SWAN

USER MANUAL

SWAN Cycle III version 40.72A

SWAN USER MANUAL

by	:	The SWAN team
mail address	:	Delft University of Technology Faculty of Civil Engineering and Geosciences Environmental Fluid Mechanics Section P.O. Box 5048 2600 GA Delft The Netherlands
e-mail	:	swan-info-citg@tudelft.nl
home page	:	*http://www.fluidmechanics.tudelft.nl/swan/index.htm*

Contents

Chapter 1

Introduction

The information about the SWAN package is distributed over five different documents. This User Manual describes the complete input and usage of the SWAN package. The Implementation Manual explains the installation procedure of SWAN on a single- or multi-processor machine with shared or distributed memory. The System documentation outlines the internals of the program and discusses program maintenance. The Programming rules is meant for programmers who want to develop SWAN. The Scientific/Technical documentation discusses the physical and mathematical details and the discretizations that are used in the SWAN program. The mapping of these numerical techniques in SWAN code is also discussed.

In Chapter 2 some general definitions and remarks concerning the usage of SWAN, the treatment of grids, boundary conditions, etc. is given. It is advised to read these definitions and remarks before consulting the rest of the manual. Chapter 3 gives some remarks concerning the input and output files of SWAN. Chapter 4 describes the complete set of commands of the program SWAN.

It is strongly advised that users who are not so experienced in the use of SWAN first read Chapters 2 and 3.

This Manual also contains some appendices. In Appendix A definitions of several output parameters are given. Appendix B outlines the syntax of the command file (or input file). A complete set of all the commands use in SWAN can be found in Appendix C. Appendix D described the format of the files for spectral input and output by SWAN.

Chapter 2

General definitions and remarks

2.1 Introduction

The purpose of this chapter is to give some general advice in choosing the basic input for SWAN computations.

SWAN is a third-generation wave model for obtaining realistic estimates of wave parameters in coastal areas, lakes and estuaries from given wind, bottom and current conditions. However, SWAN can be used on any scale relevant for wind-generated surface gravity waves. The model is based on the wave action balance equation with sources and sinks.

An important question addressed is how to choose various grids in SWAN (resolution, orientation, etc.) including nesting. In general, we consider two types of grids: structured and unstructured. Structured grids may be recti-linear and uniform or curvi-linear. They always consist of quadrilaterals in which the number of grid cells that meet each other in an internal grid point is 4. In unstructured grids, this number can be arbitrarily (usually between 4 and 10). For this reason, the level of flexibility with respect to the grid point distribution of unstructured grids is far more optimal compared to structured grids. Unstructured grids may contain triangles or a combination of triangles and quadrilaterals (so-called hybrid grids). In the current version of SWAN, however, only triangular meshes can be employed.

Often, the characteristic spatial scales of the wind waves propagating from deep to shallow waters are very diverse and would required to allow local refinement of the mesh near the coast without incurring overhead associated with grid adaptation at some distance offshore. Traditionally, this can be achieved by employing a nesting approach.

The idea of nesting is to first compute the waves on a coarse grid for a larger region and then on a finer grid for a smaller region. The computation on the fine grid uses boundary conditions that are generated by the computation on the coarse grid. Nesting can be repeated on ever decreasing scales using the same type of coordinates for the coarse computations and the nested computations (Cartesian or spherical). Note that curvi-linear grids

can be used for nested computations but the boundaries should always be rectangular.

The use of unstructured grids in SWAN offers a good alternative to nested models not only because of the ease of optimal adaption of mesh resolution but also the modest effort needed to generate grids about complicated geometries, e.g. islands and irregular shorelines. This type of flexible meshes is particularly useful in coastal regions where the water depth varies greatly. As a result, this variable spatial meshing gives the highest resolution where it is most needed. The use of unstructured grids facilitates to resolve the model area with a relative high accuracy but with a much fewer grid points than with regular grids.

It must be pointed out that the application of SWAN on ocean scales is not recommended from an efficiency point of view. The WAM model and the WAVEWATCH III model, which have been designed specifically for ocean applications, are probably one order of magnitude more efficient than SWAN. SWAN can be run on large scales (much larger than coastal scales) but this option is mainly intended for the transition from ocean scales to coastal scales (transitions where nonstationarity is an issue and spherical coordinates are convenient for nesting).

A general suggestion is: start simple. SWAN helps in this with default options. Furthermore, suggestions are given that should help the user to choose among the many options conditions and in which mode to run SWAN (first-, second- or third-generation mode, stationary or nonstationary and 1D or 2D).

2.2 Limitations

The DIA approximation for the **quadruplet wave-wave interactions** depends on the width of the directional distribution of the wave spectrum. It seems to work reasonably in many cases but it is a poor approximation for long-crested waves (narrow directional distribution). It also depends on the frequency resolution. It seems to work reasonably in many cases but it is a poor approximation for frequency resolutions with ratios very different from 10% (see command `CGRID`). This is a fundamental problem that SWAN shares with other third-generation wave models such as WAM and WAVEWATCH III.

The LTA approximation for the **triad wave-wave interactions** depends on the width of the directional distribution of the wave spectrum. The present tuning in SWAN (the default settings, see command `TRIAD`) seems to work reasonably in many cases but it has been obtained from observations in a narrow wave flume (long-crested waves).

As an option SWAN computes **wave-induced set-up**. In 1D cases the computations are based on exact equations. In 2D cases, the computations are based on approximate equations. This approximation in SWAN can only be applied to open coast (unlimited supply of water from outside the domain, e.g. nearshore coasts and estuaries) in contrast to closed basin, e.g. lakes, where this option should not be used. The effects of wave-induced cur-

rents are always ignored.

SWAN does not calculate **wave-induced currents**. If relevant, such currents should be provided as input to SWAN, e.g. from a circulation model which can be driven by waves from SWAN in an iteration procedure.

In areas where variations in wave height are large within a horizontal scale of a few wave lengths, **diffraction** should be used. However, the computation of diffraction in arbitrary geophysical conditions is rather complicated and requires considerable computing effort. To avoid this, a phase-decoupled approach is employed in SWAN so that same qualitative behaviour of spatial redistribution and changes in wave direction is obtained. This approach, however, does not properly handle diffraction in harbours or in front of reflecting obstacles.

SWAN can be used on **any scale** relevant for wind generated surface gravity waves. However, SWAN is specifically designed for coastal applications that should actually not require such flexibility in scale. The reasons for providing SWAN with such flexibility are:

- to allow SWAN to be used from laboratory conditions to shelf seas and
- to nest SWAN in the WAM model or the WAVEWATCH III model which are formulated in terms of spherical coordinates.

Nevertheless, these facilities are not meant to support the use of SWAN on oceanic scales because SWAN is less efficient on oceanic scales than WAVEWATCH III and probably also less efficient than WAM.

2.3 Internal scenarios, limiters, shortcomings and coding bugs

Sometimes the user input to SWAN is such that SWAN produces unreliable and sometimes even unrealistic results. This may be the case if the bathymetry or the wave field is not well resolved. Be aware here that the grid on which the computations are performed interpolates from the grids on which the input is provided; different resolutions for these grids (which are allowed) can therefore create unexpected interpolation patterns on the computational grid. In such cases SWAN may invoke some **internal scenarios** or **limiters** instead of terminating the computations. The reasons for this model policy is that

- SWAN needs to be robust, and
- the problem may be only very local, or
- the problem needs to be fully computed before it can be diagnosed.

Examples are:

- The user can request that refraction over one spatial grid step is limited to about 90^0(see command `NUMERIC`). This may be relevant when the depth varies considerably over one spatial grid step (e.g. at the edge of oceans or near oceanic islands with only one or two grid steps to go from oceanic depths to a shallow coast). This implies inaccurate refraction computations in such grid steps. This may be acceptable when refraction has only local effects that can be ignored but, depending on the topography, the inaccurately computed effects may radiate far into the computational area.

- SWAN cannot handle wave propagation on super-critical current flow. If such flow is encountered during SWAN computations, the current is locally reduced to sub-critical flow.

- If the water depth is less than some user-provided limit, the depth is set at that limit (default is 0.05 m, see command `SET`).

- The user-imposed wave boundary conditions may not be reproduced by SWAN employing structured grids, as SWAN replaces the *imposed* waves at the boundaries that propagate into the computational area with the *computed* waves that move out of the computational area at the boundaries.

- SWAN may have convergence problems. There are three iteration processes in SWAN:

  1. an iteration process for the spatial propagation of the waves,
  2. if ambient currents are present, an iteration process for spectral propagation (current-induced refraction and frequency shift) and
  3. if wave-induced set-up is requested by the user, an iteration process for solving the set-up equation.

  ad 1 For spatial propagation the change of the wave field over one iteration is limited to some realistic value (usually several iterations for stationary conditions or one iteration or upgrade per time step for nonstationary conditions; see command `NUMERIC`). This is a common problem for all third-generation wave models (such as WAM, WAVEWATCH III and also SWAN). It does not seem to affect the result seriously in many cases but sometimes SWAN fails to converge properly.

  For curvi-linear grids, convergence problems may occur locally where in some points in the grid, the directions separating the 4 sweeping quadrants coincide with the given spectral directions.

  ad 2 For spectral propagation (but only current-induced refraction and frequency shift) SWAN may also not converge.

  ad 3 For the wave-induced set-up SWAN may also not converge.

Information on the actual convergence of a particular SWAN run is provided in the `PRINT` file (see SWAN Implementation Manual).

Some other problems which the SWAN user may encounter are due to more **fundamental shortcomings** of SWAN (which may or may not be typical for third-generation wave models) and unintentional **coding bugs**.

Because of the issues described above, the results may look realistic, but they may (locally) not be accurate. Any change in these scenarios, limiters or shortcomings, in particular newly discovered coding bugs and their fixes, are published on the SWAN web site and implemented in new releases of SWAN.

2.4 Relation to WAM, WAVEWATCH III and others

The basic scientific philosophy of SWAN is identical to that of WAM (Cycle 3 and 4). SWAN is a third-generation wave model and it uses the same formulations for the source terms (although SWAN uses the adapted code for the DIA technique). On the other hand, SWAN contains some additional formulations primarily for shallow water. Moreover, the numerical techniques are very different. WAVEWATCH III not only uses different numerical techniques but also different formulations for the wind input and the whitecapping.

This close similarity can be exploited in the sense that

- scientific findings with one model can be shared with the others and
- SWAN can be readily nested in WAM and WAVEWATCH III (the formulations of WAVEWATCH III have not yet been implemented in SWAN).

When SWAN is nested in WAM or WAVEWATCH III, it must be noted that the boundary conditions for SWAN provided by WAM or WAVEWATCH III may not be model consistent even if the same physics are used. The potential reasons are manifold such as differences in numerical techniques employed and implementation for the geographic area (spatial and spectral resolutions, coefficients, etc.). Generally, the deep water boundary of the SWAN nest must be located in WAM or WAVEWATCH III where shallow water effects do not dominate (to avoid too large discontinuities between the two models). Also, the spatial and spectral resolutions should not differ more than a factor two or three. If a finer resolution is required, a second or third nesting may be needed.

2.5 Units and coordinate systems

SWAN expects all quantities that are given by the user to be expressed in S.I. units: m, kg, s and composites of these with accepted compounds, such as Newton (N) and Watt (W). Consequently, the wave height and water depth are in m, wave period in s, etc. For wind and wave direction both the Cartesian and *a* nautical convention can be used (see

below). Directions and spherical coordinates are in degrees (^{0}) and not in radians.

For the output of wave energy the user can choose between variance (m^2) or energy (**spatial**) density (Joule/m^2, i.e. energy per unit sea surface) and the equivalents in case of energy transport (m^3/s or W/m, i.e. energy transport per unit length) and spectral energy density (m^2/Hz/Degr or Js/m^2/rad, i.e. energy per unit frequency and direction per unit sea surface area). The wave−induced stress components (obtained as spatial derivatives of wave-induced radiation stress) are always expressed in N/m^2 even if the wave energy is in terms of variance. Note that the energy density is also in Joule/m^2 in the case of spherical coordinates.

SWAN operates either in a Cartesian coordinate system or in a spherical coordinate system, i.e. in a flat plane or on a spherical Earth. In the Cartesian system, all geographic locations and orientations in SWAN, e.g. for the bottom grid or for output points, are defined in one common Cartesian coordinate system with origin (0,0) by definition. This geographic origin may be chosen totally arbitrarily by the user. In the spherical system, all geographic locations and orientations in SWAN, e.g. for the bottom grid or for output points, are defined in geographic longitude and latitude. Both coordinate systems are designated in this manual as the problem coordinate system.

In the input and output of SWAN the direction of wind and waves are defined according to either

- the Cartesian convention, i.e. the direction to where the vector points, measured counterclockwise from the positive $x-$axis of this system (in degrees) or
- a nautical convention (there are more such conventions), i.e. the direction where the wind or the waves come from, measured clockwise from geographic North.

All other directions, such as orientation of grids, are according to the Cartesian convention!

For regular grids, i.e. uniform and rectangular, Figure 4.1 (in Section 4.5) shows how the locations of the various grids are determined with respect to the problem coordinates. All grid points of curvi-linear and unstructured grids are relative to the problem coordinate system.

2.6 Choice of grids, time windows and boundary / initial / first guess conditions

2.6.1 Introduction

Several types of grids and time window(s) need to be defined: (a) spectral grid, (b) spatial (geographic) grids and time window(s) in case of nonstationary computations.

The **spectral** grid that need to be defined by the user is a computational spectral grid on

which SWAN performs the computations.

SWAN has the option to make computations that can be nested in (coarse) SWAN, WAM or WAVEWATCH III. In such cases, the spectral grid need not be equal to the spectral grid in the coarse SWAN, WAM or WAVEWATCH III run.

The **spatial** grids that need to be defined by the user are (if required):

- a computational spatial grid on which SWAN performs the computations,
- one (or more) spatial input grid(s) for the bottom, current field, water level, bottom friction and wind (each input grid may differ from the others) and
- one (or more) spatial output grid(s) on which the user requires output of SWAN.

The wind and bottom friction do not require a grid if they are uniform over the area of interest.

For one-dimensional situations, i.e. $\partial/\partial y \equiv 0$, SWAN can be run in 1D mode.

If a uniform, rectangular computational spatial grid is chosen in SWAN, then all input and output grids must be uniform and rectangular too, but they may all be different from each other.

If a curvi-linear computational spatial grid is chosen in SWAN, then each input grid should be either uniform, rectangular or identical to the used curvi-linear grid or staggered with respect to the curvi-linear computational grid.

If an unstructured computational spatial grid is chosen in SWAN, then each input grid should be either uniform, rectangular or identical to the used unstructured grid.

SWAN has the option to make computations that are nested in (coarse) SWAN, WAM or WAVEWATCH III. In such runs, SWAN interpolates the spatial boundary of the SWAN, WAM or WAVEWATCH III grid to the (user provided) grid of SWAN (that needs to (nearly) coincide along the grid lines of WAM or WAVEWATCH III or the output nest grid boundaries of SWAN). Since, the computational grids of WAM and WAVEWATCH III are in spherical coordinates, it is recommended to use spherical coordinates in a nested SWAN when nesting in WAM or WAVEWATCH III.

SWAN using an unstructured mesh may be nested in SWAN employing a regular grid and vice versa. However, SWAN using an unstructured grid cannot be nested in WAM or WAVEWATCH III.

Nesting from a 2D model to a 1D model is possible although is should not be done by using the commands `NGRID` and `NEST`. Instead, define the boundary point of the 1D model as an output point (using command `POINTS`) and write the spectra for that point using the command `SPECout`. In the 1D model, this spectra is used as boundary condition using

the BOUNDSPEC command.

Similarly, the wind fields may be available in different **time** windows than the current and water level fields and the computations may need to be carried out at other times than these input fields. For these reasons SWAN operates with different time windows with different time steps (each may have a different start and end time and time step):

- one computational time window in which SWAN performs the computations,
- one (or more) input time window(s) in which the bottom, current field, water level, bottom friction and wind field (if present) are given by the user (each input window may differ form the others) and
- one (or more) output time window(s) in which the user requires output of SWAN.

In case of nesting, SWAN searches the boundary conditions in the relevant output file of the previous SWAN, WAM or WAVEWATCH III runs to take the boundary conditions at the start time of the nested run. It will not take the initial condition (i.e. over the entire computational grid) for the nested run from the previous SWAN, WAM or WAVEWATCH III run.

During the computation SWAN obtains bottom, current, water level, wind and bottom friction information by tri-linear **interpolation** from the given input grid(s) and time window(s). The output is in turn obtained in SWAN by bi-linear interpolation in space from the computational grid; there is no interpolation in time, the output time is shifted to the nearest computational time level. Interpolation errors can be reduced by taking the grids and windows as much as equal to one another as possible (preferably identical). It is recommended to choose output times such that they coincide with computational time levels.

2.6.2 Input grid(s) and time window(s)

The bathymetry, current, water level, bottom friction and wind (if spatially variable) need to be provided to SWAN on so-called input grids. It is best to make an input grid so large that it completely covers the computational grid.

In the region outside the input grid SWAN assumes that the bottom level, the water level and bottom friction are identical to those at the nearest boundary of the input grid (lateral shift of that boundary). In the regions not covered by this lateral shift (i.e. in the outside quadrants of the corners of the input grid), a constant field equal to the value at the nearest corner point of the input grid is taken. For the current and wind velocity, SWAN takes 0 m/s for points outside the input grid.

In SWAN, the bathymetry, current, water level, wind and bottom friction may be time varying. In that case they need to be provided to SWAN in so-called input time windows

(they need not be identical with the computational, output or other input windows). It is best to make an input window larger than the computational time window. SWAN assumes zero values at times before the earliest begin time of the input parameters (which may be the begin time of any input parameter such as wind). SWAN assumes constant values (the last values) at times after the end time of each input parameter. The input windows should start early enough so that the initial state of SWAN has propagated through the computational area before reliable output of SWAN is expected.

One should choose the spatial resolution for the input grids such that relevant spatial details in the bathymetry, currents, bottom friction and wind are properly resolved. Special care is required in cases with sharp and shallow ridges (sand bars, shoals) in the sea bottom and extremely steep bottom slopes. Very inaccurate bathymetry can result in very inaccurate refraction computations the results of which can propagate into areas where refraction as such is not significant (the results may appear to be unstable). For instance, waves skirting an island that is poorly resolved may propagate beyond the island with entirely wrong directions. In such a case it may even be better to deactivate the refraction computations (if refraction is irrelevant for the problem at hand e.g. because the refracted waves will run into the coast anyway and one is not interested in that part of the coast). In such cases the ridges are vitally important to obtain good SWAN results (at sea the waves are 'clipped' by depth-induced breaking over the ridges which must therefore represented in SWAN computation). This requires not only that these ridges should be well represented on the input grid but also after interpolation on the computational grid. This can be achieved by choosing the grid lines of the input grid along the ridges (even if this may require some slight "shifting" of the ridges) and choosing the computational grid to be identical to the input grid (otherwise the ridge may be "lost" in the interpolation from the bottom input grid to the computational grid).

Finally, one should use a time step that is small enough that time variations in the bathymetry, current, water level, wind and bottom friction are well resolved.

2.6.3 Computational grids and boundary / initial / first guess conditions

The **computational spatial grid** must be defined by the user. The orientation (direction) can be chosen arbitrarily.

The boundaries of the computational spatial grid in SWAN are either land or water. In the case of land there is no problem: the land does not generate waves and in SWAN it absorbs all incoming wave energy. But in the case of a water boundary there may be a problem. Often no wave conditions are known along such a boundary and SWAN then assumes that no waves enter the area and that waves can leave the area freely. These assumptions obviously contain errors which propagate into the model. These boundaries must therefore be chosen sufficiently far away from the area where reliable computations are needed so

that they do not affect the computational results there. This is best established by varying the location of these boundaries and inspect the effect on the results. Sometimes the waves at these boundaries can be estimated with a certain degree of reliability. This is the case if (a) results of another model run are available (nested computations) or, (b) observations are available. If model results are available along the boundaries of the computational spatial grid, they are usually from a coarser resolution than the computational spatial grid under consideration. This implies that this coarseness of the boundary propagates into the computational grid. The problem is therefore essentially the same as if no waves are assumed along the boundary except that now the error may be more acceptable (or the boundaries are permitted to be closer to the area of interest). If observations are available, they can be used as input at the boundaries. However, this usually covers only part of the boundaries so that the rest of the boundaries suffer from the same error as above.

A special case occurs near the coast. Here it is often possible to identify an up-wave boundary (with proper wave information) and two lateral boundaries (with no or partial wave information). The affected areas with errors are typically regions with the apex at the corners of the water boundary with wave information, spreading towards shore at an angle of 30° to 45° for wind sea conditions to either side of the imposed mean wave direction (less for swell conditions; the angle is essentially the one-sided width of the directional distribution of wave energy). For propagation of short crested waves (wind sea condtions) an example is given in Figure 2.1. For this reason the lateral boundaries should be sufficiently far away from the area of interest to avoid the propagation of this error into this area. Such problems do not occur if the lateral boundaries contain proper wave information over their entire length e.g. obtained from a previous SWAN computation or if the lateral boundaries are coast.

When output is requested along a boundary of the computational grid, it may occur that this output differs from the boundary conditions that are imposed by the user. The reason is that SWAN accepts only the user-imposed incoming wave components and that it replaces the user-imposed outgoing wave components with computed outgoing components (propagating to the boundary from the interior region). **This is only the case for structured grids (both regular and curvi-linear ones)**. The user is informed by means of a warning in the output when the computed significant wave height differs more than 10%, say (10% is default), from the user-imposed significant wave height (command `BOUND`...). The actual value of this difference can be set by the user (see the `SET` command). Note that this warning will not apply in the case of unstructured grids.

If the computational grid extends outside the input grid, the reader is referred to Section 2.6 to find the assumptions of SWAN on depth, current, water level, wind, bottom friction in the non-overlapping area.

The spatial resolution of the computational grid should be sufficient to resolve relevant details of the wave field. Usually a good choice is to take the resolution of the computational grid approximately equal to that of the bottom or current grid. If necessary, an

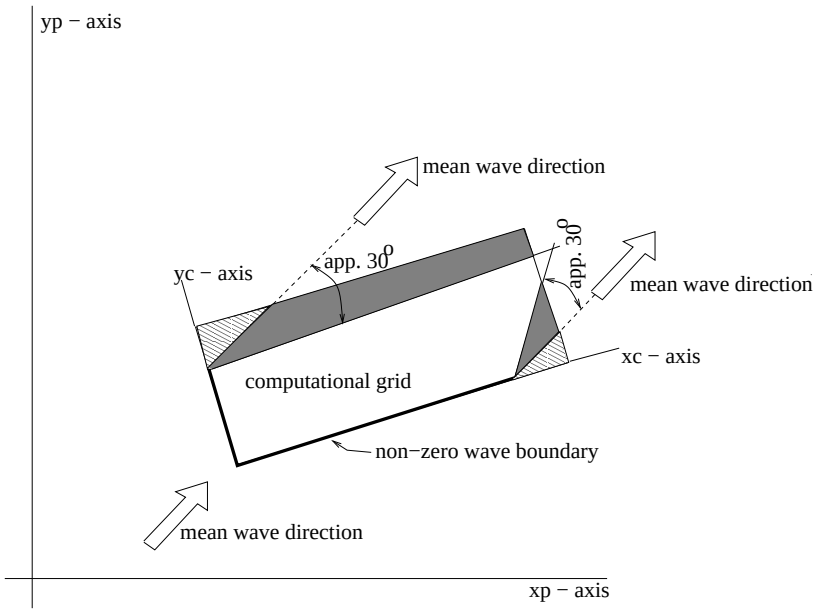


Figure 2.1: Disturbed regions in the computational grid due to erroneous boundary conditions are indicated with shaded areas.

unstructured grid may be used.

SWAN may not use the entire user-provided computational grid if the user defines exception values on the depth grid (see command `INPGRID BOTTOM`) or on the curvi-linear computational grid (see command `CGRID`). It must be noted that for parallel runs using MPI the user must indicate an exception value when reading the bottom levels (by means of command `INPGRID BOTTOM EXCEPTION`), in order to obtain good load balancing.

A computational grid point is either

- wet, i.e. the grid point is included in the computations since it represents water; this may vary with time-dependent or wave-induced water levels or
- dry, i.e. the grid point is excluded from the computations since it represents land which may vary with time-dependent or wave-induced water levels or
- exceptional, i.e. the grid point is permanently excluded from the computations since it is so defined by the user.

If exceptional grid points occur in the computational grid, then SWAN filters the entire computational grid as follows:

- each grid line between two adjacent wet computational grid points (a wet link) without an adjacent, parallel wet link, is removed,

- each wet computational grid point that is linked to only one other wet computational grid point, is removed and
- each wet computational grid point that has no wet links is removed.

The effect of this filter is that if exception values are used for the depth grid or the curvi-linear computational grid, one-dimensional water features are ignored in the SWAN computations (results at these locations with a width of about one grid step may be unrealistic). If no exception values are used, the above described filter will not be applied. As a consequence, one-dimensional features may appear or disappear due to changing water levels (flooding may create them, drying may reduce two-dimensional features to one-dimensional features).

The **computational time window** must be defined by the user in case of nonstationary runs. The computational window in time must start at a time that is early enough that the initial state of SWAN has propagated through the computational area before reliable output of SWAN is expected. Before this time the output may not be reliable since usually the initial state is not known and only either no waves or some very young sea state is assumed for the initial state. This is very often erroneous and this erroneous initial state is propagated into the computational area.

The computational time step must be given by the user in case of nonstationary runs. Since, SWAN is based on implicit numerical schemes, it is not limited by the Courant stability criterion (which couples time and space steps). In this sense, the time step in SWAN is not restricted. However, the accuracy of the results of SWAN are obviously affected by the time step. Generally, the time step in SWAN should be small enough to resolve the time variations of computed wave field itself. Usually, it is enough to consider the time variations of the driving fields (wind, currents, water depth, wave boundary conditions). But be careful: relatively(!) small time variations in depth (e.g. by tide) can result in relatively(!) large variations in the wave field.

