SFINCS Documentation

Release 2.0.2_Blockhaus

Tim Leijnse

Jun 09, 2023

INTRODUCTION:

'SFINCS' is Deltares' new reduced-complexity model designed for super-fast modelling of compound flooding in a dynamic way! This online documentation describes the input files and parameters of the SFINCS model (Super-Fast INundation of CoastS), model output and how to get started.

Contact:

Mail 'sfincs@deltares.nl' for questions that are not answered yet by this documentation.

License:

Quick information:

- For an introduction of the SFINCS model see the introduction journal publication, see: https://doi.org/10.1016/j.coastaleng.2020.103796
- For more general information about SFINCS and Deltares, see: https://deltares.nl/en/software/sfincs/
- To download a free pre-compiled windows executable of SFINCS, see: https://download.deltares.nl/en/download/sfincs/
- To collaborate with us on improving the model, see: https://github.com/Deltares/SFINCS
- To raise possible issues with the latest version of the model, see: https://github.com/Deltares/SFINCS/issues
- For a background on SFINCS, application areas, performed international projects and publiscations, see "Introduction" in this manual.

Outline manual: This manual is outlined containing the following parts, see also the overview bar on the left:

- Introduction
- Developments
- Getting started
- User manual
- Input parameters and files
- Model output and messages

CHAPTER

ONE

ACKNOWLEDGEMENTS

SFINCS is developed at Deltares and initiated by Maarten van Ormondt. This documentation is developed and maintained by Tim Leijnse and Roel de Goede.

1.1 Introduction

1.1.1 Introduction to SFINCS

What is SFINCS?

SFINCS (Super-Fast Inundation of CoastS) is a reduced-complexity model capable of simulating compound flooding with a high computational efficiency balanced with an adequate accuracy. In SFINCS a set of momentum and continuity equations are solved with a first order explicit scheme based on Bates et al. (2010). Traditionally SFINCS neglects the advection term (SFINCS-LIE) which generally justified for sub-critical flow conditions. For super-critical flow conditions or when modelling waves, the advection term needs to be solved. For this purpose, the SFINCS-SSWE version can be used (including advection). For more information see Leijnse et al. (2020): https://doi.org/10.1016/j.coastaleng.2020.103796

Why SFINCS?

Compound flooding during tropical cyclones and other extreme events result in tremendous amounts of property damage and loss of life. Early warning systems and multi-hazard risk analysis can reduce these impacts. However, large numbers of computations need to be run in a probabilistic approach and in a short time due to uncertainties in the meteorological forcing. Current modelling approaches are either fast but too simple (bathtub approach) or models are very accurate but too slow (e.g. Delft3D, XBeach). SFINCS balances a high computational efficiency with adequate accuracy. Therefore the model is very appropriate to either to model a large stochastic set of scenarios, run the same model on a higher resolution or model larger scales. Thereby, one can use it as a quickscan tool to quickly test possible adaptation and mitigation measures to flooding. The scope of the model is therefore different than other models in Deltares' modelling suite. When very high-detailed simulations are needed with a lot of detailed features (or including morphology, salinity etc.), one can use the Delft3D FM Suite, see: https://www.deltares.nl/en/software/delft3d-flexible-mesh-suite/

Fig. 1.1: The goal of SFINCS: speed! (Icon made by https://www.flaticon.com/authors/vectors-market)

Compound flooding?

Compound flooding is described as events occurring in coastal areas where the interaction of high sea levels, large river discharges and local precipitation causes (extreme) flooding (Wahl et al., 2015). To simulate compound flooding events, a model needs to be able to model all these types of forcings. Therefore, SFINCS includes fluvial, pluvial, tidal, wind- and wave-driven processes!

1.1.2 Application areas

Coastal model

A SFINCS model in coastal regions can be forced with marine forcings like tides, storm surge, local wind setup and wave driven processes. Generally a model is setup with the offshore boundary in the swash zone, good practice is in about 2 meters water depth. In SFINCS it is possible to distinguish cells that are made inactive in the computation so it will not slow your model down (in this case everywhere deeper than 2m water depth). In some cases local rainfall might be relevant too for a coastal model.

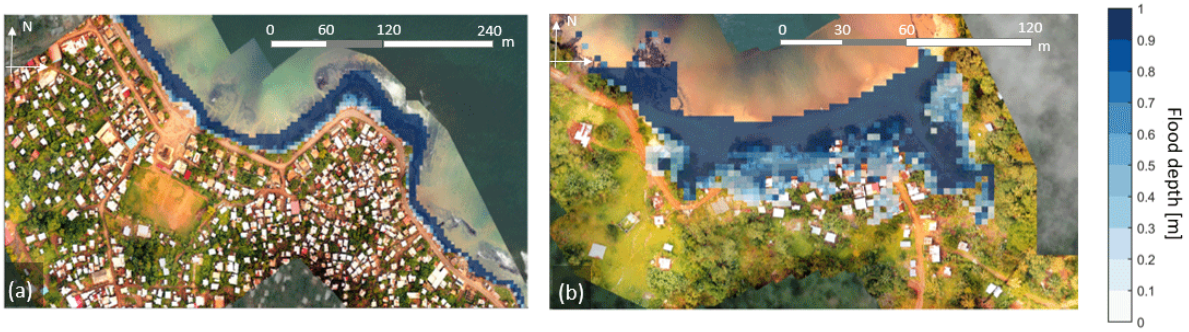


Fig. 1.2: SFINCS model for Sao Tome en Principe, figure from: https://doi.org/10.5194/nhess-20-2397-2020

Coral reef model

SFINCS models have also been setup in coral reef type environments, where individual waves are forced to compute wave-driven flooding. This generally has a large contribution to flooding for Small Island Developping States (SIDS) or other coasts/islands with coral reef type coasts.

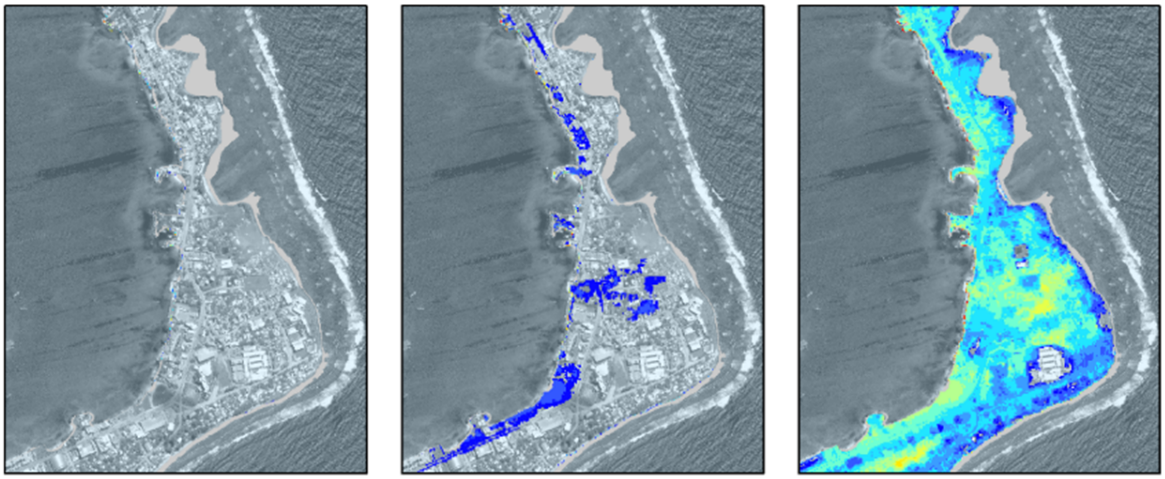


Fig. 1.3: SFINCS model for Majuro.

Tsunami model

As an additional type of coastal model, SFINCS has also been used for modelling tsunami's. Generally this would be an overland model forced with a tsunami wave as computed by an offshore hydrodynamic model. However, in the paper of Robke et al. 2021 SFINCS was also used for the first time to calculate the offshore propagation in a very short amount of time too. Get in touch to hear more about possibilities for tsunami modelling with SFINCS.

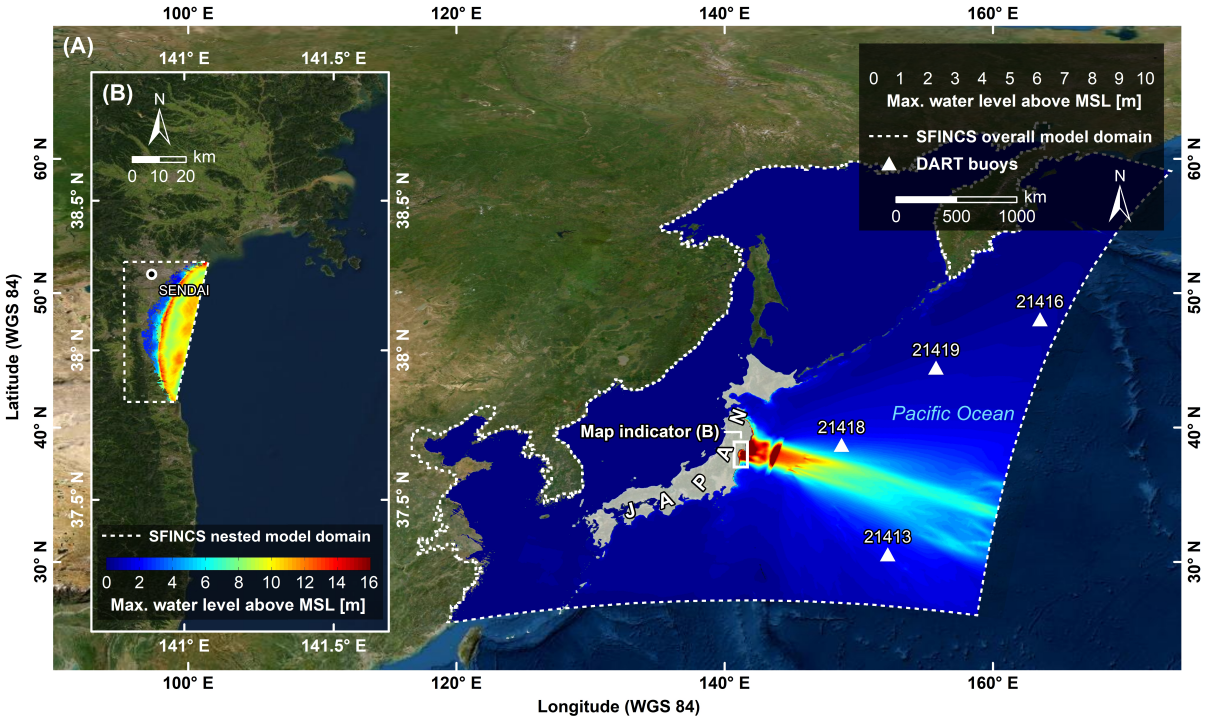


Fig. 1.4: Overland and offshore SFINCS models modelling the 2011 Tohoku tsunami near Japan, figure from: https://doi.org/10.3390/jmse9050453

Storm surge model

Since speed is wanted everywhere, also tests have been done to let SFINCS model offshore storm surge during tropical cyclones. Get in touch to hear more about possibilities for storm surge modelling with SFINCS.

Riverine model

For inland riverine types of environments, boundary conditions are generally different than for coastal models. Generally at the upstream end of rivers, one can provide discharge points with discharge time-series. At the downstream end of rivers, water level time-series need to be specified, which in case of sub-critical flow conditions will influence the flow upstream. Additionaly, besides the general river discharge, local rainfall adding water to the river can be very relevant too.

Urban model

For urban environments the local situation of varying land use conditions can heavily influence the local flow. Therefore spatially varying input of manning roughness and infiltration is possible. The curve number method of infiltration will distinguish what part of falling precipitation can infiltrate or will run-off. To test out the effect of interventions, it is possible to insert different types of structures into the SFINCS model. These can be thin dams, levees, sea walls, simple drainage pumps or culverts.

Fig. 1.5: SFINCS model for Vientiane, Laos.

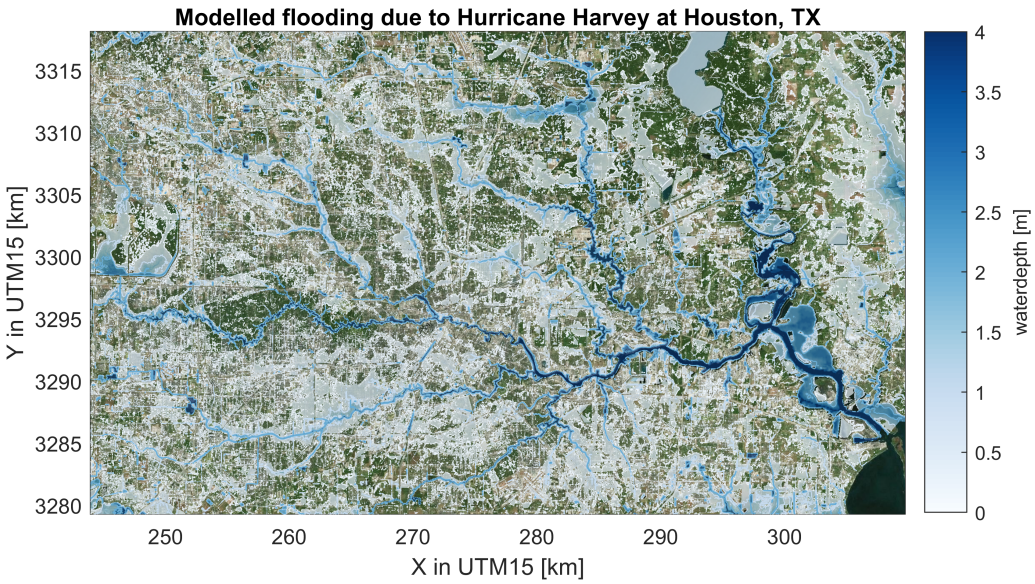


Fig. 1.6: SFINCS model for Houston, TX, during Hurricane Harvey (2017)

Flash flood model

In recent tests, SFINCS has also been used to model flash floods. In these events, a short but intense rainfall event falls onto a domain and together with a steep profile can lead to significant water depths and flow velocities. Get in touch to hear more about possibilities for fast flash-flood modelling with SFINCS.

Compound flooding model

In a compound flooding model, all relevant types of forcing from either coastal, coral, riverine or urban models can be combined into 1 domain. Hereby the joint effect of multiple flood drivers that can enhance flooding can be taken into account.

1.1.3 Applied international projects

SFINCS has been applied in these international projects, with attached links to news articles:

- Modelling of urban flooding and adaptation measures in the USA (https://www.deltares.nl/en/news/development-community-oriented-decision-support-tool-compound-flood-events-us/)
- Modelling coastal driven flooding at Beira, Mozambique (https://www.deltares.nl/en/news/dutch-mozambican-consortium-to-protect-beira-against-coastal-flooding/)
- Modelling sea level rise and storm driven flooding at 18 countries in the Caribbean (https://openknowledge.worldbank.org/handle/10986/36417)
- Modelling multi-hazard driven flooding for the atoll of Majuro in the Marshall islands (https://storymaps.arcgis.com/stories/8c715dcc5781421ebff46f35ef34a04d)
- Modelling compound flooding along the whole US Southeast coast (https://www.deltares.nl/app/uploads/2021/10/RD-Highlights-2021.pdf)

SFINCS has been applied (or still is) in multiple other international projects:

