# SWAN

## IMPLEMENTATION MANUAL

**SWAN Cycle III version 40.51**

# SWAN IMPLEMENTATION MANUAL


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

# Contents

# Chapter 1

# Introduction

This Implementation Manual is a part of the total material to implement the SWAN wave model on your computer system. The total material consists of:

- the SWAN source code,
- the SWAN (serial) executable for MS Windows,
- the User Manual,
- this Implementation Manual,
- the Technical documentation,
- the SWAN programming rules,
- utilities and
- some test cases.

All of the material can be found on the following SWAN web page *http://www.fluidmechanics.tudelft.nl/swan/download/info.htm*http://www.fluidmechanics.tudelft.nl/sw

On the SWAN home page *http://www.fluidmechanics.tudelft.nl/swan/index.htm*http://www.fluidmecha general information is given about the functionalities, physics and limitations of SWAN. Moreover, the modification history of SWAN is given. Finally, information on support, links to the related web pages and various free software are provided.

After downloading the material, you may choose between

- direct usage of the SWAN executable for Windows and
- implementation of SWAN on your computer system.

If you want to use the SWAN executable available on the SWAN web site, please read Chapters 5 and 6 for further information.

For the purpose of implementation, you have access to the source code of SWAN and additional files, e.g. for testing SWAN. Please read the copyright in this manual and in the source code with respect to the terms of usage and distribution of SWAN. You are permitted to implement SWAN on your computer system.

Implementation involves the following steps:

1. Copying the source code from the SWAN web page to the computer system on which you want to run SWAN.
2. If necessary, applying patches for an upgrade of the source code due to e.g., bug fixes, new features, etc.
3. Making a few adaptions in installation-dependent parts of the code.
4. Compiling and linking the source code to produce an executable of SWAN.
5. Testing of the executable SWAN.

After the last step you should have the executable SWAN ready for usage. Note that steps 3 and 4 can be done fully automatically.

## 1.1 The material

The downloaded file `swan4051.tgz` contains the SWAN source code. You can unzip this file either with WinZip (in case of Windows) or with the command `tar xzf` (in case of UNIX or Linux). The SWAN source code consists of the following files:

| | | |
|---|---|---|
| main program | : | swanmain.ftn |
| pre-processing routines | : | swanpre1.ftn |
| | | swanpre2.ftn |
| computational routines | : | swancom1.ftn |
| | | swancom2.ftn |
| | | swancom3.ftn |
| | | swancom4.ftn |
| | | swancom5.ftn |
| post-processing routines | : | swanout1.ftn |
| | | swanout2.ftn |
| service routines | : | swanser.ftn |
| routines for support parallel MPI runs | : | swanparll.ftn |
| routines for installation | : | ocpids.ftn |
| command reading routines | : | ocpcre.ftn |
| miscellaneous routines | : | ocpmix.ftn |
| modules | : | swmod1.ftn |
| | | swmod2.ftn |
| | | swmod3.ftn |

The source code is written in fixed form Fortran 90. Depending on your system, the extension may be `for`, `f` or `F`. The conversion from `ftn` to one of these extensions can be done automatically or manually; see Chapter 3.

You are allow to make changes in the source code of SWAN, but Delft University of Technology will not support modified versions of SWAN. If you ever want your modifications to be implemented in the authorized version of SWAN (the version on the SWAN web page), you need to submit these changes to the SWAN team (*swan-info-citg@tudelft.nl*).

The source code is being attended with the following files:
The Latex files (*.tex) can be read and written by any editor and can be compiled with LaTeX $2_\varepsilon$. This compilation can also be done automatically; see Section 3.3.

On the SWAN web page, you also find some test cases with some output files for making a configuration test of SWAN on your computer. You may compare your results with those in the provided output files.

| | | |
|---|---|---|
| installation procedures | : | INSTALL.README |
| | | Makefile |
| | | Makefile.latex |
| | | macros.inc |
| | | getcmpl |
| | | platform.pl |
| | | switch.pl |
| | | adjlfh.pl |
| run procedures | : | SWANRUN.README |
| | | swanrun |
| | | swanrun.bat |
| machinefile for parallel MPI runs | : | machinefile |
| edit file | : | swan.edt |
| for conversion of spectra | : | convrt1d.for |
| | | cvspec1d.for |
| | | cvspec2d.for |
| documentations | : | swanuse.tex |
| | | swanuse.pdf |
| | | swanimp.tex |
| | | swanimp.pdf |
| | | swantech.tex |
| | | swantech.pdf |
| | | swanpgr.tex |
| | | swanpgr.pdf |
| | | latexfordummies.tex |
| | | latexfordummies.pdf |
| | | conc.lst |
| | | *.eps |
| | | *.ps |

