  0.011  0.050  0.000
convolute_test : grad test:  0.0000108  0.0000262    2.42423    2.42426   614.6799  0.011  0.048  0.000
convolute_test : grad test:  0.0003475 -0.0004107   -1.18147   -1.18188   593.5391  0.011  0.049  0.000
convolute_test : grad test:  0.0004246  0.0004183    0.98487    0.98529   563.8976  0.011  0.048  0.000
convolute_test : grad test:  0.0001864 -0.0002078   -1.11472   -1.11493   544.4073  0.011  0.049  0.000
convolute_test : grad test:  0.0000047 -0.0000055    1.15967    1.15967   518.1990  0.011  0.048  0.000
convolute_test : grad test:  0.0000056 -0.0000060   -1.06744   -1.06744   500.3280  0.011  0.049  0.000
convolute_test : grad test:  0.0000024  0.0000050    2.11822    2.11823   477.0464  0.011  0.048  0.000
convolute_test : grad test:  0.0004378  0.0002236   -0.51094   -0.51071   460.7380  0.011  0.048  0.000
convolute_test : grad test:  0.0001599  0.0003207    2.00487    2.00519   439.9901  0.011  0.047  0.000
convolute_test : grad test:  0.0004077  0.0003791   -0.93028   -0.92990   425.1633  0.011  0.048  0.000
```

Table 3.7 First part of the output of program *CostFuncTest* with the tests of routines *Testfrq2grd*, *Testpc2uv*,and *TestConvolute.*

	rel err	df-g*dx	g*dx	df	f0	normx	normg	normh
z2y_test : grad test:	0.0001256	0.0000803	0.63908	0.63916	902.9109	0.012	0.024	0.000
z2y_test : grad test:	0.0230545	0.0003433	0.01455	0.01489	688.4641	0.011	0.021	0.000
z2y_test : grad test:	0.0005491	0.0000173	0.03148	0.03149	508.3807	0.011	0.017	0.000
z2y_test : grad test:	0.0005475	-0.0001033	0.18876	0.18866	360.0691	0.011	0.014	0.000
z2y_test : grad test:	0.0014486	0.0001425	-0.09852	-0.09837	241.4145	0.011	0.011	0.000
z2y_test : grad test:	0.0174897	-0.0000798	0.00464	0.00456	150.6736	0.011	0.009	0.000
z2y_test : grad test:	0.0022473	0.0000274	-0.01221	-0.01218	86.3643	0.011	0.006	0.000
z2y_test : grad test:	0.0004614	0.0000004	0.00091	0.00091	47.0364	0.011	0.003	0.000
z2y_test : grad test:	0.1109104	0.0000243	0.00020	0.00022	30.3155	0.010	0.001	0.000
z2y_test : grad test:	0.2844236	0.0000125	-0.00006	-0.00004	24.2754	0.010	0.001	0.000
z2y_test : grad test:	0.0099582	0.0000116	0.00115	0.00116	24.0884	0.010	0.001	0.000
z2y_test : grad test:	0.0025717	0.0000039	-0.00152	-0.00152	24.3489	0.010	0.002	0.000
z2y_test : grad test:	0.0011396	-0.0000220	0.01935	0.01933	30.0720	0.010	0.004	0.000
z2y_test : grad test:	0.0009528	0.0000106	0.01114	0.01115	29.7102	0.010	0.005	0.000
z2y_test : grad test:	0.0014341	-0.0000193	0.01349	0.01347	33.5914	0.010	0.006	0.000
z2y_test : grad test:	0.0012917	-0.0000253	-0.01955	-0.01957	30.0517	0.010	0.005	0.000
z2y_test : grad test:	0.0004434	0.0000055	0.01245	0.01246	33.4120	0.010	0.006	0.000
z2y_test : grad test:	0.0009454	-0.0000103	-0.01091	-0.01093	29.9022	0.010	0.006	0.000
z2y_test : grad test:	0.0013313	-0.0000040	0.00302	0.00302	33.2813	0.010	0.006	0.000
z2y_test : grad test:	0.0028686	-0.0000070	-0.00244	-0.00245	29.8128	0.010	0.006	0.000

Table 3.8 Second part of the output of program *CostFuncTest* with the test of routine *Testz2y*.

