Prepared for:
National Institute for Coastal and Marine Management (RIKZ)

Quality Assessment source code SWAN

November 1999

WL | delft hydraulics
Quality Assessment source code SWAN

November 1999

WL | delft hydraulics
CLIENT: National Institute for Coastal and Marine Management (RIKZ)

TITLE: Quality Assessment source code SWAN

ABSTRACT:

SWAN is a third-generation wave model developed at Delft University. For new development(s) to be included in the authorised SWAN version, programming guidelines should be followed.

For a sound Quality Assessment a judgement of all parts of the SWAN system is necessary. This assessment is only a pilot study: a quality assessment of a small part the source code, representative for the main part of the SWAN system. It is a step in the process of further improvement of the SWAN system, aimed to acquire an official quality mark.

This study demonstrates various options for retrieving a good view in the quality of (a part of) the source code of SWAN with respect to source code documentation, program structure and use of the ‘SWAN Programming Protocol’.

REFERENCES:
1) RIKZ/OS/996271 (dated July 8, 1999) of ir J.J.W. Seijffert
2) order 22991923 (dated July 22, 1999) of ir J.H. Andorka Gal

<table>
<thead>
<tr>
<th>VER.</th>
<th>ORIGINATOR</th>
<th>DATE</th>
<th>REMARKS</th>
<th>REVIEW</th>
<th>APPROVED BY</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>J.C.M. Dijkzeul</td>
<td>October 1999</td>
<td>Draft</td>
<td>G.K. Verboom</td>
<td>W.M.K. Tilmans</td>
</tr>
<tr>
<td>0.1</td>
<td>J.C.M. Dijkzeul</td>
<td>October 1999</td>
<td>Draft</td>
<td></td>
<td>W.M.K. Tilmans</td>
</tr>
<tr>
<td>1.0</td>
<td>J.C.M. Dijkzeul</td>
<td>November 1999</td>
<td>Final</td>
<td>J. Dekker</td>
<td></td>
</tr>
</tbody>
</table>

PROJECT IDENTIFICATION: H3595

KEYWORDS: SWAN wave model, Quality Assessment source code

CONTENTs: TEXT PAGES
19 TABLES - FIGURES - APPENDICES 2

STATUS: □ PRELIMINARY □ DRAFT ☒ FINAL
# Contents

1 Introduction........................................................................................................... 1–1

2 Outline of the assessment .................................................................................. 2–1

2.1 Programming Protocol .............................................................................. 2–1

2.2 Programming practice .............................................................................. 2–1

2.3 Limitations ................................................................................................. 2–1

2.4 Audit aspects and criteria .......................................................................... 2–2

2.5 Tools used for inspection ........................................................................... 2–3

3 Marginal check source code ............................................................................ 3–1

3.1 Subroutine documentation (header) ......................................................... 3–1

3.2 Program structure (structured English) .................................................. 3–3

3.3 Source code ............................................................................................... 3–3

3.3.1 Methods implemented ........................................................................ 3–3

3.3.2 Standard FORTRAN .......................................................................... 3–4

4 Functional check source code ....................................................................... 4–1

4.1 Internal documentation (header) ............................................................. 4–1

4.2 Structure (structured English) ................................................................. 4–2

4.3 Source code ............................................................................................... 4–3

5 FORTRAN90 ................................................................................................. 5–1

5.1 FORTRAN90 compliance ....................................................................... 5–1

5.2 Transition to FORTRAN90 ...................................................................... 5–2

6 Conclusions and recommendations ................................................................. 6–1

7 References.................................................................................................... 7–1
Appendices

A  SWAN Programming Protocol

B  SWAN source code

B.1 Subroutine INAR2D.FOR .................................................................B–1
B.2 Subroutine SSURF.FOR .................................................................B–6
B.3 Subroutine SWCOMP.FOR ...............................................................B–11
B.4 Subroutine SWOEXF.FOR ...............................................................B–34
I Introduction

SWAN is a third-generation wave model developed at Delft University. It is used, adapted and extended by many researchers at various institutes. If a developer wants to have his SWAN development(s) included in the authorised SWAN version, guidelines should be followed: for new developments and extensions of existing functionality of SWAN, attention has to be paid by the developers to:

1. System- and source code documentation;
2. User manual;
3. Source code (program structure, 'Programming Protocol', etc.);
4. Version management;
5. Testing.

The SWAN-team, taking care of the maintenance of SWAN at Delft University, pays much attention to the above mentioned aspects of the further development of the SWAN system. To assess the current status of SWAN a sound Quality Assessment of all parts of the SWAN system is necessary.

This assessment is only a pilot study: a quality assessment of one of the above aspects, source code. It is a step in the process of further improvement of the SWAN system, aimed to acquire an official quality mark.

Goal

The purpose of this study is to demonstrate various options for retrieving a good view in the quality of (part of) the source code of SWAN.

Four representative subroutines have been selected by Delft University of Technology and although these routines are meant to be representative for the main part of the SWAN system, the conclusions are only for this part of the SWAN source code.
2 Outline of the assessment

2.1 Programming Protocol

A SWAN 'Programming Protocol' (see Appendix A) has been defined by the development team of SWAN with the aim to reduce maintenance costs and to improve the readability of the source code of SWAN. A developer of new SWAN subroutines has to conform to certain guidelines for structure, contents and implementation in the source code. Special attention has to be given to:

- Subroutine headers;
- Names of variables and subroutines;
- Documentation of variables;
- declaration of variables;
- Conformance to ANSI FORTRAN77 (and ANSI FORTRAN 90 in the future);
- Change management;
- Documentation of the subroutine (e.g. purpose, method and structure);
- Program structure (SWAN pre- and post-processing and computational parts).

2.2 Programming practice

The actual source code of SWAN however does not always comply completely to the above given guidelines. The reason for this is that developers of SWAN are mostly engineers who do not have a specific background in Information Technology and changes in the source code have not been implemented / documented in the headers.

Further the structure and design of the SWAN source code (pre- and post-processing) are based to a large extent on the HISWA model developed in 1985. Subroutines from HISWA have been re-used for SWAN and are coded before the definition of the SWAN Programming Protocol. Therefore a number of subroutines in SWAN have become more or less out-of-date (e.g. the POOL-structure for memory allocation).

2.3 Limitations

This quality assessment is meant only as a 'pilot study' to gain insight in the overall quality of SWAN. Therefore it is decided that only (a part of) the source code of SWAN (version 40.00) and its (internal) documentation (headers and system documentation) will be checked on all levels against a set of criteria:

- INAR2D.FOR (reads bottom topography and other data fields);
- SSURF.FOR (computes dissipation by surf breaking);
- SWCOMP.FOR (main subroutine of the computational part);
- SWOEXF.FOR (computes wave driven forces).
It is noted here that only four selected subroutines have been considered. Therefore no information or conclusions can be given/provided with respect to the total program structure, the data structures used (pool mechanism), error handling etc. in this report.

This assessment will evaluate these four subroutines for the following aspects:

- Internal documentation (header);
- Structure and structure description (Structured English);
- Source code;

and levels:

- Marginal check;
- Functional check.

First a marginal check will be done for each of the subroutines, followed by a more detailed evaluation, resulting in additional comments. After that a description will be given of the actions needed for transformation of the subroutines for compliance with standard ANSI FORTRAN90.

### 2.4 Audit aspects and criteria

The SWAN ‘Programming Protocol’ is used for reference, and some programming tools are used for code analysis (see Section 2.5).

Some remarks can be made here about the SWAN ‘Programming Protocol’:

- The protocol deals only with system- and source code documentation, source code and version numbering; no recommendations are given for the user manual and testing;
- Beside the example no additional information is provided in the protocol with respect to the layout of the source code (i.e. rules on how to program etc.);
- The SWAN Programming Protocol gives several recommendations instead of rules and contains some inconsistencies that should be removed;
- No clear criteria are given with respect to accuracy, effectiveness and efficiency of source code, so only general remarks can be formulated in this assessment;
- A distinction is made between ‘old’ and ‘new’ subroutines for documentation directives, but the protocol gives also new mandatory directives for existing subroutines;
- Fortran77 is recommended; the IMPLICIT NONE statement however is an extension;
- The same variable name must be used if already defined elsewhere in SWAN with the same meaning; it is not clear how this can be achieved. A tool is needed to generate such a data dictionary.

The Programming Protocol is meant to take care of certain quality aspects on the level of the source code:

- The use of standard Fortran makes the code portable to different type of computer platforms;
- The IMPLICIT NONE statement forces the programmer to explicitly define all variables so the compiler can perform a more rigorous check.
• Definition and documentation of all variables in the header(s) (part 4 to 6) is an important part of the system documentation;
• Subroutine headers are (together with common blocks) interfaces between the different modules in a complex program. The description of these interfaces in part 4 and 7 of the header is a vital piece of documentation of (global) data exchange between the different subroutines in the program;
• The functional relations between subroutines are described in part 8 and 9 and is an important part of the documentation of the structure of the system. The function and internal structure of a subroutine is described in part 2, 3 and 12;
• For maintenance (change management) the administrative parts 0 and 1 are needed.

In the next chapters first a marginal inspection regarding these quality aspects is reported, followed by a more detailed evaluation. The topics reviewed are given there, together with the findings. Also some recommendations will be given for improvement of the Programming Protocol, the source code and its documentation.

2.5 Tools used for inspection

The subroutines will be checked visually (source code inspection) and with help of the following software tools:

• FORCHECK of Leiden University, Version 12.63 as a (standard) tool for analysis of Fortran source code;
• DIGITAL Visual Fortran Version 6.0 compiler to check the code;
• VAST/77to90 from Pacific-Sierra Research; a demo version (V 4.4 D) for translating / upgrading existing Fortran77 programs to Fortran90/95.

It is noted here that also other software analysis tools are available on the market to access and analyse the quality of software (e.g. QA-Fortran or McCabe). The selection of these tools depends on the desired level of detail, the way it fits in the organisation of software development and maintenance and the available budget.
3 Marginal check source code

This check is meant to verify whether the general guidelines for the development of source code are fulfilled. In the SWAN Programming Protocol (Appendix A) this is formulated as:

All variables, even simple counters or indices need to be documented. Common variables are documented in the include files, other variables in the subroutine headers. There are four levels of other system documentation in SWAN. Three of these levels are in the subroutine headers:

- **Purpose**: A short description what the subroutine should do;
- **Method**: An extensive description regarding the method that is used to reach the purpose, with reference to the literature if applicable;
- **Structure**: The source code written in structured English (refer to the source code for examples).

Extensive documentation should be included by interposing the source code with comments. A developer should be able to understand what a subroutine does after reading 'purpose' and 'method'.

In the first paragraph the presence of the prescribed header is verified. In the following paragraphs the documentation of the subroutine structure (design) in part 12 is examined, followed by a check of implementation in the source code (part 13).

3.1 Subroutine documentation (header)

The SWAN Programming Protocol prescribes the structure and format of the subroutine headers and gives an example (see Appendix A). A distinction is made between ‘old’ and ‘new’ subroutines: new subroutines must comply with the directives, old routines must be adapted in future releases.

In this paragraph it is verified whether:

- The header contains all necessary documentation parts (1 - 13);
- All required information is provided and correct.

The protocol does not explicitly define the type of information needed, so here it is only checked if any information is given and if this information is correct.

In the following the remarks will be given for each of the four subroutines.

**INAR2D.FOR**

The subroutine header is **not** conform the guideline; only some parts in the header are given. It is found that not all the relevant information is given or correct:
Only part 0 to 2, 4 and 8 are provided;
- Argument variables are described as input, but used as output (part 4);
- Subroutines STRACE, FOR and MSGERR are called but not mentioned (part 8);
- Common blocks are included (SWCOMM4.INC and OCPCOMM4.INC) but not described;
- None of the items, declared in the include files OCPCOMM1.INC and OCPCOMM3.INC is used;
- The subroutine header starts with a software agreement that is not described in the Programming Protocol.

**SSURF.FOR**

The subroutine header is conform the guideline; all parts in the header are given. It is found however that not all the relevant information is given or correct:

- The subroutine STRACE is called but not mentioned;
- Local variables are described but not used (FRDEP, NORM, RES, RESD);
- Local variables are mentioned but not explained (SURFA0) AND SURFA1);
- Common blocks are included (SWCOMM4.INC and OCPCOMM4.INC) but not described.

Remark: I/O is implemented for testing; it is recommended to include in the protocol the description of I/O in the header and guidelines to include test output.

**SWCOMP.FOR**

The subroutine header is set up conform the guideline; (nearly) all parts in the header are given. However a lot of the information/documentation given is incomplete, not to the point or incorrect, so it is assumed that this is a subroutine adjusted to the new guidelines:

- In the top of this subroutine additional comment is given that should have been placed elsewhere (e.g. part 2 or 11);
- The IMPLICIT NONE statement is missing;
- Purpose and method (part 2 and 3) are described for the overall functionality of SWAN, not for this subroutine;
- Not all the subroutine arguments are described and declared in part 4; declarations are also given in part 13 (source text);
- In part 4 only one local variable is described and declared; all other declarations are given in part 13 (source text);
- In part 4 common variables are described; the protocol explicitly locates documentation of common variables in the include files;
- None of the variables in the common blocks in the include files OCPCOMM1.INC, OCPCOMM2.INC and OCPCOMM3.INC are used;
- Part 5 (parameter variables) is missing;
- The data pool subroutines DP*, the subroutines BND4WW, FAC4WW, SETUPP and STRACE and function IADRS are not mentioned in part 8; none of them is described;
- The documentation in part 11 (remarks) and 12 (structure) is copied from a subroutine called SCOMPU and so not for SWCOMP. Therefore the contents does not fit the actual source code. The structure of the subroutine is not documented.
SWOEXF.FOR

The subroutine header is conform the guideline; all parts in the header are given. It is found that not all the relevant information is given or correct:

• Function EQREAL is used but not described (part 8);
• Argument variables are defined 'local' (in part 4); this might be confused with part 6, so better use 'work' as a description.

3.2 Program structure (structured English)

In part 1 to 11 of the header documentation the purpose and methods used in the subroutine must be described. The next step in the development of a subroutine is the design of the general structure. This structure is documented in part 12 of the header; here the algorithms for the various functions are described in so called 'structured English' (also called 'pseudo code'). In this paragraph this part it is checked for each of the four subroutines.

Only in the subroutines SSURF and SWOEXF the purpose and methods are documented and can be used as documentation / specification of the functionality's in the subroutines (taken into account the remarks in the preceding paragraph). The structure of these two (computational) subroutines is clear and straightforward (SSURF is a small subroutine). The use of IF-THEN-ELSE-statements is clear and additional comments are given in the code.

The structure of the subroutine INAR2D is clear from the source code (but the method used is not documented in the header). It reads data from a data file.

The documentation of subroutine SWCOMP appears to be inadequate or incorrect. The source code however has a clear structure, so the purpose and method of this subroutine can be derived from the code. The subroutine performs several different tasks: printing of variables, memory management and (model) computation. The length of the subroutine is very long which makes it difficult for a programmer to get a good overall picture quickly.

3.3 Source code

3.3.1 Methods implemented

In part 3 and 12 of the header the method and structure of a subroutine are documented. In part 13 (the remaining part) of the subroutine the accompanying source text is given. In this paragraph it is checked whether the documented methods in the header are implemented (coded) in executable statements (as far as documentation is available).

Only in the subroutines SSURF and SWOEXF the purpose and methods are well documented and specified; for these subroutines it can be concluded that all methods documented are implemented (coded) in executable statements with sufficient additional comments. The purpose and method are not documented for the subroutines INAR2D and SWCOMP so it cannot be checked whether the implementation is correct. Additional documentation of the
purpose and the methods implemented is necessary and may be derived from the source text.

3.3.2 Standard FORTRAN

The Programming Protocol recommends to use the ISO FORTRAN77 standard, without the features that became obsolescent with the introduction of FORTRAN90. The software analysis tool FORCHECK can be used to check for standard FORTRAN.

FORCHECK generates analysis messages flagged with an 'I' are informative, with a 'W' are warnings, those flagged with an 'E' are errors. Informative messages hold no conflicts with the Fortran standard. Warnings indicate the usage of extensions to the standard. Error messages will arise when the Fortran standard has been violated.

In the following the messages generated by FORCHECK for the four subroutines are given together with a short evaluation (number of occurrences x [message type] and description). General remarks in this analysis:

- The use of IMPLICIT NONE (used in SSURF and SWOEXF as recommended in the Programming Protocol) is non-standard Fortran77 syntax;
- '&&' is used as a continuation character but not in the Fortran77 character set;
- The use of lower case character(s) (in all routines except SSURF) is not standard Fortran77;
- The character declarations used (mainly in included commons) are obsolete in Fortran77.

**SSURF.FOR**

messages presented:
17x [ 49 W] continuation character not in Fortran character set
1x [ 60 W] fixed source form used
1x [ 95 W] nonstandard Fortran syntax
6x [ 227 I] extension of COMMON
2x [ 344 I] implicit conversion of constant (expression) to higher accuracy

number of warnings: 19
number of informative messages: 8

No additional comments.

**SWCOMP.FOR**

messages presented:
2x [ 1 I] (too many messages for this statement or argument list)
8x [ 49 W] continuation character not in Fortran character set
1x [ 50 W] lower case character(s) used
3x [ 58 I] none of the items, declared in the include file, is used
1x [ 60 W] fixed source form used
3x [ 71 W] nonstandard Fortran statement
24x [ 96 W] obsolescent Fortran feature

1 Options used in FORCHECK:
   -- ftn compiler emulation
   NOFF90/NOFF/FO/NOE/NOFF/NOINR/NOF95/14/SB/S/NOSH/SI/TR/NORI/PR/CO/AR/NORI
   /NOLG/NOID/NO
   and with local options: /NOCN/DD/NOE/DDX/FOE/FF77/NOFF90/NOFF
   /NOE/NOID/NOINR/NOF95/14/SB/S/NOSH/SI/NORI/TR/INF/NA
35x[110 W] name or operator too long
85x[227 I] extension of COMMON
2x[323 I] unreferenced
1x[345 I] implicit conversion to less accurate data type
3x[465 E] statement label expected

number of error messages: 3
number of warnings: 145
number of informative messages: 93

Summary:

- In 3 do loops a Fortran90 syntax has been used, resulting in 3 warnings for non-standard Fortran and 3 error messages for missing labels;
- Many warnings are given for operator names longer than the required maximum of 6 characters (which is a potential risk in Fortran77).

**INAR2D.FOR**

messages presented:
22x[49 W] continuation character not in Fortran character set
1x[ 50 W] lower case character(s) used
2x[58 I] none of the items, declared in the include file, is used
1x[60 W] fixed source form used
17x[96 W] obsolescent Fortran feature
13x[227 I] extension of COMMON
6x[323 I] unreferenced

number of warnings: 41
number of informative messages: 21

Summary:

- computed goto’s are an obsolescent Fortran77 feature.

