SWAN

IMPLEMENTATION MANUAL

SWAN Cycle III version 41.51

SWAN IMPLEMENTATION MANUAL

by	:	The SWAN team
mail address	:	Delft University of Technology Faculty of Civil Engineering and Geosciences Environmental Fluid Mechanics Section P.O. Box 5048 2600 GA Delft The Netherlands
e-mail	:	m.zijlema@tudelft.nl
homepage	:	http://www.swan.tudelft.nl

Copyright (c) 1993-2024 Delft University of Technology.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is available at http://www.gnu.org/licenses/fdl.html#TOC1.

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 pre-built SWAN release for Windows,
- the User Manual,
- this Implementation Manual,
- the Scientific/technical documentation,
- the SWAN programming rules,
- utilities and
- some test cases.

All of the material can be found at the official SWAN homepage.

Since version 41.41, the SWAN source code is also hosted on GitLab and can be cloned from this repository. For details see Section 3.3.

On the SWAN homepage, general information is given about the model functionalities, physics and limitations of SWAN. Also, the modification history (or release notes) of SWAN and information on support are provided.

After downloading the material, you may choose between

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

If you want to use the pre-built SWAN 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. However, for any use of the SWAN source code in your environment, proper reference must be made to the origin of the software.

Implementation involves the following steps:

1. Copying the source code from the SWAN homepage 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, resolved issues, 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 built SWAN.

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

1.1 The material

The source tarball `swan4151.tar.gz` contains the SWAN source code and consists of the following files:

main program	: swanmain.ftn
pre-processing routines	: swanpre1.ftn
	swanpre2.ftn
	SwanBndStruc.ftn90
computational routines	: swancom1.ftn
	swancom2.ftn
	swancom3.ftn
	swancom4.ftn
	swancom5.ftn
post-processing routines	: swanout1.ftn
	swanout2.ftn
	SwanVTKWriteHeader.ftn90
	SwanVTKWriteData.ftn90
	SwanVTKPDataSets.ftn90

service routines	: swanser.ftn
	SwanIntgratSpc.ftn90
routines for Bragg scattering	: SwanBraggScat.ftn90
routines for QCM	: SwanQCM.ftn90
routines for ST6 package	: SdsBabanin.ftn90
routines for support parallel MPI runs	: swanparll.ftn
routines for unstructured grids	: SwanReadGrid.ftn90
	SwanReadADCGrid.ftn90
	SwanReadTriangleGrid.ftn90
	SwanReadEasymeshGrid.ftn90
	SwanInitCompGrid.ftn90
	SwanCheckGrid.ftn90
	SwanCreateEdges.ftn90
	SwanGridTopology.ftn90
	SwanGridVert.ftn90
	SwanGridCell.ftn90
	SwanGridFace.ftn90
	SwanPrintGridInfo.ftn90
	SwanFindPoint.ftn90
	SwanPointinMesh.ftn90
	SwanBpntlist.ftn90
	SwanPrepComp.ftn90
	SwanVertlist.ftn90
	SwanCompUnstruc.ftn90
	SwanDispParm.ftn90
	SwanPropvelX.ftn90
	SwanSweepSel.ftn90
	SwanPropvelS.ftn90
	SwanTranspAc.ftn90
	SwanTranspX.ftn90
	SwanGradDepthorK.ftn90
	SwanGradVel.ftn90
	SwanDiffPar.ftn90
	SwanGSECorr.ftn90
	SwanInterpolatePoint.ftn90
	SwanInterpolateAc.ftn90
	SwanInterpolateOutput.ftn90
	SwanConvAccur.ftn90
	SwanConvStopc.ftn90
	SwanThreadBounds.ftn90

	SwanFindObstacles.ftn90
	SwanCrossObstacle.ftn90
	SwanComputeForce.ftn90
routines for parallel, unstructured grids	: SwanParallel.ftn90
modules and subroutines for netCDF	: nctablemd.ftn90
	agioncmd.ftn90
	swn_outnc.ftn90
couplers	: couple2adcirc.ftn90
	swan2coh.ftn90
routines for installation	: ocpids.ftn
command reading routines	: ocpcre.ftn
miscellaneous routines	: ocpmix.ftn
general modules	: swmod1.ftn
	swmod2.ftn
modules for XNL	: m_constants.ftn90
	m_fileio.ftn90
	serv_xnl4v5.ftn90
	mod_xnl4v5.ftn90
modules for unstructured grids	: SwanGriddata.ftn90
	SwanGridobjects.ftn90
	SwanCompdata.ftn90
modules for spectral partitioning	: SwanSpectPart.ftn
routines for the fast Fourier transform	: fftpack51.ftn90