```
setbogusobs [0]
      jrowmax:            104
      inodemax:            76
 setbogusobs [1]
```

	rel err	df-g*dx	g*dx	df	f0	normx	normg	normh
Jb : grad test:	0.0001582	0.0047932	30.29501	30.29980	8810.0977	0.044	0.086	0.000
Jb : grad test:	0.0001508	0.0038929	25.81740	25.82129	7925.1416	0.042	0.082	0.000
Jb : grad test:	0.0000339	0.0009308	27.44389	27.44482	7126.5098	0.040	0.078	0.000
Jb : grad test:	0.0002405	-0.0052910	22.00334	21.99805	6405.7959	0.038	0.074	0.000
Jb : grad test:	0.0000528	0.0010967	20.78552	20.78662	5755.4102	0.037	0.070	0.000
Jb : grad test:	0.0000115	-0.0001822	15.88495	15.88477	5168.5010	0.035	0.066	0.000
Jb : grad test:	0.0002740	0.0040121	14.63612	14.64014	4638.8809	0.033	0.063	0.000
Jb : grad test:	0.0001014	-0.0014229	14.03316	14.03174	4160.9785	0.032	0.059	0.000
… … … …	… …	… … …	… … …	… … …	… …	… …	… …	…
Jb : grad test:	0.0000646	0.0000002	-0.00309	-0.00308	0.8904	0.011	0.001	0.000
Jb : grad test:	0.0000040	0.0000000	0.00019	0.00019	0.0557	0.011	0.000	0.000
Jb : grad test:	0.0000336	0.0000001	-0.00283	-0.00283	0.8887	0.011	0.001	0.000
Jb : grad test:	0.0000802	0.0000000	0.00019	0.00019	0.0561	0.011	0.000	0.000
Jb : grad test:	0.0003499	-0.0000011	-0.00328	-0.00328	0.8871	0.011	0.001	0.000
Jb : grad test:	0.0001842	0.0000000	0.00021	0.00021	0.0566	0.011	0.000	0.000
Jb : grad test:	0.0000320	0.0000001	-0.00318	-0.00318	0.8854	0.011	0.001	0.000
Jb : grad test:	0.0001183	0.0000000	0.00016	0.00016	0.0570	0.011	0.000	0.000

Table 3.9 Third part of the output of *CostFuncTest* showing the results for the gradients of J_b. Only the first and last eight lines are shown.

Jo : grad test:	rel err	df-g*dx	g*dx	df	f0	normx	normg	normh
Jo : grad test:	0.0568292	0.3304305	5.48402	5.81445	3235.4436	0.044	0.138	0.000
Jo : grad test:	0.1051882	0.1243716	-1.30674	-1.18237	2070.7771	0.044	0.077	0.000
Jo : grad test:	0.7153471	0.0252362	-0.06051	-0.03528	1526.3838	0.044	0.036	0.000
Jo : grad test:	0.3084716	0.0217271	-0.09216	-0.07043	1286.4680	0.044	0.023	0.000
Jo : grad test:	0.0751531	0.0559887	0.68901	0.74500	1309.5875	0.044	0.052	0.000
Jo : grad test:	0.2873660	0.0974491	0.24166	0.33911	1255.8923	0.044	0.046	0.000
Jo : grad test:	1.2135067	0.1346530	-0.02369	0.11096	1273.0304	0.043	0.052	0.000
Jo : grad test:	0.1174089	0.0607826	-0.57848	-0.51770	1228.9159	0.043	0.046	0.000
… … … …	… …	… … …	… … …	… … …	… …	… …	… …	…
Jo : grad test:	0.0318511	0.0097124	0.29522	0.30493	1078.0762	0.042	0.048	0.000
Jo : grad test:	0.0645247	0.0085854	0.12447	0.13306	1064.3875	0.042	0.046	0.000
Jo : grad test:	0.0170491	0.0183041	-1.09191	-1.07361	1077.9684	0.042	0.048	0.000
Jo : grad test:	0.0998519	0.0129081	0.11636	0.12927	1064.3137	0.042	0.046	0.000
Jo : grad test:	0.0595253	0.0250832	-0.44647	-0.42139	1077.8608	0.042	0.048	0.000
Jo : grad test:	0.2872958	0.0109069	-0.04887	-0.03796	1064.2410	0.042	0.046	0.000
Jo : grad test:	0.0239324	0.0290507	1.18482	1.21387	1077.7539	0.042	0.048	0.000
Jo : grad test:	0.0390451	0.0227541	0.56001	0.58276	1064.1672	0.041	0.046	0.000

Table 3.10 Fourth part of the output of *CostFuncTest* showing the results for the gradients of J_o. Only the first and last eight lines are shown.

Jt : grad test:	rel err	df-g*dx	g*dx	df	f0	normx	normg	normh
Jt : grad test:	0.0191559	0.8628197	44.17917	45.04199	10980.3789	0.044	0.134	0.000
Jt : grad test:	0.0153907	0.5655785	36.18247	36.74805	9727.8477	0.043	0.104	0.000
Jt : grad test:	0.0140463	0.4002228	28.09294	28.49316	8748.3984	0.041	0.089	0.000
Jt : grad test:	0.0080941	0.1883888	23.08651	23.27490	7921.4971	0.039	0.080	0.000
Jt : grad test:	0.0105575	0.2022476	18.95449	19.15674	7195.5962	0.037	0.075	0.000
Jt : grad test:	0.0136143	0.2179747	15.79277	16.01074	6548.9170	0.035	0.070	0.000
Jt : grad test:	0.0110471	0.1678696	15.02793	15.19580	5969.6201	0.034	0.066	0.000
Jt : grad test:	0.0100713	0.1320581	12.98025	13.11230	5449.4028	0.032	0.063	0.000
Jt : grad test:	0.0104593	0.1443472	13.65643	13.80078	4981.6147	0.031	0.059	0.000
… … … …	… …	… … …	… … …	… … …	… …	… …	… …	…
Jt : grad test:	0.3385797	-0.0022112	0.00874	0.00653	951.0865	0.011	0.007	0.000
Jt : grad test:	0.0071996	0.0003019	0.04163	0.04193	962.8463	0.011	0.016	0.000
Jt : grad test:	0.7282393	-0.0035559	-0.00133	-0.00488	951.0956	0.011	0.007	0.000
Jt : grad test:	0.0019097	0.0001300	0.06792	0.06805	962.8263	0.011	0.016	0.000
Jt : grad test:	0.0359622	0.0006102	-0.01758	-0.01697	951.1051	0.011	0.007	0.000
Jt : grad test:	0.0118557	-0.0012048	0.10283	0.10162	962.8040	0.011	0.016	0.000
Jt : grad test:	0.1422794	0.0030568	0.01843	0.02148	951.1137	0.011	0.007	0.000
Jt : grad test:	0.0416156	-0.0020523	0.05137	0.04932	962.7843	0.011	0.016	0.000

Table 3.11 Fifth (and last) part of the output of *CostFuncTest* showing the results for the gradients of J_t. Only the first and last eight lines are shown.

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 3.12.

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 3.13.

