libzsf

Tjerk Vreeken, Otto Weiler

May 03, 2023

CONTENTS:

1 Getting Started **3**
- 1.1 Excel workbook . . . 3
- 1.2 Python package . . . 5
- 1.3 C library . . . 6
- 1.4 Fortran interface . . . 7
- 1.5 From Source . . . 7

2 Examples **9**
- 2.1 Excel Examples . . . 9
- 2.2 Python Examples . . . 17

3 Theory **33**
- 3.1 Introduction . . . 33
- 3.2 Processes and definitions . . . 35
- 3.3 Equations per locking phase . . . 51
- 3.4 Cycle-averaged flows and salinities . . . 57
- 3.5 Numerical approach cycle-averaged values . . . 59

4 API **61**
- 4.1 C API . . . 61
- 4.2 Python API . . . 67

5 Support **69**

6 Indices and tables **71**

Index **73**

This is the documentation for the ZSF, a Deltares tool to calculate salt intrusion through shipping locks. The documentation covers:

- How to get the ZSF running on your computer. For end-users this means downloading either the Excel workbook, or installing the Python package.
- Making your own calculations, by means of several illustrative examples for both the Excel workbook and Python package.
- The theory on which the ZSF is based, serving as background information on the inner workings of the libzsf core.
- An overview of the C and Python API, e.g. to embed the ZSF in other software.

Note: ZSF is an abbreviation for "Zeesluisformulering", which would translate to Sea Lock Formulation.

CHAPTER

ONE

GETTING STARTED

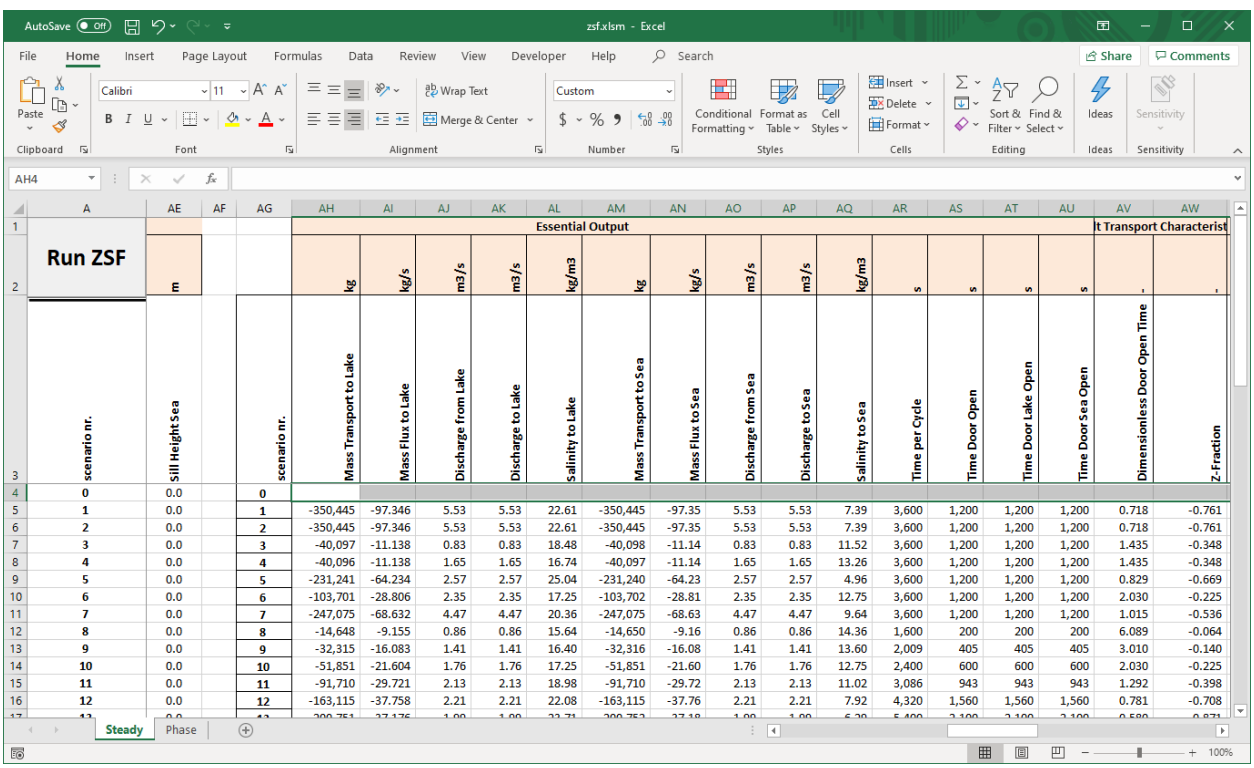
Typical end-users should use either the Excel wrapper, or the Python *pyzsf* package, instructions for which are in the respective sections below. When embedding the ZSF in other software, the easiest way is to download *libzsf* shared or static libraries for Linux or Windows, see *C library*. It is of course also possible to build the Python and C libraries from source, see *From Source*.

1.1 Excel workbook

One of the easiest ways of using the ZSF on Windows is to download the Excel workbook from the releases page on GitLab. This will download a zip file containing a macro-enabled Excel file, and two DLLs (one for 32-bit systems, and one for 64-bit systems). These DLLs contain computational routines that are called using Excel macros.

When opening the workbook, you need to make sure to enable the macros if so prompted, otherwise it will not be possible to calculate anything.

To test whether the workbook runs as intended, select cells *AH4:BI4* that contain the output of the first row of input, and delete their contents.

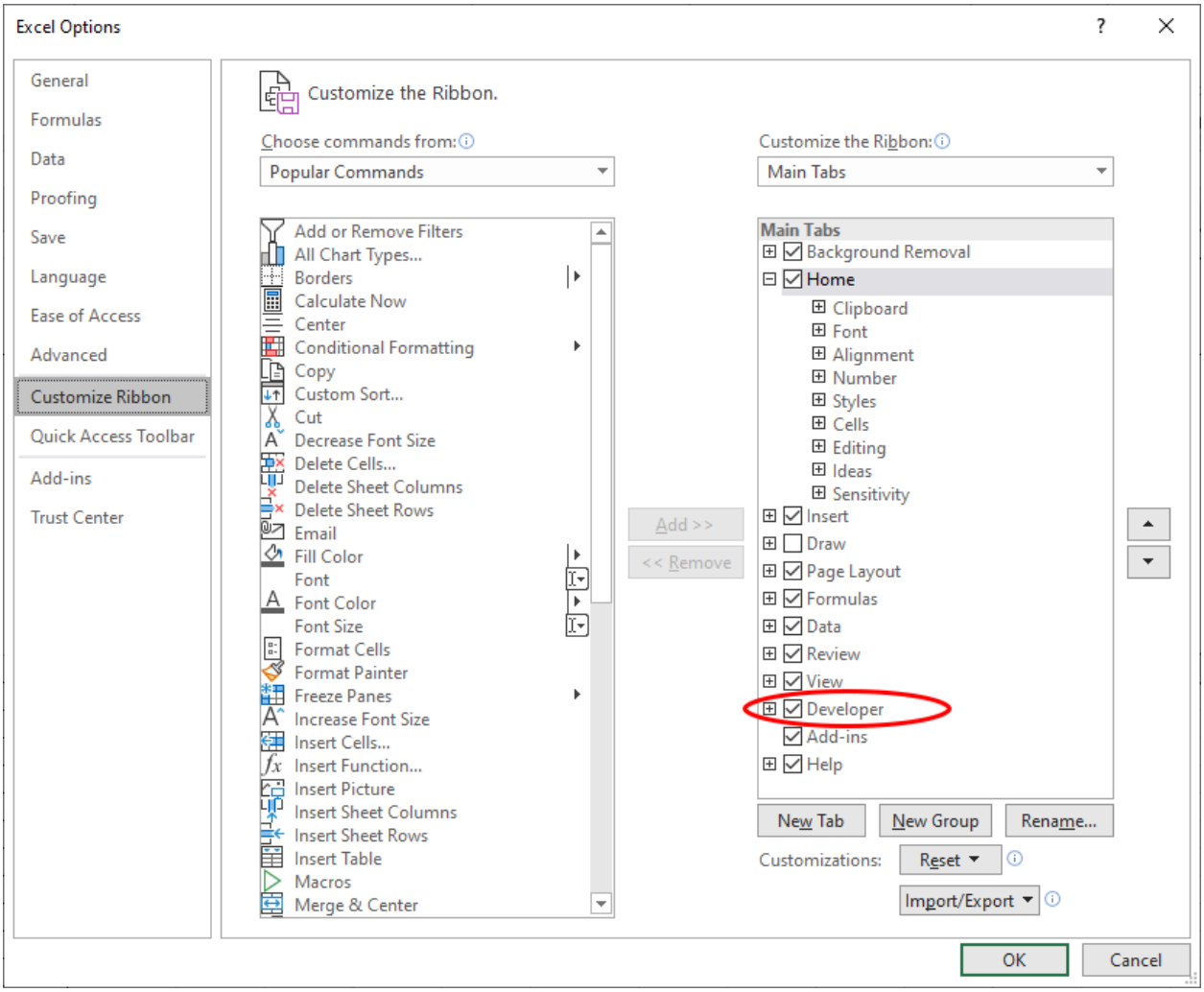


scenario nr.	Sill Height Sea	scenario nr.	Mass Transport to Lake	Mass Flux to Lake	Discharge from Lake	Discharge to Lake	Salinity to Lake	Mass Transport to Sea	Mass Flux to Sea	Discharge from Sea	Discharge to Sea	Salinity to Sea	Time per Cycle	Time Door Open	Time Door Lake Open	Time Door Sea Open	Dimensionless Door Open Time	Z-Fraction
	m		kg	kg/s	m3/s	m3/s	kg/m3	kg	kg/s	m3/s	m3/s	kg/m3	s	s	s	s	-	-
0	0.0	0																
1	0.0	1	-350,445	-97.346	5.53	5.53	22.61	-350,445	-97.35	5.53	5.53	7.39	3,600	1,200	1,200	1,200	0.718	-0.761
2	0.0	2	-350,445	-97.346	5.53	5.53	22.61	-350,445	-97.35	5.53	5.53	7.39	3,600	1,200	1,200	1,200	0.718	-0.761
3	0.0	3	-40,097	-11.138	0.83	0.83	18.48	-40,098	-11.14	0.83	0.83	11.52	3,600	1,200	1,200	1,200	1.435	-0.348
4	0.0	4	-40,096	-11.138	1.65	1.65	16.74	-40,097	-11.14	1.65	1.65	13.26	3,600	1,200	1,200	1,200	1.435	-0.348
5	0.0	5	-231,241	-64.234	2.57	2.57	25.04	-231,240	-64.23	2.57	2.57	4.96	3,600	1,200	1,200	1,200	0.829	-0.669
6	0.0	6	-103,701	-28.806	2.35	2.35	17.25	-103,702	-28.81	2.35	2.35	12.75	3,600	1,200	1,200	1,200	2.030	-0.225
7	0.0	7	-247,075	-68.632	4.47	4.47	20.36	-247,075	-68.63	4.47	4.47	9.64	3,600	1,200	1,200	1,200	1.015	-0.536
8	0.0	8	-14,648	-9.155	0.86	0.86	15.64	-14,650	-9.16	0.86	0.86	14.36	1,600	200	200	200	6.089	-0.064
9	0.0	9	-32,315	-16.083	1.41	1.41	16.40	-32,316	-16.08	1.41	1.41	13.60	2,009	405	405	405	3.010	-0.140
10	0.0	10	-51,851	-21.604	1.76	1.76	17.25	-51,851	-21.60	1.76	1.76	12.75	2,400	600	600	600	2.030	-0.225
11	0.0	11	-91,710	-29.721	2.13	2.13	18.98	-91,710	-29.72	2.13	2.13	11.02	3,086	943	943	943	1.292	-0.398
12	0.0	12	-163,115	-37.758	2.21	2.21	22.08	-163,115	-37.76	2.21	2.21	7.92	4,320	1,560	1,560	1,560	0.781	-0.708

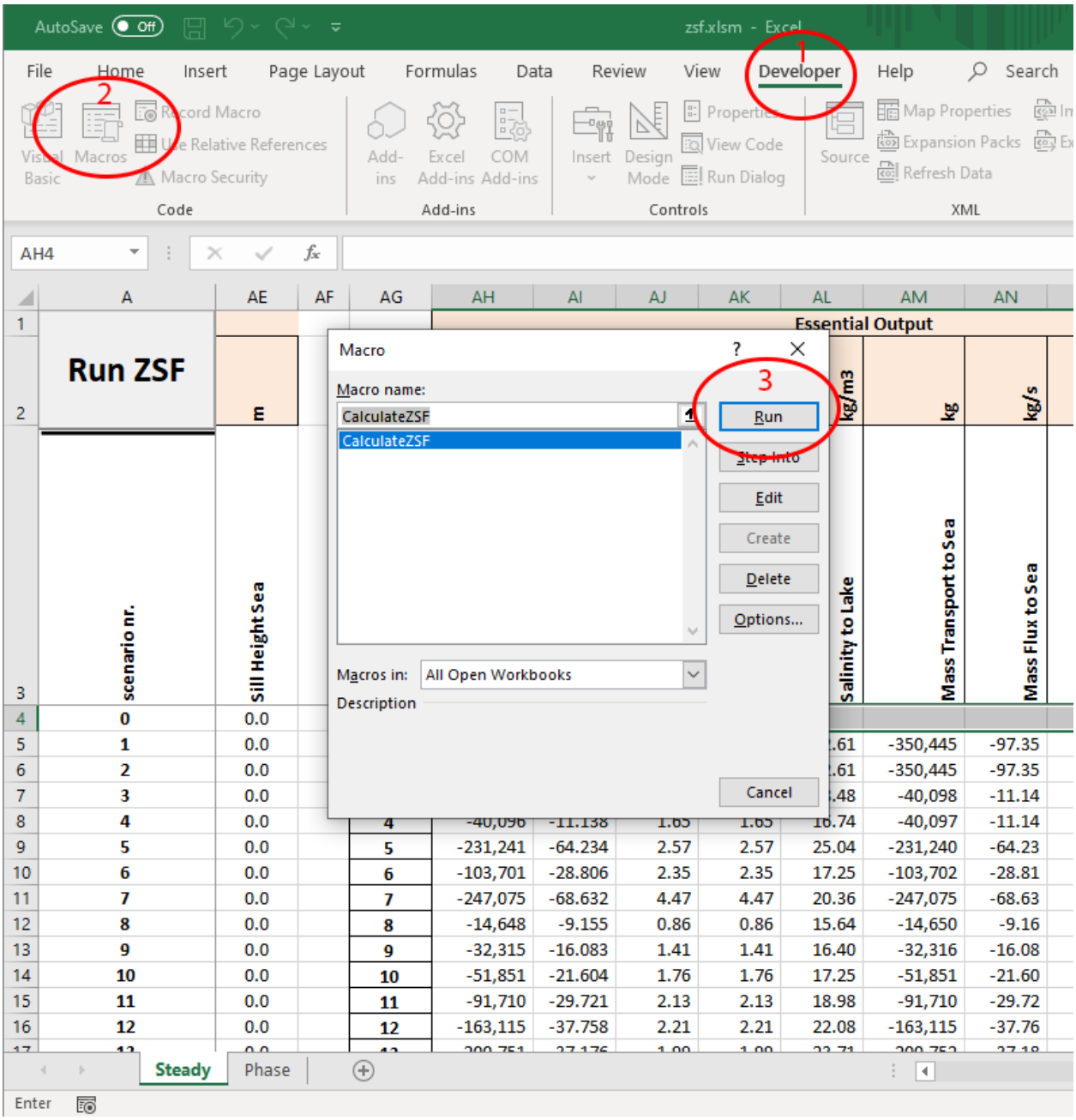
Next, with the cells still selected, press the *Run ZSF* button. If correct, the output that was there should reappear. For further instructions, it is best to go through the examples.