# Chapter 2

# Use of patch files

Between releases of authorised SWAN versions, it is possible that bug fixes or new features are published on the SWAN web page. These are provided by patch files that can be downloaded from the web site. Typically, a patch can be installed over the top of the existing source code. Patches are indicated by a link to **patchfile**. The names refer to the current version number supplemented with letter codes. The first will be coded 'A' (i.e. 40.51.A), the second will be coded 'B', the third will be coded 'C', etc. The version number in the resulting output files will be updated to 40.51ABC, indicating the implemented patches.

To use a patch file, follow the next instructions:

1. download the file (right-click the file and choose *save link as*)
2. place it in the directory where the source code of SWAN is located
3. execute `patch -p0 < patchfile`

After applying a patch or patches, you need to recompile the SWAN source code.

It is important to download the patch and not cut and paste it from the display of your web browser. The reason for this is that some patches may contain tabs, and most browsers will not preserve the tabs when they display the file. Copying and pasting that text will cause the patch to fail because the tabs would not be found. If you have trouble with patch, you can look at the patch file itself.

Note to UNIX/Linux users: the downloaded patch files are MS-DOS ASCII files and contain carriage return (CR) characters. To convert these files to UNIX format, use the command `dos2unix`. Alternatively, execute `cat 40.51.[A-C] | tr -d '\r' | patch` that apply the patch files 40.51.A to 40.51.C to the SWAN source code at once after which the conversion is carried out.

Note to Windows users: `patch` is a UNIX command. Download the patch program from the SWAN web site, which is appropriate for Windows operating system (NT/2000/XP).

# Chapter 3

# Installation of SWAN on your computer

The portability of the SWAN code between single processor machines is guaranteed by the use of standard ANSI FORTRAN 90. Hence, virtually all Fortran compilers can be used for installing SWAN. See also the manual Programming rules.

The SWAN code is parallelized, which enables a considerable reduction in the turn-around time for relatively large CPU-demanding calculations. Two parallelization strategies are available:

- A message passing modelling is employed based on the Message Passing Interface (MPI) standard that enables communication between independent processors. Only simple point-to-point and collective communications have been employed. Hence, users can optionally run SWAN on a cluster of PC nodes.
- The computational kernel of SWAN contains a number of OpenMP compiler directives, so that users can optionally run SWAN on shared-memory supercomputers.

The material on the SWAN web site provides a `Makefile` and two Perl scripts (`platform.pl` and `switch.pl`) that enables the user to quickly install SWAN on the computer in a proper manner. For this, the following platforms, operating systems and compilers are supported:

| **platform** | **OS** | **F90 compiler** |
|---|---|---|
| SGI Origin 3000 (Silicon Graphics) | IRIX | SGI |
| IBM SP | AIX | IBM |
| Compaq True 64 Alpha (DEC ALFA) | OSF1 | Compaq |
| Sun SPARC | Solaris | Sun |
| PA-RISC (HP 9000 series 700/800) | HP-UX v11 | HP |
| Intel Pentium (32-bit) PC | Linux | GNU (g95) |
| Intel Pentium (32-bit) PC | Linux | GNU (gfortran) |
| Intel Pentium (32-bit) PC | Linux | Intel |
| Intel Itanium (64-bit) PC | Linux | Intel |
| Intel Pentium (32-bit) PC | Linux | Portland Group |
| Intel Pentium (32-bit) PC | Linux | Lahey |
| Intel Pentium (32-bit) PC | MS Windows | Compaq Visual |
| Power Mac G4 | Mac OS X | IBM |

If your computer and available compiler is mentioned in the table, you may consult Section 3.1 for a quick installation of SWAN. Otherwise, read Section 3.2 for a detailed description of the manual installation of SWAN.

Note that for a successful installation, a Perl package must be available on your computer. In most cases, it is available for Linux and a UNIX operating system. Check it by typing `perl -v`. Otherwise, you can download a free distribution for Windows called ActivePerl; see *http://aspn.activestate.com/ASPN/Downloads/Ac*
The Perl version should be at least 5.0.0 or higher!

Before installation, the user may first decide how to run the SWAN program. There are three possibilities:

- serial runs,
- parallel runs on shared memory systems or
- parallel runs on distributed memory machines.