**SWOEKF.FOR**

messages presented:
33x[49 W] continuation character not in Fortran character set
1x[ 60 W] fixed source form used
1x[ 95 W] nonstandard Fortran syntax
10x[ 96 W] obsolescent Fortran feature
1x[110 W] name or operator too long
63x[227 I] extension of COMMON

number of warnings: 46
number of informative messages: 63

Summary:

- a warning is given for an operator name in included common (SWCOMM1.INC) longer than the required maximum of 6 characters (a potential risk in Fortran77).

**Conclusion**

Although the Programming Protocol recommends to use standard Fortran77 and demands Fortran77 for variable- and subroutine names, none of the four subroutines fully complies to ANSI Fortran77. The use of a tool like FORCHECK will generate sufficient information to the programmer for Fortran77 (and Fortran90) compliance.
4 Functional check source code

In this chapter the subroutines are evaluated in more detail. The methods implemented are checked for documentation (if available), design and coding. To what extend the subroutines are accurate, effective and efficient is hard to judge, because no clear criteria are available in the SWAN Programming Protocol.

A programmer has many options to implement a certain functionality and the actual choice depends on various requirements like accuracy, efficiency, clarity of coding, simplicity of algorithm, performance etc. Whether the methods implemented in a subroutine are correct, effective and efficient is difficult to judge without knowledge about the backgrounds of the various choices of implementation. Therefore these arguments are usually incorporated in the subroutine documentation. With this in mind it will be clear that in the next paragraphs no absolute statements can be made so only general remarks are given.

4.1 Internal documentation (header)

Fundamental for the design of a subroutine is the description of its purpose and function and the choice of the method(s) for the implementation of this functionality. Therefore it is important that the methods used are documented accurately in the header of the subroutine (see Chapter 3). It is also important that there is a one to one correspondence between documentation and implementation; inconsistencies can lead easily to errors during development but also during maintenance.

The following (additional) remarks about the header can be made for the four subroutines (see Section 3.3.1 for method description):

INAR2D.FOR

No documentation of the methods used is given in the header. This subroutines appears to read data files so information about the file content, file types and (machine-dependent) formatting is essential.

SSURF.FOR

Although the header documentation contains all required parts, it is not (completely) accurate. In part 6 for example local variables are mentioned that are not used (anymore?) and in part 9 an unknown subroutine SOURCE is mentioned. The method description contains the derivation of the formulas used in the computation but there is no direct link from this part of the documentation to the formulas implemented in the executable statements; in part 12 (structure) the formulas used should be mentioned.
**SWCOMP.FOR**

The documentation of subroutine SWCOMP is not adequate and incorrect (see preceding chapters and paragraphs).

**SWOEXF.FOR**

This subroutine contains a good balance between documentation and executable statements. The method description is compact and further clarified in part 11 (remarks). The source text has sufficient interline documentation.

### 4.2 Structure (structured English)

Part 12 of the header should contain a description of the structure of the subroutine and the algorithms used. This paragraph verifies whether the documented methods have been worked out in a accurate, effective and efficient design and algorithms.

**INAR2D.FOR**

The subroutine header is **not** conform the guideline. There is no documentation of subroutine structure and design.

**SSURF.FOR**

In part 12 only a brief description of the structure is given. This description is not accurate and incomplete (e.g. some variables are given in the parameter list and not provided by a call to subroutine SDISPA). Because the documentation and implementation are not consistent even a small routine like SSURF takes much time to understand.

**SWCOMP.FOR**

In section 12 only the structure of the computational part of the subroutine is documented. The structure of this computational part is conform the selected numerical procedure.

**SWOEXF.FOR**

Section 12 contains a brief description of the subroutine structure which is clear and straightforward.

### Remark

The level of description of the design and structure of a subroutine depends strongly on the requirements defined for this part of SWAN. In most cases a general description of the work flow is sufficient for a programmer if additional documentation is available in the source code. Here the (special) choices made by the programmer should be documented. Only
additional requirements about accuracy, effectiveness and efficiency are documented in the 
header should result in additional documentation in part 12 and the source code.

4.3 Source code

In this paragraph for each subroutine the implementation (coding) of a documented method 
is evaluated in more detail. It is also examined whether debugging-facilities are formulated 
(note: this is not required in the Programming Protocol).

A quality assessment is not suitable to test whether the source code is accurate; this must be 
verified by series of testing, starting from testing by the programmer of all functionality's 
specified in the design document and implemented in the various subroutines to overall 
system tests. An important role in testing play the criteria that must have been defined for 
acceptance of the code and that should be based on the requirements; the more stringent 
these criteria, the more important is testing (which can vary roughly between 20% up to 
80% of the total time for development).

For coding in Fortran77 general guidelines are available in literature (see for example [1]). 
General remarks can be given when following such guidelines, for example (see e.g. 
INAR2D):

- Obsolete Fortran77 statements are used e.g. (computed) GOTO'S.
- Exiting of the subroutine is not at the end of the subroutine (several RETURN 
  statements);
- Unit numbers are used in a test command;
- Fixed values are used instead of parameter values in tests;
- Labels are not in ascending order;
- Labels are given to executable statements.

For robustness additional programming guidelines can be formulated (e.g. with respect to 
the use of IF-THEN-ELSE constructions, 'defensive programming', error handling etc.).

All four subroutines could be compiled with the DIGITAL Visual Fortran Version 6.0 
compiler, using the default compiling options (accepting the several Fortran90 statements!).

    INAR2D.FOR

This subroutine is not conform the guideline; the source code however has a clear structure 
and comment lines are added for each subsection. No debugging-facilities are formulated.

    SSURF.FOR

There is little documentation between the source text; it is advised to copy the implemented 
formulas from the header documentation to the source text. Variables used in the 
documentation do not have the same name in the source text! The use of SAVE and DATA 
statement is unclear. Some output arrays (DISSC*) are filled, but no documentation is given. 
SSURF contains some 'dead code'.
**SWCOMP.FOR**

The source text of this subroutine is very long and contains several parts for different tasks; it is advised to use separate subroutines, for example for printing of the various parameters. The source text for the computational part is consistent with the documentation in section 12 and contains also additional (and adequate) documentation. Some parts however are not documented (e.g. the call to PLTSRC) and some comments are confusing (e.g. if ... NSTATC=0 GOTO 550). This subroutine contains many GOTO and RETURN statements and many labels (several are not used); this conceals the structure of the routine and is confusing. Some common blocks are included but the variables are not used; this does not contribute to clarity.

**SWOEXF.FOR**

The subroutine structure is implemented in the source text as described in the header.
5 FORTRAN90

The Programming Protocol recommends now to use the ISO FORTRAN77 standard, without the features that became obsolescent with the introduction of FORTRAN90. In this chapter the consequences of a transition of SWAN to FORTRAN90 is evaluated for obvious reasons: most of the present compilers are for FORTAN90 and this version of the language offers many additional features that are very useful for software applications like SWAN.

5.1 FORTRAN90 compliance

Because FORTRAN77 is still a subset of the ANSI FORTRAN90 standard the present software will comply (by definition) to the ANSI FORTRAN90 standard. The present guideline for SWAN demands that no obsolete (FORTRAN77) features must be used in the source. With FORTRAN90 additional recommendations are given for adapting the present code (see e.g. [2]).

The software analysis tool FORCHECK can be used to check the present subroutines for standard FORTRAN90 (and obsolete features). The option used in the analysis is now FORTRAN90 (in stead of 77) conformance.

The messages generated by FORCHECK for the four subroutines are given. General remarks:

- The use of IMPLICIT NONE is standard FORTRAN90;
- The continuation character "&" is in the FORTRAN90 character set;
- Lower case character(s) are allowed in FORTRAN90;
- The character declarations used (mainly in included commons) are also obsolete in FORTRAN90;
- For do loops a FORTRAN90 syntax can be used;
- Operator names longer than 6 characters are allowed in FORTRAN90.

**SSURF**

messages presented:
1x[ 60 W] fixed source form used
6x[227 I] extension of COMMON
2x[344 I] implicit conversion of constant (expression) to higher accuracy

number of warnings: 1
number of informative messages: 8

**SWCOMP**

1x[ 45 W] too many continuation lines
3x[ 58 I] none of the items, declared in the include file, is used
1x[ 60 W] fixed source form used
24x[ 96 W] obsolescent Fortran feature
85x[227 I] extension of COMMON
2x[323 I] unreferenced
1x[345 I] implicit conversion to less accurate data type

number of warnings: 26
number of informative messages: 91
INAR2D

messages presented:
2x [58 I] none of the items, declared in the include file, is used
1x [60 W] fixed source form used
17x [96 W] obsolescent Fortran feature
13x [227 I] extension of COMMON
6x [323 I] unreferenced

number of warnings: 18
number of informative messages: 21

SWOEXF

messages presented:
1x [60 W] fixed source form used
10x [96 W] obsolescent Fortran feature
63x [227 I] extension of COMMON

number of warnings: 11
number of informative messages: 63

Fortran90 has introduced more features, so the analysis now shows that many former messages have disappeared; the remaining messages have already been explained in the preceding chapters.

5.2 Transition to FORTRAN90

An important question for the further development and maintenance of SWAN is whether the existing source code has to be changed to comply to the ANSI FORTRAN90 standard. The answer is simple: no (see previous paragraph). Of course there are several recommendations for a transition to FORTRAN90 (see for example [2]).

In a joint project of Rijkswaterstaat and WL | Delft Hydraulics several possibilities for the migration of the existing modelling systems Simona and Delft3D from FORTRAN77 to FORTRAN90 have been investigated. The most important subjects are the conversion of existing array declarations in FORTRAN77 ‘data structures’ to more flexible FORTRAN90 ‘data structures’ and the use of FORTRAN90 array operations. In report [3] attention is also given to the need for FORTRAN90 programming standards and directives and the possibilities in Fortran90 to improve maintainability of the software (e.g. use of modules and interfaces).

A summary of the recommendations in [3] is (see this report for details):

- Replace COMMON blocks by modules;
- Use modules to combine subroutines and functions;
- Use explicit interfaces;
- Use the IMPPLICIT NONE statement for explicit declarations (already in the SWAN Programming Protocol);
- Use the possibilities for explicit array declaration (and dimensions) for robustness;
- Use dynamic array declarations for work arrays;
- Don’t use GOTO statements except for error handling;
- Don’t use arithmetic IF’s and computed GOTO’S;
- Never use the EQUIVALENCE statement.
It is concluded that these FORTRAN77 programs can be compiled with Fortran90 without hardly any noticeable differences in operation. The various libraries can be combined.

Several of these recommendations will affect the current source code of SWAN. A gradual transition of SWAN to Fortran90 is possible due to the upwards compatibility of FORTRAN90:

- The SWAN Programming Protocol must be extended with restrictions for FORTRAN90 constructions;
- (new) FORTRAN90 subroutines with a free source form can be allowed.

A more rigorous recommendation (and outside the scope of this assessment) for the use of FORTRAN90 in SWAN is:

- Replace the memory management routines (POOL-structure) routines by FORTRAN90 features in future. This will affect the data structures used (like the common blocks and use of equivalence statements) so this is a major change, affecting the whole SWAN program. On the other hand there will be a reduction in maintenance (and routines) and compilers do much of the data type checking;

In the transition to FORTRAN90 several of the above recommendations can be combined with the adjustment of the documentation of the subroutine. In general a substantial part of software development is needed for documentation (user manual and system documentation). Therefore it can be expected that for improvement of the internal documentation of the given four subroutines an effort is needed in the order of a few days including testing.

The introduction however of new (data) structures and modules will have a bigger impact on the source code of SWAN. Although conversion tools are available hardly any experience is available for transition to FORTRAN90 of numerical modelling systems like SWAN. Some experience has recently become available in the pilot study [3].

It is emphasised that a major restructuring or reprogramming of the existing source code of SWAN can have an negative effect on the reliability and accuracy of the code; a major effort for testing the SWAN system will then be necessary. The existing test set for SWAN will therefore be essential.
6 Conclusions and recommendations

In this pilot study for a quality assessment of SWAN the examination of the four subroutines results in the following conclusions and recommendations:

Conclusions

1. Only one of the four subroutines is fairly well documented according to the guidelines in the SWAN Programming Protocol. The contents of the required documentation is insufficient/incomplete for two routines. One routine does not comply to the Programming Protocol;
2. The SWAN Programming Protocol gives only a set of recommendations instead of rules and gives a programmer some freedom in interpretation;
3. The header of a SWAN subroutine and all parts of it is a very important part of the system documentation and should be complete, accurate and refer only to the subroutine itself. It may not contain redundant information;
4. Non-standard Fortran77 features and obsolete statements (in Fortran90) are used many times in the four subroutines;
5. With the above results the evaluation of only four subroutines (out of approximately 150) is insufficient for understanding the software quality of the total SWAN system.

Recommendations

Programming Protocol:
1. The SWAN Programming Protocol needs to be corrected and extended with additional requirements to make it more clear and unambiguous;
2. The source code example in the Programming Protocol is not clear and should be replaced with a real example (that complies fully with the protocol);
3. The SWAN Programming Protocol should give specific rules for layout of the source code and for documentation;
4. The criteria for incorporation of source code in SWAN must be described very clearly in the SWAN Programming Protocol;
5. Include references to Fortran books in the Programming Protocol;
6. Include in part 0 also the institute of the authors;
7. Give recommendations for the use of compilers (and flags) to SWAN developers.

Current source code:
1. Obsolete FORTRAN77 statements (arithmetic IF’s, computed GOTO’s and GOTO), should be replaced as much as possible by other available statements in FORTRAN77;
2. Mandatory requirements in the Programming Protocol must be checked strictly;
3. Define a tool to check the requirements for source code and make it available to SWAN developers;
4. Use always the Fortran code analysis tool FORCHECK.
FORTRAN90:
1. Use always the (FORTRAN90) IMPLICIT NONE statement for explicit declarations;
2. Evaluate the recommendations for Fortran90 in Chapter 5 and extend the SWAN Programming Protocol for the selected items. Here results from ongoing developments for the modelling systems Simona and Delft3D can be used;
3. Migrate gradually to Fortran90, working from the top (main) program down to the lower subroutines (leaving them unchanged as far as possible), and for new subroutines to be developed.

General:
1. A global evaluation of the total SWAN program with respect to program structure, data structures used (pool mechanism), error handling etc. is also necessary for a good understanding of the software quality of SWAN;
2. All SWAN subroutines must be evaluated and classified (good, moderate, poor). Then the effort needed for the required adjustments in the source code can be made;
3. All SWAN subroutine headers should be made complete and accurate without redundant information. This activity could be included in further development or maintenance of the software, or defined within a separate project;
4. Investigate the possibilities and impact of replacing the memory pool in SWAN with new Fortran90 functionality’s for data handling/management.

Here only the source code and its (internal) documentation has been reviewed because it is a very important aspect for ‘software quality’. However it is emphasised that for modelling systems like SWAN the control of the processes of software development and software maintenance is very important for maintaining and improving the existing level of ‘software quality’ (ISO9000). Therefore it is necessary that besides the SWAN Programming Protocol also the existing procedures for maintenance and incorporation of new or changed parts of the SWAN software are clearly documented for the SWAN developer. An important part in this procedure is testing of the software: for software quality several test-tools can be used and test sets with detailed and global tests should be used for testing the implemented functionality’s.
7 References


A  SWAN Programming Protocol

The following description of the SWAN Programming Protocol can be found on the SWAN Homepage (see http://swan.ct.tudelft.nl/protocol.htm):

**SWAN programming protocol**

*Introduction*

If a developer aspires to have his SWAN development(s) included eventually in the authorized SWAN version, the following guidelines should be followed

*FORTRAN77 compliance*

To avoid problems with different compilers it is recommended to use the ISO FORTRAN77 standard, without the features that became obsolescent with the introduction of FORTRAN90. The reason to avoid obsolescent features is that they might be deleted from a future FORTRAN standard. A list with obsolescent features can be found in many FORTRAN90/95 reference books.

*Version numbers*

*Obtaining*

To distinguish between the various experimental versions of SWAN it is recommended for a developer to obtain a version number from Delft University of Technology (swan.info@ct.tudelft.nl) (in the early stage of development. Without proper use of a unique version number it is impossible for Delft University to include developments from other institutes into the authorized version of SWAN.

*Usage*

The version number is placed in the columns 75 and beyond in the source code. In FORTRAN everything placed in column 73 and beyond is treated as comment, so is the version number. The version number is placed at each new or changed line of source code in existing subroutines. New subroutines are placed in a separate file, there the version number is placed only in the header. Changes are documented in the header of each subroutine with version number, author, month, year and description as in the provided example. When using an experimental version of SWAN the version number (VERNUM) in swanmain.for should be modified. This is to distinguish output from experimental versions from other versions.

*Subroutine headers*

The structure and format of the subroutine header is given. For new subroutines this format is required. Existing subroutine headers in SWAN will be adapted to the new format in future releases.

*New subroutines*

A new subroutine should have an IMPLICIT NONE statement directly after the SUBROUTINE statement. Although SWAN does not use the IMPLICIT NONE statement yet, it is required for new subroutines.

*Declaration of variables*

All variables are declared explicitly in both existing and new subroutines. The variables are declared and documented at the appropriate places in the subroutine headers. The following variables are distinguished in the subroutine header: local, argument and parameter variables; common variables are declared in the include-files. Further, functions need to be declared in the "subroutines used" section of the header.
Variable- and subroutine names
Variable names should follow the FORTRAN77 standard and should be unique in the first 6 characters. All common variable names are reserved, as are the names used by FORTRAN itself. The use of reserved names should be avoided. It is required to use mnemonic names, i.e. names that indicate the meaning of a variable or the function of a subroutine. If in SWAN already a variable name exists, then the same variable name must be used in all subroutines if this variable has the same meaning.

Documentation of variables
All variables, even simple counters or indices need to be documented. Common variables are documented in the include files, other variables in the subroutine headers.

Other documentation
There are four levels of other system documentation in SWAN. Three of these levels are in the subroutine headers:
- Purpose: A short description what the subroutine should do;
- Method: An extensive description regarding the method that is used to reach the purpose, with reference to the literature if applicable;
- Structure: The source code written in structured English (refer to the source code for examples).

Extensive documentation should be included by interposing the source code with comments. A developer should be able to understand what a subroutine does after reading "purpose" and "method".

Coding
Adding statements in existing subroutines
Each new statement should have a version number in columns 75 and beyond. The authors and updates sections in the subroutine headers must be updated, including the version numbers, as well as the other relevant sections. Do not forget the "structure" section.
New variables are declared and documented. Calls to new subroutines need to be documented in both the routine that is calling as well as the routine that is called!

