

SWASH

IMPLEMENTATION MANUAL

SWASH version 4.01A

SWASH IMPLEMENTATION MANUAL

by : The SWASH 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

website : *<http://swash.sourceforge.net>*

Copyright (c) 2010-2017 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

1	Introduction	1
1.1	The material	2
2	Use of patch files	7
3	Installation of SWASH on your computer	9
3.1	Quick installation	11
3.2	Manual installation	12
3.2.1	Modifications in the source code	12
3.2.2	Compiling and linking SWASH source code	13
3.3	Installation of MPICH2 on Windows	14
4	User dependent changes and the file swashinit	17
5	Usage of SWASH executable	21
6	Testing the system	25

Chapter 1

Introduction

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

- the SWASH source code,
- the SWASH executable for Microsoft Windows,
- the User Manual,
- this Implementation Manual,
- some test cases.

All of the material can be found on the SWASH website
<http://swash.sourceforge.net>.

On the SWASH website, general information is given about the functionalities, physics and limitations of SWASH. Moreover, the modification history of SWASH is given. Finally, information on support, links to the related websites and various free software are provided.

After downloading the material, you may choose between

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

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

For the purpose of implementation, you have access to the source code of SWASH and additional files, e.g. for testing SWASH. Please read the copyright in this manual and in the source code with respect to the terms of usage and distribution of SWASH. You are permitted to implement SWASH on your computer system. However, for any use of the SWASH 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 SWASH website to the computer system on which you want to run SWASH.
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 adaptations in installation-dependent parts of the code.
4. Compiling and linking the source code to produce an executable of SWASH.
5. Testing of the executable SWASH.

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

1.1 The material

The downloaded file `swash-4.01.tar.gz` contains the SWASH source code. You can unzip this file either with WinRAR (in case of Windows) or with the command `tar xzf` (in case of UNIX or Linux). The SWASH source code consists of the following files:

general modules	: ocpmod.ftn SwashModule1.ftn90 SwashModule2.ftn90
modules for flow	: SwashFlowdata.ftn90
modules for solvers	: SwashSolvedata.ftn90
modules for support parallel MPI runs	: m_parall.ftn
main program	: Swash.ftn90 SwashMain.ftn90
pre-processing routines	: SwashBackup.ftn90 SwashBCboundwave.ftn90 SwashBCshortwave.ftn90 SwashBCspecfile.ftn90 SwashBCspectrum.ftn90 SwashBounCond.ftn90 SwashCheckPrep.ftn90 SwashInit.ftn90 SwashInitBCtrans.ftn90

SwashInitCompGrid.ftn90
 SwashInitCompUgrid.ftn90
 SwashInitCond.ftn90
 SwashInitSteady.ftn90
 SwashInputField.ftn90
 SwashInputGrid.ftn90
 SwashReadBndval.ftn90
 SwashReadInput.ftn90
 SwashReadTestpnts.ftn90
 SwashReqOutL.ftn90
 SwashReqOutQ.ftn90
 computational routines : SwashAntiCreep1DH.ftn90
 SwashAntiCreep2DH.ftn90
 SwashBotFrict.ftn90
 SwashComputFlow.ftn90
 SwashComputStruc.ftn90
 SwashComputTrans.ftn90
 SwashComputTurb.ftn90
 SwashDensity.ftn90
 SwashDryWet.ftn90
 SwashBreakPoint.ftn90
 SwashExpDep1DHflow.ftn90
 SwashExpDep1DHtrans.ftn90
 SwashExpDep2DHflow.ftn90
 SwashExpDep2DHtrans.ftn90
 SwashExpLay1DHflow.ftn90
 SwashExpLayP1DHflow.ftn90
 SwashExpLay1DHtrans.ftn90
 SwashExpLay2DHflow.ftn90
 SwashExpLayP2DHflow.ftn90
 SwashExpLay2DHtrans.ftn90
 SwashFlowdata.ftn90
 SwashFlowDP.ftn90
 SwashGeometrics.ftn90
 SwashHDiffZplane1DH.ftn90
 SwashHDiffZplane2DH.ftn90
 SwashHorzVisc.ftn90
 SwashImpDep1DHflow.ftn90
 SwashImpDep2DHflow.ftn90
 SwashImpLay1DHflow.ftn90
 SwashImpLayP1DHflow.ftn90
 SwashImpLay2DHflow.ftn90
 SwashImpLayP2DHflow.ftn90