```
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: 07-10-2005
time: 17:30:23.52
```

Table 3.12 Output of program *TestDateTimeMod*.

```
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 3.13 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 3.14.

```
 Starting fileunit test program
 ===== lun_manager ======
 fileunit:           21  was not in use !!!
 free_lun returns without freeing any fileunit
 fileunit:           88  was not in the range that is
handled
 by this module ! (           20  -           29 )
 free_lun returns without freeing any fileunit
 fileunit:           88  was not in the range that is
handled
 by this module ! (           20  -           29 )
 enable_lun returns without enabling any fileunit
 fileunit:           88  was not in the range that is
handled
 by this module ! (           20  -           29 )
 disable_lun returns without disabling any fileunit
 unit:           22  is used?:   F
 unit:           22  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            22  is open
  lun            22  has a name: TestLunManager.F90
 end of inspect_luns
 fileunit:           22  is still in use !
 disabling it is only possible if it is not used !
 disable_lun returns without disabling any fileunit
 fileunit:           20  is in use
 fileunit:           21  is disabled
 fileunit:           22  is in use
 fileunit:           23  is still available
 fileunit:           24  is still available
 fileunit:           25  is still available
 fileunit:           26  is still available
 fileunit:           27  is still available
 fileunit:           28  is still available
 fileunit:           29  is still available
 fileunit:           22  was already enabled !!!
 enable_lun returns without enabling any fileunit
```

Table 3.14 Output of program *TestLunManager*.

3.9 Module *MultiFFT* (genscat support)

Module *MultiFFT.F90* in directory `genscat/support/multifft` contains a number of general routines for forward and inverse Fast Fourier Transformation (FFT). Program *MultiFFTtest* contains three tests for the multi-dimensional (e.g. 2D) Fourier method. The first part of the test is a forward/backward transformation test. Table 3.15 gives typical values of the (Euclidian) norm and of the real and imaginary parts of the difference field between an original input field and the field created by the subsequent application of the forward and backward method.

```
 TestInverse - transform forward and backward n1,n2:     8    4
 TestInverse - compare result with initial values
 NormCompare2D: Simple  re:    4.8475123E-14   im:    4.4438845E-14
 TestInverse - compare result with initial values
 NormCompare2D: FFT  re:    1.3496149E-14   im:    9.3866980E-15
 TestCompare - transform forward, make equal and transform backward n1,n2:     8    4
 TestCompare - compare result of Simple and FFT after forward transform
 NormCompare2D: fw: Simple-FFT  re:    8.7014007E-12   im:    4.6145102E-12
 TestCompare - compare result of Simple and FFT after backward transform
 NormCompare2D: bw: Simple-FFT  re:    2.3924615E-13   im:    1.6883336E-13

 TestInverse - transform forward and backward n1,n2:    16    8
 TestInverse - compare result with initial values
 NormCompare2D: Simple  re:    1.5297629E-13   im:    1.5070734E-13
 TestInverse - compare result with initial values
 NormCompare2D: FFT  re:    1.4581330E-14   im:    1.0193938E-14
 TestCompare - transform forward, make equal and transform backward n1,n2:    16    8
 TestCompare - compare result of Simple and FFT after forward transform
 NormCompare2D: fw: Simple-FFT  re:    1.3613137E-10   im:    1.4311996E-10
 TestCompare - compare result of Simple and FFT after backward transform
 NormCompare2D: bw: Simple-FFT  re:    1.3359370E-12   im:    8.9315225E-13

 TestInverse - transform forward and backward n1,n2:    32   16
 TestInverse - compare result with initial values
 NormCompare2D: Simple  re:    6.0022641E-13   im:    4.5660687E-13
 TestInverse - compare result with initial values
 NormCompare2D: FFT  re:    1.8041122E-14   im:    1.8373525E-14
 TestCompare - transform forward, make equal and transform backward n1,n2:    32   16
 TestCompare - compare result of Simple and FFT after forward transform
 NormCompare2D: fw: Simple-FFT  re:    2.8387792E-09   im:    2.4731994E-09
 TestCompare - compare result of Simple and FFT after backward transform
 NormCompare2D: bw: Simple-FFT  re:    5.2960583E-12   im:    4.7728917E-122

 TestInverse - transform forward and backward n1,n2:    64   32
 TestInverse - compare result with initial values
 NormCompare2D: Simple  re:    2.1652521E-12   im:    2.1109030E-12
 TestInverse - compare result with initial values
 NormCompare2D: FFT  re:    2.1942209E-14   im:    2.1276510E-14
 TestCompare - transform forward, make equal and transform backward n1,n2:    64   32
 TestCompare - compare result of Simple and FFT after forward transform
 NormCompare2D: fw: Simple-FFT  re:    4.6588273E-08   im:    3.9916234E-08
 TestCompare - compare result of Simple and FFT after backward transform
 NormCompare2D: bw: Simple-FFT  re:    2.2838266E-11   im:    1.8644074E-11

FFTforward adjoint : grad test:    rel err    df-g*dx      g*dx        df         f0  normx  normg  normh
FFTforward adjoint : grad test:  0.0000152 -0.0000470    3.09255    3.09250   109.2720  0.082  0.163  0.001
FFTforward adjoint : grad test:  0.0000166 -0.0000455    2.73545    2.73540    98.6180  0.078  0.155  0.001
FFTforward adjoint : grad test:  0.0000089  0.0000191    2.13916    2.13918    89.0027  0.074  0.147  0.001
FFTforward adjoint : grad test:  0.0000047  0.0000141    2.99072    2.99074    80.3249  0.070  0.140  0.001
FFTforward adjoint : grad test:  0.0000036  0.0000088    2.47929    2.47929    72.4932  0.067  0.133  0.001

FFTbackward adjoint : grad test:    rel err    df-g*dx      g*dx        df         f0  normx  normg  normh
FFTbackward adjoint : grad test:  0.0000020 -0.0000072    3.66198    3.66197   116.9874  0.085  0.169  0.002
FFTbackward adjoint : grad test:  0.0000034 -0.0000110    3.18476    3.18475   105.5811  0.080  0.161  0.001
FFTbackward adjoint : grad test:  0.0000053 -0.0000186    3.53794    3.53793    95.2869  0.076  0.153  0.001
FFTbackward adjoint : grad test:  0.0000080  0.0000207    2.59269    2.59271    85.9965  0.072  0.145  0.001
FFTbackward adjoint : grad test:  0.0000038 -0.0000086    2.23831    2.23830    77.6118  0.069  0.138  0.001
```