Changing statements in existing subroutines
The old statement is commented by inserting CDEL in the first 4 columns of each line. Further the version number is modified (now in columns 79 and beyond). For Delft University CDEL indicates that a statement should be deleted. After this, a new statement can be added as described above that replaces the old one.
Last updated 12 March 1998

Example:

```c
C*******************************************************************************
C                        * SUBROUTINE SWSTUP(RS,WS)                        *
C*******************************************************************************
C IMPLICIT NONE
C INCLUDE 'swcomm3.inc'
C 0. Authors
C 40.12: IJsbrand Haagsma
C 42.16: Nico Booij
C 1. Updates
C 40.12, Jan. 97: New subroutine
C 42.16, Dec. 97: Added wave setup
C 2. Purpose
C Calculates wave induced setup
C 3. Method (updated 42.16)
C Extensive description of the method that is used to calculate setup, what is assumed as input and
```
how is the output calculated

4. Argument variables
RS    input: Radiation stresses
WS    output: Wave setup
REAL  RS(*),  WS(*)

5. Parameter variables
--

6. Local variables
I     counter
J     counter
INTEGER I, J

7. Common Blocks used
COMMON / COMPD / *** pointers in the pool array

8. Subroutines used
SUB1  *** description
FUNC1 *** description
LOGICAL FUNC1

9. Subroutines calling
SUB1  *** description

10. Error messages
--

11. Remarks
---

12. Structure
See examples in SWAN

13. Source text
CALL SUB1(RS)
.
RETURN
END
B SWAN source code

B.1 Subroutine INAR2D.FOR

******************************************************************************
* SUBROUTINE INAR2D (ARR, MXA, MXA, NDSL, NDSD, IDFM, RFORM, 
& IDLA, VFAC, NHED, NHEDF) 40.00
* 40.00
******************************************************************************
C
C INCLUDE 'opccomm1.inc' 30.74
C INCLUDE 'opccomm2.inc' 30.74
C INCLUDE 'opccomm3.inc' 30.74
C INCLUDE 'opccomm4.inc' 30.74
C
C --------|---------------------------|--------------------------|
C | Delft University of Technology | |
C | Faculty of Civil Engineering, Fluid Mechanics Group | |
C | P.O. Box 5048, 2600 GA Delft, the Netherlands | |
C | |
C | Authors : R.C. Ris, N. Booij, R. Padilla-Hernandez, | |
C | L.H. Holthuijsen | |
C --------|---------------------------|--------------------------|
C
C
C BY USING THE SWAN SOFTWARE, YOU ARE CONSENTING TO BE BOUND BY 
C THIS AGREEMENT.
C
C Delft University of Technology grants you a non-exclusive license 
C to use the SWAN Software free of charge.
C
C DISCLAIMER OF WARRANTY.
C
C The Swan Software is provided on an "AS IS" basis, without 
C warranty of any kind, including without limitation the warranties 
C of merchantability, fitness for a particular purpose and 
C non-infringement. The entire risk as to the quality and 
C performance of the Software is borne by you. You must determine 
C that the Software sufficiently meets your requirements.
C
C LIMITATION OF LIABILITY.
C
C UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, TORT, CONTRACT, 
C OR OTHERWISE, SHALL DELFT UNIVERSITY OF TECHNOLOGY BE LIABLE TO 
C YOU OR ANY OTHER PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, 
C OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT 
C LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER 
C FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES 
C OR LOSSES. IN NO EVENT WILL DELFT UNIVERSITY OF TECHNOLOGY BE 
C LIABLE FOR ANY DAMAGES, EVEN IF DELFT UNIVERSITY OF TECHNOLOGY 
C SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES, OR 
C FOR ANY CLAIM BY ANY OTHER PARTY. YOU AGREE TO INDEMNIFY AND HOLD 
C DELFT UNIVERSITY OF TECHNOLOGY HARMLESS WITH RESPECT TO ALL 
C CLAIMS BY THIRD PARTIES ARISING OUT OF YOUR USE OF THE RESULTS 
C OR OPERATION OF THE SOFTWARE.
C
C TITLE.
C
C Title, ownership rights, and intellectual property rights in the
Software shall remain with the authors of the Software. The
Software is protected by the copyright laws and treaties. Title
and related rights in the content accessed through the Software
is the property of the applicable content owner and may be
protected by applicable law. This License gives you no rights to
such content.

MISCELLANEOUS.
You may not remove any proprietary notices or labels on the
Software, nor remove this disclaimer. If any provision of this
Agreement is held to be unenforceable, such provision shall be
reformed only to the extent necessary to make it enforceable.

This Agreement shall be governed by the law of the Netherlands.

0. Authors
30.72: IJbrand Haagsma
30.74: IJbrand Haagsma (Include version)
30.82: IJbrand Haagsma
40.00: Nico Booij
34.01: Jeroen Adema

1. Updates
01.05, Feb. 90: Before reading values in the array are divided by VFAC,
in order to retain correct values for points where no
value was given
01.06, Apr. 91: i/o status is printed if read error occurs
30.72, Sept 97: Changed DO-block with one CONTINUE to DO-block with
two CONTINUE's
30.72, Sept 97: Corrected reading of heading lines for SERIES of files
in dynamic mode
30.74, Nov. 97: Prepared for version with INCLUDE statements
40.00, July 98: SWAN specific statements modified
unformatted read: heading lines also read unformatted
distinction between NDS (data file) and NDSL (file list)
30.82, Sep. 98: Added INQUIRE statement to produce correct file name in
case of a read error
34.01, Feb. 99: Introducing STPNOW

2. Purpose
Reads a 2d pool-array from dataset

4. Argument variables
ARR inp/real results appear in this array
MXX inp/int number of points along x-side of grid
MYA inp/int number of points along y-side of grid
NDSL inp/int unit number of the file containing the list of filenames
NDSF inp/int unit number of the file from which to read the dataset
IDRM inp/int format index
RFORM inp/char format used in reading data (char. string)
IDLA inp/int lay-out indicator
VFAC inp/real factor by which data must be multiplied.
NHED inp/int number of heading lines in the file
before each array
NHEDF inp/int number of heading lines in the file (first lines).

8. Subroutines used
LOGICAL STPNOW

CHARACTER RFORM *("'), HEDLIN *80
INTEGER STATUS
REAL ARR(MXX,MYA)
DATA IENT /0/
CALL STRACE (IENT, 'INARZD')
999 IF (NDSF.LT.0) RETURN
40.00
* no reading from file due to open error
*** NUMFIL is the number of that is open in one time step **
NUMFIL = 0

IF (ITEST.GE.100) THEN
  WRITE (PRINTF, 12) MXA, MYA, NDSD, IDFM, RFORM,
  IDLA, VFAC, NHED
  FORMAT (" * TEST INAR2D *", 4I4, 1X, A16, I3, 1X, E12.4, I3)
ENDIF

* Read heading lines, and print the same:

11 IF (NHED.GT.0) THEN
  IF (IDFM.LT.0) THEN
    IF (ITEST.GE.30)
      WRITE (PRINTF, '(13,A)') NHED, ' Heading lines'
      DO 28 IH=1, NHED
        READ (NDSD)
      ENDIF
  ENDIF
  CONTINUE
ELSE
  WRITE (PRINTF, '(A)') ** Heading lines **
  DO 30 IH=1, NHED
  READ (NDSD, '(A80)') HEDLIN
  WRITE (PRINTF, '(A4,A80)') ' -> ', HEDLIN
  CONTINUE
ENDIF

* divide existing values in the array by VFAC

DO 39 IY = 1, MYA
  DO 38 IX = 1, MXA
    ARR(IX,IY) = ARR(IX,IY) / VFAC
  CONTINUE
39 CONTINUE

* start reading of 2D-array

IF (IDFM.EQ.0) THEN
  free format read
100  GOTO (110, 120, 130, 140, 150, 160), IDLA
110  DO 118 IY=MYA, 1, -1
    READ (NDSD, *, END=910, ERR=920, IOSTAT=STATUS)
    (ARR(IX,IY), IX=1,MXA)
  CONTINUE
GOTO 900
120  READ (NDSD, *, END=910, ERR=920, IOSTAT=STATUS)
    (ARR(IX,IY), IX=1,MXA), IY=MYA, 1,-1)
GOTO 900
130  DO 138 IY=1, MYA
    READ (NDSD, *, END=910, ERR=920, IOSTAT=STATUS)
    (ARR(IX,IY), IX=1,MXA)
  CONTINUE
GOTO 900
140  READ (NDSD, *, END=910, ERR=920, IOSTAT=STATUS)
    (ARR(IX,IY), IX=1,MXA), IY=1,MYA)
GOTO 900
150  DO 158 IX=1, MXA
    READ (NDSD, *, END=910, ERR=920, IOSTAT=STATUS)
    (ARR(IX,IY), IX=1,MYA)
  CONTINUE
GOTO 900
160  READ (NDSD, *, END=910, ERR=920, IOSTAT=STATUS)
    (ARR(IX,IY), IX=1,MYA), IY=1,MXA)
GOTO 900
ELSE IF (IDFM.GT.0) THEN
  read with fixed format
200  GOTO (210, 220, 230, 240, 250, 260), IDLA
210  DO 218 IY=MYA, 1, -1
    READ (NDSD, RFORM, END=910, ERR=920, IOSTAT=STATUS)
    (ARR(IX,IY), IX=1,MXA)
  CONTINUE
GOTO 900
220  READ (NDSD, RFORM, END=910, ERR=920, IOSTAT=STATUS)
    (ARR(IX,IY), IX=1,MXA), IY=MYA, 1,-1)
GOTO 900
DO 238 IY=1, MYA
   READ (NDSD, RFORM, END=910, ERR=920, IOSTAT=STATUS)
   \* (ARR(IY,IY), IX=1,MXA)
238 CONTINUE
GOTO 900
240 READ (NDSD, RFORM, END=910, ERR=920, IOSTAT=STATUS)
   \* (ARR(IY,IY), IX=1,MXA), IY=1,MYA)
   GOTO 900
250 DO 258 IX=1, MXA
   READ (NDSD, RFORM, END=910, ERR=920, IOSTAT=STATUS)
   \* (ARR(IY,IY), IY=1,MYA)
258 CONTINUE
GOTO 900
260 READ (NDSD, RFORM, END=910, ERR=920, IOSTAT=STATUS)
   \* (ARR(IY,IY), IY=1,MYA), IX=1,MXA)
   GOTO 900
ELSE
* unformatted read
300 GOTO (310, 320, 330, 340, 350, 360), IDLA
310 DO 318 IY=MYA, 1,-1
   READ (NDSD, END=910, ERR=920, IOSTAT=STATUS)
   \* (ARR(IY,IY), IX=1,MXA)
318 CONTINUE
GOTO 900
320 READ (NDSD, END=910, ERR=920, IOSTAT=STATUS)
   \* (ARR(IY,IY), IX=1,MXA), IY=MYA,1,-1)
   GOTO 900
330 DO 338 IY=1, MYA
   READ (NDSD, END=910, ERR=920, IOSTAT=STATUS)
   \* (ARR(IY,IY), IY=1,MYA)
338 CONTINUE
GOTO 900
340 READ (NDSD, END=910, ERR=920, IOSTAT=STATUS)
   \* (ARR(IY,IY), IX=1,MXA), IY=1,MYA)
   GOTO 900
350 DO 358 IX=1, MXA
   READ (NDSD, END=910, ERR=920, IOSTAT=STATUS)
   \* (ARR(IY,IY), IY=1,MYA)
358 CONTINUE
GOTO 900
360 READ (NDSD, END=910, ERR=920, IOSTAT=STATUS)
   \* (ARR(IY,IY), IY=1,MYA), IX=1,MXA)
   GOTO 900
ENDIF
*** End of data file, in case SERIES next file is opened
** unit = NDSD is closed before the next one is opened
*
910 CONTINUE
CLOSE(NDSD)
NUMFIL = NUMFIL + 1
IF (NUMFIL .GE. 2) GO TO 911
   IF (NDSL.GT.0) THEN
      READ (NDSL, '(A1') FILEMN
      IF (IDFM.NE.-1) THEN
         IOSTAT = 0
         CALL FOR (NDSD, FILEMN, 'OF', IOSTAT)
         IF (STPNOW()) RETURN
      ELSE
         IOSTAT = 0
         CALL FOR (NDSD, FILEMN, 'OU', IOSTAT)
         IF (STPNOW()) RETURN
      ENDIF
      \* Read heading lines, and print these:
      \* 2 IF (NHEDF.GT.0) THEN
      \* IF (IDFM.LT.0) THEN
      \* IF (ITEST.GE.30) WRITE (PRINTF, '(I3,A)') NHEDF,
      \*  ' Heading lines at begin of file'
      \* DO 828 IH=1, NHEDF
      \* READ (NDSD)
      \* 828 CONTINUE
      ELSE
WRITE (PRINTF, '(A)') ' ** Heading lines **'
DO 830 IH=1, NHDEF
  READ (NDSD, '(A80)') HEDLIN
  WRITE (PRINTF, '(A4,A80)') ' -> ', HEDLIN
  CONTINUE
ENDIF
830
ENDIF
GO TO 11
ENDIF

C error message when end of file is encountered
C
911 INQUIRE (UNIT=NDSD, NAME=FILENM)
  CALL MSGERR (2, 'Unexpected end of file while reading ' //FILENM)
  NDSD = 0
  IDLA = -1
C Value of IDLA=-1 signals end of file to calling program
C
GOTO 900
C
920 INQUIRE (UNIT=NDSD, NAME=FILENM)
  CALL MSGERR (2, 'Error while reading file ' //FILENM)
  WRITE (PRINTF, 922) STATUS
  IDLA = -2
C Value of IDLA=-2 signals read error to calling program
C
C Multiply all values in the array by VFAC
C
900 DO 909 IY = 1, MYA
  DO 908 IX = 1, MXA
    ARR(IX,IY) = ARR(IX,IY) * VFAC
  CONTINUE
908
909 CONTINUE

* 990 IF (ITEST.GE.100 .OR. IDLA.LT.0) THEN
    DO 996 IY=MYA, 1, -1
      WRITE (PRINTF, 994) (ARR(IX,IY), IX=1,MXA)
    CONTINUE
994 FORMAT ((1X, 10E12.4))
996 CONTINUE
ENDIF
*
end of subr. INAR2D *
RETURN
END
B.2 Subroutine SSURF.FOR

C******************************************************************************
C                  SUBROUTINE SSURF (MDC, MSC, MSURF, ETOT, HM)
C
C                        PI, QB, SMEBRK, AC2, IMATRA
C                        IMATDA, PSURF, IDCMIN, IDCMAX, PLWRK
C                        ISSTOP, DISSC0, DISSC1, RCGRD, MCGRD, 30.21
C                        ICMAX
C******************************************************************************

C******************************************************************************
C |--------------------------------------------------------------------------|
C |                              Delft University of Technology                |
C |                              Faculty of Civil Engineering, Fluid Mechanics |
C |                              Group                                     |
C |                              P.O. Box 5048, 2600 GA Delft, the Netherlands  |
C |--------------------------------------------------------------------------|
C | Authors: R.C. Ris, N. Boolj, R. Padilla-Hernandez, L.H. Holthuijsen      |
C |--------------------------------------------------------------------------|

C

BY USING THE SWAN SOFTWARE, YOU ARE CONSENTING TO BE BOUND BY
THIS AGREEMENT.

Delft University of Technology grants you a non-exclusive license
to use the SWAN Software free of charge.

DISCLAIMER OF WARRANTY.

The Swan Software is provided on an "AS IS" basis, without
warranty of any kind, including without limitation the warranties
of merchantability, fitness for a particular purpose and
non-infringement. The entire risk as to the quality and
performance of the Software is borne by you. You must determine
that the Software sufficiently meets your requirements.

LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, TORT, CONTRACT,
OR OTHERWISE, SHALL DELFT UNIVERSITY OF TECHNOLOGY BE LIABLE TO
YOU OR ANY OTHER PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL,
OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT
LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER
FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES
OR LOSSES. IN NO EVENT WILL DELFT UNIVERSITY OF TECHNOLOGY BE
LIABLE FOR ANY DAMAGES, EVEN IF DELFT UNIVERSITY OF TECHNOLOGY
SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES, OR
FOR ANY CLAIM BY ANY OTHER PARTY. YOU AGREE TO INDEMNIFY AND HOLD
DELFT UNIVERSITY OF TECHNOLOGY HARMLESS WITH RESPECT TO ALL
CLAIMS BY THIRD PARTIES ARISING OUT OF YOUR USE OF THE RESULTS
OR OPERATION OF THE SOFTWARE.

TITLE.

Title, ownership rights, and intellectual property rights in the
Software shall remain with the authors of the Software. The
Software is protected by the copyright laws and treaties. Title
and related rights in the content accessed through the Software
is the property of the applicable content owner and may be
protected by applicable law. This license gives you no rights to
such content.

MISCELLANEOUS.

You may not remove any proprietary notices or labels on the
Software, nor remove this disclaimer. If any provision of this

Agreement is held to be unenforceable, such provision shall be
reformed only to the extent necessary to make it enforceable.
This Agreement shall be governed by the law of the Netherlands.

```
IMPLICIT NONE

INCLUDE 'swcomm4.inc'
INCLUDE 'ocpcomm4.inc'
```

0. Authors

30.62: IJsbrand Haagsma
30.81: Annette Kieftenburg
30.82: IJsbrand Haagsma

1. Updates

30.62, Aug. 97: Prevented a possible division by zero
30.81, Sep. 98: Derivative of WS adjusted
30.82, Sep. 98: Changed indices of PLWBK-array declaration
30.82, Oct. 98: Made subroutine intrinsic DOUBLE PRECISION

2. Purpose

Computation of the source term due to wave breaking.
White capping is not taken into account

3. Method

The source term for surf breaking is implemented following
the approach of Battjes/Jansen (1978) for the energy dissipation:

\[
\text{Dt} = \alpha \frac{2}{4} Qb \cdot f \cdot Hm
\]

with

\[
f = \frac{1}{2 \cdot \pi}
\]

Now the source term is:

\[
\text{Sbr} = \frac{\text{Sigm} \cdot AC2(ID,IS,IX,IY)}{\text{Etot}}
\]

\[
\text{Sigm} \cdot AC2(ID,IS,IX,IY) \frac{\alpha \cdot \text{SMEBRK} \cdot Qb \cdot Hm \cdot Hm}{8 \cdot \pi \cdot \text{Etot}} = \text{WS} \cdot \text{Sigm} \cdot AC2(ID,IS,IX,IY) = \text{WS} \cdot E
\]

With:

\[
\alpha = \text{PSURF}(1);
\]

\[
\text{SMEBRK} \cdot Qb
\]

\[
\text{WS} = \alpha \frac{1}{\text{Pi} \cdot \text{BB}}
\]

\[
\text{BB} = 8 \text{Etot} / Hm = (1 - Qb) / \ln(Qb);
\]

The local maximum wave height \( Hm \) is computed in subroutine SSDIPA.
The fraction of breaking waves \( Qb \) is calculated in the subroutine FRABRE2
The new value for the dissipation is computed implicitly using
the last computed value for the action density (at the spatial
gridpoint under consideration).