	SwashKepsMod1DH.ftn90
	SwashKepsMod2DH.ftn90
	SwashLayerIntfacs.ftn90
	SwashLogLaw.ftn90
	SwashPorousStruc.ftn90
	SwashPorFricDep.ftn90
	SwashPorFricLay.ftn90
	SwashSolvers.ftn90
	SwashSpongeLayer.ftn90
	SwashUpdateData.ftn90
	SwashUpdateDepths.ftn90
	SwashUpdateFld.ftn90
	SwashUpdFlowFlds.ftn90
	SwashUpdPress.ftn90
	SwashVeget.ftn90
	SwashVertVisc.ftn90
	SwashWindStress.ftn90
post-processing routines	: swashout2.ftn
	SwashCoorOutp.ftn90
	SwashDecOutL.ftn90
	SwashDecOutQ.ftn90
	SwashOutput.ftn90
	SwashQuanOutp.ftn90
	SwashAverOutp.ftn90
service routines	: SwashServices.ftn90
	SwashCleanMem.ftn90
	SwashPrintSettings.ftn90
SWAN 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

The source code is written in Fortran 90. Most of the routines are written in free form and are indicated by extension `f90`. Some routines are written in fixed form and depending on your system, the extension may be `for` or `f`. 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 SWASH, but Delft University of Technology will not support modified versions of SWASH. If you ever want your modifications to be implemented in the authorized version of SWASH (the version on the SWASH

website), you need to submit these changes to the SWASH team.

The source code is being attended with the following files:

installation procedures	:	INSTALL.README
		Makefile
		macros.inc
		getcml
		platform.pl
		switch.pl
run procedures	:	SWASHRUN.README
		swashrun
		swashrun.bat
machinefile for parallel		
MPI runs	:	machinefile
edit file	:	swash.edt

On the SWASH website, you also find some test cases for making a configuration test of SWASH on your computer. You may compare your results with e.g. analytical or laboratory data.

Chapter 2

Use of patch files

Between releases of authorised SWASH versions, it is possible that bug fixes or new features are published on the SWASH website. These are provided by patch files that can be downloaded from the website. 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. 4.01.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 4.01ABC, 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 SWASH is located
3. execute `patch -p0 < patchfile`

After applying a patch or patches, you need to recompile the SWASH 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 4.01.[A-C] | tr -d '\r' | patch` that apply the patch files 4.01.A to 4.01.C to the SWASH 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 SWASH website, which is appropriate for Windows operating system (7/8.1/10).

Chapter 3

Installation of SWASH on your computer

The portability of the SWASH code between different platforms is guaranteed by the use of standard ANSI FORTRAN 90. Hence, virtually all Fortran compilers can be used for installing SWASH. See also the manual Programming rules (to be found on the SWAN website http://swanmodel.sourceforge.net/online_doc/swanpgr/swanpgr.html).

The SWASH code is parallelized, which enables a considerable reduction in the simulation time for relatively large CPU-demanding calculations. 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 SWASH on a Linux cluster.

The material on the SWASH website provides a `Makefile` and two Perl scripts (`platform.pl` and `switch.pl`) that enables the user to quickly install SWASH 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
IBM Power6 (pSeries 575)	Linux	IBM
Intel Pentium (32-bit) PC	Linux	GNU (g95)
Intel Pentium (32-bit) PC	Linux	GNU (gfortran)
Intel Pentium (32-bit) PC	Linux	Intel
Intel Pentium (64-bit) PC	Linux	Intel
Intel Itanium (64-bit) PC	Linux	Intel
Intel Pentium (64-bit) PC	Linux	Portland Group
Intel Pentium (32-bit) PC	Linux	Lahey
Intel Pentium (32-bit) PC	MS Windows	Intel
Intel Pentium (64-bit) PC	MS Windows	Intel
Intel Pentium (32-bit) PC	MS Windows	Compaq Visual
Power Mac G4	Mac OS X	IBM
MacBook	macOS	GNU (gfortran)
MacBook	macOS	Intel

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

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://www.activestate.com/activeperl/downloads>. The Perl version should be at least 5.0.0 or higher!

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

- serial runs or
- parallel runs on a multi-core PC or a Linux cluster.

For a typical depth-averaged flume computation, it may be sufficient to choose the serial mode, i.e. one SWASH program running on one processor. The parallel mode is more convenient for a relatively large CPU-demanding multi-layer flume (2DV) or basin-like (2DH/3D mode) calculation.

For a proper installation of MPI on Windows (7/8.1/10), please consult Section 3.3.