Table 3.15 Output of program *MultiFFTtest.*

The second part of the test consists of a comparison between the two methods (simple and FFT). Table 3.15 gives typical values for the norms of the difference fields. Routine *GradTest*, see section 2.1, is used to test the adjoint.

```
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          1.7976931348623167E+308
 r8: minexponent        -1021
 r8: maxexponent         1024
 r8: precision             15
 r8: radix                  2
 r8: range                307
 r8: tiny          2.2250738585072010E-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 3.16 Output of program *test_mumerics*.

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 3.16.

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 3.17.

```
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 3.17 Output of program *SortModTest*

3.12 Module *SwsData*

Module *SwsData.F90* in directory `sdp/sws` contains the data structure definitions for the interface between the SeaWinds data and the generic genscat routines. It is tested by program *SwsDataTest*, the output of which is listed in table 3.18.

```
 Test Dummy Cell============= CELL ==========
 Test Dummy CellNodeNr         :           75
 Test Dummy CellRowNr          :          200
 Test Dummy CellRevNr          :         1000
 Test Dummy Cell--
 Test Dummy CellAcross_Track_Res:     25.00000
 Test Dummy CellAlong_Track_Res :     25.00000
 Test Dummy CellLat            :     61.00000
 Test Dummy CellLon            :     71.00000
 Test Dummy Cell--
 Test Dummy Cell Time: year :       2000   month:        100   day:              1
 Test Dummy Cell Time: hour:           1   min:            1   sec:              1
 Test Dummy CellTime_to_Edge    :     300.0000
 Test Dummy Cell--
 Test Dummy Cell Model: Speed:     1.000000
 Test Dummy Cell Model: Dir  :     10.00000
 Test Dummy Cell EC: Speed:     1.000000
 Test Dummy Cell EC: Dir  :     10.00000
 Test Dummy Cell TWODV: Speed:     1.000000
 Test Dummy Cell TWODV: Dir  :     10.00000
 Test Dummy Cell JPL: Speed:     1.000000
 Test Dummy Cell JPL: Dir  :     10.00000
 Test Dummy Cell--
 Test Dummy CellSigma0_In_Cell  :            3
 Test Dummy Cell Beam: 1:  NUM:            1
 Test Dummy Cell Beam: 1:  Sigma0: Lat         :    12.00000
 Test Dummy Cell Beam: 1:  Sigma0: Lon         :    13.00000
```

```
 Test Dummy Cell Beam: 1:  Sigma0: Atten_Value :    1.000000
 Test Dummy Cell Beam: 1:  Sigma0: Azimuth     :    10.00000
 Test Dummy Cell Beam: 1:  Sigma0: Incidence   :    30.00000
 Test Dummy Cell Beam: 1:  Sigma0: Value       :   -1.000000
 Test Dummy Cell Beam: 1:  Sigma0: Variance_QC :    1.000000
 Test Dummy Cell Beam: 1:  Sigma0:Qual_Flag  int  :            0
 Test Dummy Cell Beam: 1:  Sigma0:Mode_Flag  int  :            0
 Test Dummy Cell Beam: 1:  Sigma0:Surf_Flag  int  :            0
 Test Dummy Cell Beam: 1:  Kp: Alpha:      1.000000
 Test Dummy Cell Beam: 1:  Kp: Beta:       2.000000
 Test Dummy Cell Beam: 1:  Kp: Gamma:      3.000000
 Test Dummy Cell Beam: 1:  K_Polar:     1.000000
 Test Dummy Cell Beam: 2:  NUM:            1
 Test Dummy Cell Beam: 2:  Sigma0: Lat         :    12.00000
 Test Dummy Cell Beam: 2:  Sigma0: Lon         :    13.00000
 Test Dummy Cell Beam: 2:  Sigma0: Atten_Value :    1.000000
 Test Dummy Cell Beam: 2:  Sigma0: Azimuth     :    10.00000
 Test Dummy Cell Beam: 2:  Sigma0: Incidence   :    30.00000
 Test Dummy Cell Beam: 2:  Sigma0: Value       :   -1.000000
 Test Dummy Cell Beam: 2:  Sigma0: Variance_QC :    1.000000
 Test Dummy Cell Beam: 2:  Sigma0:Qual_Flag  int  :            0
 Test Dummy Cell Beam: 2:  Sigma0:Mode_Flag  int  :            0
 Test Dummy Cell Beam: 2:  Sigma0:Surf_Flag  int  :            0
 Test Dummy Cell Beam: 2:  Kp: Alpha:      1.000000
 Test Dummy Cell Beam: 2:  Kp: Beta:       2.000000
 Test Dummy Cell Beam: 2:  Kp: Gamma:      3.000000
 Test Dummy Cell Beam: 2:  K_Polar:     1.000000
 Test Dummy Cell Beam: 3:  NUM:            1
 Test Dummy Cell Beam: 3:  Sigma0: Lat         :    12.00000
 Test Dummy Cell Beam: 3:  Sigma0: Lon         :    13.00000
 Test Dummy Cell Beam: 3:  Sigma0: Atten_Value :    1.000000
 Test Dummy Cell Beam: 3:  Sigma0: Azimuth     :    10.00000
 Test Dummy Cell Beam: 3:  Sigma0: Incidence   :    30.00000
 Test Dummy Cell Beam: 3:  Sigma0: Value       :   -1.000000
 Test Dummy Cell Beam: 3:  Sigma0: Variance_QC :    1.000000
 Test Dummy Cell Beam: 3:  Sigma0:Qual_Flag  int  :            0
 Test Dummy Cell Beam: 3:  Sigma0:Mode_Flag  int  :            0
 Test Dummy Cell Beam: 3:  Sigma0:Surf_Flag  int  :            0