As default, the first guess conditions of a stationary run of SWAN are determined with the 2^{nd} generation mode of SWAN. The initial condition of a nonstationary run of SWAN is by default a JONSWAP spectrum with a $\cos^2(\theta)$ directional distribution centred around the local wind direction.

A quasi-stationary approach can be employed with stationary SWAN computations in a time-varying sequence of stationary conditions.

The **computational spectral grid** needs to be provided by the user. In frequency space, it is simply defined by a minimum and a maximum frequency and the frequency resolution which is proportional to the frequency itself (i.e. logarithmic, e.g., $\Delta f = 0.1\, f$). The frequency domain may be specified as follows (see command `CGRID`):

- The lowest frequency, the highest frequency and the number of frequencies can be chosen.

- Only the lowest frequency and the number of frequencies can be chosen. The highest frequency will be computed by SWAN such that $\Delta f = 0.1 f$. This resolution is required by the DIA method for the approximation of nonlinear 4-wave interactions (the so-called quadruplets).

- Only the highest frequency and the number of frequencies can be chosen. The lowest frequency will be computed by SWAN such that $\Delta f = 0.1 f$. This resolution is required by the DIA method for the approximation of nonlinear 4-wave interactions.

- Only the lowest frequency and the highest frequency can be chosen. The number of frequencies will be computed by SWAN such that $\Delta f = 0.1 f$. This resolution is required by the DIA method for the approximation of nonlinear 4-wave interactions.

The value of lowest frequency must be somewhat smaller than 0.7 times the value of the lowest peak frequency expected. The value of highest frequency must be at least 2.5 to 3 times the highest peak frequency expected. For the XNL approach, however, this should be 6 times the highest peak frequency. Usually, it is chosen less than or equal to 1 Hz.

SWAN has the option to make computations that can be nested in WAM or WAVEWATCH III. In such runs SWAN interpolates the spectral grid of WAM or WAVEWATCH III to the (user provided) spectral grid of SWAN. The WAM Cycle 4 source term in SWAN has been retuned for a highest prognostic frequency (that is explicitly computed by SWAN) of 1 Hz. It is therefore recommended that for cases where wind generation is important and WAM Cycle 4 formulations are chosen, the highest prognostic frequency is about 1 Hz.

In directional space, the directional range is the full 360° unless the user specifies a limited directional range. This may be convenient (less computer time and/or memory space), for example, when waves travel towards a coast within a limited sector of 180°. The directional resolution is determined by the number of discrete directions that is provided by the user. For wind seas with a directional spreading of typically 30° on either side of the mean wave direction, a resolution of 10° seems enough whereas for swell with a directional spreading of less than 10°, a resolution of 2° or less may be required. If the user is confident that no energy will occur outside a certain directional sector (or is willing to ignore this amount of energy), then the computations by SWAN can be limited to the directional sector that does contain energy. This may often be the case of waves propagating to shore within a sector of 180° around some mean wave direction.

It is recommended to use the following discretization in SWAN for applications in coastal areas:

direction resolution for wind sea	$\Delta\theta = 15^o - 10^o$
direction resolution for swell	$\Delta\theta = 5^o - 2^o$
frequency range	$0.04 \leq f \leq 1.00$ Hz
spatial resolution	$\Delta x, \Delta y = 50 - 1000$ m

The numerical schemes in the SWAN model require a minimum number of discrete grid points in each spatial directions of 2. The minimum number of directional bins is 3 per directional quadrant and the minimum number of frequencies should be 4.

2.6.4 Output grids

SWAN can provide output on uniform, recti-linear spatial grids that are independent from the input grids and from the computational grid. In the computation with a curvi-linear computational grid, curvi-linear output grids are available in SWAN. This also holds for triangular meshes. An output grid has to be specified by the user with an arbitrary resolution, but it is of course wise to choose a resolution that is fine enough to show relevant spatial details. It must be pointed out that the information on an output grid is obtained from the computational grid by bi-linear interpolation (output always at computational time level). This implies that some inaccuracies are introduced by this interpolation. It also implies that bottom or current information on an output plot has been obtained by interpolating twice: once from the input grid to the computational grid and once from the computational grid to the output grid. If the input-, computational- and output grids are identical, then no interpolation errors occur.

In the regions where the output grid does not cover the computational grid, SWAN assumes output values equal to the corresponding exception value. For example, the default exception value for the significant wave height is -9. The exception values of output quantities can be changed by means of the `QUANTITY` command.

In nonstationary computations, output can be requested at regular intervals starting at a given time always at computational times.

2.7 Activation of physical processes

SWAN contains a number of physical processes (see Scientific/Technical documentation) that add or withdraw wave energy to or from the wave field. The processes included are: wind input, whitecapping, bottom friction, depth-induced wave breaking, obstacle transmission, nonlinear wave-wave interactions (quadruplets and triads) and wave-induced set-up. SWAN can run in several modes, indicating the level of parameterization. SWAN can operate in first-, second- and third-generation mode. The first- and second-generation modes are essentially those of Holthuijsen and De Boer (1988); first-generation with a constant Phillips "constant" of 0.0081 and second-generation with a variable Phillips "constant". An overview of the options is given in Table below. The processes are activated as follows:

- *Wind input* is activated by commands `GEN1`, `GEN2` or `GEN3`[1].

[1] active by default, can be deactivated with command `OFF`.

Table 2.1: Overview of physical processes and generation mode in SWAN.

process	**authors**	**generation mode**		
		1st	**2nd**	**3rd**
Linear wind growth	Cavaleri and Malanotte-Rizzoli (1981) (modified)	×	×	
	Cavaleri and Malanotte-Rizzoli (1981)			×
Exponential wind growth	Snyder *et al.* (1981) (modified)	×	×	
	Snyder *et al.* (1981)			×
	Janssen (1989, 1991)			×
Whitecapping	Holthuijsen and De Boer (1988)	×	×	
	Komen *et al.* (1984)			×
	Janssen (1991)			×
Quadruplets	Hasselmann *et al.* (1985)			×
Triads	Eldeberky (1996)	×	×	×
Depth-induced breaking	Battjes and Janssen (1978)	×	×	×
Bottom friction	JONSWAP (1973)	×	×	×
	Collins (1972)	×	×	×
	Madsen *et al.* (1988)	×	×	×
Obstacle transmission	Seelig (1979), d'Angremond (1996)	×	×	×
Wave-induced set-up		×	×	×

- *Whitecapping* is activated by commands `GEN1`, `GEN2` or `GEN3`[2].
- *Quadruplets* is activated by command `GEN3`[3].
- *Triads* is activated by command `TRIAD`.
- *Bottom friction* is activated by command `FRICTION`.
- *Depth-induced breaking* is activated by command `BREAKING`[4].
- *Obstacle transmission* is activated by command `OBSTACLE`.
- *Wave-induced set-up* is activated by command `SETUP`.

For the preliminary SWAN runs, it is strongly advised to use the default values of the model coefficients. First, it should be determined whether or not a certain physical process is relevant to the result. If this cannot be decided by means of a simple hand computation,

[2]active by default, can be deactivated with command `OFF`.
[3]active by default, can be deactivated with command `OFF`.
[4]active by default, can be deactivated with command `OFF`.

one can perform a SWAN computation without and with the physical process included in the computations, in the latter case using the standard values chosen in SWAN.

After it has been established that a certain physical process is important, it may be worthwhile to modify coefficients. In the case of wind input one may at first try to vary the wind velocity. Concerning the bottom friction, the best coefficients to vary are the friction coefficient. Switching off the depth-induced breaking term is usually unwise, since this may lead to unacceptably high wave heights near beaches (the computed wave heights 'explode' due to shoaling effects).

2.8 Time and date notation

SWAN can run for dates (i.e. nonstationary mode)

- between the years 0 and 9999, if ISO-notation is used in the input (recommended) or
- between the years 1931 and 2030 if two-digit code for years is used (formats 2-6 in every command that contains moments in time).

Be careful when nesting SWAN in WAM, since WAM does not use ISO-notation.

Chapter 3

Input and output files

3.1 General

SWAN is one single computer program. The names of the files provided by the user should comply with the rules of file identification of the computer system on which SWAN is run. In addition: SWAN does not permit file names longer than 36 characters. Moreover, the maximum length of the lines in the input files for SWAN is 120 positions.

The user should provide SWAN with a number of files (input files) with the following information:

- a file containing the instructions of the user to SWAN (the command file),
- file(s) containing: grid, bottom, current, friction, and wind (if relevant) and
- file(s) containing the wave field at the model boundaries (if relevant).

3.2 Input / output facilities

To assist in making the command file, an edit file is available to the user (see Appendix C). In its original form this file consists only of comments; all lines begin with exclamation mark. In the file, all commands as given in this User Manual (Chapter 4) are reproduced as mnemonics for making the final command file. Hence, the user does not need to consult the User Manual every time to check the correct spelling of keywords, order of data, etc. The user is advised to first copy the edit file (the copy file should have a different name) and then start typing commands between the comment lines of the edit file.

SWAN is fairly flexible with respect to output processing. Output is available for many different wave parameters and wave related parameters (e.g., wave-induced stresses and bottom orbital motion). However, the general rule is that output is produced by SWAN only at the user's request. The instructions of the user to control output are separated into three categories:

- Definitions of the geographic location(s) of the output. The output locations may be either on a geographic grid, or along user specified lines (e.g., a given depth contour line) or at individual output locations.
- Times for which the output is requested (only in nonstationary runs).
- Type of output quantities (wave parameters, currents or related quantities).

3.3 Print file and error messages

SWAN always creates a print file. Usually the name of this file is identical to the name of the command file of the computations with the extension (.SWN) replaced with (.PRT). Otherwise, it depends on the batch file that is used by the user. Consult the Implementation Manual for more information.

The print file contains an echo of the command file and error messages. These messages are usually self-explanatory (if not, users may address the SWAN forum-page on the SWAN homepage). The print file also contains computational results if the user so requests (with command `BLOCK` or `TABLE`).

Chapter 4

Description of commands

4.1 List of available commands

The following commands are available to users of SWAN (to look for the commands quickly, see table of contents and index).

Start-up commands

(a) Start-up commands:

`PROJECT`	title of the problem to be computed
`SET`	sets values of certain general parameters
`MODE`	requests a stationary / nonstationary or 1D-mode / 2D-mode of SWAN
`COORD`	to choose between Cartesian and spherical coordinates

Commands for model description

(b) Commands for computational grid:

`CGRID`	defines dimensions of computational grid
`READGRID`	reads a curvi-linear or unstructured computational grid

(c) Commands for input fields:

`INPGRID`	defines dimensions of bottom, water level, current and friction grids
`READINP`	reads input fields
`WIND`	activates constant wind option

(d) Commands for boundary and initial conditions:

`BOUND`	defines the shape of the spectra at the boundary of geographic grid
`BOUNDSPEC`	defines (parametric) spectra at the boundary of geographic grid
`BOUNDNEST1`	defines boundary conditions obtained from (coarse) SWAN run
`BOUNDNEST2`	defines boundary conditions obtained from WAM run
`BOUNDNEST3`	defines boundary conditions obtained from WAVEWATCH III run
`INITIAL`	specifies an initial wave field

(e) Commands for physics:

`GEN1`	SWAN runs in first generation mode
`GEN2`	SWAN runs in second generation mode
`GEN3`	SWAN runs in third generation mode
`WCAPPING`	activates cumulative steepness method for whitecapping
`QUAD`	controls the computation of quadruplets
`BREAKING`	activates dissipation by depth-induced wave breaking
`FRICTION`	activates dissipation by bottom friction
`TRIAD`	activates three wave-wave interactions
`LIMITER`	de-actives quadruplets if a certain Ursell number exceeds
`OBSTACLE`	defines characteristics of sub-grid obstacles
`SETUP`	activates the computation of the wave-induced set-up
`DIFFRAC`	activates diffraction
`OFF`	de-activates certain physical processes

(f) Commands for numerics:

`PROP`	to choose the numerical propagation scheme
`NUMERIC`	sets some of the numerical properties of SWAN

Output commands

(g) Commands for output locations:

`FRAME`	defines an output frame (a regular grid)
`GROUP`	defines an output group (for regular and curvi-linear grids)
`CURVE`	defines an output curve
`RAY`	defines a set of straight output lines (rays)
`ISOLINE`	defines a depth- or bottom contour (for output along that contour)
`POINTS`	defines a set of individual output points
`NGRID`	defines a nested grid

(h) Commands to write or plot output quantities:

`QUANTITY`	defines properties of output quantities
`OUTPUT`	influence format of block, table and/or spectral output
`BLOCK`	requests a block output (geographic distribution)
`TABLE`	requests a table output (set of locations)
`SPECOUT`	requests a spectral output
`NESTOUT`	requests a spectral output for subsequent nested computations

(i) Commands to write or plot intermediate results:

`TEST`	requests an output of intermediate results for testing purposes

Lock-up commands

(j) Commands to lock-up the input file:

`COMPUTE`	starts a computation
`HOTFILE`	stores results for subsequent SWAN run
`STOP`	end of user's input

4.2 Sequence of commands

SWAN executes the above command blocks (a,...,j) in the above sequence except (f), (i) and (j). The commands of the blocks (f) and (i) may appear anywhere before block (j), except that `TEST POINTS` must come after `READINP BOTTOM`. The commands of block (j) may appear anywhere in the command file (all commands after `COMPUTE` are ignored by SWAN, except `HOTFILE` and `STOP`). A sequence of commands of block (g) is permitted (all commands will be executed without overriding). Also a sequence of commands of block (h) is permitted (all commands will be executed without overriding).

Within the blocks the following sequence is to be used:

In block (a) : no prescribed sequence in block
In block (b) : `READGRID` after `CGRID`
In block (c) : `READINP` after `INPGRID` (repeat both in this sequence for each quantity)
In block (d) : `BOUND SHAPE` before `BOUNDSPEC`, otherwise no prescribed sequence in block
In block (e) : use only one `GEN` command; use command `OFF` only after a `GEN` command (note that `GEN3` is default)
In block (f) : no prescribed sequence in block

In block (g) : `ISOLINE` after `RAY` (`ISOLINE` and `RAY` can be repeated independently)
In block (h) : no prescribed sequence in block
In block (i) : no prescribed sequence in block
In block (j) : `HOTFILE` immediately after `COMPUTE`, `STOP` after `COMPUTE`

It must be noted that a repetition of a command may override an earlier occurrence of that command.

Many commands provide the user with the opportunity to assign values to coefficients of SWAN (e.g. the bottom friction coefficient). If the user does not use such option SWAN will use a default value.

Some commands cannot be used in 1D-mode (see individual command descriptions below).

4.3 Command syntax and input / output limitations

The command syntax is given in Appendix B.

Limitations:

- The maximum length of the input lines is 120 characters.
- The maximum length of the file names is 36 characters.
- The maximum length of the plot titles is 36 characters.
- The maximum number of file names is 99. This can be extended (edit the file `swaninit` to change highest unit number of 99 to a higher number).

4.4 Start-up

```
PROJect 'name' 'nr'

        'title1'

        'title2'

        'title3'
```

With this required command the user defines a number of strings to identify the SWAN run (project name e.g., an engineering project) in the print and plot file.

`'name'`	is the name of the project, at most 16 characters long. Default: blanks.
`'nr'`	is the run identification (to be provided as a character string; e.g. the run number) to distinguish this run among other runs for the same project; it is at most 4 characters long. It is the only required information in this command.
`'title1'`	is a string of at most 72 characters provided by the user to appear in the output of the program for the user's convenience. Default: blanks.
`'title2'`	same as `'title1'`.
`'title3'`	same as `'title1'`.

```
SET  [level] [nor] [depmin] [maxmes] [maxerr] [grav] [rho]     &

                          |    NAUTical  |
     [inrhog] [hsrerr] <                 > [pwtail] [froudmax] [printf] [prtest]
                          | -> CARTesian |
```

With this optional command the user assigns values to various general parameters.

`[level]`	increase in water level that is constant in space and time can be given with this option, `[level]` is the value of this increase (in m). For a variable water level reference is made to the commands `INPGRID` and `READINP`. Default: `[level]`=0.
`[nor]`	direction of North with respect to the $x-$axis (measured counterclockwise); default `[nor]`= 90°, i.e. $x-$axis of the problem coordinate system points East. When spherical coordinates are used (see command `COORD`) the value of `[nor]` may not be modified.
`[depmin]`	threshold depth (in m). In the computation any positive depth smaller than `[depmin]` is made equal to `[depmin]`. Default: `[depmin]` = 0.05.
`[maxmes]`	maximum number of error messages (not necessarily the number of errors!) during the computation at which the computation is terminated. During the computational process messages are written to the print file. Default: `[maxmes]` = 200.
`[maxerr]`	during pre-processing SWAN checks input data. Depending on the severity of the errors encountered during this pre-processing, SWAN does not start computations. The user can influence the error level above which SWAN will not start computations (at the level indicated the computations will continue). The error level `[maxerr]` is coded as follows:

1 : warnings,
2 : errors (possibly automatically repaired or repairable by SWAN),
3 : severe errors.
Default: `[maxerr]` = 1.

`[grav]` is the gravitational acceleration (in m/s^2).
Default: `[grav]` = 9.81.

`[rho]` is the water density ρ (in kg/m^3).
Default: `[rho]` = 1025.

`[inrhog]` to indicate whether the user requires output based on variance or based on true energy (see Section 2.5).
`[inrhog]` = 0 : output based on variance
`[inrhog]` = 1 : output based on true energy
Default: `[inrhog]` = 0.

`[hsrerr]` the relative difference between the user imposed significant wave height and the significant wave height computed by SWAN (anywhere along the boundary of structured grid) above which a warning will be given. This relative difference is the difference normalized with the user provided significant wave height. This warning will be given for each boundary grid point where the problem occurs (with its $x-$ and $y-$index number of the computational grid). The cause of the difference is explained in Section 2.6.3. To supress these warnings (in particular for nonstationary computations), set `[hsrerr]` at a very high value or use command `OFF BNDCHK`.
Default: `[hsrerr]` = 0.10.

`NAUTICAL` indicates that the Nautical convention for wind and wave direction (SWAN input and output) will be used instead of the default Cartesian convention.
For definition, see Section 2.5 or Appendix A.

`CARTESIAN` indicates that the Cartesian convention for wind and wave direction (SWAN input and output) will be used. For definition, see Section 2.5 or Appendix A.

`[pwtail]` power of high frequency tail; defines the shape of the spectral tail above the highest prognostic frequency `[fhigh]` (see command `CGRID`). The energy density is assumed to be proportional to frequency to the power `[pwtail]`.
Default values depend on formulations of physics:
command `GEN1` : `[pwtail]` = 5
command `GEN2` : `[pwtail]` = 5
command `GEN3 KOMEN` : `[pwtail]` = 4
command `GEN3 JANSEN` : `[pwtail]` = 5
If the user wishes to use another value, then this `SET` command should be located in the command file after the `GEN1`, `GEN 2` or `GEN3` command (these will override the `SET` command with respect to `[pwtail]`).

`[froudmax]` is the maximum Froude number ($U/\sqrt{gd}$ with U the current and d the water depth). The currents taken from a circulation model may mismatch with given water depth d in the sense that the Froude number becomes larger than 1. For this, the current velocities will be maximized by Froude number times $\sqrt{gd}$.

Default: `[froudmax]` = 0.8.

`[printf]` unit reference number of the `PRINT` file. As default, `[printf]` is equal to 4. If it is changed to 6 all print output will be written to the screen. This is useful if print output is lost due to abnormal end of the program, while information about the reason is expected to be in the `PRINT` file.

`[prtest]` unit reference number of the test output file. As default, `[prtest]` is equal to 4. If it is changed to 6 all test output will be written to the screen. This is useful if test print output is lost due to abnormal end of the program, while information about the reason is expected to be in the test output file.

```
      |-> STATionary    |     |-> TWODimensional |
MODE <                   >   <                    >
      |   NONSTationary |     |   ONEDimensional |
```

With this optional command the user indicates that the run will be either stationary or nonstationary and one-dimensional (1D-mode) or two-dimensional (2D-mode). Nonstationary means either (see command `COMPUTE`):

(a) one nonstationary computations or

(b) a sequence of stationary computations or

(c) a mix of (a) and (b).

The default option is `STATIONARY TWODIMENSIONAL`.

```
             | -> CARTesian                  |
COORDINATES <                  | -> CCM |    > REPeating
             |    SPHErical   <          >  |
                              |  QC    |
```

Command to choose between Cartesian and spherical coordinates (see Section 2.5).

A nested SWAN run must use the same coordinate system as the coarse grid SWAN run.

`CARTESIAN` all locations and distances are in m. Coordinates are given with respect to $x-$ and $y-$axes chosen by the user in the various commands.

`SPHERICAL` all coordinates of locations and geographical grid sizes are given in degrees; x is longitude with $x = 0$ being the Greenwich meridian and $x > 0$ is East of this meridian; y is latitude with $y > 0$ being the Northern hemisphere. Input and output grids have to be oriented with their $x-$axis to the East; mesh sizes

are in degrees. All other distances are in meters.

`CCM` defines the projection method in case of spherical coordinates. CCM means central conformal Mercator. The horizontal and vertical scales are uniform in terms of cm/degree over the area shown. In the centre of the scale is identical to that of the conventional Mercator projection (but only at that centre). The area in the projection centre is therefore exactly conformal.

`QC` the projection method is quasi-cartesian, i.e. the horizontal and vertical scales are equal to one another in terms of cm/degree.

`REPEATING` this option is only for academic cases. It means that wave energy leaving at one end of the domain (in computational $x-$direction) enter at the other side; it is as if the wave field repeats itself in $x-$direction with the length of the domain in $x-$direction.
This option cannot be used in combination with computation of set-up (see command `SETUP`). This option is available only with regular grids.

Note that spherical coordinates can also be used for relatively small areas, say 10 or 20 km horizontal dimension. This may be useful if one obtains the boundary conditions by nesting in an oceanic model which is naturally formulated in spherical coordinates.

Note that in case of spherical coordinates regular grids must always be oriented E-W, N-S, i.e. `[alpc]`=0, `[alpinp]`=0, `[alpfr]`=0 (see commands `CGRID`, `INPUT GRID` and `FRAME`, respectively).

4.5 Model description

4.5.1 Computational grid

```
       | -> REGular [xpc] [ypc] [alpc] [xlenc] [ylenc] [mxc] [myc] |
       |                                                           |
CGRID <     CURVilinear [mxc] [myc]  (EXCeption  [xexc]  [yexc])    >  &
       |                                                           |
       |    UNSTRUCtured                                           |

          | -> CIRcle                |
         <                            > [mdc] [flow] [fhigh] [msc]
          |    SECtor [dir1] [dir2] |
```

With this required command the user defines the geographic location, size, resolution and orientation of the computational grid in the problem coordinate system (see Section 2.6.3) in case of a uniform, recti-linear computational grid, a curvi-linear grid or unstructured

mesh. The origin of the regular grid and the direction of the positive $x-$axis of this grid can be chosen arbitrary by the user. Must be used for nested runs. Note that in a nested case, the geographic and spectral range (directional sector inclusive) and resolution may differ from the previous run (outside these ranges zero's are used).

`REGULAR` this option indicates that the computational grid is to be taken as uniform and rectangular.

`CURVILINEAR` this option indicates that the computational grid is to be taken as curvi-linear. The user must provide the coordinates of the grid points with command `READGRID COOR`.

`UNSTRUCTURE` this option indicates that the computational grid is to be taken as unstructured. The user must provide the coordinates of the vertices and the numbering of triangles with the associated connectivity table with vertices with command `READGRID UNSTRUC`.

`[xpc]` geographic location of the origin of the computational grid in the problem coordinate system ($x-$coordinate, in m). See command `COORD`.
Default: `[xpc]` = 0.0 (Cartesian coordinates).
In case of spherical coordinates there is no default, the user must give a value.

`[ypc]` geographic location of the origin of the computational grid in the problem coordinate system ($y-$coordinate, in m). See command `COORD`.
Default: `[ypc]` = 0.0 (Cartesian coordinates).
In case of spherical coordinates there is no default, the user must give a value.

`[alpc]` direction of the positive $x-$axis of the computational grid (in degrees, Cartesian convention). In 1D-mode, `[alpc]` should be equal to the direction `[alpinp]` (see command `INPGRID`).
Default: `[alpc]` = 0.0.

`[xlenc]` length of the computational grid in $x-$direction (in m). In case of spherical coordinates `[xlenc]` is in degrees.

`[ylenc]` length of the computational grid in $y-$direction (in m). In 1D-mode, `[ylenc]` should be 0. In case of spherical coordinates `[ylenc]` is in degrees.

`[mxc]` number of meshes in computational grid in $x-$direction for a uniform, recti-linear grid or $\xi-$direction for a curvi-linear grid (this number is one less than the number of grid points in this domain!).

`[myc]` number of meshes in computational grid in $y-$direction for a uniform, recti-linear grid or $\eta-$direction for a curvi-linear grid (this number is one less than the number of grid points in this domain!). In 1D-mode, `[myc]` should be 0.

`EXCEPTION` only available in the case of a curvi-linear grid. If certain grid points are to be ignored during the computation (e.g. land points that remain dry i.e. no flooding; saving computer time and memory), then this can be indicated by identifying these grid points in the file containing the grid point coordinates (see command `READGRID`). For an alternative, see command `INPGRID BOTTOM`.

`[xexc]` the value which the user uses to indicate that a grid point is to be ignored in the computations (this value is provided by the user at the location of the

$x-$coordinate considered in the file of the $x-$coordinates, see command READGRID COOR). Required if this option EXCEPTION is used.
Default: [xexc] = 0.0.

[yexc] the value which the user uses to indicate that a grid point is to be ignored in the computations (this value is provided by the user at the location of the $y-$coordinate considered in the file of the $y-$coordinates, see command READGRID COOR). Required if this option EXCEPTION is used.
Default: [yexc] = [xexc].

CIRCLE this option indicates that the spectral directions cover the full circle.
This option is default.