\[
\text{Sbr} = \text{WS} \cdot E
\]
C = WS * Enew + WSD * (Enew - Eold)
C = (WS + WSD) * Enew - WSD * Eold
C = SURF1 * Enew - SURF0 * Eold
C
In order to do this we need the derivative of WS to BB, which will be called WSD.
C
With:
C d Qb = 1
C d BB = (d BB / d Qb)
C
C d Qb = 2
C d BB = ln (Qb)
C
C Qb (1 - Qb) = BB (BB - Qb)
C
Now:
C
SMERK d (Qb / BB)
C
WSD = Alpha ------- -------
C Pi d BB
C
C BB - 1
C BB (Qb - BB)
C
4. Argument variables
C
AC2 input: Action density array
C DISSCO output:
C DISSC1 output:
C ETOT input: Total energy per spatial gridpoint
C HM input: Maximum wave height
C ICMAX input: Maximum number of elements in KCGRD array
C IDCMIN input: Minimum number for counter IDDM
C IDCMAX input: Maximum number for counter IDDM
C IMTDA output: Coefficient of diagonal matrix (2D)
C IMATRA output: Coefficient of righthandside of matrix
C ISSTOP input: Maximum for counter IS
C KCGRD input: Array for indirect addressing
C MCGRD input: Number of gridpoints
C MDC input: Number of points in directional domain
C MDC input: Number of points in frequency domain
C MSURF input: Number of elements in PSURF array
C PI input: Number 3.141
C PLWBRK output:
C PSURF input: Array containing breaker information
C QB input: Fraction of breaking waves
C SMERK input: Mean frequency according to first order moment
C
INTEGER ICMAX, 30.21
C ISTTOP, MCGRD, MDC, MSC, MSURF,
C IDCMIN(MSC), IDCMAX(MSC),
C KCGRD(ICMAX) 30.21
C
REAL AC2(MDC,MSC,MCGRD) ,
C DISSCO(MDC,MSC) ,
C DISSC1(MDC,MSC) ,
C IMATDA(MDC,MSC) ,
C IMATRA(MDC,MSC) ,
C PSURF(MSURF) ,
C PLWBRK(MDC,MSC,NPTST) 40.00
C
REAL ETOT, HM, PI, QB, SMERK
C
5. Parameter variables
6. Local variables

BB  Rate between the total energy and the energy
according to the maximum wave height HM
DIS0  Dummy variable
FRDEP  Frequency dependent function
ID  Counter for directional steps
IDDOM  Counter
IENT  Number of entries
IS  Counter for frequency steps
NORM  Normalization factor for frequency dependent function
RES  Dummy variable
RESD  Dummy variable
SURFA0
SURFA1
WS  Wavebreaking source term coefficient = DTOT/ETOT
WSD  Derivative of WS

INTEGER ID, IDDOM, IENT, IS

DOUBLE PRECISION BB, DIS0,
& SURFA0, SURFA1, WS, WSD

7. Common Blocks used

8. Subroutines used

---

9. Subroutines calling

SOURCE

10. Error messages

---

11. Remarks

---

12. Structure

Get HM, QB and ETOT from the subroutine SDISPA
For spectral direction IS and ID do,
  get the mean energy frequency average over the full spectrum
  if ETOT > 0 then
    compute source term for energy dissipation SURFA0 and SURFA1
  else
    source term for wave breaking is 0.
  end if

Compute source terms for energy averaged frequency
Store results in the arrays IMATDA and IMATRA

End of SSURF

13. Source text

SAVE IENT
DATA IENT/0/
IF (LTRACE) CALL STRACE (IENT, 'SSURF')

ALFA = PSURF(1) <default = 1.0>

BB = 8. * DBLE(ETOT) / ( DBLE(HM)**2 ) 30.82
SURFA0 = 0.
SURFA1 = 0.
IF (REAL(BB) .GT. 0. .AND.
  REAL(ABS(BB * (BB - DBLE(QB))) .GT. 0.) THEN
  IF ( BB .LT. 1. ) THEN
WS = ( DBLE(PSURF(1)) / DBLE(PI)) * DBLE(Q8) * DBLE(SMEBRK) / BB
WSD = WS * (1. - BB) / (BB * (BB - DBLE(Q8)))
ELSE
WS = ( DBLE(PSURF(1)) / DBLE(PI)) * DBLE(SMEBRK)
WSD = 0.
END IF
SURFA0 = WSD
SURFA1 = WS + WSD
ELSE
SURFA0 = 0.
SURFA1 = 0.
END IF

*** store the results for surf wave breaking ***
in the matrices IMATDA and IMATRA ***

DO 101 IS = 1, ISSTOP
DO 100 IDUM = IDCMIN(IS), IDCMAX(IS)
   ID = MOD ( IDUM - 1 + MDC , MDC ) + 1
   IMATDA(ID,IS) = IMATDA(ID,IS) + REAL(SURFA1)
   DISO = SURFA0 * DBLE(AC2(ID,IS,KGRID(1)))
   IMATRA(ID,IS) = IMATRA(ID,IS) + REAL(DISO)
   IF(TESTFL) FLMBRK(IPTST,ID,IS) = SURFA0 - SURFA1
   IF(TESTFL) FLMBRK(ID,IS,IPTST) = -1. * REAL(SURFA1)
   DISSCO(ID,IS) = DISSCO(ID,IS) - REAL(DISO)
   DISSCI(ID,IS) = DISSCI(ID,IS) + REAL(SURFA1)
100 CONTINUE
101 CONTINUE

*** test output ***

IF ( TESTFL .AND. ITEST .GE. 110 ) THEN
WRITE(PRINTF,6021) SURFA1,SURFA0
6021 FORMAT (' SSURF : SURFA1 SURFA0 ',2D12.4)
WRITE(PRINTF,6020) HNM, QB, ETOT, SMEBRK
6020 FORMAT (' HNM QB ETOT SMEBRK ',4E12.4)
END IF

end of the subroutine SSURF
RETURN
END
B.3 Subroutine SWCOMPFOR

C*****************************************************************************
C C For the following arguments a memory reservation is made in the
C datapool structure. Changes can be made in the *.ADC files and
C the main COMMON files.
C
SUBROUTINE SWCOMP (WAREA , RWAREA , 30.90
  % LWAREA , AC1 , 30.90
  % AC2 , COMPD A , 30.92
  % SPCDIR , SPCSIG , 30.72
  % SWTSDA , XXTST ,
  % IT , KGRPNT , 30.72
  % XGRID , YGRID , 300597
  % OBSTA , CROSS )
C*****************************************************************************
C INCLUDE 'timecomm.inc' 30.74
INCLUDE 'ocpcomm1.inc' 30.74
INCLUDE 'ocpcomm2.inc' 30.74
INCLUDE 'ocpcomm3.inc' 30.74
INCLUDE 'ocpcomm4.inc' 30.74
INCLUDE 'swcomm1.inc' 30.74
INCLUDE 'swcomm2.inc' 30.74
INCLUDE 'swcomm3.inc' 30.74
INCLUDE 'swcomm4.inc' 30.74

--|-----------------------------------------------|--
| Delft University of Technology |
| Faculty of Civil Engineering, Fluid Mechanics Group |
| P.O. Box 5048, 2600 GA Delft, the Netherlands |
| Authors: R.C. Ris, N. Booij, R. Padilla-Hernandez, |
| L.H. Holthuijsen |
--|-----------------------------------------------|--

BY USING THE SWAN SOFTWARE, YOU ARE CONSENTING TO BE BOUND BY
THIS AGREEMENT.

Delft University of Technology grants you a non-exclusive license
to use the SWAN Software free of charge.

DISCLAIMER OF WARRANTY.

The Swan Software is provided on an "AS IS" basis, without
warranty of any kind, including without limitation the warranties
of merchantability, fitness for a particular purpose and
non-infringement. The entire risk as to the quality and
performance of the Software is borne by you. You must determine
that the Software sufficiently meets your requirements.

LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, TORT, CONTRACT,
OR OTHERWISE, SHALL DELFT UNIVERSITY OF TECHNOLOGY BE LIABLE TO
YOU OR ANY OTHER PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL,
OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT
LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER
FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES
OR LOSSES. IN NO EVENT WILL DELFT UNIVERSITY OF TECHNOLOGY BE
LIABLE FOR ANY DAMAGES, EVEN IF DELFT UNIVERSITY OF TECHNOLOGY
SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES, OR
FOR ANY CLAIM BY ANY OTHER PARTY. YOU AGREE TO INDEMNIFY AND HOLD
DELT UNIVERSITY OF TECHNOLOGY HARMLESS WITH RESPECT TO ALL
CLAIMS BY THIRD PARTIES ARISING OUT OF YOUR USE OF THE RESULTS
OR OPERATION OF THE SOFTWARE.

TITLE,
Title, ownership rights, and intellectual property rights in the
Software shall remain with the authors of the Software. The
Software is protected by the copyright laws and treaties. Title
and related rights in the content accessed through the Software
is the property of the applicable content owner and may be
protected by applicable law. This License gives you no rights to
such content.

MISCELLANEOUS,
You may not remove any proprietary notices or labels on the
Software, nor remove this disclaimer. If any provision of this
Agreement is held to be unenforceable, such provision shall be
reformed only to the extent necessary to make it enforceable.

This Agreement shall be governed by the law of the Netherlands.

0. Authors
30.70: Nico Booyj
30.72: IJsebrand Haagsma
30.74: IJsebrand Haagsma (Include version)
30.75: IJsebrand Haagsma (Bug Fix)
30.81: Annette Kieftenburg
30.82: IJsebrand Haagsma
30.90: IJsebrand Haagsma (Equivalence version)
31.03: Annette Kieftenburg
32.02: Roeland Ris & Cor van der Schelde (ID-version)
34.01: Jeroen Adema

1. Updates
30.72, Nov. 97: Declaration of MSC4MI, MSC4MA, MDC4MI, MDC4MA and
ISTAT removed because they are common and already
declared in the INCLUDE file
30.72, Nov. 97: ITERMX can be chosen freely with NUM ACCUR also in dynamic
mode. Default ITERMX=6. Needs extensive testing
30.74, Nov. 97: Prepared for version with INCLUDE statements
32.02, Jan. 98: Introduced 1D-version
30.72, Feb. 98: Introduced generic names XGRID, YGRID and SPCEIG for SWAN
30.70, Feb. 98: call WINDP0 removed, function taken over by WINDP1
30.72, Mar. 98: Current switched off for the first iteration, when
preconditioning is required
30.72, Mar. 98: Writes the result of the iteration step to the PRINT
file
30.75, Mar. 98: Renamed SLOW to SIGLOW, because SLOW was used only locally
31.03, Feb. 98: Call SETUPP added, initialisation of array SETPDA
30.90, Oct. 98: Introduced EQUIVALENCE POOL-arrays
30.82, Oct. 98: Updated description of several variables
30.81, Jan. 99: Replaced variable STATUS by IERR (because STATUS is a
reserved word)
34.01, Feb. 99: Introducing STPNOW

2. Purpose
The aim of this model is to simulate the wave energy in
shallow water areas. In the subroutine SCOMP the main processes
taking place in the shallow water zone are determined in
several subroutines.
The input for this subroutine comes from SWANPRE1, SWANPRE2,
and SWANPRE3. The output is send to the subroutines SWANOUT1,
SWANOUT2 and SWANOUT3. The output consist of some characteristic
wave parameters and the wave action density. The equations are
called based on the action density N which is a function of the
spatial position (x,y) the relative frequency (s) and the
cospectral direction (d).
C  3. Method
C
C  Keywords:
C  Action density, propagation terms, refraction, reflection,
C  white capping, wave breaking, bottom friction, nonlinear
C  and nonhomogeneous wind- and current-fields, wave blocking
C  fully spectral description, non linear wave-wave interaction,
C  CGSTAB solver
C
C  4. Argument variables
C
C  RWAREA: Real EQUIVALENCE of WAREA            30.90
C  i  SPCDIR: (*,1); spectral directions (radians)  30.82
C  (*,2); cosine of spectral directions            30.82
C  (*,3); sine of spectral directions              30.82
C  (*,4); cosine*2 of spectral directions          30.82
C  (*,5); cosine*sine of spectral directions       30.82
C  (*,6); sine*2 of spectral directions            30.82
C  i  SPCSIG: Relative frequencies in computational domain in sigma-space 30.72
C  i  XGRID: Coordinates of computational grid in x-direction 30.72
C  i  YGRID: Coordinates of computational grid in y-direction 30.72
C
C  REAL  RWAREA(*)                             30.90
C  REAL  SPCDIR(MDC,6)                         30.82
C  REAL  SPCSIG(MSC)                           30.72
C  REAL  XGRID(MXC,MYC), YGRID(MXC,MYC)         30.72
C
C  RWAREA: Logical EQUIVALENCE of WAREA         30.90
C
C  LOGICAL LWAREA(*)                           30.90
C
C  INTEGERS:
C
C  IC  Dummy variable: ICode gridpoint:
C  IC = 1 Top or Bottom gridpoint
C  IC = 2 Left or Right gridpoint
C  IC = 3 Central gridpoint
C  Whether which value IC has, depends of the sweep
C  If necessary ic can be enlarged by increasing
C  the array size of ICMAX
C
C  ITER  Counter of iterations per 4 sweeps for accuracy
C  ITERMX Maximum number of iterations in model
C  IX  Counter of gridpoints in x-direction
C  IY  Counter of gridpoints in y-direction
C  IS  Counter of relative frequency band
C  IT  Counter in time space
C  ID  Counter of directional distribution
C  IBOT  Indicator for bottom friction
C  IBOT = 0 no bottom friction dissipation
C  IBOT = 1 Jonswap bottom dissipation model
C  IBOT = 2 Dinghamans bottom dissipation model
C  IBOT = 3 Madsen bottom dissipation model
C  ICUR  Indicator for current
C  ISURF  Indicator for wave breaking
C  ITRIAD  Indicator for nonlinear triad interactions
C  IQUDAD  Indicator for nonlinear quadruplet interactions
C  IWCAP  Indicator for wave capping
C  IWIND  Indicator for which wind generation model is used
C  IWIND = 1 first generation wind growth model
C  IWIND = 2 second generation wind growth model
C  IWIND = 3 third generation wind growth model
C  IREFR  indicator for refraction (can be turned off)
C  ITRFRE  indicator for transport of action in frequency
C  space
C  ICMAX  Maximum array size for the points of the molecule
C  KSX  input  Dummy variable to get the right sign in the
C  numerical difference scheme in X-direction
C  depending of the sweep direction, KSX = fl
C  KSY  input  Dummy variable to get the right sign in the
C  numerical difference scheme in Y-direction
C  depending of the sweep direction, KSY = fl
C  MXC  Maximum counter of gridpoints in x-direction
C  computational model: (XLEN/IX + 1)
C  MYC  Maximum counter of gridpoints in y-direction in
computational model: \( (YLEN/DY + 1) \)

MSC: Maximum counter of relative frequency in computational model.

MDC: Maximum counter of directional distribution in computational model \((2PI / DDIR + 1)\).

MTC: Maximum counter of the time, i.e.: (total time in prototype) / (time step).

MBOT: Maximum array size for PBOT.

MSURF: Maximum array size for PSURF.

MTRIAD: Maximum array size for PT RIAD.

MWCAP: Maximum array size for PWCAP.

MWIND: Maximum array size for PWIND.

REALS:

ALEN: Part of side length of an angle side.

BLEN: Angle between DX and DY.

BLR: Part of side length of an angle side.

DIR: Spectral direction (i.e., ID*DDIR).

DX: Input Length of spatial cell in X-direction.

DY: Input Length of spatial cell in Y-direction.

DS: Input Width of frequency band (is not constant because of the logarithmic distribution of the frequency.

DDIR: Input Width of directional band.

DT: Input Time step.

DDX: Input Same as DX but with correct sign depending of the direction of the sweep (+1. OR -1.) no input.

DDY: Input Same as DY but with correct sign depending of the direction of the sweep (+1. OR -1.) no input.

FAC_A: Factor representing the influence of the action-density deepening of the propagation velocity.

FAC_B: Factor representing the influence of the action-density deepening of the propagation velocity.

PI - alpha - beta.

HM: Maximum waveheight (breaking source term).

GRAV: Input Gravitational acceleration.

PI: 3.141592654

one and more dimensional arrays:

AC1: 4D Action density as function of D,S,X,Y at time T

AC2: 4D (Nonstationary case) action density as function of D,S,X,Y at time T+DT

CGO: 2D Group velocity as function of IC and IS in the direction of wave propagation in absence of currents.

CG: 3D Group velocity as function of IC and IS and D in the direction of wave propagation in presence of currents.

CAX: 3D Wave transport velocity in x-direction, function of (ID,IS,IC).

CAY: 3D Wave transport velocity in y-direction, function of (ID,IS,IC).

CAS: 3D Wave transport velocity in S-direction, function of (ID,IS,IC).

CAD: 3D Wave transport velocity in D-direction, function of (ID,IS,IC).