1.1.1 Troubleshooting

If nothing happens, there is a chance the macro did not run, because the button is unresponsive. To test whether this is the case, and check that macros are indeed able to run, go to *File -> Options -> Customize Ribbon* and enable the Developer mode.



With that done, select the Developer tab on the ribbon, and run the macro manually (still with the cells on row 4 selected).



1.2 Python package

The Python package of the ZSF is called *pyzsf*. Although not required, it is recommended to install it in a virtual environment. See the official Python tutorial for more information on how to set up and activate a virtual environment.

pyzsf, including its dependencies, can be installed using the pip package manager:

```
# Install pyzsf using the pip package manager
pip install pyzsf
```

Note: Pip version 20.0 or higher is required to install *pyzsf*, or it will fail to find a matching distribution for your platform. If you have an older version, please run `python -m pip install -U pip` before installing pyzsf.

To test whether it works, import pyzsf and call its `zsf_calc_steady()` function.

```
Windows PowerShell
Successfully installed pip-20.2.4
(venv) C:\test_pyzsf> pip install pyzsf
Collecting pyzsf
  Using cached pyzsf-0.0.1.dev1-cp36-abi3-win_amd64.whl (20 kB)
Collecting cffi>=1.0.0
  Downloading cffi-1.14.3-cp38-cp38-win_amd64.whl (179 kB)
     |                                | 179 kB 3.3 MB/s
Collecting pycparser
  Using cached pycparser-2.20-py2.py3-none-any.whl (112 kB)
Installing collected packages: pycparser, cffi, pyzsf
Successfully installed cffi-1.14.3 pycparser-2.20 pyzsf-0.0.1.dev1
(venv) C:\test_pyzsf> python
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import pyzsf
>>> import pprint
>>> pprint.pprint(pyzsf.zsf_calc_steady())
{'discharge_from_lake': 0.6943730052974024,
 'discharge_from_sea': 0.6943730052891575,
 'discharge_to_lake': 0.6943730052974024,
 'discharge_to_sea': 0.6943730052891575,
 'mass_transport_lake': -144970.17305977637,
 'mass_transport_sea': -144970.17007884194,
 'salinity_to_lake': 29.99701760392959,
 'salinity_to_sea': 1.0029829919743198,
 'salt_load_lake': -20.134746258302272,
 'salt_load_sea': -20.134745844283604}
>>>
```

1.3 C library

Static and dynamic libraries for both Windows and Linux are available on the releases page on GitLab. The Linux libraries are built using the manylinux2010 docker image, and should be therefore be compatible with most versions of Linux.

1.4 Fortran interface

A wrapper is provided to easily call the static and dynamic libraries from Fortran. See the releases page on GitLab, or download the `zsf.f90` interface file directly from the git tree.

1.5 From Source

The latest libzsf source can be downloaded using Git:

```
# Get libzsf source
git clone https://gitlab.com/deltares/libzsf.git
```

Note that *cmake* is needed to build libzsf, and a working Python installation is required to build the pyzsf wrapper. For more detailed build instructions, it is probably easiest to look at the `build:windows` and `build:linux` sections in the *.gitlab.yml* file in the root of the source tree. These instructions are always up to date, and give a concise and clear overview of the steps required to build from source.

CHAPTER

TWO

EXAMPLES

2.1 Excel Examples

2.1.1 Steady-state calculation

Overview

The purpose of this example is to understand the basic steps to calculate the salt transports through a shipping lock in steady state operation. The scenario is the following: A single lock connects a canal to the sea, and the lock is busy during the day but quiet during the night. We want to know how much salt comes in on average, and figure out ways to reduce the salt intrusion by means of mitigating measures like bubble screens.

Properties of the lock and its operation

The physical dimensions of our single shipping lock are:

- length: 148 m
- width: 14 m
- bottom: -4.4 mNAP (Dutch Ordnance Datum)

For the boundary conditions we will assume that the sea level is the same as the canal level, with both being equal to 0.0 mNAP. The salinity on the salt side is significantly higher than on the canal side.

- salinities: 5 kg/m^3 on the canal side close to the lock, and 25 kg/m^3 on the sea side
- head: 0.0 mNAP on both sides
- temperature: 15.0 °C on both sides

The last step is to derive basic parameters from the locking information.

- During daytime, the lock operates at a pace of 1.25 cycles per hour (= 30 cycles per 24 hours). During the night, the lock operates at a pace of 10 cycles per 24 hours.
- It takes 5 minutes to open or close the doors on either side, and also 5 minutes to level.
- There are the same number of ships going from the sea to the canal, and vice-versa.
- The ships going to and from the canal also have an equal displacement of 1000 m^3.

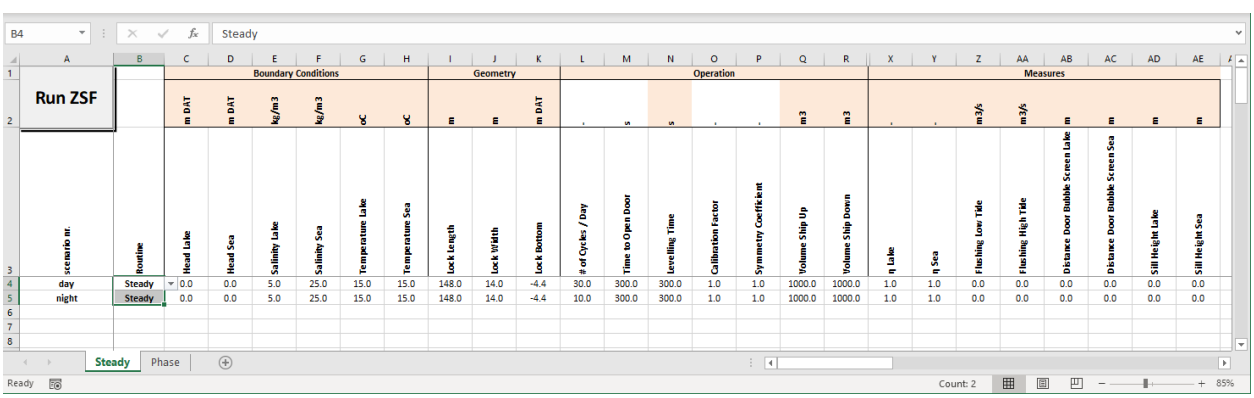
In the first calculation, the lock does not have any sills or bubble screens, nor is there any flushing. These are of course measures that we will take a look at later in this example to reduce the amount of salt intrusion.

See also:

For an overview of these parameters, and more in-depth discussion on them, see *Input* and *Processes and definitions*.

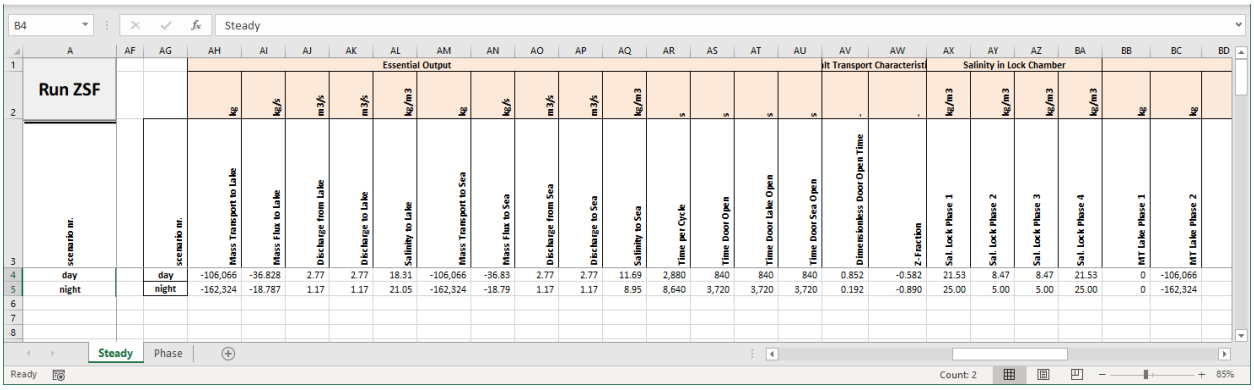
Salt load without measures

The next step is to enter all these physical and operational characteristics in the Excel workbook. As we are assuming constant operation and boundary conditions, we can use the *Steady* tab. We need a total of two rows, one for daytime operation and one for nighttime. Entering the values above into the Excel sheet should result in a something like shown in the image below. Note that some field are not used (yet); they will be discussed below in detail.



		Boundary Conditions						Geometry			Operation							Measures							
Run ZSF		m DAT	m DAT	kg/m3	kg/m3	oC	oC	m	m	m DAT	-	s	s	-	-	m3	m3	-	-	m3/s	m3/s	m	m	m	m
scenario nr.	routine	Head Lake	Head Sea	Salinity Lake	Salinity Sea	Temperature Lake	Temperature Sea	lock length	lock Width	lock Bottom	# of Cycles / Day	Time to Open Door	Levelling Time	Calibration Factor	Symmetry Coefficient	Volume Ship Up	Volume Ship Down	η Lake	η Sea	Flushing Low Tide	Flushing High Tide	Distance Door Bubble Screen Lake	Distance Door Bubble Screen Sea	Sill Height Lake	Sill Height Sea
day	Steady	0.0	0.0	5.0	25.0	15.0	15.0	148.0	14.0	-4.4	30.0	300.0	300.0	1.0	1.0	1000.0	1000.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
night	Steady	0.0	0.0	5.0	25.0	15.0	15.0	148.0	14.0	-4.4	10.0	300.0	300.0	1.0	1.0	1000.0	1000.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0

The next step is to select (at least) one cell on each of the rows you want to calculate, and press the *Run ZSF* button in the top-left. The output columns in the sheet should then take on values similar to those shown in the image below.



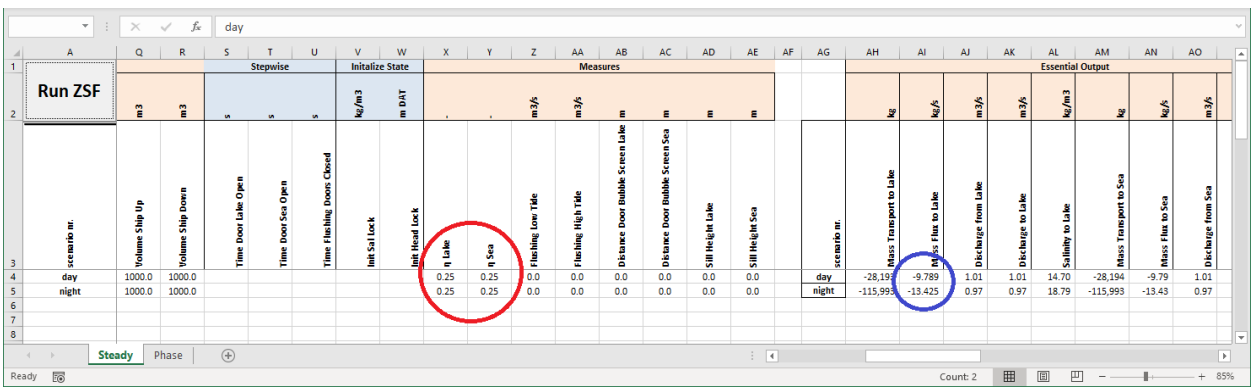
Run ZSF			Essential Output														Salinity in Lock Chamber					
			kg	kg/s	m3/s	m3/s	kg/m3	kg	kg/s	m3/s	m3/s	kg/m3	s	s	s	s	-	-	kg/m3	kg/m3	kg/m3	kg/m3	kg	kg
scenario nr.		scenario nr.	Mass Transport to Lake	Mass Flux to Lake	Discharge from Lake	Discharge to Lake	Salinity to Lake	Mass Transport to Sea	Mass Flux to Sea	Discharge from Sea	Discharge to Sea	Salinity to Sea	Time per Cycle	Time Door Open	Time Door Lake Open	Time Door Sea Open	Dimensionless Door Open Time	Z-Fraction	Sal. Lock Phase 1	Sal. Lock Phase 2	Sal. Lock Phase 3	Sal. Lock Phase 4	MT Lake Phase 1	MT Lake Phase 2
day		day	-106,066	-36.828	2.77	2.77	18.31	-106,066	-36.83	2.77	2.77	11.69	2,880	840	840	840	0.852	-0.582	21.53	8.47	8.47	21.53	0	-106,066
night		night	-162,324	-18.787	1.17	1.17	21.05	-162,324	-18.79	1.17	1.17	8.95	8,640	3,720	3,720	3,720	0.192	-0.890	25.00	5.00	5.00	25.00	0	-162,324

From these results we can see that there is a salt load of almost 37 kg/s during daytime, and 19 kg/s during nighttime.