The SWAN source code is written in Fortran 90. Some routines are written in fixed form and depending on your system, the extension may be `for` or `f`. Other routines are written in free form and are indicated by extension `f90`. The conversion from `ftn` or `ftn90` 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 want your modifications to be implemented in the authorized version of SWAN (the version on the SWAN homepage), you need to submit these changes to the SWAN team (e-mail: m.zijlema@tudelft.nl).

As a part of the package, `fftpack51` is the Fast Fourier Transform library translated to Fortran 90 by John Burkardt of Florida State University.

The SWAN source code is additionally accompanied by the following files:

installation procedures	: INSTALL.README
	Makefile
	macros.inc
	which.cmd
	platform.pl
	switch.pl
run procedures	: SWANRUN.README
	swanrun
	swanrun.bat
machinefile for parallel MPI runs	: machinefile
for concatenation of multiple hotfiles	: swanhcat.ftn
	hcat.nml
edit file	: swan.edt
Matlab scripts for unstructured grids	: plotunswan.m
	plotgrid.m

On the SWAN homepage, 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. See Chapter 6 for further details.

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 homepage. These are provided by patch files that can be downloaded from the website. Typically, a patch can be installed on 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. 41.51.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 41.51ABC, 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 Linux/UNIX 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 41.51.[A-C] | tr -d '\r' | patch` that apply the patch files 41.51.A to 41.51.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 website, which is appropriate for Windows operating system.

Chapter 3

Installation

3.1 Introduction

SWAN can be installed on various architectures, including laptops and supercomputers. The portability of the SWAN source code is guaranteed by the use of standard ANSI Fortran 90. (See also the manual Programming rules.) Hence, virtually all Fortran compilers can be used for installing SWAN. It should be noted that there are two Fortran commands used in the source code of SWAN (v41.41+) which were introduced in later versions of Fortran: stream I/O (Fortran 2003 standard) and execution OS command line (Fortran 2008 standard). They are, however, supported by currently maintained Fortran compilers, including gfortran and Intel® Fortran.

The SWAN source code also supports parallelization, which enables a considerable reduction in the wall-clock time for relatively large CPU-demanding calculations. Two parallelization strategies are available:

- The computational kernel of SWAN contains a number of OpenMP compiler directives, so that users can optionally run SWAN on a cluster or laptop with multi shared-memory processors.

- A message passing modelling is employed based on the Message Passing Interface (MPI) standard that enables communication between independent processors. Hence, users can optionally run SWAN on a Linux cluster containing memory-distributed nodes.

The SWAN software can be build in the usual way via GNU make or from scratch. This building process is explained in Section 3.2. However, since version 41.41, the option to install SWAN using CMake is supported and is elaborated in Section 3.3.

3.2 Classic build instructions

3.2.1 Configuring the build

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

platform	**OS**	**F90 compiler**
Intel/AMD desktop/laptop	Linux	gfortran
Intel Core desktop/laptop	Linux	Intel®
Intel Xeon desktop/laptop	Linux	Intel®
x86-64 processor-based system	Linux	Portland Group
Intel/AMD desktop/laptop	Linux	Lahey
Intel Core desktop/laptop	MS Windows	Intel®
Intel Xeon desktop/laptop	MS Windows	Intel®
Intel/AMD desktop/laptop	MS Windows	Lahey
MacBook	macOS	gfortran
MacBook	macOS	Intel®
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
IBM Power6 (pSeries 575)	Linux	IBM
Power Mac G4	Mac OS X	IBM

If your computer and available compiler is mentioned in the table, you may consult Section 3.2.2 for a complete build of the software without making any modifications. If desired, you may install SWAN manually; see Section 3.2.3.