 Test Dummy Cell Beam: 3:  Kp: Alpha:      1.000000
 Test Dummy Cell Beam: 3:  Kp: Beta:       2.000000
 Test Dummy Cell Beam: 3:  Kp: Gamma:      3.000000
 Test Dummy Cell Beam: 3:  K_Polar:     1.000000
 Test Dummy Cell Beam: 4:  NUM:            1
 Test Dummy Cell Beam: 4:  Sigma0: Lat         :    12.00000
 Test Dummy Cell Beam: 4:  Sigma0: Lon         :    13.00000
 Test Dummy Cell Beam: 4:  Sigma0: Atten_Value :    1.000000
 Test Dummy Cell Beam: 4:  Sigma0: Azimuth     :    10.00000
 Test Dummy Cell Beam: 4:  Sigma0: Incidence   :    30.00000
 Test Dummy Cell Beam: 4:  Sigma0: Value       :   -1.000000
 Test Dummy Cell Beam: 4:  Sigma0: Variance_QC :    1.000000
 Test Dummy Cell Beam: 4:  Sigma0:Qual_Flag  int  :            0
 Test Dummy Cell Beam: 4:  Sigma0:Mode_Flag  int  :            0
 Test Dummy Cell Beam: 4:  Sigma0:Surf_Flag  int  :            0
 Test Dummy Cell Beam: 4:  Kp: Alpha:      1.000000
 Test Dummy Cell Beam: 4:  Kp: Beta:       2.000000
 Test Dummy Cell Beam: 4:  Kp: Gamma:      3.000000
 Test Dummy Cell Beam: 4:  K_Polar:     1.000000
 Test Dummy Cell--
 Test Dummy CellSelection      :            1
 Test Dummy CellNum_Ambigs     :            4
 Test Dummy Cell Ambi: 001: wind Speed:     1.000000
 Test Dummy Cell Ambi: 001: wind Dir  :     10.00000
 Test Dummy Cell Ambi: 001: error Speed:     1.000000
 Test Dummy Cell Ambi: 001: error Dir  :     10.00000
 Test Dummy Cell Ambi: 001: prob:            0.8000000
 Test Dummy Cell Ambi: 001: conedistance:    0.5000000
 Test Dummy Cell Ambi: 002: wind Speed:     1.000000
 Test Dummy Cell Ambi: 002: wind Dir  :     10.00000
 Test Dummy Cell Ambi: 002: error Speed:     1.000000
```

```
Test Dummy Cell Ambi: 002: error Dir  :     10.00000
Test Dummy Cell Ambi: 002: prob:            0.8000000
Test Dummy Cell Ambi: 002: conedistance:    0.5000000
Test Dummy Cell Ambi: 003: wind Speed:     1.000000
Test Dummy Cell Ambi: 003: wind Dir  :     10.00000
Test Dummy Cell Ambi: 003: error Speed:     1.000000
Test Dummy Cell Ambi: 003: error Dir  :     10.00000
Test Dummy Cell Ambi: 003: prob:            0.8000000
Test Dummy Cell Ambi: 003: conedistance:    0.5000000
Test Dummy Cell Ambi: 004: wind Speed:     1.000000
Test Dummy Cell Ambi: 004: wind Dir  :     10.00000
Test Dummy Cell Ambi: 004: error Speed:     1.000000
Test Dummy Cell Ambi: 004: error Dir  :     10.00000
Test Dummy Cell Ambi: 004: prob:            0.8000000
Test Dummy Cell Ambi: 004: conedistance:    0.5000000
Test Dummy Cell Rain: MP :                1
Test Dummy Cell Rain: NOF:                1
Test Dummy Cell Rain: Rate:      0.0000000
Test Dummy Cell Rain: Atten.:    0.0000000
Test Dummy Cell Cell QualFlag: POOR QualSigma0
Test Dummy Cell Cell QualFlag:      MonFlag
Test Dummy Cell Cell QualFlag: POOR large
Test Dummy CellQuality_Flag  int  :        36896
Test Dummy Cell Antenna: 1: Num          :              1
Test Dummy Cell Antenna: 1: Polarization :     0.0000000
Test Dummy Cell Antenna: 1: Tb_Mean      :      10.00000
Test Dummy Cell Antenna: 1: Tb_StdDev    :      1.000000
Test Dummy Cell Antenna: 2: Num          :              2
Test Dummy Cell Antenna: 2: Polarization :     0.0000000
Test Dummy Cell Antenna: 2: Tb_Mean      :      10.00000
Test Dummy Cell Antenna: 2: Tb_StdDev    :      1.000000
Test Dummy Cell -------
Test Dummy CellSatellite_ID    :            1
Test Dummy CellSat_Motion      :     10.00000

Test Dummy CellInstrument_ID   :            1
Test Dummy CellGMF_ID          :            1
Test Dummy CellSoftware_ID     :            1
Test Dummy Cell -------
Test Dummy Cell Process Flag: OK RevNr
Test Dummy Cell Process Flag: OK RowNr
Test Dummy Cell Process Flag: OK NodeNr
Test Dummy Cell Process Flag: OK Lat
Test Dummy Cell Process Flag: OK Lon
Test Dummy Cell Process Flag: OK mleqc
Test Dummy Cell Process Flag: OK Along_track_res
Test Dummy Cell Process Flag: OK Across_track_res
Test Dummy Cell Process Flag: OK Time2Edge
Test Dummy Cell Process Flag: OK year
Test Dummy Cell Process Flag: OK Day
Test Dummy Cell Process Flag: OK Hour
Test Dummy Cell Process Flag: OK minute
Test Dummy Cell Process Flag: OK second
Test Dummy Cell Process Flag: OK ModelWind
Test Dummy Cell Process Flag: OK rain
Test Dummy Cell Process Flag: OK Tb
Test Dummy Cell Process Flag: OK Beam(4) OuterAft
Test Dummy Cell Process Flag: OK Beam(3) InnerAft
Test Dummy Cell Process Flag: OK Beam(2) OuterFore
Test Dummy Cell Process Flag: OK Beam(1) InnerFore
Test Dummy Cell Process Flag: OK Sigma0_In_Cell
Test Dummy Cell Process Flag: OK Ambiguity
Test Dummy Cell Process Flag: OK Selection
Test Dummy Cell Process Flag: POOR month
Test Dummy Cell -------
```