SECTOR this option means that only spectral wave directions in a limited directional sector are considered; the range of this sector is given by [dir1] and [dir2].
It must be noted that if the quadruplet interactions are to be computed (see command GEN3), then the SECTOR should be 30^o wider on each side than the directional sector occupied by the spectrum (everywhere in the computational grid). If not, then these computations are inaccurate. If the directional distribution of the spectrum is symmetric around the centre of the SECTOR, then the computed quadruplet wave-wave interactions are effectively zero in the 30^o range on either end of the SECTOR. The problem can be avoided by not activating the quadruplet wave-wave interactions (use command GEN1 or GEN2) or, if activated (with command GEN3), by subsequently de-activating them with command OFF QUAD.

[dir1] the direction of the right-hand boundary of the sector when looking outward from the sector (required for option SECTOR) in degrees.

[dir2] the direction of the left-hand boundary of the sector when looking outward from the sector (required for option SECTOR) in degrees.

[mdc] number of meshes in $\theta-$space. In the case of CIRCLE, this is the number of subdivisions of the 360 degrees of a circle, so $\Delta\theta = [360^o]/$[mdc] is the spectral directional resolution. In the case of SECTOR, $\Delta\theta =$ ([dir2] - [dir1])/[mdc].
The minimum number of directional bins is 3 per directional quadrant.

[flow] lowest discrete frequency that is used in the calculation (in Hz).

[fhigh] highest discrete frequency that is used in the calculation (in Hz).

[msc] one less than the number of frequencies. This defines the grid resolution in frequency-space between the lowest discrete frequency [flow] and the highest discrete frequency [fhigh]. This resolution is not constant, since the frequencies are distributed logarithmical: $f_{i+1} = \gamma f_i$ with γ is a constant. The minimum number of frequencies is 4.
The value of [msc] depends on the frequency resolution Δf that the user requires. Since, the frequency distribution on the frequency axis is logarithmic, the relationship is:

$$\Delta f = \left(-1 + \left[\frac{\texttt{[fhigh]}}{\texttt{[flow]}} \right]^{1/\texttt{[msc]}} \right) f$$

Vice versa, if the user chooses the value of $\Delta f/f$ $(= \gamma - 1.)$, then the value of `[msc]` is given by:

$$\texttt{[msc]} = \log(\texttt{[fhigh]}/\texttt{[flow]})/\log(1 + \Delta f/f)$$

In this respect, it must be observed that the DIA approximation of the quadruplet interactions (see command `GEN3`) is based on a frequency resolution of $\Delta f/f = 0.1$ and hence, $\gamma = 1.1$. The actual resolution in the computations should therefore not deviate too much from this. Alternatively, the user may only specifies one of the following possibilities:

- `[flow]` and `[msc]`; SWAN will compute `[fhigh]`, such that $\gamma = 1.1$, and write it to the `PRINT` file.
- `[fhigh]` and `[msc]`; SWAN will compute `[flow]`, such that $\gamma = 1.1$, and write it to the `PRINT` file.
- `[flow]` and `[fhigh]`; SWAN will compute `[msc]`, such that $\gamma = 1.1$, and write it to the `PRINT` file.

For illustration of a regular grid with its dimensions, see Figure 4.1.

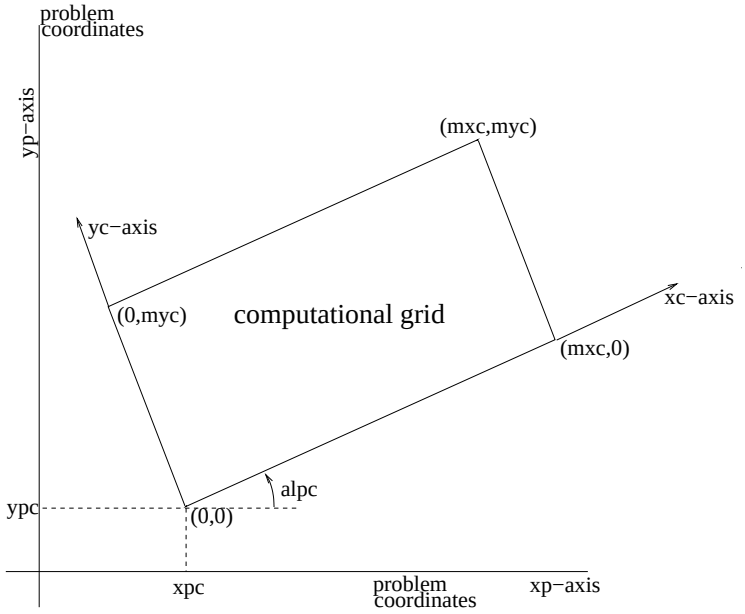


Figure 4.1: Coordinates of the origin `[xpc]` and `[ypc]`, the orientation `[alpc]` and the grid point numbering of the computational grid with respect to the problem coordinates system. Note that in case of spherical coordinates the $xc-$ and $xp-$axes both point East.

```
READgrid COORdinates [fac] 'fname' [idla] [nhedf] [nhedvec] &

                            | -> FREe                 |
                            |                         |
                            |             | 'form' |  |
                           <     FORmat  <          >  >
                            |             | [idfm] |  |
                            |                         |
                            |    UNFormatted          |
```

CANNOT BE USED IN 1D-MODE.

This command READGRID COOR must follow a command CGRID CURV. With this command (required if the computational grid is curvi-linear; not allowed in case of a regular grid) the user controls the reading of the co-ordinates of the computational grid points. These co-ordinates must be read from a file as a vector ($x-$coordinate, $y-$coordinate of each single grid point). See command READINP for the description of the options in this command READGRID. SWAN will check whether all angles in the grid are > 0 and < 180 degrees. If not, it will print an error message giving the coordinates of the grid points involved. It is recommended to use grids with angles between 45 and 135 degrees.

```
                          | -> ADCirc
                          |
READgrid UNSTRUCtured  <      TRIAngle |
                          |              > 'fname'
                          |    EASYmesh |
```

CANNOT BE USED IN 1D-MODE.

This command READGRID UNSTRUC must follow a command CGRID UNSTRUC. With this command (required if the computational grid is unstructured; not allowed in case of a regular or curvi-linear grid) the user controls the reading of the (x, y) co-ordinates of the vertices including boundary markers and a connectivity table for triangles (or elements). This table contains three corner vertices around each triangle in counterclockwise order. This information should be provided by a number of files generated by one of the following grid generators currently supported by SWAN:

- ADCIRC (*http://www.adcirc.org*)
- Triangle (*http://www.cs.cmu.edu/afs/cs/project/quake/public/www/triangle.html*)

- Easymesh (*http://www-dinma.univ.trieste.it/nirftc/research/easymesh/easymesh.html*)

After setting up the vertices and the connectivity tables for cells and faces (automatically done in SWAN), SWAN will print some information concerning the used mesh, among others, number of vertices, cells and faces and minimum and maximum gridsizes. Furthermore, SWAN will check at two levels for a possible occurence of badly shaped triangles. Firstly, the number of triangles that meet at each vertex inside the mesh should not be smaller than 4 or larger than 10. Secondly, the angles inside each triangle should not be higher than 143^o. If, at least, one of these two situations occur, SWAN will print an error message.

`ADCIRC`	the necessary grid information is read from file `fort.14` as used by ADCIRC. This file also contains the depth information that is read as well.
`TRIANGLE`	the necessary grid information is read from two files as produced by Triangle. The `.node` and `.ele` files are required. The basename of these files must be indicated with parameter `'fname'`.
`EASYMESH`	the necessary grid information is read from two files as produced by Easymesh. The `.n` and `.e` files are required. The basename of these files must be indicated with parameter `'fname'`.
`'fname'`	basename of the required files, i.e. without extension. Only meant for Triangle and Easymesh.

4.5.2 Input grids and data

```
              | BOTtom      |
              |             |
              | WLEVel      |
              |             |
              |  | CURrent  |
              | <           |
              |  | VX       |
              |  | VY       |
              |             |
INPgrid     (<               >)                                    &
              | FRiction    |
              |             |
              |  | WInd     |
              | <           |
              |  | WX       |
              |  | WY       |
```

```
       | -> REGular [xpinp] [ypinp] [alpinp] [mxinp] [myinp] [dxinp] [dyinp] |
       |                                                                      |
      <     CURVilinear  [stagrx] [stagry] [mxinp] [myinp]                    > &
       |                                                                      |
       |    UNSTRUCtured                                                      |

        (EXCeption  [excval])                                                  &

                                            | -> Sec  |
        (NONSTATionary [tbeginp] [deltinp] <     MIn   >  [tendinp])
                                            |    HR   |
                                            |    DAy  |
```

OPTIONS CURVILINEAR AND UNSTRUCTURED NOT FOR 1D-MODE.

With this required command the user defines the geographic location, size and orientation of an input grid and also the time characteristics of the variable if it is not stationary. If this is the case (the variable is not stationary), the variable should be given in a sequence of fields, one for each time step `[deltinp]`. The actual reading of values of bottom levels, currents, etc. from file is controlled by the command `READINP`. This command `INPGRID` must precede the following command `READINP`.

There can be different grids for bottom level (`BOTTOM`), flow current (`CURRENT`), bottom friction coefficient (`FRICTION`) and wind velocity (`WIND`). If the current velocity components are available on different grids, then option `VX`, `VY` can define these different grids for the $x-$ and $y-$component of the current, respectively (but the grids must have identical orientation). Different grids for `VX` and `VY` may be useful if the data are generated by a circulation model using a staggered grid. The same holds for the wind velocity components. If the command `INPGRID` is given without any of the keywords `BOTTOM`, `WIND`, etc. it is assumed that all the input grids are the same.

In the case of a regular grid (option `REGULAR` in the `INPGRID` command) the current and wind vectors are defined with the $x-$ and $y-$component of the current or wind vector with respect to the $x-$axis of the input grid. In case of a curvi-linear grid (option `CURVILINEAR` in the `INPGRID` command) the current and wind vectors are defined with the $x-$ and $y-$component of the current or wind vector with respect to the $x-$axis of the problem coordinate system. For wind velocity and friction coefficient it is also possible to use a constant value over the computational field (see commands `WIND` and `FRICTION`). No grid definition for wind and friction is then required.

Note that in case of option `BOTTOM` only stationary input field is allowed.

If the computational grid is unstructured (generated by Triangle or Easymesh), the input

grids can be either regular or identical to the used computational grid.

Do not use the command `INP BOTTOM` when the unstructured grid of ADCIRC is employed! The file `fort.14` contains the bottom levels and will be read by SWAN through the command `READ UNSTRUC ADCIRC`.

If land points remain dry during the computation (no flooding!), then these points can be ignored. In this way, turn-around time and internal memory can be saved. This can be done by indicating bottom level in these points as exception value. See command `INPGRID BOTTOM EXCEPTION`.

For parallel runs using MPI, an exception value for bottom levels should be prescribed in order to have a good load-balancing!

See Section 2.6 for more information on grids.

`BOTTOM`	defines the input grid of the bottom level. (For the definition of the bottom level, see command `READINP`).
`WLEV`	water level relative to datum level, positive upward (in m).
`CURRENT`	defines the input grid of the current field (same grid for $x-$ and $y-$components).
`VX`	defines the input grid of the $x-$component of the current field (different grid than $y-$component but same orientation).
`VY`	defines input grid of the $y-$component of the current field (different grid than $x-$component but same orientation).
`FRICTION`	defines input grid of the bottom friction coefficient (defined in command `FRICTION`, not to be confused with this option `FRICTION`!).
`WIND`	defines input grid of the wind field (same grid for $x-$ and $y-$component). If neither of the commands `WIND` and `READINP WIND` is used it is assumed that there is no wind.
`WX`	defines input grid of the $x-$component of the wind field (different grid than x-component but same orientation).
`WY`	defines input grid of the $y-$component of the wind field (different grid than $y-$component but same orientation).
`REGULAR`	means that the input grid is uniform and rectangular.
`CURVILINEAR`	means that the input grid is curvi-linear; this option is available only if the computational grid is curvi-linear as well. The input grid is identical (which is default) to the computational grid, or it is staggered in $x-$ and/or $y-$direction. NOT FOR 1D-MODE.
`UNSTRUCTURE`	means that the input grid is unstructured; this option is available only if the computational grid is unstructured as well. The input grid must be identical to the computational grid. NOT FOR 1D-MODE.

For a REGULAR grid:

[xpinp] geographic location ($x-$coordinate) of the origin of the input grid in problem coordinates (in m) if Cartesian coordinates are used or in degrees if spherical coordinates are use (see command COORD).
Default: [xpinp] = 0. In case of spherical coordinates there is no default, the user must give a value.

[ypinp] geographic location ($y-$coordinate) of the origin of the input grid in problem coordinates (in m) if Cartesian coordinates are used or in degrees if spherical coordinates are use (see command COORD).
Default: [ypinp] = 0. In case of spherical coordinates there is no default, the user must give a value.

[alpinp] direction of the positive $x-$axis of the input grid (in degrees, Cartesian convention). See command COORD.
Default: [alpinp] = 0.

[mxinp] number of meshes in $x-$direction of the input grid (this number is one less than the number of grid points in this direction!).

[myinp] number of meshes in $y-$direction of the input grid (this number is one less than the number of grid points in this direction!).
In 1D-mode, [myinp] should be 0.

[dxinp] mesh size in $x-$direction of the input grid,
in m in case of Cartesian coordinates or
in degrees if spherical coordinates are used, see command COORD.

[dyinp] mesh size in $y-$direction of the input grid,
in m in case of Cartesian coordinates or
in degrees if spherical coordinates are used, see command COORD.
In 1D-mode, [dyinp] may have any value.
Default: [dyinp] = [dxinp].

For a CURVILINEAR input (not fully tested for spherical coordinates):

[mxinp] number of meshes in $\xi-$direction of the input grid (this number is one less than the number of grid points in this direction!).
Default: [mxinp] = [mxc].

[myinp] number of meshes in $\eta-$direction of the input grid (this number is one less than the number of grid points in this direction!).
Default: [myinp] = [myc].

[stagrx] staggered $x'-$direction with respect to computational grid; default: 0.
Note: e.g. [stagrx]=0.5 means that the grid points are shifted a half step in $x'-$direction; in many flow models $x-$velocities are defined in points shifted a half step in $x'-$direction.

[stagry] staggered $y'-$direction with respect to computational grid; default: 0.
Note: e.g. [stagry]=0.5 means that the grid points are shifted a half step in

$y'-$direction; in many flow models $y-$velocities are defined in points shifted a half step in $y'-$direction.

`EXCEPTION` certain points inside the given grid that are to be ignored during the computation can be identified by means of an exception value as given in the corresponding input file as controlled by the command `READINP`.
NOT FOR 1D-MODE.

`[excval]` exception value; required if the option `EXCEPTION` is used.
Note: if `[fac]` $\neq 1$ (see command `READINP`), `[excval]` must be given as `[fac]` times the exception value.

`NONSTATION` the variable is nonstationary (given in a time sequence of fields).
NOT FOR 1D-MODE.

`[tbeginp]` begin time of the first field of the variable, the format is:

1 : ISO-notation	19870530.153000
2 : (as in HP compiler)	'30−May−87 15:30:00'
3 : (as in Lahey compiler)	05/30/87.15:30:00
4 :	15:30:00
5 :	87/05/30 15:30:00'
6 : as in WAM	8705301530

This format is installation dependent. See Implementation Manual or ask the person who installed SWAN on your computer. Default is ISO-notation.

`[deltinp]` time interval between fields, the unit is indicated in the next option:

SEC	unit seconds
MIN	unit minutes
HR	unit hours
DAY	unit days

`[tendinp]` end time of the last field of the variable, the format is:

1 : ISO-notation	19870530.153000
2 : (as in HP compiler)	'30−May−87 15:30:00'
3 : (as in Lahey compiler)	05/30/87.15:30:00
4 :	15:30:00
5 :	87/05/30 15:30:00'
6 : as in WAM	8705301530

This format is installation dependent. See Implementation Manual or ask the person who installed SWAN on your computer. Default is ISO-notation.

```
|  BOTtom      |
|              |
|  WLEVel      |
|              |
|  CURrent     |
|              |           |  'fname1'           |
```

```
READinp  <   WInd          >  [fac]  <                        >   [idla]    &
          |                |           |  SERIes   'fname2'   |
          |  FRiction      |

                                        | -> FREe                        |
                                        |                                |
                                        |                  | 'form' |    |
          [nhedf] ([nhedt]) ([nhedvec])  <     FORmat      <          >    >
                                        |                  | [idfm] |    |
                                        |                                |
                                        |    UNFormatted                 |
```

With this required command the user controls the reading of values of the indicated variables from file. This command READINP must follow a command INPGRID. Note that for each stationary or nonstationary field, one combination of INPGRID and READINP suffices if one has more than one COMPUTE command in a run.

If the variables are in one file, then the READINP commands should be given in the same sequence as the sequence in which the variables appear in the file.

BOTTOM with this option the user indicates that bottom levels (m) are to be read from file (bottom level positive downward relative to an arbitrary horizontal datum level). The sign of the input can be changed with option [fac] $= -1$. (see below).

WLEV with this option the user indicates that water levels (m) are to be read from file (water level positive upward relative to the same datum level as used in option BOTTOM). Sign of input can be changed with option [fac] $= -1$. If the water level is constant in space and time, the user can use the command SET to add this water level to the water depth.

CURRENT recti-linear (curvi-linear) input grid: with this option the user indicates that the $x-$ and $y-$component ($\xi-$ and $\eta-$component) are to be read from one and the same file (with one READINP command). With this option SWAN reads first all $x-$components ($\xi-$components), and then all $y-$components ($\eta-$components) (see below). The firs component ($x-$ or $\xi-$component) is always eastward oriented and the second one ($y-$ or $\eta-$component) is always northwise oriented. The components ξ and η are taken along the directions of the grid lines of the curvi-linear grid! If the current velocity is relatively large, i.e. the Froude number $U/\sqrt{gd}$ is larger than 0.8, it will be reduced such that the Froude number becomes equal to 0.8.

FRICTION with this option the user indicates that friction coefficient is to be read from file for Collins: [cfw] and for Madsen: [kn] (no space- or time-variable coefficient for the Jonswap expression, see command FRICTION). If the coefficients are constant in space and time: see command FRICTION.

WIND recti-linear (curvi-linear) input grid: with this option the user indicates that

the $x-$ and $y-$component ($\xi-$ and $\eta-$component) are to be read from one and the same file (with one `READINP` command). With this option SWAN reads first all $x-$components ($\xi-$components), and then all $y-$component ($\eta-$components) (see below). The components ξ and η are taken along the directions of the grid lines of the curvi-linear grid! If the wind is constant, see command `WIND`.

`[fac]` SWAN multiplies all values that are read from file with `[fac]`. For instance if the bottom levels are given in unit decimeter, one should make `[fac]`=0.1 to obtain levels in m. To change sign of bottom level use a negative value of `[fac]`. Note that `[fac]` = 0 is not allowed!
Default: `[fac]`=1.

`'fname1'` name of the file with the values of the variable.

`SERIES` with this option (only for `MODE NONSTATIONARY`) the user indicates that the names of the files containing the nonstationary variable(s) are located in a separate file with name `'fname2'` (see below).

`'fname2'` name of file that contains the names of the files where the variables are given. These names are to be given in proper time sequence. SWAN reads the next file when the previous file end has been encountered. In these files the input should be given in the same format as in the above file `'fname1'` (that implies that a file should start with the start of an input time step).

`[idla]` prescribes the order in which the values of bottom levels and other fields should be given in the file.

=1: SWAN reads the map from left to right starting in the upper-left-hand corner of the map (it is assumed that the $x-$axis of the grid is pointing to the right and the $y-$axis upwards). A new line in the map should start on a new line in the file. The lay-out is as follows:

```
1,myc+1     2,myc+1     ...     mxc+1, myc+1
1,myc       2,myc       ...     mxc+1, myc
...         ...         ...     ...
1,1         2,1         ...     mxc+1, 1
```

=2: as `[idla]`=1 but a new line in the map need not start on a new line in the file.

=3: SWAN reads the map from left to right starting in the lower-left-hand corner of the map. A new line in the map should start on a new line in the file. The lay-out is as follows:

```
1,1         2,1         ...     mxc+1, 1
1,2         2,2         ...     mxc+1, 2
...         ...         ...     ...
1,myc+1     2,myc+1     ...     mxc+1, myc+1
```

=4: as `[idla]`=3 but a new line in the map need not start on a new line

in the file.

=5: SWAN reads the map from top to bottom starting in the lower-left-hand corner of the map. A new column in the map should start on a new line in the file. The lay-out is as follows:

```
1,1          1,2          ...      1, myc+1
2,1          2,2          ...      2, myc+1
...          ...          ...      ...
mxc+1,1      mxc+1,2      ...      mxc+1, myc+1
```

=6: as [idla]=5 but a new column in the map need not start on a new line in the file.

Default: [idla]=1.

[nhedf] is the number of header lines at the start of the file. The text in the header lines is reproduced in the print file created by SWAN (see Section 3.3). The file may start with more header lines than [nhedf] because the start of the file is often also the start of a time step and possibly also of a vector variable (each having header lines, see below, [nhedt] and [nhedvec]).
Default: [nhedf]=0.

[nhedt] only if variable is time dependent: number of header lines in the file at the start of each time level. A time step may start with more header lines than [nhedt] because the variable may be a vector variable which has its own header lines (see below [nhedvec]).
Default: [nhedt]=0.

[nhedvec] for each vector variable: number of header lines in the file at the start of each component (e.g., $x-$ or $y-$component).
Default: [nhedvec]=0.

FREE With this option the user indicates that the values are to be read with free format. Free format is a standard of the computer programming language FORTRAN. The free format conventions in reading from a file are almost the same as the conventions for the command syntax given elsewhere in this manual; the most important differences are:

1. There are no continuation marks, reading continues until the required number of data has been read, or until a slash (/) is encountered.
2. Input lines can be longer than 80 characters (depending on the operating system of the computer).
3. Comment is not allowed.

With free format empty fields, repetition factors, and closure of a line by a slash, can be used.

FORMAT with this option the user indicates that fixed format (FORTRAN convention) is to be used when reading the values from file. The format can be defined in one of two ways, by giving the format number [idfm] or the format string 'form'.

'form' a user−specified format string according to Fortran convention, e.g.

`'(10X,12F5.0)'`.

=1: Format according to BODKAR convention (a standard of the Ministry of Transport and Public Works in the Netherlands).
Format string: `(10X,12F5.0)`.

=5: Format `(16F5.0)`, i.e. an input line consists of 16 fields of 5 places each.

=6: Format `(12F6.0)`, i.e. an input line consists of 12 fields of 6 places each.

=8: Format `(10F8.0)`, i.e. an input line consists of 10 fields of 8 places each.

`UNFORMATTED` is a form of reading without conversion (binary files). Not recommended for ordinary use.

If the file does not contain a sufficient number of data (i.e. less than the number of grid points of the input grid), SWAN will write an error message to the `PRINT` file, and if `[itest]`>0 (see command `TEST`) it will reproduce the data in the `PRINT` file, using the lay-out according to `[idla]`=1. This echo of the data to print file is also made if the READINP command is embedded between two `TEST` commands in the command file as follows:

```
TEST 120
READINP ....
TEST 0
```

`WIND  [vel] [dir]`

With this optional command the user indicates that the wind is constant.

`[vel]` wind velocity at 10 m elevation (m/s).

`[dir]` wind direction at 10 m elevation (in degrees, Cartesian or Nautical convention, see command `SET`).

Both quantities are required if this command is used. Note that SWAN converts U_{10} to U_* (see Scientific/Technical documentation).

4.5.3 Boundary and initial conditions

```
                  | -> JONswap  [gamma]  |
                  |                      |    | -> PEAK  |
BOUNd SHAPespec   <     PM                >  <           >   &
                  |                      |    |    MEAN  |
                  |    GAUSs  [sigfr]    |
```

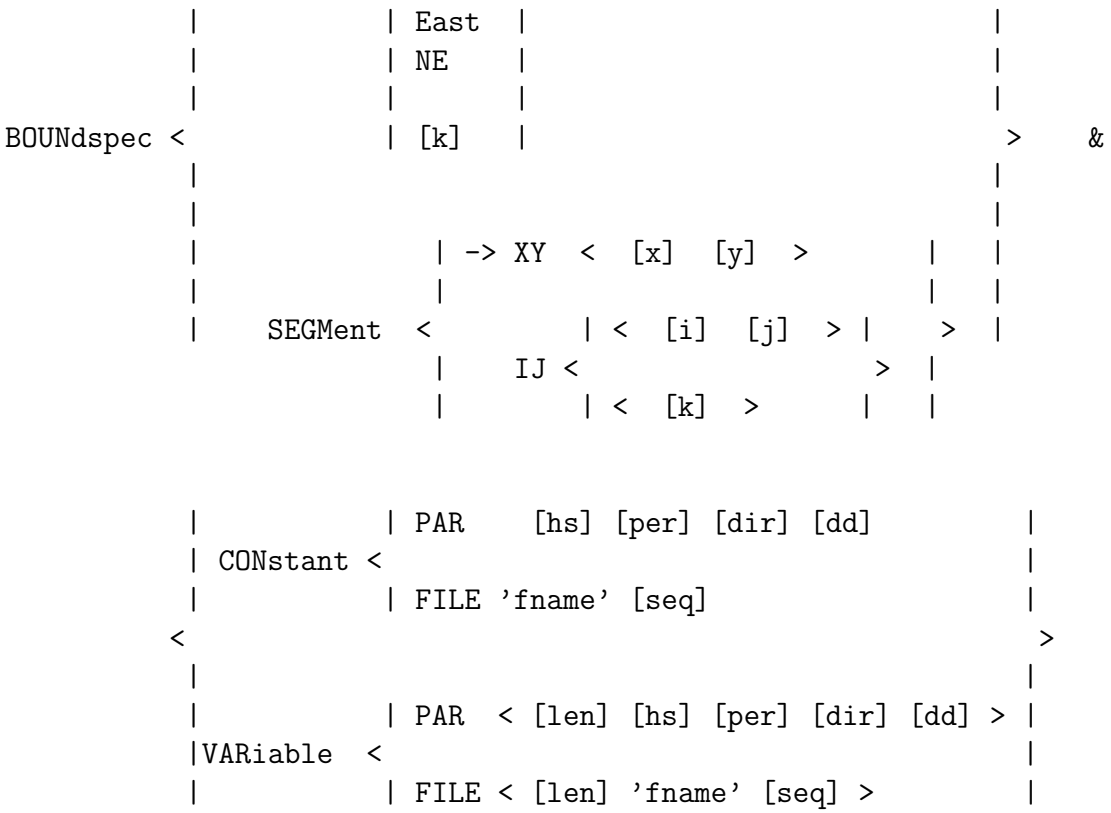
```
                |                        |
                |    BIN                 |

                     | -> POWer        |
          DSPR   <                      >
                     |    DEGRees      |
```

This command BOUND SHAPESPEC defines the shape of the spectra (both in frequency and direction) at the boundary of the computational grid in case of parametric spectral input (see command BOUNDSPEC).