Note that for a successful installation, a Perl package must be available on your computer. Usually, it is available for macOS, Linux and a UNIX-like operating system. Check it by typing `perl -v`. You can download Perl for MS Windows from Strawberry Perl. The Perl version should be at least 5.0.0 or higher!

Before starting the build process, the user may first decide how to run the SWAN program. There are three run modes:

- 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 enables the compiler to generate multi-threaded code on a shared-memory computer. For this, you need a Fortran 90 compiler supporting OpenMP. The performance is good for a limited number of threads. This type of parallelism can be used e.g., on laptops containing multi-core 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 (e.g. cluster of Linux PC's connected via fast Ethernet switches). The implementation is based on the Message Passing Interface (MPI) standard (e.g., MPICH and OpenMPI). 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.
- For a proper installation of MPI-based application on Windows, please consult Section 3.2.4.

3.2.2 Building SWAN using GNU make

Carry out the following steps for building 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.
  - To build SWAN on Windows platforms by means of a Makefile you need the command-line utility `Nmake`, which is provided by the Microsoft® Visual Studio.
  - This setup does support OpenMP for Windows systems, if Intel® Fortran Compiler is provided.
  - This installation currently supports Intel® MPI library for Windows. See Section 3.2.4 for further information.
- One of the commands `make ser`, `make omp` and `make mpi` must be preceded once by `make config`.
- If desirable, you may clean up the generated object files and modules by typing `make clean`. If you want to delete any stray files from a previous compilation, just type `make clobber`.
- If you are unable to install SWAN using the Makefile and Perl scripts for whatever reason, see Section 3.2.3 that includes instructions for a custom installation.

3.2.3 Building SWAN from scratch

It is recommended to consult Section 3.2.2 for a complete build of SWAN on your computer. However, if you want to build SWAN on your system from scratch, then please follow the instructions below.

Modifying 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 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] [-jac] [-mpi] [-metis] [-cray] [-sgi]
               [-cvis] [-timg] [-matl4] [-impi] [-adcirc] [-netcdf] *.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 Linux/UNIX 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`).

`-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.

`-jac` In case of parallel runs on distributed memory systems an efficient algorithm is required to parallelize the implicit propagation operator. The simplest strategy consists in treating the data on subdomain interfaces explicitly, which in mathematical terms amounts to using a block Jacobi approximation of the implicit operator. To minimize the communication volume, a recursive application of alternately horizontal and vertical stripwise partitioning is carried out. This strategy possess a high degree of parallelism, but may lead to a certain degradation of convergence properties. Another strategy is the block wavefront approach. This approach does not alter the order of computing operations of the sequential algorithm and thus preserving the convergence properties, but reduces parallel efficiency to a lesser extent because of the serial start-up and shut-down phases. The user is advised to choose the block Jacobi approach in case of non- or quasi-stationary SWAN simulations, otherwise the block wavefront method is preferable.

By default, the block wavefront approach will be applied and so the switch !WFR will be removed automatically. However, if the user want to apply the block Jacobi method then the switch !JAC must be removed while the switch !WFR should not be removed. This can be realized with the option `-jac`. Note that this option must be followed by the next option `-mpi`.

`-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`.

`-metis` To enable to partition the unstructured mesh, the switch !METIS must be removed at several places in different files.

`-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.

`-matl4` By default, the created binary Matlab files are of Level 5 MAT-File format and are thus compatible with MATLAB version 5 and up. In this case the switch !MatL5 must be removed. However, some machines do not support a 1-byte unit for the record length (e.g. IBM Power6). At those computers, the binary Matlab files must be formatted of Level 4. In this case the switch !MatL4 must be removed while the switch !MatL5 should not be removed. Level 4 MAT-files are compatible with MATLAB versions 4 and earlier. However, they can be read with the later versions of MATLAB.

`-impi` Some Fortran compilers do not support `USE MPI` statement and therefore, the module `MPI` in `swmod1.ftn` must be included by removing the switch !/impi.

`-adcirc` To enable to do coupled ADCIRC+SWAN simulation, the switch !ADC must be removed at several places in different files. However, for a standalone SWAN simulation, the switch !NADC must be removed while the switch !ADC should not be removed.