Comparing salt intrusion measures

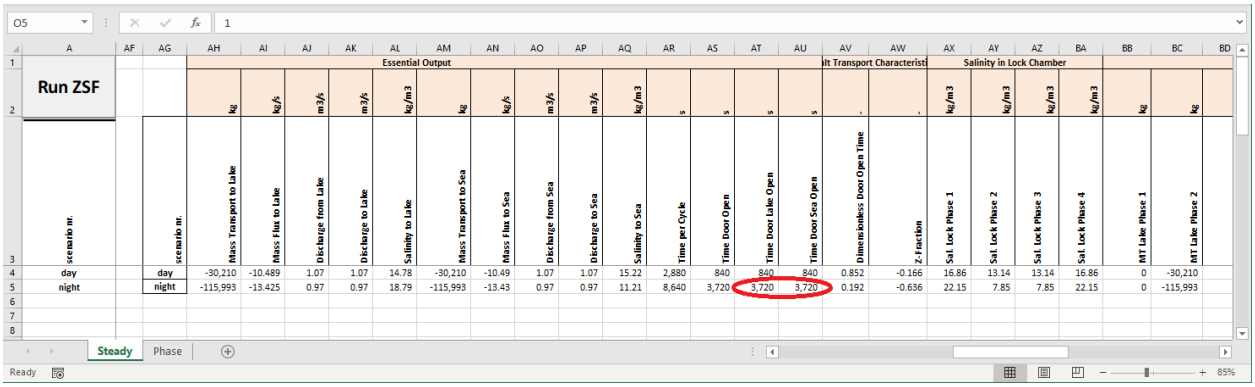
The maximum allowable salt load has been determined to be 12 kg/s, so the salt load during daytime and nighttime are currently not acceptable. The ZSF can help compare various salt intrusion measures that can be taken to reduce the salt load to acceptable levels. For this particular lock sills are not a feasible option, but bubble screens and flushing discharges could be.

A typical maximally efficient bubble screen can reduce the pace of the lock exchange to about 25%. If we fill in 0.25 at both *Lake* and *Sea*, the salt load during daytime is reduced to about 10 kg/s. However, the salt load at nighttime is reduced much less. With a value of about 13.5 kg/s it is now even higher than that during the daytime, even though there are fewer ships passing through the lock.



Run ZSF					Stepwise			Initalize State		Measures									Essential Output							
			m3	m3	s	s	s	kg/m3	m DAT	-	-	m3/s	m3/s	m	m	m	m		kg	kg/s	m3/s	m3/s	kg/m3	kg	kg/s	m3/s
scenario nr.			Volume Ship Up	Volume Ship Down	Time Door Lake Open	Time Door Sea Open	Time Flushing Doors Closed	Init Sal Lock	Init Head Lock	n Lake	n Sea	Flushing Low Tide	Flushing High Tide	Distance Door Bubble Screen Lake	Distance Door Bubble Screen Sea	Sill Height Lake	Sill Height Sea	scenario nr.	Mass Transport to Lake	Mass Flux to Lake	Discharge from Lake	Discharge to Lake	Salinity to Lake	Mass Transport to Sea	Mass Flux to Sea	Discharge from Sea
day			1000.0	1000.0						0.25	0.25	0.0	0.0	0.0	0.0	0.0	0.0	day	-28,19	-9.789	1.01	1.01	14.70	-28,194	-9.79	1.01
night			1000.0	1000.0						0.25	0.25	0.0	0.0	0.0	0.0	0.0	0.0	night	-115,993	-13.425	0.97	0.97	18.79	-115,993	-13.43	0.97

Bubble screens are only effective if the doors are closed well before the (reduced) lock is reduced. With only 2 to 3 locking cycles during the night, the doors are open for more than an hour at a time, see the output columns *Time Door Lake Open* and *Time Door Sea Open*.



Run ZSF		Essential Output																							
		kg	kg/s	m3/s	m3/s	kg/m3	kg	kg/s	m3/s	m3/s	kg/m3	s	s	s	s	-	-	kg/m3	kg/m3	kg/m3	kg/m3	kg	kg
scenario nr.	scenario nr.	Mass Transport to Lake	Mass Flux to Lake	Discharge from Lake	Discharge to Lake	Salinity to Lake	Mass Transport to Sea	Mass Flux to Sea	Discharge from Sea	Discharge to Sea	Salinity to Sea	Time per Cycle	Time Door Open	Time Door Lake Open	Time Door Sea Open	Dimensionless Door Open Time	Z-Fraction	Sal. Lock Phase 1	Sal. Lock Phase 2	Sal. Lock Phase 3	Sal. Lock Phase 4	NT Lake Phase 1	NT Lake Phase 2
day	day	-30,210	-10.489	1.07	1.07	14.78	-30,210	-10.49	1.07	1.07	15.22	2,880	840	840	840	0.852	-0.166	16.86	13.14	13.14	16.86	0	-30,210
night	night	-115,993	-13.425	0.97	0.97	18.79	-115,993	-13.43	0.97	0.97	11.21	8,640	3,720	3,720	3,720	0.192	-0.636	22.15	7.85	7.85	22.15	0	-115,993

If we can tell the lock operator to close the doors right after ships have finished sailing out, we can reduce the salt intrusion significantly. To reduce these door-open durations, we can use *Calibration Factor*. If we know that the doors are open about 20 minutes at a time during the night, we can fill in a value of approximately 0.3 here to reduce the current duration of about an hour with. Recalculating with this will give a salt load of about 4 kg/s during the night, which is acceptable.

2.1.2 Phase-wise calculation

Overview

Note: This example focuses on performing a phase-wise calculation of salt-intrusion through a shipping lock. It assumes basic exposure to the Excel interface. If you are a first-time user of the ZSF, see the *Steady-state calculation* example.

The purpose of this example is to understand the basic steps to perform a phase-wise calculation of the salt intrusion through a shipping lock. The scenario is similar to that of the *Steady-state calculation* example, in that there is a single lock connecting a canal to the sea. The differences with the steady state example are:

- there is a head difference now, with the sea at 2 mNAP

- we are only going to calculate one full locking cycle during the daytime operation

Different phases and routines

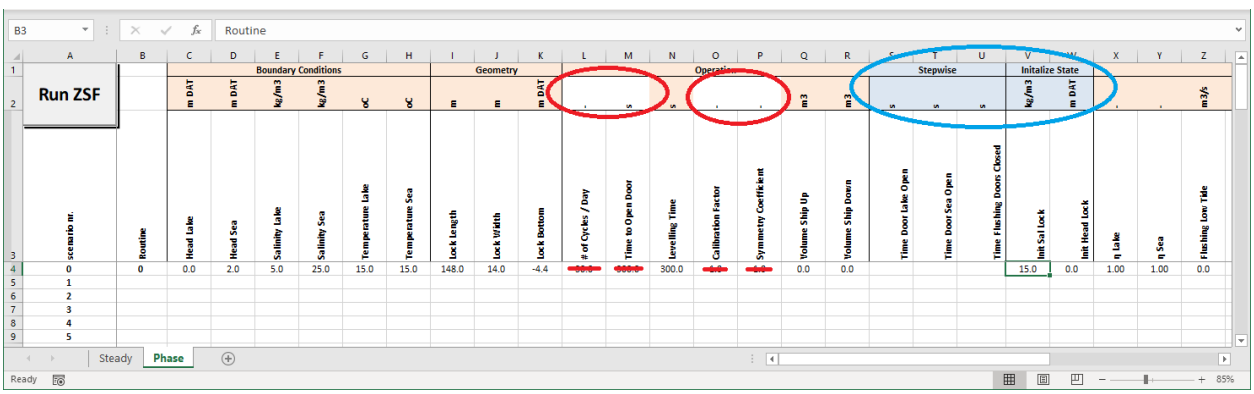
There are a few phases the lock can go through, each having associated salt transports. See Section 3.3 in the theory for more explanation on what each phase entails. The term *routines* refers to the actual functions that are called, and their naming mostly corresponds to that of the phases. The only major difference is that there is an initialization routine **0** to set the starting state, but no such phase as there are no transports yet.

See also:

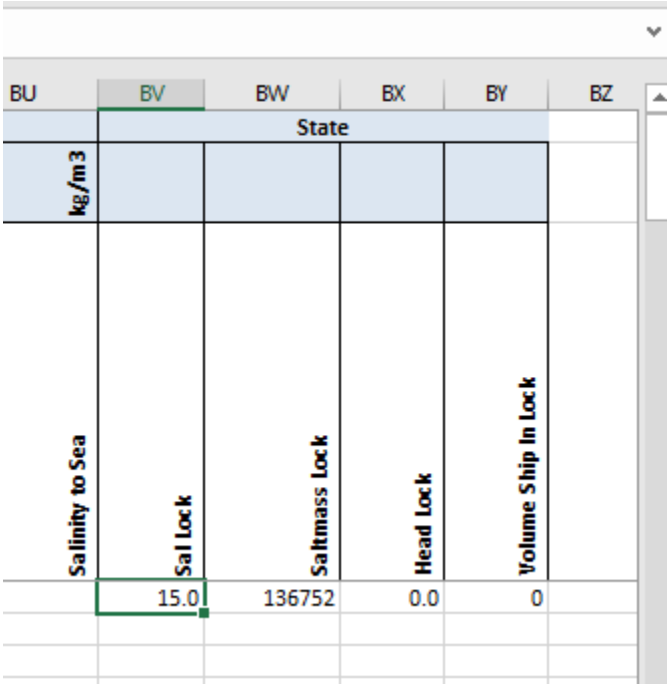
For an overview of phases and how the transports are determined, see *Equations per locking phase*.

Initializing the lock

First we need to initialize the lock with a certain salinity and head. The dimensions of the lock are equal to that of the *Steady-state calculation* example. You can therefore copy the row of input parameters to the first calculation row in the *Phase* worksheet. There are a few parameters that are no longer needed, because we will set them explicitly ourselves. These input parameters that needed for steady state calculation, but not needed for phase-wise calculation, can be spotted by the lack of shading in the second row. Instead, we will have to enter values in the columns shaded blue. We set the head of the sea to *2.0 mNAP*, and set the initial head and salinity inside the lock chamber to *0.0 mNAP* and *15 kg/m*^{3} respectively.



Next, we make sure that the *routine* column is set to **0**. Then, just like with the *Steady-state calculation* example, we can select one or more cells on this row and press *Run ZSF*. If all goes well, the output columns will then show the following results



BU	BV	BW	BX	BY	BZ
	State				
kg/m3					
Salinity to Sea	Sal Lock	Saltmass Lock	Head Lock	Volume Ship In Lock	
	15.0	136752	0.0	0	

Note: The lock is always initialized empty, i.e., without a ship in it.

Leveling to the lake side

The next step is to level the lock to the lake side. The lock was already initialized to the head of the lake side, so we expect to see no transports in this phase. Copy the input of the first row to the second row., and set

- the leveling time to 300 seconds
- the routine to **1**

Now press the *Run ZSF* button. Note that if you had any remaining values in the *Initialize State* columns, these values will be cleared as they are not needed. Check that the output columns have zero transport of both water and salt.

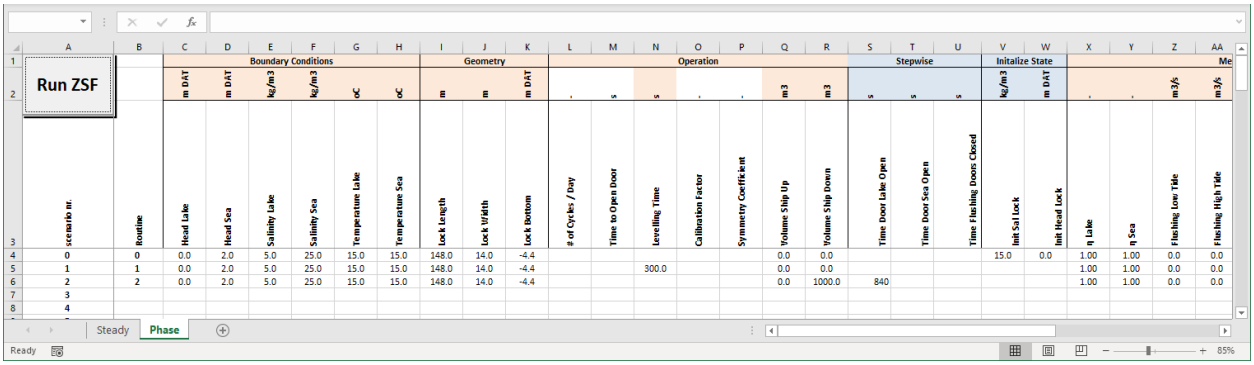
Opening the door to the lake side

With the lock leveled to the lake side, the doors can now be opened.

Important: Make sure that the leveling routines (**1** and **3**) have matching heads for the boundary conditions as the subsequent door-open routines (**2** and **4** respectively). If this is not the case, an exception is raised stating this requirement.