COMPD: 3D array containing depth and other arrays of (IX,IY) 20.39

JD: 3D Depth as function of X and Y at time T

JDP: (Nonstationary case) depth as function of X and Y at time T+DT

JVX: 1D Array containing current velocity of X and Y at time T

JVX2: (Nonstationary case) X-component of current velocity in (X,Y) at time T+DT

JYV: 1D Y-component of current velocity in (X,Y) at time T

JYV2: (Nonstationary case) Y-component of current velocity in (X,Y) at time T+DT

JXW: 1D Array containing wind velocity in (X,Y) at time T

JXW2: X-component of wind velocity in (X,Y) at time T+DT (nonstationary case)

JYW: 1D Array containing wind velocity in (X,Y) at time T

JYW2: Y-component of wind velocity in (X,Y) at time T+DT (nonstationary case)

JUBOT: Absolute orbital velocity in a gridpoint (IX,IY).
C SWTSDA 4D intermediate data computed for the test points;
C there are MTSVAR subarrays:
C JFWNDA wind source term part A
C JFWNDB wind source term part B
C JFWCAP whitecapping source term
C JPBFR bottom friction
C JFWBB surf breaking
C JP4S quadruplet interactions
C JP4D quadruplet interactions
C JPTRI triad interactions
C ALIMW 1D Maximum energy by wind growth. This dummy array is
C used because the maximum value has to be checked
C direct after the solver of the tridiagonal matrix
C see the subroutine SOLMAT
C GROWW 1D Check for a certain frequency if the waves are
C growing or not in a spectral direction (LOGICAL)
C HSACC1 2D Represent the significant wave height at time T
C HSACC2 2D Represent the significant wave height at time T+1
C IMATDA 2D Coefficients of diagonal of matrix
C IMATLA 2D Coefficients of lower diagonal of matrix
C IMATUA 2D Coefficients of upper diagonal of matrix
C IMATLL 2D Coefficients of lower diagonal of matrix
C IMATUL 2D Coefficients of upper diagonal of matrix
C IMATRA 2D Coefficients of right hand side of matrix
C KAVE 2D wavenumber as a function of the relative frequency S
C and position IC(iix,iyy)
C PBOT 1D Coefficient for the bottom friction models
C PSURF 1D Coefficient for the wave breaking model
C PTIAD 1D Coefficient for the triad interaction model
C PWCAP 1D Coefficient for the white capping model
C PWIN 1D Coefficient for the wind growth model
C SACC1 2D Represents the mean wave period at time T
C SACC2 2D Represents the mean wave period at time T+1
C PWTAIL 1D Coefficients for tail of spectrum
C ICDMIN 1D frequency dependent counter in directional space
C no current <--- current
C ICDMAX 1D frequency dependent counter in directional space
C no current <--- current
C ISCMIN 1D frequency dependent counter in frequency space
C no current <--- current
C ISCMAX 1D frequency dependent counter in frequency space
C no current <--- current
C SECTOR 1D Indicates which configuration is present (see
C subroutine SWSEL )
C ANYBIN 2D Set a particular bin TRUE or FALSE depending on
C SECTOR
C WWINT 1D Counters for 4 wave-wave interactions
C WWWG 1D Weight coefficients for the 4 wave-wave interactions
C WWWSG 1D Weight coefficients for the semi-implicit computation
C COLUZ 1D In presence of a current the spectral direction can
C be circular and closed. Matrixcoefficients appear in
C the top right and bottom left corner of the matrix
C After pivoting --> coefficients are stored in COLUZ
C space
C DIFLOW 2D Lower diagonal in solver for diffusion
C DIFDIZ 2D Diagonal in solver for diffusion
C DIGUP 2D Upper diagonal in solver for diffusion
C DIFHVC 2D Right hand vector
C
C Coefficients for the arrays:
C
C default
C value:
C
C PBOT(1) = CFC 0.005 (Collins equation)
C PBOT(2) = CFW 0.01 (Collins equation)
C PBOT(3) = GAMJNS 0.0038 (Jonswap equation)
C PBOT(4) = MF -0.08 (Madsen equation)
C PBOT(5) = KN 0.05 (bottom roughness)
C ISURF
C 1 (Constant breaking coefficient)
C 2 (variable breaking coefficient
C according to Nelson (1994))
C PSURF(1) = ALFA 1.0  (Battjes Jansen)
C PSURF(2) = GAMMA 0.8  (Breaking criterium)
C
C PWCAP(1) = ALFWC 2.36e-5  (Empirical coefficient)
C PWCAP(2) = ALFAPM 3.02e-3  (Alpha of Pierson Moskowitz frequency)
C
C PWIND(1) = CF10 188.0  (second generation wind growth model)
C PWIND(2) = CF20 0.59  (second generation wind growth model)
C PWIND(3) = CF30 0.12  (second generation wind growth model)
C PWIND(4) = CF40 250.0  (second generation wind growth model)
C PWIND(5) = CF50 0.0023  (second generation wind growth model)
C PWIND(6) = CF60 -0.2233  (second generation wind growth model)
C PWIND(7) = CF70 0.  (second generation wind growth model)
C PWIND(8) = CF80 -0.56  (second generation wind growth model)
C PWIND(9) = RHOMW 0.00125  (density air / density water)
C PWIND(10) = EDMLFM 0.0036  (limit energy Pierson Moskowitz)
C PWIND(11) = CDRAF 0.0012  (drag coefficient)
C PWIND(12) = UMIN 1.0  (minimum wind velocity)
C PWIND(13) = FMLM 0.13  ( )
C
C PNMS(1) = DREL relative error in Hs and Tm
C PNMS(2) = DHABS absolute error in Hs
C PNMS(3) = DTABS absolute error in Tm
C PNMS(4) = NPNTS number of points were accuracy is reached
C
C PNMS(4) = NOT USED
C PNMS(5) = NOT USED
C PNMS(6) = CDD numerical diffusion in theta space
C PNMS(7) = CSS numerical diffusion in sigma space
C PNMS(8) = NUMFRE numerical scheme in frequency space :
C 1) implicit scheme
C 2) explicit scheme CFL limited
C 3) explicit scheme filter after iteration
C PNMS(9) = DIFFC if explicit scheme is used, then numerical
C diffusion coefficient can be chosen
C PNMS(10) = PREC type of preconditioner
C PNMS(11) = EPS1 epsilon 1 in equation to terminate iteration
C PNMS(12) = EPS2 epsilon 2 in equation to terminate iteration
C PNMS(13) = OUTF request for output for solver
C PNMS(14) = NITER maximum number of iterations for solver
C PNMS(15) = NOT USED
C PNMS(16) = NOT USED
C PNMS(17) = NOT USED
C PNMS(18) = NOT USED
C PNMS(19) = CFL CFL criterion for option explicit scheme
C in frequency space (see PNMS(8)
C PNMS(20) = GRWMX maximum growth inspersal bin
C
C arrays for the 4-wave interactions:
C
C WWINT ( 1 = IPD WWAWG ( = AGW1 WWSWG ( = SWG1
C 2 = IPD1 = AGW2 = SWG2
C 3 = IDM = AGW3 = SWG3
C 4 = IDM1 = AGW4 = SWG4
C 5 = ISP = AGW5 = SWG5
C 6 = ISP1 = AGW6 = SWG6
C 7 = ISM = AGW7 = SWG7
C 8 = ISM1 = AGW8 ) = SWG8 )
C 9 = ISLOW
C 10 = ISGHG
C 11 = ISICLW
C 12 = ISICHG
C 14 = IDLOW
C 15 = IDHG
C 16 = MSC4MI
C 17 = MSC4MA
C 18 = MDC4MI
C 19 = MDC4MA
C 20 = MSCMAX
C 21 = MDCMAX )
C
C
C 6. Local variables
SIGLOW: recommend lowest frequency when TRIADS are activated

REAL SIGLOW

8. Subroutines used

INSAC
SWOMPU
SACCUR
PFLSRC

LOGICAL STPNOW

34.01

9. Subroutines calling

SWANPREn, SWANGOUTn

10. Error messages

---

11. Remarks

SCOMPU is the main subroutine is and called of the main program SWAN.
The main program SWAN is build of three main subroutines:

1. SPRE (preparation of the computation (reading parameters))
2. SCOMPU (computation of the action densities (discussed below))
3. SOUT (output of the program)

In this part the subroutine SCOMPU is discussed:

SCOMPU

Sweeps: loop over spatial points

_______

|-------+ INSAC determine initial values for
|       | accuracy check
|-------+ SINSTR read incoming waves (this
|       | subroutine should be implemented
|       | in the SWANMAIN instead of in
|       | SWANCOM1

|-------+ SWOMPU
determ. of waveparameters

|-------+ SWAPAR comp. of propagation
|       | velocities of energy:
|       | CAX, CAY

|-------+ SPROXY comp. of propagation
|       | velocities of energy:
|       | CAS, CAD

|-------+ SPROSD compute absolute wind, FPM
|       | mean wind direction, min. and
|       | max. counters for the wind,
|       | wind friction velocity

|-------+ WINDF1 Compute contributions to
|       | spectrum due to high frequency
tail

|-------+ CNTAIL predict energy density in
|       | gridpoints for first
|       | iteration

|-------+ SPREDT check stability of comp.

|-------+ STABIL comp. Ub, Etot, Hmax, Qb,
|       | SME, SMA, SMESPC, SMASPC

|-------+ FRABRE2 comp. of fraction of
|       | 30.77
C | | | | breaking waves
C | | | | (This subroutine replaces
C | | | | FRABRE)
C | | |------- SOURCE | comp. of source terms
C | | | | +---------- SBOT | bottom friction
C | | | | +---------- SWCAP | white capping
C | | | | +---------- SSURF | wave breaking
C | | | | +---------- STRIAD | nonlinear wave interactions
C | | | | +---------- SWIND1 | first generation wind model
C | | | | +-- WINDP2 | compute total wind sea energy o
C | | | | | | SWIND2 | second generation wind model
C | | | | + SWIND3 | third generation wind model
C | | |------- ACTION | comp. of ACTION balance eq.
C | | | | ( $ CAN/$n )
C | | | | +------- STIME | comp. of ($AC2/$t)
C | | | | +------- STRSX | $[CAX AC2]/$X
C | | | | +------- STRSY | $[CAY AC2]/$Y
C | | | | +------- STRSS | $[CAS AC2]/$S
C | | | | +------- STRSD | $[CAD AC2]/$D
C | | |------- SOLMAT | solve the matrix which is
C | | | | filled in SOURCE and ACTION
C | | |------- FILIMP | filter the frequency spectrum
C | | | | in presence of a current using
C | | | | a diffusion model (important for
C | | | | wave blocking) -->IMPLICIT SCHEME
C | | |------- DIFSEL | The matrix filled in FILIMP is
C | | | | solved for each direction separately
C | | |------- WINDP3 | Limit the energy spectrum
C | | | | for first and second
C | | | | generation wind model
C | | | | | AAAA+ PLTSRC | write source term after an
C | | | | | iteration to a file SOURCE
C | | | | | AAAA+ SACCUR | check accuracy of the comp.
C | | | | END SCOMPU

C 12. Structure
C
C The general numerical procedure in SWAN is based on the next
C principle of sweeps (see Holthuijsen et al 1993) where
C
C
C {***********************************************************************}
C | definition of the sweep directions
C | |}
C | | { swp_NW = 4 \ | N / | / \ swp_EN = 3
C | | \ | | |}
C | | W ------------------------ E (0 degrees, id=1)
C | | |}
C | | swp_WS = 1 / | | / \ swp_SE = 2
C | | | |
Call INSAC to give values to HSASCC en SASCC to check the accuracy

For IT = 1 to end of computation time (MTC), do,

Call SINSTR to read incoming waves at the boundaries (N,E,S,W) in stationary and nonstationary case SINSTR changes

If accuracy <= given accuracy, then do iteration,

give argument for sweep : swpdire = 1
KSX = -1
KSY = -1
give number of direction a start and an end value:
For IY=2 to MYC and IX=2 to MXC, do,
Call SWOMPU to compute the wave field

give argument for sweep : swpdire = 2
KSX = +1
KSY = +1
give number of direction a start and an end value:
For IX=MXC-1 to l and IY=2 to MYC, do,
Call SWOMPU to compute the wave field

give argument for sweep : swpdire = 3
KSX = +1
KSY = +1
give number of direction a start and an end value:
For IY=MYC-1 to 1 and IX=MXC-1 to 1, do,
Call SWOMPU to compute the wave field

give argument for sweep : swpdire = 4
KSX = +1
KSY = +1
give number of direction a start and an end value:
For IX=2 to MXC and IY=MYC-1 to 1, do,
Call SWOMPU to compute the wave field

CALL PLTSCR to write the source term to a file

CALL SACCUR to check the accuracy of the computation

End of SCOMPU

13. Source text

* MAIN SUBROUTINE OF COMPUTATIONAL PART

* -- SCOMPU --
C * Definition of variables in main program *
C *****************************************************
C
INTEGER ITER, IX, IY, IS, IT, 30.72
& KSX, KSY, MSCMAX, MDCMAX, SWPDIR,
& IERR, 30.81
& INOCONV 30.72
C
REAL DDX, DDY, ACCUR, XIS, SNLC1, DAL1, 30.74
& DAL2, DAL3
C
CHARACTER PTYPE *1
C
LOGICAL PRECOR
C
INTEGER WAREA(*)
& XITST(2*NPTST), KGRPMT(MXC,MYC) 30.21
& OBSTA(*), CROSS(2,MCGRD) 300597
C
REAL AC2(MDC, MSC, MCGRD)
& AC1(MDC, MSC, MCGRD), 30.21
& COMEPA(MCGRD, MCIMVAR), 30.21
& SWTSDA(MDC, MSC, MFTSTA, MTSMVAR) 40.00
C
SAVE IENT DATA IENT/0/
IF (LTRACE) CALL STRACE (IENT, 'SWCOMP')
C
IF (IT .EQ. 1 .AND. ITEST.GE.1) THEN
  WRITE(PRINTF,333) 'SWAN'
  333 FORMAT(/,
  HERE,---------------------------------------
  HERE, COMPUTATIONAL PART OF ', A
  HERE,---------------------------------------
  HERE/)
C
IF (ONED) THEN
  WRITE(PRINTF,*) 'One-dimensional mode of SWAN is activated'
  ENDIF
C
WRITE(PRINTF,7001) MXC,MYC
7001 FORMAT(' Grid resolution : MXC ',I12, ' , MYC ',I12)
WRITE(PRINTF,7101) MDC, MDC
7101 FORMAT(' ', MXC, ', I12 , ', MDC, ', , I12)
WRITE(PRINTF,7201) MTC, ICIMAX
7201 FORMAT(' ', MTC, ', I12 , ', ICIMAX, ', I12)
WRITE(PRINTF,7301) NSTATC, ITERMX
7301 FORMAT(' ', NSTATC, ', I12 , ', ITERMX, ', I12)
WRITE(PRINTF,7013) ITFRE, IREFR
7013 FORMAT(' Propagation flags : ITFRE ',I12, ', , IREFR , I12)
WRITE(PRINTF,7014) IBOT, ISURF
7014 FORMAT(' Source term flags : IBOT ',I12, ', ', ISURF, ', I12)
WRITE(PRINTF,7114) IWCAP, IWIND
7114 FORMAT(' ', IWCAP, ', I12 , ', IWIND, ', I12)
WRITE(PRINTF,7015) ITRIAD, IQUAD
7015 FORMAT(' ', ITRIAD, ', I12 , ', IQUAD, ', I12)
IF (ICUR.GT.0) THEN
  WRITE (PRINTF,7115) 'ON'
ELSE
  WRITE (PRINTF,7115) 'OFF'
ENDIF
7115 FORMAT(' Current is ', A3)
WRITE(PRINTF,7004) DX, DY
7004 FORMAT(' Spatial step : DX ',E12.4,' , DY ',E12.4)
WRITE(PRINTF,7104) DDIR*180./PI
7104 FORMAT(' ', DDIR ', E12.4)
WRITE(PRINTF,7003) GRAV, PI

7003 FORMAT(' Physical constants : GRAV ',E12.4,' PI ',E12.4)

WRITE(PRINTF,7027) UI0, WDIC*180./PI

7027 FORMAT(' Wind input : WSPEED ',E12.4,' DIR ',E12.4)

WRITE(PRINTF,7123) PWTAIL(1),PWTAIL(2)

7123 FORMAT(' Tail parameters : E(T) ',E12.4,' E(k) ',E12.4)

WRITE(PRINTF,7133) PWTAIL(3),PWTAIL(4)

7133 FORMAT(' : A(T) ',E12.4,' A(k) ',E12.4)

WRITE(PRINTF,8013) PNUNS(1), PNUNS(2)

8013 FORMAT(' Accuracy command : DREL ',E12.4,' DHABS ',E12.4)

WRITE(PRINTF,8213) PNUNS(3), PNUNS(4)

8213 FORMAT(' : DTABS ',E12.4,' NPTS ',E12.4)

WRITE(PRINTF,3613) PNUNS(20)

3613 FORMAT(' : GRWMX ',E12.4)

WRITE(PRINTF,8513) PNUNS(6)

8513 FORMAT(' : CDD ',E12.4)

WRITE(PRINTF,8613) INT(PNUNS(8))

8613 FORMAT(' Scheme freq. space : NUMFRE ',I12)

WRITE(PRINTF,4113) PNUNS(7)

4113 FORMAT(' 1) Implicit (CGSTAB) : CSS ',E12.4)

WRITE(PRINTF,8113) INT(PNUNS(10)), PNUNS(11)

8113 FORMAT(' : PREC ',I12,' EPS1 ',E12.4)

WRITE(PRINTF,2213) PNUNS(12), INT(PNUNS(13))

2213 FORMAT(' : EPS2 ',E12.4,' OUTPUT',I12)

WRITE(PRINTF,8223) INT(PNUNS(14))

8223 FORMAT(' : NITER ',I12)

WRITE(PRINTF,4213) PNUNS(19)

4213 FORMAT(' 2) Explicit with CFL : CFL ',E12.4)

WRITE(PRINTF,8413) IQUAD, PNUNS(20)

8413 FORMAT(' Quadruplets : IQUAD ',I12,' GRWMX ',E12.4)

WRITE(PRINTF,9413) ITRIAD, PTRIAD(1)

9413 FORMAT(' Triads : ITRIAD ',I12,' PAR ',E12.4)

IF (IBOT.EQ.2) THEN

WRITE(PRINTF,7005) PBOT(2),PBOT(1)

7005 FORMAT(' Collins (72) : CFW ',E12.4,' CFC ',E12.4)

ELSE IF (IBOT.EQ.3) THEN

WRITE(PRINTF,7335) PBOT(4),PBOT(5)

7335 FORMAT(' Madsen et al. (84) : MF ',E12.4,' KN ',E12.4)

ELSE IF (IBOT.EQ.1) THEN

WRITE(PRINTF,7325) PBOT(3)

7325 FORMAT(' JONSWAP (73) : GAMMA ',E12.4)

ELSE

WRITE (PRINTF, '*') ' Bottom friction is off'

ENDIF

IF (IWCAP.EQ.1) THEN

WRITE(PRINTF,6005) PWCAP(1),PWCAP(2)

6005 FORMAT(' W-cap Komen (*84) : EMPCOF ',E12.4,

APM ',E12.4)

ELSE IF (IWCAP.EQ.2) THEN

WRITE(PRINTF,6335) PWCAP(3),PWCAP(4)

6335 FORMAT(' W-cap Janssen (*90) : CFJANG ',E12.4,

DELTA ',E12.4)

ELSE IF (IWCAP.EQ.3) THEN

WRITE(PRINTF,6135) PWCAP(5)

6135 FORMAT(' W-cap Longuet-Higgins: CFLHIG ',E12.4)

ELSE IF (IWCAP.EQ.4) THEN

WRITE(PRINTF,6136) PWCAP(6),PWCAP(7)

6136 FORMAT(' W-cap Battjes/Janssen: BJSTF ',E12.4,

BJALF ',E12.4)

ELSE IF (IWCAP.EQ.5) THEN

WRITE(PRINTF,6137) PWCAP(8)

6137 FORMAT(' : KCONV ',E12.4)

ELSE

WRITE (PRINTF, '*') ' Whitecapping is off'

ENDIF

IF (ISURF.EQ.1) THEN

WRITE(PRINTF,7012) PSURF(1),PSURF(2)

7012 FORMAT(' Battjes&Janssen (*78) : ALPHA ',E12.4,

GAMMA ',E12.4)