3.1 Quick installation

Carry out the following steps for setting up SWASH 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, SWASH can be built for serial or parallel mode, as follows:

mode	instruction
serial	<code>make ser</code>
parallel	<code>make mpi</code>

IMPORTANT NOTES:

- To Windows users:
 - To execute the above instructions, just open a command prompt.
 - To build SWASH on Windows platforms by means of a Makefile you need a `Nmake` program. Such a program can be downloaded freely from Internet.
 - This installation currently supports MPICH2 for Windows 7/8.1/10; older versions are probably not supported. See Section 3.3 for further information.
 - It is assumed that both the directories `include` and `lib` are resided in `C:\PROGRAM FILES\MPICH2`. If not, the file `macros.inc` should be adapted such that they can be found by the Makefile.
- One of the commands `make ser` 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 SWASH 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 SWASH 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] [-mpi] [-cray] [-sgi]
               [-cvis] [-ting] [-mat14] [-impi] *.ftn[90]
```

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).
- 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 SWASH 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 files `swanpar11.ftn`, `Swash*2DHflow.ftn90` and `m_parallel.ftn`.
- 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 SWASH executables. This can be remedied by using the option SHARED in the OPEN statement for shared access. For this, just remove the switch !CVIS.

- timing** If the user want to print the timings (both wall-clock and CPU times in seconds) of different processes within SWASH then remove the switch `!TIMG`. Otherwise, no timings will be kept up and subsequently printed in the `PRINT` file.
- mat14** 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 `m_parallel.ftn` must be included by removing the switch `!/impi`.

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`.

3.2.2 Compiling and linking SWASH 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 SWASH. The source code cannot be compiled with a Fortran 77 compiler. If you intended to use MPI for parallel runs, you may use the command `mpif90` instead of the original compiler command.

The SWASH 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 SWASH 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, SWASH will not run correctly.

Furthermore, SWASH can generate binary MATLAB files on request, which are unformatted. Some compilers, e.g. Intel Fortran version 11.x, 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 for Windows this is `/assume:byterecl` and for Linux this is `-assume byterecl`.

The modules must be compiled first. Several subroutines use the modules. These subroutines need the compiled versions of the modules 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 `swash.exe` after linking.

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

```
mpif90 SwashModule?.f90 m_parallel.f ocpmod.f SwashFlowdata.f90
      SwashSolvedata.f90 SwanGriddata.f90 SwanGridobjects.f90
      SwanCompdata.f90 ocp*.f swanser.f Swash*.f90 Swan*.f90
      -o swash.exe
```

3.3 Installation of MPICH2 on Windows

SWASH 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. This section instructs you how to install MPI in the Windows environment. It is assumed that your OS is either Windows 7, 8.1 or 10.

- Install MPICH2 version 1.4.1p1 on your machine using the installer downloaded from the downloads page <http://www.mpich.org/static/downloads/1.4.1p1/>. Be aware to choose either a 32-bit version or 64-bit one depending on your machine.
- During installation select MPICH2 to be installed for **Everyone**.
- Also make sure that you keep the default setting for passphrase (usually **behappy**) during the installation.
- After installing MPICH2 successfully, you may add rules to your firewall to allow the programs `mpiexec.exe` and `smpd.exe` to communicate through firewall. This process depends on your firewall.
- You need to add the `bin` folder of MPICH2 (usually `C:\PROGRAM FILES\MPICH2\bin`) to your `PATH`. Note that this should be added before the `bin` folder of Intel Fortran compiler (version 14 or later, which supports MPI as well)!
- Next, open an admin command prompt by right-clicking on the command prompt icon and selecting **run as administrator**. In the command prompt type the following commands in sequence:


```
smpd -install
mpiexec -remove
mpiexec -register
```

`mpiexec -validate` (it should return **SUCCESS**)

`smpd -status` (it should return 'smpd running on <hostname>')

If the last two commands are successfully, then you are ready to run SWASH in parallel. See Chapter 5.

Note that when you register, you should indicate your username and password as a user. Moreover, if your machine is not part of a domain, do not specify any domain name.

Alternatively, the Intel MPI library may be employed instead of MPICH2 to build SWASH. This library is a part of Intel Fortran compiler 14.0 or higher. In this respect, the following modifications need to be made

- adapt the file `macros.inc` where the variables `INCS_MPI` and `LIBS_MPI`, referring to the directories `include` and `lib` of MPICH2, respectively, are emptied,
- change the value of the variable `F90_MPI` by replacing `ifort` by `mpiifort`, and
- the `bin` folder of MPICH2 must be removed from your `PATH`.

Chapter 4

User dependent changes and the file `swashinit`

SWASH 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 `swashinit`, which is created during the first time SWASH is executed on your computer system. The changes in `swashinit` only affect the runs executed in the directory that contains that file.

A typical initialisation file `swashinit` may look like:

```
5 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
7 default time coding option
-1 grid partitioning 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 SWASH can verify whether the file it reads is a valid initialisation file. The current version is 5.
- 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 website, 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 `swashinit`, 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)
 7. 153000.000

Option 7 only represents the time (hours, minutes, seconds and milliseconds) and hence, does not taken into account the date. Note that this option can not be used for a simulation longer than a day. In most cases, this option is the appropriate one for SWASH. Note that the ISO-notation has no millenium problem, therefore the ISO-notation is recommended. In case of other options, except the last one, the range of valid dates is in between January 1, 1911 and December 31, 2010 (both inclusive).

- For a parallel MPI run the computational grid needs to be decomposed into a number of subdomains. This decomposition is based on the grid partition. Two methods are available:
 1. stripwise manner
 2. orthogonal recursive bisection (ORB)

The ORB method starts with a single part (the entire domain) after which each part is recursively partitioned by bisecting it, until all parts have been created. The bisection direction is swapped in each direction. This partitioning method can only be applied in the 2D multi-layered mode. In the other modes, 1D or depth-averaged, a stripwise partitioning will be applied. This will be done automatically by choosing option `-1`. Otherwise, you may choose option 1 for stripwise partitioning along the x - or y -axis, option 2 for ORB partitioning, option 3 for stripwise partitioning along the x -axis or option 4 for stripwise partitioning along the y -axis.

- 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, SWASH 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 SWASH executable

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

Two run procedures are provided among the source code, one for the Windows platform, called `swashrun.bat`, and one for the UNIX/Linux platform, called `swashrun`. Basically, the run procedure carries out the following actions:

- Copy the command file with extension `sws` to `INPUT` (assuming `INPUT` is the standard file name for command input, see Chapter 4).
- Run SWASH.
- 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` is needed and is provided in the MPICH2 distribution.

Before calling the run procedure, the environment variable `PATH` need to be adapted by including the pathname of the directory where `swash.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*). In case of UNIX or Linux running the bash shell (`sh` or `ksh`), the environment variable `PATH` may be changed as follows:

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

if `/usr/local/swash` is the directory where the executable `swash.exe` is resided. In case of the C shell (`csh`), use the following command:

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

If appropriate, you also need to add the directory path where the `bin` directory of MPICH2 is resided to `PATH` to have access to the command `mpiexec`.

The provided run procedures enable the user to properly and easily run SWASH both serial

as well as parallel. Note that for parallel MPI runs, the executable `swash.exe` should be accessible by copying it to all the multiple machines or by placing it in a shared directory. When running the SWASH program, the user must specify the name of the command file. However, it is assumed that the extension of this file is `sws`. 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 `swashrun.bat` from a command prompt:

```
swashrun filename [nprocs]
```

where `filename` is the name of your command file without extension (assuming it is `sws`) and `nprocs` indicates how many processors 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 `swashrun` is as follows:

```
./swashrun -input filename -mpi n
```

where `filename` is the name of your command file without extension. Note that the script `swashrun` need to be made executable first, as follows:

```
chmod +rx ./swashrun
```

The parameter `-mpi n` specifies a parallel run on n cores using MPI. The parameter `-input` is obliged, whereas the parameter `-mpi n` can be omitted (default: $n = 1$). To redirect screen output to a file, use the sign `>`. Use an ampersand to run SWASH in the background. An example:

```
./swashrun -input l51con01 -mpi 4 > swashout &
```

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`.

SWASH 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 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 SWASH 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 occurred during the calculation, such as non-convergence of an iterative matrix solver. Existence of this file is an indication to study the flow in that grid point with more care.

Chapter 6

Testing the system

The SWASH system consists of one executable file (`swash.exe`), a command file (`swash.edt`) and a run procedure (`swashrun.bat` or `swashrun`). Depending on your system, you may use 32-bit or 64-bit executable. These executables for Windows 7/8.1/10 can be obtained from the SWASH website. The input and output to a number of test problems is provided on the SWASH website. The files with extension `sws` 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 SWASH on your computer. Compare the results with those in the provided plot files.

To run the SWASH program for the test cases, at least 500 MBytes of free internal memory is recommended. For more realistic cases 1 to 2.5 GBytes may be needed, whereas for more simple 1D cases significant less memory is needed (less than 100 MBytes).