Technical Report No. 14

The WASAR algorithm for retrieving ocean wave spectra from SAR image spectra

by
Susanne Hasselmann • Patrick Heimbach • Christian Bennefeld
Max-Planck-Institut für Meteorologie, Hamburg, Germany

Edited by:

Modell Betreuungsgruppe DKRZ
Hamburg, April 1998

ISSN 0940-9327

CONTENTS

1. THEORETICAL BACKGROUND 1

1.1 The mapping of a wave spectrum into a SAR image spectrum
(Program sarinv, subroutine fullmap) 1

1.1.1 Calibration of ERS-1 SAR spectra
(Routine rdsarspec,calibra) 1

1.1.1 Computation of the cut off wave length
(Routine calccut) 2

1.2 Inversion of the mapping relation
(Program sarinv, routine invert) 2

1.3 Matching the wave number grids of the observed
and computed SAR spectra
(Program sarinv, subroutines invert, uwafilter) 5

1.4 Extended retrieval algorithm
(Programs sarinv, partinv, findbest) 6

1.5 The wave partitioning scheme
(Program partinv, subroutine swellsep) 7

1.6 Correction of the input spectrum and the cross-assignment of spectral wave systems
(Program partinv, subroutines tustre, crossas) 8

1.7 The optimal retrieved wave spectrum
(Program findbest) 10

2. THE SOFTWARE 11

2.1 Routines provided by the user 11

2.2 The SAR inversion program sarinv 11

2.2.1 The input files for sarinv 13

2.2.1 The output files of sarinv 18

2.3 Spectral partitioning and correction program partinv 19

2.3.1 Flowtrace calling tree 20

2.3.1 Input & output 21

2.4 Best estimate wave spectrum (findbest) 21

2.4.1 Flowtrace calling tree 21

2.4.1 Input files 22

2.5 The grids 23

Appendix A: The UNIX chainjob runsar.jcl. 25

Appendix B: Data formats 31

Appendix C:The CRAY FFT routine cfft2d. 37

Appendix D: The NAG library routine F04JGE - NAG for solving linear least square problems. 41

1. THEORETICAL BACKGROUND

1.1 The mapping of a wave spectrum into a SAR image spectrum (Program sarinv, subroutine fullmap)

The mapping $P(\boldsymbol{k}) = \Phi(F(\boldsymbol{k}))$ of an ocean wave spectrum $F(\boldsymbol{k})$ into a SAR image spectrum $P(\boldsymbol{k})$ is given by a closed integral transform, which may be written as a series expansion in the form (Hasselmann and Hasselmann, 1991, referred to below as (HH)).

$$P(\boldsymbol{k}) = exp(-k_x^2 \xi'^2) \sum_{n=1}^{\infty} \sum_{m=2n-2}^{2n} (k_x \beta)^m P_{nm}(\boldsymbol{k}) \tag{1.1.1}$$

where k_x denotes the wave number component of the long waves in the azimuthal direction, β is the ratio of the slant range to the platform velocity, and the spectral factors P_{nm} consist of Fourier transforms of higher order products of the auto- and cross-co-variance functions of the radial orbital velocity and the (real-aperture-radar) cross-section modulation function. The index n indicates the nonlinearity order with respect to the input wave spectrum and the index m the order with respect to the velocity bunching parameter β. The (nonlinear) exponential factor depends on the mean square azimuthal displacement ξ'^2 of a scattering element,

$$\xi'^2 = \beta^2 \langle u_r^2 \rangle = \beta^2 \int \left| T_k^v \right|^2 F(\boldsymbol{k}) d\boldsymbol{k} \tag{1.1.2}$$

where < .. > denotes the ensemble average and u_r the radial component of the orbital velocity of the ocean waves. The range-velocity transfer function T_k^v is given, according to classical surface wave theory, by

$$T_k^v = -\omega \left(sin\theta \frac{k_y}{|\boldsymbol{k}|} + i cos\theta \right) \tag{1.1.3}$$

where ω denotes the radian frequency of the long waves, θ the incidence angle and k_y the wave number component of the long waves in the range direction.

Equation (1.1.1) can be rapidly evaluated by Fast Fourier Transforms, enabling the forward-mapping relation to be inverted by iterative methods.

1.1.1 Calibration of ERS-1 SAR spectra (Routine rdsarspec,calibra)

The observed ERS-1 SAR Wave Mode spectra are calibrated with a factor ‘xcali’ , for which a clutter noise level ‘xnoise’ is computed of the original UWA - SAR - image spectra as the average over the five points with minimal energy on the 100m wave length circle

The noise is then subtracted from the observed SAR spectrum and the spectrum calibrated with a fac-

tor `xcali` (Details see in Alpers and Hasselmann, 1982):

$$xcali = xcalpar \cdot \rho_a \cdot \rho_r / ((2 \cdot \pi)^2 \cdot nlook \cdot xnoise)$$

where $\rho_a, \rho_r = 33m$ denote the azimuthal and range SAR resolution. The parameter xcalpar (0.78 for ERS-1) is a correction factor due to 3 look amplitude rather than energy averaging (for details see Brüning, et al, 1994)

1.1.1 Computation of the cut off wave length (Routine `calccut`)

The tail beyond the azimuthal cut off wave number is very steep i.e. the spectrum falls off into the noise level within a very small wave number range. Therefore, the cut-off wave number can be defined as the wave number beyond which the spectral energy falls below the noise level.

The cut - off wave length of a SAR image spectrum is computed as:

Compute the average over k_y from 3 grid points above to 3 grid points below the largest peak of the spectrum. Then find the index n of the wave number k_x (move from highest wave number to lower wave numbers) where this average is above a level: $xcalpar \cdot \rho_a \cdot \rho_r / ((2 \cdot \pi)^2 \cdot nlook)$), interpolate to cut-off wave number between index n and n+1 or n-1 and compute the equivalent wave length.

1.2 Inversion of the mapping relation (Program `sarinv`, routine `invert`)

To invert the forward mapping relation Equation (1.1.1) we apply the standard technique of minimization of a suitably defined cost-function, (HH, Hasselmann, S. *et al.*, 1996, referred to below as HBHH). The cost function contains three error terms: the deviation between the simulated and the observed SAR spectrum; a second ‘regulaziation’ term penalizing the deviation between the first-guess and retrieved wave spectrum, which is needed to resolve the 180° angular ambiguity of the observed frozen-image SAR spectrum and to provide the missing information at high wave numbers beyond the azimuthal cut-off of the SAR; and a third term penalizing the deviation between the observed and simulated high wave number cut-off. For this purpose a clutter cut-off length scale is defined as follows: First a mean one-dimensional azimuthal SAR spectrum is computed by averaging over the range bins. To reduce the noise contribution, the range-bin averaging is restricted to the seven range bins bracketing the range bin of the spectral peak. The azimuthal cut-off length scale λ_{cl} is then defined as the wave length at which the mean azimuthal SAR spectrum has decreased to a value 3 dB above the noise floor.

To correct the cut-off, a free energy-scaling parameter α is introduced. By applying the energy-scaling factor to the entire spectrum, rather than only to the low-wave number part of the spectrum for which SAR information is directly available, the factor α modifies also the high-wave number components beyond the azimuthal cut-off. These contribute significantly to the rms orbital velocity, which directly affects the rms azimuthal displacement ξ' Equation (1.1.2) and thereby the cut-off length scale λ_{cl}. Assuming that ξ' and λ_{cl} are proportional, application of the energy scaling factor α leads to a modification $\lambda_{cl} \rightarrow \sqrt{(\alpha)}\lambda_{cl}$ of the simulated cut-off length scale. The scaling factor α is automatically adjusted to reduce the error between the observed and simulated cut-off.

Thus the cost function takes the form:

$$J = \int [P(\boldsymbol{k}) - \hat{P}(\boldsymbol{k})]^2 P(\boldsymbol{k}) d\boldsymbol{k} + \mu \int \frac{[F(\boldsymbol{k}) - \hat{F}(\boldsymbol{k})]^2}{[B + min\{F(\boldsymbol{k}), \hat{F}(\boldsymbol{k})\}]^2} d\boldsymbol{k} + \eta \frac{[\alpha\lambda_{cl}^2 - \hat{\lambda}_{cl}^2]^2}{max\left\{\lambda_{cl}^4, \hat{\lambda}_{cl}^4\right\}} \qquad (1.2.1)$$

where $\hat{F}(\boldsymbol{k})$ denotes the first-guess wave spectrum, $\hat{P}(\boldsymbol{k})$ the observed SAR spectrum, $P(\boldsymbol{k})$ the SAR spectrum computed from the best-fit wave spectrum $F(\boldsymbol{k})$ and $\hat{\lambda}_{cl}$, λ_{cl} denote the clutter cut-off length scales of the observed and simulated SAR image spectrum, respectively. The weighting factor μ is chosen to be sufficiently small $(\mu = 10^{-3}\hat{P}_{max}^3)$ that the form of the first-guess wave spectrum $\hat{F}(\boldsymbol{k})$ has only a small impact on the final solution where SAR information is available, but is still large enough to resolve the 180° directional ambiguity and determine the form of the high wave number part of the spectrum beyond the SAR azimuthal wave number cut-off. The weight $\eta = 0.5 \cdot 10^5 [\int \hat{P}(\boldsymbol{k}) d\boldsymbol{k}]^3$ is chosen such that the first and third term of Equation (1.2.1) are of the same order of magnitude. A small constant $B = 10^{-4}\hat{F}_{max}$ is introduced to prevent the denominator of the second term from vanishing.

The iteration scheme used by HHBHH to minimize the cost-function proceeds as follows: Start from a first-guess wave spectrum $F^1(\boldsymbol{k}) = \hat{F}(\boldsymbol{k})$ and associated SAR spectrum $P^1(\boldsymbol{k})$, computed from $F^1(\boldsymbol{k})$ using the fully nonlinear mapping relation and $\alpha^1 = 1$. Assume that at the beginning of the *n*'th iteration step a wave spectrum $F^n(\boldsymbol{k}) = F^n$ and its associated SAR spectrum $P^n(\boldsymbol{k}) = \Phi(F^n) = P^n$ have been determined. An improved estimate of the wave spectrum

$$F^{n+1} = \alpha^n F^n + \Delta F^n \qquad (1.2.2)$$

with associated estimate

$$P^{n+1} = \Phi(F^{n+1}) \qquad (1.2.3)$$

of the SAR spectrum is then constructed in two stages.