- Modelling compound flooding for the islands of Sao Tome en Principe

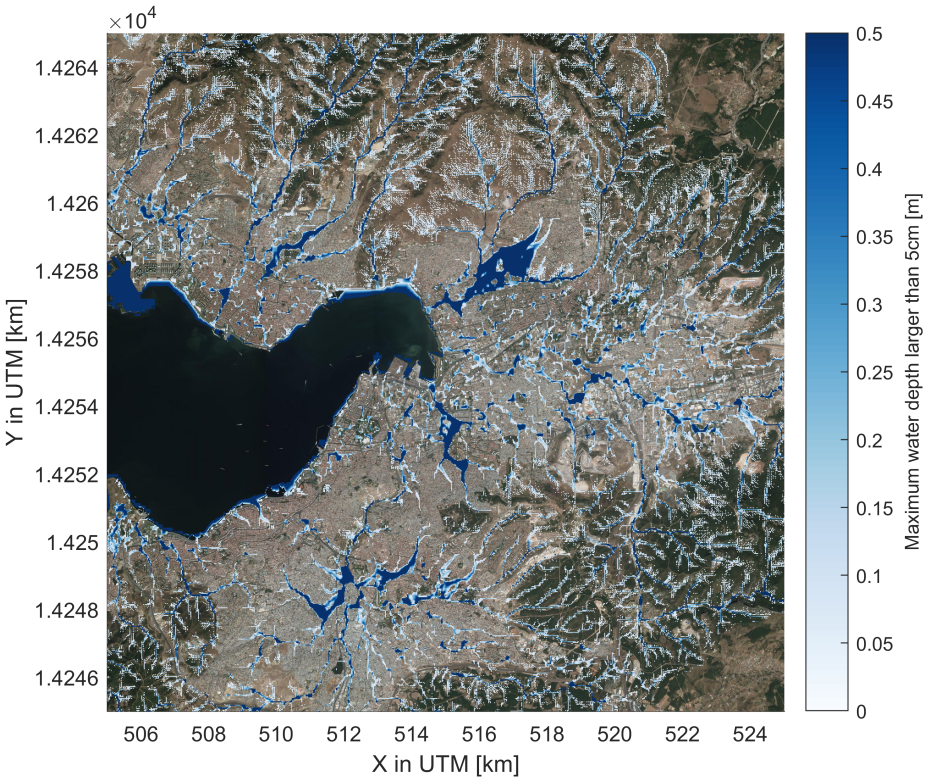


Fig. 1.7: SFINCS model for Izmir, Turkey

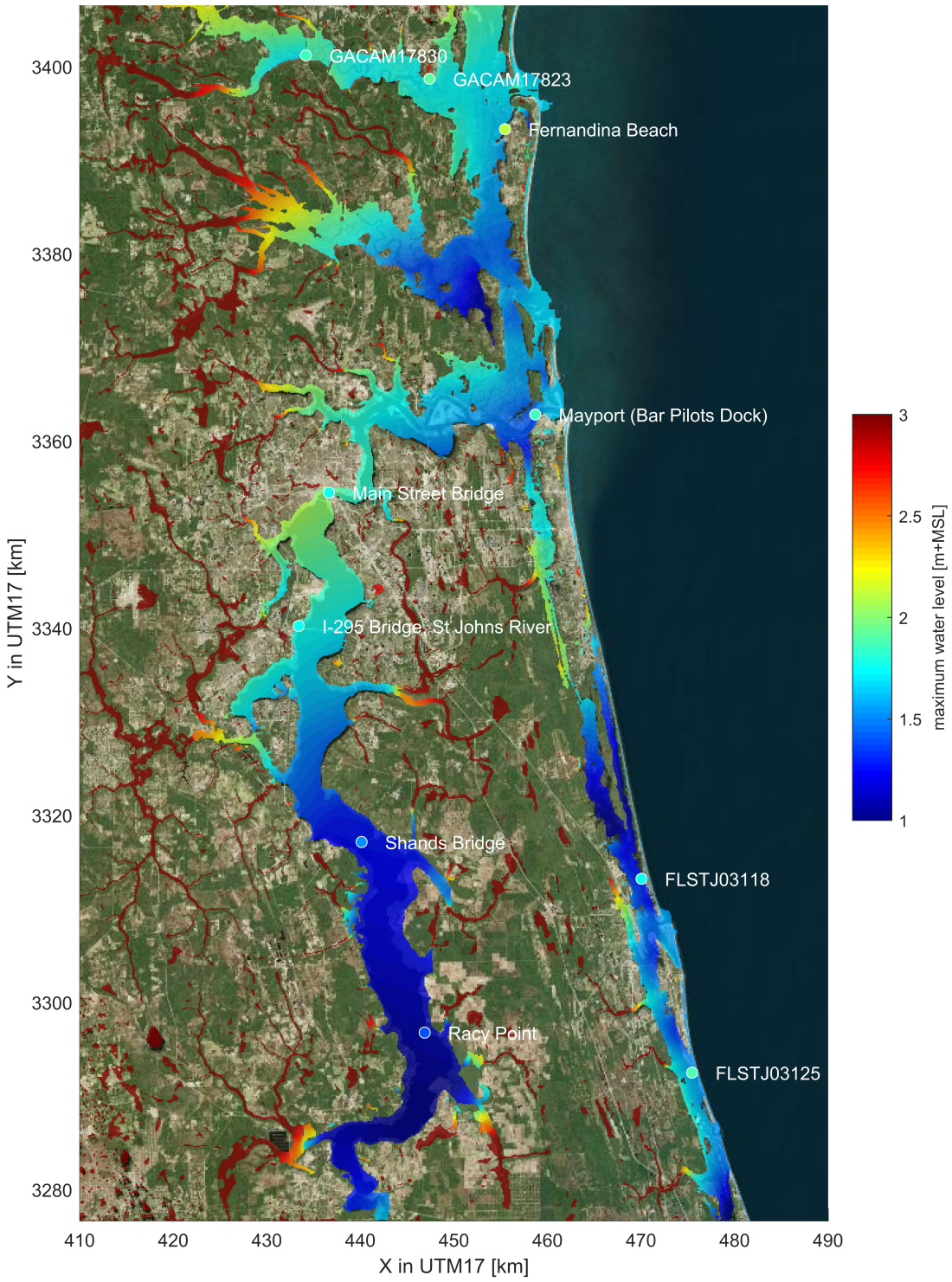


Fig. 1.8: SFINCS model for Jacksonville, FL, during Hurricane Irma (2017), figure from: https://doi.org/10.1016/j.coastaleng.2020.103796

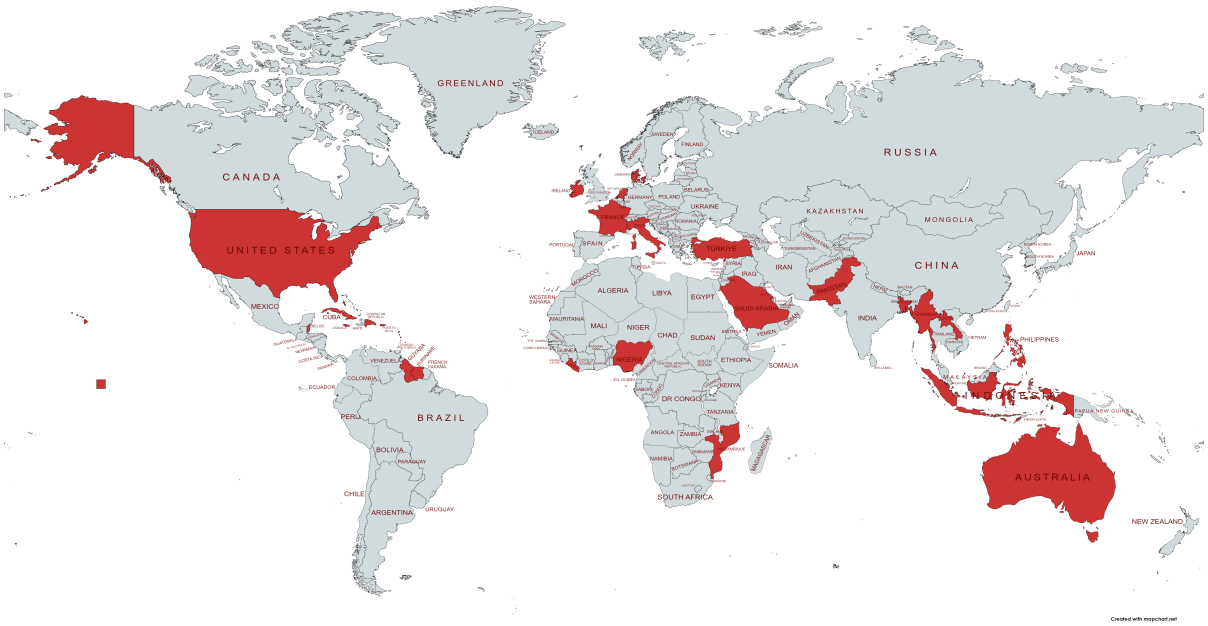


Fig. 1.9: Overview of countries globally where SFINCS has been used, including all SIDS

- Modelling tropical cyclone and sea level rise driven flooding in polders of Bangladesh
- Modelling compound flooding at Monrovia, Liberia
- Modelling sea level rise driven flooding at all the islands of the Marshall Islands
- Modelling wave and groundwater -driven flooding across the Puget Sound, US West coast
- Modelling wave-driven flooding at Miami, Florida
- Modelling urban flooding in 100 global cities
- Modelling sea level rise and storm driven flooding for all SIDS globally
- Modelling coastal flooding in Denmark
- Modelling wave-driven flooding in Cuba
- Modelling flash-floods in Turkey
- Modelling large scale compound flooding in Australia in a Delft-FEWS early warning system
- Modelling wave-driven flooding on coral reeflined coasts of Puerto Rico
- Modelling coastal and riverine flooding in Indonesia
- Modelling wave-driven flooding in the Philippines
- Modelling of urban flooding and adaptation measures in Ireland
- Modelling emergency response of flooding in Pakistan and Nigeria during the 2022 floods

1.1.4 Publications

There have been various journal publications and conference posters where SFINCS has been used and/or validated:

- Introduction paper of SFINCS: "Modeling compound flooding in coastal systems using a computationally efficient reduced-physics solver: including fluvial, pluvial, tidal, wind- and wave-driven processes". Leijnse et al. (2021). https://doi.org/10.1016/j.coastaleng.2020.103796.
- "Uncertainties in coastal flood risk assessments in small island developing states" - Parodi et al. (2020) https://nhess.copernicus.org/articles/20/2397/2020/
- "Hindcast of Pluvial, Fluvial, and Coastal Flood Damage in Houston, Texas during Hurricane Harvey (2017) using SFINCS". Sebastian et al. (2021). Sebastian et al. https://doi.org/10.1007/s11069-021-04922-3
- "Rapid Assessment of Tsnuami offshore propagation and Inundation with D-FLOW Flexible Mesh and SFINCS for the 2011 Tohoku Tsunami in Japan": Röbke et al. (2021) https://doi.org/10.3390/jmse9050453
- "Efficient and accurate modeling of wave-driven flooding on coral reef-lined coasts: Case Study of Majuro Atoll, Republic of the Marshall Islands". Bertoncelj et al. (2021). https://doi.org/10.5194/egusphere-egu21-5418
- "Multilevel multifidelity Monte Carlo methods for assessing coastal flood risk". Clare et al. (2022) https://doi.org/10.5194/nhess-22-2491-2022
- "A globally-applicable framework for compound flood hazard modeling". Eilander et al. (2022) https://doi.org/10.5194/egusphere-2022-149
- "Developing large scale and fast compound flood models for Australian coastlines". Leijnse et al. (2022). 'International Conference on Coastal Engineering 2022, Sydney'
- "Developing a real-time data and modelling framework for operational flood inundation forecasting in Australia". De Kleermaeker et al. (2022). https://search.informit.org/doi/abs/10.3316/informit.916755150845355
- "Flooding at the Fringe: A Reduced-physics Model for Assessing Compound Flooding from Pluvial, Fluvial, and Coastal Hazards". Grimley et al. (2022). https://ui.adsabs.harvard.edu/abs/2022AGUFMNH36A..03G/abstract
- "Tropical cyclones or extratropical storms: What drives the compound flood hazard, impact and risk for the US Southeast Atlantic coast?" Nederhoff et al. (2023). https://eartharxiv.org/repository/view/5123/.
- "Dynamic modeling of coastal compound flooding hazards due to tides, extratropical storms, waves, and sea-level rise: a case study in the Salish Sea, Washington (USA)". Nederhoff et al. (2023). https://eartharxiv.org/repository/view/5140/
- "RAPID MODELING OF COMPOUND FLOODING ACROSS BROAD COASTAL REGIONS AND THE NECESSITY TO INCLUDE RAINFALL DRIVEN PROCESSES: A CASE STUDY OF HURRICANE FLORENCE (2018)". Leijnse et al. (2023). https://doi.org/10.1142/9789811275135_0235
- "FORECASTING HURRICANE IMPACTS ON COASTS USING COASTAL STORM MODELING SYSTEM (COSMOS)". Van Dongeren et al. (2023). https://doi.org/10.1142/9789811275135_0242
- "An Integrated Assessment of Climate Change Impacts and Implications on Bonaire". Van Oosterhout (2023). https://link.springer.com/article/10.1007/s41885-023-00127-z
- More information regarding recent advancements with subgrid features can be seen in this online poster: https://agu2020fallmeeting-agu.ipostersessions.com/Default.aspx?s=9C-05-18-CF-F1-2B-17-F0-7A-21-93-E6-13-AE-F3-24

Leijnse et al. (2020)

- More information regarding recent comparisons using SFINCS to model waves can be seen in this online poster: https://agu2020fallmeeting-agu.ipostersessions.com/?s=30-38-5A-0C-8E-22-C8-84-CC-46-3C-95-18-80-C2-76&token=5r05NKrASrFrnkybTgXa4Y_XvgfV233Dazers0d2Zzo

Lasserre et al. (2020)

1.2 Developments

SFINCS has continuely being developed since 2017, and many great features have been added over the years. Hereby some examples regarding subgrid features and GPU computing.

1.2.1 Releases

Official open source version Q2 2023: v2.0.2 Blockhaus release

As the first out of 2 official 2023 releases, the v2.0.2 Blockhaus release, 'Smoothly cycling over challenges in compound flood modelling', is now available as Windows executable: https://download.deltares.nl/en/download/sfincs/

And Docker container: docker pull deltares/sfincs-cpu:sfincs-v2.0.2-Blockhaus

This contains open access to the source code from Github: https://github.com/Deltares/SFINCS/releases/tag/v2.0.2.

The code consists of all functionality of the v2.0.0 release, with the following changes/additions:

- Potentially breaking change: flipped x&y coordinates in Netcdf map output to be Sgrid compliant. Note; might impact Matlab/Python post-processing scripts (fixed in new HydroMT-SFINCS release v1.1.0)
- Improved 2D component of advection scheme
- Option to not use rainfall in spiderweb, keyword: usespwprecip = 0
- The x&y-coordinates of input weirfiles as snapped on grid internally in SFINCS are now written to the sfincs_his.nc file; structure_x, structure_y & structure_height
- Option to include viscosity, enabling running on theta=1.0, with viscosity = 1. The values 'nuvisc' will be automatically determined based on your grid resolution, and written to the log screen. Value can still be overruled by specifying 'nuvisc = value' directly, or increased with e.g. a factor 2 using 'nuviscdim = 2'.
- Save maximum velocity proxy (in m/s) on 'dtmaxout' interval: storevelmax = 1
- Save maximum flux (h * U in m^2/s) on 'dtmaxout' interval: storefluxmax = 1
- Save maximum discharge through drainage strucuture from 'drnfile' input on 'dthisout' interval: storeqdrain = 1
- Bugfix in weir formulation
- Updated documentation
- Added tests in skillbed report
- Compliance with new Python setup tools HydroMT-SFINCS release v1.1

Official open source version: v2.0.0 Alpe d'Huez release

On the 16th of November 2022, we have made SFINCS open source available as the SFINCS v2.0.0 Alpe d'Huez release, 'Moving Dutch Mountains in compound flood modelling'. This contains open access to the source code and executables from Github: https://github.com/Deltares/SFINCS. The code consists of all functionality of v1, with the large addition of the subgrid mode and first GPU functionality using openacc. For more details, see below.

Pre-release version(s): v1 revision XXX

Before making SFINCS open source, version history was controlled using subversion numbering. Therefore papers using pre-release versions of SFINCS for instance refer to 'trunk revision 141', as in Leijnse et al. 2021. These version 1 revisions contained all standard SFINCS functionality for the regular mode.

1.2.2 Recent advancements in accuracy: subgrid mode

What are subgrid features?

Subgrid features are a method in which flux computations are performed on a coarser grid than the update of the water levels which is done on a much finer resolution. In this way computations can be sped up, while still using high resolution information of topography and bathymetry.

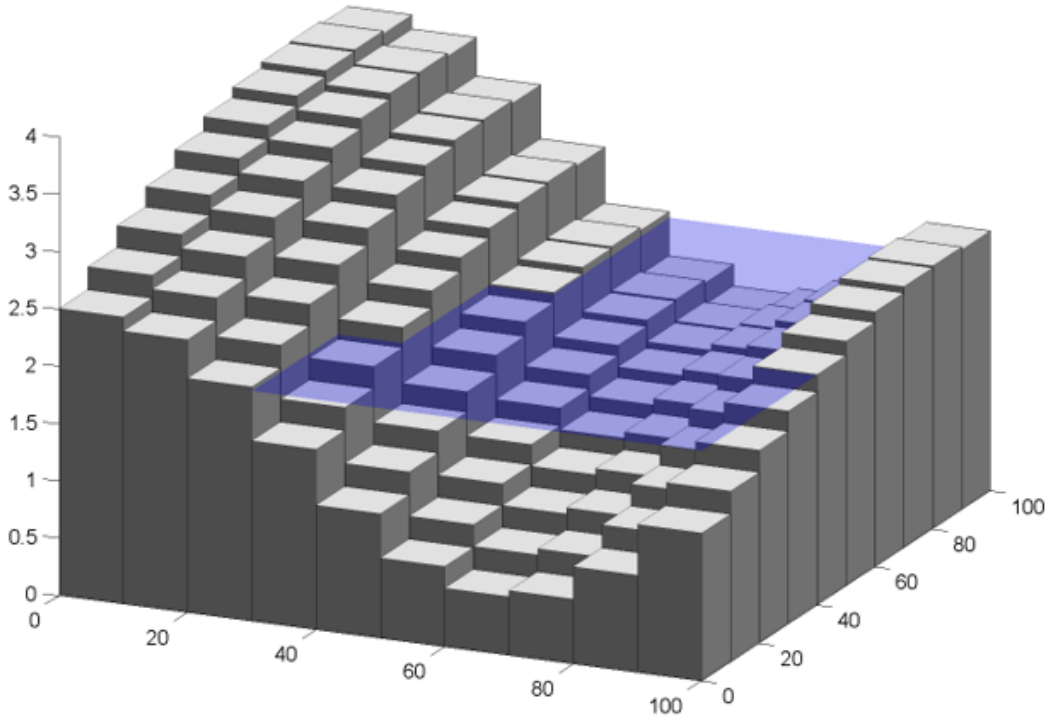


Fig. 1.10: Example subgrid features within one grid cell

Why subgrid features?

Often model runtimes are too large to go to very fine resolution modelling because refining a grid size with a factor 2, leads to a 2^3 longer model runtime due to the time step limitation in the CFL-criteria. This can be overcome by using a subgrid approach for the continuity update. This has the benefit that larger grid domains can be used while keeping accurate results.

How does it work?

The subgrid method implemented so that subgrid tables are derived in pre-processing that contain relations between the water level and volume for every grid cell. These tables are derived using high resolution topography and bathymetry data. In the SFINCS model itself, these subgrid tables are used to determine an accurate estimation of the water level after calculating fluxes on a coarser grid resolution. Additionally, for calculating the fluxes between cells, a representative water depth is determined. The makes is possible to compute on a coarser grid resolution (improvement of efficiency) while still detailed information about the local elevation is incorporated when determining corresponding water levels leading to accurate results.

Increase in computational efficiency?

Due to this time step limitation, if one can calculate fluxes on a 200 m grid instead of a 100m grid, the computational speedup is a factor 8. Our case study in Houston shows that even larger increases in speed are possible! See: https://agu2020fallmeeting-agu.ipostersessions.com/Default.aspx?s=9C-05-18-CF-F1-2B-17-F0-7A-21-93-E6-13-AE-F3-24

1.2.3 Recent advancements in speed: GPU enabled

The SFINCS source code has now been GPU enabled to make optimal use of fast Graphics Processing Unit computers. For more information get in touch with us!