JONSWAP	JONSWAP spectrum will be used. This is default.
[gamma]	peak enhancement parameter of the JONSWAP spectrum. Default: [gamma]=3.3.
PM	Pierson-Moskowitz spectrum will be used.
GAUSS	a Gaussian-shaped frequency spectrum will be used.
BIN	energy is located in one frequency bin (the frequency bin closest to the [per] value of command BOUNDSPEC).
[sigfr]	width of the Gaussian frequency spectrum expressed as a standard deviation in Hz.
PEAK	Default: the peak period (for definition, see Appendix A) is used as characteristic wave period. This is default.
MEAN	T_{m01} (for definition, see Appendix A) is used as the characteristic wave period.
DSPR	option for expressing the width of the directional distribution (the distribution function itself is $\cos^m(\theta - \theta_{\text{peak}})$.
POWER	the directional width is expressed with the power m itself, this option is default (note that the directional resolution should accommodate the directional width, see command CGRID).
DEGREES	the directional width is expressed in terms of the directional standard deviation of the $\cos^m(\theta - \theta_{\text{peak}})$ distribution (for definition, see Appendix A). (Note that the directional resolution should accommodate the directional width, see command CGRID).

If this command is not used, the JONSWAP option will be used by SWAN with [gamma]=3.3 and POWER for the directional width.

```
                     | North |
                     | NW    |
                     | West  |
                     | SW    |   | -> CCW      |
          | -> SIDE  <  South  > <               >          |
          |          | SE    |   | CLOCKWise   |          |
```



This command BOUNDSPEC defines parametric spectra at the boundary. It consists of two parts, the first part defines the boundary side or segment where the spectra will be given, the second part defines the spectral parameters of these spectra. Note that in fact only the incoming wave components of these spectra are used by SWAN. The fact that complete spectra are calculated at the model boundaries from the spectral parameters should not be misinterpreted. Only the incoming components are effective in the computation.

There are two ways to define the part of the boundary at which the spectra are imposed. The first (SIDE) is easiest if the boundary is one full side of the computational grid, although it should not be used for curvi-linear grids. The second (SEGMENT) can be used if the boundary segment goes around the corner of the grid, or if the segment is only part of one side of the grid.

This BOUNDSPEC command can be given a number of times, i.e. to define incident wave fields on various sides or segments of the boundary. One BOUNDSPEC command can be used for only one side or one contiguous segment.

SIDE — the boundary is one full side of the computational grid (in 1D cases either of the two ends of the 1D-grid).
SHOULD NOT BE USED IN CASE OF CURVI-LINEAR GRIDS!

NORTH, ... — indicates on which side the boundary condition is applied. N means the

boundary is the north edge (if present) of the computational area, likewise for `W` is west, `S` is south, `E` is east, `NW` is northwest, `NE` is northeast, `SW` is southwest and `SE` is southeast. The side does not have to face exactly the given direction (the nearest direction of the normal to the side is taken; this direction is determined as the normal to the sum of the vectors joining the grid points on the boundary; there is an interruption in the boundary (due to the occurrence of exception values) then this interruption is ignored in the summation).
Note: in case of Cartesian coordinates, the direction of the problem coordinate system must be defined by the user (see the `SET ...[north]` command), by default the positive $x-$axis points East.
ONLY MEANT FOR REGULAR GRIDS.

`[k]` indicates on which side of the unstructured grid the boundary condition is applied. The value of `[k]` corresponds to the boundary marker as indicated in file(s) produced by a grid generator (such as in the last column of the Triangle `.node` file and the Easymesh `.n` file or the last part of file `fort.14`). Boundary markers are tags to identify which vertices occur on a boundary of the mesh.
ONLY MEANT FOR UNSTRUCTURED MESHES.

`CCW`, `CLOCKWISE` see description of `[len]` below; these option are only effective if the option `VARIABLE` is used (see below).

`SEGMENT` is used if `SIDE` is not used, i.e. either the boundary segment goes around a corner of the grid, or the segment is only part of one side of the grid. The distance along the segment (see `[len]` below) is measured from the first point of the segment (see `XY` or `IJ`).

`XY` the segment is defined by means of a series of points in terms of problem coordinates; these points do not have to coincide with grid points. The (straight) line connecting two points must be close to grid lines of the computational grid (the maximum distance is one hundredth of the length of the straight line).
This option is default.

`[x], [y]` problem coordinates of a point of the boundary segment (see command `COORD`).

`IJ` the segment is defined by means of a series of computational grid points given in terms of grid indices (origin at 0,0); not all grid points on the segment have to be mentioned. If two points are on the same grid line, intermediate points are assumed to be on the segment as well.

`[i], [j]` grid indices of a point of the segment. Values of `[i]` range between 0 and `[mxc]` (see command `CGRID`), values of `[j]` between 0 and `[myc]` (inclusive).
ONLY MEANT FOR STRUCTURED GRIDS.

`[k]` index of boundary vertex of the segment. This can be obtained in a grid generator file (`fort.14`, `.node` and `.n` files of ADCIRC, Triangle and Easymesh, respectively).
ONLY MEANT FOR UNSTRUCTURED MESHES.

`CONSTANT` with this option the wave spectra are constant along the side or segment.

`VARIABLE` with this option the wave spectra can vary along the side or segment. The incident wave field is prescribed at a number of points of the side or segment, these points are characterized by their distance from the begin point of the side or segment. The wave spectra for grid points on the boundary of the computational grid are calculated by SWAN by the spectral interpolation technique described in Section 2.6.3.

`PAR` the wave spectra are defined by means of the following spectral parameters (see command `BOUND SHAPE` for spectral shape).

`[hs]` the significant wave height (in m).

`[per]` the characteristic period of the energy spectrum (relative frequency; which is equal to absolute frequency in the absence of currents);

- `[per]` is the value of the peak period (in s), if option `PEAK` is chosen in command `BOUND SHAPE` or
- `[per]` is the value of the mean period, if option `MEAN` was chosen in command `BOUND SHAPE`.

`[dir]` the peak wave direction (θ_{peak}, direction in degrees, constant over frequencies).

`[dd]` coefficient of directional spreading; a $cos^m(\theta)$ distribution is assumed.

- `[dd]` is interpreted as the directional standard deviation in degrees, if the option `DEGREES` is chosen in the command `BOUND SHAPE`. Default: `[dd]`=30.
- `[dd]` is interpreted as the power m, if the option `POWER` is chosen in the command `BOUND SHAPE`. Default: `[dd]`=2.

`[len]` is the distance from the first point of the side or segment to the point along the side or segment for which the incident wave spectrum is prescribed. Note: these points do no have to coincide with grid points of the computational grid. `[len]` is the distance in m or degrees in the case of spherical coordinates, **not** in grid steps. The values of `[len]` should be given in ascending order. The length along a `SIDE` is measured in clockwise or counterclockwise direction, depending on the options `CCW` or `CLOCKWISE` (see above). The option `CCW` is default. In case of a `SEGMENT` the length is measured from the indicated begin point of the segment.

`FILE` means that the incoming wave data are read from a file. There are three types of files:

- TPAR files containing nonstationary wave parameters,
- files containing stationary or nonstationary 1D spectra (usually from measurements),
- files containing stationary or nonstationary 2D spectra (from other computer programs or other SWAN runs).

A `TPAR` file is for only one location; it has the string `TPAR` on the first line of the file and a number of lines which each contain 5 numbers, i.e.:

Time (ISO-notation), Hs, Period (average or peak period depending on the choice given in command `BOUND SHAPE`), Peak Direction (Nautical or Cartesian, depending on command `SET`), Directional spread (in degrees or as power of cos depending on the choice given in command `BOUND SHAPE`).

Example of a `TPAR` file:

```
TPAR
19920516.1300    4.2    12.    -110.    22.
19920516.1800    4.2    12.    -110.    22.
19920517.0000    1.2    8.     -110.    22.
19920517.1200    1.4    8.5    -80.     26
19920517.2000    0.9    6.5    -95.     28
```

The structure of the files containing 1D or 2D spectra is described in Appendix D (there is no relation with the definition of the boundary file generated by WAM or WAVEWATCH III). 1D and 2D files can be used for stationary and nonstationary boundary conditions, and for one or more than one location. The spectral frequencies (and directions in the case of a 2D spectrum) do not have to coincide with the frequencies and directions used in the present SWAN run (in a nested run SWAN will interpolate to these frequencies and directions). The coordinates of locations in the 1D and 2D files are ignored when SWAN reads this file (SWAN uses the geographical information in this `BOUNDSPEC` command instead).

`'fname'` name of the file containing the boundary condition.

`[seq]` sequence number of geographic location in the file (see Appendix D); useful for files which contain spectra for more than one location. Default: `[seq]` = 1 (i.e. first location). Note: a `TPAR` file always contains only one location so in this case `[seq]` must always be 1.

```
                          | -> CLOSed |
BOUNdnest1  NEST 'fname' <             >
                          |    OPEN   |
```

With this optional command a nested SWAN run can be carried out with the boundary conditions obtained from a coarse grid SWAN run (generated in that previous SWAN run with command `NESTOUT` not to be confused with option NEST in this command `BOUNDNEST1`). For this nested SWAN run the user has to give the `CGRID` command to define the computational grid before this `BOUNDNEST1` command. The computational grid for SWAN in geographic space is the area bounded by the SWAN coarse run nest (SWAN boundary

points of the nest). This implies that the boundaries of the SWAN coarse run nest and the boundaries of the SWAN nested computational area should be (nearly) identical (see below). The spectral frequencies and directions of the coarse grid run do not have to coincide with the frequencies and directions used in the nested SWAN run (as defined in the `CGRID` command); SWAN will interpolate to these frequencies and directions in the nested run (see Section 2.6.3).

To generate the nest boundary in the coarse grid run, use command `NGRID`. For the nested run, use the command `CGRID` with identical geographical information except the number of meshes (which will be much higher for the nested run).

This `BOUNDNEST` command is not available for 1D computations; in such cases the commands `SPECOUT` and `BOUNDSPEC` can be used for the same purpose.

A nested SWAN run must use the same coordinate system as the coarse grid SWAN run. For a curvi-linear grid, it is advised to use the commands `POINTS` or `CURVE` and `SPECOUT` instead of `NGRID` and `NESTOUT`.

`NEST` with this option the user indicates that the boundary conditions (all four sides of the computational grid) are to be retrieved from a file created by a previous SWAN run (the present SWAN run is a nested run). The spectral frequencies (and directions in the case of a 2D spectrum) of the previous run do not have to coincide with the frequencies and directions used in the present SWAN run (see command `CGRID`); SWAN will interpolate the energy densities to these frequencies and directions (see Section 2.6.3).

`'fname'` name of the file containing the boundary conditions for the present run, created by the previous SWAN coarse grid run. This file is structured according to the rules given in Appendix D for 2D spectra.

`CLOSED` the boundary represented in the file is a closed rectangle; this is always the case if the `NESTOUT` command was used to generate the boundary condition file.

`OPEN` the boundary represented in the file is not a closed rectangle.

```
                                        |-> CRAY |
                             | UNFormatted <          > |
                             |               | WKstat |  |
                             |                          |
BOUNdnest2  WAMNest 'fname' <                           > [xgc] [ygc]
                             |                          |
                             | FREE                     |
```

CANNOT BE USED IN CASE OF UNSTRUCTURED GRIDS.

With this optional command (**not fully tested**) a nested SWAN run can be carried out with the boundary conditions obtained from a coarse grid WAM run (WAM Cycle 4.5, source code as distributed by the Max Planck Institute in Hamburg). For this nested SWAN run the user has to give the `CGRID` command to define the computational grid before this `BOUNDNEST2` command. The computational grid for SWAN in geographic space is the area bounded by the WAM coarse run nest (WAM boundary points of the nest). This implies that the boundaries of the WAM nest and the boundaries of the SWAN computational area should be (nearly) identical (see below). The spectral frequencies and directions of the coarse grid run do not have to coincide with the frequencies and directions used in the nested SWAN run (as defined in the `CGRID` command); SWAN will interpolate to these frequencies and directions in the nested run (see Section 2.6.3).

Note that SWAN will accept output of a WAM output location only if the SWAN grid point on the nest boundary lies within a rectangle between two consecutive WAM output locations with a width equal to 0.1 times the distance between these output locations on either side of the line between these WAM output locations.

This BOUNDNEST command is not available for 1D computations.

Only boundary conditions generated by WAM Cycle 4.5 can be read properly by SWAN.

A nested SWAN run may use either Cartesian or spherical coordinates. A curvi-linear grid may be used in the nested grid but the boundaries of this nest should conform to the rectangular course grid nest boundaries.

WAM output files are unformatted (binary); this usually implies that WAM and SWAN have to run on the same computer. For those cases where WAM and SWAN run on different types of machines (binary files do not transfer properly), the option `FREE` is available in this command. The distributed version of WAM does not support the required free format nesting output; WAM users who modify WAM such that it can make formatted output, must modify WAM such that the files made by WAM can be read in free format, i.e. with at least a blank or comma between numbers.

Note that the format of time and date that can be accepted by SWAN is `YYMMDDHHMMSS` (i.e. include seconds).

`'fname'` a file name that contains all the names of WAM files containing the nested boundary conditions in time-sequence (usually one file per day). For example, the contents of `'fname'` can look like:
CBO9212010000
CBO9212020000
CBO9212030000
....
SWAN will read the boundary data from these WAM files one after the other.

`UNFORMATTED` the user indicates that the WAM files are binary.

`CRAY`	input will be read from file created by the CRAY version of WAM.
`WKSTAT`	input will be read from file created by the WORKSTATION version of WAM.
`FREE`	the user indicates that the WAM files can be read with free format (these files are not generated standard by WAM!).
`[xgc]`	if SWAN is used with Cartesian coordinates: longitude of south-west corner of SWAN computational grid (in degrees); if the south-west corner of the nest in the WAM computation is on land this value is required. If SWAN is used with spherical coordinates then `[xgc]` is ignored by SWAN. Default: the location of the first spectrum encountered in the nest file.
`[ygc]`	if SWAN is used with Cartesian coordinates: longitude of south-west corner of SWAN computational grid (in degrees); if the south-west corner of the nest in the WAM computation is on land this value is required. If SWAN is used with spherical coordinates then `[ygc]` is ignored by SWAN. Default: the location of the first spectrum encountered in the nest file.

```
                              | -> CLOSed |
BOUNdnest3  WWIII  'fname' <              >  [xgc] [ygc]
                              |    OPEN   |
```

CANNOT BE USED IN CASE OF UNSTRUCTURED GRIDS.

With this optional command (**not fully tested**) a nested SWAN run can be carried out with the boundary conditions obtained from a coarse grid WAVEWATCH III run. For this nested SWAN run the user has to give the `CGRID` command to define the computational grid before this `BOUNDNEST3` command. The computational grid for SWAN in geographic space is the area bounded by the WAVEWATCH III nest (WAVEWATCH III boundary points of the nest). This implies that the boundaries of the WAVEWATCH III nest and the boundaries of the SWAN computational area should be (nearly) identical (see below). The spectral frequencies and directions of the coarse grid run do not have to coincide with the frequencies and directions used in the nested SWAN run (as defined in the CGRID command); SWAN will interpolate to these frequencies and directions in the nested run (see Section 2.6.3).

The output files of WAVEWATCH III (version 1.18 as distributed by NOAA) have to be created with the post-processor of WAVEWATCH III as output transfer files with

```
WW_3 OUTP (output type 1 sub type 3)
```

at the locations along the nest boundary (i.e. computational grid points in WAVEWATCH III). These locations are equal to the corner points of the SWAN nested grid

and optionally also distributed between the corner points of the SWAN nested grid (the boundary of the WAVEWATCH III nested grid need not be closed and may cover land). The locations should be output by WAVEWATCH III in sequence (going along the nest boundary, clockwise or counterclockwise). Note that SWAN will accept output of a WAVEWATCH III output location only if the SWAN grid point on the nest boundary lies within a rectangle between two consecutive WAVEWATCH III output locations with a width equal to 0.1 times the distance between these output locations on either side of the line between these WAVEWATCH III output locations.

This BOUNDNEST command is not available for 1D computations.

A nested SWAN run may use either Cartesian or spherical coordinates. A curvi-linear grid may be used in the nested grid but the boundaries of this nest should conform to the rectangular course grid nest boundaries.

`'fname'`	the name of the file that contains the spectra computed by WAVEWATCH III.
`CLOSED`	the boundary condition represented in the file is defined on a closed rectangle.
`OPEN`	the curve on which the boundary condition is given, is not closed.
`[xgc]`	if SWAN is used with Cartesian coordinates: longitude of south-west corner of SWAN computational grid (in degrees); if the south-west corner of the nest in the WAM computation is on land this value is required. If SWAN is used with spherical coordinates then `[xgc]` is ignored by SWAN. Default: the location of the first spectrum encountered in the nest file.
`[ygc]`	if SWAN is used with Cartesian coordinates: longitude of south-west corner of SWAN computational grid (in degrees); if the south-west corner of the nest in the WAM computation is on land this value is required. If SWAN is used with spherical coordinates then `[ygc]` is ignored by SWAN. Default: the location of the first spectrum encountered in the nest file.

Note that `[xgc]` and `[ygc]` are ignored if SWAN is used with spherical coordinates; if SWAN is used with Cartesian coordinates the values must be provided by the user.

```
          | -> DEFault
          |
INITial   <   ZERO
          |
          |   PAR   [hs] [per] [dir] [dd]
          |
          |              | -> MULTiple |
          |   HOTStart <                >   'fname'
          |              |    SINGle    |
```

This command can be used to specify the initial values for a stationary (`INITIAL HOTSTART` only) or nonstationary computation. The initial values thus specified override the default initialization (see Section 2.6.3). Note that it is possible to obtain an initial state by carrying out a previous stationary or nonstationary computation.

`DEFAULT`	the initial spectra are computed from the local wind velocities, using the deep-water growth curve of Kahma and Calkoen (1992), cut off at values of significant wave height and peak frequency from Pierson and Moskowitz (1964). The average (over the model area) spatial step size is used as fetch with local wind. The shape of the spectrum is default JONSWAP with a $\cos^2$-directional distribution (options are available: see command `BOUND SHAPE`).
`ZERO`	the initial spectral densities are all 0; note that if waves are generated in the model only by wind, waves can become non-zero only by the presence of the "A" term in the growth model; see the keyword `AGROW` in command `GEN3`.
`PAR`	the spectra in the entire computational area are generated from integral parameters `[hs]` etc. in the same way as done for the boundary using the command `BOUNDSPEC`.
`[hs]`	the significant wave height.
`[per]`	characteristic wave period of the energy spectrum (either peak or mean period, as determined by the options `PEAK` and `MEAN` in the command `BOUND SHAPE`).
`[dir]`	the peak wave direction (direction in degrees, Nautical or Cartesian convention, see command `SET`).
`[dd]`	the coefficient of directional spreading; a $\cos^m(\theta)$ distribution is assumed. See the options `DEGREES` and `POWER` in the command `BOUND SHAPE`.
`HOTSTART`	initial wave field is read from file; this file was generated in a previous SWAN run by means of the `HOTFILE` command. If the previous run was nonstationary, the time found on the file will be assumed to be the initial time of computation. It can also be used for stationary computation as first guess. The computational grid (both in geographical space and in spectral space) must be identical to the one in the run in which the initial wave field was computed.
`MULTIPLE`	input will be read from multiple hotfiles obtained from a previous parallel MPI run. The number of files equals the number of processors. Hence, for the present run the same number of processors must be chosen.
`SINGLE`	input will be read from a single (concatenated) hotfile. In the case of a previous parallel MPI run, the concatenated hotfile can be created from a set of multiple hotfiles using the program `hcat.exe`, see Implementation Manual. Note: with this option you may change the number of processors when restart a parallel MPI run.
`'fname'`	name of the file containing the initial wave field.

4.5.4 Physics

`GEN1 [cf10] [cf20] [cf30] [cf40] [edmlpm] [cdrag] [umin] [cfpm]`

With this command the user indicates that SWAN should run in first-generation mode (see Scientific/Technical documentation).

`[cf10]`	controls the linear wave growth. Default: `[cf10]` = 188.
`[cf20]`	controls the exponential wave growth. Default: `[cf20]` = 0.59
`[cf30]`	controls the exponential wave growth. Default: `[cf30]` = 0.12
`[cf40]`	controls the dissipation rate, i.e., the time decay scale. Default: `[cf40]` = 250.
`[edmlpm]`	maximum non-dimensionless energy density of the wind sea part of the spectrum according to Pierson Moskowitz. Default: `[edmlpm]` = 0.0036
`[cdrag]`	drag coefficient. Default: `[cdrag]` = 0.0012
`[umin]`	minimum wind velocity (relative to current; all wind speeds are taken at 10m above sea level). Default: `[umin]` = 1.
`[cfpm]`	coefficient which determines the Pierson Moskowitz frequency: $\sigma_{PM} = 2\pi\, g$ `[cfpm]` $/U_{10}$ Default: `[cfpm]` = 0.13

`GEN2 [cf10] [cf20] [cf30] [cf40] [cf50] [cf60] [edmlpm] [cdrag] [umin] [cfpm]`

With this command the user indicates that SWAN should run in second-generation mode (see Scientific/Technical documentation). The variables are identical to those in the `GEN1` command except that `[cf50]` and `[cf60]` are added.

`[cf50]`	controls the spectral energy scale of the limit spectrum. Default: `[cf50]` = 0.0023
`[cf60]`	controls the spectral energy scale of the limit spectrum. Default: `[cf60]` = −0.223

```
          |    JANSsen [cds1] [delta] |
          |                           |
GEN3    < --> KOMen   [cds2] [stpm]   > (AGROW [a])
          |                           |
          |    WESTHuysen             |
```

With this command the user indicates that SWAN should run in third-generation mode for wind input, quadruplet interactions and whitecapping. Triads, bottom friction and depth-induced breaking are not activated by this command. See the Scientific/Technical documentation for more information. The option `GEN3 KOMEN` is default.

`JANSSEN` linear growth : Cavaleri and Malanotte-Rizzoli (1981), activated only if the keyword `AGROW` is present (see below)
exponential growth: Janssen (1989, 1991).

`[cds1]` coefficient for determining the rate of whitecapping dissipation ($=C_{\mathrm{ds}}/\tilde{s}^4_{\mathrm{PM}}$).
Default: `[cds1]` = 4.5.

`[delta]` coefficient which determines the dependency of the whitecapping on wave number (mix with Komen et al. formulation).
Default: `[delta]` = 0.5.

`KOMEN` linear growth : Cavaleri and Malanotte-Rizzoli (1981), activated only if the keyword `AGROW` is present (see below)
exponential growth: Komen et al. (1984).

`[cds2]` coefficient for determining the rate of whitecapping dissipation ($=C_{\mathrm{ds}}$).
Default: `[cds2]` = 2.36e-5.

`[stpm]` value of the wave steepness for a Pierson-Moskowitz spectrum ($=\tilde{s}^2_{\mathrm{PM}}$).
Default: `[stpm]` = 3.02e-3.

`WESTH` nonlinear saturation-based whitecapping combined with wind input of Yan (1987).

`AGROW` if this keyword is used, the wave growth term of Cavaleri and Malanotte (1981) is activated.
if this keyword is NOT used, the wave growth term of Cavaleri and Malanotte (1981) is NOT activated.
Note that in nonstationary runs SWAN start with `INIT ZERO` (see command `INIT`), wave energy remains zero unless wave energy penetrates over the boundary or `AGROW` is activated. In case of stationary runs, however, SWAN will start with a first guess.

`[a]` if the wave growth term of Cavaleri and Malanotte (1981) is activated, `[a]` is the proportionality coefficient in that term.
Default: `[a]` = 0.0015.

`WCAPping CSM [cst] [pow]`

With this command the user wants to choose the Cumulative Steepness Method (CSM) for approximating whitecapping (see Scientific/Technical documentation) and **not** the formulation of Komen *et al.* (1984) and **not** Janssen (1991a).

`[cst]` the tuneable coefficient C_{wc}^{st}
Default: `[cst]` = 4.0.

`[pow]` power m.
Default: `[pow]` = 2.0.

Note that the CSM method in SWAN is still in its experimental phase. Its results are promising, but the method still suffers some numerical problems.

`QUADrupl [iquad] [lambda] [Cnl4] [Csh1] [Csh2] [Csh3]`

With this option the user can influence the computation of nonlinear quadruplet wave interactions. Default: the quadruplets are included in the computations. Can be deactivated with command `OFF QUAD`. Note that the DIA approximation of the quadruplet interactions is a poor approximation for long-crested waves and frequency resolutions very different from 10% (see command `CGRID`).

`[iquad]` the quadruplets can be integrated by four different numerical procedures:
= 1 semi-implicit computation of the nonlinear transfer with DIA per sweep
= 2 fully explicit computation of the nonlinear transfer with DIA per sweep
= 3 fully explicit computation of the nonlinear transfer with DIA per iteration
= 8 fully explicit computation of the nonlinear transfer with DIA per iteration, but neighbouring interactions are interpolated in piecewise constant manner.
other techniques for the computation of quadruplets are
= 4 Multiple DIA
= 6 FD-RIAM
= 51 XNL (deep water transfer)
= 52 XNL (deep water transfer with WAM depth scaling)
= 53 XNL (finite depth transfer)
Default: `[iquad]` = 2.