First, it is assumed that an approximation δP^n of the SAR spectral increment $\Delta P^n = P^{n+1} - P^n$ can

be computed from ΔF^n using the quasilinear SAR-wave spectral mapping relation

$$P(\boldsymbol{k}) \sim exp(-k_x^2 \xi'^2) P_1(\boldsymbol{k}) \tag{1.2.4}$$

which is given by the truncation of Equation (1.1.1) at the nonlinearity index $n = 1$,

$$\begin{aligned} P_1(\boldsymbol{k}) &= \sum_{m=0}^{2} (k_x \beta)^m P_{1m}(\boldsymbol{k}) \\ &= \left|T_{SAR}(\boldsymbol{k})\right|^2 \cdot F(\boldsymbol{k}) + \left|T_{SAR}(-\boldsymbol{k})\right|^2 \cdot F(-k) \end{aligned} \tag{1.2.5}$$

where $T_{SAR}(\boldsymbol{k})$ represents the transfer function of the standard linear SAR image mapping relation. Thus we set

$$\delta P^n = W_{\boldsymbol{k}} \Delta F^n(\boldsymbol{k}) + W_{-\boldsymbol{k}} \Delta F^n(-\boldsymbol{k}) \tag{1.2.6}$$

where

$$W_{\boldsymbol{k}} = \left|T_{SAR}(\boldsymbol{k})\right|^2 exp(-k_x^2 \xi'^2) \tag{1.2.7}$$

Since the relation between δP^n and ΔF^n is linear (the factor $exp(-k_x^2 \xi'^2)$ in Equation (1.2.4) is regarded as known from the previous iteration step). Substitution of these changes into the cost function yields a solvable quadratic minimization problem for ΔF^n.

Substituting Equation (1.2.2) into Equation (1.2.1), setting $P^{n+1} \cong P^n + \delta P^n$ and defining, for brevity, $\mu^n = \mu[B + min\{F^n, \hat{F}\}]^{-2}$ and $\eta^n = \eta(max((\lambda_{cl}^n)^4, \hat{\lambda}_{cl}^4))^{-1}$, the cost function, expressed in terms of ΔF^n and δP^n then becomes

$$J = \int \hat{P}[\delta P^n - (\hat{P} - P^n)]^2 d\boldsymbol{k} + \int \mu^n [\Delta F^n - (\hat{F} - \alpha^n F^n)]^2 d\boldsymbol{k} + \eta^n (\alpha^n (\lambda_{cl}^n)^2 - \hat{\lambda}_{cl}^2)^2 \tag{1.2.8}$$

The general linearized variational problem

$$\delta J / \delta \Delta F^n(\boldsymbol{k}) = 0 \tag{1.2.9}$$

and

$$\delta J / \delta \alpha = 0 \tag{1.2.10}$$

is solved first with respect to ΔF^n for fixed α^n, which is taken as the value determined from the last iteration step, $\alpha^n = \alpha^{n-1}$ with $\alpha^1 = 1$. This yields (cf. HHBHH)

$$\Delta F_{\boldsymbol{k}}^n = \frac{A_{-\boldsymbol{k}}[W_{\boldsymbol{k}} \Delta \hat{P}_{\boldsymbol{k}}^n + \mu^n \Delta \hat{F}_{\boldsymbol{k}}^n] - B_{\boldsymbol{k}}[W_{-\boldsymbol{k}} \Delta \hat{P}_{\boldsymbol{k}}^n + \mu^n \Delta \hat{F}_{-\boldsymbol{k}}^n]}{A_{\boldsymbol{k}} A_{-\boldsymbol{k}} - B_{\boldsymbol{k}}^2} \tag{1.2.11}$$

where

$$\Delta\hat{P}_{\boldsymbol{k}}^{n} = \hat{P}(\boldsymbol{k}) - P^{n}(\boldsymbol{k}) = \hat{P}(-\boldsymbol{k}) - P^{n}(-\boldsymbol{k}) \tag{1.2.12}$$

$$\Delta\hat{F}_{\boldsymbol{k}}^{n} = \hat{F}(\boldsymbol{k}) - \alpha^{n}F^{n}(\boldsymbol{k}) \tag{1.2.13}$$

$$A_{\boldsymbol{k}} = W_{k}^{2} + 2\mu^{n} \tag{1.2.14}$$

and

$$B_{\boldsymbol{k}} = W_{\boldsymbol{k}}W_{-\boldsymbol{k}} \tag{1.2.15}$$

Subsequently, the energy scaling parameter α^{n} is obtained by solving equation (1.2.10):

$$\alpha^{n} = \frac{\eta^{n}\hat{\lambda}_{cl}^{2}(\lambda_{cl}^{n})^{2} + \mu^{n}\int F_{\boldsymbol{k}}^{n}[\Delta F_{\boldsymbol{k}}^{n} - \hat{F}_{\boldsymbol{k}}]dk}{\eta^{n}(\lambda_{cl}^{n})^{4} + \mu^{n}\int (F_{\boldsymbol{k}}^{n})^{2}d\boldsymbol{k}} \tag{1.2.16}$$

The computation of ΔF and α is repeated iteratively until the difference between two iterations is smaller than a given ε.

Having determined $\Delta F_{\boldsymbol{k}}^{n}$ and α^{n} and thus F^{n+1}, the associated SAR spectrum P^{n+1} is again computed from F^{n+1} using the full nonlinear SAR transformation relation, and the iteration is repeated. To maintain numerical stability it was found necessary to restrict $\Delta F_{\boldsymbol{k}}^{n}$ to the range $\left|\Delta F_{\boldsymbol{k}}^{n}\right| \le min[\hat{F}_{\boldsymbol{k}}, \alpha F_{\boldsymbol{k}}^{n}]/4$ if

$$\frac{\mu\Delta F_{\boldsymbol{k}}^{n^{2}}}{[B + min\{F^{n}(\boldsymbol{k}), \hat{F}(\boldsymbol{k})\}]^{2}} \ge 0.25[P^{n}(\boldsymbol{k}) - \hat{P}(\boldsymbol{k})]^{2}\hat{P}(\boldsymbol{k}) \tag{1.2.17}$$

The iteration generally converges after 6-10 iteration steps for satellite data and in general after one iteration for data from an airborne SAR. If the azimuthal clutter cut-off length scale is not well defined (for SAR spectra with a low signal-to-noise ratio) we reduce Equation (1.2.8) to the original form of HH by setting $\alpha = 1$ and $\eta = 0$.

1.3 Matching the wave number grids of the observed and computed SAR spectra (Program `sarinv`, `subroutines invert, uwafilter`)

This routine runs for ERS-1 data only, i.e. input parameter jwmode=1.

To make the inversion algorithm consistent with the smoothing interpolation used in the generation of the ESA (ERS-1) fast delivery product an additional smoothing filter was introduced. A proper matching of the filters of the computed and observed SAR spectra is needed also to correctly repro-

duce the azimuthal cut-off.

The inversion is performed on a 128 x 128 wave number grid with a Nyquist wave number $k_x^{Nyq} = k_y^{Nyq} = 2\pi/32m^{-1}$. To carry out the inversion the ERS-1 fast delivery imagette spectra, which are given on a coarse polar coordinate grid (12 wave numbers, 12 directions from 0^o to 180^o on one half plane) are transformed from polar to cartesian (k_x, k_y) coordinates. The SAR terms in the cost function are evaluated only in the domain for which reliable SAR data were available, i.e. in the ring $(2\pi/800) \quad m^{-1} \le |\boldsymbol{k}| \le (2\pi/100)m^{-1}$. The coarse resolution of the ERS-1 SAR wave mode product causes a smearing and broadening of the retrieved SAR image spectra, affecting in particular the estimate of the azimuthal clutter cut-off length scale. To avoid unrealistic changes of the total wave energy induced by the cut-off term in the cost-function due to this artificial spectral broadening (and to achieve also some smoothing across the azimuthal cut-off region), the SAR spectra computed from the model spectra were treated in the same way as the ESA spectra: they were first transformed from the high resolution cartesian (k_x, k_y) grid to the ESA low resolution polar (k, ϕ) grid and then reinterpolated to the high resolution cartesian (k_x, k_y) grid at each computation of a simulated SAR spectrum in the inversion scheme (this can, of course, be expressed as a single net smoothing operation).

The above considerations apply for SAR spectra without clutter-noise contamination. In practice, the observed SAR spectrum consists of a superposition of the wave image spectrum and a background clutter spectrum. To first order, the two spectra are simply linearly superimposed, the modulation of the clutter noise by the ocean waves being negligible. The clutter spectrum can thus be removed by subtraction. The clutter level is estimated as the average of the five lowest spectral values of the highest wave number $(k = 2\pi/100m)$ bins of the SAR (fast delivery) spectral product (see section 1.1.1). This value is subtracted from the SAR spectra prior to processing. The clutter level is used also to calibrate the observed SAR spectrum. (Alpers and Hasselmann, 1982)

1.4 Extended retrieval algorithm (Programs `sarinv`, `partinv`, `findbest`)

In addition to the inversion scheme discussed in the previous sections, an iteration scheme for successively adapting the wave systems contained in the first-guess input wave spectrum to the wave systems of the spectrum computed from a SAR inversion is carried out. This resolves the occurence of discontinuities in the retrieved spectra in the neighbourhood of the azimuthal cut- off wave number. It also yields an improved agreement between the simulated and observed SAR spectrum when the first guess is rather far from the observations. The introduction of a term in the cost function which penalizes deviations from the fist-guess spectrum tends to inhibit large departures of the retrieved spectrum from the first guess, thus preventing the retrieved spectrum from adjusting to the observed SAR spectrum if the first guess is poor.

The iteration loop successively modifies the first-guess spectrum. At each iteration, the original information from the first guess spectrum is diluted at the cost of the new SAR information. Consequently,

the only initial information which is retained asymptotically is information which is not contained in the SAR data.

The iterations yield second-guess, third-guess, etc input spectra. We shall refer to these in the following as *input* wave spectra, as opposed to the original first-guess spectra. To distinguish the final retrieved spectra from the intermediate inversion products obtained in the course of the iteration of input spectra we shall refer to the final product as the *retrieved* spectra and the intermediate products as *inverted* spectra.

Note, that the SAR spectrum computed from the input spectrum usually does not agree with the observation as well as the SAR spectrum computed from the inverted wave spectrum from the former outer iteration loop, because only wave height, mean frequency and direction are adjusted but not the sharpness of the wave systems.

The iteration scheme makes use of two techniques: a partitioning scheme for the decomposition of wave spectra into a small number of discrete wave systems, and an algorithm for the cross-assignment of the wave systems of different spectra.