ELSE IF (ISURF.EQ.2) THEN

970219
WRITE(PRINTF,7212) PSURF(1), PSURF(4), PSURF(5)
7212 FORMAT(' Nelson ('94): ALPHA ',E12.4,
&   ' GAMmin ',E12.4, ' GAMmax ',E12.4)
ELSE
WRITE (PRINTF, *) ' Surf breaking is off'
ENDIF
*
WRITE(PRINTF,7126) PWIND(14), PWIND(15)
7126 FORMAT(' Janssen ('89, 90) : ALPHA ',E12.4,
&   ' KAPPA ',E12.4)
WRITE(PRINTF,7136) PWIND(16), PWIND(17)
7136 FORMAT(' Janssen ('89, 90) : RHOA ',E12.4,
&   ' RHOW ',E12.4)
WRITE(PRINTF,*)
WRITE(PRINTF,1012) PWIND(1), PWIND(2)
1012 FORMAT(' 1st and 2nd gen. wind: CF10 ',E12.4,
&   ' CF20 ',E12.4)
WRITE(PRINTF,1013) PWIND(3), PWIND(4)
1013 FORMAT(' CF30 ',E12.4,
&   ' CF40 ',E12.4)
WRITE(PRINTF,1014) PWIND(5), PWIND(6)
1014 FORMAT(' CF50 ',E12.4,
&   ' CF60 ',E12.4)
WRITE(PRINTF,1015) PWIND(7), PWIND(8)
1015 FORMAT(' CF70 ',E12.4,
&   ' CF80 ',E12.4)
WRITE(PRINTF,1016) PWIND(9), PWIND(10)
1016 FORMAT(' RHOAW ',E12.4,
&   ' EDMLPM',E12.4)
WRITE(PRINTF,1017) PWIND(11), PWIND(12)
1017 FORMAT(' CODHAG ',E12.4,
&   ' UNIM ',E12.4)
WRITE(PRINTF,1018) PWIND(13)
1018 FORMAT(' LIM_PM ',E12.4)
C
* IF ( ITST .GT. 2
&   ) THEN
DO IS = 1, MSC
WRITE(PRINTF,*), IS and SPCSIG(IS) : ',IS,SPCSIG(IS)
30.72
ENDDO
ENDIF
END IF
C
*** check resonance condition for triads ***
C
IF ( ITRIAD .GE. 1 .AND. ITST .GE. 1 ) THEN
IJ2 = INT (FLOAT(MSC) / 2.)
IJ1 = IJ2 - 1
FAC1 = SPCSIG(IJ2) / SPCSIG(IJ1)
IRES = INT ( LOG10( 2. ) / LOG10( FAC1 ) )
FACRES = 10.**( LOG10(2.) / FLOAT(IRES) )
SIGLOW = SPCSIG(MSC) / ( FACRES**(FLOAT(MSC-1)) )
30.75
WRITE(PRINTF,*)
WRITE(PRINTF,*) '---------------------------------------------------'
46 WRITE(PRINTF,*), 'Triad wave-wave interactions are activated: '
47 WRITE(PRINTF,*)
48 WRITE(PRINTF,51) SPCSIG(MSC-IRES), SPCSIG(MSC)
51 FORMAT(' Higher harmonic of f1= ',F10.4,' Hz is equal f2= ',
&   ' F10.4,' ' Hz')
52 WRITE(PRINTF,52) SPCSIG(MSC)/SPCSIG(MSC-IRES), IRES
52 FORMAT(' resulting in a f2/f1 of','F10.4,' [resonance condition =
&   ',12,']')
53 WRITE(PRINTF,53) SIGLOW / (2.*PI)
53 FORMAT(' For good scaling behaviour triads set [FLOW] equal: ' 
&   ',F0.2, ' s')
54 WRITE(PRINTF,*)
54 '---------------------------------------------------'
C
*** print test points ***
C

ws | delta hydraulics

B – 22
IF (NPTST.GT.0) THEN
  DO 121 II = 1, NPTST
    WRITE(PRINTF,1001) II, XYTST(2*II-1), XYTST(2*II-1)
  1001 FORMAT(’ Test points ’,I3)
  121 CONTINUE
ENDIF

*** calculate auxiliary variables MSCMAX and MDCMAX ***
*** for the 4-WAVE interactions to allocate required ***
*** memory in the WAREA (The size of the arrays for ***
*** the quadruplets are on the foremost not known ***

IF ( IQ [A_0] = .GE. 1 )
& CALL BND4WW (MSCMAX,MDCMAX,SPCSIG ) 34.00

*** skip data pool procedure for second time step ***

IF (IT .GT. 1) GO TO 498 30.00

*** organize computational data in array WAREA ***

IERR = -1 30.81
LENARR = -1 20.62
CALL DPBLDP (WAREA, LENARR, 2, 0, IERR) 30.81 20.62
CALL DPBLDP (WAREA, -1, 2, 0, STATUS) deleted 20.62

*** put all counters for arrays in data pool ***

CALL DPADD (WAREA, ’HSAC1’, JHSAC1, ’S’, IHSAC1, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPADD (WAREA, ’HSAC2’, JHSAC2, ’S’, IHSAC2, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPADD (WAREA, ’SACC1’, JSACC1, ’S’, ISACC1, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPADD (WAREA, ’SACC2’, JSACC2, ’S’, ISACC2, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPADD (WAREA, ’CG’, JCG, ’S’, IC G, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPADD (WAREA, ’CA’, JCA , ’S’, IC A, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPADD (WAREA, ’CAY’, JCAY, ’S’, ICAY, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPADD (WAREA, ’CAD’, JCAD, ’S’, ICAD, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPADD (WAREA, ’CGO’, JCGO, ’S’, ICGO, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPADD (WAREA, ’KNUM’, JNUM, ’S’, IKNUM, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPADD (WAREA, ’SWMAT’, JSWMAT, ’S’, ISWMAT, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPADD (WAREA, ’ALIMW’, JALIMW, ’S’, IALIMW, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPADD (WAREA, ’GROWW’, JGROWW, ’S’, IGROWW, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPADD (WAREA, ’IDCIMIN’, JIDCIMIN, ’S’, IIDCIMIN, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPADD (WAREA, ’ISCMIN’, JISCIMIN, ’S’, IISCIMIN, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPADD (WAREA, ’SECTOR’, JS SECTOR, ’S’, IS SECTOR, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPADD (WAREA, ’WUSWM’, JWUSWM, ’S’, IWUSWM, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPADD (WAREA, ’WUSWM’, JWUSWM, ’S’, IWUSWM, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPADD (WAREA, ’AF11’, JAF11, ’S’, IAF11, IERR) 30.81

*** quadruplets pointers for iquad = 1, 2, 3 ***
CALL DPADD (WAREA, ’WINT’, JIWINT, ’S’, IIWINT, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPADD (WAREA, ’WWAWG’, JWWAWG, ’S’, IWWAWG, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPADD (WAREA, ’WWSWG’, JWWSWG, ’S’, IWWSWG, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPADD (WAREA, ’AF11’, JAF11, ’S’, IAF11, IERR) 30.81
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'UE', JUE, 'S', IUE, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'SA1', JSA1, 'S', ISA1, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'SA2', JSA2, 'S', ISA2, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'SN1', JSN1, 'S', ISN1, IERR) RETURN
IF (STPNOW()) RETURN
C
*** igquad = 1 ***
CALL DPADDP (WAREA, 'DA1C', JD1AC, 'S', ID1AC, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'DA1P', JD1AP, 'S', ID1AP, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'DA1M', JD1AM, 'S', ID1AM, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'DA2C', JD2AC, 'S', ID2AC, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'DA2P', JD2AP, 'S', ID2AP, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'DA2M', JD2AM, 'S', ID2AM, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'DSN1', JDSN1, 'S', IDSN1, IERR) RETURN
IF (STPNOW()) RETURN
C
*** igquad = 3 ***
CALL DPADDP (WAREA, 'MEMNL', JMEMNL, 'S', IMEMNL, IERR) RETURN
IF (STPNOW()) RETURN
C
*** explicit scheme in frequency space and diffusion model ***
CALL DPADDP (WAREA, 'COLO2', JCOLO2, 'S', ICOLO2, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'DIFLOW', JDIFLOW, 'S', IDIFLOW, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'DIFDIF', JDIFDIF, 'S', IDIFDIF, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'DIFUPP', JDIFUPP, 'S', IDIFUPP, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'DIFRHV', JDIFRHV, 'S', IDIFRHV, IERR) RETURN
IF (STPNOW()) RETURN
C
*** ILU CGSTAB solver ***
CALL DPADDP (WAREA, 'BAND', JBAND, 'S', IBAND, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'EXACT', JEXACT, 'S', IEXACT, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'RHV', JRVH, 'S', IRVH, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'RINSOL', JRINSOL, 'S', IRINSOL, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'SOLUT', JSOLUT, 'S', ISOLUT, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'WORK', JWORK, 'S', IWORK, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'PRECON', JPRECON, 'S', IPRECON, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'UPPERI', JUPPERI, 'S', IUPPERI, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'LOPERI', JLOPERI, 'S', ILOPERI, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'INFMAT', JINFMAT, 'S', IINFMAT, IERR) RETURN
IF (STPNOW()) RETURN
CALL DPADDP (WAREA, 'INSOL', JINSOL, 'S', IINSOL, IERR) RETURN
IF (STPNOW()) RETURN
C
*** Jansen wind input model ***
cdel
CALL DPADDP (WAREA, 'ANYWIND', JANYWIND, 'S', IANYWIND, STATUS) RETURN
CALL DPADDP (WAREA, 'ANYWIND', JANYWIND, 'S', IANYWIND, IERR) RETURN
IF (STPNOW()) RETURN
C
*** for obstacles ***
CALL DPADDP (WAREA, 'OBDF', JOBDF, 'S', IOBDF, IERR)
C
040697
IF (STPNOW()) RETURN
C
*** Auxiliar array ***
C
*** In case of SETUP generate array for setup data ***
C
IF (LSETUP.GT.0) THEN
  CALL DPDPP (WAREA, 'SETPDA', JSTPDA, 'S', ISTPDA, IERR)
ENDIF

C******************************************************************************

C

C *** fill pool array (DATA POOL EXPAND RECORD) ***

C******************************************************************************

C

JMATD = 1
JMATR = 2
JMATL = 3
JMATU = 4
JMAT5 = 5
JMAT6 = 6
JABIN = 7
JABLR = 8
JDISO = 9
JDISO = 10
JLXK1 = 11
JAOAD = 12
MSWMAT = 12

CALL DEEXPR (WAREA, JHSAC1, MCGRD , IHSAC1 , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DEEXPR (WAREA, JHSAC2, MCGRD , IHSAC2 , IERR) 30.81 30.21
IF (STPNOW()) RETURN 34.01
CALL DEEXPR (WAREA, JSAC1C, MCGRD , ISAC1 , IERR) 30.81 30.21
IF (STPNOW()) RETURN 34.01
CALL DEEXPR (WAREA, JSAC2C, MCGRD , ISAC2 , IERR) 30.81 30.21
IF (STPNOW()) RETURN 34.01
CALL DEEXPR (WAREA, JCG , MDC*MSC*ICMAX, JCG , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DEEXPR (WAREA, JCAX , MDC*MSC*ICMAX, ICAX , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DEEXPR (WAREA, JCAY , MDC*MSC*ICMAX, ICAY , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DEEXPR (WAREA, JCAS , MDC*MSC*ICMAX, ICAS , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DEEXPR (WAREA, JCAD , MDC*MSC*ICMAX, ICAD , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DEEXPR (WAREA, JCGO , MSC*ICMAX , JCGO , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DEEXPR (WAREA, JKNUM , MSC*ICMAX , JKNUM , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DEEXPR (WAREA, JSWMAT, MDC*MSC*MSWMAT, ISWMAT, IERR) 34.01
IF (STPNOW()) RETURN 34.01
CALL DEEXPR (WAREA, JALIMW, MDC*MSC , JALIMW , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DEEXPR (WAREA, JGROWW, MDC*MSC , JGROWW , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DEEXPR (WAREA, JIDCMIN, MDC , IIDCMIN, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DEEXPR (WAREA, JIDCMAX, MDC , IIDCMAX , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DEEXPR (WAREA, JSECTOR, MDC , JSECTOR, IERR) 34.01
IF (STPNOW()) RETURN 30.81
CALL DEEXPR (WAREA, JISCMIN, MDC , JISCMIN , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DEEXPR (WAREA, JISCMAX, MDC , JISCMAX , IERR) 30.81
IF (STPNOW()) RETURN 34.01

C******************************************************************************

C

*** quadruplets ***

CALL DEEXPR (WAREA, JWINT, 20 , JWINT , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DEEXPR (WAREA, JWAWG, 8 , JWAWG , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DEEXPR (WAREA, JWWSW, 8 , JWWSW , IERR) 30.81
IF (STPNOW()) RETURN 34.01
IF ( IQUAD .GE. 1 ) THEN
  CALL DEEXPR (WAREA, JAF11, MSCMAX , JAF11 , IERR) 30.81
  IF (STPNOW()) RETURN 34.01
  CALL DEEXPR (WAREA, JUE , MSCMAX*MDMAX , JUE , IERR) 30.81
  IF (STPNOW()) RETURN 34.01

we, dalt hydraulics
CALL DPEXPR (WAREA, JSA1, MSCMAX*MDCMAX, ISA1, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JSA2, MSCMAX*MDCMAX, ISA2, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JSFNL, MSCMAX*MDCMAX, ISFNL, IERR) 30.81
IF (STPNOW()) RETURN 34.01
IF (IQAUD .EQ. 1) THEN
*** semi-implicit calculation ***
CALL DPEXPR (WAREA, JDA1C, MSCMAX*MDCMAX, IDA1C, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JDA1P, MSCMAX*MDCMAX, IDA1P, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JDA1M, MSCMAX*MDCMAX, IDA1M, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JDA2C, MSCMAX*MDCMAX, IDA2C, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JDA2P, MSCMAX*MDCMAX, IDA2P, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JDA2M, MSCMAX*MDCMAX, IDA2M, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JDSNL, MSCMAX*MDCMAX, IDSNL, IERR) 30.81
IF (STPNOW()) RETURN 34.01
ELSE IF (IQAUD .EQ. 3) THEN
*** prior to every iteration full directional domain ***
CALL DPEXPR (WAREA, JMEMNL, MDC*MSC*MCGRD, JMEMNL, IERR) 30.81 30.21
IF (STPNOW()) RETURN 34.01
END IF
ELSE
*** no quadruplets ***
CALL DPEXPR (WAREA, JAF11, 0, IAF11, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JUE , 0, IUE , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JSA1 , 0, ISA1 , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JSA2 , 0, ISA2 , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JSFNL , 0, ISFNL , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JDA1C , 0, IDA1C , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JDA1P , 0, IDA1P , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JDA1M , 0, IDA1M , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JDA2C , 0, IDA2C , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JDA2P , 0, IDA2P , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JDA2M , 0, IDA2M , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JDSNL , 0, IDSNL , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JMEMNL , 0, JMEMNL , IERR) 30.81
IF (STPNOW()) RETURN 34.01
END IF
*** explicit sche in frequency space and diffusion model ***
CALL DPEXPR (WAREA, JCOLU2 , MDC , ICOLOU2 , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JDIFLOW, MDC*MSC, IDIFLOW, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JDIFDIG, MDC*MSC, IDIFDIG, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JDIFUPP, MDC*MSC, IDIFUPP, IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WAREA, JIDFRHV, MDC*MSC, JIDFRHV, IERR) 30.81
IF (STPNOW()) RETURN 34.01
*** In case of SETUP expand array for setup data ***
*** in case of SETUP expand array for setup data
IF (LSUPP.CT.0) THEN
MSTPDA = 40
CALL DPEXPR (WAREA, JSTPDA, MSTPDA*MSC*MYC, ISTPDA, IERR) 30.81 31.04
B – 26

ws | dekt hydraulics
IF (STPNOW()) RETURN
ENDIF

C

IF ( DYNDEF .OR. ICUR .EQ. 1 ) THEN

*** ILU-CGSTAB solver ***
CALL DPEXPR (WARDE, JBAND , MDC*MSC*9 , IEBAND , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WARDE, JEXACT , MDC*MSC , IEXACT , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WARDE, JRUHV , MDC*MSC , IRUHV , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WARDE, JRIINSOL, 7 , IRINSOL , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WARDE, JSOLUT , MDC*MSC , ISOLUT , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WARDE, JWORK , MDC*MSC*10, IWORK , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WARDE, JPRECON, MDC*MSC*9 , IPRECON , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WARDE, JUPPERI, MSC , IUPPERI , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WARDE, JLOPERI, MSC , ILOPERI , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WARDE, JINFMAT, 10 , IINFMAT , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WARDE, JIIINSOL, 14 , IIIINSOL , IERR) 30.81
IF (STPNOW()) RETURN 34.01
ELSE
CALL DPEXPR (WARDE, JBAND , 0 , IEBAND , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WARDE, JEXACT , 0 , IEXACT , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WARDE, JRUHV , 0 , IRUHV , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WARDE, JRIINSOL, 7 , IRINSOL , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WARDE, JSOLUT , 0 , ISOLUT , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WARDE, JWORK , 0 , IWORK , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WARDE, JPRECON, 0 , IPRECON , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WARDE, JUPPERI, 0 , IUPPERI , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WARDE, JLOPERI, 0 , ILOPERI , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WARDE, JINFMAT, 10 , IINFMAT , IERR) 30.81
IF (STPNOW()) RETURN 34.01
CALL DPEXPR (WARDE, JIIINSOL, 14 , IIIINSOL , IERR) 30.81
IF (STPNOW()) RETURN 34.01
END IF

C

*** wind input term Janssen (1989, 1990) ***

CALL DPEXPR (WARDE, JANYNWD , MDC , IANYNWD , IERR) 30.81
IF (STPNOW()) RETURN 34.01

C

*** For obstacles ***
*** The new array have dimensions (MDC,MSC,ICMAX-1) ***
CALL DPEXPR (WARDE, JOBDF , MDC*MSC*(ICMAX-1), IOBDF , IERR) 30.81

060697
IF (STPNOW()) RETURN 34.01

C

******************************************************************************
C
DATA POOL INQUIRE POINTER
C
******************************************************************************
C
498 CONTINUE
CALL DPINQP (WARDE,'HSAC1', JHSAC1 , PTYPE, IHSAC1, LENREC,IERR) 30.81
CALL DPINQP (WARDE,'HSAC2', JHSAC2 , PTYPE, IHSAC2 ,LENREC,IERR) 30.81
CALL DPOIQ (WAREA,'SACC1' ,JSACC1 ,PTYPE, ISACC1 ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'SACC2' ,JSACC2 ,PTYPE, ISACC2 ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'CG'    ,JCG    ,PTYPE, ICG    ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'CAX'    ,JCAX   ,PTYPE, ICAX   ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'CAY'    ,JCAY   ,PTYPE, ICAY   ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'CAS'    ,JCAS   ,PTYPE, ICAS   ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'CAD'    ,JCAD   ,PTYPE, ICAD   ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'CG0'    ,JCG0   ,PTYPE, ICGO   ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'KNUM'   ,JKNUM  ,PTYPE, IKNUM  ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'JNUMB'  ,JJNUMB ,PTYPE, IJNUMB ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'ALIMM'  ,JALIMM ,PTYPE, IALIMM ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'GROWN'  ,JGROWN ,PTYPE, IGROWN ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'ICDMN'  ,JICDMN ,PTYPE, IIICDMN,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'ISCMN'  ,JISCMMN,PTYPE, IJISCMMN,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'ISCMX'  ,JISCMMX,PTYPE, IJISCMMX,LENREC, IERR) 30.81

C *** quadruplets ***
CALL DPOIQ (WAREA,'WWINT' ,JWWINT,PTYPE, IWWINT ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'WWWGW' ,JWWWGW,PTYPE, IWWWGW,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'AF11'  ,JAF11  ,PTYPE, IAFA11 ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'UE'    ,JUE    ,PTYPE, IUE    ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'SA1'   ,JSA1   ,PTYPE, ISA1   ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'SA2'   ,JSA2   ,PTYPE, ISA2   ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'SFNL'  ,JSFNL  ,PTYPE, ISFNL  ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'DAIC'  ,JDAIC  ,PTYPE, IDAIC  ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'DA1P'  ,JDA1P  ,PTYPE, IDA1P  ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'DA1M'  ,JDA1M  ,PTYPE, IDA1M  ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'DAZC'  ,JDAZC  ,PTYPE, IDAZC  ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'DAZP'  ,JDAZP  ,PTYPE, IDAZP  ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'DAZM'  ,JDAZM  ,PTYPE, IDAZM  ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'DSNL'  ,JDSNL  ,PTYPE, IDSNL  ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'MEMNL' ,JMEMNL,PTYPE, IMEMNL ,LENREC, IERR) 30.81