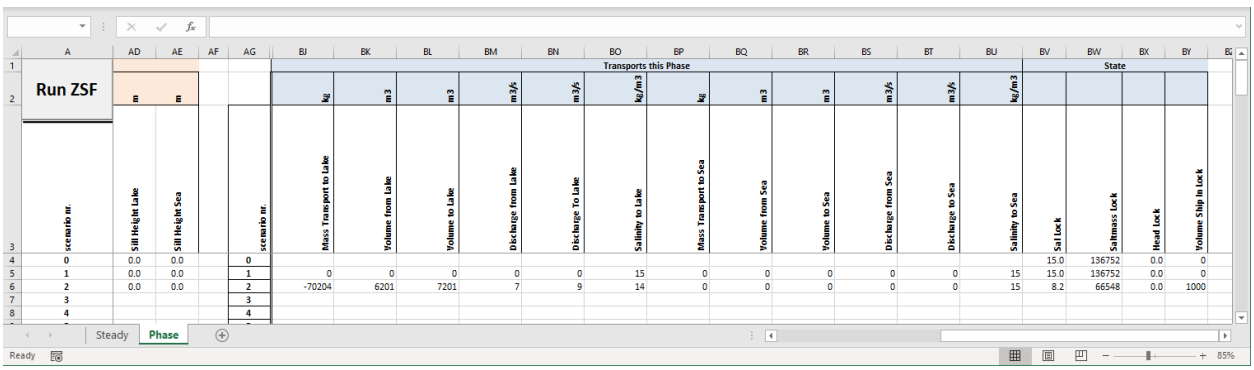
Once again, copy the lock dimensions and other inputs to a new row below the two already existing ones. Remove the leveling time, and set the following parameters:

- *volume ship down* (lake to sea) to 1000.0 m^3
- *door open time* on the lake side to 840 seconds
- the routine to **2**



Run ZSF			Boundary Conditions						Geometry			Operation								Stepwise			Initalize State						
			m DAT	m DAT	kg/m3	kg/m3	oC	oC	m	m	m DAT	-	s	s	-	-	m3	m3	s	s	s	kg/m3	m DAT	-	-	m3/s	m3/s
scenario nr.		Routine	Head Lake	Head Sea	Salinity Lake	Salinity Sea	Temperature Lake	Temperature Sea	Lock Length	Lock Width	Lock Bottom	# of Cycles / Day	Time to Open Door	Levelling Time	Calibration Factor	Symmetry Coefficient	Volume Ship Up	Volume Ship Down	Time Door Lake Open	Time Door Sea Open	Time Flashing Doors Closed	Init Sal Lock	Init Head Lock	η Lake	η Sea	Flashing Low Tide	Flashing High Tide
0		0	0.0	2.0	5.0	25.0	15.0	15.0	148.0	14.0	-4.4						0.0	0.0				15.0	0.0	1.00	1.00	0.0	0.0
1		1	0.0	2.0	5.0	25.0	15.0	15.0	148.0	14.0	-4.4			300.0			0.0	0.0						1.00	1.00	0.0	0.0
2		2	0.0	2.0	5.0	25.0	15.0	15.0	148.0	14.0	-4.4						0.0	1000.0	840					1.00	1.00	0.0	0.0
3																											
4																											

Press the *Run ZSF* button, and if all is correct the outputs should be similar to the following image:



Run ZSF					Transports this Phase												State			
		m	m		kg	m3	m3	m3/s	m3/s	kg/m3	kg	m3	m3	m3/s	m3/s	kg/m3				
scenario nr.		Sill Height Lake	Sill Height Sea	scenario nr.	Mass Transport to Lake	Volume from Lake	Volume to Lake	Discharge from Lake	Discharge To Lake	Salinity to Lake	Mass Transport to Sea	Volume from Sea	Volume to Sea	Discharge from Sea	Discharge to Sea	Salinity to Sea	Sal Lock	Salmass Lock	Head Lock	Volume Ship in Lock
0		0.0	0.0	0													15.0	136752	0.0	0
1		0.0	0.0	1	0	0	0	0	0	15	0	0	0	0	0	15	15.0	136752	0.0	0
2		0.0	0.0	2	-70204	6201	7201	7	9	14	0	0	0	0	0	15	8.2	66548	0.0	1000
3				3																
4				4																

The last columns show the state of the lock, and now indicate that there is a ship in the lock chamber.

Leveling to the sea side

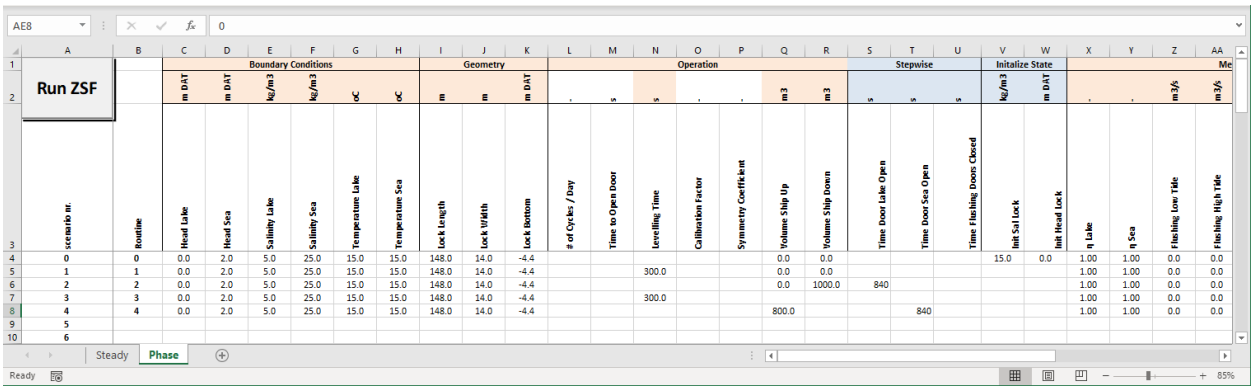
The next step is to level the lock to the sea side. The instructions are equal to those of leveling to the lake side, except that you should set the routine to **3** instead of **1**. Press the *Run ZSF* button, and inspect the output.

Opening the door to the sea side

The last step is to open the doors to the sea side, and let the ship sail out and a new ship sail in. Copy the last row to a new one, and set:

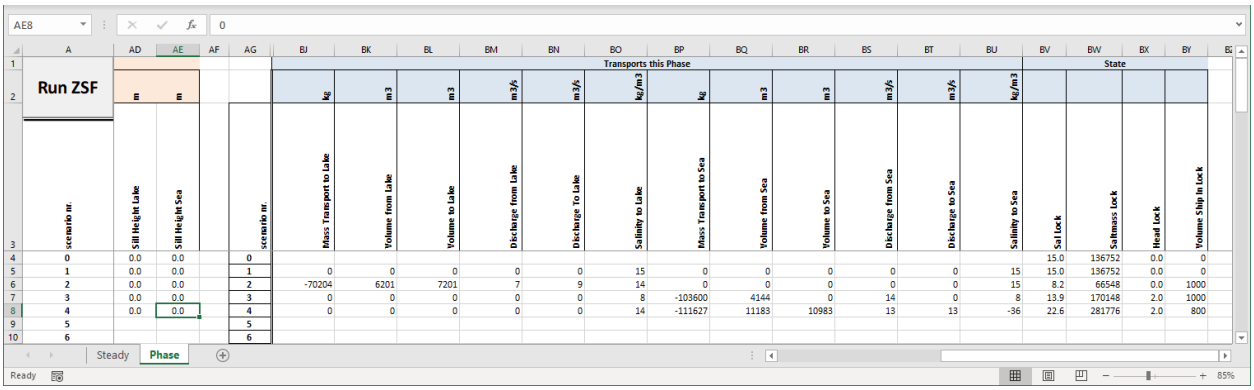
- *volume ship up* (sea to lake) to 800.0 m^3
- *door open time* on the sea side to 840 seconds
- the routine to **4**

The inputs should look as follows:



Run ZSF			Boundary Conditions						Geometry						Operation						Stepwise			Initalize State					
			m DAT	m DAT	kg/m3	kg/m3	oC	oC	m	m	m DAT	-	s	s	-	-	m3	m3	s	s	s	kg/m3	m DAT	-	-	m3/s	m3/s
scenario nr.	Routine		Head Lake	Head Sea	Salinity Lake	Salinity Sea	Temperature Lake	Temperature Sea	Lock Length	Lock Width	Lock Bottom	# of Cycles / Day	Time to Open Door	Levelling Time	Calibration Factor	Symmetry Coefficient	Volume Ship Up	Volume Ship Down	Time Door Lake Open	Time Door Sea Open	Time Flushing Doors Closed	Init Sal Lock	Init Head Lock	η Lake	η Sea	Flushing Low Tide	Flushing High Tide
0	0		0.0	2.0	5.0	25.0	15.0	15.0	148.0	14.0	-4.4						0.0	0.0				15.0	0.0	1.00	1.00	0.0	0.0
1	1		0.0	2.0	5.0	25.0	15.0	15.0	148.0	14.0	-4.4			300.0			0.0	0.0						1.00	1.00	0.0	0.0
2	2		0.0	2.0	5.0	25.0	15.0	15.0	148.0	14.0	-4.4						0.0	1000.0	840					1.00	1.00	0.0	0.0
3	3		0.0	2.0	5.0	25.0	15.0	15.0	148.0	14.0	-4.4			300.0										1.00	1.00	0.0	0.0
4	4		0.0	2.0	5.0	25.0	15.0	15.0	148.0	14.0	-4.4						800.0			840				1.00	1.00	0.0	0.0
5																											
6																											

After pressing *Run ZSF*, the output should like like:



Run ZSF				Transports this Phase												State			
	m	m		kg	m3	m3	m3/s	m3/s	kg/m3	kg	m3	m3	m3/s	m3/s	kg/m3				
scenario nr.	Sill Height Lake	Sill Height Sea	scenario nr.	Mass Transport to Lake	Volume from Lake	Volume to Lake	Discharge from Lake	Discharge To Lake	Salinity to Lake	Mass Transport to Sea	Volume from Sea	Volume to Sea	Discharge from Sea	Discharge to Sea	Salinity to Sea	Sal Lock	Saltmass Lock	Head Lock	Volume Ship in Lock
0	0.0	0.0	0													15.0	136752	0.0	0
1	0.0	0.0	1	0	0	0	0	0	15	0	0	0	0	0	15	15.0	136752	0.0	0
2	0.0	0.0	2	-70204	6201	7201	7	9	14	0	0	0	0	0	15	8.2	66548	0.0	1000
3	0.0	0.0	3	0	0	0	0	0	8	-103600	4144	0	14	0	8	13.9	170148	2.0	1000
4	0.0	0.0	4	0	0	0	0	0	14	-111627	11183	10983	13	13	-36	22.6	281776	2.0	800
5			5																
6			6																

Note that the volume of ship inside the lock chamber has changed from 1000 m^3 to 800.0 m^3.

Calculating more lockages

One can repeat the above process, again adding a row for leveling to the lake side next. If the water levels and salinities on the lake and/or sea side are changing, you can change these parameters accordingly. Typically, the water level is set to the average water level during the door-open phase, with the preceding leveling phase also leveling to said water level. It quickly becomes rather tedious and error-prone to calculate many lockages this way using Excel, especially if a lot of preprocessing is involved to get the parameters per locking phase and the source data is not set in stone. Depending on your experience with Excel and Python, it might then be easier to use the Python wrapper to do these types of calculations, see the Python *Phase-wise calculation* example.

2.2 Python Examples

2.2.1 Steady-state calculation

Overview

The purpose of this example is to understand the basic steps to calculate the salt transports through a shipping lock in a steady state operation. The scenario is the following: A single lock connects a canal to the sea, and the lock is busy during the day but quiet during the night. We want to know how much salt comes in on average, and figure out ways to reduce the salt intrusion by means of mitigating measures like bubble screens.

Properties of the lock and its operation

The physical dimensions of our single shipping lock are:

- length: 148 m
- width: 14 m
- bottom: -4.4 mNAP (Dutch Ordnance Datum)

For the boundary conditions we will assume that the sea level is the same as the canal level, with both being equal to 0.0 mNAP. The salinity on the salt side is significantly higher than on the canal side.

- salinities: 5 kg/m^3 on the canal side close to the lock, and 25 kg/m^3 on the sea side
- head: 0.0 mNAP on both sides
- temperature: 15.0 °C on both sides

The last step is to derive basic parameters from the locking information.

- During daytime, the lock operates at a pace of 1.25 cycles per hour (= 30 cycles per 24 hours). During the night, the lock operates at a pace of 10 cycles per 24 hours.

- It takes 5 minutes to open or close the doors on either side, and also 5 minutes to level.
- There are the same number of ships going from the sea to the canal, and vice-versa.
- The ships going to and from the canal also have an equal displacement of 1000 m^3.

In the first calculation, the lock does not have any sills or bubble screens, nor is there any flushing. These are of course measures that we will take a look at later in this example to reduce the amount of salt intrusion.

See also:

For an overview of these parameters, and more in-depth discussion on them, see *Input* and *Processes and definitions*.

Salt load without measures

The first step in your Python file or Jupyter Notebook is to import `pyzsf`:

```
import pyzsf
```

The next step is to enter all these physical and operational characteristics. For readability, it can make sense to group the parameters, as can be seen in the code:

```
lock_parameters = {
    "lock_length": 148.0,
    "lock_width": 14.0,
    "lock_bottom": -4.4,
}

boundary_conditions = {
    "head_lake": 0.0,
    "salinity_lake": 5.0,
    "temperature_lake": 15.0,
    "head_sea": 0.0,
    "salinity_sea": 25.0,
    "temperature_sea": 15.0,
}

operational_parameters = {
    "num_cycles": 30,
    "door_time_to_open": 300.0,
    "leveling_time": 300.0,
    "ship_volume_sea_to_lake": 1000.0,
    "ship_volume_lake_to_sea": 1000.0,
}
```

We can merge all dictionaries into one set for the daytime, and one set for the nighttime parameters. This makes passing the arguments later on a bit shorter and easier to understand.

```
daytime_parameters = {**lock_parameters, **boundary_conditions, **operational_parameters}
nighttime_parameters = {**daytime_parameters, "num_cycles": 10}
```

The next step is to actually calculate the salt flux to the lake during day- and nighttime. As we are assuming constant operation and boundary conditions, we can use the *`pyzsf.zsf_calc_steady()`* function. We log this salt flux to the console.

```
# Calculate the transports without protection
print("No measures:")
results = pyzsf.zsf_calc_steady(**daytime_parameters)
print("Day = {:.1f} kg/s".format(-1 * results["salt_load_lake"]))

results = pyzsf.zsf_calc_steady(**nighttime_parameters)
print("Night = {:.1f} kg/s".format(-1 * results["salt_load_lake"]))
```

The console output of these lines is as follows:

```
No measures:
Day = 36.8 kg/s
Night = 18.8 kg/s
```

Comparing salt intrusion measures

The maximum allowable salt load has been determined to be 12 kg/s, so the salt load during daytime and nighttime are currently not acceptable. The ZSF can help compare various salt intrusion measures that can be taken to reduce the salt load to acceptable levels. For this particular lock sills are not a feasible option, but bubble screens and flushing discharges could be.