1.5 The wave partitioning scheme (Program `partinv`, `subroutine swellsep`)

The wave partitioning method subdivides the two-dimensional wave spectrum into a number of separate wave systems, each of which can be characterized by a relatively small number of mean characteristic parameters, such as the significant wave height H_s, mean frequency $\bar{f}$, and propagation direction $\bar{\theta}$.

The wave systems are defined in term of inverted ‘catchment areas’. The inverted spectrum is regarded as an orographic domain in the wave number plane. A wave system is then defined to consist of all spectral points whose ‘run-off’ drains into a local (inverted) peak. Mathematically, the wave systems can be defined (and constructed) by a simple induction rule: each spectral grid-point is assigned to the same wave system as its immediate steepest-ascent neighbour. If a grid-point has higher energy than all of its neighbours, the grid-point represents a local peak and defines a wave system.

Formally separate wave systems are coalesced if they satisfy at least one of the following three conditions: (i) they lie too close to one another, namely their peaks are only one grid-point apart, (ii) the ‘valley’ separating the peaks is not sufficiently low; namely if the minimum spectral value between two peaks is greater than 85 % of the smaller of the two peaks, or (iii) the frequency spread $\overline{\delta f^2}$ of both systems is larger than the square distance $\Delta f^2 = (f_x^{(1)} - f_x^{(2)})^2 + (f_y^{(1)} - f_y^{(2)})^2$ between the two peaks $(f_x^{(1)}, f_y^{(1)})$ and $(f_x^{(2)}, f_y^{(2)})$, where the spectral spread is defined as $\overline{f^2} = \overline{(f_x - \overline{f}_x)^2} + \overline{(f_y - \overline{f}_y)^2}^2$

with $\overline{f}_x = \overline{f\cos\theta}$, $\overline{f}_y = \overline{f\sin\theta}$, the overbar denoting standard spectral averages weighted with the spectral density.

For applications in general like wave data assimilation schemes in which not only the wave field but also the wave generating wind field is updated, a classification of wave systems into windsea, old windsea, swell and mixed windsea-swell is carried out. This is not needed for the retrieval algorithm.

1.6 Correction of the input spectrum and the cross-assignment of spectral wave systems (Program partinv, subroutines tustre, crossas)

In order to adjust iteratively the first guess wave systems to the mean characteristic parameters of the wave systems of the inverted spectrum we must first define a method for cross-assigning the separate wave systems obtained for the modelled and observed wave spectra.

For this purpose we introduce a dimensionless square "distance" between two wave systems (*i*), (*j*),

$$\Delta^2 = \frac{\left(k_x{}^i - k_x{}^j\right)^2 + \left(k_y{}^i - k_y{}^j\right)^2}{\left(k_x{}^{i^2} + k_x{}^{j^2}\right) + \left(k_y{}^{i^2} + k_y{}^{j^2}\right)} \qquad (1.6.1)$$

and "cross-assign" two wave systems, i.e. regard them as representing the same physical wave system, if $\Delta^2 < 0.75$. The cross assignment threshold value 0.75 was determined empirically as an acceptable compromise between the conflicting requirements of being able to differentiate between similar but separate wave systems on the one hand and slightly modified but nontheless matching wave systems on the other.

In general, it is possible that a given system of one of the spectra of a spectral pair can satisfy the cross assignment condition (1.6.1) for more than one system of the other spectrum. However, this occrus rather seldom, since multiple wave systems within a spectrum which lie close to one another are already coalesced, as mentioned above, during the partitioning operation (Hasselmann, S. et al., 1996) prior to cross assignment. Nevertheless, to define a unique cross assignment algorithm, the distances of all combinations of wave system pairs satisfying the condition (1.6.1) are ordered in a descending sequence, and the cross assignment is carried out following down this sequence. This can leave finally individual systems in either spectrum for which no match is found.

Wave systems of a fist-guesss spectrum without a match in the observed spectrum are transferred unchanged into the analysed spectrum. This occurs in the cross-assignment of first-guesss wave spec-

tra and SAR retrieved spectra in the iterative wave spectral retrieval algorithm for SAR data (cf. Hasselmann, S. et al., 1996) mainly for wave systems of the first-guess spectrum which lie beyond the azimuthal cutoff of the SAR retrieved spectrum.

Conversely, if the observed wave spectrum contains a system not contained in the first guess spectrum, the system is superimposed on the first guess spectrum.

Each model first guess partitioned wave system F_i (f,θ) is then rotated and rescaled such that the mean propagation direction, energy and mean frequency of the new retrieved wave system, F_i' (f,θ) agree with the corresponding parameters derived from the inverted wave systems:

$$F_i'(f,\theta) = AF_i(B\dot{f},\theta') \tag{1.6.2}$$

where θ = θ´+ const and A and B are suitably defined scaling constants. The corrected wave systems are then superimposed to yield a new spectrum. This requires some adjustment along the wave system boundaries in the spectral frequency-direction grid. Since the rotation and frequency rescaling differs for different wave systems, the modified wave systems will generally no longer fit together to exactly cover the frequency-direction grid: along some wave system boundaries there will be some overlap between adjacent wave systems, while along other boundaries gaps will arise.

No special measures are undertaken in areas of overlap. The spectral systems are simply superimposed, as in other regions of the spectral grid. Since overlapping occurs in regions of low energy, and the spectral densities are smoothed through interpolation onto the prescribed grid after rotation and frequency rescaling, no serious local energy distortions were found to be incurred through this simple approach.

Gaps are filled by parabolic interpolation in accordance with the general relation

$$y = a_0 + a_{10}x_1 + a_{20}x_2 + a_{11}x_1^2 + a_{22}x_2^2 + a_{12}x_1x_2 \tag{1.6.3}$$

where y represents the spectral energy at the frequency-direction gridpoints x_1, x_2 within a connected gap region. The coefficients a_{ij} are computed from a least square fit to the energies of a set of gridpoints i surrounding the gap region consisting of the boundary gridpoints and their next non-gap neighbours,

$$J = \sum_I (y_i - y)^2 = min \tag{1.6.4}$$

The conditions $\frac{\delta J}{\delta a_{ij}} = 0$ yield six equations for the six unknown coefficients.

The characteristic wave number vectors are defined as $k_x^{w/s} = k^{w/s}cos\tilde{\theta}$ and $k_y^{w/s} = k^{w/s}sin\tilde{\theta}$, where $k^{w/s} = (\tilde{T}^{w/s})^2/g$ and the tilde denotes average values which are modified relative to the

usual spectral-weighted definition by the introduction of an additional term $\frac{1}{f}$ to confer more weight to the low-frequency part of the spectrum which contains the relevant SAR information. Thus

$$\tilde{T} = \frac{\iint F(f,\theta)\frac{T}{f}df\,d\theta}{\iint F(f,\theta)\frac{1}{f}df\,d\theta} = \frac{\overline{T^2}}{\overline{T}} \tag{1.6.5}$$

(1.6.6)

1.7 The optimal retrieved wave spectrum (Program `findbest`)

We have carried out 5 iteration steps. Due to the 180° directional ambiguity a retrieved spectrum can develop wave systems towards the opposite direction of the first guess wave systems. Therefore, after a few iteration steps, during which the error between the simulated and observed SAR spectra decreases, the error can begin to grow again. To prevent a drift into an unrealistic retrieval, the iteration is therefore regarded as optimal when the normalized square error

$$\varepsilon^2 = \frac{\iint (S_s(f,\theta) - S_o(f,\theta))^2 df\,d\theta}{\sqrt{\iint (S_s(f,\theta))^2 df\,d\theta \iint (S_o(f,\theta))^2 df\,d\theta}} \tag{1.7.1}$$

between the simulated and observed two-dimensional SAR image specta S_s and S_o, respectively, is a minimum.

2. THE SOFTWARE

The retrieval system consists of the following programs:

1. Two programs which are run in an iteration loop, to carry out a retrieval based on an iterative up-dating of the input spectrum. (sarinv.f & partinv.f).
2. A postprocessing program (findbest.f).

For a quasi-operational retrieval a UNIX chain job (modu.jcl) is available (see appendix)

2.1 Routines provided by the user.

The user has to provide the routines for the input of the first guess wave spectra (READWSP) and the SAR observations RDSARSPEC und RDSARDATE. These are provided with the software for WAModel first guess spectra and ERS-1 SAR-UWA data respectively. For any other data the routines have to be replaced.

The software contains different routines fftsub.f for workstation (c06gcf, c06fuf) and CRAY (cfft2d)

The parameter files sargenpar.txt and sarindpar.txt containing general program parameters and parameters of the particular SAR, respectively, have to be adjusted .

The file sarinv.par (same for other programs) containing the program parameter file has to be adjusted. An example for the ERS-1 runs is provided with the software.

If the program is run for ERS-2 data with the new UWA - grid, the routine UWAFILTER has to be adjusted to the new grid.

2.2 The SAR inversion program `sarinv`

The program `sarinv` carries out one inversion (not a full retrieval!) of given SAR spectra using as input (first - or, in general, n'th guess) collocated wave spectra.

Routines:

***sarinv*: Main program.**

- `iecf_len` - computes length of character arrays.
- fftsub - FFT initialization.
- rdgenpar - reads general simulation parameters from unit 5.
- readwsp - reads all WAM first guess wave spectra in WAM format. Can be replaced by user.
- rdsardate - reads locations and dates of all SAR observations.
- select - collocates first guess and observed spectra.
- rdcollidx - reads collocation indices for retrieval iterations greater than 0.

- •rdsarspec - reads observed SAR image spectra.Routine has to be replaced for different data. Routines for ERS-1 and SWADE are provided.
- •waven - initializes wave number grid.
- •rarmtf - computes rar mtfs.
- •vbmtf - velocity bunching transfer function.
- •fttok - transforms wave spectra from f-theta (logarithmic frequency grid as in WAM) into k-space. Calculates orbital velocity and mean wave parameters. Routine can be replaced by user.
- •fullmap - full nonlinear mapping of wave spectra into SAR image spectra.
- •calibra - calibrates SAR spectra.
- •invert - inverts a SAR image spectrum into a wave spectrum.
- •wrires - interpolates wave spectra from k to (f,theta) space and outputs results and statistical *parameters.*
- •isoplot - outputs spectral fields for plotting.
- •wrcollidx - output of index array collocating first guess and observed spectra (jiter=0).