1.3 Setting up models

1.3.1 Introduction

SFINCS models can be set up using simple ascii text and/or binary input files, which can be generated on whatever platform suiting you as a user best. While SFINCS itself is coded in Fortran, it does not matter whether you create the input using a text editor, Matlab or Python. To make the setting up of basic models easier, making a SFINCS model is supported by 3 open source available options:

- Graphical User Interface (GUI) to make SFINCS models interactively: **'Delft Dashboard'**
- Matlab scripts toolbox in the **'Open Earth Toolbox'**
- Command line/Python script framework called **'HydroMT'**, with its dedicated plugin **'hydromt_sfincs'**

If you need a more tailor-made solution for setting up your SFINCS models get in touch with us!

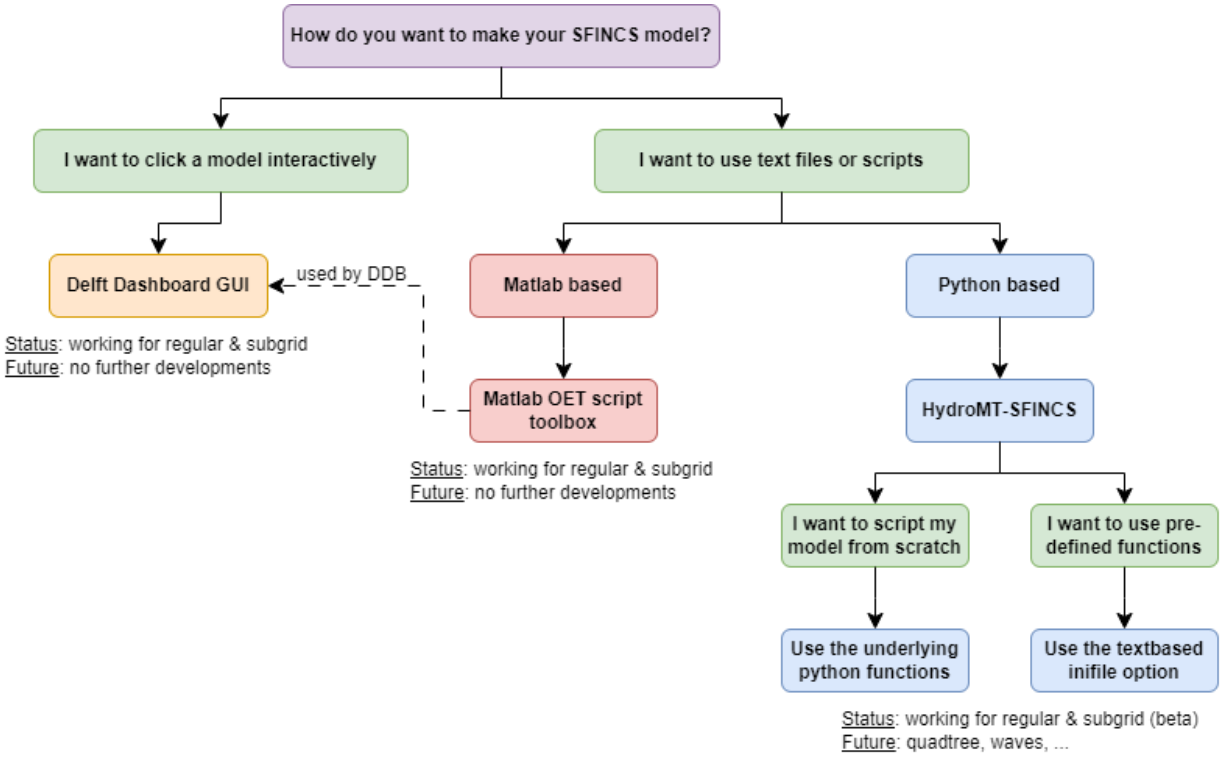


Fig. 1.11: Overview of input file of SFINCS with indication whther they are required or not

1.3.2 Delft Dashboard

Delft Dashboard (Van Ormondt et al. 2020 https://doi.org/10.2166/hydro.2020.092) is a quick set-up tool GUI for hydrodynamic models that includes setting up SFINCS models. The tool can be run on Matlab or as standalone executable and has all the basic functionality to setup your basic model using globally available DEMs in an interactive way.

For more information see: https://publicwiki.deltares.nl/display/DDB

1.3.3 Open Earth Tools

In the Open Earth Tools a toolbox of Matlab scripts is included to have more flexibility in setting up your SFINCS models, that is also used by Delft Dashboard. Throughout this User Manual examples of how to use these scripts are given in the code blocks '**Matlab example using OET**'. For a general overview of possible input files and how to create them using Matlab scripts see: https://svn.oss.deltares.nl/repos/openearthtools/trunk/matlab/applications/sfincs/ For getting started with Open Earth Tools see: https://publicwiki.deltares.nl/display/OET/OpenEarth

1.3.4 HydroMT-SFINCS

HydroMT (Eilander et al. 2022 https://doi.org/10.5194/egusphere-2022-149) is a more recent addition to the tools available for setting up SFINCS models, and is a Python based command-line alternative. Besides globally available DEMs it can also retrieve spatially varying infiltration and manning roughness data based on landuse maps. Also, it is possible to burn in rivers or setup a offline coupled model together with the hydrological Wflow model that will provide boundary conditions as river discharge.

- For more information regarding the SFINCS plugin of HydroMT see: https://deltares.github.io/hydromt_sfincs/
- For more information regarding HydroMT in general see: https://deltares.github.io/hydromt/
- For more user flexibility, it is also possible now to access individual setup components to build your own model or forcing from scratch, see: https://deltares.github.io/hydromt_sfincs/latest/user_guide/sfincs.html
- For an example of building a model from scratch in Python see: https://deltares.github.io/hydromt_sfincs/latest/_examples/build_from_py.html

1.4 Executable

We provide pre-compiled versions of SFINCS for you to use directly:

- Windows: https://download.deltares.nl/en/download/sfincs/
- Platform independent through Docker (Windows/linux/singularity/HPC): https://hub.docker.com/r/deltares/sfincs-cpu

1.5 Compiling yourself

If you want to compile SFINCS yourself for windows to test certain improvements, you need Visual Studio and an Intel Fortran compiler. Find here at tested and free available combination:

- Visual Studio Community 2022 - freely available from: https://visualstudio.microsoft.com/vs/community/
- Intel Fortran Compiler Classic and Intel Fortran Compiler for Windows* 2022.1.0 - freely available from: https://www.intel.com/content/www/us/en/developer/articles/tool/oneapi-standalone-components.html

After installation and configuring (see respective websites for help), you can compile your own SFINCS model locally and add features. Make a checkout of the SFINCS github page, and open the VS solution file to do this: https://github.com/Deltares/SFINCS/blob/main/source/sfincs.sln

To collaborate together and under the GNU GPL-v3.0 license SFINCS comes with, share any improvements with us as pull request on Github: https://github.com/Deltares/SFINCS/pulls

1.6 Running SFINCS

SFINCS can be run on multiple different platforms, both local, HPC and cloud based. The simplest way is to run SFINCS on Windows using a batch-file.

1.6.1 On windows (standard)

The standard method to run SFINCS locally is on a windows machine using a batch-file. This batch file you copy to the folder where your input files to be used by SFINCS are located. The batch file simply calls the executable (add the right path to the folder where sfincs.exe is located) and the general output text file is written to a new text file called 'sfincs_log.txt', see below for an example.

Using batch-file

run.bat

```
make a text file called 'run.bat' and add here:

       call "c:\..\folder_where_exe_is_located\sfincs.exe">sfincs_log.txt
```

1.6.2 On linux

Dedicated linux compiled version

Generally for Linux (HPC) systems, running using Docker or Singularity is the most generic and succesfull way. In case you need a dedicated Linux build, get in touch and we can create a version specifically for you. We have experience doing this for our own Deltares cluster.

1.6.3 Using Docker

For always using the last build version of SFINCS on Windows, Mac, Linux or a cloud based cluster a convenient solution is running a Docker container version of SFINCS. This can be done on a local desktop or in a cloud based cluster supporting docker (or using singularity, see below).

Note that this Docker version of SFINCS is available under the same GNU GPL-v3 License as the windows executable.

Local desktop version

After downloading Docker desktop for your operating system (https://www.docker.com/products/docker-desktop), you can run a model using:

Example

```
docker pull deltares/sfincs-cpu

docker run -vC:/Users/../SFINCS:/data deltares/sfincs-cpu

(here 'C:/Users/../SFINCS' is the folder where the SFINCS input files to be used are␣
↪located)
```

Instead of using the latest automatically cmpiled version, you can also pull a verified tagged release from: https://hub.docker.com/r/deltares/sfincs-cpu/tags This can be for instance the open source release version 'sfincs-v2.0.0-AlpeDHuez'.

Cloud based cluster

The same principle is also possible on a cloud based cluster that supports running docker containers

1.6.4 Using Singularity

On cloud based clusters like Surfsara/Azure/Amazon that **supports singularity**, it is possible to run the Docker container version of SFINCS directly. Depending on the application it could be wise to pull the docker container once and save as new image, after which this image can be run multiple times. This prevents unnesissarily loading the Docker container every time a simulation is performed.

Note that this Docker version of SFINCS is available under the same GNU GPL-v3 License as the windows executable.

Example

```
Pulling and running the docker container immediately:

        singularity run -B$(pwd):/data --nv docker://deltares/sfincs-cpu


First pulling the docker container and creating a singularity image, then running this␣
↪image:

        singularity pull docker://deltares/sfincs-cpu sfincs-cpu.img

        singularity run -B$(pwd):/data sfincs-cpu.img
```

Also here, instead of using the latest automatically cmpiled version, you can also pull a verified tagged release from: https://hub.docker.com/r/deltares/sfincs-cpu/tags This can be for instance the open source release version 'build-v0.0.1-2022-11-16'.

1.7 Courses learning SFINCS

Besides the elaborate information available in this manual, we do offer courses to learn from the experts how to set up a SFINCS model, and the theory and philosophy behind the model:

- In person training during the Delft Software Days on November 16th, 2022, in Delft, the Netherlands: https://softwaredays.deltares.nl/-/compound-flooding-training
- SFINCS trainings during the DSD are planned to be regularly (yearly for now), for an up to date agenda see: https://softwaredays.deltares.nl/welcome
- Short couse on SFINCS during the Coastal Sediments conference April 11th, 2023, in New Orleans, USA: http://coastalsediments.cas.usf.edu/shortcourses.html

If these dates don't suit you or your organisation, or you want a more advanced training; get in touch and we can set up a tailor-made course for you.

1.8 Questions and support

As Deltares is a NGO project-based organisation, and while we do try to answer all your questions, we simply cannot do this to the ultimate end-degree-level as the model is provided to you free of charge.

If you do have more structurally returning questions about how to set up SFINCS models for your application, considering getting a Software Service Package. For more information, send an email to: software@deltares.nl

Additionally, if you want the experts to set up a first working framework of SFINCS models for you, get in touch with product manager tim.leijnse@deltares.nl to discuss options for collaborations to set this up.

1.9 Contributing

1.9.1 Documentation

The code of this documentation is available from https://github.com/Deltares/SFINCS/docs. Get in touch if you have suggestions how to improve this manual, or put in a pull request with improvements yourself: https://github.com/Deltares/SFINCS/pulls

1.9.2 Code

The SFINCS code is open source as of 16-11-2022, see: https://github.com/Deltares/SFINCS/source

Get in touch if you would like to join us in developing the SFINCS code, or put in a pull request on Github with improvements yourself: https://github.com/Deltares/SFINCS/pulls

1.10 User manual - general

1.10.1 Overview

The input for SFINCS is supplied using various text and binary files, which are linked through the main input file: sfincs.inp. Within this section of the input description all major input settings and files are discussed. The figure below gives an overview of all different types of input files and whether they are required or not. Below an example is given of this file, which uses a keyword/value layout. For more information regarding specific parameters see the pages 'Input parameters' or 'Output parameters'.

NOTE - In the manual below, blocks named 'Matlab example using OET' are included, referring to easy setup scripts included in the SFINCS' Open Earth Tools Matlab set of scripts: https://svn.oss.deltares.nl/repos/openearthtools/trunk/matlab/applications/sfincs

1.10.2 Example of sfincs.inp

```
x0              = 0
y0              = 0
mmax            = 100
nmax            = 100
dx              = 100
dy              = 100
rotation        = 0
```

(continues on next page)

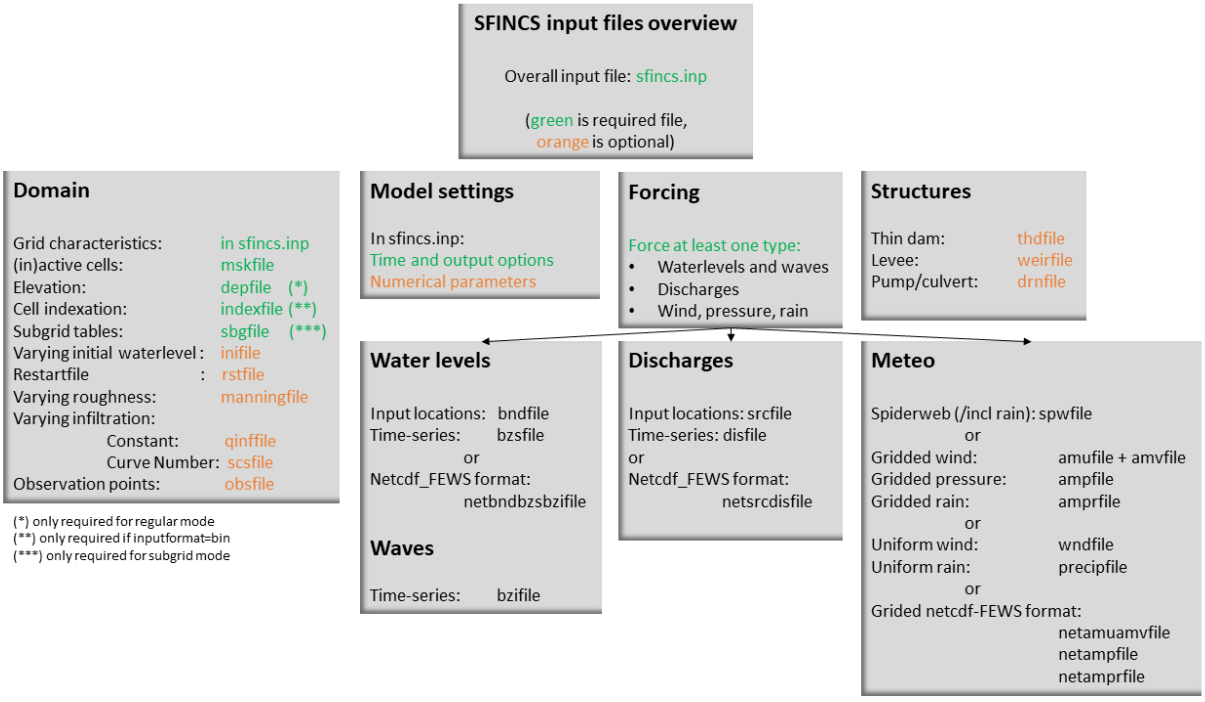


Fig. 1.12: Overview of input file of SFINCS with indication whther they are required or not

(continued from previous page)

```
tref            = 20221116 000000
tstart          = 20221116 180000
tstop           = 20221116 235959

depfile         = sfincs.dep
mskfile         = sfincs.msk
indexfile       = sfincs.ind

bndfile         = sfincs.bnd
bzsfile         = sfincs.bzs
spwfile         = sfincs.spw
srcfile         = sfincs.src
disfile         = sfincs.dis

advection           = 0
alpha           = 0.75
huthresh            = 0.05
manning         = 0.04
theta               = 1.0
qinf            = 0.0

dtout           = 3600
dtmaxout        = 86400
dthisout        = 600
```

(continues on next page)

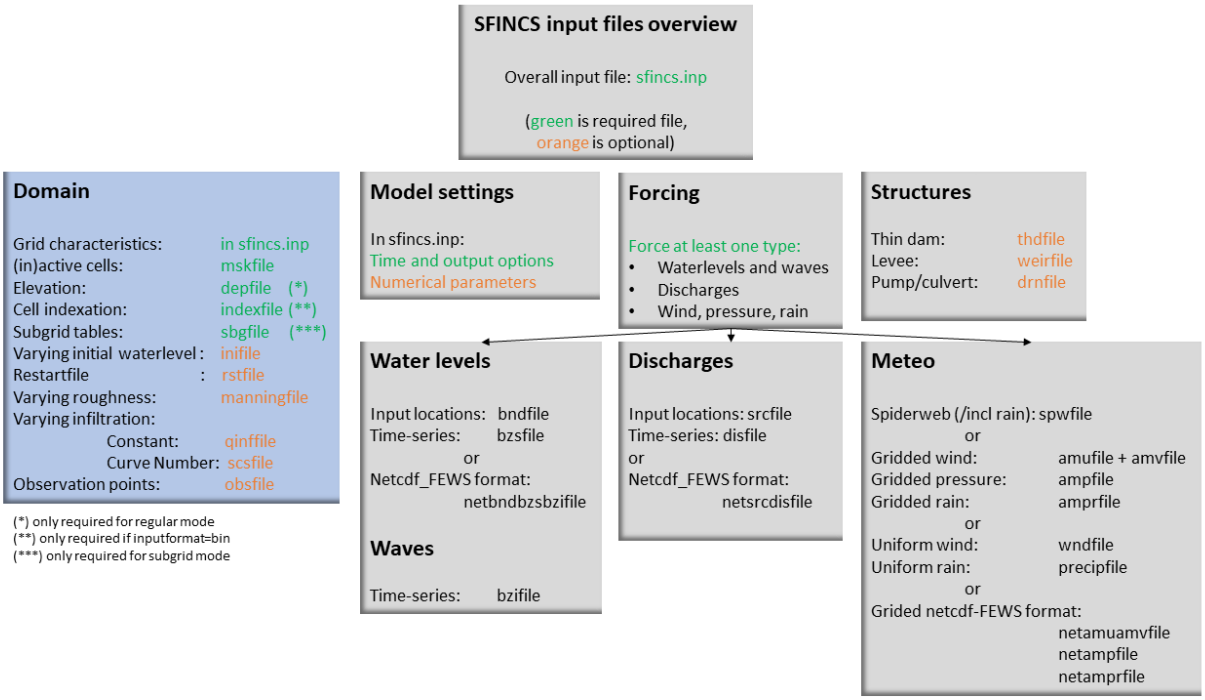
(continued from previous page)

```
inputformat     = bin
outputformat    = net

obsfile         = sfincs.obs
```

1.10.3 Domain

To set up a SFINCS model a number of parameters and files need to be specified to define the domain of the location where a model is being set up for, see the figure below. This consists of parameters of the grid characteristics in the main sfincs.inp-file and multiple separate input files. Some of these are required (elevation, active cells, indexfile in case of binary files) and others are optional (roughness, infiltration, subgrid tables, observation points).