`-netcdf` For the proper use of netCDF, you must remove the switch !NCF at several places in different files.

For example, you work on a Linux 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 *.ftn90
```

Note that due to the option `-unix` the extension `ftn` is automatically changed into `f` and `ftn90` into `f90`. Also note that the block wavefront algorithm is chosen for parallel runs.

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 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` (or `mpiifort` in case of the Intel® compiler) instead of the original compiler command.

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. Intel® Fortran, measured record length in 4-byte or longword 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 Intel® Fortran this is `/assume:byterecl` (Windows) or `-assume byterecl` (Linux/UNIX).

The modules must be compiled first. Several subroutines use these modules. These subroutines need the compiled versions of the modules before they can be compiled. You can find here below the complete list of modules in the proper order.

- `swmod1.f`, `swmod2.f`
- `SwanSpectPart.f90`
- `m_constants.f90`, `m_fileio.f90`, `serv_xnl4v5.f90`, `mod_xnl4v5.f90`
- `SwanGriddata.f90`, `SwanGridobjects.f90`, `SwanCompdata.f90`
- `SwanParallel.f90`
- `SdsBabanin.f90`
- `SwanIEM.f90`
- `SwanBraggScat.f90`
- `SwanQCM.f90`

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 <list of modules> ocp*.f swan*.f Swan*.f90 -o swan.exe
```

3.2.4 Building with MPI support

SWAN can be built with support for MPI. It is assumed that MPI has been installed already in the Linux environment. However, this is probably not the case for Windows. At any rate, the Intel® MPI library may be employed to build an MPI application. This library is included in the Intel® oneAPI HPC Toolkit. In this respect, the following steps need to be made first

- make sure that the variables `INCS_MPI` and `LIBS_MPI` in the file `macros.inc` are emptied, and
- change the value of the variable `F90_MPI` by replacing `ifort` by `mpiifort`.

Build SWAN by executing the command `make mpi`.

3.2.5 Building with Metis support

SWAN 41.45A+ can be compiled with support for Metis to partition an unstructured mesh so that simulations can be carried out on distributed-memory machines[1]. For this, an MPI implementation is still required; see Section 3.2.4. The actual mesh partitioning implemented in SWAN is the multilevel k-way method. For details see the Metis manual.

For a proper building, the Metis software package must be installed first on your machine. Installation details can be found at https://github.com/KarypisLab/METIS. Make sure that both C++ constants `IDXTYPEWIDTH` and `REALTYPEWIDTH` in header file `metis.h`) have been set to 32. Note that you also need to install the GKlib library. It is recommended to install the lastest version of Metis; high-resolution simulations with the unstructured mesh version of SWAN have been tested with Metis 5.2.1.

Carry out the following steps.

1. Define the variable `METISROOT` in file `macros.inc` to refer to the Metis root directory (e.g. `METISROOT=/opt/metis` in case of Linux/macOS or `METISROOT=BINARY_DIR` in case of Windows).
2. Build SWAN as usual (e.g. `make mpi`); see Section 3.2.2.

3.2.6 Building with netCDF support

SWAN 40.91A+ contains extensions that provide netCDF output of spectra and maps of several wave parameters. Carry out the following steps.

1. Make sure that netCDF 4.5.x or greater is compiled and available on your system. For details, consult netCDF downloads. The Fortran interface must have been enabled when netCDF was built.

[1] Building with parallel libraries from the ADCIRC suite is no longer necessary.

2. Define the variable `NETCDFROOT` in file `macros.inc` to point to the netCDF root directory (e.g. `NETCDFROOT=/usr/local/netcdf` in case of Linux or `NETCDFROOT=C:\PROGRAM FILES\netcdf` in case of Windows). This enables to compile netCDF I/O into SWAN.

3. Build SWAN as usual (e.g. `make mpi`); see Section 3.2.2.

3.3 Building SWAN with CMake

CMake is a cross-platform build system that creates native build files (for use with a generator GNU Make, Nmake or Ninja) for command line builds or project files for an IDE (e.g. Visual Studio). CMake makes use of configuration files that control the build process. We recommend to use CMake 3.12+ for building SWAN. There are installers available for Windows, Linux and macOS. See the download page for installation instructions.