Routines called from subroutines:
FFTs are computed with CRAY routine cfft2d. Possibly has to be replaced by user.
A replacement with NAG routine c06fue(f) is installed in fftsub_wkst.f

- •aftfft - correlation and cross correlation factors.
- •calccut - compute clutter cut off.
- •chasar - input of UWA data and interpolation to (kx,ky)-grid.
- •costp - computes the cost function.
- •fimagep - computes SAR mapping series integral.
- •newspec - computes f+(delta f).
- •julian - computes number of minutes from beginning of year.
- •prepfft - computes Fourier amplitudes.
- •rdindpar - input of general parameters.
- •relmax - compute three maxima.
- •stepinfo - output of integration parameters to standard output.
- •uwafilter - smoothes spectrum by interpolating from (kx,ky)-grid into UWA-grid and vice versa.
- •variance - computes the variance of a spectrum.

Flowtrace calling tree

•`unit 5` names of input/output files. If `modu.jcl` (chain job control described in the appendix) is used this file is created automatically.

Note: The FFT-routine used in the subroutine FFTSUB is a CRAY routine `CFFT2d` and might have to be replaced on other computers. (see appendix D for a description of `CFFT2 c`)

2.2.1 The input files for `sarinv`

In order to recognise data sets easily from the file name, the following convention is used throughout the algorithm: names of data sets consist of a 4 character prefix (WAMS for WAModel first guess spectra), a 10 character date (e.g. 9211011200 for November 1st 1992 12:00)[1] and an extension .wam or .inv which denotes the data format WAM or inversion respectively (see Appendix B).

Table 1 Input units of `sarinv`

Unit	description
1	2 parameter configuration files
5	filenames to be used
7	SAR spectra in `.inv` format
30	first guess spectra in `.wam` format

The program `sarinv` needs two ASCII input files, read from `unit 1 (example for ERS-1 FDP SAR wave Mode data: sargenpar, sarindpar)` and `unit 5 (example: sarinv.in` (see Table 1)); the program assigns names to all units except unit 5 and opens the files. File names are read from unit 5. (The input/output formats are listed in Appendix B):

- `unit 1` SAR system parameters
- `unit 1` SAR imaging parameters.

1. In case of Gregorian dates or a 5 character date (e.g. 94172 for 21 June 1994) in case of Julian dates

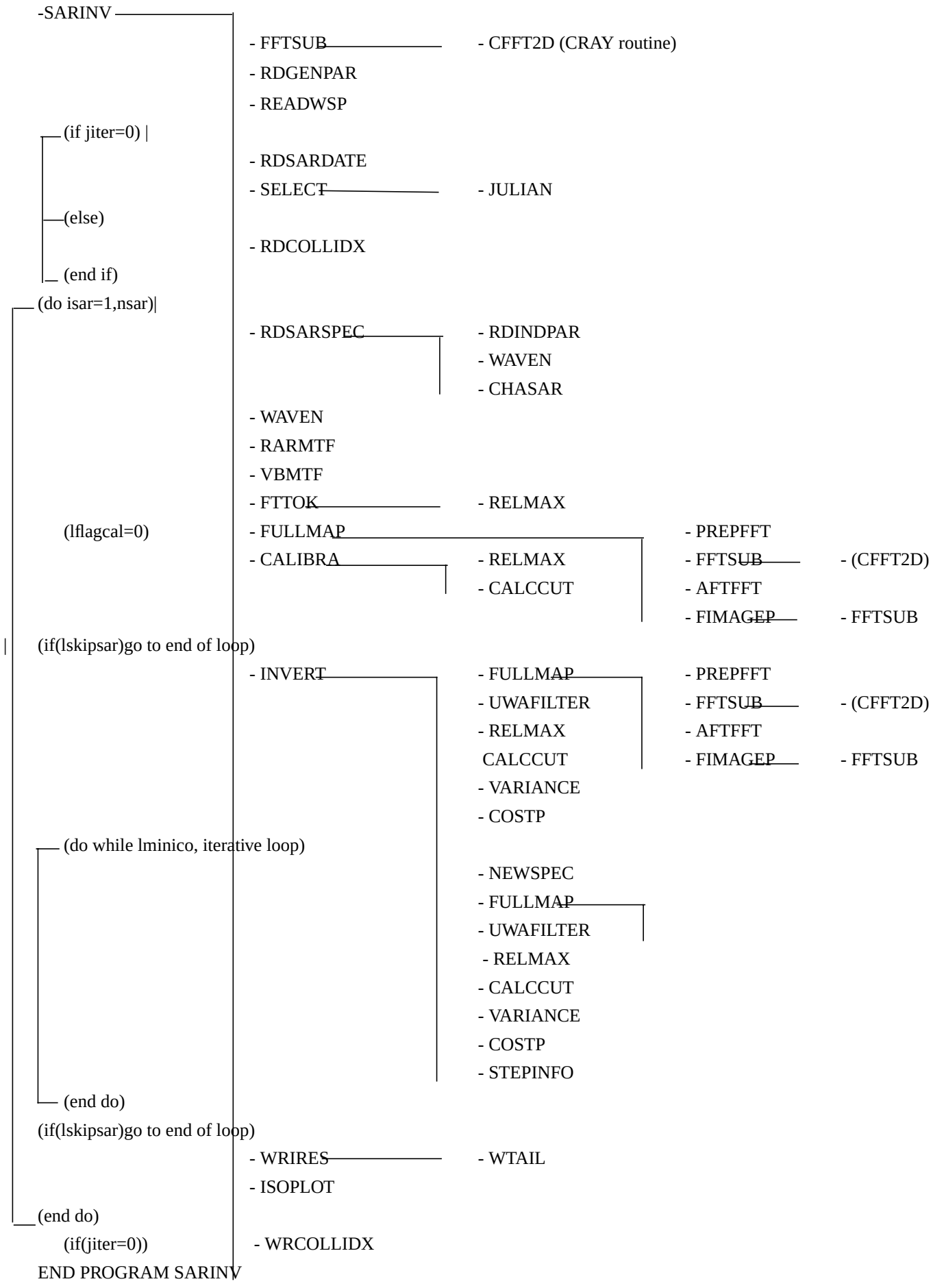
Example for an ERS-1 input file (unit 5): `sarinv.in`

```
#
# output filename for the inversion process(unit 32)
# ----------------------------------------
#
0_ou.9412241200.inv
#
# input filename for SAR data (in .inv format)(unit 7)
# ----------------------------------------
#
0_in.9412241200.inv
#
# input filename for WAM data (in .wam format)(unit 30)
# ----------------------------------------
#
0_gs.9412241200.wam
#
# filename for SAR parameter file(unit 1)
# ------------------------------
#
/pf/m/m210055/sarinv/asspar
#
# number of iterations
# --------------------
0
#
```

Example for an ERS-1 input file (unit 1): `sargenpar`

```
                    RAR MTF modulation parameter
                    ----------------------------------------
5.0     RAR MTF modulus
45.0    RAR MTF phase in [degree]
0.5     relaxation parameter of hydrodynamic modulation [1/sec]
0.0     modulus of hydrodynamic MTF associated with wind input
0.0     phase hydrod. MTF associated with wind input [degrees]

                    simulation control parameters
                    ---------------------------------
3.      max. allowed distance  [degree] (latitude/longitude)
       between WAM and colocated SAR data points.
181     max. allowed time between first guess and observed spectra.
1       print flag (1=detailed  output, 0= limited output )(unit 6)
0       output flag (0 = no output of wave and SAR spectra
        in wave number space, 1 = output)
0       flag describing form of transfer function:
        = 0 theoretical (HH), =1 parametrized.
T       True = SAR dat contain noise (calibrate SAR spectrum using
       noise.
1       flag (1 - ERS-1 wave mode data, 0 - other data)
```


-SARINV
- FFTSUB
- CFFT2D (CRAY routine)
- RDGENPAR
- READWSP
(if jiter=0) |
- RDSARDATE
- SELECT
- JULIAN
(else)
- RDCOLLIDX
(end if)
(do isar=1,nsar)|
- RDSARSPEC
- RDINDPAR
- WAVEN
- CHASAR
- WAVEN
- RARMTF
- VBMTF
- FTTOK
- RELMAX
(lflagcal=0)
- FULLMAP
- PREPFFT
- CALIBRA
- RELMAX
- FFTSUB
- (CFFT2D)
- CALCCUT
- AFTFFT
- FIMAGEP
- FFTSUB
| (if(lskipsar)go to end of loop)
- INVERT
- FULLMAP
- PREPFFT
- UWAFILTER
- FFTSUB
- (CFFT2D)
- RELMAX
- AFTFFT
CALCCUT
- FIMAGEP
- FFTSUB
- VARIANCE
- COSTP
(do while lminico, iterative loop)
- NEWSPEC
- FULLMAP
- UWAFILTER
- RELMAX
- CALCCUT
- VARIANCE
- COSTP
- STEPINFO
(end do)
(if(lskipsar)go to end of loop)
- WRIRES
- WTAIL
- ISOPLOT
(end do)
(if(jiter=0))
- WRCOLLIDX
END PROGRAM SARINV

Example for an ERS-1 input file (unit 1): `sarindpar`

```
                          radar parameters
                          ----------------
  1     radar polarisation (1 = VV, -1 = HH)
  3     number of azimuthal looks (incoherent amplitude averaging)
0.056   radar wave length (C-Band corresponding to 5.3 GHz) [m]
834850. distance radar - target [m]
7455.   satellite velocity [m/sec]
19.9    incidence angle [degrees]
 -1     radar look direction [1 = left, -1 = right]
0.78    calibr. fac.: 1 for n-look image with energy averaging,
        0.78 for 3-look ERS-1 SAR image with amplitude averaging
33.     single look azimuthal resolution.
33.     single look range resolution.
```

The parameter file `sarinv.par`:

All modules are compiled including a parameter file, which has to be provided by the user.