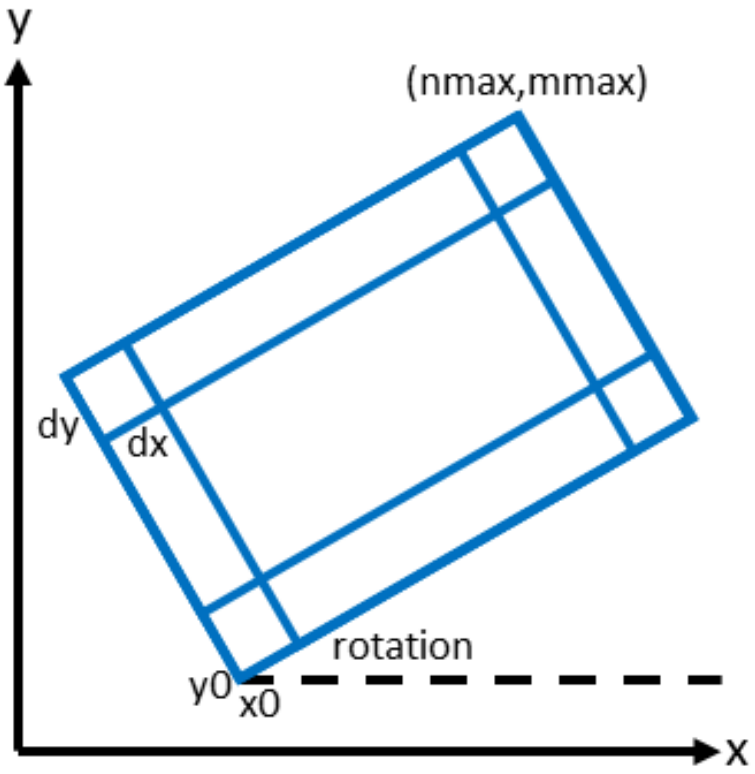
Grid characteristics

SFINCS uses a staggered equidistant recti-linear grid, grid sizes for x- a y-direction can be different. SFINCS can only be used in cartesian coordinates (e.g. UTM zone). The grid is initialised by stating an origin location of the cell edges (x0, y0), a number of grid cells in x-&y-direction (mmax, nmax) and the grid sizes in x-&y-direction (dx,dy). If desired the grid can also be rotated using 'rotation', in degrees from the x-axis (east) in anti-clockwise direction.

```
e.g. in sfincs.inp:

x0              = 0
y0              = 0
mmax            = 250
nmax            = 150
dx              = 100
```

(continues on next page)



(continued from previous page)

```
dy              = 100
rotation        = 45
```

Matlab example using OET

```
inp = sfincs_initialize_input;

inp.x0              = 1000
inp.y0              = 2000

sfincs_write_input('sfincs.inp',inp)
```

Depth file

To describe the local topography and bathymetry, elevation data has be supplied to the model. This can be of any multiple of sources, but it is advised that the transition zone between different datasets and between above/below water level are checked with care. The elevation is described in the cell centres of the grid.

The elevation is defined in sfincs.dep based on the specified grid, positive is upwards with respect to a certain reference level (topography has positive values, bathymetry has negative values). The reference level is not known to SFINCS (and not relevant for the computation), so a user itself must be consistent in the use of specifying elevations in different files (elevation, initial water level, boundary conditions) always to the same vertical reference level (whether it is local MSL, NAP, EGM96 etc. etc.).

NOTE - The depfile is not used when running SFINCS in the subgrid mode (see below)

depfile = sfincs.dep

```
<zb x0,y0> <zb x1,y0>

<zb x0,y1> <zb x1,y1>

e.g.
2.0     2.2
1.8     2.4
```

Matlab example using OET

```
z = 5 * ones(nmax,mmax);
msk = ones(nmax,mmax);

% inputformat = bin:
sfincs_write_binary_inputs(z,msk,inp.indexfile,inp.depfile,inp.mskfile)

% inputformat = asc:
sfincs_write_ascii_inputs(z,msk,inp.depfile,inp.mskfile)
```

Mask file

To distinguish active from inactive areas and cells where boundary conditions need to be forced, a mask file needs to be supplied. This mask indicates for every cell whether it is an inactive cell (msk=0), active cell (msk=1), boundary cell (msk=2) or outflow boundary cell msk=3). This allows great flexibility in optimising the model domain and thereby reducing the computational runtime as much as possible.

If boundary water levels are supplied, these are only forced to the cells with a value of 2. Cells with a value of 0 are inactive and no fluxes from/to these cells are calculated. The file can be made with the OET script 'sfincs_make_mask.m', whereby default a value of -2 m to MSL is used to distinguish the cells.

mskfile = sfincs.msk

```
<msk (x0,y0)> <msk (x1,y0)>

<msk (x0,y1)> <msk (x1,y1)>

e.g.
0       1
2       3
```

Matlab example using OET

```
z = 5 * ones(nmax,mmax);
msk = ones(nmax,mmax);

% inputformat = bin:
sfincs_write_binary_inputs(z,msk,inp.indexfile,inp.depfile,inp.mskfile)

% inputformat = asc:
sfincs_write_ascii_inputs(z,msk,inp.depfile,inp.mskfile)
```

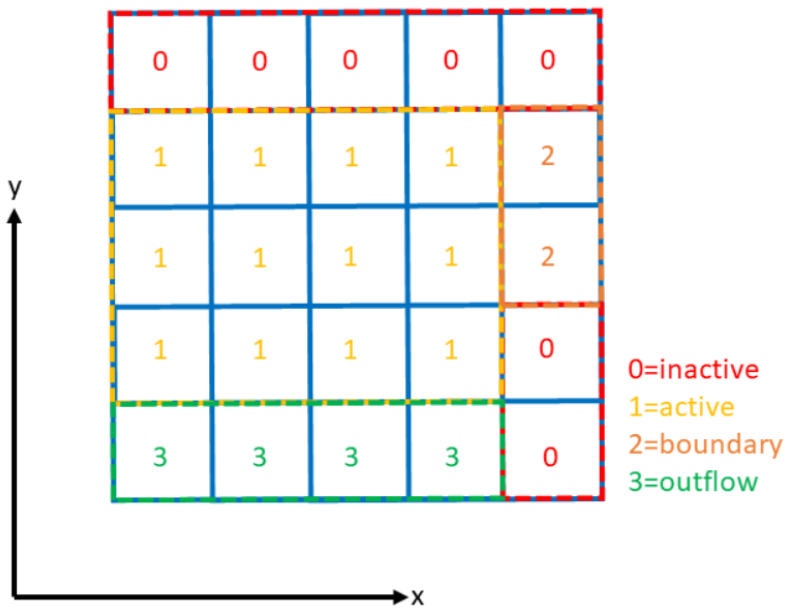


Fig. 1.13: Example of the different mask values on a grid.

Index file

Additionally a index file is needed when supplying binary input files (inputformat = bin). This file is automatically generated when using the Matlab script sfincs_write_binary_inputs as in the example above.

indexfile = sfincs.ind

```
<cell number 1> <cell number 2> <cell number 3>
```

Subgrid tables

Currently the SFINCS model functionality has been extended so that SFINCS can also calculate flooding with the use of subgrid tables. Hereby high-resolution elevation data is used to derive relations between the water level and the volume in a cell to do the continuity update, and a representative water depth used to calculate momentum fluxes. The derivation of these subgrid tables is a pre-processing step outside of the model, that only needs to be done once! The advantage of the subgrid version of SFINCS is that generally one can compute on coarsed grid sizes, while still having accurate results utilizing the high-resolution elevation data to its full potential.

Making subgrid tables is an advanced option that is only shown here as example using Matlab OET using the function 'sfincs_build_model.m'. Get in touch with us to discuss the best solution to make subgrid features for your application (e.g. in Python).

The easiest way to make a SFINCS model using the subgrid functionality, is by using a recent version of the Delft Dashboard GUI.

After supplying a subgrid file so SFINCS (sbgfile = sfincs.sbg), SFINCS will automatically run in subgrid mode!

Matlab example using OET

> For more information see: https://svn.oss.deltares.nl/repos/openearthtools/trunk/matlab/applications/sfincs/sfincs_modelsetup/sfincs_build_model.m

```
inp = sfincs_initialize_input;

inp.sbgfile = sfincs.sbg;

> change wanted grid settings dx/dy/mmax/nmax etc.
inp.dx = 200;

> define output folder; folder = 'c:\test\'

> define bathymetry source from Delft Dashboard data source selection, e.g.;
bathy(1).name = 'ngdc_crm';
bathy(1).zmin = 0;
bathy(1).zmax = 10000;
bathy(1).vertical_offset = 0;

bathy(2).name = 'ngdc_crm';
bathy(2).zmin = -10000;
bathy(2).zmax = 0;
bathy(2).vertical_offset = 0;

The first defined bathymetric source is always used first (given elevation constraints␣
↪of 'zmin' and 'zmax'), after which the other sources are used in given hierarchy.
For types of possible data sources or adding your own one, see the Delft Dashboard␣
↪manual: https://publicwiki.deltares.nl/display/DDB/Delft+Dashboard
This elevation data is used to make the subgrid tables including relations between water␣
↪level and volume within a cell (among others) using detailed elevation data on subgrid␣
↪resolution.
By default the subgrid cells are set to be 10 times smaller than the input dx/dy.

> define wanted project coordinate reference system (UTM zone), e.g.;
cs.name = 'WGS 84 / UTM zone 17N';
cs.type = 'projected';

sfincs_build_model(inp,folder,bathy,cs)
```

Friction

Different roughness values can great impact modelled flooding and thereby SFINCS allows the specification of a uniform value, differentiating land and sea with 2 different values or specifying a specific value per grid cell.

Friction is specified with a Manning roughness coefficient ‘n’ [s/m^{1/3}] and can be done spatially uniform, land/sea value based or spatially varying.

The following options are **ONLY** relavant for the **regular** version of SFINCS, in the subgrid version of SFINCS roughness is already included in the subgrid sbgfile and supplied additional keywords and files will **NOT** be used!

Spatially uniform:

Specify the keyword:

```
manning = 0.04 (default)
```

Land/sea value:

For spatially varying a reference level in meters 'rgh_lev_land' is used to distinguish land 'manning_land' (elevation > rgh_lev_land) and sea 'manning_sea' (elevation < rgh_lev_land) with different friction values.

```
rgh_lev_land = 0 (default)

manning_land = 0.04

manning_Sea = 0.02
```

Spatially varying:

For spatially varying friction values per cell use the manningfile option, with the same grid based input as the depfile using a binary file.

manningfile = sfincs.man

```
<manning x0,y0> <manning x1,y0>

<manning x0,y1> <manning x1,y1>

e.g.
0.02    0.02
0.06    0.04
```

Matlab example using OET

```
inp.manningfile = 'sfincs.man';

manning = 0.02 * ones(nmax,mmax);
msk = ones(nmax,mmax);

sfincs_write_binary_inputs(manning,msk,inp.indexfile,inp.manningfile,inp.mskfile)
```

Infiltration

Infiltration can significantly alter the amount of flooding when including precipitation. SFINCS allows the specification of a uniform constant value, spatially varying constant value or the Curve Number method. The Curve Number is a generally used method to determine what parts of falling rainfall can infiltrate or will run-off, hereby a limited time component is taken into account as well.

Infiltration is specified with either constant in time values in mm/hr (both uniform and spatially varying), or using a Curve Number method (only spatially varying).

NOTE - Infiltration in SFINCS is only turned on when any rainfall is forced'

NOTE - Infiltration methods in SFINCS are not designed to be stacked

Spatially uniform constant in time:

Specify the keyword:

```
qinf = 1.0
```

Spatially varying constant in time:

For spatially varying infiltration values per cell use the qinffile option, with the same grid based input as the depfile using a binary file.

qinffile = sfincs.qinf

```
<infiltrationrate x0,y0> <infiltrationrate x1,y0>

<infiltrationrate x0,y1> <infiltrationrate x1,y1>

e.g.
1.0     5.0
0.0     6.0
```

Matlab example using OET

```
inp.qinffile = 'sfincs.qinf';

infiltration = 2.2 * ones(nmax,mmax);
msk = ones(nmax,mmax);

sfincs_write_binary_inputs(infiltration,msk,inp.indexfile,inp.qinffile,inp.mskfile)
```

Spatially varying Curve Number:

For spatially varying infiltration values per cell using the Curve Number method use the scsfile option, with the same grid based input as the depfile using a binary file. Note here that in pre-processing the wanted CN values should be converted to S values following:

S = (1000./CN - 10)

There is also an advanced version of the Curve Number method that also allows for the recharge of soil capacity after time progresses. Get it touch if you would like to know more information.

scsfile = sfincs.scs

```
<curve_number_value x0,y0> <curve_number_value x1,y0>

<curve_number_value x0,y1> <curve_number_value x1,y1>

e.g.
0       10
5       20
```

Matlab example using OET

```
CN_values = 50 * ones(nmax,mmax);
S_values = (1000./CN - 10)
msk = ones(nmax,mmax);

sfincs_write_binary_inputs(S_values,msk,inp.indexfile,inp.scsfile,inp.mskfile)
```

Observation points

Observation points with water depth and water level output can be specified. Per observation point as minimal the x-and y- coordinates are stated, an standard name will then be added per point. Also, names of a station can be provided with quotes '' (maximum of 256 characters):

obsfile = sfincs.obs

```
<obs1 x1> <obs1 y1>  <obs1 'name1'>

<obs2 x2> <obs2 y2>  <obs2 'name2'>

e.g.:
592727.98 2969420.51 'NOAA_8722548_PGABoulevardBridge,PalmBeach'
594279.00 2961312.47 'NOAA_8722588_PortofWestPalmBeach'
595006.75 2944069.38 'NOAA_8722669_LakeWorthICW'
```

Matlab example using OET

```
inp.obsfile = 'sfincs.obs';
obs.x = [592727.98, 594279.00, 595006.75];
obs.y = [2969420.51, 2961312.47, 2944069.38];
obs.names = {'NOAA_8722548_PGABoulevardBridge,PalmBeach', 'NOAA_8722588_
↪PortofWestPalmBeach', 'NOAA_8722669_LakeWorthICW'};

sfincs_write_obsfile(inp.obsfile,obs)
```

Cross-sections for discharge output

Cross-sections to get the discharge flowing through as output can be specified. SFINCS will keep track of the discharge in m3/s flowing through the specified cross-section(s). Per cross-section as minimal a name, the number of points (size data) and the x-and y- coordinates are stated, using the Delft3D 'tekal' format. You can specify more than 2 points per cross-section.

The output is available as 'crosssection_discharge' in sfincs_his.nc, see the description in "Output description".

crsfile = sfincs.crs

```
NAME1
2 2 %size data
<x0> <y0> %start of polyline 1
<xend> <yend> %end of polyline 1

NAME2
2 2 %size data
```

(continues on next page)

(continued from previous page)

```
<x0> <y0> %start of polyline 2
<xend> <yend>  %end of polyline 1

e.g.

CRS01
3 2
0 100
10 100
20 100
CRS02
2 2
20 200
25 200
```

Matlab example using OET

```
inp.crsfile = 'sfincs.crs';

cross_sections(1).x = [0 10 20];
cross_sections(1).y = [100 100 100];
cross_sections(1).name = {'CRS01'};
cross_sections(2).x = [20 25];
cross_sections(2).y = [200 200];
cross_sections(2).name = {'CRS02'};
cross_sections.length = length(cross_sections);

sfincs_write_cross_sections(inp.crsfile,cross_sections);
```

Initial water level

The water level is by default initiated at 0 meters above mean water level, but can be changed. In the initialisation phase within the model, all cells with an elevation below specified user value are given the specified value of 'zsini', thereby starting without a completely dry bed. For more flexibility, this can also be prescribed spatially varying which can be relevant for coastal, riverine and tsunami cases. This 'inifile' is now only supported using a **binary** file.