A typical maximally efficient bubble screen can reduce the pace of the lock exchange to about 25%. We define a new dictionary where we set the density current factor to this percentage on both sides of the lock.

```
# Transports with a bubble screen
bubble_screen_parameters = {
    "density_current_factor_lake": 0.25,
    "density_current_factor_sea": 0.25,
}
```

We then call *pyzsf* again for both day- and nighttime operation, and pass these additional bubble screen parameters:

```
print("\nBubble screen (25%):")
results = pyzsf.zsf_calc_steady(**daytime_parameters, **bubble_screen_parameters)
print("Day = {:.1f} kg/s".format(-1 * results["salt_load_lake"]))

results = pyzsf.zsf_calc_steady(**nighttime_parameters, **bubble_screen_parameters)
print("Night = {:.1f} kg/s".format(-1 * results["salt_load_lake"]))
```

The console output of these lines is as follows:

```
Bubble screen (25%):
Day = 9.8 kg/s
Night = 13.4 kg/s
```

The salt load during daytime is reduced to about 10 kg/s with these bubble screens. However, the salt load at nighttime is reduced by much less. With a value of about 13.5 kg/s it is now even higher than that during the daytime, even though there are fewer ships passing through the lock.

Bubble screens are only effective if the doors are closed well before the (reduced) lock is reduced. With only 2 to 3 locking cycles during the night, the doors are open for more than an hour at a time. We can get these calculated door open times by also requesting the auxiliary results. This can be done by setting the first positional argument to *pyzsf.zsf_calc_steady()* to `True`.

```
# Auxiliary results, showing how long the doors are open
print("\nDoor open times at night:")
results = pyzsf.zsf_calc_steady(True, **nighttime_parameters, **bubble_screen_parameters)
for k, v in results.items():
    if k.startswith("t_open"):
        print(f"{k} = {v}")
```

The console output of these lines is as follows:

```
Door open times at night:
t_open = 3720.0
t_open_lake = 3720.0
t_open_sea = 3720.0
```

If we can tell the lock operator to close the doors right after ships have finished sailing out, we can reduce the salt intrusion significantly. To reduce these door-open durations, we can use the *zsf_param_t.calibration_coefficient*. If we know that the doors are open about 20 minutes at a time during the night, we can fill in a value of approximately 0.3 here to reduce the current duration of about an hour with.

```
# Transports at night with bubble screen and closing the doors sooner
print("\nBubble screen (25%), and close doors sooner:")
results = pyzsf.zsf_calc_steady(
    **nighttime_parameters, **bubble_screen_parameters, calibration_coefficient=0.3
)
print("Night = {:.1f} kg/s".format(-1 * results["salt_load_lake"]))
```

This gives us a salt load of about 4 kg/s during the night, which is acceptable:

```
Bubble screen (25%), and close doors sooner:
Night = 4.1 kg/s
```

The whole script

All together, the whole example script is as follows:

```
import pyzsf


lock_parameters = {
    "lock_length": 148.0,
    "lock_width": 14.0,
    "lock_bottom": -4.4,
}

boundary_conditions = {
    "head_lake": 0.0,
    "salinity_lake": 5.0,
    "temperature_lake": 15.0,
    "head_sea": 0.0,
    "salinity_sea": 25.0,
    "temperature_sea": 15.0,
}
```

(continues on next page)

(continued from previous page)

```python

operational_parameters = {
    "num_cycles": 30,
    "door_time_to_open": 300.0,
    "leveling_time": 300.0,
    "ship_volume_sea_to_lake": 1000.0,
    "ship_volume_lake_to_sea": 1000.0,
}

daytime_parameters = {**lock_parameters, **boundary_conditions, **operational_parameters}
nighttime_parameters = {**daytime_parameters, "num_cycles": 10}

# Calculate the transports without protection
print("No measures:")
results = pyzsf.zsf_calc_steady(**daytime_parameters)
print("Day = {:.1f} kg/s".format(-1 * results["salt_load_lake"]))

results = pyzsf.zsf_calc_steady(**nighttime_parameters)
print("Night = {:.1f} kg/s".format(-1 * results["salt_load_lake"]))

# Transports with a bubble screen
bubble_screen_parameters = {
    "density_current_factor_lake": 0.25,
    "density_current_factor_sea": 0.25,
}

print("\nBubble screen (25%):")
results = pyzsf.zsf_calc_steady(**daytime_parameters, **bubble_screen_parameters)
print("Day = {:.1f} kg/s".format(-1 * results["salt_load_lake"]))

results = pyzsf.zsf_calc_steady(**nighttime_parameters, **bubble_screen_parameters)
print("Night = {:.1f} kg/s".format(-1 * results["salt_load_lake"]))

# Auxiliary results, showing how long the doors are open
print("\nDoor open times at night:")
results = pyzsf.zsf_calc_steady(True, **nighttime_parameters, **bubble_screen_parameters)
for k, v in results.items():
    if k.startswith("t_open"):
        print(f"{k} = {v}")

# Transports at night with bubble screen and closing the doors sooner
print("\nBubble screen (25%), and close doors sooner:")
results = pyzsf.zsf_calc_steady(
    **nighttime_parameters, **bubble_screen_parameters, calibration_coefficient=0.3
)
print("Night = {:.1f} kg/s".format(-1 * results["salt_load_lake"]))
```

2.2.2 Phase-wise calculation

Overview

Note: This example focuses on performing a phase-wise calculation of salt-intrusion through a shipping lock. It assumes basic exposure to the Python interface. If you are a first-time user of the ZSF, see the *Steady-state calculation* example.

The purpose of this example is to understand the basic steps to perform a phase-wise calculation of the salt intrusion through a shipping lock. The scenario is similar to that of the *Steady-state calculation* example, in that there is a single lock connecting a canal to the sea. The differences with the steady state example are:

- there is a head difference now, with the sea at 2 mNAP
- we are only going to calculate one full locking cycle during the daytime operation

Initializing the lock

First we need to initialize the lock with a certain salinity and head. The dimensions of the lock are equal to that of the *Steady-state calculation* example. The boundary conditions are also equal, except that the `head_sea` is 2 mNAP. The operational parameters differ more, as the two steady-state operation parameters `num_cycles`, `leveling_time` and `door_time_to_open` are removed.

```
import pprint

import pyzsf


lock_parameters = {
    "lock_length": 148.0,
    "lock_width": 14.0,
```

(continues on next page)

(continued from previous page)

```
    "lock_bottom": -4.4,
}

boundary_conditions = {
    "head_lake": 0.0,
    "salinity_lake": 5.0,
    "temperature_lake": 15.0,
    "head_sea": 2.0,
    "salinity_sea": 25.0,
    "temperature_sea": 15.0,
}

operational_parameters = {
    "ship_volume_sea_to_lake": 1000.0,
    "ship_volume_lake_to_sea": 1000.0,
}
```

The next step is to initialize a *pyzsf.ZSFUnsteady* instance, with an initial salinity of 15 kg/m^3 and head of 0.0 mNAP. We also directly pass all other parameters to the constructor. These updated parameters are stored in the instance, and we do not need to specify them again when calling the method to level or open the doors (contrary to the Excel interface), unless we want to change the value of one of them of course.

```
print("State after initialization")
pprint.pprint(z.state)
```

The state of the lock after initialization is logged to the console with the `pprint.pprint` statements:

```
State after initialization
{'head_lock': 0.0,
 'salinity_lock': 15.0,
 'saltmass_lock': 136752.00000000003,
 'volume_ship_in_lock': 0.0}
```

Note: The lock is always initialized empty, i.e., without a ship in it.

Leveling to the lake side

The next step is to level the lock to the lake side, which is Phase 1. See Section 3.3 in the theory for more explanation on the phases in a locking cycle. The lock was already initialized to the head of the lake side, so we expect to see no transports in this phase. We call `pyzsf.ZSFUnsteady.zsf_step_phase_1`, which needs the leveling time as an argument.

```
results = z.step_phase_1(300.0)
```

The results of this method call, and the resulting state of the lock are printed to the console:

```
Phase 1:
********
Transports:
{'discharge_from_lake': 0.0,
```

(continues on next page)

(continued from previous page)

```
 'discharge_from_sea': 0.0,
 'discharge_to_lake': 0.0,
 'discharge_to_sea': 0.0,
 'mass_transport_lake': 0.0,
 'mass_transport_sea': 0.0,
 'salinity_to_lake': 15.0,
 'salinity_to_sea': 15.0,
 'volume_from_lake': 0.0,
 'volume_from_sea': 0.0,
 'volume_to_lake': 0.0,
 'volume_to_sea': 0.0}
State:
{'head_lock': 0.0,
 'salinity_lock': 15.000000000000002,
 'saltmass_lock': 136752.00000000003,
 'volume_ship_in_lock': 0.0}
```

Note that the console output shows zero transport of both water and salt. Also note that state of the lock after this phase is equal (barring rounding errors) to the state after initialization.

Opening the door to the lake side

With the lock leveled to the lake side, the doors can now be opened.

Important: Make sure that the leveling methods (**step_phase_1** and **step_phase_3**) have matching heads for the boundary conditions as the subsequent door-open methods (**step_phase_2** and **step_phase_4** respectively). If this is not the case, an exception is raised stating this requirement.

To open the doors, we call `pyzsf.ZSFUnsteady.zsf_step_phase_2`:

- *volume ship down* (lake to sea) is the default of 1000.0 m^3. We therefore do not need to pass this argument.
- *door open time* on the lake side to 840 seconds.

```
results = z.step_phase_2(840.0)
```

The transports are logged to the console. Note that there are no transports to the sea in this phase, as we would expect without any flushing discharge.

Leveling to the sea side

The next step is to level the lock to the sea side. The instructions are equal to those of leveling to the lake side, except that we call `pyzsf.ZSFUnsteady.zsf_step_phase_3`.

```
results = z.step_phase_3(300.0)
```

Opening the door to the sea side

The last step is to open the doors to the sea side, and let the ship sail out and a new ship sail in. This time we will also change the displacement of the ship that enters:

- override the *volume ship up* (sea to lake) to 800.0 m^3
- *door open time* on the sea side to 840 seconds

```
results = z.step_phase_4(840.0, ship_volume_sea_to_lake=800.0)
```

Check the state output that is logged to the console after this phase:

```
{'head_lock': 2.0,
 'salinity_lock': 22.612960757739405,
 'saltmass_lock': 281775.5814100392,
 'volume_ship_in_lock': 800.0}
```

Note that the volume of ship inside the lock chamber has changed from 1000 m^3 to 800.0 m^3.

Calculating more lockages

One can repeat the above process, again adding a method call to `pyzsf.ZSFUnsteady.zsf_step_phase_1` for leveling to the lake side next. If the water levels and salinities on the lake and/or sea side are changing, you can change these parameters accordingly for each method call. Typically, the water level is set to the average water level during the door-open phase, with the preceding leveling phase also leveling to said water level. It quickly becomes rather tedious and error-prone to calculate many lockages this way, and it is better to write a loop over some input data. For an example of this, see *Phase-wise with multiple lockages* example.

The whole script

All together, the whole example script is as follows:

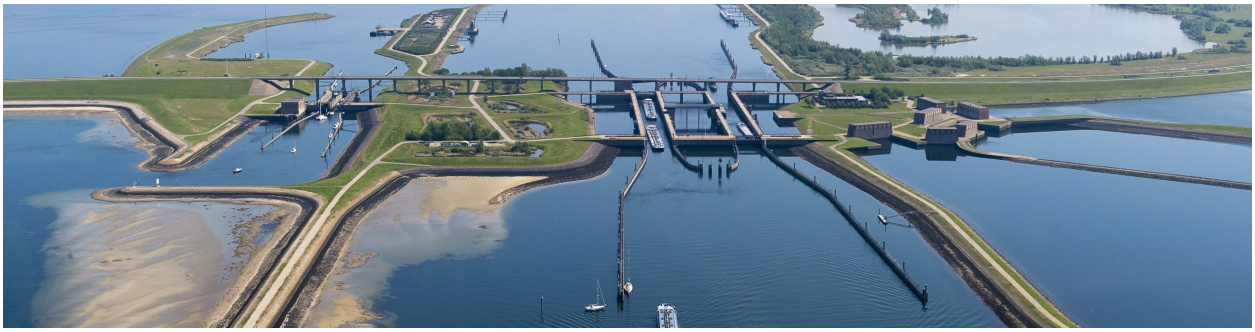
```
import pprint

import pyzsf


lock_parameters = {
    "lock_length": 148.0,
    "lock_width": 14.0,
    "lock_bottom": -4.4,
}

boundary_conditions = {
    "head_lake": 0.0,
    "salinity_lake": 5.0,
    "temperature_lake": 15.0,
    "head_sea": 2.0,
    "salinity_sea": 25.0,
    "temperature_sea": 15.0,
}

```

(continues on next page)

(continued from previous page)

```
operational_parameters = {
    "ship_volume_sea_to_lake": 1000.0,
    "ship_volume_lake_to_sea": 1000.0,
}

z = pyzsf.ZSFUnsteady(15.0, 0.0, **lock_parameters, **boundary_conditions, **operational_
↪parameters)

print("State after initialization")
pprint.pprint(z.state)

print("\nPhase 1:\n" + "*" * 8)

results = z.step_phase_1(300.0)
print("Transports:")
pprint.pprint(results)
print("State:")
pprint.pprint(z.state)

print("\nPhase 2:\n" + "*" * 8)
results = z.step_phase_2(840.0)
print("Transports:")
pprint.pprint(results)
print("State:")
pprint.pprint(z.state)

print("\nPhase 3:\n" + "*" * 8)
results = z.step_phase_3(300.0)
print("Transports:")
pprint.pprint(results)
print("State:")
pprint.pprint(z.state)

print("\nPhase 4:\n" + "*" * 8)
results = z.step_phase_4(840.0, ship_volume_sea_to_lake=800.0)
print("Transports:")
pprint.pprint(results)
print("State:")
pprint.pprint(z.state)
```

2.2.3 Phase-wise with multiple lockages



Overview

Note: This example focuses on performing a phase-wise calculation of many lockages based on an input file. It assumes basic exposure to the Python interface and phase-wise calculation. If you are a first-time user of the ZSF, see the *Steady-state calculation* and *Phase-wise calculation* examples.