Example of a parameter file for an inversion of ERS-1 UWA spectra.

```
c                 User changeable Parameters
c                 --------------------------
c
C     NANG   - INTEGER  NUMBER OF DIRECT. IN MODEL SPECTRUM.
C     NFRE   - INTEGER  NUMBER OF FREQUENCIES IN MODEL SPECTRUM.
c     ksxt   - integer  azimuthal resolution of SAR spectrum (must be even)
c     ksyt   - integer  range resolution of SAR spectrum (must be even)
c
            integer nfre, nang, ksxt, ksyt
            parameter (nfre = 25, nang = 24, ksxt = 128, ksyt = 128)
c
c     jpwam  - maximum number of wam wave spectra
c     jpnsar - maximum number of observed SAR spectra
c
            integer   jpwam, jpnsar
            parameter(jpwam=3000, jpnsar=3000)
c
c     xsstep - step width sar image plane (Nyquist wave length)[m]
c
            real xsstep
            parameter (xsstep = 32.)
c
c                 Additional Parameters
c                 ---------------------
c     jpmax1 - number of maxima selected
c
            integer   jpmax1
            parameter(jpmax1=3)

c     ksx    - half of wve NUMBER plane kx.
c     ksy    - half of wve NUMBER plane ky.
c     nor    - order of Taylor expansion of exponential factor
c              in formula (50) of Hasselmann and Hasselmann (1991).
c
            integer   ksx, ksy, lot, ksxlot,KSXP1, KSYP1,  nor
            integer   ksxm1, ksym1
            PARAMETER (KSX = KSXT/2, KSY = KSYT/2, LOT = KSYT,
                  KSXLOT = 4*KSXT*LOT,KSXP1 = KSX+1,KSXM1 = KSX-1,
                  KSYP1 = KSY+1,KSYM1 = KSY-1,NOR = 13)
c
c     jpspec - spectral length
c     jpimaf - length of the real image fields (jpspec - 1)
c     jpim2  - jpimaf*2
c     jphalf - jpimaf/2 + 1

            integer jpspec, jpimaf, jpim2, jphalf
            parameter(jpspec=ksxt+1,jpimaf=ksxt,jpim2=2*ksxt,
                  jphalf=ksxt/2+1)

c           Dummy workarray parameters for Cray FFTs.
c           ----------------------------------------
            integer   ntable, nwork
            parameter(ntable=100+2*(ksxt*2))
            parameter(nwork=4*ksxt*ksyt)
```

```
c
c           Parameters not to be changed.
c           -----------------------------
c
c     zgrav   - gravitation constant
c     xeps    - values smaller than xeps are treated as zero
c     pi      - arctan(1)*4
c     zpi     - zpi = 2*pi
c
            real zgrav, xeps, pi, zpi
            parameter(zgrav = 9.806, xeps = 1.0E-12,
     &            pi = 3.1415926535897932,
     &            zpi = 2*pi)

c     akmax  - wave number of nyquest wave length
c     deltak - grid wave number increment
c
            real akmax, deltak
            parameter(akmax  = 2*pi/xsstep,
     &            deltak = 4*pi/(jpimaf*xsstep))
c
c           special ERS-1 parameters.
c           -------------------------
c
c     jpksar - wave number grid points of observed ERS-1 SAR spectrum
c     jptsar - direction grid points of observed ERS-1 SAR spectrum
c     jpisar - number of grid points of a ERS-1 SAR spectrum
c
            integer   jpksar,jptsar, jpisar, jpdata
            parameter(jpksar=12, jptsar=12, jpisar=144, jpdata=192
```

2.2.1 The output files of sarinv

The names for the named output files are read in from sarinv.in. Table 2 gives an overview over all output files including also files which are defined only by a unit number and have to be assigned to permanent files within the job control.

Table 2 Output units of `sarinv`

Unit	ass. name	description
fort.2		statistics of data quality
fort.6		standard output
fort.32	*file name*	inversion output in `.inv` format
fort.36		error between simul. and obs. sprectra
fort.40	ers1plxxx	spectral output in cartesian k-space (sub-routine isoplot)

2.3 Spectral partitioning and correction program partinv

`partinv` partitions the first guess and SAR-inverted wave spectra, carries out a cross assignment and corrects the first guess wave systems to agree in mean parameters with those of the SAR-inverted spectra. New corrected wave spectra are then computed from the inverted wave spectra.

The routines

• *partinv*: main program

• *readspec*: reads spectra in .inv format

• *correl*: combines SAR inverted partitionings if for both the distance to an input wave system is less than 0.75 (see 1.6)

• *crossas*: crossassigns WAM and SAR partitionings

• *meanst*: computes integrated values of total spectra

• *swellsep*: partitions spectra and combines wave systems

- *checkswellseppara*: checks input parameters
- *combinepeaks*: checks conditions for combining partitionings
- *docombine*: combines partitionings
- *halfspread*: combines peaks if distance between peaks is less than half of spreads
- *initswellsep*: array initialization
- *lasttwobins*: combines peaks in the last two frequency bins with closest peak
- *lowenergy*: combines partitionings with too low energy with closest wave system
- *means*: computes mean direction and mean frequency of partitionings
- *onestepapart*: combines peaks which are only one grid point apart
- *partitioning*: partitions spectra
- *sumenergy*: computes energy and frequency spread
- *threshhold*: combines peaks after threshold check
- *windsea*: defines systems as windsea or swell

•*tustre*: transforms partitionings of input spectra to mean values of partitionings of SAR inverted wave spectra and combines these to new input spectra.

• *writearrow*: output of mean values of partitionings

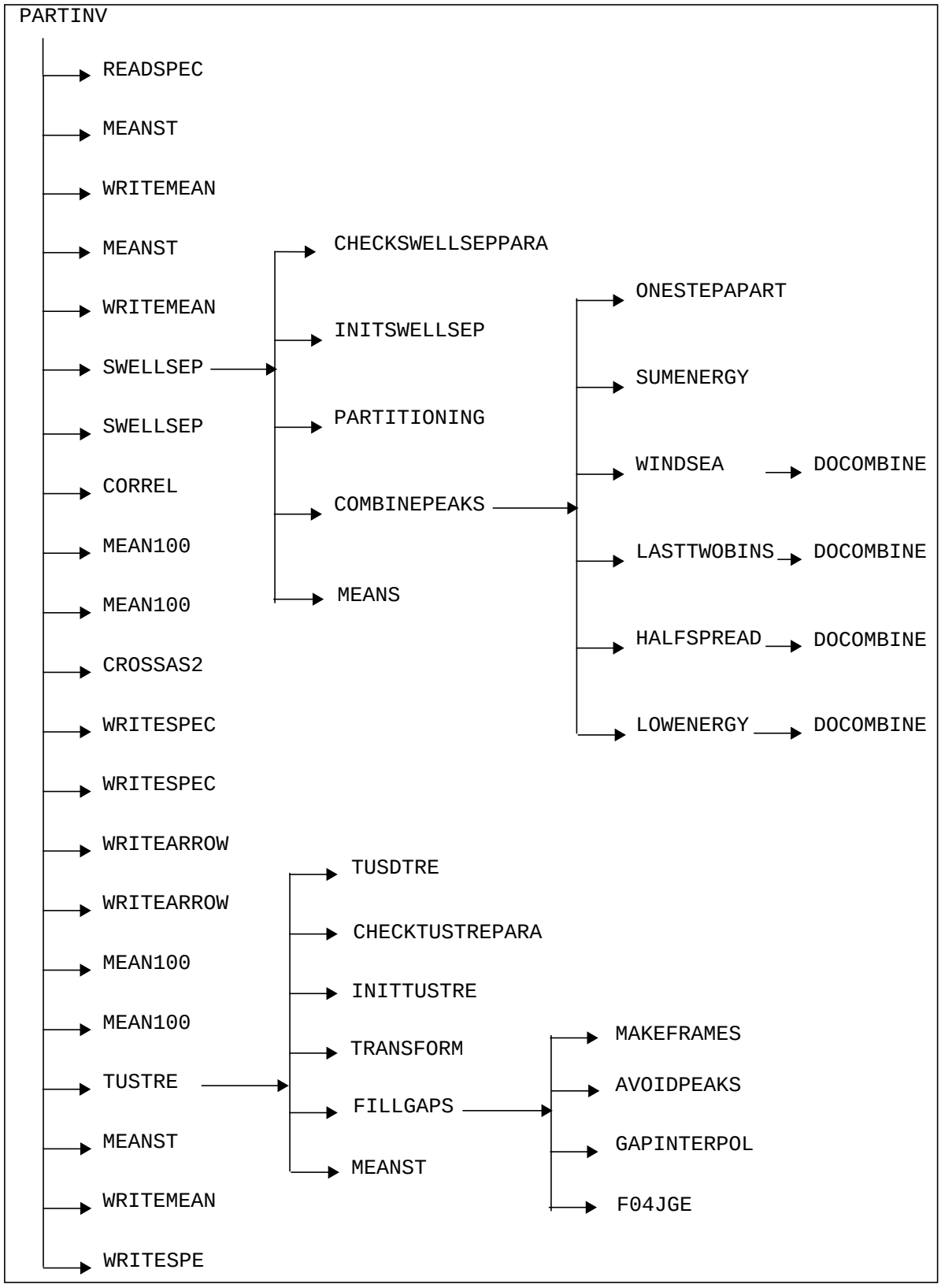
•*writemean*: output of integrated values of total spectra

• *writespec*: output of spectra and their partitionings

• *writespe*: output of corrected spectra in WAM format

Note: The routine F04JGE is a NAG library routine which solves linear least square problems. For a description see Appendix D.

2.3.1 Flowtrace calling tree



2.3.1 Input & output

There is no standard input file for `partinv`. All units are hardcoded into the program. The spectra to be partitioned are read from unit 10 (in `.inv` format). A summary of all output files is shown in Table 3.

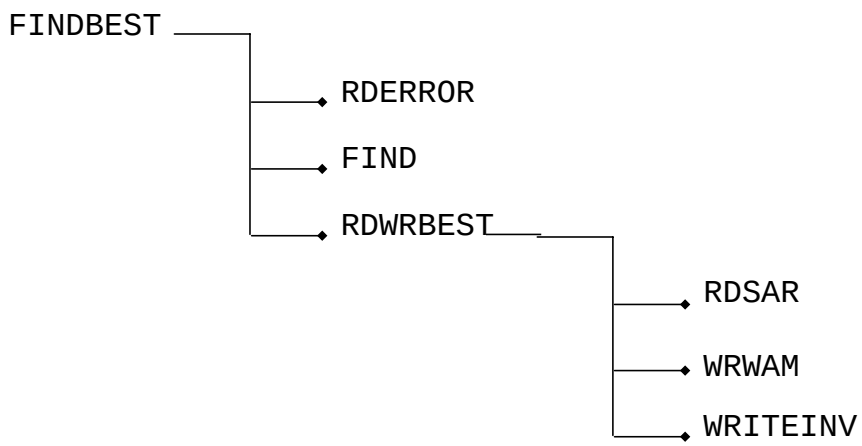
Table 3 Output units used by `partinv`

Unit	Description
20	partitionings of first guess spectra
21	Hs of first guess spectra
22	mean wave length of first guess spectra
23	Hs of partitionings of first guess spectra
24	mean wave length of partitionings of first guess spectra
26	Hs of total corrected spectra
27	mean wave length of total corrected spectra
28	corrected spectra
30	partitionings of inverted spectra
31	Hs of inverted spectra
32	mean wave length of inverted spectra
33	Hs of partitionings of inverted spectra
34	mean wave length of partitionings of inverted spectra

2.4 Best estimate wave spectrum (findbest)

`findbest` finds iteration with the smallest error between the SAR spectrum computed from the input spectrum and the observed spectrum (equation (1.7.1)) and outputs the first guess and best estimate retrieved wave spectra.