`[lambda]` coefficient for quadruplet configuration in case of DIA.
Default: `[lambda]`=0.25.

`[Cnl4]` proportionality coefficient for quadruplet interactions in case of DIA.
Default: `[Cnl4]`=3×10^7.

`[Csh1]` coefficient for shallow water scaling in case of DIA.
Default: `[Csh1]`=5.5.

`[Csh2]` coefficient for shallow water scaling in case of DIA.
Default: `[Csh2]`=6/7.

`[Csh3]` coefficient for shallow water scaling in case of DIA.
Default: `[Csh3]`$=-1.25$.

`BREaking CONSTANT [alpha] [gamma]`

With this command the user can influence depth-induced wave breaking in shallow water in the SWAN model.

If this command is not used, SWAN will account for wave breaking anyhow (with default options and values). If the user wants to specifically ignore wave breaking, he should use the command: `OFF BREAKING`.

`CONSTANT` indicates that a constant breaker parameter is to be used.
`[alpha]` proportionality coefficient of the rate of dissipation.
Default: `[alpha]` = 1.0.
`[gamma]` the ratio of maximum individual wave height over depth.
Default: `[gamma]` = 0.73.

```
           | -> JONswap [cfjon]
           |
FRICTION   <     COLLins [cfw]
           |
           |     MADsen  [kn]
```

With this optional command the user can activate bottom friction. If this command is not used, SWAN will not account for bottom friction.

In SWAN three different formulations are available, i.e., that of Hasselmann et al. (1973, JONSWAP), Collins (1972), and Madsen et al. (1988). The default option is: `JONSWAP`.

`JONSWAP` indicates that the semi-empirical expression derived from the JONSWAP results for bottom friction dissipation (Hasselmann et al., 1973, JONSWAP) should be activated. This option is default.
`[cfjon]` coefficient of the JONSWAP formulation. `[cfjon]` is equal to $0.038 m^2 s^{-3}$ for swell conditions (Hasselmann et al., 1973) and equal to $0.067 m^2 s^{-3}$ for wind sea conditions.
Default: `[cfjon]` = 0.067.
`COLLINS` indicates that the expression of Collins (1972) should be activated.

`[cfw]` Collins bottom friction coefficient.
Default: `[cfw]` = 0.015.
Note that `[cfw]` is allowed to vary over the computational region; in that case use the commands `INPGRID FRICTION` and `READINP FRICTION` to define and read the friction data. The command `FRICTION` is still required to define the type of friction expression. The value of `[cfw]` in this command is then not required (it will be ignored).

`MADSEN` indicates that the expression of Madsen et al. (1988) should be activated.

`[kn]` equivalent roughness length scale of the bottom (in m).
Default: `[kn]` = 0.05.
Note that `[kn]` is allowed to vary over the computational region; in that case use the commands `INPGRID FRICTION` and `READINP FRICTION` to define and read the friction data. This command `FRICTION` is still required to define the type of friction expression. The value of `[kn]` in this command is then not required (it will be ignored).

`TRIad [trfac] [cutfr] [urcrit] [urslim]`

With this command the user can activate the triad wave-wave interactions using the LTA method in the SWAN model. If this command is not used, SWAN will not account for triads.

`[trfac]` the value of the proportionality coefficient α_{EB}.
Default: `[trfac]` = 0.05.

`[cutfr]` controls the maximum frequency that is considered in the triad computations. The value of `[cutfr]` is the ratio of this maximum frequency over the mean frequency.
Default: `[cutfr]` = 2.5.

`[urcrit]` the critical Ursell number appearing in the expression for the biphase.
Default: `[urcrit]` = 0.2.

`[urslim]` the lower threshold for Ursell number; if the actual Ursell number is below `[urslim]` triad interactions will not be computed.
Default: `[urslim]` = 0.01.

`LIMiter [ursell] [qb]`

With this command the user can de-activate permanently the quadruplets when the actual Ursell number exceeds `[ursell]`. Moreover, as soon as the actual fraction of breaking waves exceeds `[qb]` then the limiter will not be used in case of decreasing action density.

`[ursell]` the upper threshold for Ursell number.
Default: `[ursell]` = 10.0.

`[qb]` the threshold for fraction of breaking waves.
Default: `[qb]` = 1.0.

```
          |  TRANSm [trcoef]                          |
OBSTacle <                                            |
          |       | -> GODA [hgt] [alpha] [beta]       >                    &
          |  DAM <                                    |
                 |    DANGremond [hgt] [slope] [Bk]  |

                        | -> RSPEC        |
          (REFL [reflc] <                   > ) LINe <[xp] [yp]>
                        |    RDIFF [pown] |
```

CANNOT BE USED IN 1D-MODE.

With this optional command the user provides the characteristics of a (line of) sub-grid obstacle(s) through which waves are transmitted or against which waves are reflected (possibly both at the same time). The obstacle is sub-grid in the sense that it is narrow compared to the spatial meshes; its length should be at least one mesh length.

The location of the obstacle is defined by a sequence of corner points of a line. The obstacles interrupt the propagation of the waves from one grid point to the next wherever this obstacle line is located between two neighbouring grid points (of the computational grid; the resolution of the obstacle is therefore equal to the computational grid spacing). This implies that an obstacle to be effective must be located such that it crosses at least one grid line. This is always the case when an obstacle is larger than one mesh length.

1. If a straight line is defined with more than two points, then the sum of the reflection of the parts may differ from the situation when you define it with just two points. This is due to the way obstacles are handled numerically in SWAN. It defines from computational grid point to its neighbor whether there is a crossing with an obstacle. In defining which directions of the wave spectrum should be reflected, i.e which directions are pointed towards the obstacle, it uses the obstacle coordinates as defined by the user to define the angle of inclusion. This angle will be smaller if more points are defined, and so the reflected energy will be less for the computational grid point. This problem becomes smaller if the computational grid points are closer to the obstacle.

   So the advise is to define obstacles with the least amount of points possible.

2. In case of sharp angles in the obstacles, it is very likely that there are more than one crossing between two computational grid points. In this case SWAN does not give correct reflection results.
   - Avoid sharp angles in the obstacle definition.
   - If necessary, put corner point of the sharp edge exactly on line between two computational grid points, but not exactly on the grid point.
3. At the boundaries of the computational area, the reflected spectrum is not taken into account. This can only be resolved by a different treatment of the boundaries in the program. Until this time, it is recommended to place obstacles at the inner area of the computational grid, not at or through the boundaries.

The computation of transmission and reflection is problematic if an obstacle runs exactly through one or more grid points of the computational structured grid; SWAN will move the obstacle over a small distance (0.01 of the mesh size) if this occurs. Note that this will not be done in case of unstructured grids.

The reflection results are incorrect if more than one obstacle crosses the same grid line between two neighbouring grid points. SWAN is not able to detect this, so the user must check if his model fulfills this condition.

`TRANSM` with this option the user indicates that the transmission coefficient is a constant.
`[trcoef]` constant transmission coefficient, formulated in terms of wave height, i.e. ratio of transmitted significant wave height over incoming significant wave height. Default: `[trcoef]`=0.0 (no transmission = complete blockage).
`DAM` with this option the user indicates that the transmission coefficient depends on the incident wave conditions at the obstacle and on the obstacle height (which may be submerged).
`GODA` with this option the user indicates to use the Goda/Seelig formula (1979) for computing transmission coefficient.
`[hgt]` the elevation of the top of the obstacle above reference level (same reference level as for bottom etc.); use a negative value if the top is below that reference level. If this command is used, this value is required.
`[alpha]` coefficient determining the transmission coefficient for Goda's transmission formula. Default: `[alpha]`=2.6.
`[beta]` another coefficient determining the transmission coefficient for Goda's transmission formula. Default: `[beta]`=0.15.
`DANGREMOND` with this option the user indicates to use the d'Angremond/Van der Meer formula (1996) for computing transmission coefficient.
`[hgt]` the elevation of the top of the obstacle above reference level (same reference level as for bottom etc.); use a negative value if the top is below that reference level. If this command is used, this value is required.

`[slope]` the slope of the obstacle (in degrees). If this command is used, this value is required.

`[Bk]` the crest width of the obstacle. If this command is used, this value is required.

`REFL` if this keyword is present the obstacle will reflect wave energy (possibly in combination with transmission). Reflections will be computed only if the spectral directions cover the full 360°, i.e. if in the command `CGRID` the option `CIRCLE` is activated.

`[reflc]` constant reflection coefficient, formulated in terms of wave height, i.e. ratio of reflected significant wave height over incoming significant wave height.
Restriction: $0 \leq$ `[reflc]` ≤ 1.
Default: `[reflc]`=1, if the keyword REFL is present.
NOTE: the program checks if the criterion $0 \leq$ `[reflc]`^{2}+`[trcoef]`$^2 \leq 1$ is fulfilled.

`RSPEC` indicates specular reflection which is the default. The angle of reflection equals the angle of incidence.

`RDIFF` indicates diffuse reflection, i.e. specular reflection where incident waves are scattered over reflected direction.

`[pown]` each incoming direction θ is scattered over reflected direction θ_{refl} according to $\cos^{[\texttt{pown}]}(\theta - \theta_{\text{refl}})$. The parameter `[pown]` indicates the width of the redistribution function.
Default: `[pown]` = 1.

`LINE` with this required keyword the user defines the location of the obstacle(s).

`[xp], [yp]` coordinates of a corner point of the line that defines the location of the obstacle(s) (in problem coordinates):
if Cartesian coordinates are used in m or
if spherical coordinates are used in degrees (see command `COORD`).
At least two corner points must be provided.

`SETUP [supcor]`

CANNOT BE USED IN CASE OF UNSTRUCTURED GRIDS.

If this optional command is given, the wave-induced set-up is computed and accounted for in the wave computations (during the computation it is added to the depth that is obtained from the `READ BOTTOM` and `READ WLEVEL` commands). This approximation in SWAN can only be applied to open coast (unlimited supply of water from outside the domain, e.g. nearshore coasts and estuaries) in contrast to closed basin, e.g. lakes, where this option should not be used. Note that set-up is not computed correctly with spherical coordinates. Note that set-up is not supported in case of parallel runs using MPI and also not tested with OpenMP!

`[supcor]` by default the wave-induced set-up is computed with a constant added such that the

set-up is zero in the deepest point in the computational grid. The user can modify this constant by the value of `[supcor]`. The user can thus impose a set-up in any one point (and only one) in the computational grid by first running SWAN, then reading the set-up in that point and adding or subtracting the required value of `[supcor]` (in m; positive if the set-up has to rise).
Default: `[supcor]`=0.

`DIFFRACtion [idiffr] [smpar] [smnum] [cgmod]`

CANNOT BE USED IN CASE OF UNSTRUCTURED GRIDS.

If this optional command is given, the diffraction is included in the wave computation. But the diffraction approximation in SWAN does not properly handle diffraction in harbours or in front of reflecting obstacles (see Scientific/Technical documentation). Behind breakwaters with a down-wave beach, the SWAN results seem reasonable. The spatial resolution near (the tip of) the diffraction obstacle should be 1/5 to 1/10 of the dominant wave length.

Without extra measures, the diffraction computations with SWAN often converge poorly or not at all. Two measures can be taken:

1. (RECOMMENDED) The user can request under-relaxation. See command `NUMERIC` parameter `[alpha]` and Scientific/Technical documentation (Eq. (3.31)). Very limited experience suggests `[alpha]` = 0.01.

2. Alternatively, the user can request smoothing of the wave field for the computation of the diffraction parameter (the wave field remains intact for all other computations and output). This is done with a repeated convolution filtering. The mother filter is

$$E^n_{i,j} = E^{n-1}_{i,j} - a\left[E_{i-1,j} + E_{i,j-1} - 4E_{i,j} + E_{i+1,j} + E_{i,j+1}\right]^{n-1}$$

   For $a = 0.2$ (recommended), the final width of the filter is $\varepsilon_x = \frac{1}{2}\sqrt{3n}\Delta x$ (in $x-$direction and similarly in $y-$direction) and n is the number of repetitions (see Scientific/Technical documentation, Eq. (2.100)).

`[idiffr]` indicates the use of diffraction. If `[idiffr]`=0 then no diffraction is taken into account.
Default: `[idiffr]`=1.

`[smpar]` smoothing parameter for the calculation of $\nabla \cdot \sqrt{E_{\rm tot}}$. During every smoothing step all grid points exchange `[smpar]` times the energy with their neighbours. Note that `[smpar]` is parameter a in the above text.
Default: `[smpar]` = 0.

`[smnum]` number of smoothing steps (n in the above text). For $a = 0.2$, it should be

approximately equal to $\lfloor \frac{4\varepsilon_x^2}{3\Delta x^2} \rfloor$.
Default: `[smnum]` = 0.

`[cgmod]` adaption of propagation velocities in geographic space due to diffraction.
If `[cgmod]`=0 then no adaption.
Default: `[cgmod]`=1.

```
     |  WINDGrowth |
     |             |
     |  QUADrupl   |
     |             |
     |  WCAPping   |
OFF <               >
     |  BREaking   |
     |             |
     |  REFrac     |
     |             |
     |  FSHift     |
     |             |
     |  BNDCHK     |
```

With this optional command the user can change the default inclusion of various physical processes (e.g. for research purposes). This command is not recommended for operational use.

`WINDGROWTH` switches off wind growth (in commands `GEN1`, `GEN2` and `GEN3`).
`QUADRUPL` switches off quadruplet wave-wave interactions (in command `GEN3`).
`WCAPPING` switches off whitecapping (in command `GEN3`).
`BREAKING` switches off depth-induced breaking dissipation. Caution: wave heights may diverge in very shallow water.
`REFRAC` switches off refraction (action transport in $\theta-$direction).
`FSHIFT` switches off frequency shifting in frequency space (action transport in $\sigma-$space).
`BNDCHK` switches off the checking of the difference between imposed and computed significant wave height at the boundary of the computational grid (see also command `SET`).

4.5.5 Numerics

```
          |    BSBT
PROP      <                        | Sec |
          |    GSE  [waveage] <    MIn   >
          |                        |  HR |
          |                        | DAy |
```

Command to choose:

- the BSBT scheme (stationary and nonstationary) instead of the default S&L scheme (in case of nonstationary cases) or the default SORDUP scheme (in case of stationary cases) or
- the wave age (for the default nonstationary S&L scheme).

`BSBT` the BSBT scheme will be used in the computations.

`GSE` garden-sprinkler-effect is to be counteracted in the S&L propagation scheme (default for nonstationary computations) by adding a diffusion term to the basic equation. This may affect the numerical stability of SWAN (see Scientific/ Technical documentation).

`[waveage]` the time interval used to determine the diffusion which counteracts the so-called garden-sprinkler effect. The default value of `[waveage]` is zero, i.e. no added diffusion. The value of `[waveage]` should correspond to the travel time of the waves over the computational region.

Notes:

- All schemes (BSBT, SORDUP and S&L) can be used in combination with curvilinear grids. With the higher order schemes (S&L and SORDUP) it is important to use a gradually varying grid, otherwise there may be a severe loss of accuracy. If sharp transitions in the grid cannot be avoided it is safer to use the BSBT scheme.
- In the computation with unstructured meshes, a first order upwind scheme will be employed. This scheme is very robust but rather diffusive. This may only be significant for the case when swell waves propagate over relative large distances (in the order of thousands of kilometers) within the model domain.

```
            | -> ACCUR [drel] [dhoval] [dtoval] [npnts]
NUMeric ( <                                                    &
            |    STOPC [dabs] [drel] [curvat] [npnts]

                     | -> STAT [mxitst] [alfa] |
```

```
                  <                              >  [limiter]       ) &
                   |    NONSTAT [mxitns]      |

          ( DIRimpl [cdd] [cdlim]                                   ) &

          ( SIGIMpl [css] [eps2] [outp] [niter]                     ) &

          ( SETUP [eps2] [outp] [niter]                             )
```

With this optional command the user can influence some of the numerical properties of SWAN.

`ACCUR` With this option the user can influence the criterion for terminating the iterative procedure in the SWAN computations (both stationary and nonstationary mode). SWAN stops the iterations **if**:

a) the change in the local significant wave height (H_s) from one iteration to the next is less than
   1) fraction `[drel]` of that height **or**
   2) fraction `[dhoval]` of the average significant wave height (average over all wet grid points)

and

b) the change in the local mean wave period (T_{m01}) from one iteration to the next is less than
   1) fraction `[drel]` of that period **or**
   2) fraction `[dtoval]` of the average mean wave period (average over all wet grid points)

and

c) conditions a) and b) are fulfilled in more than fraction `[npnts]`% of all wet grid points.

DEFAULT IN CASE OF STRUCTURED GRIDS.

`STOPC` With this alternative option the user can influence the criterion for terminating the iterative procedure in the SWAN computations (both stationary and nonstationary). The criterion make use of the second derivative, or curvature, of the iteration curve of both the significant wave height and the mean period. As the solution of a simulation approaches full convergence, the curvature of the iteration curve will tend to zero. SWAN stops the process **if** the absolute change in both H_s and T_{m01} from one iteration to the next is less than `[dabs]` **or** the relative change in H_s and T_{m01} from one iteration to the next is less than `[drel]` **and** the curvature of the iteration curve of H_s normalized with H_s and that of T_{m01} normalized with T_{m01} is less than `[curvat]`.

DEFAULT IN CASE OF UNSTRUCTURED GRIDS.

`[dabs]` Default: `[dabs]` = 0.00 [−] in case of structured grids; `[dabs]` = 0.005 [−] in case of unstructured grids.

`[drel]` Default: `[drel]` = 0.02 [−] in case of `ACCUR`; `[drel]` = 0.01 [−] in case of `STOPC`.
`[dhoval]` Default: `[dhoval]` = 0.02 [−]
`[dtoval]` Default: `[dtoval]` = 0.02 [−]
`[curvat]` Default: `[curvat]` = 0.005 [−]
`[npnts]` Default: `[npnts]` = 98. [−] in case of structured grids; `[npnts]` = 99.5 [−] in case of unstructured grids.
`STAT` indicates the use of parameters in a stationary computation.
`[mxitst]` the maximum number of iterations for stationary computations. The computation stops when this number is exceeded. Default: `[mxitst]` = 50. Note that `[mxitst]` can be set to 0 if one wants to check the input to the model without making computations.
`[alfa]` proportionality constant used in the frequency-dependent under-relaxation technique. Based on experiences, a suggestion for this parameter is `[alfa]` = 0.01. In case of diffraction computations, the use of this parameter is recommended. Default: `[alfa]` = 0.00. NOT MEANINGFUL FOR NONSTATIONARY COMPUTATIONS.
`NONSTAT` indicates the use of parameters in a nonstationary computation.
`[mxitns]` the maximum number of iterations per time step for nonstationary computations. The computation moves to the next time step when this number is exceeded. Default: `[mxitns]` = 1. Note that `[mxitns]` can be set to 0 if one wants to check the input to the model without making computations.
`[limiter]` determines, in both stationary and nonstationary runs, the maximum change per iteration of the energy density per spectral (σ,θ)-bin, given in terms of a fraction of the omni-directional Phillips level (see Scientific/ Technical documentation). Default: `[limiter]` = 0.1.
`DIRIMPL` this option is used to influence the numerical scheme for refraction.
`[cdd]` A value of `[cdd]`=0 corresponds to a central scheme and has the largest accuracy (diffusion $\approx$ 0) but the computation may more easily generate spurious fluctuations. A value of `[cdd]`=1. corresponds to an first order upwind scheme and it is more diffusive and therefore preferable if (strong) gradients in depth or current are present. Default: `[cdd]` = 0.5.
`[cdlim]` If the spatial discretization of the bathymetry or the flow currents is too coarse, the waves may turn too far (more than 90 degrees, say) over one spatial grid step. The computational results will then be very inaccurate. In such a case SWAN can limit the maximum turning of the waves over one spatial grid to 90 degrees to obtain robust (but not necessarily correct results).
`[cdlim]` < 0 then no limiter is used (this is default)
`[cdlim]` $= 0$ refraction is off (same effect as command `OFF REFRAC`)
`[cdlim]` $= 4$ waves turning limited to about 90^o over one spatial grid

step.

`SIGIMPL` controls the accuracy of computing the frequency shifting and the stopping criterion and amount of output for the SIP solver (used in the computations in the presence of currents or time varying depth).

`SETUP` controls the stopping criterion and amount of output for the SOR solver in the computation of the wave-induced set-up.

`[css]` A value of `[css]`=0 corresponds to a central scheme and has the largest accuracy (diffusion ≈ 0) but the computation may more easily generate spurious fluctuations. A value of `[css]`=1. corresponds to an first order upwind scheme and it is more diffusive and therefore preferable if (strong) gradients in depth or current are present.
Default: `[css]` = 0.5.

`[eps2]` Relative stopping criterion to terminate the linear solver (SIP or SOR). The criterion for the SIP solver is based on $\|A\vec{N}_k - \vec{b}\|_2 \leq$ `[eps2]` $\|\vec{b}\|_2$ where A is a matrix, $\vec{N}$ is the action density vector, $\vec{b}$ is the right hand vector and k is the iteration number.
The criterion for the SOR solver is based on $\|\overline{\eta}_{k+1} - \overline{\eta}_k\|_\infty \leq$ `[eps2]` where $\overline{\eta}$ is the set-up.
Default: `[eps2]` = 1.e-4 in case of SIP and `[eps2]` = 1.e-6 in case of SOR.

`[outp]` output for the iterative solver:
0 = no output
1 = additional information about the iteration process is written to the `PRINT` file
2 = gives a maximal amount of output concerning the iteration process
3 = summary of the iteration process
Default: `[outp]` = 0.

`[niter]` maximum number of iterations for the linear solver.
Default: `[niter]` = 20 in case of SIP and `[niter]` = 1000 in case of SOR.

4.6 Output

There are two categories of output commands:

1. Locations
   commands defining sets of output locations at which the user requires output. Each set is indicated with a name (`'sname'` in this manual) which must be unique and not more than 8 characters long.

   Types of sets of output points:

   `FRAME` to define a set of output locations on a regular grid
   `GROUP` to define a set of output locations on a regular or curvi-linear grid
   `CURVE` to define a set of output locations along a curve

RAY	to define a set of output locations along a depth or bottom contour line (with ISOLINE)
ISOLINE	to define a set of output locations along a depth- or bottom contour line (with RAY)
POINTS	to define a set of isolated output locations
NGRID	to define a set of output locations for a nested grid to be used in a subsequent SWAN run

Commands FRAME, GROUP, RAY, ISOLINE and NGRID cannot be used in 1D-MODE and command GROUP cannot be used in case of unstructured meshes. If one gives one name for two sets of output locations, the first set is lost (first in the sequence in the command file). Two special names: BOTTGRID and COMPGRID are reserved for use by SWAN (see below). The user may not define sets with these names.

2. Write / plot
commands defining data file output (write) at the above defined set(s) of output locations:

BLOCK	write spatial distributions (only for FRAMEs and GROUPs)
TABLE	write output for (set of) output location(s)
SPECOUT	write to data file the variance / energy (see command SET) density spectrum for (set of) output location(s)
NESTOUT	write to data file two-dimensional action density spectra (relative frequency) along the boundary of a nested grid (see command NGRID) to be used in a subsequent SWAN run.

Commands BLOCK and NESTOUT cannot be used in 1D-MODE.

4.6.1 Output locations

```
FRAme 'sname' [xpfr] [ypfr] [alpfr] [xlenfr] [ylenfr] [mxfr] [myfr])
```

CANNOT BE USED IN 1D-MODE.

With this optional command the user defines output on a rectangular, uniform grid in a regular frame.

If the set of output locations is identical to a part of the computational grid, then the user can use the alternative command GROUP.

`'sname'`	name of the frame defined by this command
`[xpfr]`	$x-$coordinate of the origin of the frame in problem coordinates if Cartesian coordinates are used in m if spherical coordinates are used in degrees (see command `COORD`)
`[ypfr]`	$y-$coordinate of the origin of the frame in problem coordinates if Cartesian coordinates are used in m if spherical coordinates are used in degrees (see command `COORD`)
`[alpfr]`	direction of the $x-$axis of the frame (in degrees, Cartesian convention; must be 0 in case of spherical coordinates)
`[xlenfr]`	length of the frame in $x-$direction if Cartesian coordinates are used in m if spherical coordinates are used in degrees (see command `COORD`)
`[ylenfr]`	length of the frame in $y-$direction if Cartesian coordinates are used in m if spherical coordinates are used in degrees (see command `COORD`)
`[mxfr]`	number of meshes in $x-$direction of the rectangular grid in the frame (one less than the number of grid points in this direction) Default: `[mxfr]`=20
`[myfr]`	number of meshes in $y-$direction of the rectangular grid in the frame (one less than the number of grid points in this direction) Default: `[myfr]`=20

Some output may be required on a frame that is identical with the input (bottom/current) grid or with the computational grid (e.g. for test purposes or to avoid interpolation errors in the output). These frames need not be defined by the user with this command `FRAME`; the frames are always generated automatically by SWAN under the names `'sname' = 'BOTTGRID'` (for the bottom/current grid) and `'sname' = 'COMPGRID'` (for the computational grid).

`GROUP 'sname' SUBGrid [ix1] [ix2] [iy1] [iy2]`

CANNOT BE USED IN 1D-MODE AND IN CASE OF UNSTRUCTURED GRIDS.

With this optional command the user defines a group of output locations on a rectangular or curvi-linear grid that is identical with (part of) the computational grid (recti-linear or curvi-linear). Such a group may be convenient for the user to obtain output that is not affected by interpolation errors (which would occur when an output grid is used that is not identical with (part of) the computational grid).

Command `CGRID` should precede this command `GROUP`.

The subgrid contains those points (ix,iy) of the computational grid for which:

[ix1] ≤ ix ≤ [ix2] and [iy1] ≤ iy ≤ [iy2]

For convenience the size of the group, the corner coordinates and the angle with the problem coordinate system are written to PRINT file. The origin of the computational grid is (ix=0,iy=0)!