The purpose of this example is to understand the basic steps to perform a phase-wise calculation of many lockages. The goal is to determine the average transports of water and salt over a 60-day period.

Initializing the lock

When initializing the lock with `pyzsf.ZSFUnsteady` we pass all parameters that stay constant throughout the 60-day period. This includes the parameters for salt intrusion measures like the bubble screens. Note that the head on the lake side is constant, but that the head on the sea side varies. Furthermore, the salinity on both sides of the lock varies over time as well.

```
lock_parameters = {
    "lock_length": 300.0,
    "lock_width": 25.0,
    "lock_bottom": -7.0,
}

constant_boundary_conditions = {
    "head_lake": 0.0,
    "temperature_lake": 15.0,
    "temperature_sea": 15.0,
}

mitigation_parameters = {
    "density_current_factor_lake": 0.25,
    "density_current_factor_sea": 0.25,
    "distance_door_bubble_screen_lake": 10.0,
    "distance_door_bubble_screen_sea": 10.0,
    "flushing_discharge_high_tide": 0.0,
    "flushing_discharge_low_tide": 0.0,
    "sill_height_lake": 0.5,
}
```

(continues on next page)

(continued from previous page)

```

# Initialize the lock
z = pyzsf.ZSFUnsteady(
```

Reading the input data

The lockages over the 60-day period are defined in a CSV-file:

Table 1: Lockages

time	head_sea	routine	salinity_lake	salinity_sea	ship_volume_lake_to_sea	ship_volume_sea_to_lake	t_flushing	t_level	t_open_lake	t_open_sea
2960	0.03760156	99	33.33333333	28.57145202582857142				300.0		
3380	0.03760156	99	33.33333333	28.57145202582857142	1884.2					420.0
3920	0.18119711	2 1	0.89790309	28.35574497857142 6				240.0		
4280	0.18119711	2 2	0.89790309	28.35574497857142 6					1020.0	
5420	0.74484631	3	1.02360534	728.62057304				340.0		

We read this CSV file using pandas, and convert it to a list of parameter dictionaries:

```
)

# Read the lockages from a file
```

Stepping through all lockages

Just like in the *Phase-wise calculation* example we step through all phases in the locking cycle. We do this by iterating over all locking-phase entries defined in the input CSV file. Depending on the respective phase, we pass either the leveling time, the door-open duration, or the duration of flushing (with the doors closed).

We store the results of every individual locking phase in a list called `all_results`, to be aggregated later on in the script:

```
lockages = list(df_lockages.to_dict("records"))

# Go through all lockages
all_results = []

for parameters in lockages:
    routine = int(parameters.pop("routine"))
    t_open_lake = parameters.pop("t_open_lake")
    t_open_sea = parameters.pop("t_open_sea")
    t_level = parameters.pop("t_level")
    t_flushing = parameters.pop("t_flushing")

    parameters["ship_volume_sea_to_lake"] = 0.0
    parameters["ship_volume_lake_to_sea"] = 0.0

    if routine == 1:
        assert t_level > 0
        results = z.step_phase_1(t_level, **parameters)
```

(continues on next page)

(continued from previous page)

```
    elif routine == 2:
        results = z.step_phase_2(t_open_lake, **parameters)
    elif routine == 3:
        assert t_level > 0
        results = z.step_phase_3(t_level, **parameters)
    elif routine == 4:
        results = z.step_phase_4(t_open_sea, **parameters)
    elif routine in {-2, -4}:
        results = z.step_flush_doors_closed(t_flushing, **parameters)
    else:
        raise Exception(f"Unknown routine '{routine}'")
```

Aggregating output

For many cases we would only be interested in what happens on the lake side, e.g. the average salt flux in *kg/s* over a certain period of time. For illustrative purposes we however aggregate all outputs. For volumes and mass transports this means summing them. For discharges and salt fluxes, this means averaging them.

```
    all_results.append(results)

# Aggregate results
duration = 60 * 24 * 3600  # 60 days

overall_results = {}
overall_mass_to_sea = 0.0
overall_mass_to_lake = 0.0

for results in all_results:
    for k, v in results.items():
        if k.startswith(("volume_", "mass_")):
            overall_results[k] = overall_results.get(k, 0.0) + v

    overall_mass_to_sea += results["volume_to_sea"] * results["salinity_to_sea"]
    overall_mass_to_lake += results["volume_to_lake"] * results["salinity_to_lake"]

overall_results["salinity_to_sea"] = overall_mass_to_sea / overall_results["volume_to_sea
↪"]
overall_results["salinity_to_lake"] = overall_mass_to_lake / overall_results["volume_to_
↪lake"]

overall_discharges = {}
for k, v in overall_results.items():
    if k.startswith("volume_"):
        overall_discharges[f"discharge_{k[7:]}"] = v / duration
overall_results.update(overall_discharges)
```

Finally, the average fluxes are logged to the console

```
assert overall_results.keys() == all_results[0].keys()

# Log to console
```

The output should show something like:

```
Overall results (60 day aggregates and averages):
{'discharge_from_lake': 2.626938120554457,
 'discharge_from_sea': 2.9855208663562096,
 'discharge_to_lake': 2.7409909136666557,
 'discharge_to_sea': 2.872713264556223,
 'mass_transport_lake': -215507325.24071646,
 'mass_transport_sea': -215288300.05404428,
 'salinity_to_lake': 16.334431315958305,
 'salinity_to_sea': 13.297877648040927,
 'volume_from_lake': 13618047.216954306,
 'volume_from_sea': 15476940.17119059,
 'volume_to_lake': 14209296.896447944,
 'volume_to_sea': 14892145.563459458}
```

The whole script

All together, the whole example script is as follows:

```python
import pprint

import pandas as pd

import pyzsf


lock_parameters = {
    "lock_length": 300.0,
    "lock_width": 25.0,
    "lock_bottom": -7.0,
}

constant_boundary_conditions = {
    "head_lake": 0.0,
    "temperature_lake": 15.0,
    "temperature_sea": 15.0,
}

mitigation_parameters = {
    "density_current_factor_lake": 0.25,
    "density_current_factor_sea": 0.25,
    "distance_door_bubble_screen_lake": 10.0,
    "distance_door_bubble_screen_sea": 10.0,
    "flushing_discharge_high_tide": 0.0,
    "flushing_discharge_low_tide": 0.0,
    "sill_height_lake": 0.5,
}

# Initialize the lock
z = pyzsf.ZSFUnsteady(
    15.0, 0.0, **lock_parameters, **constant_boundary_conditions, **mitigation_parameters
```

(continues on next page)

(continued from previous page)

```
)

# Read the lockages from a file
df_lockages = pd.read_csv("lockages.csv", index_col=0)
lockages = list(df_lockages.to_dict("records"))

# Go through all lockages
all_results = []

for parameters in lockages:
    routine = int(parameters.pop("routine"))
    t_open_lake = parameters.pop("t_open_lake")
    t_open_sea = parameters.pop("t_open_sea")
    t_level = parameters.pop("t_level")
    t_flushing = parameters.pop("t_flushing")

    parameters["ship_volume_sea_to_lake"] = 0.0
    parameters["ship_volume_lake_to_sea"] = 0.0

    if routine == 1:
        assert t_level > 0
        results = z.step_phase_1(t_level, **parameters)
    elif routine == 2:
        results = z.step_phase_2(t_open_lake, **parameters)
    elif routine == 3:
        assert t_level > 0
        results = z.step_phase_3(t_level, **parameters)
    elif routine == 4:
        results = z.step_phase_4(t_open_sea, **parameters)
    elif routine in {-2, -4}:
        results = z.step_flush_doors_closed(t_flushing, **parameters)
    else:
        raise Exception(f"Unknown routine '{routine}'")

    all_results.append(results)

# Aggregate results
duration = 60 * 24 * 3600  # 60 days

overall_results = {}
overall_mass_to_sea = 0.0
overall_mass_to_lake = 0.0

for results in all_results:
    for k, v in results.items():
        if k.startswith(("volume_", "mass_")):
            overall_results[k] = overall_results.get(k, 0.0) + v

    overall_mass_to_sea += results["volume_to_sea"] * results["salinity_to_sea"]
    overall_mass_to_lake += results["volume_to_lake"] * results["salinity_to_lake"]

overall_results["salinity_to_sea"] = overall_mass_to_sea / overall_results["volume_to_sea
```

(continues on next page)

(continued from previous page)

```
↪"]
overall_results["salinity_to_lake"] = overall_mass_to_lake / overall_results["volume_to_
↪lake"]

overall_discharges = {}
for k, v in overall_results.items():
    if k.startswith("volume_"):
        overall_discharges[f"discharge_{k[7:]}"] = v / duration
overall_results.update(overall_discharges)

assert overall_results.keys() == all_results[0].keys()

# Log to console
print("Overall results (60 day aggregates and averages):")
pprint.pprint(overall_results)
```

CHAPTER

THREE

THEORY

3.1 Introduction

3.1.1 Motivation and goals

The Directorate-General for Public Works and Water Management (Dutch: Rijkswaterstaat), but also other private and public parties around the world in charge of water reserves, would like to calculate the influence of the salt intrusion through shipping locks on the salinity of the fresh water at a reasonable distance from the lock. For example, to calculate the salinity of the water that farmers use to irrigate their land. Such calculations should be able to consider a long period of time and area, as the spread and build-up of salt can take months or years. It is also necessary to be able to consider multiple economical- and climate scenarios, along with the possible measures both on the shipping lock itself and those elsewhere in the system. For this reason, there is a need to have a fast and compact formulation of the processes on the lock, such that this formulation can be used inside a far-field model that models the spread. Such a formulation can then also be used to calculate the transport of water and salt through a shipping lock in a stand-alone fashion (i.e. without coupling it to- or incorporating it in a far-field model).

3.1.2 Approach

The chosen approach for the ZSF is to set up the equations that describe the flows into and out of the lock, for all phases of the locking cycle. Concretely, this means the volumes due leveling, the lock exchange process, ship displacement, and flushing discharges through the lock chamber. Note that this phase-wise approach is opposed to a fixed time-step approach.

In case registrations of the door movements are available, one can calculate per locking phase what the transports of water and salt are in that particular phase over both lock heads (salt side and fresh side). That way one can calculate based on historical registrations what the salt transport has been.

When no registrations are available, a cycle-averaged transport of salt and water can be calculated based on a few key parameters of the lock operation. These parameters are then translated into a repeating pattern of locking phases, i.e. when the door is open and how long leveling is supposed to take. Then, the transports of water and salt per phase can be added up, leading to cycle-averaged values of the transports. These averaged values can then serve as in a forcing in a far-field model.

An overview of the concept of the ZSF model is given in figure below. A more detailed discussion of all the physical quantities involved will be explained later. What is visible is that the boundary conditions of the model are the temperature, salinity and water level on each side of the lock. These boundary conditions, together with the geometry and operation of the lock, determine the discharge that go into the lock with the governing salinity, and out of the lock with a salinity in the lock chamber that follows from the locking process. The flows into the lock chamber are a withdrawal from the approach harbor on each side of the lock, Q_F^- and Q_S^-, and the flows going out of the lock are discharges to the approach harbors, Q_F^+ and Q_S^+.

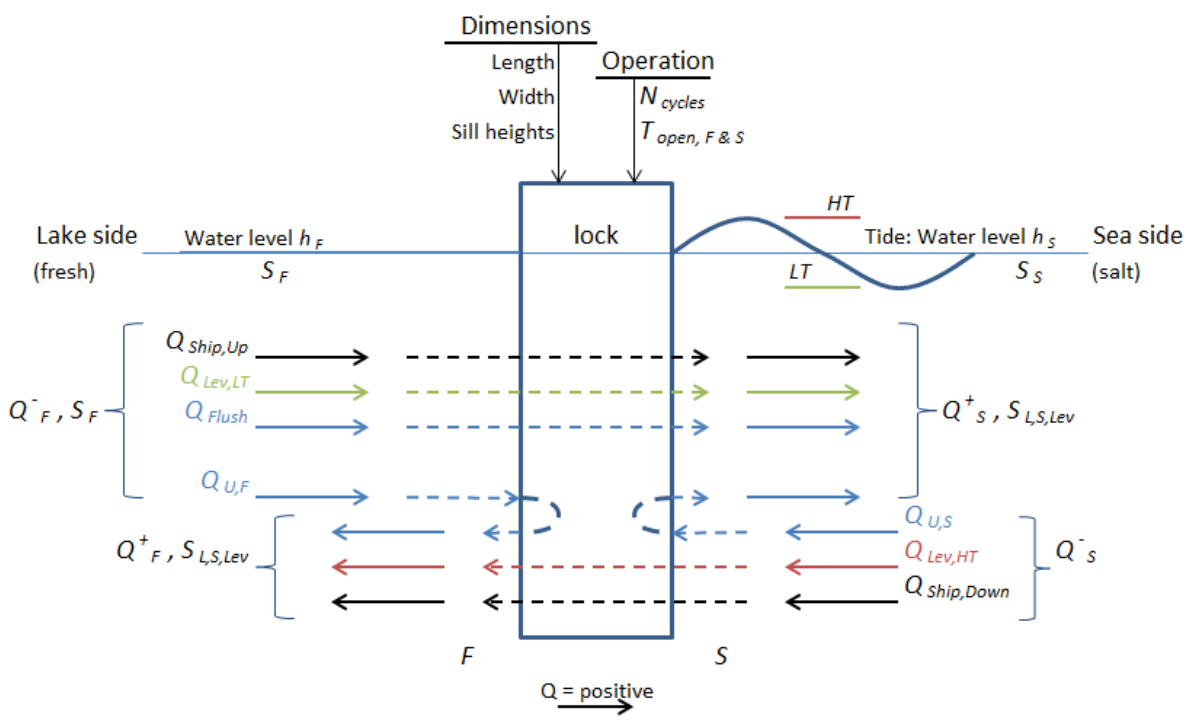


Fig. 1: Schematic overview of the model concept: discharges going in and out on both sides of the lock, with their corresponding salinities.