2.4.1 Flowtrace calling tree



The routines

• *rderror*: reads the spectral errors of all iterations

• *find*: finds the iteration with minimum error for each spectrum

• *rdwrbest*: reads the spectra for all iterations and outputs the spectra for the iteration with minimum error.

2.4.1 Input files

The program reads all unit numbers required from standard input (unit 5), the number of iterations, etc. This input is generated automatically if the chainjob `modu.jcl` is used. Table 4 gives an example of each input line. `INUNITRT, INUNITER` are the first unit numbers of the iteration incremented by one for each iteration. Example: Five iterations are carried out. Setting `INUNITRT` to 20, the 0th iteration will be in `fort.20`, the 1st in `fort.21` and so on up to `fort.25` for the 5th iteration.

If `modu.jcl` is used, the resulting files will be automatically moved to the directory `res` (for results) inside the current-date-directory.

Variable	Type	description	format	
IDATE	integer	reference date		
NITER	integer	max number of iterations		
INUNITFG	integer	unit number of first guess data		Input
INUNITRT	integer	first unit number of inverted data		
INUNITER	integer	first unit number of errorfiles	fort. 36 of sarinv	
OUTINVU-NIT	integer	unit number of retrieved output	`.inv` format	Output
OUTUNI-TRT	integer	unit number of retrieved output	`.opt` format	
OUTU-NITFG	integer	unit number of first guess spectra	`.wam` format	
IDXUNIT	integer	output unit for a summary file best iter. no and error	`.idx` format	
LOLD	boolean	TRUE	**old** `.inv` format (26 bins)	

2.5 The grids

The follwoing grids are used throughout the inversion algorithm:

- **The ERS-1 SAR wave mode FDP grid (**this can be changed to other grids**):**
  The observed SAR wave mode spectra extracted from the fast delivery product are given on a polar wave number grid:
  12 wave numbers corresponding to wave lengths from 100 m to 1000 m with logarithmic increment:

  $k_j = 10^{-1/11(j-1)} k_e \quad , j=1, ..., 12$, with $k_e = \frac{2\pi}{100}$ 1/m

  12 directional sectors of 15^o between 0^o and 180^o counter-clockwise relative to the satellite flight direction (azimuth), corresponding to a half-plane spectrum.

- **The WAM model grid** (normally used for WAM model runs, can be changed):
  The first guess WAM model and the retrieved wave spectra are given on a frequency-directional grid

- $f_j = 1.1^{(j-1)} f_1 \quad , j=1, ..., NFRE$, $f_1 = 0.04177$

  where NFRE is typically 25 for global runs correspondig to wave lengths between roughly 9 m and 895 m

- **The cartesian wave number grid:**
  The nonlinear mapping and computation of the cost function is performed on a cartesian (k_x, k_y) wave number grid

  $k_{x_j}, k_{y_j} = \pm\Delta k, \pm 2 \cdot \Delta k, \ldots, \pm n \cdot \Delta k$ with n=ksx, $\Delta k = \frac{2\pi}{\lambda_{NQ}}$ *1/m*.

  Parameters λ_{NQ}, *ksx* are defined in the input parameter file sarinv.par.

Appendix A: The UNIX chainjob `runsar.jcl`

The jcl provided with the software will run the test job on a unix system. There are some variables at the beginning of the job, which have to be provided by the user.
The job contains a program `colldata which` extracts input data for an inversion run, i.e. extracts input wave data in .wam format from a .inv formated file.
This is not a program contained in the official inversion software.

```
#!/bin/csh -e
#QSUB -eo
#QSUB -q M3
#QSUB -o runerssar.out
#QSUB -x
#
#============================================
# This job runs a full retrieval for ERS-1 SAR spectra.
#
# Please fill in the required data below
#
#============================================
#
# To be set up by user
# ====================
# directory containing executable files.
# ----------------------------------------------
set cycle = sar_cycle_3
# path to executables except sarinv.x
# ------------------------------------------
set execpath    = $MFHOME/$cycle/exec
# path to sarinv.x
# -----------------
set execsar     = $MFHOME/$cycle/exec
#directory of input file for program colldata
# ------------------------------------------------
set sourcepath  = $HOME/$cycle/jobs
# path to data
# ---------------
set datapath    = $HOME/$cycle/testdata
#“working” directory
# ------------------------------------
set workingpath = $MFHOME/sar
# Prefix of the dataname - the time will be added
# ------------------------------------------------------
# ---file containing the spectra in .inv format
set wadana = TEST
# initial date
# -------------
#  Form of YYMMDDHHMM or YYJJJ
```

```
set juld = 9211031200
# Number of iterations
#-------------------------
set niter = 1
# Unitnumber for collocation index
# ---------------------------------------
set collidxunit = 66
#
# specify ieee = y for input AND output in IEEE-format on cray
# specify ieee = n for output only (write to res/${juld}.inv in IEEE)
# anything else will use standard binary encoding (recomended)
# ------------------------------------------------------------------------
set ieee = n
#
#=====================================================
# Thats all the data needed. Thanks for filling out !
#=====================================================
#
# Create file containing all relevant job initialisations
#-----------------------------------------------------------------
# set assign options
# ---------------------
set asopt = “-N ieee_dp -f 77 -F f77”
# initialise iteration counter:
#---------------------------------
set jiter = 0
@ jiter_one = ${jiter} + 1
# initialise process
#---------------------
setenv proc pre
# create directory for respective day or time window of data:
#-------------------------------------------------------------------------
cd ${workingpath}
if (-d ${juld}) rm -r ${juld}
mkdir ${juld}
cd ${workingpath}/${juld}
# Preparation of original data for sar inversion
#-----------------------------------------------------
if (! -e ${datapath}/${wadana}${juld}) then
     echo “No data available on data directory” | write $user
    exit
endif
if (! -d ${jiter}.it) mkdir ${jiter}.it
cd ${jiter}.it
# copy original data to user’s disk
#---------------------------------------
# create file in .wam format from given
# .inv format using colldata.f
#---------------------------------------------
```

```
cp ${datapath}/${wadana}${juld} fort.40
if (${ieee} == y) assign ${asopt} u:40
if (-e coll${jiter}.out) rm coll${jiter}.out
# run colldata.x
#---------------
grep -v “#” ${sourcepath}/modu.coll.in | ${execpath}/colldata.x > coll${jiter}.out
#
mv fort.40 ${jiter}_in.${juld}.inv
mv fort.20 ${jiter}_gs.${juld}.wam
if (${ieee} == y) assign -R u:40
echo 0th iteration done
###############################################################################
# -----------------------------------------
# run jiter.it inversion (start of loop)
#-----------------------------------------
###############################################################################
inversion:
        set proc = sar
        cd ${workingpath}/${juld}/${jiter}.it
# create input file for sarinv.f
#----------------------------------
        if ( -e ${execsar}/sargenpar.in ) rm ${execsar}/sargenpar.in
        if ( -e ${execsar}/sarindpar.in ) rm ${execsar}/sarindpar.in
        grep -v “#” $HOME/$cycle/sarinv/sargenpar.txt > ${execsar}/sargenpar.in
        grep -v “#” $HOME/$cycle/sarinv/sarindpar.txt > ${execsar}/sarindpar.in
        if (-e sarinv.in) rm sarinv.in
        echo “${juld}”                                  >sarinv.in
        echo “${jiter}_ou.${juld}.inv”          >>sarinv.in
        if (${jiter} == 0) then
           if (${ieee} == y) assign ${asopt} u:7
              echo “${jiter}_in.${juld}.inv”  >>sarinv.in
        else
              if (${ieee} == y) assign -R u:7
              echo “${workingpath}/${juld}/0.it/0_ou.${juld}.inv” >>sarinv.in
              cp ${workingpath}/${juld}/0.it/fort.$collidxunit .
        endif
        echo “${jiter}_gs.${juld}.wam”        >>sarinv.in
        echo “${execsar}/sargenpar.in”        >>sarinv.in
        echo “${execsar}/sarindpar.in”        >>sarinv.in
        echo “${jiter}”                   >>sarinv.in
        echo “$collidxunit”                 >>sarinv.in
        echo “ “                          >>sarinv.in
        if (-e sar${jiter}.out) rm sar${jiter}.out
        # run sarinv.x
        #---------------
        cat sarinv.in | ${execsar}/sarinv.x  > sar${jiter}.out
        ##########################################
        # partitioning and correction of inverted spectrum
        #-------------------------------------------------------
```

```
        cd ${workingpath}/${juld}/${jiter}.it
        if (! -d part) mkdir part
        cd part
        cp ../${jiter}_ou.${juld}.inv fort.10
        if (-e part${jiter}.out) rm part${jiter}.out
        # run partinv.x
        #-----------------
        ${execpath}/partinv.x > part${jiter}.out
        cp fort.28 ${jiter_one}_gs.${juld}.wam
        cp fort.10 ${jiter_one}_in.${juld}.inv
        cd ${workingpath}/${juld}
        # create directory and job for next iteration inversion
        # and move data to this directory
        #-------------------------------------------------------------
        mkdir ${jiter_one}.it
        mv ${jiter}.it/part/${jiter_one}_gs.${juld}.wam ${jiter_one}.it/.
        mv ${jiter}.it/part/${jiter_one}_in.${juld}.inv ${jiter_one}.it/.
        # prevent too many iterations
        #----------------------------------
        if (${jiter} == 10) setenv niter 10
        if (${jiter} == ${niter}) then
            set proc = pos
            goto postproc
        else
            set proc = prt
        endif
        echo iteration ${jiter}
        @ jiter = ${jiter} + 1
        @ jiter_one = ${jiter} + 1
        #
goto inversion
#
#############################################################################
#
#
# find iteration with smallest error.
#---------------------------------------
postproc:
#
        cd ${workingpath}/${juld}
        if (-d res) rm -r res
        mkdir res
        cd res
        pwd
        #run findbest
        #--------------
        # set specific unit numbers required by program findbest
        #
        set unitfg = 10
        set unitrt = 20
```

```
        set uniter = 50
        set outinvunit = 30
        set outunitrt = 80
        set outunitfg = 81
        set idxunit = 82
        set lold = false
        # link input files
        #----------------
        set iiter = 0
        do_it:
            @ ufg = $unitfg + $iiter
            @ uer = $uniter + $iiter
            ln ${workingpath}/${juld}/${iiter}.it/${iiter}_ou.${juld}.inv fort.${ufg}
            ln ${workingpath}/${juld}/${iiter}.it/fort.36 fort.${uer}
            @ iiter++
        if (${iiter} <= ${jiter}) goto do_it
        # create standard input file for findbest.f
        #------------------------------------------------
        if (-e findbest.in) rm findbest.in
        echo ${juld}          >findbest.in
        echo ${jiter}         >>findbest.in
        echo ${unitfg}         >>findbest.in
        echo ${unitrt}         >>findbest.in
        echo ${uniter}         >>findbest.in
        echo ${outinvunit}  >>findbest.in
        if (${ieee} == y || ${ieee} == outonly) assign ${asopt} u:${outinvunit}
        echo ${outunitrt}       >>findbest.in
        echo ${outunitfg}       >>findbest.in
        echo ${idxunit}        >>findbest.in
        echo ${lold}           >>findbest.in
        # ------------------------------------------------
        # also use next lines to obtain retrieved specta with
        # certain quality flags only
        # ------------------------------------------------
        #echo “1”                        >>findbest.in
        #echo “2”                        >>findbest.in
        #echo “4”                        >>findbest.in
        #echo “6”                        >>findbest.in
        echo ““                      >>findbest.in
        if (-e ${workingpath}/${juld}/find${jiter}.out) \
            rm ${workingpath}/${juld}/find${jiter}.out
        # execute findbest.x
        #----------------------
        ${execpath}/findbest.x <findbest.in \
            >${workingpath}/${juld}/find${jiter}.out
        mv fort.${idxunit} ${juld}.idx
        mv fort.${outinvunit} ${juld}.inv
        mv fort.${outunitrt} ${juld}.opt
        mv fort.${outunitfg} ${juld}.wam
        rm -f fort.* findbest.in
# end post processing
###############################################################################
# collect results
#-----------------
@ jiter_min = $jiter - 1
cp ${datapath}/${wadana}${juld} .
```