'sname'	name of the set of output locations defined by this command
[ix1]	lowest grid index of subgrid in terms of computational grid in ix-direction
[iy1]	lowest grid index of subgrid in terms of computational grid in iy-direction
[ix2]	highest grid index of subgrid in terms of computational grid in ix-direction
[iy2]	highest grid index of subgrid in terms of computational grid in iy-direction

Limitations:
[ix1]≥0, [ix2]≤[mxc], [iy1]≥0, [iy2]≤[myc] ([mxc] and [myc] as defined in the command CGRID).

CURve 'sname' [xp1] [yp1] < [int] [xp] [yp] >

With this optional command the user defines output along a curved line. Actually this curve is a broken line, defined by the user with its corner points. The values of the output quantities along the curve are interpolated from the computational grid. This command may be used more than once to define more curves.

'sname'	name of the curve
[xp],[yp1]	problem coordinates of the first point of the curve if Cartesian coordinates are used in m if spherical coordinates are used in degrees (see command COORD)
[int]	SWAN will generate output at [int]−1 equidistant locations between two subsequent corner points of the curve (including the two corner points of the curve)
[xp],[yp]	problem coordinates of a corner point of the curve. Repeat the group [int] [xp] [yp] in proper order if there are more corner points are on the curve.

RAY 'rname' [xp1] [yp1] [xq1] [yq1] < [int] [xp] [yp] [xq] [yq] >

CANNOT BE USED IN 1D-MODE.

With this optional command the user provides SWAN with information to determine output locations along the depth contour line(s) defined subsequently in command ISOLINE

(see below).

The locations are determined by SWAN as the intersections of the depth contour line(s) and the set of straight rays defined in this command RAY. These rays are characterized by a set of master rays defined by their start and end positions ([xp],[yp]) and ([xq],[yq]). Between each pair of sequential master rays thus defined SWAN generates [int]−1 intermediate rays by linear interpolation of the start and end positions.

Note that the rays thus defined have nothing in common with wave rays (e.g. as obtained from conventional refraction computations).

'rname'	name of the set of rays defined by this command.
[xp1],[yp1],	problem coordinates of the begin and end points of the first master ray
[xq1],[yq1]	if Cartesian coordinates are used in m if spherical coordinates are used in degrees (see command COORD)
[int]	number of subdivisions between the previous master ray and the following master ray defined by the following data (number of subdivisions is one more than the number of interpolated rays)
[xp],[yp],	problem coordinates of the begin and end points of each subsequent master ray
[xq],[yq]	if Cartesian coordinates are used in m if spherical coordinates are used in degrees (see command COORD)

```
                              | -> DEPth   |
ISOline  'sname'  'rname'  <               >  [dep]
                              |    BOTtom  |
```

CANNOT BE USED IN 1D-MODE.

With this optional command the user defines a set of output locations along one depth or bottom level contour line (in combination with command RAY).

'sname'	name of the set of output locations defined by this command
'rname'	name of the set of rays (as defined in command RAY)
[dep]	the depth (in m) of the depth contour line along which output locations are generated by SWAN. If the keyword DEPTH appears in front of the value the true depth is used, if the keyword BOTTOM appears the water level is ignored, i.e. the depth with respect to datum level is used.

The set of output locations along the depth contour lines created with this command is of the type CURVE.

```
                    |  < [xp] [yp] >  |
POINts   'sname'  <                    >
                    |  FILE  'fname'  |
```

With this optional command the user defines a set of individual output locations (points). The coordinates of these points are given in the command itself or read from a file (option FILE).

'sname'	name of the points
[xp],[yp]	problem coordinates of one output location if Cartesian coordinates are used in m if spherical coordinates are used in degrees (see command COORD)

```
                | [xpn] [ypn] [alpn] [xlenn] [ylenn] [mxn] [myn]
                |
NGRid 'sname' <                  | -> TRIAngle |
                | UNSTRUCtured  <               > 'fname'
                |                 |    EASYmesh |
```

CANNOT BE USED IN 1D-MODE.

If the user wishes to carry out nested SWAN run(s), a separate coarse-grid SWAN run is required. With this optional command NGRID, the user defines in the present coarse-grid run, a set of output locations along the boundary of the subsequent nested computational grid. The set of output locations thus defined is of the type NGRID.

Command NESTOUT is required after this command NGRID to generate some data for the (subsequent) nested run (not with command BLOCK because a set of locations of the type NGRID does represent an outline and not a geographic region).

'sname'	name of the set of output locations along the boundaries of the following nested computational grid defined by this command
[xpn]	geographic location of the origin of the computational grid of this coarse-grid run in the problem coordinate system ($x-$coordinate) if Cartesian coordinates are used in m if spherical coordinates are used in degrees (see command COORD)
[ypn]	geographic location of the origin of the computational grid of this coarse-grid run in the problem coordinate system ($y-$coordinate) if Cartesian coordinates are used in m if spherical coordinates are used in degrees (see command COORD)

`[alpn]` direction of the positive $x-$axis of the computational grid of this coarse-grid run (in degrees, Cartesian convention).

`[xlenn]` length in the $x-$direction of the nested grid
if Cartesian coordinates are used in m
if spherical coordinates are used in degrees (see command `COORD`)

`[ylenn]` length in the $y-$direction of the nested grid
if Cartesian coordinates are used in m
if spherical coordinates are used in degrees (see command `COORD`)

`[mxn]` number of meshes of the output grid in the $x-$direction of this grid (this number is one less than the number of grid points in this direction!). `[mxn]` does not have to be equal to the number of meshes in the nested computation; SWAN will interpolate the required information.
Default: `[mxn]` is chosen such that the mesh size of the output grid is (roughly) equal to the mesh size of the coarse grid, but at least 1.

`[myn]` number of meshes of the output grid in the $y-$direction of this grid (this number is one less than the number of grid points in this direction!). `[myn]` does not have to be equal to the number of meshes in the nested computation; SWAN will interpolate the required information.
Default: `[myn]` is chosen such that the mesh size of the output grid is (roughly) equal to the mesh size of the coarse grid, but at least 1.

`UNSTRUCTURE` with this option the user indicates that the subsequent nested grid is an unstructured one. Only grids generated by Triangle and Easymesh are supported by SWAN.

`TRIANGLE` the necessary grid information is read from two files as produced by Triangle. The `.node` and `.ele` files are required. The basename of these files must be indicated with parameter `'fname'`.

`EASYMESH` the necessary grid information is read from two files as produced by Easymesh. The `.n` and `.e` files are required. The basename of these files must be indicated with parameter `'fname'`.

`'fname'` basename of the required files, i.e. without extension.

4.6.2 Write or plot computed quantities

For definitions of output parameters, see Appendix A.

WARNING:
When integral parameters are computed by the user from the output spectrum of SWAN, differences with the SWAN-computed parameters may occur. The reasons are:

- SWAN accepts at the boundaries of the computational grid only the user-imposed incoming wave components and it replaces the user-imposed outgoing wave components with computed components (propagating to the boundary from the interior region). Note that this will not happen in case of triangular meshes.

- during the computation of the parameters, SWAN adds an analytical (diagnostic) high-frequency tail to the discrete spectrum.
- SWAN has an option to only compute within a pre-set directional sector (pre-set by the user). Wave components outside this sector are totally ignored by SWAN (no additions or replacements).

This is particularly relevant along the boundaries of SWAN where the user-imposed integral parameters (boundary conditions) may differ from the SWAN-computed parameters. The user is informed by means of a warning in the output (`PRINT` file) when the computed significant wave height differs more than 10%, say, from the user-imposed significant wave height (command `BOUNDSPEC`). The actual value of this difference can be set by the user (see the `SET` command; Section 4.4).

```
          | ..........|
QUANTity <             > 'short' 'long' [lexp] [hexp]  [excv]                  &
          | ..........|

     [power]               (For output quantities PER, RPER and WLEN)           &

     [ref]                 (For output quantity TSEC)                           &

     [fswell]              (For output quantity HSWELL)                         &

     [fmin] [fmax]         (For all integral parameters, like HS, (R)TM01 ...)  &

      |-> PROBLEMcoord |
     <                  >  (For directions (DIR, TDIR, PDIR)
      |   FRAME         |    and vectors (FORCE, WIND, VEL, TRANSP))
```

With this command the user can influence:

- the naming of output quantities,
- the accuracy of writing output quantities,
- the definition of some output quantities and
- reference direction for vectors.

```
 |...|
<     >
 |...|
```
the output parameters are the same as given in command `BLOCK`.

`'short'` user preferred short name of the output quantity (e.g. the name appearing in

the heading of a table written by SWAN). If this option is not used, SWAN will use a realistic name.

`'long'` long name of the output quantity (e.g. the name appearing in the heading of a block output written by SWAN). If this option is not used, SWAN will use a realistic name.

`[lexp]` lowest expected value of the output quantity.

`[hexp]` highest expected value of the output quantity; the highest expected value is used by SWAN to determine the number of decimals in a table with heading. So the `QUANTITY` command can be used in case the default number of decimals in a table is unsatisfactory.

`[excv]` in case there is no valid value (e.g. wave height in a dry point) this exception value of the output quantity is written in a table or block output.

The following data are accepted only in combination with some specific output quantities.

`[power]` power p appearing in the definition of `PER`, `RPER` and `WLEN` (see Appendix A). Note that the value for `[power]` given for `PER` affects also the value of `RPER`; the power for `WLEN` is independent of that of `PER` or `RPER`.
Default: `[power]`=1.

`[ref]` reference time used for the quantity `TSEC`.
Default value: starting time of the first computation, except in cases where this is later than the time of the earliest input. In these cases, the time of the earliest input is used.

`[fswell]` upper limit of frequency range used for computing the quantity `HSWELL` (see Appendix A).
Default: `[fswell]` = 0.1 Hz.

`[fmin]` lower limit of frequency range used for computing integral parameters.
Default: `[fmin]` = 0.0 Hz.

`[fmax]` upper limit of frequency range used for computing integral parameters.
Default: `[fmax]` = 1000.0 Hz (acts as infinity).

`PROBLEMCOORD` − vector components are relative to the $x-$ and $y-$axes of the problem coordinate system (see command `COORD`)
− directions are counterclockwise relative to the positive $x-$axis of the problem coordinate system if Cartesian direction convention is used (see command `SET`)
− directions are relative to North (clockwise) if Nautical direction convention is used (see command `SET`)

`FRAME` If output is requested on sets created by command `FRAME` or automatically (`COMPGRID` or `BOTTGRID`)
− vector components are relative to the $x-$ and $y-$axes of the frame coordinate system (see command `COORD`)

− directions are counterclockwise relative to the positive $x-$axis of the frame coordinate system if Cartesian direction convention is used (see command SET)
− directions are relative to North (clockwise) if Nautical direction convention is used (see command SET)

Examples:

`QUANTITY Xp hexp=100.`	for simulations of lab. experiments
`QUANTITY HS TM01 RTMM10 excv=-9.`	to change the exception value for H_s, T_{m01} and relative T_{m-10}
`QUANTITY HS TM02 FSPR fmin=0.03 fmax=0.5`	to compute H_s, T_{m02} and frequency spreading by means of integration over $f \in [0.03, 0.5]$
`QUANTITY Hswell fswell=0.08`	to change the value of `[fswell]`
`QUANTITY Per short='Tm-1,0' power=0.`	to redefine average wave period
`QUANTITY Transp Force Frame`	to obtain vector components and direction with respect to the frame

---

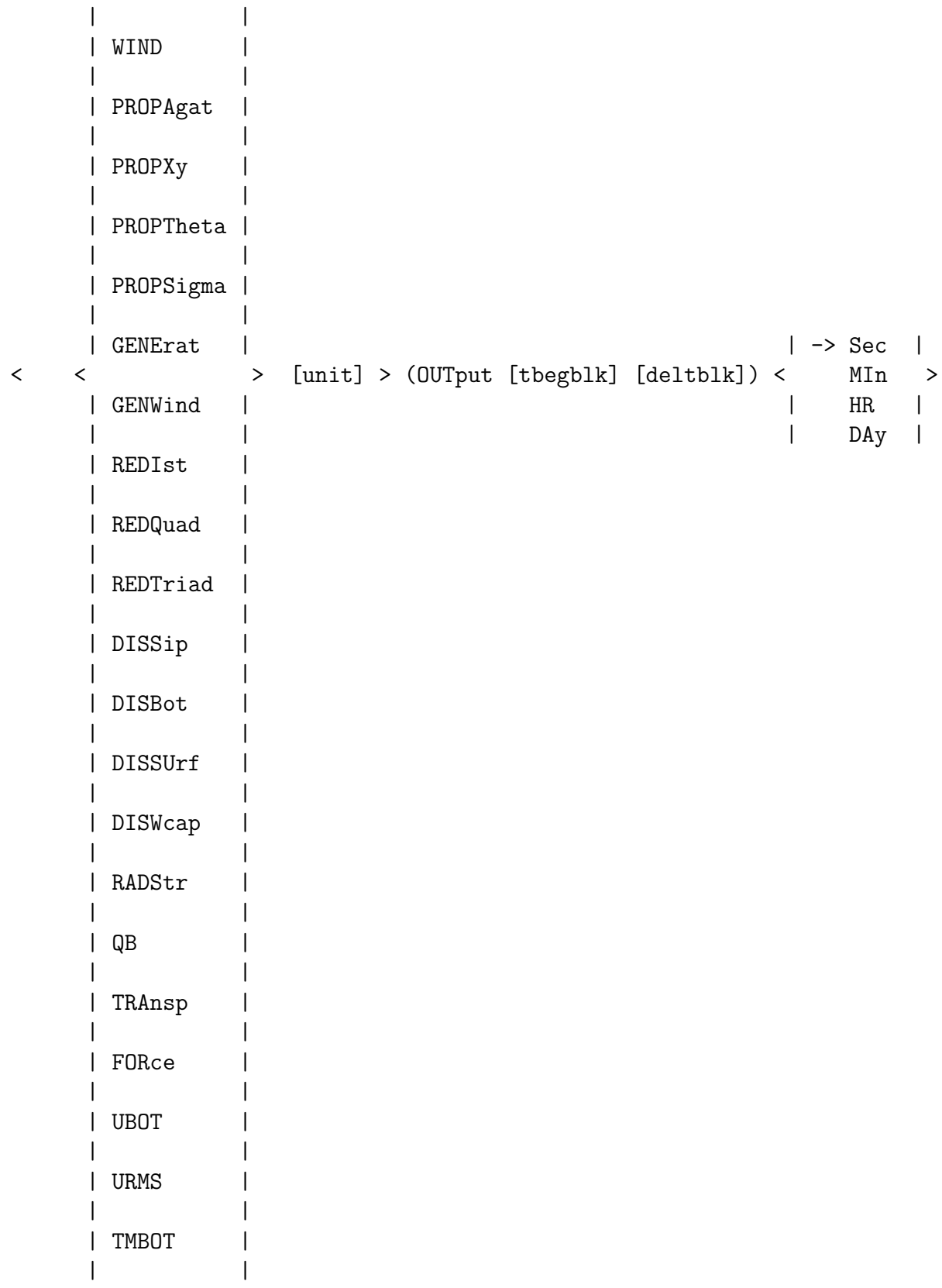
`OUTPut OPTIons 'comment' (TABle [field]) (BLOck [ndec] [len]) (SPEC [ndec])`

This command enables the user to influence the format of block, table and spectral output.

`comment`	a comment character; is used in comment lines in the output Default: `comment` = %
`field`	length of one data field in a table. Minimum is 8 and maximum is 16. Default: `field` = 12
`ndec`	number of decimals in block (if appearing after keyword `BLOCK`) or 2D spectral output (if appearing after keyword `SPEC`). Maximum is 9. Default: `ndec` = 4 (in both block and spectral outputs)
`len`	number of data on one line of block output. Maximum is 9999. Default: `len` = 6

```
                | -> HEADer   |
BLOck  'sname'  <               > 'fname' (LAYout [idla])
                |    NOHEADer |
```

HSign
HSWEll
DIR
PDIR
TDIR
TM01
RTM01
RTP
TPS
PER
RPER
TMM10
RTMM10
TM02
FSPR
DSPR
QP
DEPth
WATLev
BOTLev
VEL
FRCoef


WIND
PROPAgat
PROPXy
PROPTheta
PROPSigma
GENErat
GENWind
REDIst
REDQuad
REDTriad
DISSip
DISBot
DISSUrf
DISWcap
RADStr
QB
TRAnsp
FORce
UBOT
URMS
TMBOT
[unit]
(OUTput [tbegblk] [deltblk])
-> Sec
MIn
HR
DAy

```
| WLENgth   |
|           |
| STEEpness |
|           |
| BFI       |
|           |
| DHSign    |
|           |
| DRTM01    |
|           |
| LEAK      |
|           |
| TIME      |
|           |
| TSEC      |
|           |
| XP        |
|           |
| YP        |
|           |
| DIST      |
|           |
| SETUP     |
```

CANNOT BE USED IN 1D-MODE.

With this optional command the user indicates that one or more spatial distributions should be written to a file.

`'sname'` name of frame or group (see commands `FRAME` or `GROUP`)

`HEADER` with this option the user indicates that the output should be written to a file with header lines. The text of the header indicates run identification (see command `PROJECT`), time, frame name or group name (`'sname'`), variable and unit. The number of header lines is 8.
Note: the numerical values in the file are in the units indicated in the header.

`NOHEADER` with this option the user indicates that the output should be written to a file without header lines.

`'fname'` name of the data file where the output is to be written to. Default for option `HEADER` is the `PRINT` file. In case of `NOHEADER` the filename is required. Note that when the extension is '**.mat**', a binary MATLAB file will be generated automatically. This file requires less space on your computer and can be loaded in MATLAB much faster than the ASCII-file.
Binary MATLAB files are particularly useful for the computation with

unstructured grids. Some MATLAB scripts are provided with the SWAN source code that can be used to plot wave parameters as maps in a quick manner.

`LAY-OUT` with this option the user can prescribe the lay-out of the output to file with the value of `[idla]`.

`[idla]` see command `READINP` (options are: `[idla]`=1, 3, 4). Option 4 is recommended for postprocessing by MATLAB, however, in case of a generated binary MATLAB file option 3 is recommended.
Default: `[idla]` = 1.
ONLY MEANT FOR STRUCTURED GRIDS.

For definitions of the output quantities, see Appendix A.

Note that the wave parameters in the output of SWAN are computed from the wave spectrum over the prognostic part of the spectrum with the diagnostic tail added. Their value may therefore deviate slightly from values computed by the user from the output spectrum of SWAN which does not contain the diagnostic tail.

`HSIGN` significant wave height (in m).

`HSWELL` swell wave height (in m).

`DIR` mean wave direction (Cartesian or Nautical convention, see command `SET`).
For Cartesian convention: relative to $x-$axis of the problem coordinate system (counterclockwise); possible exception: in the case of output with `BLOCK` command in combination with command `FRAME`, see command `QUANTITY`.

`PDIR` peak wave direction in degrees.
For Cartesian convention: relative to $x-$axis of the problem coordinate system (counterclockwise); possible exception: in the case of output with `BLOCK` command in combination with command `FRAME`, see command `QUANTITY`.

`TDIR` direction of energy transport in degrees.
For Cartesian convention: relative to $x-$axis of the problem coordinate system (counterclockwise); possible exception: in the case of output with `BLOCK` command in combination with command `FRAME`, see command `QUANTITY`.

`TM01` mean absolute wave period (in s).

`RTM01` mean relative wave period (in s).

`RTP` peak period (in s) of the variance density spectrum (relative frequency spectrum).

`TPS` 'smoothed' peak period (in s).

`PER` mean absolute wave period (in s).

`RPER` mean relative wave period (in s).

`TMM10` mean absolute wave period (in s).

`RTMM10` mean relative wave period (in s).

`TM02` mean absolute zero-crossing period (in s).

`FSPR` the normalized width of the frequency spectrum.

`DSPR` directional spreading of the waves (in degrees).

QP	peakedness of the wave spectrum (dimensionless).
DEPTH	water depth (in m) (not the bottom level!).
WATLEV	water level (in m). Output is in both active and non-active points. Note: exception value for water levels must be given! (See command INPGRID WLEVEL EXCEPTION).
BOTLEV	bottom level (in m). Output is in both active and non-active points. Note: exception value for bottom levels must be given! (See command INPGRID BOTTOM EXCEPTION).
VEL	current velocity (vector; in m/s).
FRCOEF	friction coefficient (equal to [cfw] or [kn] in command FRICTION).
WIND	wind velocity (vector; in m/s).
PROPAGAT	total energy propagation (in W/m^2 or m^2/s, depending on command SET).
PROPXY	energy propagation in geographic space (in W/m^2 or m^2/s, depending on command SET).
PROPTHETA	energy propagation in theta space (in W/m^2 or m^2/s, depending on command SET).
PROPSIGMA	energy propagation in sigma space (in W/m^2 or m^2/s, depending on command SET).
GENERAT	total energy generation (in W/m^2 or m^2/s, depending on command SET).
GENWIND	energy generation due to wind (in W/m^2 or m^2/s, depending on command SET).
REDIST	total energy redistribution (in W/m^2 or m^2/s, depending on command SET).
REDQUAD	energy redistribution due to quadruplets (in W/m^2 or m^2/s, depending on command SET).
REDTRIAD	energy redistribution due to triads (in W/m^2 or m^2/s, depending on command SET).
DISSIP	total energy dissipation (in W/m^2 or m^2/s, depending on command SET).
DISBOT	energy dissipation due to bottom friction (in W/m^2 or m^2/s, depending on command SET).
DISSURF	energy dissipation due to surf breaking (in W/m^2 or m^2/s, depending on command SET).
DISWCAP	energy dissipation due to whitecapping (in W/m^2 or m^2/s, depending on command SET).
RADSTR	radiation stress (in W/m^2 or m^2/s, depending on command SET).
QB	fraction of breaking waves due to depth-induced breaking.
TRANSP	transport of energy (vector; in W/m or m^3/s, depending on command SET).

FORCE	wave-induced force per unit surface area (vector; in N/m^2).
UBOT	the rms-value of the maxima of the orbital velocity near the bottom (in m/s). Output only if command FRICTION is used. If one wants to output UBOT but friction is ignored in the computation, then one should use the command FRICTION with the value of the friction set to zero (FRICTION COLLINS 0).
URMS	the rms-value of the of the orbital velocity near the bottom (in m/s). If one wants to output URMS but friction is ignored in the computation, then one should use the command FRICTION with the value of the friction set to zero (FRICTION COLLINS 0).
TMBOT	the bottom wave period (in s).
WLEN	average wavelength (in m).
STEEPNESS	average wave steepness (dimensionless).
BFI	Benjamin-Feir index (dimensionless).
DHSIGN	the difference in significant wave height as computed in the last two iterations. This is not the difference between the computed values and the final limit of the iteration process, at most an indication of this difference.
DRTM01	the difference in average wave period (RTM01) as computed in the last two iterations. This is not the difference between the computed values and the final limit of the iteration process, at most an indication of this difference.
LEAK	numerical loss of energy equal to $c_\theta E(\omega,\theta)$ across boundaries θ_1 = [dir1] and θ_2 = [dir2] of a directional sector (see command CGRID).
TIME	Full date-time string as part of line used in TABLE only. Useful only in case of nonstationary computations.
TSEC	Time in seconds with respect to a reference time (see command QUANTITY). Useful only in case of nonstationary computations.
XP	user instructs SWAN to write the $x-$coordinate in the problem coordinate system of the output location.
YP	user instructs SWAN to write the $y-$coordinate in the problem coordinate system of the output location.
DIST	if output has been requested along a curve (see command CURVE) then the distance along the curve can be obtained with the command TABLE. DIST is the distance along the curve measured from the first point on the curve to the output location on the curve in meters (also in the case of spherical coordinates).
SETUP	Set-up due to waves (in m).
[unit]	this controls the scaling of output. The program divides computed values by [unit] before writing to file, so the user should multiply the written value by [unit] to obtain the proper value. Default: if HEADER is selected, value is written as a 5 position integer. SWAN takes [unit] such that the largest number occurring in the block can be printed. If NOHEADER is selected, values are printed in floating-point format, [unit] = 1.
OUTPUT	the user requests output at various times. If the user does not use this option, the

program will give `BLOCK` output for the last time step of the computation.

`[tbegblk]` begin time of the first field of the variable, the format is:

1	: ISO-notation	19870530.153000
2	: (as in HP compiler)	'30−May−87 15:30:00'
3	: (as in Lahey compiler)	05/30/87.15:30:00
4	:	15:30:00
5	:	87/05/30 15:30:00'
6	: as in WAM	8705301530

This format is installation dependent. See Implementation Manual or ask the person who installed SWAN on your computer. Default is ISO-notation.

`[deltblk]` time interval between fields, the unit is indicated in the next option:

SEC	unit seconds
MIN	unit minutes
HR	unit hours
DAY	unit days

```
                | -> HEADer   |
                |             |
TABle  'sname'  <     NOHEADer  >  'fname'                                  &
                |             |
                |    INDexed  |

             | ...      |                                        | -> Sec  |
       <     <            >   > (OUTput [tbegtbl] [delttbl]  <     MIn   >)
             | ...      |                                        |    HR   |
                                                                 |    DAy  |
```

With this optional command the user indicates that for each location of the output location set `'sname'` (see commands `POINTS`, `CURVE`, `FRAME` or `GROUP`) one or more variables should be written to a file. The keywords `HEADER` and `NOHEADER` determine the appearance of the table; the filename determines the destination of the data.

`'sname'`	name of the set of `POINTS`, `CURVE`, `FRAME` or `GROUP`
`HEADer`	output is written in fixed format to file with headers giving name of variable and unit per column. A disadvantage of this option is that the data are written in fixed format; numbers too large to be written will be shown as: `****`. Number of header lines is 4.
`NOHEADer`	output is written in floating point format to file and has no headers; it is intended primarily for processing by other programs. With some spreadsheet programs, however, the `HEADER` option works better.

`INDexed` a table on file is produced which can be used directly (without editing) as input to ARCVIEW, ARCINFO, etc. The user should give two `TABLE` commands, one to produce one file with `XP` and `YP` as output quantities, the other with `HS`, `RTM01` or other output quantities, such as one wishes to process in ARCVIEW or ARCINFO. The first column of each file produced by SWAN with this command is the sequence number of the output point. The last line of each file is the word END.