Table 3.18 Output of program *SwsDataTest*.

Chapter 4

Integration tests

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.

Directory `sdp/tests` contains file `QS_D02001_S0006_E0120_B1320303`, a NOAA BUFR file which is an input file for SDP. Directory `sdp/execs` contains a number of shell scripts of which `sdp_025` and `sdp_qdp` are used to produce the test output BUFR files named `SDP_testrun_1`, `SDP_Testrun_2`, and `SDP_Testrun_3` located in directory `sdp/tests`. Table 4.1 summarizes the SDP test runs.

Command	**Result identical with**
`sdp_025 -f ../tests/QS_D02001_S0006_E0120_B1320303`	`SDP_Testrun_1`
`sdp_025 -f ../tests/QS_D02001_S0006_E0120_B1320303 –mss`	`SDP_Testrun_2`
`sdp_qdp -f ../tests/QS_D02001_S0006_E0120_B1320303`	`SDP_Testrun_3`

Table 4.1 SDP testruns.

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 testrun. SeaWinds covered part of the Indian Ocean southeast of India, part of the Barentz Sea north of Scandinavia, 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 legendum. Figure 2.3 shows the results of testrun number 2, but the two other testruns will yield very similar results for the magnitude of the wind speed.

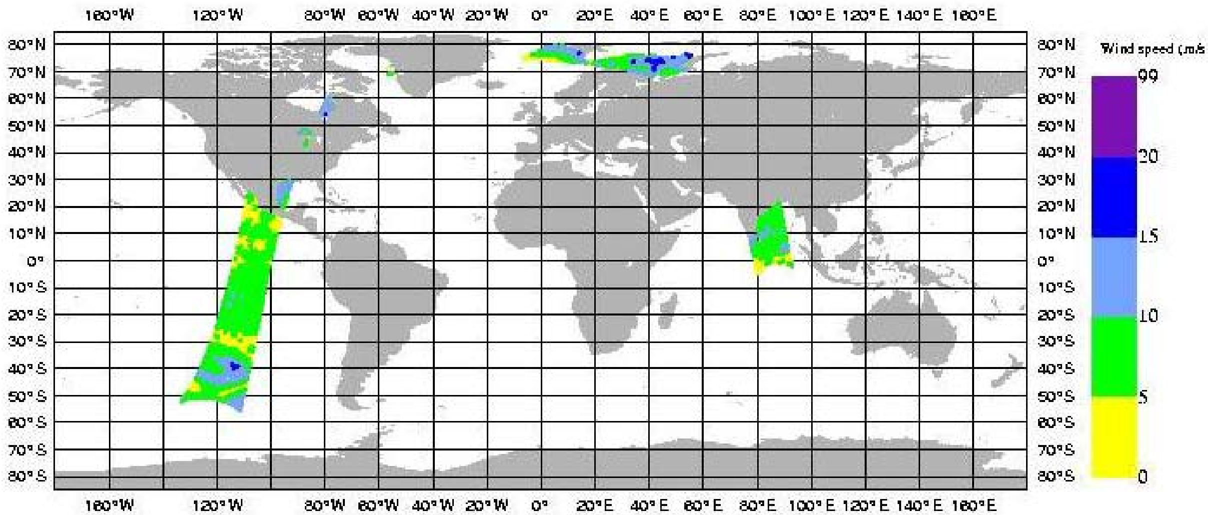

Figure 4.1 Global coverage of the testruns. Wind speed results for testrun 2 are shown.

Figures 4.2 and 4.3 show detailed wind fields at 25 km resolution over the Pacific, west of South Amecica (lower left part in figure 4.1). Figure 4.2 is obtained with the testrun 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 testrun 2 data, i.e., with the Multiple Solution Scheme switched on. Now the wind field is much smoother.

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_Testrun` file. Such software is not part of the SDP package, but is available at KNMI and may be obtained upon request.

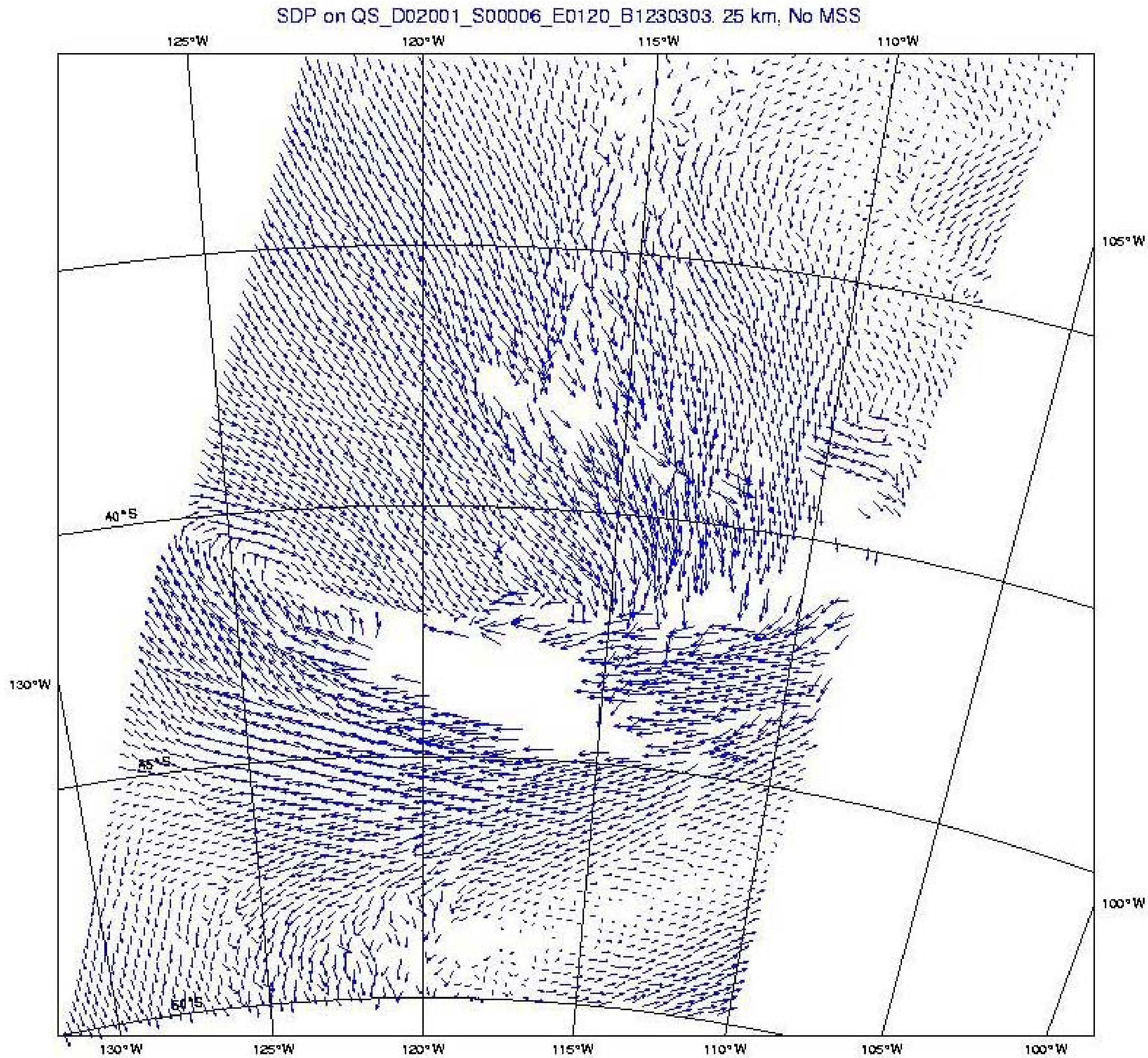

Figure 4.2 Part of testrun 1 (Pacific west of South America) obtained with SDP standard processing at 25 km resolution.