Appendix B: Data formats

All file formats are identified by a three letter extension to the file name.

> *Example:* `mydata.wam` is a file in `.wam` format - this does NOT necessarily mean, that the data were obtained by running the WAM-model! The extension ONLY describes the file format and NOT its contents.

All formats are ”Fortran unformatted sequential”.

The .wam format

Files in `.wam` format contain ocean wave spectra. This could either be spectral output from the WAM-model, or n’th guess spectra computed during the iteration of the input spectra.

An example of a Fortran code for a .wam - format:

```
DO ISPEC=1, NSPEC
   WRITE(OUTUNIT) LONG(ISPEC),LAT(ISPEC),DATE(ISPEC), FLOAT(NANG),
                  FLOAT(NFRE),TH0,FR1,CO
   WRITE(OUTUNIT) HST(ISPEC)
   WRITE(OUTUNIT) THQT(ISPEC)
   WRITE(OUTUNIT) FMEANT(ISPEC)
   WRITE(OUTUNIT) U10(ISPEC)
   WRITE(OUTUNIT) THW(ISPEC)
   WRITE(OUTUNIT)
            ((SPEC(ISPEC,IANG,IFRE),IANG=1,NANG),IFRE=1,NFRE)}
END DO
```

NSPEC gives the number of spectra to be written to the file. The remaining variables are explained in Table 5.

Table 5 Variables of `.wam` format

Variable	Meaning	Type
LONG	longitude of spectrum (degree)	real
LAT	latitude of spectrum (degree)	real
DATE	date of spectrum (YYMMDDHHMM)	real
NANG	number of directions	integer
NFRE	number of frequencies	real
TH0	first direction with respect to true North (=0)	real
FRE1	first frequency (0.04177) on frequency grid	real
CO	factor for logarithmic frequency array (=1.1)	real
HST	significant wave height (meters)	real
THQT	mean direction (degree clockwise relative to North)	real
FMEANT	mean frequency (Hz)	real

Table 5 Variables of `.wam` format

Variable	Meaning	Type
U10	local wind speed in 10 meters height (m/s)	real
THW	local wind direction (degree, clockwise relative to North)	real
SPEC	2d wave spectrum	real

The `.opt` format

The `.opt` format is identical to the `.wam` format. The suffix `.opt` is only being used to identify the final retrieved spectra.

The `.inv` format

The `.inv` format is the output format of `sarinv`. The data consist of six parts:

• Lines 1-4

The time, location, significant wave height, direction and mean frequency of the first guess and the SAR image spectrum.

• Lines 5-7

Statistical parameters for simulated, first guess and observed spectra respectively.

• Lines 8:

- track: satellite flight direction clockwise from North in degrees.

- final value of cost function

- u10: u_{10} windspeed at 10m height

- thw: wind direction in degree clockwise from North

- xcorwa: correlation between input and inverted wave spectrum

- xcorfg: correlation between simulated and observed SAR spectrum

- xcorbe: correlation between simulated best estimate and observed SAR spectrum

• Line 9:

- jpksp: number of frequencies

- jpthe: number of directions

- quality flag (jqual):

* = 0: cost ≤ 0.1, excellent results

* = 1: 0.1 ≤ cost ≤ 0.5, results will be accepted

* = 2: cost ≥ 0.5, results are questionable
* = 3: iteration unstable
* = 4: no azimuthal clutter cut-off adjustment
* = 5: H_s ≤ 0.1 m, spectrum rejected
* = 6: signal to noise ratio ≤ 3dB

- jpdata: length of array x5 data containing ERS-1 ASR Wave Mode FDP product.

• Lines 10-11 The first guess and the inverted spectrum, respectively.

• Line 12 The original SAR Wave Mode Product (ERS.SWM.UWA). This is a real array of 192 elements:

- 1. satellite ID
- 2. satellite track number
- 3. satellite instrument
- 4. year
- 5. month
- 6. day
- 7. hour
- 8. min
- 9. sec
- 10. latitude of observation (center of imagette)
- 11. longitude of observation (center of imagette)
- 12. SAR polarization (for ERS-1 =1: vv pol.)
- 13. Bragg frequency [GHz] (5.3)
- 14. number of azimuthal looks (3)
- 15. range resolution [m] (33)
- 16. azimuth resolution [m] (33)
- 17. SAR incidence angle [degree] (19.9 until May 1995, 23 from June 1995)

- 18. flight direction [degree] clockwise relative to North.

- 19. radar platform velocity [m/sec]

- 20. flight altitude of satellite [m]

- 21. noise (is computed again in the program)

- 22. dummy

- 23. dummy

- 24. dummy

- 25-36: Array of wave numbers corresponding to wave lengths:
  100.0, 123.0, 152.0, 187.0,
  231.0, 285.0, 351.0, 433.0, 534.0, 658.0, 811.0, 1000. (for ERS-1)

- 37-48: directions in degrees from 7.5 to 172.5 in steps of 15^o.

- 49-192: SAR power spectrum 12 wave numbers (outer loop), 12 directions (inner loop), intensities in 8 bit from 0 to 255

An example of a Fortran code for the `.inv` format:

```
DO ISPEC = 1, NSPEC
DO I = 1,2
WRITE(OUTUNIT)LONG(ISPEC,I),LAT(ISPEC,I),DATE(ISPEC,I)
WRITE(OUTUNIT)HST(ISPEC,I), FMEANT(ISPEC,I),xkp(ISPEC,I),
             THQT(ISPEC,I), xthp(ISPEC,I), xsigt(ISPEC,I)
END DO

WRITE(OUTUNIT)xvssi(ispec),xkmsi(ispec),xkpsi(ispec)
           ,xthmsi(ispec),xthpsi(ispec),xsigsi(ispec)
WRITE(OUTUNIT)xvsbe(ispec),xkmbe(ispec),xkpbe(ispec)
           ,xthmbe(ispec),xthpbe(ispec),xsigbe(ispec)
WRITE(OUTUNIT)xvsob(ispec),xkmob(ispec),xkpob(ispec)
           ,xthmob(ispec),xthpob(ispec),xsigob(ispec)

WRITE(OUTUNIT) TRACK(ISPEC), COST(ISPEC),USTAR(ISPEC),
               THW(ISPEC), xcorwa(ispec),
               xcorfg(ispec),xcorbe(ispec)
WRITE(OUTUNIT)jpksp(ispec),jpthe(ispec),jqual(ispec),
              jpdata(ispec)

WRITE(OUTUNIT)
     ((SPEC(ISPEC,IANG,IFRE,1),IANG=1,NANG),IFRE=1,NFRE)
WRITE(OUTUNIT)
     ((SPEC(ISPEC,IANG,IFRE,2),IANG=1,NANG),IFRE=1,NFRE)

WRITE(OUTUNIT) (xsdata(ispec,icount),icount=1,jpdata)
END DO
(see routine accuwa or wrires)
```

The .idx format

To every retrieved spectrum the program findbest gives some quality statistics of the best iteration. These are written to the unit IDXUNIT in the following format. An example of a Fortran code for a .idx format :is:

```
DO ISPEC=1, NSPEC
  WRITE(IDXUNIT,'(11(1X,E10.4),1X,F13.0,1X,I2)')
&  erfgob(INDEX(ISPEC),(ISPEC)),ersiob(INDEX(ISPEC),(ISPEC))
&  xcost(INDEX(ISPEC),(ISPEC)),xcorwa(INDEX(ISPEC),(ISPEC))
&  xcorfg(INDEX(ISPEC),(ISPEC)),xcorbe(INDEX(ISPEC),(ISPEC))

& xlonw(INDEX(ISPEC), xlatw(INDEX(ISPEC), xdatwj(INDEX(ISPEC), jqual(INDEX(ISPEC),(ISPEC))
END DO
```

Table 6 The variables have the following meaning**.**

Variable	Meaning
INDEX	number of iterations required
jqual	quality flag (see .inv format)
erfgob	error between simulated SAR image spectrum and observed SAR image spectrum
ersiob	error between best estimate simulated SAR image spectrum and observed SAR image spectrum
xcost	cost function value
xcorwa	pattern correlation between first guess and inverted wave spectrum
xcorfg	pattern correlation between SAR image spectrum computed from first guess and observed SAR spectrum
xcorbe	pattern correlation between best estimate and observed SAR image spectrum

Appendix C:The CRAY FFT routine cfft2d

NAME

CFFT2D- Applies a multitasked two-dimensional complex Fast Fourier Transform (FFT)

SYNOPSIS

CALL CFFT2D (*isign, n1, n2, scale, x, inc1x, inc2x, y, inc1y, inc2y, table, ntable, work, nwork*)

DESCRIPTION

CFFT2D computes the two-dimensional complex Fourier Transform of each column of the complex
matrix X, and it stores the results in the complex matrix Y. For most purposes, CFFT2D is superseeded by the Cray Standard FFT routine CCFFT2D(3).