`'fname'` name of the data file where the output is to be written to.
Default for option HEADER is output to the `PRINT` file. In case of `NOHEADER` the filename is required.

` |...|`
`<  >` the output parameters are the same as given in command `BLOCK`.
` |...|`

`OUTPUT` the user requests output at various times. If the user does not use this option, the program will give `TABLE` output for the last time step of the computation.

`[tbegtbl]` begin time of the first field of the variable, the format is:

1 : ISO-notation	19870530.153000
2 : (as in HP compiler)	'30−May−87 15:30:00'
3 : (as in Lahey compiler)	05/30/87.15:30:00
4 :	15:30:00
5 :	87/05/30 15:30:00'
6 : as in WAM	8705301530

This format is installation dependent. See Implementation Manual or ask the person who installed SWAN on your computer. Default is ISO-notation.

`[delttbl]` time interval between fields, the unit is indicated in the next option:

SEC	unit seconds
MIN	unit minutes
HR	unit hours
DAY	unit days

Otherwise see command `BLOCK` except that the $x-$ and $y-$components of the vectorial quantities `VEL`, `FORCE` and `TRANSPORT` are always given with respect to the problem coordinate system.

The number of decimals in the table varies for the output parameters; it depends on the value of `[hexp]`, given in the command `QUANTITY`.

```
                  |    SPEC1D  |    | -> ABSolute   |
SPECout  'sname' <             >  <                 >  'fname' &
                  | -> SPEC2D  |    |    RELative   |

                                     | -> Sec  |
```

```
OUTput [tbegspc] [deltspc]  <     MIn   >
                            |     HR    |
                            |     DAy   |
```

With this optional command the user indicates that for each location of the output location set `'sname'` (see commands `POINTS`, `CURVE`, `FRAME` or `GROUP`) the 1D or 2D variance / energy (see command `SET`) density spectrum (either the relative frequency or the absolute frequency spectrum) is to be written to a data file. The name `'fname'` is required in this command.

`'sname'` name of the set of `POINTS`, `CURVE`, `FRAME` or `GROUP`

`SPEC2D` means that 2D (frequency-direction) spectra are written to file according to the format described in Appendix D. Note that this output file can be used for defining boundary conditions for subsequent SWAN runs (command `BOUNDSPEC`).

`SPEC1D` means that 1D (frequency) spectra are written to file according to the format described in Appendix D. Note that this output file can be used for defining boundary conditions for subsequent SWAN runs (command `BOUNDSPEC`).

`ABS` means that spectra are computed as function of absolute frequency (i.e. the frequency as measured in a fixed point).

`REL` means that spectra are computed as function of relative frequency (i.e. the frequency as measured when moving with the current).

`'fname'` name of the data file where the output is written to.

`OUTPUT` the user requests output at various times. If the user does not use this option, the program will give `SPECOUT` output for the last time step of the computation.

`[tbegspc]` begin time of the first field of the variable, the format is:

1 : ISO-notation	19870530.153000
2 : (as in HP compiler)	'30−May−87 15:30:00'
3 : (as in Lahey compiler)	05/30/87.15:30:00
4 :	15:30:00
5 :	87/05/30 15:30:00'
6 : as in WAM	8705301530

This format is installation dependent. See Implementation Manual or ask the person who installed SWAN on your computer. Default is ISO-notation.

`[deltspc]` time interval between fields, the unit is indicated in the next option:

SEC	unit seconds
MIN	unit minutes
HR	unit hours
DAY	unit days

`| -> Sec |`

```
NESTout 'sname' 'fname' OUTput [tbegnst] [deltnst] <     MIn   >
                                                   |    HR   |
                                                   |    DAy  |
```

CANNOT BE USED IN 1D-MODE

With this optional command the user indicates that the 2D spectra along a nest boundary `'sname'` (see command `NGRID`) should be written to a data file with name `'fname'`. This name is required in this command.

`'sname'` name of the set of output locations, as defined in a command `NGRID`

`'fname'` name of the data file where the output is written to. The file is structured according to the description in Appendix D, i.e. also the information about the location of the boundary are written to this file. SWAN will use this as a check for the subsequent nested run.

`OUTPUT` the user requests output at various times. If the user does not use this option, the program will give `NESTOUT` output for the last time step of the computation.

`[tbegnst]` begin time of the first field of the variable, the format is:

1 : ISO-notation	19870530.153000
2 : (as in HP compiler)	'30−May−87 15:30:00'
3 : (as in Lahey compiler)	05/30/87.15:30:00
4 :	15:30:00
5 :	87/05/30 15:30:00'
6 : as in WAM	8705301530

This format is installation dependent. See Implementation Manual or ask the person who installed SWAN on your computer. Default is ISO-notation.

`[deltnst]` time interval between fields, the unit is indicated in the next option:

SEC	unit seconds
MIN	unit minutes
HR	unit hours
DAY	unit days

4.6.3 Write or plot intermediate results

```
                                |        | < [i] [j] >  |   |
                                | -> IJ <               >  |
TEST  [itest] [itrace]  POINTS <         | < [k] >      |    >   &
                                |                         |
                                |    XY < [x] [y] >       |

      (PAR 'fname')  (S1D 'fname') (S2D 'fname')
```

If SWAN produces unexpected results, this optional command can be used to instruct the program to produce intermediate results during a SWAN run (test output). A `TEST` command may change between commands in the file to change the level of test output during a SWAN run. This change occurs during the execution of the run. A `TEST` command controls the test output until the next `TEST` command. Such a next `TEST` command may have level 0, thus stopping test output.

`[itest]` the level of test output. For values under 100 the amount is usually reasonable, for values above 200 it can be very large. For values of `[itest]` up to 50 the test output can be interpreted by the user. For higher values of `[itest]` the test output can only be interpreted by those who have the program source listing at their disposal. Note that for sequential or parallel runs, it might be interesting to print the timings (both wall-clock and CPU times in seconds) in the `PRINT` file; for this `[itest]` should be set to 1.
Default: `[itest]` = 1.

`[itrace]` SWAN writes a message (name of subroutine) to the `PRINT` file at the first `[itrace]` entries of each subroutine.
Default: `[itrace]` = 0.

`POINTS` if this option is used, the user instructs SWAN to produce detailed print output during the computational process for a grid point of the computational grid. Output at a maximum of 50 grid points is possible. This option can be used only after the bathymetry has been read (see command `READINP BOTTOM`).

`IJ` the test points are defined by means of grid indices.

`[i], [j]` grid indices of a test point. Values of `[i]` range between 0 and `[mxc]` (see command `CGRID`), values of `[j]` between 0 and `[myc]` (inclusive).
ONLY MEANT FOR STRUCTURED GRIDS.

`[k]` vertex index of a test point. This can be obtained in a grid generator file (`fort.14`, `.node` and `.n` files of ADCIRC, Triangle and Easymesh, respectively).
ONLY MEANT FOR UNSTRUCTURED GRIDS.

`XY` the test points are defined in terms of problem coordinates; SWAN will determine the nearest grid points. Output will be made for this selected grid point.

`[x], [y]` coordinates of a test point (problem coordinates in m in case of Cartesian coordinates, or longitude and latitude in degrees in case of spherical coordinates, see command `COORD`).

`PAR` integral parameters for test points are written: `HSIGN`, `RTM01` (see Appendix A for definitions) and Swind, Swcap, Ssurf, Sfric, Snl3 and Snl4 which are the integrals over frequency and direction of the respective source terms (wind input, whitecapping, depth-induced breaking, bottom friction, absolute value of the triad wave-wave interactions and absolute value of the quadruplet wave-wave interactions).

`S1D` if the keyword `S1D` appears variance densities and 6 source terms (see end of this list of options) as computed will be printed as 1D spectral (frequency) output.

	The definition of this file is given in Appendix D. This output will be made after every iteration in the case of `MODE STATIONARY`, and after every time step in the case `MODE NONSTATIONARY` (see command `MODE`).
`'fname'`	name of the file to which the output is written; default filename: `SWSRC1D`.
`S2D`	if the keyword `S2D` appears variance densities and 6 source terms (see end of this list of options) as computed will be printed as 2D (frequency and direction) spectral output. The format of this file is defined in Appendix D. This output will be made after every iteration in the case of `MODE STATIONARY`, and after every time step in the case `MODE NONSTATIONARY` (see command `MODE`).
`'fname'`	name of the file to which the output is written; default filename: `SWSRC2D`.

Note that the keywords `PAR`, `S1D` and `S2D` need to be given in that order.

The source terms written due to the presence of the keyword `S1D` or `S2D` are source terms of variance density. The 6 source terms are: wind input, whitecapping, bottom friction, breaking, 3-wave interactions and 4-wave interactions.

When a number `[maxmes]` of error messages, see command `SET`, have been written to the `PRINT` file, the computation will stop. If necessary make `[maxmes]` larger using command `SET`, and rerun the program.

4.7 Lock-up

```
             |    STATionary  [time]                              |
COMPute   ( <                                                     > )
             |                                  | -> Sec  |          |
             | -> NONSTat  [tbegc] [deltc] <    |    MIn  > [tendc] |
                                                |    HR   |
                                                |    DAy  |
```

This command orders SWAN to start the computation(s).

If the SWAN mode is stationary (see command `MODE`), then only the command `COMPUTE` should be given here (no options!).
If the SWAN mode is nonstationary (see command `MODE`), then the computation can be

- either stationary (at the specified time: option `STATIONARY` here) or
- nonstationary (over the specified period of time: `[tbegc]` etc.).

To verify input to SWAN (e.g., all input fields such as water depth, wind fields, etc.), SWAN can be run without computations (that is: zero iterations by using command `NUM`

`ACCUR MXITST = 0`).

In the case `MODE NONSTATIONARY` several commands `COMPUTE` can appear, where the wave state at the end of one computation is used as initial state for the next one, unless a command `INIT` appears in between the two `COMPUTE` commands. This enables the user to make a stationary computation to obtain the initial state for a nonstationary computation and/or to change the computational time step during a computation, to change a boundary condition etc. This also has the advantage of not using a hotfile since, it can be very large in size.

`STATIONARY`	a stationary computation is to be made.
`[time]`	time level for which the stationary run is to be made, the format is: 1 : ISO-notation 19870530.153000 2 : (as in HP compiler) '30−May−87 15:30:00' 3 : (as in Lahey compiler) 05/30/87.15:30:00 4 : 15:30:00 5 : 87/05/30 15:30:00' 6 : as in WAM 8705301530 This format is installation dependent. See Implementation Manual or ask the person who installed SWAN on your computer. Default is ISO-notation.
`NONSTATION`	a nonstationary computation is to be made.
`[tbegc]`	the start date and time of the nonstationary computation, format see `[time]`. Default: the time read from a hotfile (see command `INIT HOTSTART`), or the end time `[tendc]` of the previous nonstationary computation or the `[time]` of the previous stationary computation in the same SWAN run (if any).
`[deltc]`	the time step of the nonstationary computation, the unit is indicated in the next option: SEC unit seconds MIN unit minutes HR unit hours DAY unit days
`[tendc]`	the end time of the nonstationary computation, format see `[time]`.

`HOTFile 'fname'`

This command can be used to write the entire wave field at the end of a computation to a so-called hotfile, to be used as initial condition in a subsequent SWAN run (see command `INITIAL HOTSTART`) This command must be entered immediately after a `COMPUTE` command.

The format of the hotfile is identical to the format of the files written by the `SPECOUT` command (option SPEC2D).

`'fname'`	name of the file to which the wave field is written. Note: for parallel MPI runs, more than one hotfile will be generated depending on the number of processors (`fname-001`, `fname-002`, etc.).

`STOP`

This required command marks the end of the commands in the command file. Note that the command `STOP` may be the last command in the input file; any information in the input file beyond this command is ignored.

Appendix A

Definitions of variables

In SWAN a number of variables are used in input and output. Most of them are related to waves. The definitions of these variables are mostly conventional.

HSIGN

Significant wave height, denoted as H_s in meters, and defined as

$$H_s = 4\sqrt{\int\int E(\omega,\theta)d\omega d\theta}$$

where $E(\omega,\theta)$ is the variance density spectrum and ω is the absolute radian frequency determined by the Doppler shifted dispersion relation. However, for ease of computation, H_s can be determined as follows:

$$H_s = 4\sqrt{\int\int E(\sigma,\theta)d\sigma d\theta}$$

HSWELL

Significant wave height associated with the low frequency part of the spectrum, denoted as $H_{s,\text{swell}}$ in meters, and defined as

$$H_{s,\text{swell}} = 4\sqrt{\int_0^{\omega_{\text{swell}}}\int_0^{2\pi} E(\omega,\theta)d\omega d\theta}$$

with $\omega_{\text{swell}} = 2\pi f_{\text{swell}}$ and $f_{\text{swell}} = 0.1$ Hz by default (this can be changed with the command `QUANTITY`).

TMM10

Mean absolute wave period (in s) of $E(\omega,\theta)$, defined as

$$T_{m-10} = 2\pi\frac{\int\int \omega^{-1}E(\omega,\theta)d\omega d\theta}{\int\int E(\omega,\theta)d\omega d\theta} = 2\pi\frac{\int\int \omega^{-1}E(\sigma,\theta)d\sigma d\theta}{\int\int E(\sigma,\theta)d\sigma d\theta}$$

TM01

Mean absolute wave period (in s) of $E(\omega,\theta)$, defined as

$$T_{m01} = 2\pi\left(\frac{\int\int \omega E(\omega,\theta)d\omega d\theta}{\int\int E(\omega,\theta)d\omega d\theta}\right)^{-1} = 2\pi\left(\frac{\int\int \omega E(\sigma,\theta)d\sigma d\theta}{\int\int E(\sigma,\theta)d\sigma d\theta}\right)^{-1}$$

TM02 Mean absolute wave period (in s) of $E(\omega,\theta)$, defined as

$$T_{m02} = 2\pi \left(\frac{\int\int \omega^2 E(\omega,\theta) d\omega d\theta}{\int\int E(\omega,\theta) d\omega d\theta} \right)^{-1/2} = 2\pi \left(\frac{\int\int \omega^2 E(\sigma,\theta) d\sigma d\theta}{\int\int E(\sigma,\theta) d\sigma d\theta} \right)^{-1/2}$$

DIR Mean wave direction (in ^{o}, Cartesian or Nautical convention), as defined by (see Kuik *et al.* (1988)):

$$\mathrm{DIR} = \arctan \left\lfloor \frac{\int \sin\theta E(\sigma,\theta) d\sigma d\theta}{\int \cos\theta E(\sigma,\theta) d\sigma d\theta} \right\rfloor$$

This direction is the direction normal to the wave crests.

PDIR Peak direction of $E(\theta) = \int E(\omega,\theta) d\omega = \int E(\sigma,\theta) d\sigma$ (in ^{o}, Cartesian or Nautical convention).

TDIR Direction of energy transport (in ^{o}, Cartesian or Nautical convention). Note that if currents are present, TDIR is different from the mean wave direction DIR.

RTMM10 Mean relative wave period (in s) of $E(\sigma,\theta)$, defined as

$$RT_{m-10} = 2\pi \frac{\int\int \sigma^{-1} E(\sigma,\theta) d\sigma d\theta}{\int\int E(\sigma,\theta) d\sigma d\theta}$$

This is equal to TMM10 in the absence of currents.

RTM01 Mean relative wave period (in s) of $E(\sigma,\theta)$, defined as

$$RT_{m01} = 2\pi \left(\frac{\int\int \sigma E(\sigma,\theta) d\sigma d\theta}{\int\int E(\sigma,\theta) d\sigma d\theta} \right)^{-1}$$

This is equal to TM01 in the absence of currents.

RTP Relative peak period (in s) of $E(\sigma)$ (equal to absolute peak period in the absence of currents).
Note that this peak period is related to the absolute maximum bin of the discrete wave spectrum and hence, might not be the 'real' peak period.

TPS Relative peak period (in s) of $E(\sigma)$.
This value is obtained as the maximum of a parabolic fitting through the highest bin and two bins on either side the highest one of the discrete wave spectrum. This 'non-discrete' or 'smoothed' value is a better estimate of the 'real' peak period compared to the quantity `RTP`.

PER Average absolute period (in s) of $E(\omega,\theta)$, defined as

$$T_{m,p-1,p} = 2\pi \frac{\int\int \omega^{p-1} E(\omega,\theta) d\omega d\theta}{\int\int \omega^{p} E(\omega,\theta) d\omega d\theta}$$

The power p can be chosen by the user by means of the `QUANTITY`

command. If $p = 1$ (the default value) PER is identical to TM01 and if $p = 0$, PER = TMM10.

RPER Average relative period (in s), defined as

$$RT_{m,p-1,p} = 2\pi \frac{\int\int \sigma^{p-1} E(\sigma,\theta) d\sigma d\theta}{\int\int \sigma^{p} E(\sigma,\theta) d\sigma d\theta}$$

Here, if $p = 1$, RPER=RTM01 and if $p = 0$, RPER=RTMM10.

FSPR The normalized frequency width of the spectrum (frequency spreading), as defined by Battjes and Van Vledder (1984):

$$\text{FSPR} = \frac{|\int_0^\infty E(\omega) e^{i\omega\tau} d\omega|}{E_{\text{tot}}}, \quad \text{for } \tau = T_{m02}$$

DSPR The one-sided directional width of the spectrum (directional spreading or directional standard deviation,in ^{o}), defined as

$$\text{DSPR}^2 = \left(\tfrac{180}{\pi}\right)^2 \int_0^{2\pi} (2\sin(\tfrac{\theta-\overline{\theta}}{2}))^2 D(\theta) d\theta$$

and computed as conventionally for pitch-and-roll buoy data (Kuik *et al.* (1988); this is the standard definition for WAVEC buoys integrated over all frequencies):

$$(\text{DSPR}_{\frac{\pi}{180}})^2 = 2\left(1 - \sqrt{\left[\left(\frac{\int \sin\theta E(\sigma,\theta) d\sigma d\theta}{\int E(\sigma) d\sigma}\right)^2 + \left(\frac{\int \cos\theta E(\sigma,\theta) d\sigma d\theta}{\int E(\sigma) d\sigma}\right)^2\right]}\right)$$

QP The peakedness of the wave spectrum, defined as

$$Q_p = 2\frac{\int\int \sigma E^2(\sigma,\theta) d\sigma d\theta}{(\int\int E(\sigma,\theta) d\sigma d\theta)^2}$$

This quantity represents the degree of randomness of the waves. A smaller value of Q_p indicates a wider spectrum and thus increased the degree of randomness (e.g., shorter wave groups), whereas a larger value indicates a narrower spectrum and a more organised wave field (e.g., longer wave groups).

MS As input to SWAN with the commands `BOUNDPAR` and `BOUNDSPEC`, the directional distribution of incident wave energy is given by $D(\theta) = A(\cos\theta)^m$ for all frequencies. The parameter m is indicated as MS in SWAN and is not necessarily an integer number. This number is related to the one-sided directional spread of the waves (DSPR) as follows:

Table A.1: Directional distribution.

MS	DSPR (in ^{o})
1.	37.5
2.	31.5
3.	27.6
4.	24.9
5.	22.9
6.	21.2
7.	19.9
8.	18.8
9.	17.9
10.	17.1
15.	14.2
20.	12.4
30.	10.2
40.	8.9
50.	8.0
60.	7.3
70.	6.8
80.	6.4
90.	6.0
100.	5.7
200.	4.0
400.	2.9
800.	2.0

PROPAGAT — Energy propagation per unit time in $\vec{x}-$, $\theta-$ and $\sigma-$space (in W/m^2 or m^2/s, depending on the command `SET`).

GENERAT — Energy generation per unit time due to the wind input (in W/m^2 or m^2/s, depending on the command `SET`).

REDIST — Energy redistribution per unit time due to the sum of quadruplets and triads (in W/m^2 or m^2/s, depending on the command `SET`).

DISSIP — Energy dissipation per unit time due to the sum of bottom friction, whitecapping and depth-induced surf breaking (in W/m^2 or m^2/s, depending on the command `SET`).

RADSTR — Radiation stress per unit time, defined as

$$\int_0^{2\pi}\int_{\sigma_{\rm low}}^{\sigma_{\rm high}} |S_{\rm tot} - \frac{\partial E}{\partial t} - \nabla_{\vec{x}} \cdot [(\vec{c}_g + \vec{U})E] - \nabla_{(\sigma,\theta)} \cdot (\vec{c}_{(\sigma,\theta)}E)| d\sigma d\theta$$

(in W/m^2 or m^2/s, depending on the command `SET`).

WLEN — The mean wavelength, defined as

$$\text{WLEN} = 2\pi \left(\frac{\int\int k^p E(\sigma,\theta) d\sigma d\theta}{\int\int k^{p-1} E(\sigma,\theta) d\sigma d\theta} \right)^{-1}$$

As default, $p = 1$ (see command `QUANTITY`).

STEEPNESS — Wave steepness computed as HSIG/WLEN.

BFI — The Benjamin-Feir index or the steepness-over-randomness ratio, defined as

$$\text{BFI} = \sqrt{2\pi} \times \text{ STEEPNESS } \times \text{ QP}$$

This index can be used to quantify the probability of freak waves.

QB — Fraction of breakers in expression of Battjes and Janssen (1978).

TRANSP — Energy transport with components $P_x = \rho g \int\int c_x E(\sigma,\theta) d\sigma d\theta$ and $P_y = \rho g \int\int c_y E(\sigma,\theta) d\sigma d\theta$ with x and y the problem coordinate system, except in the case of output with `BLOCK` command in combination with command `FRAME`, where x and y relate to the $x-$axis and $y-$axis of the output frame.

VEL — Current velocity components in $x-$ and $y-$direction of the problem coordinate system, except in the case of output with `BLOCK` command in combination with command `FRAME`, where x and y relate to the $x-$axis and $y-$axis of the output frame.

WIND — Wind velocity components in $x-$ and $y-$direction of the problem coordinate sytem, except in the case of output with `BLOCK` command in combination with command `FRAME`, where x and y relate to the $x-$axis and $y-$axis of the output frame.

FORCE — Wave-induced force per unit surface area (gradient of radiation stresses) with x and y the problem coordinate system, except in the case of output with `BLOCK` command in combination with command `FRAME`, where x and y relate to the $x-$axis and $y-$axis of the output frame.

$$F_x = -\frac{\partial S_{xx}}{\partial x} - \frac{\partial S_{xy}}{\partial y}$$

$$F_y = -\frac{\partial S_{yx}}{\partial x} - \frac{\partial S_{yy}}{\partial y}$$

where S is the radiation stress tensor as given by

$$S_{xx} = \rho g \int \lfloor n \cos^2\theta + n - \tfrac{1}{2} \rfloor E d\sigma d\theta$$

$$S_{xy} = S_{yx} = \rho g \int n \sin\theta \cos\theta E d\sigma d\theta$$

$$S_{yy} = \rho g \int \lfloor n \sin^2\theta + n - \tfrac{1}{2} \rfloor E d\sigma d\theta$$

and n is the group velocity over the phase velocity.

URMS — Root-mean-square value (in m/s) of the orbital motion near the bottom $U_{\rm rms} = \sqrt{< U^2 >}$.

UBOT — Root-mean-square value (in m/s) of the maxima of the orbital motion near the bottom $U_{\rm bot} = \sqrt{2} U_{\rm rms}$.

TMBOT — Bottom wave period (in s) defined as the ratio of the bottom excursion amplitude to the bottom orbital velocity.

LEAK — Numerical loss of energy equal to $c_\theta E(\omega, \theta)$ across boundaries θ_1=[dir1] and θ_2=[dir2] of a directional sector (see command `CGRID`).

TIME — Full date-time string.

TSEC — Time in seconds with respect to a reference time (see command `QUANTITY`).

SETUP — The elevation of mean water level (relative to still water level) induced by the gradient of the radiation stresses of the waves.

Cartesian convention — The direction is the angle between the vector and the positive $x-$axis, measured counterclockwise. In other words: the direction where the waves are going to or where the wind is blowing to.

Nautical convention — The direction of the vector from geographic North measured clockwise. In other words: the direction where the waves are coming from or where the wind is blowing from.

Appendix B

Command syntax

B.1 Commands and command schemes

The actual commands of the user to SWAN must be given in one file containing all commands. This file is called the command file. It must be presented to SWAN in ASCII. It is important to make a distinction between the description of the commands in this User Manual and the actual commands in the command file. The descriptions of the commands in this User Manual are called command schemes. Each such command scheme includes a diagram and a description explaining the structure of the command and the meaning of the keyword(s) and of the data in the command. The proper sequence of the commands is given in Section 4.2.

B.2 Command

B.2.1 Keywords

Each command instructs SWAN to carry out a certain action which SWAN executes before it reads the next command. A command must always start with a keyword (which is also the name of the command) which indicates the primary function of that command; see list in Section 4.1). A simple command may appear in its command scheme as:

```
KEYword data
```

A command may contain more than one keyword (which refines the instructions to SWAN), e.g.,

```
KEY1word KEY2word data
```

where `KEY2word` is the second keyword.

Spelling of keywords

In every command scheme, keywords appear as words in both lower- and upper-case letters. When typing the command or keyword in the command file, the user **must at least** copy literally the part with upper-case letters. SWAN reads only this part and SWAN is case insensitive except for one instance (character strings), see below. When typing the keyword in the command file, any extension of the part with upper-case letters is at the users discretion as long as the extension is limited to letters or digits, as well as the characters − and _. So, in the first command outlined above one may write: `KEY` or `KEYW` or `KEY−word` or `keyhole`, etc., whereas with the abovementioned second command scheme, `key1 KEY2 data` may appear in the command file.

In the command file

- a keyword is closed by a blank or one of the following characters = or :
- a keyword is not enclosed by square brackets or quotes,
- a keyword followed by a comma (,) is interpreted as a keyword followed by an empty data field (see below).

Required and optional keywords

All keywords in a command are required except when an option is available.