C *** explicit scheme in frequency space and diffusion model ***
CALL DPOIQ (WAREA,'DIFLOW',JDIFLOW,PTYPE, IDIFLOW,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'DIFDIG',JDIFDIG,PTYPE, IDIFDIG,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'DIFUFF',JDIFUFF,PTYPE, IDIFUFF,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'DIFFRY',JDIFFRY,PTYPE, IDIFFRY,LENREC, IERR) 30.81

C *** ILU-CGSTAB solver (9-02-95) ***
CALL DPOIQ (WAREA,'BAND' ,JBAND ,PTYPE, IBAND ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'EXACT' ,JEXACT,PTYPE, IEXACT ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'RVH'   ,JRZH   ,PTYPE, IRZH   ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'RSOL'  ,JRSOL  ,PTYPE, IRRSOL ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'SOLUT' ,JSOLUT ,PTYPE, ISOLUT ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'WORK'  ,JWORK ,PTYPE, IWORK ,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'PRECON',JPRECON,PTYPE, JPRECON,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'UPPERI',JUPPERI,PTYPE, IUPPERI,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'LOPERI',JLOPERI,PTYPE, ILOPERI,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'INM1N',JINM1N,PTYPE, IINM1N,LENREC, IERR) 30.81
CALL DPOIQ (WAREA,'INM1N',JINM1N,PTYPE, IINM1N,LENREC, IERR) 30.81

C CALL DPOIQ (WAREA,'ANYWJ',JANYWT ,PTYPE, IANYWT ,LENREC, IERR) 30.81
C
C *** For obstacles ***
C CALL DPOIQ (WAREA,'OBDF' ,JOBDF ,PTYPE, IOBDF ,LENREC, IERR) 30.81

040697
C
C *** For setup ***
C
CALL DPOIQ (WAREA,'SETFD',JSETFD ,PTYPE, ISETFD,LENREC, IERR) 30.81 32.02

C IF (ITEST .GE. 50) THEN
WRITE(PRINTF,* ) 'Checking data pool (warea)'
WRITE(PRINTF,* ) 'before start computation'
IERR = 0
CALL DFCHEK (WAREA, IERR)
ENDIF
C
C*******************************************************************************
C
C *** end of data pool organization ***
** skip data pool procedure for second time step **

IF (IT .GT. 1) GO TO 499

** initialise values for determining the accuracy that **
** has been reached **

IERR = -1
CALL DPCHECK (WAREA, IERR)
CALL INSAC (AC2 ,SPCSIG ,COMPDA(1,JDP2) ,
& RWAREA(IHSAC2+1) ,RWAREA(ISACC2+1) )

499 CONTINUE

** store original wind model counter (IWIND) in auxiliary **
** variable IWORG (required for first guess) **

IWORG = IWIND

** Fill the array WX2 and WY2 in case of constant wind **
** and calculate the relative wind speed in presence of **
** a current **

DO 450 ITER = 1, ITERMAX

** When a current is present and the ILU-CGSTAB solver **
** is used it can be that the solver does not converge **
** the counter INOCNV contains the number of geograp. **
** gridpoints in which the solver did not converge **

INOCNV = 0

initialise Dissipation and Leak at 0 at begin of iteration

DO IP = 1, MCGRD
   COMPDA(IP,JDISS) = 0.
   COMPDA(IP,JLEAK) = 0.
ENDDO

** prepare constants and weight factors for nonlinear **
** 4 wave interactions and initialize array MEMNL4 = 0. **
** and set the array for the frequencies > SPCSIG(msc) **

IF ( IQUAD .GE. 1 ) THEN

** For the first iteration compute range for spectral **
** space and some other variables for 4 wave interactions **

IF ( ITER .EQ. 1 ) THEN
   IERR = -1
   CALL DPCHECK (WAREA, IERR)
   CALL FAC4WW (ITER ,XIS ,SNLC1 ,
& DAL1 ,DAL2 ,DAL3 ,SPCSIG,
& WAREA(IAF11+1) ,WAREA(IWWINT+1) ,
& WAREA(IWWAWG+1) ,WAREA(IWWWSG+1) )
END IF

** IQUAD = 3: the nonlinear wave interactions are **
** calculated just once for an iteration. First, **
** set the auxiliary array equal zero before a **
** new iteration **

IF ( IQUAD .EQ. 3 ) THEN
   DO II = 1, MCGRD*MSC*MDC
      WAREA(IMEMNL+II) = 0.
   ENDDO
END IF
END IF

C
*** to obtain a first estimate of energy density in a gridpoint considered we run the SWAN model (in case of generative wind ) in a second generation mode first. ***

C *** After 1 iteration, the options as defined by the user are activated. ***

C *** This first guess is not used in nonstationary computations (NSTRTC>0), or if a restart file was used (ICOND=4) ***

C
IF (NSTRTC.GT.0 .OR. ICOND.EQ.4) THEN
PRECOR = .FALSE.
ELSE
PRECOR = .TRUE.
ENDIF

C
IF ( PRECOR .AND. IWORG .GE. 3 ) THEN
*** third generation wave input ***
IF ( ITER.EQ. 1 ) THEN
*** set auxiliary variables --> ***
bottom friction and surf breaking are not changed
KSTATIC = NSTATIC
ICUR = FCUR
*** deep water model ***
KWIND = IWIND
KCAP = ICAP
KQUAD = IQUAD
*** shallow water model ***
KTRIAD = ITRIAD
*** set maximum change per h with equal 1.e21 ***
GRWOLD = PNUNS(20)
*** set flags equal zero ***
NSTATIC = 0
IWIND = 2
ICAP = 0
IQUAD = 0
ITRIAD = 0
PNUNS(20) = 1.E22
ELSE IF( ITER.EQ. 2 ) THEN
NSTATIC = KSTATIC
ICUR = FCUR
IWIND = KWIND
ICAP = KCAP
IQUAD = IQUAD
ITRIAD = ITRIAD
PNUNS(20) = GRWOLD
ENDIF

C ENDIF

C
*** test output ***
IF (NSTRTC.EQ.0 .AND. ITER.LE. 2 .AND. IWORG.GE. 3) THEN
WRITE(PRINTF,'( ***')
24/MAR
WRITE(PRINTF,'( ***')
6

IF ( ITER.EQ. 1 .AND. ITEST.GE. 10 ) THEN
WRITE(PRINTF,'( ***')
6
ELSE IF( ITER.EQ. 0 .AND. ITEST.GE. 10 ) THEN
WRITE(PRINTF,'( ***')
6
ENDIF

WRITE(PRINTF,001) ITER, PNUNS(20)
2001
2002
2003

WRITE(PRINTF,002) NSTATIC, ICUR, IWIND, ICAP
WRITE(PRINTF,002) NSTATIC, ICUR, IWIND, ICAP
WRITE(PRINTF,002) iQUAD, ITRIAD, IBOT, ISURF
WRITE(PRINTF,002) iQUAD, ITRIAD, IBOT, ISURF
WRITE(PRINTF,002) iQUAD, ITRIAD, IBOT, ISURF
WRITE(PRINTF,002) iQUAD, ITRIAD, IBOT, ISURF
WRITE(PRINTF,'( ***')
2003
2003

ENDIF

C
*** START ITERATION PROCESS WITH 4 SWEEPS ***

*** loop over sweep directions ***

DO 410 SWPDIR = 1, 4
   IF (SWPDIR.EQ.1) THEN
     KSX = -1
     KSY = -1
     DDX = +DX
     DDDY = +DY
     IX1 = 2
     IX2 = MXC
     IY1 = 2
     IY2 = MYC
   ELSE IF (SWPDIR.EQ.2) THEN
     KSX = +1
     KSY = -1
     DDX = -DX
     DDDY = +DY
     IX1 = MXC-1
     IX2 = 1
     IY1 = 2
     IY2 = MYC
   ELSE IF (SWPDIR.EQ.3) THEN
     KSX = +1
     KSY = +1
     DDX = -DX
     DDDY = -DY
     IX1 = MXC-1
     IX2 = 1
     IY1 = MYC-1
     IY2 = 1
   ELSE IF (SWPDIR.EQ.4) THEN
     KSX = -1
     KSY = +1
     DDX = +DX
     DDDY = -DY
     IX1 = 2
     IX2 = MXC
     IY1 = MYC-1
     IY2 = 1
   ENDIF

*** change values of variables for one-dimensional run ***

IF (ONED ) THEN
   IY1 = 1
   IY2 = 1
   KSY = 0
ENDIF

IF (SCREEN.NE.PRINTF) THEN
   IF (NISTATC.EQ.1) THEN
      WRITE(SCREEN,313) CHTIME, IT, ITER, SWPDIR
      313 FORMAT ('+time ', A18, ', step ', I4, '; iteration ' , I2, '; sweep ', I1)
   ELSE
      WRITE(PRINTF,314) ITER, SWPDIR
      314 FORMAT (' iteration ', I2, '; sweep ', I1)
   ENDIF
   IF (ITEST .GE. 100) THEN
      WRITE(PRINTF,* ) '......CHECKING DATA POOL WAREA........'
      CALL DPCHKEW (WAREA, IERR)
      WRITE(PRINTF,* ) '.......POOL WAREA CHECKED........'
   ENDIF
ENDIF

IF (ONED ) THEN
   IYSTEP = 1
ENDIF
ELSE
 IYSTEP = KSY
ENDIF

DO 400 40 = 1, 1, Y1, Y2, -IYSTEP
DO 390 IX = 1, X1, X2, -KSY
CALL SWOMP(U (SPWDIR, KSY , KSY ,
   IX , IX , DX, DXX ,
   DOY , DT , INLC1 ,
   DAL1 , DALS2 ,
   XIS , SWTSDA , INCNV ,
   AC2 , COMPDA , SPDIR ,
   DPC, YXTST , ITER , 30.72
   RWAREA (IGCW+1) , RWAREA (ICG+1) , RWAREA (ICG+1) ,
   RWAREA (ICA+1) , RWAREA (ICAY+1) , RWAREA (ICAS+1) ,
   RWAREA (ICAD+1) , RWAREA (ISWMAT+1) , LWAREA (ISWMAT+1) ,
   RWAREA (IKNUM+1) ,
   RWAREA (IALIMW+1) , LWAREA (IGROWW+1) , RWAREA (IAF11+1) ,
   RWAREA (IAF1+1) , RWAREA (IAF1+1) , RWAREA (IAF1+1) ,
   RWAREA (IDA2+1) , RWAREA (IDA2+1) , RWAREA (IDA2M+1) ,
   RWAREA (ISFNL+1) , RWAREA (IDSNL+1) , RWAREA (IMMNL+1) ,
   RWAREA (IDCMIN+1) , RWAREA (IDCMAX+1) , RWAREA (IDCMAX+1) ,
   RWAREA (IWIND+1) , RWAREA (IWANG+1) , RWAREA (IWANG+1) ,
   RWAREA (ICOLUT+1) , RWAREA (IDIFLOW+1) ,
   RWAREA (IDIFDG+1) , RWAREA (IDIFUPP+1) , RWAREA (IDIFPRV+1) ,
   RWAREA (IBAND+1) , RWAREA (IXEAC+1) , RWAREA (IRHV+1) ,
   RWAREA (IRINSOL+1) , RWAREA (ISOLUT+1) , RWAREA (IWORK+1) ,
   RWAREA (IPREC+1) , RWAREA (IPREC+1) , RWAREA (ILOGER+1) ,
   RWAREA (IINFMA+1) , RWAREA (IINSM+1) , RWAREA (IINSM+1) ,
   RWAREA (IISCMAX+1) ,
   LWAREA (INNYWND+1) , AC1 , IT , 30.90
  PREC1R , XGRID , YGRID , 30.72
  KGIPMT , CROSS , OSTA , 300597
  RWAREA (IODEP+1) ,
ENDIF
IF (STPNOW()) RETURN
CONTINUE
400 CONTINUE
410 CONTINUE

C

*** store the source terms for test gridpoints ***

C

*** in the file SOURCE ***

C

IF (NPTST.GT.0 .AND. NSTATM.EQ.0 ) THEN
   IF (IFPAR.GT.0) WRITE (IFPAR, 12) ITER
   IF (IFSID.GT.0) WRITE (IFSID, 12) ITER
   IF (IFS2D.GT.0) WRITE (IFS2D, 12) ITER
   FORMAT (14, T41, 'iteration')
   ITER = -1
   CALL DPCHEK (WAREA, IERR)
   CALL PLTSCR (SWTSDA (1,1,1,1,JPWND), SWTSDA (1,1,1,1,JPWND))
   SWTSDA (1,1,1,1,JPWCA), SWTSDA (1,1,1,1,JPWCA)
   SWTSDA (1,1,1,1,JPWBR), SWTSDA (1,1,1,1,JPWCA)
   SWTSDA (1,1,1,1,JPW), SWTSDA (1,1,1,1,JPW)
   AC2 , SPSCIG , 40.00
   COMPDA (1, JD2) , YXTST ,
   KQUAD , KGIPMT )
ENDIF

C

*** compute wave-induced setup ***

C

IF (LSETUP.GT.0) THEN
   CALL SETUP (KGIPMT, MSTDPA, WAREA (IAEDS(WAREA, JSTPD)),
   AC2, COMPDA (1, JD2), COMPDA (1, JDPAS)),
   COMPDA (1, JSTUP),
   COMPDA (1, JWFR1),
   XKRES , YGRID , SPSCIG, SPDIR, IT, ITER
   IF (STPNOW()) RETURN
ENDIF

C

*** check if numerical accuracy has been reached ***

C
C IERR = -1
CALL DPCHKEK (WAREA, IERR)
CALL SACCUR (COMPDA(1, JDF2),
  AC2, SPCSIG, ACCUR, RWAREA(IHSC1+1), RWAREA(IHSC2+1), RWAREA(ISACC1+1),
  RWAREA(ISACC2+1), COMPDA(1, JDHS), COMPDA(1, JDTHM))

*** info regarding the iteration proces and the accuracy ***
WRITE(PRINTF, 112) ACCUR, PNUMS(4)
IF (NSTATC.EQ.0) WRITE(SCREEN, 112) ACCUR, PNUMS(4)
112 FORMAT(' accuracy OK in ', F6.2,
  ' % of wet grid points (', F6.2, ' % required)', /
  ' number of points for which the ILU solver did ***
  not converge ***
  IF ((DYNDEP .OR. ICUR.EQ.1) .AND. INOCONV .NE. 0) THEN
    WRITE(PRINTF, 122) INOCONV
    WRITE(SCREEN, 122) INOCONV
    122 FORMAT(2X,'ILU-CGSTAB solver: no convergence in ', I4,
      ' gridpoints')
ENDIF

*** if accuracy has been reached then the iteration ***
*** can be terminated ---> goto 470 ***
IF (ACCUR.GE.PNUMS(4)) GOTO 470

*** write value 1 to array with source term ***
*** (another iteration) ***
450 CONTINUE
470 CONTINUE

IF (NPTST.GT.0 .AND. NSTATM.EQ.1)
  THEN
    IF (IFPAR.GT.0) WRITE (IPPAR, 11) CTIME
    IF (IFS1D.GT.0) WRITE (IFS1D, 11) CTIME
    IF (IFS2D.GT.0) WRITE (IFS2D, 11) CTIME
    11 FORMAT (A, T41, 'date-time')
  ENDIF

IERR = -1
CALL DPCHKEK (WAREA, IERR)
CALL PLTSRC (SWTSDA(1,1,1, JPWNDAD), SWTSDA(1,1,1, JPWNBDB),
  SWTSDA(1,1,1, JPWCRAP), SWTSDA(1,1,1, JPWFRD),
  SWTSDA(1,1,1, JPBWRK), SWTSDA(1,1,1, JPS4),
  SWTSDA(1,1,1, JPD4), SWTSDA(1,1,1, JPTRI),
  AC2, SPCSIG, COMPDA(1, JDF2), XXTST,
  KQQUAD, KGRFNT)

END IF

*** if calculation is steady: NSTATC = 0 GOTO 550 ***
500 CONTINUE
550 CONTINUE

* *** End of subroutine SWCOMP ***
RETURN
END
B.4 Subroutine SWOEXF.FOR

C*******************************************************************************
C
SUBROUTINE SWOEXF (MIP ,XC ,YC ,VOQR ,
& VQ ,AC2 ,DER2 ,SPCSIG ,
& WK ,CG ,SPCDIR ,NE ,
& NED ,KGRPM ,XGRID ,YGRID ,
&
C*******************************************************************************
C
--|--------------------------------------------------------------------------|--
C
BY USING THE SWAN SOFTWARE, YOU ARE CONSENTING TO BE BOUND BY
C
THIS AGREEMENT.
C
C
Delft University of Technology grants you a non-exclusive license
C
to use the SWAN Software free of charge.
C
C
DISCLAIMER OF WARRANTY.
C
C
The Swan Software is provided on an "AS IS" basis, without
C
warranty of any kind, including without limitation the warranties
C
of merchantability, fitness for a particular purpose and
C
non-infringement. The entire risk as to the quality and
C
performance of the Software is borne by you. You must determine
C
that the Software sufficiently meets your requirements.
C
C
LIMITATION OF LIABILITY.
C
C
UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, TORT, CONTRACT,
C
OR OTHERWISE, SHALL DELFT UNIVERSITY OF TECHNOLOGY BE LIABLE TO
C
YOU OR ANY OTHER PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL,
C
OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT
C
LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER
C
FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES
C
OR LOSSES. IN NO EVENT WILL DELFT UNIVERSITY OF TECHNOLOGY BE
C
LIABLE FOR ANY DAMAGES, EVEN IF DELFT UNIVERSITY OF TECHNOLOGY
C
SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES, OR
C
FOR ANY CLAIM BY ANY OTHER PARTY. YOU AGREE TO INDEMNIFY AND HOLD
C
DELFT UNIVERSITY OF TECHNOLOGY HARMLESS WITH RESPECT TO ALL
C
CLAIMS BY THIRD PARTIES ARISING OUT OF YOUR USE OF THE RESULTS
C
OR OPERATION OF THE SOFTWARE.
C
C
TITLE,
C
Title, ownership rights, and intellectual property rights in the
C
Software shall remain with the authors of the Software. The
C
Software is protected by the copyright laws and treaties. Title
C
and related rights in the content accessed through the Software
C
is the property of the applicable content owner and may be
C
protected by applicable law. This license gives you no rights to
C
such content.
C
C
MISCELLANEOUS.
C
You may not remove any proprietary notices or labels on the
C
Software, nor remove this disclaimer. If any provision of this
C
Agreement is held to be unenforceable, such provision shall be
reformed only to the extent necessary to make it enforceable.

This Agreement shall be governed by the law of the Netherlands.