For stationary and small-scale computations, it may be sufficient to choose the serial mode, i.e. one SWAN program running on one processor. However, for relatively large CPU-demanding calculations (e.g., instationary or nesting ones), two ways of parallelism for reducing the turn-around time are available:

- The SWAN code contains a number of so-called OpenMP directives that tells the compiler how to generate multi-threaded code on a shared

memory computer. For this, you need a Fortran 90 compiler having the OpenMP option. The performance is good for a restricted number of threads ($< 8$).This type of parallelism can be used e.g., on symmetric multiprocessors and Linux PC's with dual processors.

- If the user want to run SWAN on a relative large number of processors, a message passing model is a good alternative. It is based on independent processors which do not share any memory but are connected via an interconnection network, such as Beowulf systems (cluster of Linux PC's connected via fast Ethernet switches). Since, the Message Passing Interface (MPI) standard (e.g., MPICH distribution, freely available for several platforms, such as Linux and Windows NT/2000/XP, at *http://www-unix.mcs.anl.gov/mpi/mpich*http://www-unix.mcs.anl.gov/mpi/mpich) is very popular nowadays, the SWAN code contains a set of generic subroutines that call a number of MPI-routines, meant for local data exchange, gathering data, global reductions, etc. This technique is beneficial for larger simulations only, such that the communication times are relatively small compared to the computing times.

## 3.1 Automatic and quick installation

Carry out the following steps for setting up SWAN on your computer.

1. An include file containing some machine-dependent macros must be created first. This file is called `macros.inc` and can be created by typing

```
make config
```

2. Now, SWAN can be built for serial or parallel mode, as follows:

| **mode** | **instruction** |
|---|---|
| serial | `make ser` |
| parallel, shared | `make omp` |
| parallel, distributed | `make mpi` |

IMPORTANT NOTES:

- To Windows users:
  - To execute the above instructions, just open a command prompt.
  - In case of Compaq Visual Fortran compiler, use `nmake` instead of `make`.
  - This setup does not support OpenMP for Windows systems.
  - This installation currently supports MPICH for Windows NT/2000/XP (Professional); Win9x/ME are not supported.
  - It is assumed that both the directories `include` and `lib` are resided in `C:\PROGRAM FILES\MPICH\SDK`. If not, the file `macros.inc` should be adapted such that they can be found by the Makefile.
- One of the commands `make ser`, `make omp` and `make mpi` must be preceded by `make config`.
- If desirable, you may clean-up the generated object files and modules by typing `make clean`. If you want to go back to the original state with respect to the source code, i.e. removing everything that has been generated by the Makefile, just type `make allclean`.
- If you are unable to install SWAN using the Makefile and Perl scripts for whatever reason, see Section 3.2 for instructions on manual installation.

## 3.2 Manual installation

### 3.2.1 Modifications in the source code

To compile SWAN on your computer system properly, some subroutines should be adapted first depending on the operating system, use of compilers and the wish to use MPI or OpenMP for parallel runs. This can be done by removing the switches started with '!' followed by an indentifiable prefix in the first 3 or 4 columns of the subroutine. A Perl script called `switch.pl` is provided in the material that enables the user to quickly select the switches to be removed. This script can be used as follows:

```
perl switch.pl [-dos] [-unix] [-f95] [-mpi] [-omp] [-cray]
               [-sgi] [-cvis] [-timg] [-impi] *.ftn
```

where the options are all optionally. The meaning of these options are as follows.

`-dos`, `-unix` Depending on the operating system, both the TAB and directory separator character must have a proper value (see also Chapter 4). This can be done by removing the switch !DOS or !UNIX, for Windows and UNIX/Linux platforms, respectively, in the subroutines `OCPINI` (in `ocpids.ftn`) and `TXPBLA` (in `swanser.ftn`). For other operating system (e.g., Macintosh), you should change the values of the following variables manually: DIRCH1, DIRCH2 (in `OCPINI`), TABC (in `OCPINI`) and ITABVL (in `TXPBLA`). Finally, the MPICH distribution for Windows NT/2000/XP does not support `USE MPI` statement and therefore, the module `MPI` in `swmod1.ftn` must be included by removing the switch !DOS.

`-f95` If you have a Fortran 95 compiler or a Fortran 90 compiler that supports Fortran 95 features, it might be useful to activate the `CPU_TIME` statement in the subroutines `SWTSTA` and `SWTSTO` (in `swanser.ftn`) by removing the switch !F95 meant for the detailed timings of several parts of the SWAN calculation. Note that this can be obtained with the command TEST by setting **itest=1** in your command file.