3.2 Processes and definitions

To set up the equations, it is necessary to define the various physical quantities. An understanding of these physical quantities should be based on an understanding of the relevant processes. Therefore, we first (briefly) introduce these processes, followed by explaining the quantities that will appear in the equations later on.

3.2.1 Processes in the locking cycle

Leveling

The locking of ships is in essence about overcoming a difference in water level. By letting a ship moore temporarily in a lock chamber, and by changing the water level in this lock chamber, the water level difference can be bridged. This process is called leveling. Raising the water level is done by letting water from the high side ('filling'), and lowering the water level is done by letting out water to the low side ('emptying').

Lock exchange

Where there is a difference in salinity - and with that a difference in density - between both sides of the lock, a current will start to develop when the doors on either end are opened. This current is called a density current, and the phenomenon of this current exchanging the salt water in the lock chamber with fresh water (or vice versa) is called 'lock exchange'. The less dense fresh water will start to float on top of the denser salt water, and move to the salt side. After reflection of the density current on the closed doors, the process continuous until just about the entire lock volume has been exchanged or until the doors are closed.

Locking operation

The locking operation concerns the opening and closing of doors on one side, leveling, the opening and closing of the doors on the other side, and leveling back again. When the doors open, ships will leave the lock chamber, before new ones going the opposite direction enter the lock chamber and the doors close again. The discharge through the lock due to leveling is determined by the surface area of the lock chamber, the water level difference, and the number of locking cycles per unit of time. For the lock exchange process, it is the time that the doors are open that matters. The time that the doors are open determines whether the density current partially or fully exchanges the water in the lock chamber.

Shipping

The presence of ships in the locking cycle influences the lock exchange in two ways:

1. When a ship enters a lock chamber, water is pushed out of said lock chamber. The amount of water that leaves the lock is equal to the water displacement of the ship. Conversely, when ships exit the lock chamber, water will flow from the approach harbor into the lock chamber to fill the 'hole' left by the ships.
2. The upwards leveling of a lock chamber means that water comes in from the high side. In case of a lock between salt- and fresh water, it is likely that the water that enters has a different salinity than the water inside the lock chamber. At the end of leveling the lock chamber will therefore have a new average salinity. When determining this average salinity we need to take into account that, when ships are present in the lock, there is less water inside the lock chamber.

The presence of ships inside the lock chamber does not have influence on the amount of water that is needed for leveling.

In the formulation of transports through the lock it is not about the individual but about the total water displacement of the ships per cycle. This value can differ for both direction, upstream and downstream. Do note that these two values do not have to be equal to each other. For example, when through a certain lock more cargo is imported than exported, this

will be represented accordingly in the water displacements in both directions. With that, also a net discharge through the lock can arise.

Measures to prevent salt intrusion

To reduce the salt intrusion through shipping locks, a number of measures are available.

Bubble screens

Bubble screen in the lock heads reduce the velocity of the density current while the doors are open. It can therefore be an effective measure to reduce the salt intrusion, but the reduction also depends on limiting the amount of time the doors are open.

Flushing through the lock chamber

If the water level on the salt side is lower than that on the fresh side, the salt intrusion can be reduced by flushing the lock chamber with fresh water. When the lock opens to the fresh side, the water inside the lock chamber will have a lower salinity. This causes the density current velocity to drop, and also reduces the maximum amount of salt that can be transported due to lock exchange. When flushing with doors open, the flushing discharge with also reduce the velocity of the density current by means of superposition of velocities. Note that the same principle of flushing the lock can be applied when the water level on the salt side is higher, but pumps will be needed to realize the desired flushing discharge.

3.2.2 Conventions

For the positive direction of flows the dominant flow direction of the river or delta is chosen, i.e. downstream, towards the sea, from fresh to salt. This is in line with the convention of other software and model schematizations. Due to this choice, salt intrusion is a transport of salt in the negative direction.

This convention is also used in the figures in this documentation: a positive discharge means flow from left to right, with the upstream side on the left and the sea/salt side on the right.

Subscripts to denote the lock chamber, the fresh side, and the salt side are respectively L, F, and S.

3.2.3 Phases of the locking cycle

In the figure below the different phases of the locking cycle are displayed and named. Because the direction of the leveling depends on whether the water level on the salt or fresh side is higher, there is a high water and low water state.

The water levels (h) on each side of the lock are defined as h_F for the water level on the fresh side, and h_S for the water level on the salt side. From this the definition of the high and low water state follows:

- when $h_S h_F$ the state is LW
- when $h_S \geq h_F$ the state is HW

In this figure the positive direction is shown for the discharge (Q), but the same convention holds for the transported volume (V) and transported salt mass (M).

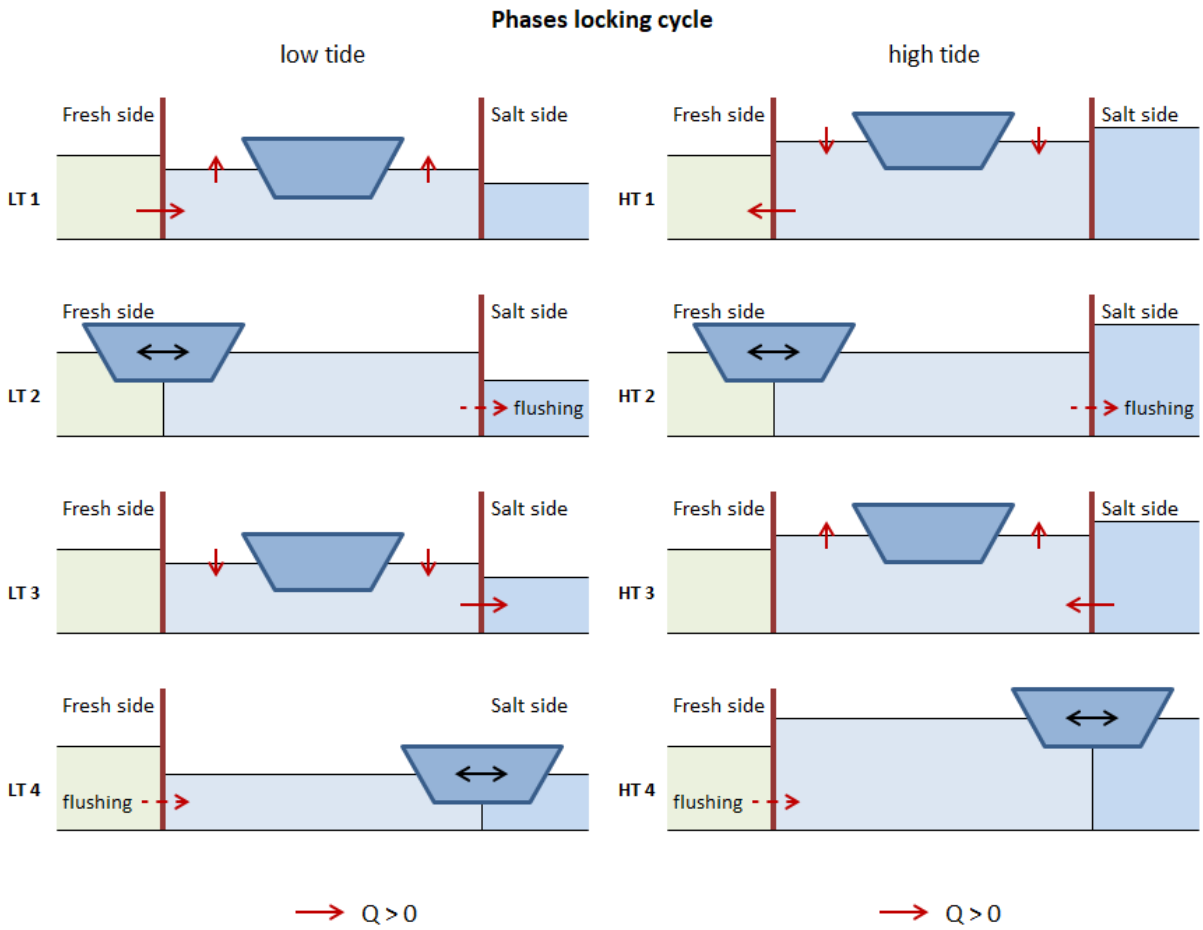


Fig. 2: The phases of the locking cycle

3.2.4 Physical quantities

The transports through a lock are determined by four categories of physical quantities:

- The boundary conditions on either side of the lock
- The geometry of the lock
- The operation of the lock
- The dimensions of the ship(s)

The calculated transports are expressed in terms of a transported salt mass per unit of time (or locking cycle), determined by the transported volumes and their corresponding salinities.

Boundary conditions

The boundary conditions on either side of the lock, the fresh side and the salt side, consist of a water level with a corresponding salinity or density. In stand-alone operation, these boundary conditions are prescribed by the user. When part of a larger (far-field) model, said model will prescribe the boundary conditions to the ZSF.

h: The water levels on each side of the lock in [mDAT]

Suffix: L, F, and S for the lock, fresh side and salt side.

S: Salinity in [kg/m^3]

Suffix: L, F, and S for the lock, fresh side and salt side.

Second suffix for S_L:

F: direction after closing the door on the fresh side

S: direction after closing the door on the salt side

Third suffix for S_L or S_S:

Lev: after leveling, that follows on closing the door on the other side

$\overline{\rho}$: The average density of the water on each side of the lock head in [kg/m^3]

These values are generally available from a far-field model. In stand-alone usage, the density is calculated from the salinity using the UNESCO-formulation.

$\overline{\rho}_{FS}$: The average density of the water on the fresh and salt side in [kg/m^3]

These values are generally available from a far-field model. In stand-alone usage, the density is calculated from the salinity using the UNESCO-formulation.

Geometry

The general geometric parameters of the lock are:

L_L: length of the lock chamber in [m]

W_L: width of the lock chamber in [m]

z_L: bottom depth of the lock chamber in [mDAT]

From these follow:

H: water level over the bottom of the lock chamber or lock head in [m].

$$H_F = h_F - z_L$$
$$H_S = h_S - z_L \tag{3.1}$$

V: the volume of the lock chamber in [m^3] at a certain water level

$$V_{L,F} = L_L W_L H_F$$
$$V_{L,S} = L_L W_L H_S \tag{3.2}$$

Difference in the lock bottom layout inside and outside the lock chamber, combined with the possible presence of sills in the lock heads, will influence the lock exchange. For the correct calculation thereof, additional parameters are necessary. That is, it is necessary to define _effective_ values of length, depth and volume. This is discussed in more detail in TODO_REFCHAP7.

If the lock bottom is equal throughout and there are no sills in the lock heads, it holds that:

$$L_{L,eff} = L_L \tag{3.3}$$

and

$$H_{F,eff} = H_F$$
$$H_{S,eff} = H_S \tag{3.4}$$

and

$$V_{L,F,eff} = V_{L,F}$$
$$V_{L,S,eff} = V_{L,S} \tag{3.5}$$

Operation

T_{in-use}: hours per day that the lock is operational [hour]

This is a constant with a value of 24 hours.

N_{cycles}: locking frequency [-]

The number of locking cycles (back and forth) per day.

T_{door}: time needed to move (open or close) the doors [min]

T_{level}: the average time needed to level the lock [min]

Research has shown that a variation in locking frequency over the day or week influences the salt intrusion significantly. To take this effect into account, two additional parameters are introduced:

c_{dot}: a calibration coefficient as a factor on the door-open time [-]

Default value is 1 (conservative); the range is $[0, 1]$

$C_{F/Avg}$: a symmetry coefficient indicating whether the doors are open equally long on both sides [-]

Default is 1 (equally long). Range is 0 (door on fresh side effectively never open) to 2 (door on salt side effectively never open).

From these parameters follow:

T_{cycle}: the average time for a complete locking cycle [s]

$$T_{cycle} = \frac{T_{in-use} \cdot 60 \cdot 60}{N_{cycles}} \tag{3.6}$$

$T_{open,avg}$: the average door-open time [s]

$$T_{open,avg} = \frac{1}{2}T_{cycle} - \left(T_{level} + \frac{2}{2}T_{door}\right) \cdot 60 \tag{3.7}$$

Do note that for the calculation of the door-open time we subtract twice half the time it takes to open- or close the door. That means that effectively we assume that the lock-exchange process starts and ends when the doors are opened/closed halfway.

T_{open}: the representative door-open time [s]

$$T_{open} = c_{dot}T_{open,avg} \tag{3.8}$$

$T_{open,F}$ and $T_{open,S}$: the representative door-open times on the fresh and salt side [s]

$$\begin{aligned} T_{open,F} &= c_{F/Avg}T_{open} \\ T_{open,S} &= \left(2 - c_{F/Avg}\right)T_{open} \end{aligned} \tag{3.9}$$

Transports

M The amount of transported salt [kg] per locking phase (over a certain head)

first suffix: F, and S for the lock head on the fresh side or salt side.

second suffix: $LT1$ - $LT4$: phase of the locking cycle, simplified to just *1-4* if the formula is equal for both tidal phases.

V a volume of water with salt that is moved between lock and approach harbor [m^3]

suffixes:

Lev: due to leveling

followed by either *LT* or *HT*, depending on the direction of the head difference over the lock

U: due to lock exchange

followed by F, and S for fresh side or salt side

$Ship$: the total water displacement of the ships

Up: For ships moving in upstream direction, i.e. from salt to fresh side

Down: For ships moving in downstream direction, i.e. from fresh to salt side

$Flush$: Due to a flushing discharge through the locking chamber

followed by either *LT* or *HT*, depending on the direction of the head difference over the lock

3.2.5 Calculation of the transported volumes

Leveling volume

The leveling volume plays a.o. a role in filling the lock chamber. Note that when filling the lock chamber, the salinity in the lock chamber changes. When emptying (i.e. leveling to the side with lower level), the salinity in the lock does not change.