---------------------------------

IMPLICIT NONE
INCLUDE 'ocpcomm4.inc'
INCLUDE 'swcomm1.inc'
INCLUDE 'swcomm3.inc'
INCLUDE 'swcomm4.inc'

0. Authors

30.72: IJsbrand Haagsma
30.80: Nico Boolij
30.81: Annette Kieftenburg
30.82: IJsbrand Haagsma

1. Updates

30.55, Mar. 97: Procedure updated for curvilinear coordinates basics is described in SWANDOC.WP5 comp. grid point coordinates are new arguments
30.72, Oct. 97: Logical function EQREAL introduced for floating point comparisons
30.72, Feb. 98: Introduced generic names XGRID, YGRID and SPCSIG for SWAN
30.80, Apr. 98: Provision for 1D computation
30.82, Oct. 98: Updated description of several variables
30.81, Dec. 98: Argument list KSCI1 adjusted
30.81, Dec. 98: Implicit none added, force for point surrounded by dry points set to 0.

2. Purpose
Calculates wave-driven force (output quantity IVTYPE=20)

3. Method
Radiation stresses are defined as:

\[
S_{xx} = \rho \text{grav } ((N \cos^2(\theta) + N - 1/2) \text{sig Ac}) \ d \text{sig} \ d \theta
\]

\[
S_{xy} = \rho \text{grav } ((N \sin(\theta) \cos(\theta) \text{sig Ac}) \ d \text{sig} \ d \theta
\]

\[
S_{yy} = \rho \text{grav } ((N \sin^2(\theta) + N - 1/2) \text{sig Ac}) \ d \text{sig} \ d \theta
\]

The force in x-direction and y-direction are:

\[
F_x = - (8S_{xx}/8x + 8S_{xy}/8y)
\]

\[
F_y = - (8S_{xy}/8x + 8S_{yy}/8y)
\]

where \(\partial\) denotes the partial derivative.

The value of \(N\) and its derivative w.r.t. the depth are calculated in the KSCI1 subroutine.

First the gradients with respect to \(i\) and \(j\) (comp. grid counters) are computed, then these are transformed into gradients in \((x,y)\)

4. Argument variables

AC2  input  action density
CG   local  group velocity in output point
DEP2 input depth at comp. grid points
KGRNPT input  index for indirect addressing
MIP  input  number of output points
NE   local ratio of group and phase velocity
NED  local derivative of NE with respect to depth
SPCDIR input  \((\cdot,1)\): spectral directions (radians)
         \((\cdot,2)\): cosine of spectral directions
C (*,3); sine of spectral directions 30.82
C (*,4); cosine^2 of spectral directions 30.82
C (*,5); cosine*sin of spectral directions 30.82
C (*,6); sin^2 of spectral directions 30.82
C
C SPCSIG input relative frequencies in computational domain in
C sigma-space 30.72
C XC, YC input comp. grid coordinates of output point
C XCGRID input coordinates of computational grid in x-direction 30.72
C YCGRID input coordinates of computational grid in y-direction 30.72
C VOQR input location in VOQ of a certain output quant.
C VOQ output values of output quantities
C WK local wavenumber in output point
C
C INTEGER MIP, VOQR(*) , XGRIDNT (MXC, MYC) 30.21
C REAL AC2 (MDC, MSC, MGRID), CG(*), DEP2 (MCGRID), NE(*), NED(*) 30.82
C REAL SPCDIR (MDC, 6) 30.72
C REAL SPCSG (MSC) 30.72
C REAL XC (MIP), YC (MIP)
C REAL XGRID (MXC, MYC), YGRID (MXC, MYC) 30.72
C REAL VOQ (MIP, *), WK(*)
C LOGICAL EQREAL 30.72
C
C 5. Parameter variables
C ---
C 6. Local variables
C
C ACWAVE Action density in output point
C ACWL, ACWJ dAC/dI and dAC/dJ
C ACWY X-gradient of local action density
C ACX, ACY Y-gradient of local action density
C DDETA determinant
C DDI, DJJ dDEP/dI and dDEP/dJ
C DX, DDX, DDI spatial depth gradients
C DJX, DJY coefficients for transformation from I-gradient
C to (X,Y)-gradients
C
C DS2
C DX, DJJ dX/dI and dX/dJ
C DDX, DDI dy/dI and dy/dJ
C DEP depth
C DEPLOC local depth
C FX, FY (preliminary) forces in X-, and Y-direction
C FXADD, FYADD Cumulated forces/(RHO*GRAV) per frequency and direction
C ID counter for steps in direction
C IENT number of entries
C IND1, IND2,
C IND3, IND4,
C IND5, IND6,
C IND7, IND8,
C IND9 indirect adresses
C IP counter
C IS counter for sigma
C IVTYPE
C JX counter in X-direction
C JXLO, JXUP lower resp. upper gridpoint number of point under consideration
C in X-direction
C JY counter in Y-direction
C JYLO, JYUP lower resp. upper gridpoint number of point under consideration
C in Y-direction
C NAX, NAY derivative of N * Ac.dens. = N * E / Sigma, w.r.t. X
C or Y, respectively.
C ONX, ONY Indicates whether or not a output point lies on a
C computational point or not
C RRD1, RRDJ multiplication factor: 0.5 in case of two-sided or 1 in case
C of one-sided differential
C SIG dummy variable
C SXLO, SXUP weight coefficients for the lower and upper x-level of the point under consideration, respectively.
C SYLO, SYUP weight coefficients for the lower and upper y-level of the point under consideration, respectively.
REAL ACWAV, ACWI, ACWX, ACWY, DDET, DDI, DDJ, DDX,
& DDY, DIX, DIY, DJX, DJY, DS2, DXI, DXJ, DYL, DYP, DEP,
& DEPLOC, FX, FY, FXADD, FYADD, NAX, NAY, RDOI, RRDJ,
& SIG, SXLO, SXUP, SYLO, SYUP
INTEGER ID, IENT, INDI, IND2, IND3, IND4, IND5, IND6, IND7,
& IND8, IND9, IP, IS, IVTYPE, JX, JXLO, JXUP, JY,
& JYLJ, JYUP
LOGICAL ONX, ONY

7. Common Blocks used
---

8. Subroutines used
C
C STRACE
KSCIPI calculates WK, CG, N and ND

9. Subroutines calling
C
C SWOUTP (SWAN/OUTP)

10. Error messages
---

11. Remarks
C
C -In determining derivatives one-sided differences are used at
C border meshes; for output points inside a mesh derivative
C over one step is taken; for output points on a computational
C grid point a central derivative is taken
C -A margin of 0.01 m is taken outside the computational grid.
C -The range of the counter runs from 1 to MXC; the range of XC(IP) runs
C from 0 to MXC-1!

C Counter: 1 2 JX JX+1 JX+2 MXC-1 MXC
C
C |=|--------|--|--|--|--|--|--|--|--|--|=
C XC: 0 1 JX MXC-2 MXC-1

-C-Order in which they are treated:
C
C |=|--------|--------|--------|--------|--------|
C Order: A B C D E

12. Structure
C
C--------------------------------------------------------------------------
C For all output points do
C Initialize both force components as 0
C Determine neighbouring points to be used for gradients in
C X and Y
C Call KSCIPI (determine derivative of N with respect to depth)
C For all spectral components do
C determine derivative of nE with respect to X
C determine derivative of nE with respect to Y
C calculate contribution to force components
C--------------------------------------------------------------------------

13. Source text
C
SAVE IENT
DATA IENT /0/
CALL STRACE (IENT, 'SWOEXP')
C
IVTYPE = 20
C
loop over all output points
DO 800 IP=1,MIP
DEP = VOQ(IP,VOQR(4))
IF (DEP.LE.0.) GOTO 700
IF (EQREAL(DEP,OVEXCV(4))) GOTO 700 30.72
IF (X(IP).LT.-0.01) GOTO 700
IF (XC(IP).GT.REAL(MXC-1)+0.01) GOTO 700
IF (YC(IP).LT.-0.01) GOTO 700
IF (YC(IP).GT.REAL(MYC-1)+0.01) GOTO 700
C
C first the action density spectrum is interpolated
C
FX = 0.
FY = 0.
JX = NINT(XC(IP))
RRDI = 1.
ONX = .FALSE.
IF (JX.EQ.0) THEN
    JXLO = 1
    JXUP = 2
    JX = 1
    SXUP = XC(IP)
    SXLO = 1.-SXUP
ELSE IF (JX.EQ.MXC-1) THEN
    JXLO = MXC-1
    JXUP = MXC
    SXLO = REAL(MXC-1)-XC(IP)
    SXUP = 1.-SXLO
    JX = JX+1
ELSE IF (XC(IP).LT.REAL(JX)-0.01) THEN
    JXLO = JX
    JXUP = JX+1
    SXLO = REAL(JX)-XC(IP)
    SXUP = 1.-SXLO
    JX = JX+1
ELSE If (XC(IP).GT.REAL(JX)+0.01) THEN
    JXLO = JX+1
    JXUP = JX+2
    SXUP = XC(IP)-REAL(JX)
    SXLO = 1.-SXUP
    JX = JX+1
ELSE
    JXLO = JX
    JXUP = JX+2
    RRDI = 0.5
    JX = JX+1.
    ONX = .TRUE.
ENDIF
IF (JNED) THEN 30.80
    JYLO = 1
    JYUP = 1
    JY = 1
    RRDJ = 0.
    ONY = .TRUE.
ELSE
    JY = NINT(YC(IP))
    RRDJ = 1.
    ONY = .FALSE.
ENDIF
IF (JY.EQ.0) THEN
    JYLO = 1
    JYUP = 2
    JY = 1
    SYUP = YC(IP)
    SYLO = 1.-SYUP
ELSE IF (JY.EQ.MYC-1) THEN
    JYLO = MYC-1
    JYUP = MYC
    SYLO = REAL(MYC-1)-YC(IP)
    SYUP = 1.-SYLO
    JY = JY+1
ELSE IF (YC(IP).LT.REAL(JY)-0.01) THEN
    JYLO = JY
    JYUP = JY+1
    SYLO = REAL(JY)-YC(IP)
    SYUP = 1.-SYLO
ENDIF
JY = JY+1
ELSE IF (YD8(JP).GT.REAL(JY)+0.01) THEN
    JYLO = JY+1
    JYUP = JY+2
    SYUP = YD8(JP)-REAL(JY)
    SYLO = 1.-SYUP
    JY = JY+1
ELSE
    JYLO = JY
    JYUP = JY+2
    RRDJ = 0.5
    JY = JY+1
ENDIF
ENDIF

*** Using indirect addressing for arrays AC2 and DEP2 ***

IND1 = KGRFNT(JXLO,JYLO)
IND2 = KGRFNT(JXUP,JYLO)
IND3 = KGRFNT(JXUP,JYUP)
IND4 = KGRFNT(JXLO,JYUP)
IND5 = KGRFNT(JXLO,JY)
IND6 = KGRFNT(JXUP,JY)
IND7 = KGRFNT(JX,JYLO)
IND8 = KGRFNT(JX,JYUP)
IND9 = KGRFNT(JX,JY)

IF (ONY) THEN
    IF (DEP2(IND5).LE.DEPMIN) GOTO 700
    IF (DEP2(IND6).LE.DEPMIN) GOTO 700
ELSE
    IF (DEP2(IND1).LE.DEPMIN) GOTO 700
    IF (DEP2(IND2).LE.DEPMIN) GOTO 700
    IF (DEP2(IND3).LE.DEPMIN) GOTO 700
    IF (DEP2(IND4).LE.DEPMIN) GOTO 700
ENDIF

IF (ONX) THEN
    IF (DEP2(IND7).LE.DEPMIN) GOTO 700
    IF (DEP2(IND8).LE.DEPMIN) GOTO 700
ELSE
    IF (DEP2(IND1).LE.DEPMIN) GOTO 700
    IF (DEP2(IND2).LE.DEPMIN) GOTO 700
    IF (DEP2(IND3).LE.DEPMIN) GOTO 700
    IF (DEP2(IND4).LE.DEPMIN) GOTO 700
ENDIF

determine depth and (x,y) derivatives w.r.t. i and j

IF (ONY) THEN
    DDI = RDI * (DEP2(IND6)-DEP2(IND5))
    DXI = RDI * (XGRID(JXUP,JY)-XGRID(JXLO,JY))
    DDI = RDI * (YGRID(JXUP,JY)-YGRID(JXLO,JY))
    ELSE
    DDI = RDI * (SYUP*DEP2(IND3)-DEP2(IND4)) +
    SYLO*(DEP2(IND2)-DEP2(IND1))
    DXI = RDI * (SYUP*XGRID(JXUP,JYUP)-XGRID(JXLO,JY)) +
    SYLO*(XGRID(JXUP,JYLO)-XGRID(JXLO,JY))
    DDI = RDI * (SYUP*XGRID(JXUP,JYUP)-XGRID(JXLO,JYUP)) +
    SYLO*(YGRID(JXUP,JYLO)-YGRID(JXLO,JYLO))
ENDIF

IF (ONX) THEN
    DDJ = RDJ * (DEP2(IND8)-DEP2(IND7))
    DXJ = RDJ * (XGRID(JX,JYUP)-XGRID(JX,JY))
    DDJ = RDJ * (SXUP*DEP2(IND4)-DEP2(IND3)) +
    SXLO*(DEP2(IND2)-DEP2(IND1))
    DXJ = RDJ * (SXUP*XGRID(JXLO,JY)-XGRID(JX,JY)) +
    SXLO*(XGRID(JXLO,JYLO)-XGRID(JX,JY))
    DDJ = RDJ * (SXUP*XGRID(JXLO,JY)-XGRID(JX,JY)) +
    SXLO*(YGRID(JXLO,JYLO)-YGRID(JXLO,JY))
ENDIF

coefficients from transformation from (i,j)-gradients to (x,y)-gradients

w. | delft hydraulics
IF (JXUP.EQ.JXLO .AND. JYUP.EQ.JYLO) THEN
  C  point surrounded by dry points
  DIX = 0.
  DJY = 0.
ELSE IF (JXUP.EQ.JXLO) THEN
  C  no forces in i-direction
  DS2 = DXI**2 + DYI**2
  DIX = 0.
  DIX = DXI/DS2
ELSE IF (JYUP.EQ.JYLO) THEN
  C  no forces in j-direction
  DS2 = DXI**2 + DYJ**2
  DIX = DXI/DS2
ELSE
  C  coefficients for transformation from (i,j)-gradients to (x,y)-gradients
  DDET = DXI*DIJ - DXJ*DI
  DDI = DXI / DDET
  DDI = DXI / DDET
  DJ = DXI /
ENDIF
C  spatial depth gradients:
  DDX = DDI*DI + DDJ*DJX
  DYY = DDI*DI + DDJ*DJY
C
IF (ITEST.EQ.80 .OR. IOUTES .GE. 20) WRITE (PRTEST, 88) IP,
  C  JXLO, JXUP, JYLO, JYUP ,SKLO, SKUP, SYLO, SYUP,
  C  DIX, DJY, DIX, DJY
C
88 FORMAT (' SWOEXP ', 516, 2X, 4F7.4, 2X, 4E12.4)
C
C  compute NE and NED
C
DEPOLC = VOQ(IP,VOQR(4))
CALL KSCP1 (MSC, SPCSIG, DEPOLC, WK, CG, NE, NED)
C
IF (ITEST.EQ.100 .OR. IOUTES .GE. 20) THEN
  WRITE (PRTEST, 98) DEPOLC, DDX, DYY
98 FORMAT (' depth & gradient ', 4(I4,F9.4))
  DO 100 IS = 1, MIN(MSC,20)
    WRITE (PRTEST, 99) IS, SPCSIG(IS), NE(IS),
      NED(IS)
99 FORMAT (' i, SPCSIG, N, Nd ', I2, 3(I4, E12.4))
  CONTINUE
ENDIF
C
C  DO 300 ID = 1, MDC
    DO 290 IS = 1, MSC
18/MAR

SIG = SPCSIG(IS)
C
C  ACWAV is local action density
C
IF (ONX.AND.ONY) THEN
  ACWAV = AC2(ID,IS,IND9)
ELSE IF (ONX) THEN
  ACWAV = SYLO * AC2(ID,IS,IND7) +
    SYUP * AC2(ID,IS,IND8)
ELSE IF (ONY) THEN
  ACWAV = SXLO * AC2(ID,IS,IND5) +
    SXUP * AC2(ID,IS,IND6)
ELSE
  ACWAV = SXLO *(SYLO * AC2(ID,IS,IND1) +
    SYUP * AC2(ID,IS,IND4)) +
  SXUP *(SYLO * AC2(ID,IS,IND2) +
    SYUP * AC2(ID,IS,IND3))
ENDIF
C
ACWX is X-gradient of local action density, ACWY is Y-gradient

IF (ONY) THEN
   ACWI = RRDI * (AC2(ID,IS,IND6) - AC2(ID,IS,IND5))
ELSE
   ACWI = RRDI * (SYLO * (AC2(ID,IS,IND2) - AC2(ID,IS,IND1)) + SYUP * (AC2(ID,IS,IND3) - AC2(ID,IS,IND4)))
ENDIF

IF (ONX) THEN
   ACWJ = RBDJ * (AC2(ID,IS,IND8) - AC2(ID,IS,IND7))
ELSE
   ACWJ = RBDJ * (SXLO * (AC2(ID,IS,IND4) - AC2(ID,IS,IND1)) + SXUP * (AC2(ID,IS,IND3) - AC2(ID,IS,IND2))
ENDIF

spatial action density gradients:
ACWX = ACWI*DIX + ACWJ*DJX
ACWY = ACWI*DIY + ACWJ*DJY

NAX is the derivative of N * Ac.dens. w.r.t. X
So NAX = ∂(N*Ac)/∂X = Ac*∂N/∂X + N*∂Ac/∂X

where ∂ denotes the partial derivative.

Further note that that ∂N/∂X = ∂N/∂Depth * ∂Depth/∂X
    = NED * DX

Analogously for NAY.

NAX = NE(IS) * ACWX + NED(IS) * DDX * ACNAV
NAY = NE(IS) * ACWX + NED(IS) * DDDY * ACNAV
FXADD = -(SPCDIR(ID,4) + 1.1) * NAX - 0.5 * ACWX + 20.44
FYADD = -(SPCDIR(ID,5) + 1.1) * NAY - 0.5 * ACWY + 20.44

integration
FX = FX + SIG * FXADD
FY = FY + SIG * FYADD

290 CONTINUE
300 CONTINUE

FX = RHO * GRAV * FX * DDR * FRINTF
FY = RHO * GRAV * FY * DDR * FRINTF
VOQ(IP,VOQR(IVTYPE)) = (COSQ*FX - SINQ*FY)
VOQ(IP,VOQR(IVTYPE)+1) = (SINQ*FX + COSQ*FY)
GOTO 800

points on land: assign exception value

700 VOQ(IP,VOQR(IVTYPE)) = OVEQCV(IVTYPE)
VOQ(IP,VOQR(IVTYPE)+1) = OVEQCV(IVTYPE)

800 CONTINUE

RETURN

end of subroutine SWOEXF
END