`-mpi` For the proper use of MPI, you must remove the switch !MPI at several places in the file `swanparll.ftn`, `swancom1.ftn` and `swmod1.ftn`.

`-omp` The subroutine `SWCOMP` (in `swancom1.ftn`) contains a number of `_OPENMP` macro that needs to be set by first preprocessing it by the C (or Fortran) preprocessor. In order to use this macro just remove the !OMP switch.

`-cray`, `-sgi` If you use a Cray or SGI Fortran 90 compiler, the subroutines `OCPINI` (in `ocpids.ftn`) and `FOR` (in `ocpmix.ftn`) should be adapted by removing the switch !/Cray or !/SGI since, these compilers cannot read/write lines longer than 256 characters by default. By means of the option RECL in the OPEN statement sufficiently long lines can be read/write by these compilers.

`-cvis` The same subroutines `OCPINI` and `FOR` need also to be adapted when the Compaq Visual Fortran compiler is used in case of a parallel MPI run. Windows systems have a well-known problem of the inability of opening a file by multiple SWAN executables. This can be remedied by using the option SHARED in the OPEN statement for shared access. For this, just remove the switch !CVIS.

`-timg` If the user want to print the timings (both wall-clock and CPU times in seconds) of different processes within SWAN then remove the switch !TIMG. Otherwise, no timings will be keeped up and subsequently printed in the `PRINT` file.

`-impi` Some Fortran compilers do not support the use of module MPI (`USE MPI`). In that case, remove the switch !/impi.

For example, you work on a Beowulf cluster where MPI has been installed and use the Intel Fortran compiler (that can handle Fortran 95 statements), then type the following:

```
perl switch.pl -unix -f95 -mpi *.ftn
```

Note that due to the option `-unix` the extension `ftn` is automatically changed into `f`.

### 3.2.2 Compiling and linking SWAN source code

After the necessary modifications are made as described in the previous section, the source code is ready for compilation. All source code is written in fixed form Fortran 90 so you must have a Fortran 90 compiler in order to compile SWAN. The source code cannot be compiled with a Fortran 77 compiler. If you intended to use MPI for parallel runs, you must use the command `mpif90` instead of the original compiler command or using the Integrated Development Environment e.g., for Visual Fortran (see Installation and User's Guide for MPICH). For parallel runs using OpenMP, the compiler must have an option to interpret OpenMP directives and the extension `ftn` must be changed into `F` (will be done automatically when using the script `switch.pl`).

The SWAN source code complies with the ANSI Fortran 90 standard, except for a few cases, where the limit of 19 continuation lines is violated. We are

currently not aware of any compiler that cannot deal with this violation of the ANSI standard.

When compiling SWAN you should check that the compiler allocates the same amount of memory for all INTEGERS, REAL and LOGICALS. Usually, for these variables 4 bytes are allocated, on supercomputers (vector or parallel), however, this sometimes is 8 bytes. When a compiler allocates 8 bytes for a REAL and 4 bytes for an INTEGER, for example, SWAN will not run correctly.

Furthermore, SWAN can generate binary MATLAB files on request, which are unformatted. Some compilers, e.g. Compaq Visual Fortran and Intel Fortran version 9.x, measured record length in 4-byte units and as a consequence, these unformatted files cannot be loaded in MATLAB. Hence, in such as case a compiler option is needed to request 1-byte units, e.g. for Compaq Visual Fortran this is `/assume:byterecl` and for Intel Fortran version 9.x this is `-assume byterecl`.

The modules (in files `swmod1.ftn`, `swmod2.ftn` and `swmod3.ftn`) must be compiled first. Several subroutines use the modules. These subroutines need the compiled versions of `swmod1.ftn`, `swmod2.ftn` and `swmod3.ftn`, before they can be compiled. Linking should be done without any options nor using shared libraries (e.g. math or NAG). It is recommended to rename the executable to `swan.exe` after linking.

Referring to the previous example, compilation and linking may be done as follows:

```
mpif90 swmod1.f swmod2.f swmod3.f ocp*.f swan*.f -o swan.exe
```

## 3.3 Make SWAN documentation

SWAN comes with 4 detailed documents which are provided as downloadable PDF files as well as browsable web-pages:

- The User Manual describes the complete input and usage of the SWAN package.
- The Implementation Manual explains the installation procedure of SWAN on a single- or multi-processor machine with shared or distributed memory.
- The Programming rules is meant for programmers who want to develop SWAN.
- The Technical documentation discusses the mathematical details and the discretizations that are used in the SWAN program.