Ninja is one of the many build generators to create executable files and libraries from source code. The way it works is very similar to GNU make; for example, it does not rebuild things that are already up to date. We recommend Ninja because it is faster than GNU make. Ninja can be downloaded from its git repository.

3.3.1 Build instructions

For a proper build, the release code including the required CMake files can be downloaded or cloned from the SWAN git repository hosted on TU Delft GitLab.

Next, carry out the following steps.

1. clone the repository and navigate to the top level source directory

```
git clone https://gitlab.tudelft.nl/citg/wavemodels/swan.git && cd swan
```

2. create the build directory

At the top of SWAN source directory execute the following commands

```
mkdir build
cd build
```

This step is required to perform an out-of-source build with CMake, that is, build files will not be created in the `/swan/src` directory.

3. build the software

Two CMake configuration files are provided as required for the build. They are placed in the following source directories: `./swan/CMakeLists.txt` and `./swan/src/CMakeLists.txt`. The following two CMake commands should suffice to build SWAN

```
cmake .. -G Ninja
cmake --build .
```

The first command refers to the source directory where the main configuration file is invoked. The second command carries out the building in the build directory. The package is actually built by invoking Ninja.

4. install the package

```
cmake --install .
```

The default install directory is `/usr/local/swan` (Unix-like operating systems, including macOS) or `C:\PROGRAM FILES\swan` (Windows). Instead, you may install SWAN in any other user-defined directory, as follows

```
cmake --install . --prefix /your/defined/directory
```

After installation a number of subdirectories are created. The executables end up in the `/bin` directory, the library files in `/lib`, and the module files in `/mod`. Additionally, the `/doc` folder contains the pdf documents, the folder `/tools` consists of some useful scripts and the `/misc` directory contains all of the files that do not fit in other folders (e.g., a machinefile and the edit file `swan.edt`).

Please note that the installation can be skipped (though not recommended). Executables and libraries are then located in subdirectories of the build directory.

3.3.2 Configuring the build

The build can be (re)configured by passing one or more options to the CMake command with prefix `-D`. A typical command line looks like

```
cmake .. -D<option>=<value>
```

where `<value>` is a string or a boolean, depending on the specified option. The table below provides an overview of the non-required options that can be used.

option	description	default value
CMAKE_INSTALL_PREFIX	user-defined installation path	/usr/local/swan
CMAKE_PREFIX_PATH	semicolon-separated list of library paths	empty
CMAKE_Fortran_COMPILER	full path to the Fortran compiler	determined by CMake
MPI	enable build with MPI	OFF
OPENMP	enable build with OpenMP	OFF
METIS	enable build with Metis	OFF
NETCDF	enable build with netCDF	OFF
CMAKE_VERBOSE_MAKEFILE	provide verbose output of the build	OFF

For example, the following commands

```
cmake .. -GNinja -DNETCDF=ON -DMPI=ON
cmake --build .
```

will configure SWAN to be built created by Ninja that supports netCDF output and parallel computing using the MPI paradigm. Note that CMake will check the availability of MPI and netCDF libraries within your environment. Also note that netCDF libraries might be installed in a custom directory (e.g., `/home/your/name/netcdf`), which must then be a priori specified on the command line as follows:

```
export NetCDF_ROOT=/path/to/netcdf/root/directory
```

or

```
cmake .. -DCMAKE_PREFIX_PATH=/path/to/netcdf/directory
```

so that CMake can find the libraries. The same holds for Metis libraries, as follows:

```
export Metis_ROOT=/path/to/metis/root/directory
```

or

```
cmake .. -DCMAKE_PREFIX_PATH=/path/to/metis/directory
```

Note: to define a path list with more than one prefixes use a semicolon as a separator.

The system default Fortran compiler (e.g., f77, g95) can be overwritten as follows

```
cmake .. -DCMAKE_Fortran_COMPILER=/path/to/desired/compiler
```

Finally, if CMake fails to configure your project, then execute

```
cmake .. -DCMAKE_VERBOSE_MAKEFILE=ON
```

which will generate detailed information that may provide some indications to debug the build process.