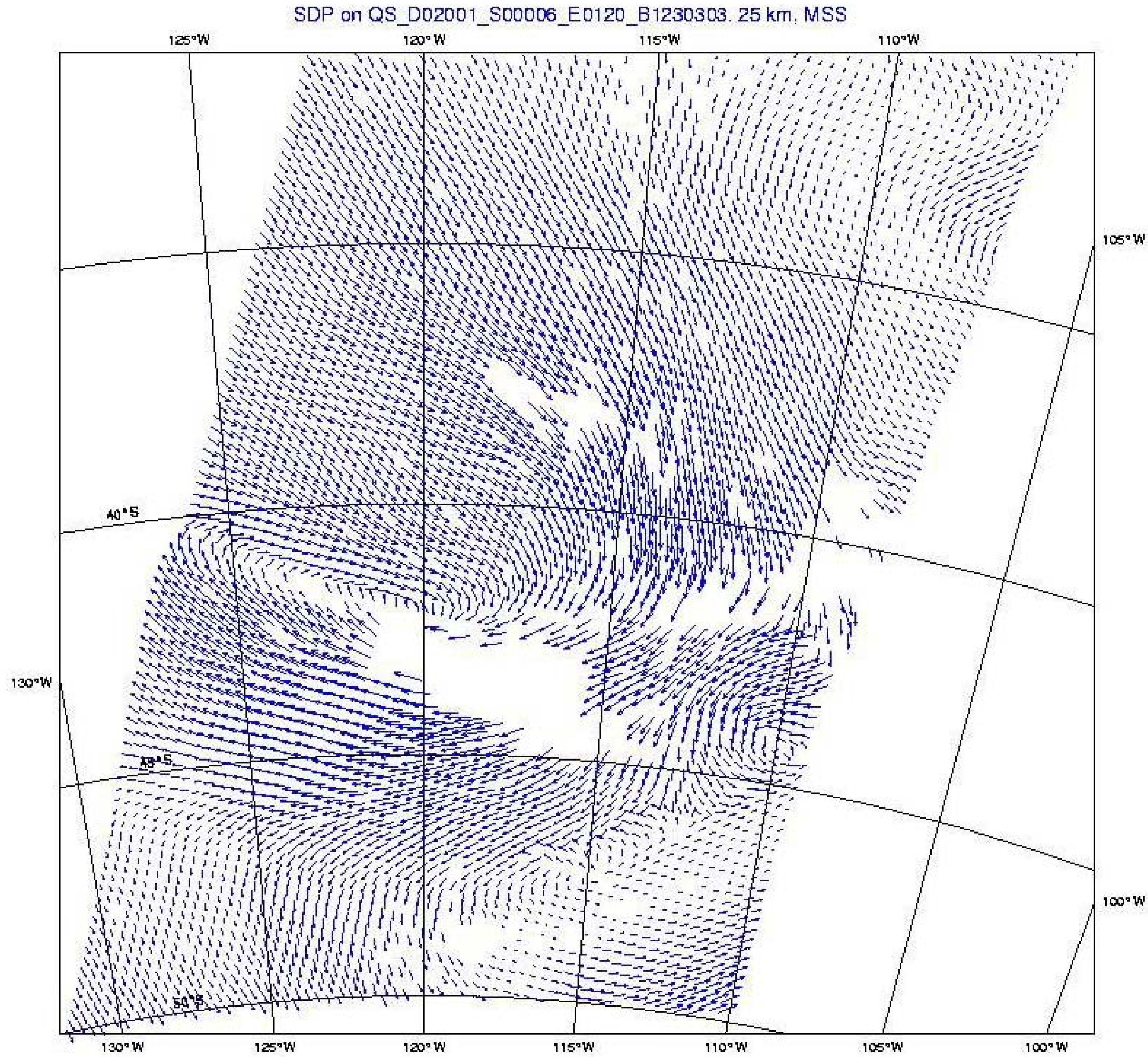

Figure 4.3 Part of testrun 2 (same area as figure 4.2) obtained with SDP at 25 km resolution with the Multiple Solution Scheme switched on.

.

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/python` 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 1.2 release.

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).

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 ERS and ASCAT scatterometer processor.

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 was tested successfully.

Platform	Operating system	Fortran compiler
HP workstation	SuSe LINUX	GNU g95, Portland f90
SUN workstation	SUN OS UNIX	Sun Fortran
Compaq workstation	Tru64 UNIX	Compaq Fortran
SGI workstation	IRIX64 UNIX	MIPSpro
PC	Windows XP with Cygwin	GNU g95

Table 6.1 Supported platforms and compilers for SDP.

Note that SDP can be run under Windows when a LINUX emulator like Cygwin or MinGW is installed.

Chapter 7

User documentation tests

The SDP package version 1.0 and its user documentation have been provided to beta testers for review. The beta tester's comments have been implemented in versions 1.1 and 1.2 of the user documentation.

The user documentation, especially the paragraph on implementation and testing, has been tested for implementation of SDP on a Windows PC with the Cygwin UNIX emulator and the GNU g95 Fortran compiler. 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.