These documents are written in LaTeX format. If you are new to LaTeX, we recommend to read first the manual LaTeX for dummies that is available in the SWAN material.

The PDF files are very easy to generate by just typing

```
make doc
```

# Chapter 4

# User dependent changes and the file `swaninit`

SWAN allows you to customize the input and the output to the wishes of your department, company or institute. This can be done by changing the settings in the initialisation file `swaninit`, which is created during the first time SWAN is executed on your computer system. The changes in `swaninit` only affect the runs executed in the directory that contains that file.

A typical initialisation file `swaninit` may look like:

```
    4                               version of initialisation file
Delft University of Technology      name of institute
    3                               command file ref. number
INPUT                               command file name
    4                               print file ref. number
PRINT                               print file name
    4                               test file ref. number
                                    test file name
    6                               screen ref. number
   99                               highest file ref. number
$                                   comment identifier
[TAB]                               TAB character
\                                   dir sep char in input file
/                                   dir sep char replacing previous one
    1                               default time coding option
  100                               speed of processor   1
```

```
100                                    speed of processor   2
100                                    speed of processor   3
100                                    speed of processor   4
```

Explanation:

- The version number of the initialisation file is included in the file so that SWAN can verify whether the file it reads is a valid initialisation file. The current version is 4.

- The initialisation file provides a character string containing the name of the institute that may carry out the computations or modifying the source code. You may assign it to the name of your institute instead of 'DELFT UNIVERSITY OF TECHNOLOGY', which is the present value.

- The standard input file and standard print file are usually named `INPUT` and `PRINT`, respectively. You may rename these files, if appropriate.

- The unit reference numbers for the input and print files are set to 3 and 4, respectively. If necessary, you can change these numbers into the standard input and output unit numbers for your installation. Another unit reference number is foreseen for output to screen and it set to 6. There is also a unit number for a separate test print file. In the version that you downloaded from our web page, this is equal to that of the print file so that test output will appear on the same file as the standard print output.

- The comment identifier to be used in the command file is usually '$', but on some computer system this may be inappropriate because a line beginning with '$' is interpreted as a command for the corresponding operating system (e.g., VAX systems). If necessary, change to '!'.

- To insert [TAB] in the initialisation file, just use the TAB key on your keyboard.

- Depending on the operating system, the first directory separation character in `swaninit`, as used in the input file, may be replaced by the second one, if appropriate.

- Date and time can be read and written according to various options. The following options are available:

    1. 19870530.153000 (ISO-notation)

2. 30-May-87 15:30:00
3. 05/30/87 15:30:00
4. 15:30:00
5. 87/05/30 15:30:00
6. 8705301530 (WAM-equivalence)

Note that the ISO-notation has no millenium problem, therefore the ISO-notation is recommended. In case of other options, the range of valid dates is in between January 1, 1911 and December 31, 2010 (both inclusive).

- In case of a parallel MPI run at the machine having a number of independent processors, it is important to assign subdomains representing appropriate amounts of work to each processor. Usually, this refers to an equal number of grid points per subdomain. However, if the computer has processors which are not all equally fast (a so-called heterogeneous machine), then the sizes of the subdomains depend on the speed of the processors. Faster processors should deal with more grid points than slower ones. Therefore, if necessary, a list of non-default processor speeds is provided. The given speeds are in % of default = 100%. As an illustrating example, we have two PC's connected via an Ethernet switch of which the first one is 1.5 times faster than the second one. The list would be

```
150    speed of processor 1
100    speed of processor 2
```

Based on this list, SWAN will automatically distribute the total number of active grid points over two subdomains in an appropriate manner. Referring to the above example, with 1000 active points, the first and second subdomains will contain 600 and 400 grid points, respectively.

# Chapter 5

# Usage of SWAN executable

To help you in editing an command file for SWAN input, the file `swan.edt` is provided.

Two run procedures are provided among the source code, one for the Windows platform, called `swanrun.bat`, and one for the UNIX/Linux platform, called `swanrun`. Basically, the following actions need to be done by the run procedure:

- Copy the command file with extension `swn` to `INPUT` (assuming `INPUT` is the standard file name for command input, see Chapter 4).
- Run SWAN.
- Copy the file `PRINT` (assuming `PRINT` is the standard file name for print output, see Chapter 4) to a file which name equals the command file with extension `prt`.