NOTE - In pre-release versions of SFINCS this was an ascii type file

Alternatively, you can specify initial conditions using a restart file, see below:

zsini

```
zsini = 1.0
```

inifile = sfincs.ini

```
<zsini_value x0,y0> <zsini_value x1,y0>

<zsini_value x0,y1> <zsini_value x1,y1>

e.g.
1.0     1.2
0.0     0.0
```

Matlab example

```
inp.inifile = 'sfincs.ini';

zini=zeros(inp.nmax, inp.mmax);
zini(:,1:24+1)=0.6;
sfincs_write_binary_inputs(zini,msk,inp.indexfile,inp.inifile,inp.mskfile)
```

Restart file

In order to run SFINCS without spinup of water levels in a subsequent simulation, the waterlevels at the final time step of the former can be saved as binary-file using either the specification of a specific time you want to have the restartfile for (trstout = XXX seconds after 'tref'), or at a certain time internal (dtrstout = XXX seconds). This restart file can then be specified as 'rstfile' in a subsequent SFINCS run, in order to start the simulation with the final water levels of the prior simulation. This currently entails 'type 1', meaning both the water level, fluxes and mean velocities (1: zs, qx, qy, umean and vmean).

```
> in the first SFINCS simulation specify:

dtrstout        = 86400 (restart output every 1 day)

or:

trstout     = 172800 (restart output 2 days after 'tref')

> in the second SFINCS simulation specify:

rstfile         = ./simulation_01/sfincs.rst
```

1.10.4 Model settings

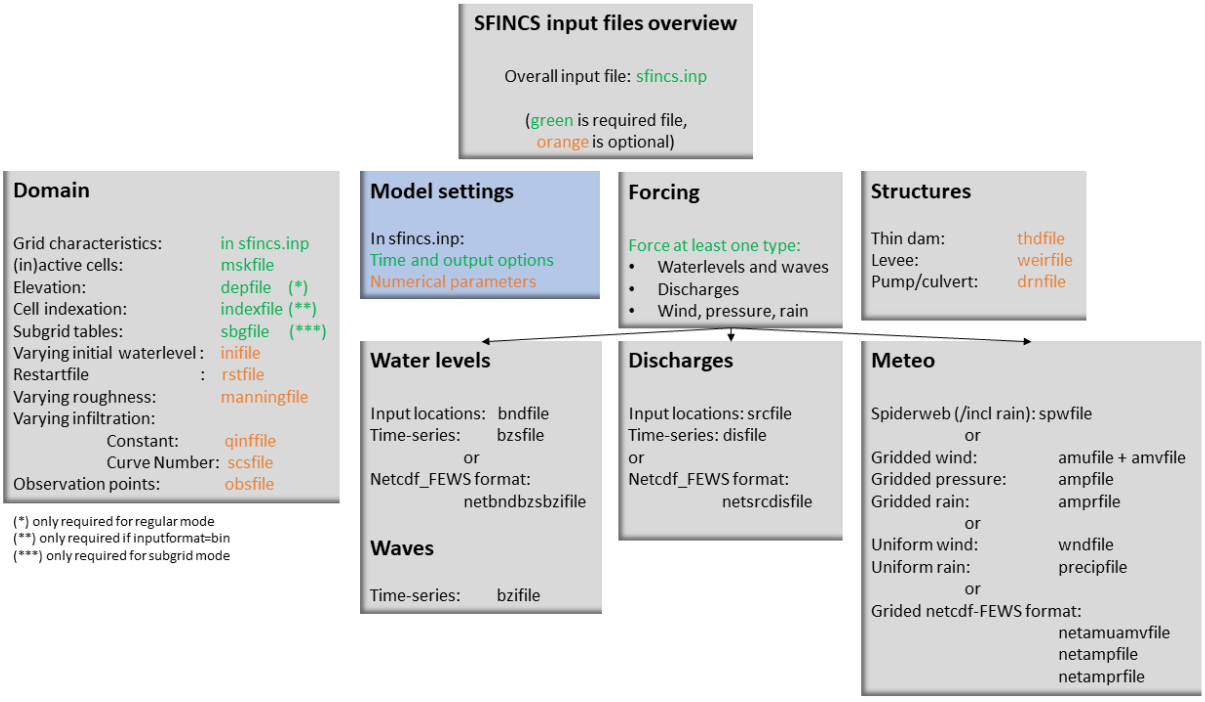
Time management

The required model runtime can be specified by setting a reference date (tref), start date (tstart) and stop date (tstop). The format is 'yyyymmdd HHMMSS', see below:

```
tref                    = yyyymmdd HHMMSS
tstart                  = yyyymmdd HHMMSS
tstop                   = yyyymmdd HHMMSS

e.g.
tref            = 20180000 000000
tstart          = 20180000 000000
tstop           = 20180001 000000
```

Also the output date interval can be controlled. For the map output there is data output every 'dtout' seconds, for optional observation points this is 'dthisout' seconds. It also possible to get the maximum output data over a specific interval (e.g. every day), specify using 'dtmaxout' in seconds. When using a spiderweb-file for the wind input, the values are updated every 'dtwnd' seconds.




```
dtout           = 3600
dtmaxout        = 86400
dthisout        = 600
dtwnd           = 1800
```

Input format

The depth/mask/index-files can be binary or ASCII files. For the former specify 'inputformat = bin' (default), for the latter specify 'inputformat = asc'.

```
inputformat = bin
```

Output format

The main map output can be netcdf, binary or ASCII files. For the former specify 'outputformat = net' (default), for the others specify 'outputformat = bin' or 'outputformat = asc'.

```
outputformat = net
```

Output files

In case of netcdf output the map output will be named 'sfincs_map.nc', in case observation points are provided also a second file will be created with observation point output named 'sfincs_his.nc'.

For more information about the variables saved to the netcdf output files, see the 'Output description' section.

For binary or ascii files the output will be written to separate files, of which the named can be changed:

```
hmaxfile        = hmax.dat
zsfile          = zs.dat
vmaxfile        = vmax.dat
```

Numerical parameters

huthresh

'huthresh' is the flow depth limiter in SFINCS, by default set to 0.05 meters, controlling what minimal water depth should be exceeded to call a cell wet, and start calculating fluxes. It is recommended to use values within the range [0.001 <> 0.1].

alpha

'alpha' is the additional time step limiter besides the courant criteria. By default this is set to 0.75, in case model simulations become instable for some reason this value can be reduced. It is recommended to use values within the range [0.1 <> 0.75].

theta

'theta' sets the implicitness of the numerical scheme of SFINCS. The default value is 1.0 which is recommended for the regular version of SFINCS, however if more smoothing in you model result is needed because it might become unstable for some reason, you could set this to theta=0.9..

advection

'advection' sets what version of the advection term to use in the momentum equation, varying between the default of no advection at all (advection = 0), 1D advection terms (advection = 1) and full 2D advection terms (advection = 2). Generally it is only needed to turn on advection in case of modelling waves or super-critical flow.

```
huthresh        = 0.05
alpha           = 0.75
theta           = 1.0
advection       = 0
```

viscosity

'viscosity' turns on the viscosity term in the momentum equation (viscosity = 1). The recommended value of viscosity 'nuvisc' to add to your model (only advised to use when you set theta = 1.0), depends on your grid size. For ease, SFINCS internally automatically determines the optimal value for you, which is displayed when running the model: 'Turning on process: Viscosity, with nuvisc= 0.5000000'. In this example corresponding to a grid resolution of 50 meters. In case you would want to increase the viscosity term, you can either specify the exact value you want 'nuvisc = XXX', or e.g. multiply it by a factor 2: nuviscdim = 2.0 (default = 1.0, dimensionless). By default the value of nuvisc is determined like this:

dx = 50 > nuvisc = 0.5

dx = 100 > nuvisc = 1.0

dx = 500 > nuvisc = 5.0

```
viscosity       = 1
nuviscdim       = 1.0 (default)
nuvisc          = XXX (automatically determined, or specify a value yourself that␣
↪overrules this)
```

Drag Coefficients:

The wind drag coefficients are varying with wind speed and implemented as in Delft3D. The default values are based on Vatvani et al. (2012). There is specified for how many points 'cd_nr' a velocity 'cd_wnd' and a drag coefficient 'cd_val' is specified, the following are the default values:

```
cd_nr           = 3

cd_wnd          = 0 28 50

cd_val          = 0.0010 0.0025 0.0015
```

1.11 User manual - forcing

1.11.1 Overview

The input for SFINCS is supplied using various text and binary files, which are linked through the main input file: sfincs.inp. Within this section of the user manual all different types of forcing settings and files are discussed. The figure below gives an overview of all different types of input files and whether they are required or not. Below an example is given of this file, which uses a keyword/value layout. For more information regarding specific parameters see the pages 'Input parameters' or 'Output parameters'.

NOTE - In the manual below, blocks named 'Matlab example using OET' are included, referring to easy setup scripts included in the SFINCS' Open Earth Tools Matlab set of scripts: https://svn.oss.deltares.nl/repos/openearthtools/trunk/matlab/applications/sfincs

Forcing

SFINCS has different functionalities regarding different relevant physical processes for compound flooding and what type of model is required. At first nearshore/offshore water levels can be specified at the different locations along the coast to include tides and storm surge levels. Inland drivers of flooding like precipitation and wind can be specified in a number of ways. This varies from simple spatially uniform time-series to spatially varying spiderwebs or grid input types. Furthermore, simple implementations for discharges are included.

Discussed in this user manual are the water-level boundaries, discharge points, wind, pressure, rain and waves.

1.11.2 Water levels

To specify water-level time-series to the boundary cells (msk=2), first the input locations have to be specified in 'sfincs.bnd'. For every boundary point there is interpolated with a weighted average between the two closest input locations.

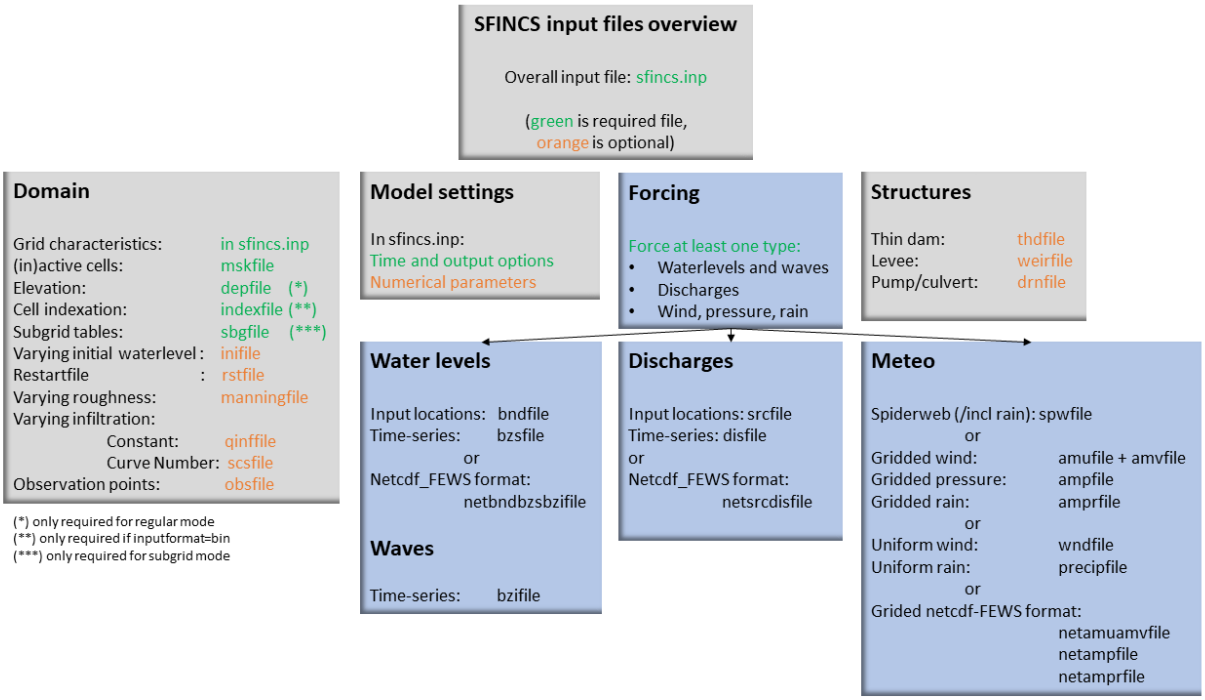


Fig. 1.14: Overview of input file of SFINCS with indication whther they are required or not

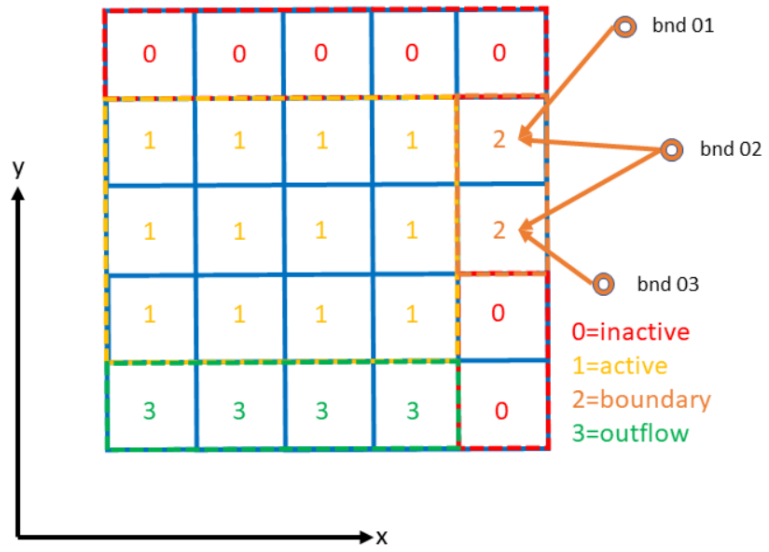


Fig. 1.15: Example of the weighted interpolation of boundary input locations to the msk=2 boundary cells.

Water level points

bndfile = sfincs.bnd

```
<bnd1 x1> <bnd1 y1>

<bnd2 x2> <bnd2 y2>

e.g.
400000  1200000
480000  1250000
```

Matlab example using OET

```
points.x(1) = 400000;
points.y(1) = 1200000;
points.x(2) = 480000;
points.y(2) = 1250000;

sfincs_write_boundary_points(inp.bndfile,points)
```

Water level time-series

Then in the file 'sfincs.bzs' the water level time-series are specified per input location. Times are specified in seconds with respect to SFINCS' internal reference time 'tref', as specified in sfincs.inp.

NOTE - The time and length of values you prescibe per boundary input location should be all the same in the bzsfile

bzsfile = sfincs.bzs

```
<time 1> <zs1 bnd1> <zs1 bnd2>

<time 2> <zs2 bnd1> <zs2 bnd2>

e.g.
0       0.50    0.75
3600    0.60    0.80
7200    0.45    0.85
```

Matlab example using OET

```
time = [0, 3600, 7200];
waterlevels = [0.5, 0.75; 0.6, 0.8; 0.45, 0.85];

sfincs_write_boundary_conditions(inp.bzsfile,time,waterlevels)
```

1.11.3 Waves

When forcing waves, besides providing a bzsfile with slowly varying water level time-series, also the same type of file with the quickly varying water level component due to waves can be prescribed. This can contain infragravity and/or short waves. Do note that the forced signal should be the incoming wave component only, not including the reflecting one, since this is computed by SFINCS internally as well. The signal should be around 0.

NOTE - Specified time should be the same in both the bzs and bzi files

bzifile = sfincs.bzi

```
<time 1> <zi1 bnd1> <zi1 bnd2>

<time 2> <zi2 bnd1> <zi2 bnd2>

e.g.
0       0.05    0.07
2       -0.02   -0.04
4       0.10    0.03
```

Netcdf format input

As alternative, the bnd/bzs/bzi data can also be specified using a single Netcdf file with FEWS input type format 'netbndbzsbzifile'. Making this format netcdf file can be easily done using the OET Matlab script 'sfincs_write_netcdf_bndbzsbzifile.m'

Matlab example using OET

```
inp.netbndbzsbzifile = 'sfincs_netbndbzsbzifile.nc';

x = [0, 100, 200];
y = [50, 150, 250];

EPSGcode = 32631;
UTMname = 'UTM31N';

refdate  = '1970-01-01 00:00:00';
% possibly use formatOut = 'yyyy-mm-dd HH:MM:SS'; datestr(tref, formatOut);

time = [0, 60];

rng('default');
bzs = -1 * randi([0 10],length(time),length(x));
bzi = -1 * randi([0 10],length(time),length(x));

sfincs_write_netcdf_bndbzsbzifile(inp.netbndbzsbzifile, x, y, EPSGcode, UTMname, refdate,
↪ time, bzs, bzi)
```

1.11.4 Discharges

A simple implementation of discharge points is added to SFINCS, specify values in m^3/s. First specify the locations in 'sfincs.src' and then the discharge time-series in 'sfincs.dis'. Alternatively, you can provide this as netcdf file in the Delft-FEWS format.

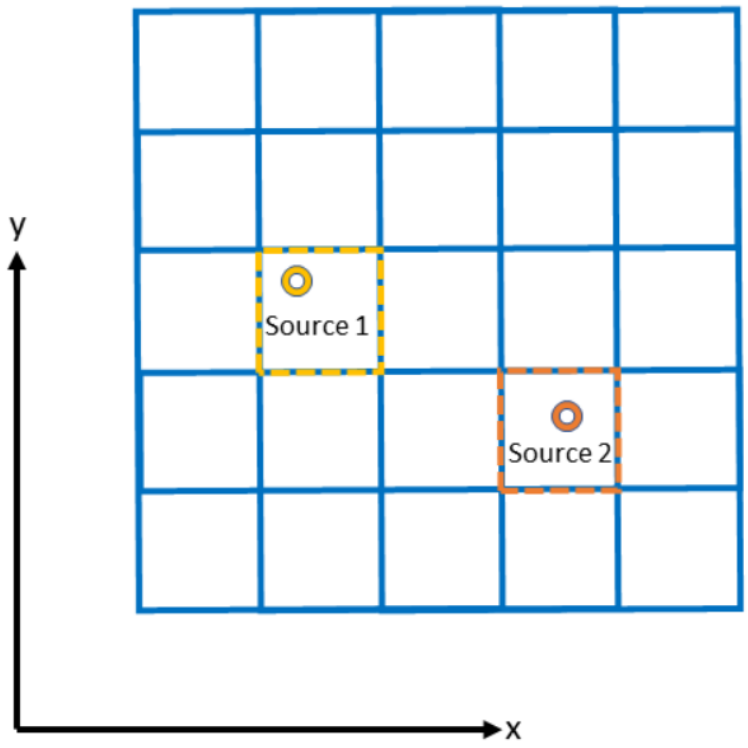


Fig. 1.16: Example of how discharge input points from 2 different sources are snapped to the grid of SFINCS.