3.3.3 Clean up the build files

To remove the build directory and all files that have been created after running `cmake --build .`, run at the top level of your project the following command:

```
cmake -P clobber.cmake
```

(The `-P` argument passed to CMake will execute a script `<filename>.cmake`.)

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 or yourself. 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
99999                               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. This is useful if print output is lost due to abnormal end of the program, while information about the reason is expected to be in the print file. There is also a unit number for a separate test print file. In the version that you downloaded from SWAN homepage, 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

Run instructions

In this chapter it is assumed that you have a built SWAN available on your computer, either after installation as described in Chapter 3, or downloaded from the SWAN website (for Windows).

IMPORTANT NOTE:

> The pre-built SWAN for Windows, available at the SWAN website, has been compiled using the Intel® Fortran Compiler Classic (as part of Intel® oneAPI HPC Toolkit) and is linked to DLL libraries. Therefore you may run into problems when attempting to run SWAN on your Windows computer, unless you have the Microsoft® Visual Studio installed on your system. At any rate, you will need to run/install the Intel® Fortran Compiler Runtime for Windows. You need only the latest version; it will work with older compiler versions.

Before running SWAN you must first complete a command file. Consult the SWAN User Manual how to specify the various settings and instructions to SWAN concerning the (input) grids, boundary conditions, physics, numerics and output. To help you in editing a command file for SWAN input, the file `swan.edt` is provided which contains the complete set of commands.

After completing the command file, you may run SWAN. Two command-line utilities are provided among the source code, one for running SWAN on the Windows platform, called `swanrun.bat`, and one for running SWAN on the Linux/UNIX platform, called `swanrun`. Basically, the run procedure carries out the following actions:

- 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` or `mpiexec` may be needed instead (usually provided in an MPI 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 setting *Environment Variables*. (Hit the Window key plus R to get command prompt. Then type *sysdm.cpl*, go to *Advanced* and select *Environment Variables*. Note that you must be an administrator in order to modify an environment variable.) In case of Linux or UNIX running the bash shell (sh or ksh), 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` is resided. In case of the C shell (csh), 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 MPI is resided to `PATH` to have access to the command `mpirun` or `mpiexec`.

You may also specify the number of threads to be used during execution of the multi-threaded implementation of SWAN on multiprocessor systems. The environment variable for this is `OMP_NUM_THREADS` and can be set like

```
export OMP_NUM_THREADS=4
```

or

```
setenv OMP_NUM_THREADS 4
```

or, in case of Windows,

```
OMP_NUM_THREADS = 4
```

When dynamic adjustment of the number of threads is enabled, the value given in `OMP_NUM_THREADS` represents the maximum number of threads allowed.

The provided run utilities 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 Linux/UNIX, 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 the parameter `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. Note that the script `swanrun` need to be made executable first, as follows:

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

The parameter `-omp n` specifies a parallel run on n cores using OpenMP. Note that the UNIX script will set `OMP_NUM_THREADS` to n. 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: $n = 1$). To redirect screen output to a file, use the sign >. Use an ampersand to run SWAN in the background. An example:

```
./swanrun -input f31har01 -omp 4 > swanout &
```

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 cores to launch on the node. This is useful e.g., for multi-core 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.

If indicated by the user, a single or multiple hotfiles will be generated depending on the number of processors, i.e. the number of hotfiles equals the number of processors (see the User Manual). Restarting a (parallel MPI) run can be either from a single (concatenated) hotfile or from multiple hotfiles. In the latter case, the number of processors must be equal to the number of generated hotfiles. If appropriate, the single hotfile can be created from a set of multiple hotfiles using the program `hcat.exe` as available from SWAN version 40.51A. This executable is generated from the Fortran program `swanhcat.ftn`. A self-contained input file `hcat.nml` is provided in the SWAN package. This file contains, e.g. the (basis) name of the hotfile. To concatenate the multiple hotfiles into a single hotfile just execute `hcat.exe`.

Chapter 6

Testing SWAN

The SWAN package consists of one executable file (`swan.exe`), a command file (`swan.edt`) and a run procedure (`swanrun.bat` or `swanrun`). The executable for Windows can be obtained from the SWAN website, but see Chapter 5 for further details. The input and output to a number of test problems is provided on the SWAN homepage. 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).