On other operating system a similar procedure can be followed. For parallel MPI runs, the program `mpirun` is needed and is provided in the MPICH distribution.

Before calling the run procedure, the environment variable PATH need to be adapted by including the pathname of the directory where `swan.exe` can be found. In case of Windows, this pathname can be specified through the category *System* of *Control Panel* (on the *Advanced* tab, click *Environment Variables*) or by adding it in the `AUTOEXEC.BAT` file. In case of UNIX or Linux running the Bourne shell, the environment variable PATH may be changed as follows:

```
export PATH=${PATH}:/usr/local/swan
```

if `/usr/local/swan` is the directory where the executable `swan.exe` can be found. In case of the C shell, use the following command:

```
setenv PATH ${PATH}:/usr/local/swan
```

If appropriate, you also need to add the directory path where the `bin` directory of MPICH is resided to PATH to have access to the command `mpirun`.

The provided run procedures enable the user to properly and easily run SWAN both serial as well as parallel (MPI or OpenMP). Note that for parallel MPI runs, the executable `swan.exe` should be accessible by copying it to all the multiple machines or by placing it in a shared directory. When running the SWAN program, the user must specify the name of the command file. However, it is assumed that the extension of this file is `swn`. Note that contrary to UNIX/Linux, Windows does not distinguish between lowercase and uppercase characters in filenames. Next, the user may also indicate whether the run is serial or parallel. In case of Windows, use the run procedure `swanrun.bat` from a command prompt:

```
swanrun filename [nprocs]
```

where `filename` is the name of your command file without extension (assuming it is `swn`) and `nprocs` indicates how many processes need to be launched for a parallel MPI run (do not type the brackets; they just indicate that `nprocs` is optional). By default, $nprocs = 1$.

The command line for the UNIX script `swanrun` is as follows:

```
./swanrun -input filename [-omp n | -mpi n]
```

where `filename` is the name of your command file without extension. The parameter `-omp n` specifies a parallel run on $n$ processors using OpenMP. The parameter `-mpi n` specifies a parallel run on $n$ processors using MPI. The parameter `-input` is obliged, whereas the parameters `-omp n` and `-mpi n` can be omitted (default: $n = 1$). Note that the script `swanrun` need to be made executable first, as follows:

```
chmod +rx ./swanrun
```

For a parallel MPI run, you may also need a `machinefile` that contains the names of the nodes in your parallel environment. Put one node per line in the file. Lines starting with the # character are comment lines. You can specify a number after the node name to indicate how many processes to launch on the node. This is useful e.g., for dual-processors. The run procedure will cycle through this list until all the requested processes are launched. Example of such a file may look like:

```
# here, eight processes will be launched
node1
node2:2
node4
node7:4
```

Note that for Windows platforms, a space should be used instead of a colon as the separation character in the `machinefile`.

SWAN will generate a number of output files:

- A print file with the name `PRINT` that can be renamed by the user with a batch (DOS) or script (UNIX) file, e.g. with the provided run procedures. For parallel MPI runs, however, a sequence of `PRINT` files will be generated (`PRINT-001`, `PRINT-002`, etc.) depending on the number of processors. The print file(s) contain(s) the echo of the input, information concerning the iteration process, possible errors, timings, etc.

- Numerical output (such as table, spectra and block output) appearing in files with user provided names.

- A file called `Errfile` (or renamed by the run procedures as well as more than one file in case of parallel MPI runs) containing the error messages is created only when SWAN produces error messages. Existence of this file is an indication to study the results with more care.

- A file called `ERRPTS` (or renamed by the run procedures as well as more than one file in case of parallel MPI runs) containing the grid-points, where specific errors occured during the calculation, such as non-convergence of an iterative matrix-solver. Existence of this file is an indication to study the spectrum in that grid-point with more care.

# Chapter 6

# Testing the system

The SWAN system consists of one executable file (`swan.exe`), a command file (`swan.edt`) and a run procedure (`swanrun.bat` or `swanrun`). The input and output to a number of test problems is provided on the SWAN web page. The files with extension `swn` are the command files for these tests; the files with extension `bot` are the bottom files for these tests, etc. This input can be used to make a configuration test of SWAN on your computer. Compare the results with those in the provided output files. Note that the results need not to be identical up to the last digit.

To run the SWAN program for the test cases, at least 50 MBytes of free internal memory is recommended. For more realistic cases 100 to 500 MBytes may be needed, whereas for more simple stationary or 1D cases significant less memory is needed (less than 5 MBytes for 1D cases).