Discharge points

srcfile = sfincs.src

```
<src1 x1> <src1 y1>

<src2 x2> <src2 y2>

e.g.
300000  1500000
380000  1650000
```

Matlab example using OET

```
points.x(1) = 300000;
points.y(1) = 1500000;
points.x(2) = 380000;
points.y(2) = 1650000;
```

(continues on next page)

(continued from previous page)

```
sfincs_write_boundary_points(inp.srcfile,points)
```

Discharge time-series

Then in the file 'sfincs.dis' the discharge time-series are specified per input location. Times are specified in seconds with respect to SFINCS' internal reference time 'tref', as specified in sfincs.inp.

disfile = sfincs.dis

```
<time 1> <dis1 src1> <dis1 src2>

<time 2> <dis2 src1> <dis2 src2>

e.g.
0       100     1000
3600    300     1100
7200    0       1300
```

Matlab example using OET

```
time = [0, 3600, 7200];
discharge = [100, 1000; 300, 1100; 0, 1300];

sfincs_write_boundary_conditions(inp.disfile,time,discharge)
```

Netcdf format input

As alternative, the src/dis data can also be specified using a single Netcdf file with FEWS input type format 'netsrcdisfile'. SFINCS assumes that the input variables 'x', 'y', 'time', 'discharge' and 'stations' are available in the netcdf file, including a reference time as UNIT in variable 'time' of the Fews time format: "minutes since 1970-01-01 00:00:00.0 +0000"

Matlab example using OET

```
inp.netsrcdisfile = 'sfincs_netsrcdisfile.nc';

x = [0, 100, 200];
y = [50, 150, 250];

EPSGcode = 32631;
UTMname = 'UTM31N';

refdate  = '1970-01-01 00:00:00';
% possibly use formatOut = 'yyyy-mm-dd HH:MM:SS'; datestr(tref, formatOut);

time = [0, 60];

rng('default');
dis = -1 * randi([0 10],length(time),length(x));
```

(continues on next page)

(continued from previous page)

```
sfincs_write_netcdf_srcdisfile(inp.netsrcdisfile, x, y, EPSGcode, UTMname, refdate, time,
↪ dis)
```

1.11.5 Meteo

There are a few different options to specify wind and rain input:

1) Use a spatially varying spiderweb input (as in Delft3D/Delft3D FM) for forcing tropical cyclones only the wind and pressure input, or for the wind as well as the rain input.
2) Use a spatially varying grid input (as in Delft3D) for u- and v- wind velocities and/or the rain and/or pressure input.
3) Use a spatially varying grid input using a netcdf file based on a FEWS input type format for wind, rain and/or atmospheric pressure input.
4) Use a spatially uniform input for wind and rain, which is faster but also more simplified.
5) Make a combination, for instance use a spiderweb for the wind input and a spatially uniform rain-input. When combining, test whether the forcing is as wanted since not all combinations of the above options might be possible and/or changing depending on specific code version.

You can know how much rainfall / wind is added to the model in the output by specifying 'storecumprcp=1' and/or 'storemeteo=1', see the description in "Input parameters".

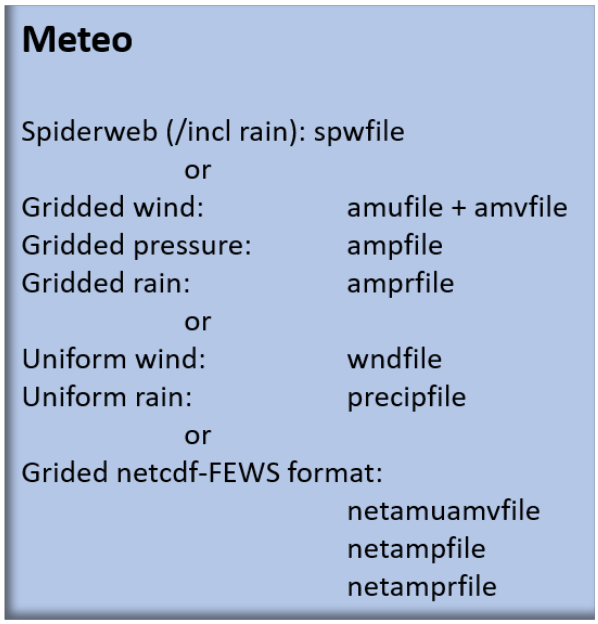


Fig. 1.17: Overview of possible meteo input file options and names

Spatially varying spiderweb

The option of forcing spiderweb files is only relevant for tropical cyclones, best is to put grid units in the same projected coordinate reference system (UTM zone) as the SFINCS grid. For generation of these spiderweb files use Deltares' Wind Enhancement Scheme tool (WES, see https://content.oss.deltares.nl/delft3d/manuals/Delft3D-WES_User_Manual.pdf or OET Matlab equivalent) or get in touch.

Spiderweb-input:

```
spwfile = tropical_cyclone.spw
```

Spatially varying gridded

Spatially varying meteo input on constant grid can be forced using the native Delft3D type meteo input files, using the same file conventions. For wind this is wind in x-&y-direction (amu, amv), precipitation (ampr) and atmospheric pressure (amp). The grid has a constant resolution dx&dy, which can be in the native (usually coarser than your SFINCS grid) resolution of the meteo data. Within SFINCS this is interpolated onto the actual SFINCS grid.

Wind:

```
**amufile = sfincs.amu**

within amufile:

quantity1        = x_wind
unit1            = m s-1

**amvfile = sfincs.amv**

within amvfile:

quantity1        = y_wind
unit1            = m s-1
```

Rain:

```
**amprfile = sfincs.ampr**

within amprfile:

quantity1        = precipitation
unit1            = mm/hr
```

Atmospheric pressure:

```
**ampfile = sfincs.amp**

within ampfile:

quantity1        = air_pressure
unit1            = Pa
```

Delft3D-meteo ascii type input:

These files have this general header of **13 lines** which SFINCS expects (**Check this after creating your input files!**), after which the TIME and data blocks are given per time frame. Only use 1 quantity per file:

```
FileVersion      = 1.03
filetype         = meteo_on_equidistant_grid
n_cols           = 2
n_rows           = 4
grid_unit        = m
x_llcorner       = 417328
y_llcorner       = 3495537
dx               = 5000
dy               = 5000
n_quantity       = 1
quantity1        = x_wind
unit1            = m s-1
NODATA_value     = -999
TIME = 90831.0 hours since 1970-01-01 00:00:00 +00:00  # 1980-05-12 15:00:00
0 0 0 0
0 0 0 0
TIME = 90831.0 hours since 1970-01-01 00:00:00 +00:00  # 1980-05-12 15:00:00
0 0 0 0
0 0 0 0
```

Matlab example using OET

```
data.parameter.time = datenum(2018,01,01):3/24:datenum(2018,01,02);
data.parameter.x = 0:5000:25000;
data.parameter.y = 10000:5000:40000;

data.parameter.val = ones(length(data.parameter.time), length(data.parameter.y),␣
↪length(data.parameter.x));

write_meteo_file_delft3d(inp.amufile, data, 'x_wind', 'm s-1', datenum(1970,01,01),␣
↪varargin);

see 'write_meteo_file_delft3d.m' for more information.
```

Spatially varying gridded netcdf

The same spatially varying gridded input as using Delft3d' ascii input files can be specified using FEWS compatible Netcdf input files. Here for the wind the amu&amv files are combined into 1 Netcdf file (netamuamvfile), the precipitation is in a separate input file (netamprfile) as well as the atmospheric pressure (netampfile).

Making this format netcdf file can be easily done using the OET Matlab scripts 'sfincs_write_netcdf_amuamvfile.m', 'sfincs_write_netcdf_amprfile.m'and 'sfincs_write_netcdf_ampfile.m'. See those files for more information.

Matlab example using OET - netamuamvfile

```
inp.netamuamvfile = 'sfincs_netamuamvfile.nc';

x = [0, 100, 200];
y = [50, 150, 250];

EPSGcode = 32631;
```

(continues on next page)

(continued from previous page)

```
UTMname = 'UTM31N';

refdate  = '1970-01-01 00:00:00';
% possibly use formatOut = 'yyyy-mm-dd HH:MM:SS'; datestr(tref, formatOut);

time = [0, 60];

rng('default');
amu = -1 * randi([0 10],length(time),length(y),length(x));
amv = 1 * randi([0 10],length(time),length(y),length(x));

sfincs_write_netcdf_amuamvfile(inp.netamuamvfile, x, y, EPSGcode, UTMname, refdate, time,
↪ amu, amv)
```

Matlab example using OET - netamprfile

```
inp.netamprfile = 'sfincs_netamprfiles.nc';

x = [0, 100, 200];
y = [50, 150, 250];

EPSGcode = 32631;
UTMname = 'UTM31N';

refdate  = '1970-01-01 00:00:00';
% possibly use formatOut = 'yyyy-mm-dd HH:MM:SS'; datestr(tref, formatOut);

time = [0, 60];

rng('default');
ampr = -1 * randi([0 10],length(time),length(y),length(x));

sfincs_write_netcdf_amprfile(inp.netamprfile, x, y, EPSGcode, UTMname, refdate, time,␣
↪ampr)
```

Matlab example using OET - netampfile

```
inp.netampfile = 'sfincs_netampfiles.nc';

x = [0, 100, 200];
y = [50, 150, 250];

EPSGcode = 32631;
UTMname = 'UTM31N';

refdate  = '1970-01-01 00:00:00';
% possibly use formatOut = 'yyyy-mm-dd HH:MM:SS'; datestr(tref, formatOut);

time = [0, 60];

rng('default');
amp = -1 * randi([0 10],length(time),length(y),length(x));
```

(continues on next page)

(continued from previous page)

```
sfincs_write_netcdf_ampfile(inp.netampfile, x, y, EPSGcode, UTMname, refdate, time, amp)
```

Spatially uniform

Spatially uniform wind:

'vmag' is the wind speed in m/s, 'vdir' is the wind direction in nautical from where the wind is coming. The file can be make using OET Matlab script 'sfincs_write_boundary_conditions.m'. Times are specified in seconds with respect to SFINCS' internal reference time 'tref', as specified in sfincs.inp.

wndfile = sfincs.wnd

```
<time 1> <vmag1> <vdir1>

<time 2> <vmag2> <vdir2>

e.g.
0       5       120
3600    15      180
7200    10      165
```

Spatially uniform rain:

Rain input in mm/hr, times are specified in seconds with respect to SFINCS' internal reference time 'tref', as specified in sfincs.inp. The file can be make using OET Matlab script 'sfincs_write_boundary_conditions.m'.

precipfile = sfincs.prcp

```
<time 1> <prcp0>

<time 2> <prcp1>

e.g.
0       0
3600    15
7200    10
```

1.12 User manual - structures

1.12.1 Overview

The input for SFINCS is supplied using various text and binary files, which are linked through the main input file: sfincs.inp. Within this section of the user manual all different types of structures to reduce flood hazards with input settings and files are discussed. The figure below gives an overview of all different types of input files and whether they are required or not. Below an example is given of this file, which uses a keyword/value layout. For more information regarding specific parameters see the pages 'Input parameters' or 'Output parameters'.

NOTE - In the manual below, blocks named 'Matlab example using OET' are included, referring to easy setup scripts included in the SFINCS' Open Earth Tools Matlab set of scripts: https://svn.oss.deltares.nl/repos/openearthtools/trunk/matlab/applications/sfincs

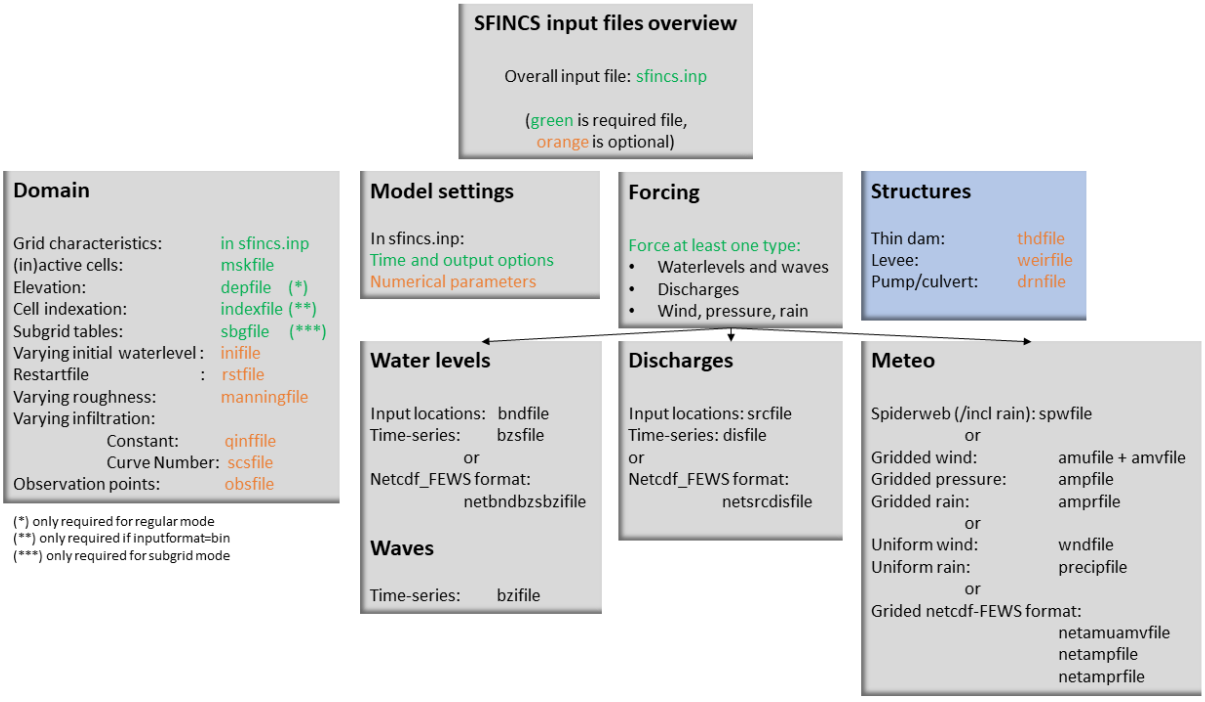


Fig. 1.18: Overview of input file of SFINCS with indication whther they are required or not

1.12.2 Structures

SFINCS consists of multiple options for adding structures that can divert or block flow of water, which can be used to simulate flood hazard reduction methods.

Thin dam

With a thin dam flow through certain grid cells is completely blocked (i.e. an infinitely high wall). One can provide multiple polylines within one file, a maximum of 5000 supplied points is supported. The supplied polylines are snapped onto the SFINCS grid within the model.

thdfile = sfincs.thd

```
NAME1
2 2 %size data
<x0> <y0> %start of polyline 1
<xend> <yend> %end of polyline 1

NAME2
2 2 %size data
<x0> <y0> %start of polyline 2
<xend> <yend>  %end of polyline 1

e.g.

THD01
3 2
```

(continues on next page)

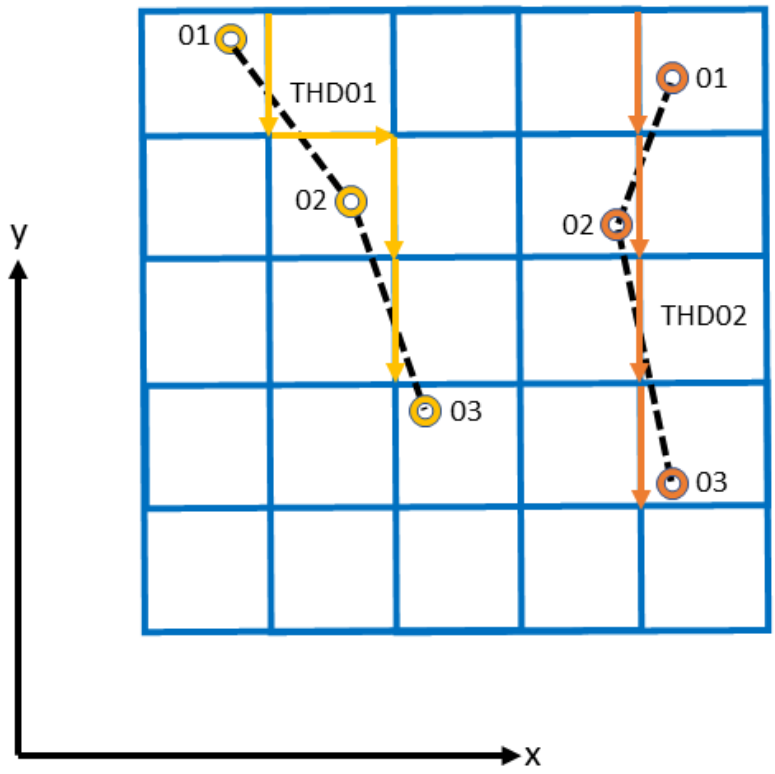


Fig. 1.19: Example of how thin dam/weir input points from 2 different polylines are snapped to the grid of SFINCS.

(continued from previous page)

```
0 100
10 100
20 100
THD02
2 2
20 200
25 200
```

Matlab example using OET

```
inp.thdfile = 'sfincs.thd';

thindams(1).x = [0 10 20];
thindams(1).y = [100 100 100];
thindams(1).name = {'THD01'};
thindams(2).x = [20 25];
thindams(2).y = [200 200];
thindams(2).name = {'THD02'};
thindams.length = length(thindams.x1);

sfincs_write_thin_dams(inp.thdfile,thindams);
```

Weirs

Weirs are in principle the same as a thin dam, but then with a certain height (levee). When the water level on either or both sides of the weir are higher than that of the weir, a flux over the weir is calculated. Hereby a situation where the weir is partly or fully submerged is distinguished. One can provide multiple polylines within one file, a maximum of 5000 supplied points is supported. Besides the x&y locations per points, also the elevation z and a Cd coefficient for the weir formula (recommended to use 0.6). The supplied polylines are snapped onto the SFINCS grid within the model.