Suppose the matrices are stored in Fortran arrays dimensioned as follows:

COMPLEX X(0:Nl-1, 0:N2-1)
COMPLEX Y(0:Nl-1, 0:N2-1)

CFFT2D computes the following formula

$$Y_{k1,\,k2} = scale \sum_{j1=0}^{n1} \sum_{j2=0}^{n2-1} x_{j1\,j2} \omega_1^{j1 \cdot k1} \omega_2^{j2 \cdot k2}$$

for *k1 = 0, ...,n1 - 1*
k2 = 0, ...,n2 - 1

where

$$\omega_1 = e^{\frac{(isign)2\pi i}{n1}}, isign = (\pm 1), \qquad \pi = 3.14159\ldots, \qquad e = 2.71828\ldots, \qquad i = \sqrt{-1}$$

$$\omega_2 = e^{\frac{(isign)2\pi i}{n2}}, n1 = N1, \qquad n2 = N2$$

In this manual, when *isign* =+1 it is called the *forward transform*, and when *isign* =-l it is called the
inverse transform.

This routine has the following arguments:

isign Integer. (input)
Specifies whether to initialize the *table* array, or whether to do the forward or inverse transform:
0 Initializes the *table* array
+1 Computes the forward transform
-1 Computes the inverse transform

n1 Integer. (input)
Transform size in the first dimension.
If *n1* is not positive, CFFT2D returns without performing a transform.

n2 Integer. (input)
Transform size in the second dimension.
If *n2* is not positive, CFFT2D returns without performing a transform.

scale Real. (input)
Real scale factor.
Each element of the output array is multiplied by *scale* after taking the Fourier transform, as defined previously.

x Complex array. (input)
Input array of values to be transformed.

inc1x Integer. (input)
x increment in the first dimension. That is, the address increment between successive complex row elements of input array *x*. To use every row element in a given column, set *inc1 x* = 1. The value of *inc1x* must not be 0.

inc2x Integer. (input)
x increment in the second dimension. That is, the address increment between successive complex column elements of input array *x*. To use every column element in a given row, set *inc1x* to be twice the leading dimension of the complex array x. The value of i*nc2x* must not be 0.

y Complex array. (output)
Output array of transformed values. The output array may be the same as the input array; in which case, the transform is done in place.

inc1y Integer. (input)
y increment in the first dimension. That is, the address increment between successive complex row elements of output array *y.* To use every row element in a given column, set *inc1y* = 1. The value of *inc1y* must not be 0.

inc2y Integer. (input)
y increment in the second dimension. That is, the address increment between successive complex column elements of output array *y.* To use every column element in a given row, set *inc1y* to be twice the leading dimension of the complex array *y.* The value of *inc2y* must not be 0.

table Real array of dimension *ntable*. (input or output)
Table of factors and trigonometric functions. This array may be initialized by a call to CFFT2D with *isign* = 0.

ntable Integer. (input)
Number of (real) words in *table*. The value of *ntable* should be at least
2(n1+n2)+ 100
If not enough space is provided, CFFT2D will print an error message and stop.

work Real array. (scratch output)
Work array of size *nwork*. This is a scratch array used for intermediate calculations.
It must be a different address space from the input and output arrays.

nwork Integer. (input)
Number of words in the *work* array. The value of *nwork* should be at least
4max(*n1 ,n2*)min(*n1 ,n2, 16ncpus*)
where *ncpus* is the number of CPUs used in the calculation.
If not enough space is provided, CFFT2 D prints an error message and stops.

Appendix D: The NAG library routine F04JGE - NAG for solving linear least square problems

Fortran Library Routine Document

1. Purpose

F04JGE finds the solution of a linear least-squares problem, $Ax = b$, where A is a real m by n (m n) matrix and$b$ is an m element vector. If the matrix of observations is not of full rank, then
the minimal least-squares solution is returned.

2. Specification

```
 SUBROUTINE F04JGE (M, N, A, NRA, B, TOL, SVD, SIGMA, IRANK, WORK,
1 LWORK, IFAIL)
INTEGERM, N, NRA, IRANK, LWORK, IFAIL
real A(NRA,N), B(M), TOL, SIGMA, WORK(LWORK)
LOGICAL SVD
```

3. Description

The minimal least-squares solution of the problem $Ax = b$ is the vector x of minimum (Euclidean) length which minimizes the length of the residual vector $r = b - Ax$..
The real m by n (mn) matrix A is factorized as

$$A = Q\begin{pmatrix} U \\ 0 \end{pmatrix}$$

where Q is an m by m orthogonal matrix and U is an n by n upper triangular matrix. If U is of
full rank, then the least-squares solution is given by

$x = (U^{-1}\ 0)\ Q^T b.$

If U is not of full rank, then the singular value decomposition of U is obtained so that U is factorized as

$U = RDP^{\mathrm{T}}$,

where R and P are n by n orthogonal matrices and D is the n by n diagonal matrix

$D = \mathrm{diag}(\sigma_1,\sigma_2,...,\sigma_n)$

with $\sigma_1\ \sigma_2\ ..\sigma_n\ 0$, these being the singular values of A. If the singular values σ_{k+1} ,...,σ_n are negligible, but σ_k is not negligible, relative to the data errors in A, then the rank of A is taken to

be *k* and the minimal least-squares solution is given by

$$x = P\begin{pmatrix} S^{-1} & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} R^T & 0 \\ 0 & 0 \end{pmatrix}Q^T b,$$

where $S = \text{diag}(\sigma_1, \sigma_2, \ldots \sigma_k)$.
This routine obtains the factorizations by a call to F02WDF.
The routine also returns the value of the standard error

$$\sigma = \sqrt{\frac{r^T r}{m-k}}, \text{ if } m > k,$$

$= 0$, if $m = k$, $r^T r$ being the residual sum of squares and *k* the rank of *A*.

4. References

[1]LAWSON, C.L. and HANSON, R.J.
Solving Least-squares Problems.
Prentice-Hall, New Jersey, 1974.

5. Parameters

1:M - INTEGER.*Input*
On entry: *m*, the number of rows of *A*.
Constraint:. M N.

2:N - INTEGER.*Input*
On entry: *n*, the number of colurnns of A.
Constraint: 1 N M.

3:A(NRA,N) - ***real*** array.*Input/Output*
On entry: the *m* by *n* matrix A.
On exit: if SVD is returned as .FALSE., *A* is overwritten by details of the *QU* factorization of *A* (see F02WDF for further details) . If SVD is returned as .TRUE., the first *n* rows of *A* are overwritten by the-right-hand singular vectors, stored by rows; and the remaining rows of the array are used as workspace.

4:NRA - INTEGER.*Input*
On entry: the first dimension of the array *A* as declared in the (sub)program from which F04JGF is called. *Constraint*: NRA M.

5: B(M) - ***real*** array.*nputl Output*
On entry: the right-hand side vector *b.*
On exit: the first *n* elements of B contain the minimal least-squares solution vector *x.* The remaining *m* - *n* elements are used for workspace.

6:OL - **real**.*Input*
On entry: *a* relative tolerance to be used to determine the rank of *A*. TOL should be chosen as approximately the largest relative error in the elements of *A*. For example, if the elements of *A* are correct to about 4 significant figures then TOL should be

set to about $5x10^{-4}$. See Section 8 for a description of how TOL is used to determine rank. If TOL is outside the range (ε, 1.0), where ε is the **machine precision**, then the value ε is used in place of TOL.
For most problems this is unreasonably small.

7:SVD - LOGICAL. *Output*
On exit: SVD is returned as .FALSE. if the least-squares solution has been obtained from the *QU* factorization of *A*. In this case *A* is of full rank. SVD is returned as .TRUE. if the least-squares solution has been obtained from the singular value decomposition of *A*.

8:SIGMA - **real**. *Output*

On exit: the standard error, i.e. the value $\sqrt{r^T r/(m-k)}$ when $m > k$, and the value zero when $m = k$. Here r is the residual vector b - Ax and k is the rank of A.

9:IRANK—INTEGER. *Output*
On exit: *k*, the rank of the matrix *A*. It should be noted that it is possible for IRANK to be returned as *n* and SVD to be returned as .TRUE.. This means that the matrix *U* only just failed the test for non-singularity.

10: WORK(LWORK — ***real*** array. *Output*
On exit: if SVD is returned as FALSE., then the first *n* elements of WORK contain information on the *QU* factorization of A (see parameter A above and F02WDF), and WORK (*n*+ *1*) contains the condition number $||U||_E \, ||U^{-1}||_E$ of the upper triangular matrix *U*.
If SVD is returned as .TRUE., then the first *n* elements of WORK contain the singular values of *A* arranged in descending order and WORK(*n*+*l*) contains the total number of iterations taken by the *QR* algorithm. The rest of WORK is used as workspace.

11:LWORK—INTEGER. *Input*
On entry: the dimension of the array WORK as declared in the (sub)program from which F04JGF is called.

Constraint: $LWORK \geq 4\chi N$.

12: FAIL - INTEGER. *InputlOutput*
On entry: IFAIL must be set to 0, -1 or 1. For users not familiar with this parameter (described in Chapter P01) the recommended value is 0.
On exit: IFAIL = 0 unless the routine detects an error (see Section 6).

References

Alpers, W. and K. Hasselmann, 1982:

Spectral signal-to-clutter and thermal noise proporties of ocean wave imaging synthetic aperture radars. *Int. J. Remote sensing, **3**, 423-446*

Hasselmann, K. and S. Hasselmann, 1991:

On the nonlinear mapping of an ocean wave spectrum into a SAR image spectrum and its inversion. *J. Geophys. Res.*, **96**, 10, 713-10, 729.

Brüning, C., S. Hasselmann, K. Hasselmann, S. Lehner and T. Gerling, 1994:

First evaluation of ERS-1 synthetic aperture radar wave mode data. *The Glob. Atm. Ocean System* **2**, 61-98.

Hasselmann, S., K. Hasselmann and C. Brüning, 1994:

Extraction of wave spectra from SAR image spectra, in: Dynamics and modelling of ocean wave spectra, edited by G. Komen, pp 391-401 *Cambridge University Press, England.*

Hasselmann, S., C.Brüning, K. Hasselmann and P. Heimbach, 1996:

An improved algorithm for the retrieval of ocean wave spectra from synthetic aperture radar image spectra. *J. Geophys. Res.*, 101, 16, 615-16, 629.