Optional keywords are indicated in the command scheme with the following signs enclosing the keywords concerned:

```
  |  KEY1word ...... data ....... |
<                                   >
  |  KEY2word ...... data ....... |
```

For the above example it may appear as:

```
            |  KEY2word data |
 KEY1word  <                  >
            |  KEY3word data |
```

In case the user does not indicate an option in a command, SWAN chooses the alternative indicated with an arrow (`->`) appearing in the command scheme (the default option). In the above example, it may appear as:

```
            |    KEY2word data |
 KEY1word  <                    >
            | -> KEY3word data |
```

where `KEY3WORD` is the default option.

Repetitions of keywords and/or other data

The use of keywords is sometimes repetitive, e.g. in a sequence of data and keywords containing many locations in $x, y-$space. In such a case, the command scheme indicates this repetitive nature by placing the keywords (and data) concerned between angle brackets < >. For instance,

```
KEY1word <data KEY2word data>
```

In the actual command in the command file the user must give such a sequence. It ends with either

- end of line
- a keyword other than the ones mentioned in the repetition group
- the character / or ;

If more than one line is required for a command, the user may continue on the next line as described in Section B.4. The repetition may consist of one instance (in fact, no repetition at all).

B.2.2 Data

Most commands contain data, either character data or numerical data.

Character data and numerical data

Character data (character strings) are represented in the command schemes by names, enclosed in quotes (' ').

Numerical data are represented in the command schemes by names enclosed in square brackets ([ ]).

As a rule, an error message will result if numerical data is given where character data should be given.

Spelling of data

Character data are represented as character strings (sequence of characters and blanks) between quotes (in the command scheme and in the command file). SWAN interprets an end of line as an end quote (a character data field can therefore never extend over more than one line).

In a command scheme the character string is always a name (which is placed between quotes as indicated). In the command file such a name can be entered in two ways:

- Replace the name by another character string at the users discretion (between quotes; this is the only occurrence where SWAN is case sensitive; e.g. for text to appear in a plot.
  Example:
  command scheme: `KEYword 'City' data`
  command file: `KEY 'Amsterdam' data`

- Copy the name of the variable (without the quotes) literally followed by an = sign and a name at the users discretion (between quotes). SWAN interprets the copied name in the command file as a keyword with all the characteristics of a keyword such as ending a sequence of optional data (see below). As with other keywords the name of the variable is case-insensitive.
  Example:
  command scheme: `KEYword 'City' data`
  command file: `KEY city='Amsterdam' data`

As a rule, an error message will result if numerical data is given where character data should be given.

Numerical data are simple numbers , e.g. 15 or −7 (integer data), or 13.7 or 0.8E−4 (real data). Whether or not integer number or real number should be given by the user is indicated in the description of the command scheme.

Note that a decimal point is not permitted in an integer number. On the other hand, an integer number is accepted by SWAN where a real number should be given.

In a command scheme, the number is always indicated with a name (which is placed between square brackets). In the command file such a name can be entered in two ways:

- Replace the name by a number (not between square brackets).
  Example:
  command scheme: `KEYword [nnn]`
  command file: `KEY 314`

- Copy the name of the variable (without the quotes) literally followed by an = sign and the number (not between square brackets). SWAN interprets the copied name in the command file as a keyword with all the characteristics of a keyword such as ending a sequence of optional data (see below). As with other keywords the name of the variable is case-insensitive.
  Example:
  command scheme: `KEYword [nnn]`
  command file: `KEY nnn=314`

Required data and optional data

All data must be given by the user in the command file in the same order as they appear in the command scheme. They are separated by blanks or comma's.

Required data (indicated in the description of each individual command) must be given explicitly as character string or numbers.

Optional data are indicated

(a) in the text of each individual command or

(b) for sets of data: in parenthesis around the data concerned

```
( data )
```

For example:

```
KEY1word KEY2word 'name' ([nnn] [mmm]) [zzz]
```

or

(c) some optional data are indicate in the same way as optional keywords are indicated:

```
 | .....data.....|
<                 >
 | .....data.....|
```

Optional data of the kind (a) or (b) may be omitted by giving blanks between comma's (SWAN then substitutes reasonable default values). If after a required datum all data is optional (till the next keyword or the next end-of-line), then the comma's may be omitted too. Optional data of the kind (c) are to be treated in the same way as optional keywords.

B.3 Command file and comments

All text after one $ or between two $ signs on one line in the command file is ignored by SWAN as comment. Such comments may be important to the user e.g., to clarify the meaning of the commands used. In fact, this option has been used to create the edit file `swan.edt` (see Appendix C). Anything appearing after two $ signs is not interpreted

as comment, but again as data to be processed (possibly interrupted again by $ or two $ signs). Since version 40.20, the exclamation mark '!' can be used as comment sign. Everthing behind a ! is interpreted as comment, also if ! or $ are in that part of the input line.

B.4 End of line or continuation

A command in the command file may be continued on the next line if the previous line terminates with a continuation mark & or _ (underscore).

Appendix C

File `swan.edt`

Below the file `swan.edt` is presented in which all the commands that can be used with SWAN are specified.

```
!   PROJECT  'name'  'nr'
!            'title1'
!            'title2'
!            'title3'
!
!   SET  [level]  [nor]  [depmin]  [maxmes]                                  &
!        [maxerr]  [grav]  [rho]  [inrhog]                                  &
!        [hsrerr]  CARTesian|NAUTical  [pwtail]                             &
!        [froudmax]  [printf]  [prtest]
!
!   MODE  / STATIONARY \  / TWODimensional
!         \ DYNAMIC    /  \ ONEDimensional
!
!   COORDinates  /  -> CARTesian          \   REPeating
!                \ SPHErical      CCM|QC  /
!
!   CGRID  / REGular [xpc] [ypc] [alpc] [xlenc] [ylenc] [mxc] [myc] \
!          <  CURVilinear [mxc] [myc]  (EXC [xexc] [yexc])           >      &
!           \ UNSTRUCtured                                          /
!
!          / CIRcle                \
!          \ SECtor [dir1] [dir2] /  [mdc]  [flow]  [fhig]  [msc]
!
!   INPgrid                                                                 &
!      BOTtom | WLEVel | CURrent | VX | VY | FRiction | WInd | WX | WY      &
!
```

```
!      | REG [xpinp] [ypinp] [alpinp]  [mxinp] [myinp]  [dxinp] [dyinp] |
!      |                                                                 |
!     <  CURVilinear [stagrx] [stagry] [mxinp] [myinp]                  >  &
!      |                                                                 |
!      | UNSTRUCtured                                                    |
!
!      (EXCeption  [excval])                                               &
!
!      (NONSTATionary [tbeginp] [deltinp] SEC|MIN|HR|DAY [tendinp])
!
!                           | -> ADCirc
!   READgrid UNSTRUCtured <  TRIAngle \
!                           | EASYmesh / 'fname'
!
!   READinp    BOTtom|WLevel|CURrent|FRiction|WInd|COORdinates              &
!        [fac]  / 'fname1'        \
!               \ SERIES 'fname2' /  [idla] [nhedf] ([nhedt]) (nhedvec])    &
!        FREE | FORMAT 'form' | [idfm] | UNFORMATTED
!
!   WIND  [vel] [dir]
!
!                        |  JONswap  [gamma]  |
!   BOUNd SHAPespec      |  PM                |
!                       <   GAUSs  [sigfr]     >  PEAK|MEAN   DSPR  POWer|DEGRees
!                        |  BIN               |
!
!              / -> SIDE  N|NW|W|SW|S|SE|E|NE | [k]  CCW|CLOCKWise       \
!   BOUNdspec <                                                            > &
!              \    SEGment  / -> XY  < [x]  [y] >            \          /
!                            \    IJ  < [i]  [j] > | < [k] >  /
!
!            /  UNIForm   /  PAR  [hs] [per] [dir] [dd]
!           <             \  FILE  'fname'  [seq]
!            \  VARiable  /  PAR  < [len] [hs] [per] [dir] [dd] >
!                         \  FILE < [len] 'fname' [seq] >
!
!   BOUNdnest1  NEST 'fname'  CLOSed|OPEN
!
!   BOUNdnest2  WAMNest 'fname' / UNFormatted  CRAY|WKstat  \
!                               \ FREE                     /   [xgc] [ygc]
!
!   BOUNdnest3  WWIII 'fname'  CLOSed|OPEN  [xgc] [ygc]
!
```

```
!             | -> DEFault
!             |
!   INITial  <   ZERO
!             |
!             |  PAR  [hs] [per] [dir] [dd]
!             |
!             |            | -> MULTiple |
!             |  HOTStart <               >  'fname'
!             |            |    SINGle   |
!
!   GEN1  [cf10] [cf20] [cf30] [cf40] [edmlpm] [cdrag] [umin]
!
!   GEN2  [cf10] [cf20] [cf30] [cf40] [cf50] [cf60] [edmlpm] [cdrag] [umin]
!
!         | JANSSEN  [cds1] [delta]     |
!         |                             |
!         | -> KOMEN [cds2] [stpm]      |
!   GEN3 <                               >    (AGROW [a])
!         | YAN                         |
!         |                             |
!         | WESTHuysen                  |
!
!
!         | ->KOMen    [cds2] [stpm] [powst] [delta] [powk]  |
!         |                                                   |
!         |   JANSsen  [cds1]  [delta] [pwtail]               |
!         |                                                   |
!         |   LHIG     [cflhig]                               |
!         |                                                   |
!   WCAP <    BJ       [bjstp] [bjalf]                         >
!         |                                                   |
!         |   KBJ      [bjstp] [bjalf] [kconv]                |
!         |                                                   |
!         |   CSM      [cst] [pow]                            |
!         |                                                   |
!         |   AB       [cds2] [br] [p0] [powst] [powk]        |
!
!   QUADrupl [iquad] [limiter] [lambda] [cnl4] [csh1] [csh2] [csh3]
!
!                              | CNL4 < [cnl4] >               |
!   MDIA LAMbda < [lambda] > <                                  >
!                              | CNL4_12 < [cnl4_1] [cnl4_2] > |
!
```

```
!   BREaking CONSTANT [alpha] [gamma]
!
!   FRICTION  JONSWAP [cfjon] | COLLINS [cfw] | MADSEN [kn]
!
!   TRIAD   [trfac] [cutfr] [urcrit] [urslim]
!
!   LIMiter [ursell] [qb]
!
!               |  TRANSm [trcoef]                          |
!   OBSTacle   <                                            |
!               |       | -> GODA [hgt] [alpha] [beta]       >              &
!               |  DAM <                                    |
!                      |    DANGremond [hgt] [slope] [Bk]  |
!
!                         | -> RSPEC        |
!        ( REFLec [reflc] <                   > )                           &
!                         |    RDIFF [pown] |
!
!              LINe < [xp] [yp] >
!
!   SETUP  [supcor]
!
!   DIFFRac  [idiffr] [smpar] [smnum] [cgmod]
!
!   OFF WINDGrowth | QUADrupl | WCAPping | BREaking | REFrac | FSHift | BNDCHK
!
!   PROP   /  BSBT
!          \  GSE  [waveage] SEC|MIN|HR|DAY
!
!            | -> ACCUR [drel] [dhoval] [dtoval] [npnts] |
!   NUMeric (<                                            >              &
!            |    STOPC [dabs] [drel] [curvat] [npnts]   |
!
!                   | -> STAT  [mxitst] [alfa] |
!                  <                           >  [limiter]   )          &
!                   | NONSTat  [mxitns]        |
!
!          ( DIRimpl [cdd] [cdlim]                           )          &
!
!          ( SIGIMpl [css] [eps2] [outp] [niter]             )          &
!
!          ( SETUP [eps2] [outp] [niter]                     )
!
```

```
!   FRAME   'sname'  [xpfr] [ypfr] [alpfr] [xlenfr] [ylenfr] [mxfr] [myfr]
!
!   GROUP   'sname'  SUBGRID [ix1] [ix2] [iy1] [iy2]
!
!   CURVE   'sname'  [xp1] [yp1]   < [int]  [xp]  [yp] >
!
!   RAY     'rname'  [xp1] [yp1] [xq1] [yq1]                              &
!                    <  [int]  [xp]  [yp]  [xq]  [yq]  >
!
!   ISOLINE 'sname'  'rname'  DEPTH|BOTTOM  [dep]
!
!   POINTS  'sname'  < [xp]  [yp]  >     |    FILE 'fname'
!
!                    | [xpn] [ypn] [alpn] [xlenn] [ylenn] [mxn] [myn]
!   NGRID  'sname'  <
!                    | UNSTRUCtured / -> TRIAngle \
!                                      \    EASYmesh / 'fname'
!
!              |...|
!   QUANTity  <     >   'short'  'long'  [lexp]  [hexp]  [excv]            &
!              |...|
!
!             [power]   [ref]   [fswell]    [fmin]  [fmax]                &
!
!             / -> PROBLEMcoord \
!             \ FRAME           /
!
!   OUTPut OPTIons  'comment'  (TABle [field])  (BLOck  [ndec]  [len])     &
!                              (SPEC  [ndec])
!
!   BLOCK   'sname'  HEADER | NOHEADER  'fname' (LAY-OUT [idla])           &
!         <  DSPR|HSIGN|DIR|PDIR|TDIR|TM01|RTM01|RTP|TM02|FSPR|DEPTH|VEL|  &
!            FRCOEFF|WIND|DISSIP|QB|TRANSP|FORCE|UBOT|URMS|WLEN|STEEPNESS| &
!            DHSIGN|DRTM01|LEAK|TSEC|XP|YP|DIST|SETUP|TMM10|RTMM10|        &
!            TMBOT|QP|BFI|WATLEV|BOTLEV|TPS|DISBOT|DISSURF|DISWCAP|        &
!            GENE|GENW|REDI|REDQ|REDT|PROPA|PROPX|PROPT|PROPS|RADS >       &
!            ([unit]) (OUTPUT [tbegblk] [deltblk] SEC|MIN|HR|DAY)
!
!   TABLE   'sname'  HEADER | NOHEADER | INDEXED 'fname'                   &
!         <  DSPR|HSIGN|DIR|PDIR|TDIR|TM01|RTM01|RTP|TM02|FSPR|DEPTH|VEL|  &
!            FRCOEFF|WIND|DISSIP|QB|TRANSP|FORCE|UBOT|URMS|WLEN|STEEPNESS| &
!            DHSIGN|DRTM01|LEAK|TIME|TSEC|XP|YP|DIST|SETUP|TMM10|RTMM10|   &
!            TMBOT|QP|BFI|WATLEV|BOTLEV|TPS|DISBOT|DISSURF|DISWCAP|        &
```

```
!             GENE|GENW|REDI|REDQ|REDT|PROPA|PROPX|PROPT|PROPS|RADS >       &
!             ([unit]) (OUTPUT [tbegtbl] [delttbl] SEC|MIN|HR|DAY)
!
!   SPECout 'sname'  SPEC1D|SPEC2D  ABS|REL   'fname'                        &
!                    (OUTput [tbeg] [delt] SEC|MIN|HR|DAY)
!
!   NESTout 'sname'  'fname'                                                 &
!                    (OUTput [tbeg] [delt] SEC|MIN|HR|DAY)
!
!                                 / -> IJ < [i] [j] > | < [k] >  \
!   TEST [itest] [itrace] POINTS <                                 >         &
!                                 \    XY < [x] [y] >            /
!
!                         PAR 'fname'  S1D 'fname'  S2D 'fname'
!
!              |  STATionary  [time]                        |
!   COMPute ( <                                            > )
!              |                     | -> Sec  |           |
!              |  ([tbegc] [deltc] <     MIn   > [tendc]) |
!                                    |    HR   |
!                                    |    DAy  |
!
!   HOTFile 'fname'
!
!   STOP
```

Appendix D

Spectrum files, input and output

This appendix described the format of the files for spectral input (command `BOUNdspec`) and output (commands `SPECout` and `NESTout`) by SWAN. The files are recognized by SWAN or another reading program by the presence of the keyword SWAN and a version number on the first line of the file. This description is valid for version number 1.

These files contain the following information:

- coordinates of locations
- frequencies
- directions (if used for 2D)
- time (if time-dependent)
- spectral energy or variance densities (and average direction and direction spreading if 1D)

Example of a 1D nonstationary spherical coordinates file

```
SWAN   1                                Swan standard spectral file, version
$ Data produced by SWAN version 40.72A
$ Project:'projname'     ;   run number:'runnum'
TIME                                    time-dependent data
     1                                  time coding option
LONLAT                                  locations in spherical coordinates
     2                                  number of locations
        1.00        1.00
        1.20        1.00
RFREQ                                   relative frequencies in Hz
    25                                  number of frequencies
    0.0418
    0.0477
```

```
    0.0545
    0.0622
    0.0710
    0.0810
    0.0924
    0.1055
    0.1204
    0.1375
    0.1569
    0.1791
    0.2045
    0.2334
    0.2664
    0.3040
    0.3470
    0.3961
    0.4522
    0.5161
    0.5891
    0.6724
    0.7675
    0.8761
    1.0000
QUANT
     3                              number of quantities in table
VaDens                              variance densities in m2/Hz
m2/Hz                               unit
   -0.9900E+02                      exception value
CDIR                                average Cartesian direction in degr
degr                                unit
   -0.9990E+03                      exception value
DSPRDEGR                            directional spreading
degr                                unit
   -0.9000E+01                      exception value
19680606.030000                     date and time
LOCATION   1
  0.3772E-03  190.1    6.3
  0.1039E-02  190.2    6.5
  0.2281E-02  190.3    6.7
  0.3812E-02  190.3    6.7
  0.4255E-02  190.3    6.6
  0.2867E-02  190.1    6.3
  0.1177E-02  189.6    5.8
```

```
  0.3892E-03  192.0   15.2
  0.8007E-03  244.5   22.9
  0.6016E-02  251.4   11.5
  0.1990E-01  251.0   11.0
  0.3698E-01  249.9   10.9
  0.3874E-01  248.1   12.1
  0.2704E-01  246.6   13.0
  0.1672E-01  247.0   13.5
  0.1066E-01  247.7   13.7
  0.5939E-02  247.3   14.0
  0.3247E-02  246.5   14.6
  0.1697E-02  245.9   14.9
  0.8803E-03  245.6   15.1
  0.4541E-03  245.5   15.3
  0.2339E-03  245.4   15.5
  0.1197E-03  245.5   15.6
  0.6129E-04  245.5   15.7
  0.3062E-04  245.3   15.9
LOCATION   2
  0.7129E-02   67.2   25.3
  0.3503E-01   67.5   21.7
  0.1299E+00   68.2   19.7
  0.5623E+00   69.7   18.0
  0.1521E+01   71.4   18.0
  0.3289E+01   74.0   18.8
  0.4983E+01   77.2   20.3
  0.4747E+01   79.9   22.0
  0.2322E+01   79.4   30.7
  0.1899E+01  341.1   56.2
  0.1900E+01  314.6   39.4
  0.6038E+01  324.3   31.9
  0.8575E+01  326.1   31.0
  0.4155E+01  325.1   30.5
  0.1109E+01  322.8   32.9
  0.7494E+00  323.1   33.3
  0.4937E+00  323.1   33.3
  0.2953E+00  323.3   33.7
  0.1661E+00  323.6   34.0
  0.9788E-01  323.7   33.8
  0.5766E-01  323.8   33.6
  0.3397E-01  324.0   33.5
  0.2001E-01  324.1   33.4
  0.1179E-01  324.2   33.3
```

```
  0.6944E-02  324.2   33.2
```

Example of a 2D stationary Cartesian coordinates file

```
SWAN   1                                Swan standard spectral file, version
$ Data produced by SWAN version 40.72A
$ Project:'projname'     ;   run number:'runnum'
LOCATIONS                               locations in x-y-space
     1                                  number of locations
    22222.22        0.00
RFREQ                                   relative frequencies in Hz
    25                                  number of frequencies
    0.0418
    0.0477
    0.0545
    0.0622
    0.0710
    0.0810
    0.0924
    0.1055
    0.1204
    0.1375
    0.1569
    0.1791
    0.2045
    0.2334
    0.2664
    0.3040
    0.3470
    0.3961
    0.4522
    0.5161
    0.5891
    0.6724
    0.7675
    0.8761
    1.0000
CDIR                                    spectral Cartesian directions in degr
    12                                  number of directions
   30.0000
   60.0000
   90.0000
  120.0000
```

```
  150.0000
  180.0000
  210.0000
  240.0000
  270.0000
  300.0000
  330.0000
  360.0000
QUANT
     1                                  number of quantities in table
VaDens                                  variance densities in m2/Hz/degr
m2/Hz/degr                              unit
   -0.9900E+02                          exception value
FACTOR
    0.675611E-06
    51   242   574   956  1288  1482  1481  1286   957   579   244    51
   129   610  1443  2402  3238  3725  3724  3234  2406  1454   613   128
   273  1287  3054  5084  6846  7872  7869  6837  5091  3076  1295   271
   665  3152  7463 12402 16712 19229 19221 16690 12419  7518  3172   662
  1302  6159 14608 24275 32688 37618 37603 32644 24309 14716  6198  1296
  2328 10989 26020 43341 58358 67109 67080 58281 43401 26213 11058  2317
  3365 15922 37712 62733 84492 97150 97110 84380 62820 37991 16021  3349
  3426 16230 38440 63939 86109 99010 98969 85995 64027 38724 16331  3410
  2027  9612 22730 37790 50909 58529 58505 50841 37843 22898  9672  2018
   672  3178  7538 12535 16892 19440 19432 16870 12552  7594  3198   669
   101   479  1135  1890  2542  2924  2923  2539  1892  1144   482   101
     2    11    26    43    57    66    66    57    43    26    11     2
     0     0     0     1     1     1     1     1     1     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
```

Note that the true variance or energy densities are obtained by multiplying each number with the factor given under the keyword `FACTOR`.

Formal description of the 1D- and 2D-spectral file

This description refers to either **write or read** energy/variance density spectra **to or from** the file.

The description of the file to write or read source terms is identical to this description except that the quantities obviously differ.

The format that is used by SWAN or should be used by the user when offering the file to SWAN is free format (FORTRAN convention) except that all keywords and names of quantities (see below) should start on the first position of the line on which they appear (see Appendix B for the syntax of keywords). This format implies that all information on each line after the required input for SWAN is ignored by SWAN. This can be used to enter user's information at the discretion of the user.

First line the keyword `SWAN` followed by version number

Then if nonstationary computation: the keyword `TIME`
if stationary computation: not present

Then if nonstationary computation: time coding option; ISO-notation (=1) is recommended
if stationary computation: not present

Then
- the description of the locations:
  if Cartesian coordinates: the keyword `LOCATIONS`
  if spherical coordinates: the keyword `LONLAT`
- number of locations
- for each location
  if Cartesian coordinates: $x-$ and $y-$coordinate (in m, problem coordinates)
  if spherical coordinates: longitude and latitude
  Note that if the file is used for input for SWAN (but not generated by SWAN) and the user so desires, the names of locations can be written behind the two coordinates; these names are ignored by SWAN when reading the file (see remark on format above).

Then the frequency data (for 1D- and 2D-spectra):
- the keyword `AFREQ` or `RFREQ` (to distinguish between absolute and relative frequencies)
- number of frequencies
- a column with frequencies always in Hz (each on a new line)

Then the direction data (only for 2D-spectra):
- the keyword `NDIR` or `CDIR` (to distinguish between nautical and Cartesian direction)
- number of directions
- a column with directions always in degrees (each on a new line)

Then the group describing the quantities in the tables of this file (see the above examples):
- the keyword `QUANT`
- number of quantities

for each quantity

- name of the quantity
- unit of the quantity
- exception value of the quantity, i.e. the value that is written instead of a computed value if that is undefined

Note for 1D spectra:
the number of quantities is always 3, and the quantities are always: energy (or variance) density, average direction (`CDIR` for Cartesian direction or `NDIR` for nautical direction) and directional spreading (`DSPR` in terms of `DEGREES` (SWAN *write* the keyword `DSPRDEGR`, SWAN *reads* the keyword `DSPRD` or `DEGR` in case of option `DEGREES` or the keywords `DSPRP` or `POWER` in case of option `POWER` in command `BOUND SHAPE`). The quantities appear in the order in which they appear in this description.
Note for 2D spectra:
the number of quantities is always 1; the quantity is always energy or variance density (`EnDens` is the (short) name of true energy densities, `VaDens` for variance densities).

Then the group with the tables of the quantities:

VVV
- date and time (not present for stationary computation)

for each location:

if 2D spectrum:

- the keyword `FACTOR`. This keyword is replaced by the keyword `ZERO` if the spectrum is identical 0 or it is replaced by `NODATA` if the spectrum is undefined (not computed e.g., on land; no numbers follow)
- the factor to multiply the values in the following table
- scaled energy/variance densities (truncated by SWAN to integer values for compact writing; SWAN accepts these values as reals when reading this file; other programs (e.g. for postprocessing) should also accept these values as reals; the values should be multiplied by the factor to get the proper values of the densities).

else, if 1D spectrum:

- the keyword `LOCATION` followed by the index of the location (on the same line). This is replaced by the keyword `NODATA` if the spectrum is undefined (not computed e.g. on land; no numbers follow).
- a table containing three columns: the 3 quantities per frequency: energy (or variance) density, average direction (`CDIR` for Cartesian direction and `NDIR` for nautical direction) and directional spreading (`DSPR` in terms of `DEGREES` (*writing or reading the file*) or `POWER` (*only reading* the file), see note for 1D spectra above); the quantities appear in the order in which they appear in this description.

For nonstationary computations repeat from VVV.

Bibliography

[1] SWAN – Implementation manual. Delft University of Technology, Environmental Fluid Mechanics Section, available from *http://www.fluidmechanics.tudelft.nl/swan/index.htm* (Version 40.72, May 2008).

[2] SWAN – Programming rules. Delft University of Technology, Environmental Fluid Mechanics Section, available from *http://www.fluidmechanics.tudelft.nl/swan/index.htm* (Version 1.3, August 2006).

[3] SWAN – System documentation. Delft University of Technology, Environmental Fluid Mechanics Section, to be available.

[4] SWAN – Scientific and Technical documentation. Delft University of Technology, Environmental Fluid Mechanics Section, available from *http://www.fluidmechanics.tudelft.nl/swan/index.htm* (Version 40.72A, October 2008).

Index