The snapped coordinates are available in sfincs_his.nc as structure_x, structure_y & structure_height from SFINCS v2.0.2 onwards.

weirfile = sfincs.weir

```
NAME1
2 4 %size data
<x0> <y0> <z0> <cd1> %start of polyline 1
<x2> <y2> <z2> <cd2> %end of polyline 1

NAME2
2 4 %size data
<x0> <y0> <z0> <cd1> %start of polyline 2
<x2> <y2> <z2> <cd2> %end of polyline 2

e.g.

weir01
3 4
0 100 5.1 0.6
10 100 5.2 0.6
20 100 5.0 0.6
weir02
2 4
20 200 5.1 0.6
25 200 5.1 0.6
```

Matlab example using OET

```
inp.weirfile = 'sfincs.weir';

weirs(1).x = [0 10 20];
weirs(1).y = [100 100 100];
weirs(1).z = [5.1 5.2 5.0];
weirs(1).par1 = [0.6 0.6 0.6];
weirs(2).x = [20 25];
weirs(2).y = [200 200];
weirs(2).z = [5.1 5.2];
weirs(2).par1 = [0.6 0.6];

sfincs_write_obstacle_file_1par(inp.weirfile,weirs)
```

Drainage pump and Culvert

Drainage pumps and culverts are both specified using the same format file, put with a different indication of the type (type=1 is drainage pump, type=2 is culvert). A drainage pump can move water from one location to another with a certain prescribed discharge given that there is sufficient water at the retraction location. For culverts also a certain discharge capacity of the culvert is prescribed, but then the actual water level gradient is used to determine how much water will actually flow through the culvert. Input consists of the x&y locations of the sink (retraction point) and source points (outflow point) followed by the type. The discharge capacity is prescribed using the par1 parameter, parameters par2<>par5 are not used right now but included for future flexibility for implementing other structure types.

You can know how much discharge is extracted by the model in the sfincs_his.nc output by specifying 'storeqdrain=1' from SFINCS v2.0.2 onwards, see the description in "Input parameters".

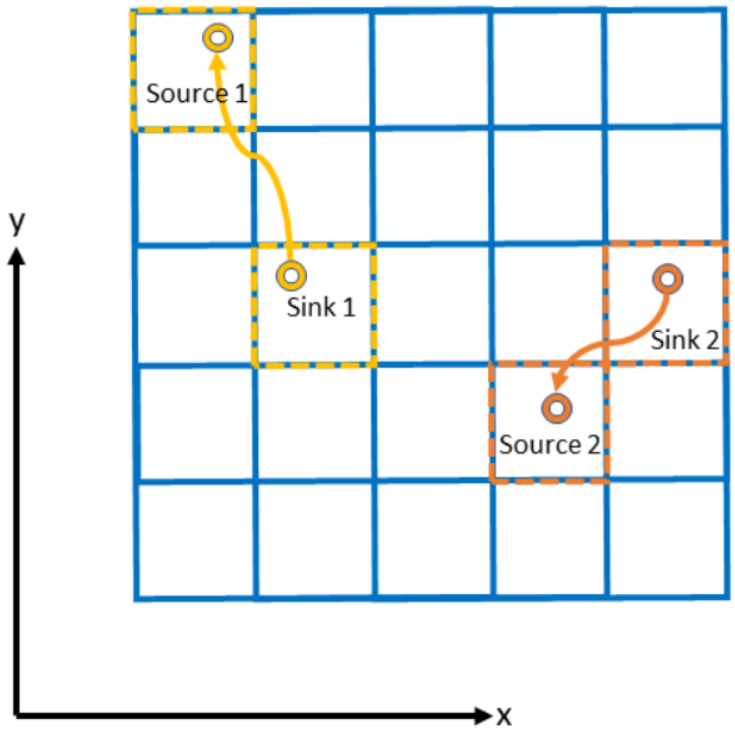


Fig. 1.20: Example of how drainage pump/culvert input points with sink and source locations from 2 different structures are snapped to the grid of SFINCS.

drnfile = sfincs.drn

```
<xsnk1> <ysnk1> <xsrc1> <ysrc1> <type1> <par1-1> par2-1 par3-1 par4-1 par5-1
<xsnk2> <ysnk2> <xsrc2> <ysrc2> <type2> <par1-2> par2-2 par3-2 par4-2 par5-2

e.g. pump:
50.00        25.00       150.00        25.00 1    0.345    0.000    0.000    0.000    0.
↪000
75.00        25.00       125.00        25.00 1    0.345    0.000    0.000    0.000    0.
↪000

e.g. culvert:
50.00        25.00       150.00        25.00 2    0.345    0.000    0.000    0.000    0.
```

(continues on next page)

(continued from previous page)

```
↪000
75.00        25.00       125.00        25.00 2    0.345    0.000    0.000    0.000    0.
↪000
```

Matlab example using OET

```
inp.drnfile = 'sfincs.drn';

jj=1;
drain(jj).xsnk = 75;    % sink x-coordinate(s), from where water is taken
drain(jj).ysnk = 25;    % sink y-coordinate(s)
drain(jj).xsrc = 125;   % source x-coordinate(s), to where water is discharged
drain(jj).ysrc = 25;    % source x-coordinate(s)
drain(jj).type = 1;     % 1= pump, 2=culvert
drain(jj).par1 = 0.345; % possible drainage discharge in m3/s
drain(jj).par2 = 0;     % not used yet
drain(jj).par3 = 0;     % not used yet
drain(jj).par4 = 0;     % not used yet
drain(jj).par5 = 0;     % not used yet

sfincs_write_drainage_file(inp.drnfile,drain)
```

1.13 Input parameters

Different parameters for model input and output of SFINCS can be changed in **sfincs.inp**, see below.

Traditionally SFINCS neglects the advection term in the SFINCS-LIE version ('advection = 0'). For super-critical flow conditions or modelling waves, the SFINCS-SSWE version can be used ('advection = 1' for 1D modelling and 'advection = 2' for 2D modelling) for better performance.

1.13.1 Parameters for model input

mmax

description
Number of grid cells in x-direction

units

-

default
0

min
1

max
Inf (recommended is to limit the total number of active cells to max 3 million)

nmax

description
Number of grid cells in y-direction

units

-

default
0

min
1

max
Inf (recommended is to limit the total number of active cells to max 3 million)

dx

description
Grid size in x-direction

units
m

default
0

min
1.0e-3

max
Inf (recommended is a maximum grid size of 1000 meters)

dy

description
Grid size in y-direction

units
m

default
0

min
1.0e-3

max
Inf (recommended is a maximum grid size of 1000 meters)

x0

description
X-coordinate of first grid cell corner (1,1), thus not center of grid cell, in projected UTM zone.

units
m in projected UTM zone

default
0

min
0

max
Inf

y0

description
Y-coordinate of first grid cell corner (1,1), thus not center of grid cell, in projected UTM zone.

units
m in projected UTM zone

default
0

min
0

max
Inf

rotation

description
Rotation of the grid in degrees from the x-axis (east) in anti-clockwise direction

units
degrees

default
0

min
0

max
359.999

advection

description
setting for advection. 0 for no advection scheme (SFINCS-LIE), 1 for 1D advection scheme for modelling in 1D OR 2 for 2D advection scheme for modelling in 2D (SFINCS-SSWE).

units

-

default
0

min
0

max
2

alpha

description
CFL-condition reduction. Decrease for additional numerical stability, minimum value is 0.1 and maximum is 0.75.

units

-

default
0.5

min
0.1 (recommended)

max
0.75 (recommended)

huthresh

description
Minimum flow depth limiter.

units
m

default
0.05

min
0.001 (recommended)

max
0.1 (recommended)

theta

description
Smoothing factor in momentum equation.

units

•

default
0.9

min
0.8

max
1.0

viscosity

description
Turns on the viscosity term in the momentum equation (viscosity = 1), advised to combine with theta = 1.0. Value of viscosity term 'nuvisc' automatically determined based on grid size..

units

•

default
0

min
0

max
1

nuviscdim

description
Dimensionless viscosity coefficient, multiplies the automatically determined value for 'nuvisc' with the specified factor for 'nuviscdim'.

units

•

default
1.0

min
0.0

max
Inf

nuvisc

description
Viscosity coefficient, by default turned off, but automatically determined if 'viscosity=1'. specifying a value for 'nuvisc' overrule default value.

units

•

default
-999.0 (=off)

min
0.0

max
Inf

zsini

description
Initial water level.

units
m above reference level

default
0

min
-Inf

max
Inf

qinf

description
Infiltration rate, specify in +mm/hr.

units
mm/hr

default
0

min
0

max
100

manning

description
Uniform manning roughness, specify in s/m^(1/3).

units
s/m^(1/3)

default
0.04

min
0

max
0.1 (advised)

rgh_level_land

description
Elevation level to distinguish land and sea roughness (when using 'manning_land' and 'manning_sea').

units
m above reference level

default
0

min
-Inf

max
Inf

manning_land

description
Varying manning roughness based on elevation (above 'rgh_level_land', overules uniform 'manning', specify in s/m^(1/3).

units
s/m^(1/3)

default
-999 (=not used)

min
0

max
0.1 (advised)

manning_sea

description
Varying manning roughness based on elevation (below 'rgh_level_land', overules uniform 'manning', specify in s/m^(1/3).

units
s/m^(1/3)

default
-999 (=not used)

min
0

max
0.1 (advised)

1.13.2 More parameters for model input (only for advanced users)

bndtype

description
Boundary type for interpretation of 'sfincs.bzs' time-series. bndtype=1 is for water levels, old types 2&3 have been removed from SFINCS v2.0.2 onwards.

units

-

default
1

min
1

max
1

rhoa

description
Density of the air

units
kg/m3

default
1.25

min

-

max

-

rhow

description
Density of the water

units
kg/m3

default
1024

min

-

max

-

stopdepth

description
Water depth anywhere in the domain after which the simulation is classified as unstable and stopped

units
m

default
100

min
0

max
Inf

advlim

description
Advection limiter when advection>0 to limit the magnitude of the advection term when calculating fluxes between cells.

units

-

default
9999

min
0

max
9999

dtmax

description
Maximum internal time step to be used

units
s

default
60

min
1.0e-3

max
Inf

dtmin

description
Minimum internal time step to be used

units
s

default
1.0e-3

min
1.0e-3

max
Inf

tspinup

description
Duration of internal spinup period before tstart

units
s

default
60

min
0

max
Inf

Drag coefficients:

cdnrb

description
Number of specified break points

units

•

default
3

min
2

max

•

cdwnd

description
Wind speed break points (including 0)

units

•

default
0 28 50

min
2 values

max

-

description
Drag coefficient brak points

units

-

default
0.001 0.0025 0.0015

min
2 values

max

-

Different parameters influencing the given output by SFINCS can be changed, see below.

1.13.3 Parameters for model output

tref

description
Reference date in 'yyyymmdd HHMMSS'

units

-

default
20000101 000000

tstart

description
Start date in 'yyyymmdd HHMMSS'

units

-

default
20000101 000000

tstop

description
Stop date in 'yyyymmdd HHMMSS'

units
m

default
20000101 000000

trstout

description
Specific time in seconds since 'tref' for binary restart file output being written away, turned of by default.

units
s

default
-999.0

dtout

description
Time-step global map output.

units
s

default
0

dthisout

description
Time-step observation points output.

units
s

default
600

dtmaxout

description
Time-step interval of global map output of maximum water level. If not specified, the maximum over the entire simulation is calculated. If no output is wanted, specify 'dtmaxout = 0'.

units
s

default
9999999

min
0

max
'tstop - start in seconds'

dtrstout

description
Time-step for binary restart file output being written away, turned of by default.

units
s

default
0

dtwnd

description
Time-interval wind update (only for spiderweb)

units
s

default
1800

outputformat

description
Choice whether the SFINCS model output is given in binary 'bin', ascii 'asc' or netcdf files 'net' (default). In case of netcdf output, global output is given in 'sfincs_map.nc', point output in 'sfincs_his.nc' in case observation points are specified.

units

•

default
net

twet_threshold

description
Threshold value of water depth to count cell as flooded for keeping track of wet cells with storetwet = 1

units
m

default
0.01

storetwet

description
Flag to turn on writing away duration that a cell was wet during simulation (storetwet = 1)

units

•

default
0

storevel

description
Flag to turn on writing away velocities on 'dtout' interval during simulation (storevel = 1)

units

•

default
0

storevelmax

description
Flag to turn on writing away maximum velocities on 'dtmaxout' interval during simulation (storevelmax = 1)

units

•

default
0

storefluxmax

description
Flag to turn on writing away maximum flux on 'dtmaxout' interval during simulation (storefluxmax = 1)

units

•

default
0

storecumprcp

description
Flag to turn on writing away cumulative precipitation on 'dtmaxout' interval during simulation (storecumprcp = 1)

units

•

default
0

storehsubgrid

description
Flag to turn on writing away unaccurate water depth estimate for subgrid mode on 'dtmaxout' interval during simulation (storehsubgrid = 1)

units

•

default
0

storeqdrain

description
Flag to turn on writing away drainage discharge during simulation (storeqdrain = 1)

units

•

default
0

storezvolume

description
Flag to turn on writing away water volumes for the subgrid mode during simulation (storezvolume = 1)

units

•

default
0

storemeteo

description
Flag to turn on writing away meteo input data during simulation (storemeteo = 1)

units

-

default
0

storemaxwind

description
Flag to turn on writing away maximum wind speed during simulation (storemaxwind = 1)

units

-

default
0

debug

description
Flag to turn on writing away every timestep to output as debug mode (debug = 1)

units

-

default
0

1.14 Input files

SFINCS consists of many different input files, this overview gives a description, whether they are required or not, unit and format (bin = binary, asc = ascii and net = netcdf).

1.14.1 Domain

sfincs.inp

description
General input file of SFINCS describing all model settings, the domain, forcing and structures.

required
yes

format
asc

depfile = sfincs.dep

description
Elevation (bathymetry and topography) at grid cell centres above a reference level.

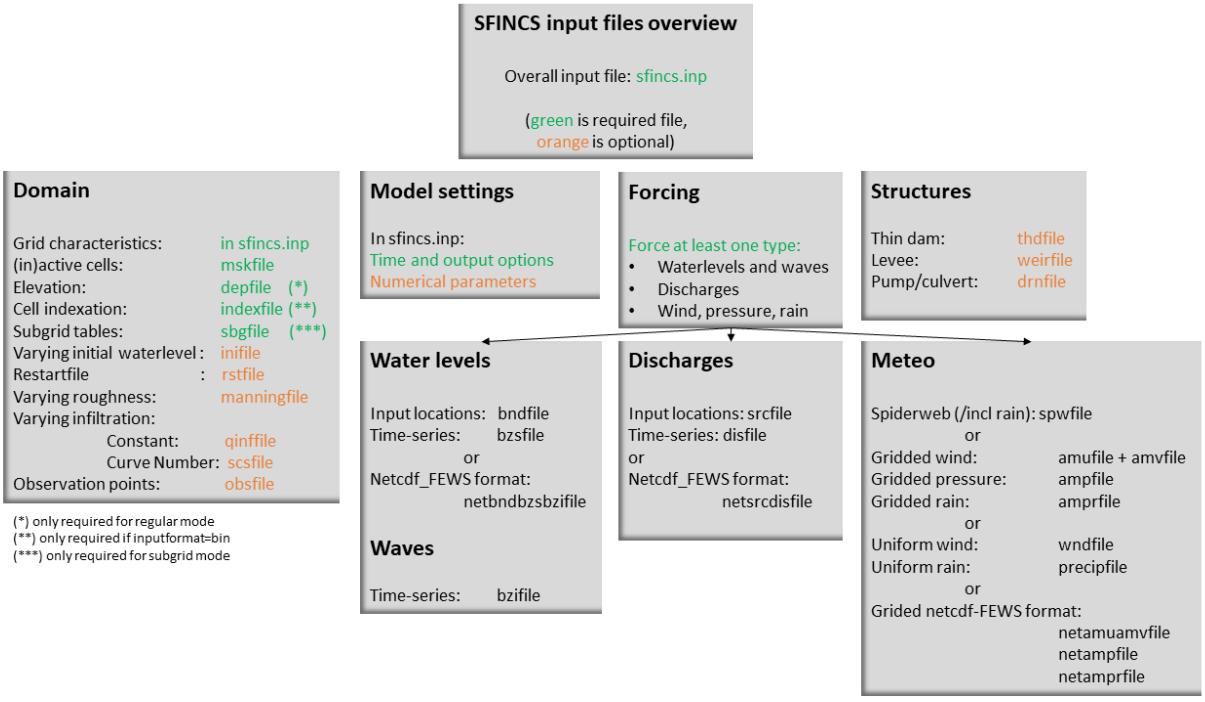


Fig. 1.21: Overview of input file of SFINCS with indication whther they are required or not

units
m above reference level

required
yes in case of regular mode, no in case of subgrid mode

format
bin or asc

mskfile = sfincs.msk

description
This mask indicates for every cell whether it is an inactive cell (msk=0), active cell (msk=1), boundary cell (msk=2) or outflow boundary cell msk=3).

units

•

required
yes

format
bin or asc

indexfile = sfincs.ind

description
File describing the indices of active grid cells within the overall grid. Not used by SFINCS with ascii input.

units

•

required
Only if 'inputformat = bin'

format
bin

mskfile = sfincs.msk

description
This mask indicates for every cell whether it is an inactive cell (msk=0), active cell (msk=1), water level boundary cell (msk=2) or outflow boundary cell msk=3).

units

•

required
yes

format
bin or asc

manningfile = sfincs.man

description
For spatially varying friction values per cell use the manningfile option, with the same grid based input as the depfile using a binary file. Not used by SFINCS in subgrid mode.

units
s/m^(1/3)

required
no in case of regular mode, ignored in case of subgrid mode

format
bin

qinffile = sfincs.qinf

description
For spatially varying constant in time infiltration values per cell use the qinffile option, with the same grid based input as the depfile using a binary file.

units
mm/hr

required
no

format
bin

scsfile = sfincs.scs

description
For spatially varying infiltration values per cell using the Curve Number method use the scsfile option, with the same grid based input as the depfile using a binary file.

units

•

required
no

format
bin

sbgfile = sfincs.sbg

description
File containing subgrid tables, only needed by SFINCS if you want to run your model in the subgrid mode.

units

-

required
Only for running SFINCS in subgrid mode

format
bin

obsfile = sfincs.obs

description
To get output time-series at individual point locations, observations points have to be specified.

units
m in projected UTM zone

required
no (only if point output is wanted)

format
asc

crsfile = sfincs.crs

description
To get output time-series of discharge through a cross-section, cross_sections have to be specified.

units
m in projected UTM zone

required
no (only if cross-section output is wanted)

format
tekal

inifile = sfincs.ini

description
For spatially varying initial water level per cell, with the same grid based input as the depfile using a **binary file**. In older version this was an ascii file still, not from official release v2.0.0 onwards!

units
m above reference level

required
no

format
bin

rstfile = sfincs.rst

description
More advanced restartfile that can also contain fluxes and velocities. As produced by SFINCS if dtrstout > 0 OR trstout > 0. Type of restart - 1: zs, qx, qy, umean and vmean - 2: zs, qx, qy - 3: zs

units

-

required
no

format
bin

1.14.2 Forcing - Water levels and waves

bndfile = sfincs.bnd

description
To specify water-level time-series to the boundary cells (msk=2), first the input locations have to be specified in 'sfincs.bnd'.

units
m in projected UTM zone

required
Only when specifying water levels and waves.

format
asc

bzsfile = sfincs.bzs

description
In the file 'sfincs.bzs' the (slowly varying) water level time-series are specified per input location.

units
m above reference level

required
Only when specifying water levels.

format
asc

bzifile = sfincs.bzi

description
Tn the file 'sfincs.bzi' the quickly varying water level time-series due to incoming waves are specified per input location. Do note that the input timestep should be the same in both the bzs and bzi files!

units
m around mean water level of bzsfile

required
Only when specifying waves.

format
asc

netbndbzsbzifile = sfincs_netbndbzsbzifile.nc

description
To specify all bnd, bzs (and bzi) input in 1 FEWS compatible netcdf input file. Specify either the netcdf version or ascii, not both.

units
m in projected UTM zone, m above reference level & m around mean water level of bzsfile

required
Only when specifying water levels and waves using netcdf input file.

format
net

1.14.3 Forcing - Discharges

srcfile = sfincs.src

description
To specify discharge points, first the input locations have to be specified in 'sfincs.src'.

units
m in projected UTM zone

required
Only when specifying discharges.

format
asc

disfile = sfincs.dis

description
In the file 'sfincs.dis' the discharge time-series are specified per input location.

units
m^3/s

required
Only when specifying discharges.

format
asc

netsrcdisfile = sfincs_netsrcdisfile.nc

description
To specify all src & dis input in 1 FEWS compatible netcdf input file. Specify either the netcdf version or ascii, not both.

units
m in projected UTM zone, discharge in m3/s

required
Only when specifying discharges.

format
net

1.14.4 Forcing - Meteo

spwfile = sfincs.spw

description
Spiderweb file including wind speed, direction, pressure (and possibly rainfall).

units
coordinates: m in projected UTM zone, data: m/s, wind_from_direction in degrees, p_drop in Pa (and precipitation in mm/hr).

required
no

format
asc

amufile = sfincs.amu

description
Delft3D-meteo ascii type input of wind speed in x-direction.

units
coordinates: m in projected UTM zone, data: m/s

required
no

format
asc

amvfile = sfincs.amv

description
Delft3D-meteo ascii type input of wind speed in y-direction.

units
coordinates: m in projected UTM zone, data: m/s

required
no

format
asc

ampfile = sfincs.amp

description
Delft3D-meteo ascii type input of atmospheric pressure.

units
coordinates: m in projected UTM zone, data: Pa

required
no

format
asc

amprfile = sfincs.ampr

description
Delft3D-meteo ascii type input of precipitation intensity.

units
coordinates: m in projected UTM zone, data: mm/hr

required
no

format
asc

wndfile = sfincs.wnd

description
Spatially uniform wind

units
wind speed in m/s, wind direction in nautical from where the wind is coming

required
no

format
asc

precipfile = sfincs.prcp

description
Spatially uniform precipitation

units
mm/hr

required
no

format
asc

netamuamvfile = sfincs_netamuamvfile.nc

description
FEWS type netcdf meteo input with wind speed in both x-&y-direction in m/s.

units
coordinates: m in projected UTM zone, data: m/s

required
no

format
net

netampfile = sfincs_netampfile.nc

description
FEWS type netcdf meteo input with atmospheric pressure in Pa.

units
coordinates: m in projected UTM zone, data: Pa

required
no

format
net

netamprfile = sfincs_netamprfile.nc

description
FEWS type netcdf meteo input with precipitation in mm/hr.

units
coordinates: m in projected UTM zone, data: mm/hr

required
no

format
net

1.14.5 Structures

thdfile = sfincs.thd

description
With a thin dam flow through certain grid cells is completely blocked (i.e. an infinitely high wall).

units
coordinates: m in projected UTM zone.

required
no

format
asc

weirfile = sfincs.weir

description
Weirs are in principle the same as a thin dam, but then with a certain height (levee).

units
coordinates: m in projected UTM zone, elevation in m above reference level, weir formula coefficient in [-]

required
no

format
asc

drnfile = sfincs.drn

description
Drainage pumps and culverts are both specified using the same format file, put with a different indication of the type (type=1 is drainage pump, type=2 is culvert).

units
coordinates: m in projected UTM zone, discharges in m^3/s.

required
no

format
asc

1.15 Output messages

1.15.1 Interpreting the information on the screen

When running SFINCS, some information will be written to the screen. This consists of what version was run (number/date not always completely up to date), some files that were read in and the number of active points. After 'Starting computation …' appears the initialisation phase is finished.

Once '0% complete, Inf s remaining …' appears, the model has actually started computing. The following times the percentual progress % is shown, a rough estimate of the time remaining untill completion of the model run is given.

Once '—Simulation is finished—', your model has run succesfully and is writing away the model output files. Additionaly some information is written to the screen regarding total runtime, time consumption per section, the average time step, and the maximum occured water depth in the entire computation. If you know the initial water depth, this can give an indication whether the model has encountered instabilities or not. Hereafter SFINCS is closed off, ready to start a new simulation.

Normally SFINCS runs using OpenMP, utilising all computing power of available cores (e.g. all 4). This does not make it efficient to runs multiple SFINCS runs in parallel, it is advised to just run multiple simulations in series.

Example

```
 ----------- Welcome to SFINCS -----------

  @@@@@  @@@@@@@ @@ @@  @@   @@@@   @@@@@
 @@@ @@@ @@@@@@@ @@ @@@ @@ @@@@@@@ @@@ @@@
 @@@     @@       @@ @@@ @@ @@   @@ @@@
  @@@@@  @@@@@@  @@ @@@@@@ @@       @@@@@
     @@@ @@       @@ @@ @@@ @@   @@     @@@
 @@@ @@@ @@       @@ @@  @@  @@@@@@ @@@ @@@
  @@@@@  @@       @@ @@   @   @@@@   @@@@@

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

Build-Revision: $Rev: v0.0.1-alpha$
Build-Date:     $Date: 2022-10-25$

Reading input file ...
Info : Running SFINCS in subgrid mode ...
Reading meteo data ...
Info : Preparing SFINCS grid on regular mesh ...
Reading sfincs.ind ...
Reading sfincs.msk ...
Number of active z points    :         4055
Number of active u/v points  :         7972
Reading sfincs.sbg ...
Reading water level boundaries ...
Reading observation points ...
Initializing output ...
```

(continues on next page)

(continued from previous page)

```
 ---------- Starting simulation ----------

   0% complete,       - s remaining ...
   5% complete,     0.9 s remaining ...
  10% complete,     0.9 s remaining ...
  15% complete,     0.9 s remaining ...
  20% complete,     0.9 s remaining ...
  25% complete,     0.8 s remaining ...
  30% complete,     0.8 s remaining ...
  35% complete,     0.7 s remaining ...
  40% complete,     0.7 s remaining ...
  45% complete,     0.6 s remaining ...
  50% complete,     0.6 s remaining ...
  55% complete,     0.5 s remaining ...
  60% complete,     0.5 s remaining ...
  65% complete,     0.4 s remaining ...
  70% complete,     0.4 s remaining ...
  75% complete,     0.3 s remaining ...
  80% complete,     0.2 s remaining ...
  85% complete,     0.2 s remaining ...
  90% complete,     0.1 s remaining ...
  95% complete,     0.1 s remaining ...
 100% complete,     0.0 s remaining ...

 Info : Write maximum values of final timestep since t=dtmaxout was not reached
 yet...

 ---------- Simulation finished ----------
              ..............
          ......:@@@@@@@@:......
       ..::::::..@@........@@.::::::..
     ..:::::::..@@..::..::..@@.::::::..
.::::::::..@@.............@@.::::::::.
   .::::::::..@@.............@@.::::::::.
  .:::::::::..@@.............@@..:::::::::.
 .::::::::::...@@.@..@@..@.@@..::::::::::.
 .::::::::::...:@@@..@@..@@@:..::::::::::..
 .............@@.@@..@@..@@.@@.............
 ^^^~~^^~~^^@@..............@@^^^~^^^~~^^
 .::::::::::@@..............@@.::::::::::.
  .......:.@@.....@.....@....@@.:........
   .::....@@......@.@@@.@....@@.....::.
.:::~@@.:...:.@@...@@.:.:.@@~::::.
     .::~@@@@@@@@@@.....@@@@@@@@@~::.
       ..:~~~~~~~:.......:~~~~~~~:..
          ......................
              ..............
 ----------------------------------------

 Total time             :      1.211
 Total simulation time  :      1.198
 Time in input          :      0.013
```

(continues on next page)

(continued from previous page)

```
Time in boundaries     :      0.042 (  3.5%)
Time in momentum       :      0.881 ( 73.5%)
Time in continuity     :      0.207 ( 17.3%)
Time in output         :      0.055 (  4.6%)

Average time step (s)  :     22.031

---------- Closing off SFINCS -----------
```

1.15.2 Possible error messages and possible solutions

In case the following message is written to the screen, it means that something in the simulation has gone wrong.

```
Maximum depth of 100.0 m reached!!! Simulation stopped.
```

This means that a too large water depth has occured somewhere in the domain, indicating that some input is probably not optimal.

Possible problems can be:

- The provided elevation file has very rapid changes in elevation, that locally lead to large water level gradients and fluxes. Possible solution: locally smooth the elevation data and provide this as a new depfile
- In general the internal timesteps of SFINCS might be too large. Possible solution: reduce timesteps by supplying a lower value of alpha (e.g. 0.5) or set a low enough value of 'dtmax'.
- Sometimes a simulation might contain too large water depths are start in too deep water. This can potentially create problems as SFINCS is intented as a shallow water model.
- When only forcing discharges in a for the rest entirely dry domain, the initial time steps can be too coarse to account for the needed timesteps when the discharge starts to flow. Possible solution: Make sure that part of the river/domain initially has water (limiting the time step) by specifying either 'zsini' or an 'inifile'.
- When forcing waves, the bzifile time-series might contain too rapid changes in water level, the internal timesteps of SFINCS are too large. Possible solution: reduce timesteps by supplying a lower value of alpha (e.g. 0.5).
- **Tip to check your model**: specify netcdf output and load in the sfincs_map.nc file (e.g. Quickplot, Panoply, Matlab, Python) and have a look at the variables 'zb' and 'msk'. Then you can see how SFINCS has interpreted the prodivided depfile and mskfile. Does map plots of these variables look weird? Probably something in your input file is not entirely correct!

Besides model instabilities, other recurring problems might be:

- A specified (forcing) file/parameters is not read in > check whether you specified the name (e.g. netamuamvfile = netamuamv.nc) with **ONLY SPACES** in between the keyword and argument. SFINCS does not interpret a mixture of spaces and tabs well. This may cause a file or parameter to be read in as 'none', whereafter this is not used in the model simulation as wanted.

1.16 Output description

1.16.1 Parameters netcdf file global (sfincs_map.nc)

In case of netcdf output, the given parameters mean the following:

x

description
x coordinate of cell centers in projected reference system

standard_name
projection_x_coordinate

units
m in projected reference system

y

description
y coordinate of cell centers in projected reference system

standard_name
projection_y_coordinate

units
m in projected reference system

zb

description
Bed level elevation (in case of subgrid version of SFINCS, this elevation is not used in the model but the sbgfile with subgrid tables is used instead).

standard_name
altitude

units
m above reference level

msk

description
Time-step global map output.

standard_name
land_binary_mask

units

-

time

description
Time of global map output.

standard_name
time

units
seconds since ‘tref’

zs

description
Instantaneous water level per 'dtout' timestep, corresponding with netcdf variable 'time'.

standard_name
sea_surface_height_above_mean_sea_level

units
m above reference level

h

description
Instantaneous water depth per 'dtout' timestep, corresponding with netcdf variable 'time'.

standard_name
depth

units
m

u

description
Instantaneous flow velocity in u-direction per 'dtout' timestep, corresponding with netcdf variable 'time'.

standard_name
sea_water_x_velocity

units
m/s

v

description
Instantaneous flow velocity in v-direction per 'dtout' timestep, corresponding with netcdf variable 'time'.

standard_name
sea_water_y_velocity

units
m/s

timemax

description
Time of global map output per 'dtmaxout' timestep.

standard_name
time

units
seconds since 'tref'

zsmax

description
Maximum water level per 'dtmaxout' timestep, only given if dtmaxout>0, corresponding with netcdf variable 'timemax'.

standard_name
maximum of sea_surface_height_above_mean_sea_level

units
m above reference level

vmax

description
Maximum flow velocity proxy per 'dtmaxout' timestep, only given if dtmaxout>0, corresponding with netcdf variable 'timemax'.

standard_name
maximum_flow_velocity

units
m/

qmax

description
Maximum flow flux proxy per 'dtmaxout' timestep, only given if dtmaxout>0, corresponding with netcdf variable 'timemax'.

standard_name
maximum_flux

units
m^2/s

cuminf

description
Cumulative infiltration depth over whole simulation.

units
m

cumprcp

description
Cumulative precipitation depth over whole simulation.

units
m

inp

description
Copy of all the supplied input to SFINCS from 'sfincs.inp'.

units

-

total_runtime

description
Total model runtime in seconds, as displayed by SFINCS to the screen.

units
s

average_dt

description
Model average timestep in seconds, as displayed by SFINCS to the screen.

units
s

1.16.2 Parameters netcdf file observation points (sfincs_his.nc)

This file is only created if observation points are supplied in the 'obsfile', or if weirs/cross-sections are supplied.

point_x

description
x coordinate of interpreted observation points in projected reference system

standard_name
projection_x_coordinate

units
m in projected reference system

point_y

description
y coordinate of interpreted observation points in projected reference system

standard_name
projection_y_coordinate

units
m in projected reference system

station_x

description
x coordinate of specified observation points in projected reference system

standard_name
projection_x_coordinate

units
m in projected reference system

station_y

description
y coordinate of specified observation points in projected reference system

standard_name
projection_y_coordinate

units
m in projected reference system

structure_x

description
x coordinate of snapped location on SFINCS grid of weirs in projected reference system

standard_name
projection_x_coordinate

units
m in projected reference system

structure_y

description
y coordinate of snapped location on SFINCS grid of weirs in projected reference system

standard_name
projection_y_coordinate

units
m in projected reference system

structure_height

description
weir height on snapped location on SFINCS grid of weirs in projected reference system

standard_name
projection_x_coordinate

units
m above reference level

point_zb

description
Bed level elevation of observation points.

standard_name
altitude

units
m above reference level

time

description
Time of his output.

standard_name
time

units
seconds since 'tref'

point_zs

description
Instantaneous water level per 'dthisout' timestep of observation points, corresponding with netcdf variable 'time'.

standard_name
sea_surface_height_above_mean_sea_level

units
m above reference level

point_h

description
Instantaneous water depth per 'dthisout' timestep of observation points, corresponding with netcdf variable 'time'.

standard_name
point_h

units
m

point_u

description
Instantaneous flow velocity in u-direction per 'dthisout' timestep of observation points, corresponding with netcdf variable 'time'.

standard_name
sea_water_x_velocity

units
m/s

point_v

description
Instantaneous flow velocity in v-direction per 'dthisout' timestep of observation points, corresponding with netcdf variable 'time'.

standard_name
sea_water_y_velocity

units
m/s

point_uvmag

description
Instantaneous absolute flow velocity per 'dthisout' timestep of observation points, corresponding with netcdf variable 'time'.

standard_name
sea_water_velocity

units
m/s

point_uvdir

description
Instantaneous absolute flow velocity per 'dthisout' timestep of observation points, corresponding with netcdf variable 'time'.

standard_name
sea_water_velocity_direction

units
degrees wrt north

point_prcp

description
Instantaneous precipitation rate 'dthisout' timestep, corresponding with netcdf variable 'time'.

standard_name
sea_surface_height_above_mean_sea_level

units
m above reference level

point_qinf

description
Instantaneous infiltration rate per 'dthisout' timestep, corresponding with netcdf variable 'time'.

standard_name
point_qinf

units
m

crosssection_discharge

description
Discharge through cross-section per 'dthisout' timestep, corresponding with netcdf variable 'time'.

standard_name
discharge

units
m3/s

drainage_discharge

description
Discharge through drainage structure per 'dthisout' timestep, corresponding with netcdf variable 'time'.

standard_name
discharge

units
m3/s