Because the expression for filling and emptying are different, the formulas are separated. In addition, it is convenient to have different identifiers for both tidal phases.

$$LT : V_{level,LT} = L_L W_L \left(h_F - h_S\right), V_{level,HT} = 0 \tag{3.10}$$

$$HT : V_{level,LT} = 0, V_{level,HT} = L_L W_L \left(h_S - h_F\right) \tag{3.11}$$

Lock exchange

The lock exchange is often the most import process for lock intrusion through shipping locks. The process of lock exchange in time can be approximated with a hyperbolic tangent function. With that, the exchange volume $U(= V_U/V_K)$, slowly goes to 1 when the doors stay open for a long period of time.

$$V_U = V_L U = V_L \tanh\left(\frac{T_{open}}{T_{LE}}\right) \tag{3.12}$$

The door-open time T_{open} is in this expression compared to the (theoretical) time it takes for the density current to travel twice the length of the lock chamber.

The (initial) velocity of the density current c_i is determined by the relative density difference $\Delta\rho/\overline{\rho}$ and the water level in the respective head H

$$c_i = \frac{1}{2}\sqrt{g'H} = \frac{1}{2}\sqrt{g\frac{\Delta\rho}{\overline{\rho}}H} \approx \frac{1}{2}\sqrt{g0.8\frac{\Delta S}{\overline{\rho}_{FS}}H} \tag{3.13}$$

The difference in density can be approximated as 0.8 times the difference in salinity (in kg/m^3) between the lock and the approach harbor. For $\overline{\rho}$ a value could be deduced from both sides of the lock head at time of opening. Using a fixed value based on the boundary conditions (the average density over the lock $\overline{\rho}_{FS}$) introduces only a small approximation error.

With this density current velocity, T_{LE} can be defined as:

$$T_{LE} = \frac{2L}{c_i} = \frac{4L}{\sqrt{g\frac{0.8\Delta S}{\overline{\rho}_{FS}}H}} \tag{3.14}$$

The presence of sills and levels difference on the bottom of the lock chamber can influence the lock exchange. In this case, the values for the length, volume and water level depth will need to be adjusted.

Bubble screens

The effectiveness of a bubble screen is often expressed as a factor on the velocity of the density current η. This density-current factor is a number in the interval [0, 1], the more effective a bubble screen the lower this value is (with a limit of about 0.2 - 0.25). With the velocity of the density current decreasing, the time it takes for a complete lock exchange increases. The factor can then be easily introduced in equation (3.12) describing the amount of volume exchanged due to lock exchange:

$$V_U = V_L U = V_L \tanh\left(\frac{\eta T_{open}}{T_{LE}}\right) \tag{3.15}$$

Note: This way of introducing η is elegant, but does imply that the only the velocity of the density current is changed, and e.g. the salinity of the density current is not. In reality the bubble screen will cause a lot of mixing to occur, and therewith the salinities of lock exchange to change.

3.2.6 Flushing discharge

A flushing discharge always goes from the fresh to the salt side. Such a discharge would generally only be possible in case of low-tide. In the high-tide case, a pump would be needed, and therefore the flushing discharge at high tide is often zero in practice. From an operational standpoint, there is a certain maximum instantaneous discharge. This is the discharge that is acceptable for the safe and speedy entry and exit of ships from the lock chamber. This momentaneous value (to be distinguished from the cycle-averaged discharge) is input by the user, with separate values for LT and HT.

$$LT : Q_{flush} = Q_{flush,LT} \tag{3.16}$$

$$HT : Q_{flush} = Q_{flush,HT} \tag{3.17}$$

In many cases there is only a discharge when either lock head is open. It is also possible to flush with both lock heads closed, but the leveling system still open. For now, we will assume the latter does not happen. That assumption then leads to the following equation for the flushed volume:

$$V_{flush} = Q_{flush} \left(T_{open,F} + T_{open,S} \right) \tag{3.18}$$

In practice, flushing is temporarily stopped when ships are sailing in from the fresh side to let them safely moor. This distinction cannot be made in the ZSF, so flushing will happen for the entire door-open time.

The flushing discharge interacts with the density current and ships. For now, we will neglect the interaction of the flushing discharge and density current with the ships. In other words, in the discussion below, we will assume that the lock chamber does not contain any ships.

Generally we can assume a simple superposition of velocities when considering the interaction between flushing and lock exchange. This has consequences for the maximum exchangeable volume by the density current, which will discussed below in more detail using four scenario's. These scenario's differ in the magnitude of the flushing velocity (relative to the velocity of the density current):

1. No flushing discharge
2. Velocity of the flushing discharge is lower than that of the density current: $v_{flush} = 0.5 \cdot c_i$
3. Velocity of the flushing discharge is equal to that of the density current: $v_{flush} = c_i$
4. Velocity of the flushing discharge is larger than that of the density current: $v_{flush} = 1.5 \cdot c_i$

For every scenario we consider the entering (density) current, the reflecting, and the steady state condition at (infinitely) long door-open times.

Because the interaction between the flushing discharge and density current differs for the fresh side and salt side, they are discussed separately.

Flushing fresh side

Entering current

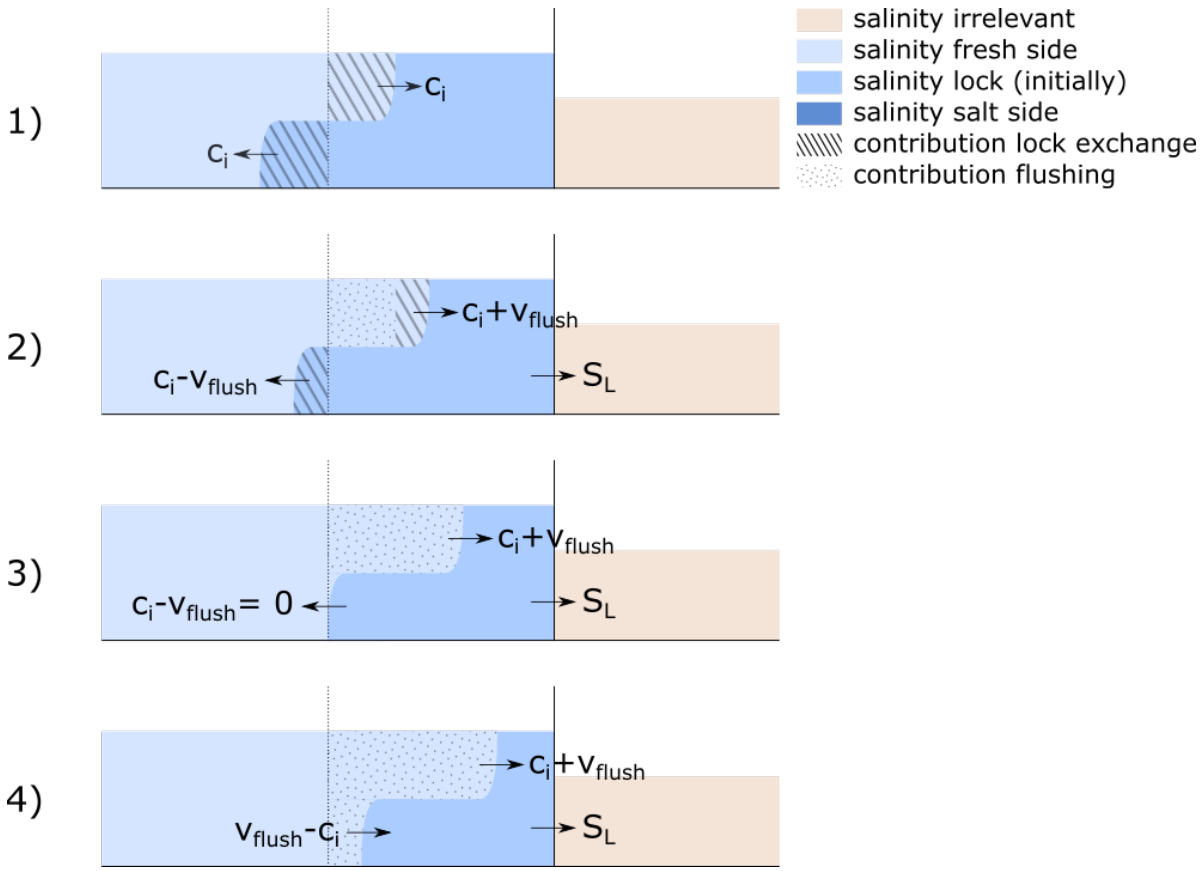


Fig. 3: Schematic overview of the entering density current as a function of the flushing discharge

In Scenario 1 the lock exchange is displayed for when there is no flushing discharge, as described in Section 3.2.5. The water that is exchange between the lock and the fresh approach harbor can be fully attributed to the density current.

In Scenario 2 is visible what happens when there is a flushing discharge with a velocity lower than that of the density current. The water from the lock chamber that enters the approach harbor can only get there because of the density current / lock exchange process. There is an equally large contribution of lock exchange in the lock chamber itself, but another part of the fresh water that has entered is due to the flushing discharge. In case of a flushing velocity of $0.5 \cdot c_i$ the ratio is 2/3 flushing discharge, and 1/3 lock exchange.

In Scenario 3 and 4 is visible what happens when the flushing velocity equals or exceeds the velocity of the density current. In these cases, the density current cannot exit the lock chamber, and as such we can say that no lock exchange has happened. The volume-averaged salinity of the lock chamber dropping can then be entirely attributed to the flushing discharge.

Reflecting current

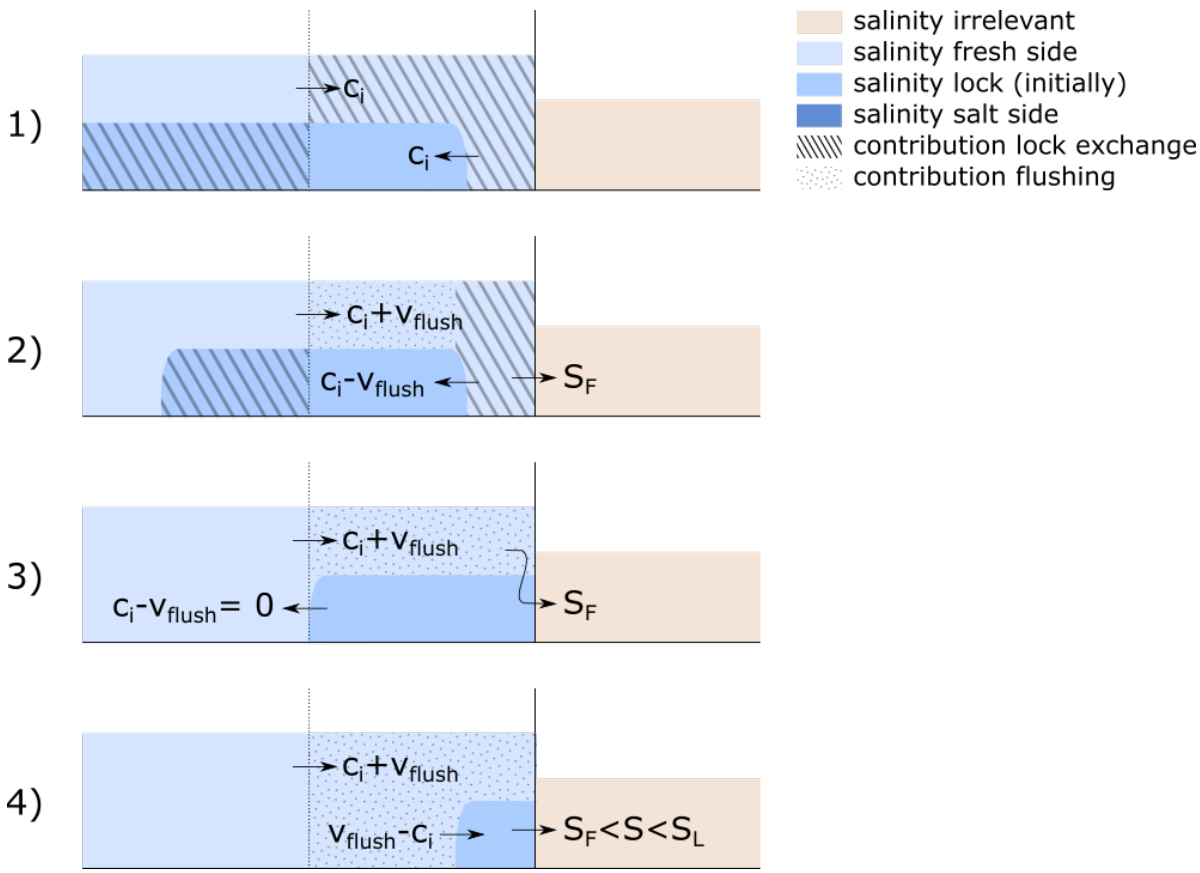


Fig. 4: Schematic overview of the reflecting density current as a function of the flushing discharge

When there is flushing discharge as in Scenario 1, the pace of the fresh water displacing salt water in the lock chamber after reflection is equal to that of an entering current. This holds when we assume velocities as shown in the figure above. In Section 3.2.5 a hyperbolic-tangent approach was used, which implies that the velocity of the density current slowly (but steadily) decreases.

In Scenario 2 is visible what happens when there would a flushing discharge. For an entering density current the water that goes to the salt side still has a salinity of S_L. That has now turned into water with salinity S_F This means that the pace of the lock chamber's salinity dropping in this phase scales with the velocity of the exiting density current, because all incoming flushing discharge with salinity S_F will pass right through the lock chamber and exit to the salt side. Contrary to Scenario 1, that means that the pace of the lock chamber's salinity dropping suddenly changes when the density current reflects. After reflection, the lock chamber becomes fresh slower than for an entering density current.

For Scenario 3 we extrapolate from Scenario 2, and therefore assume that the water leaving the lock chamber on the salt-side head has a salinity of that of the fresh side. For Scenario 4, a mixture of fresh water and water from the lock chamber exits to the salt side. As such, the salinity of the water going to the salt side is somewhere between these two.

Final (steady) state

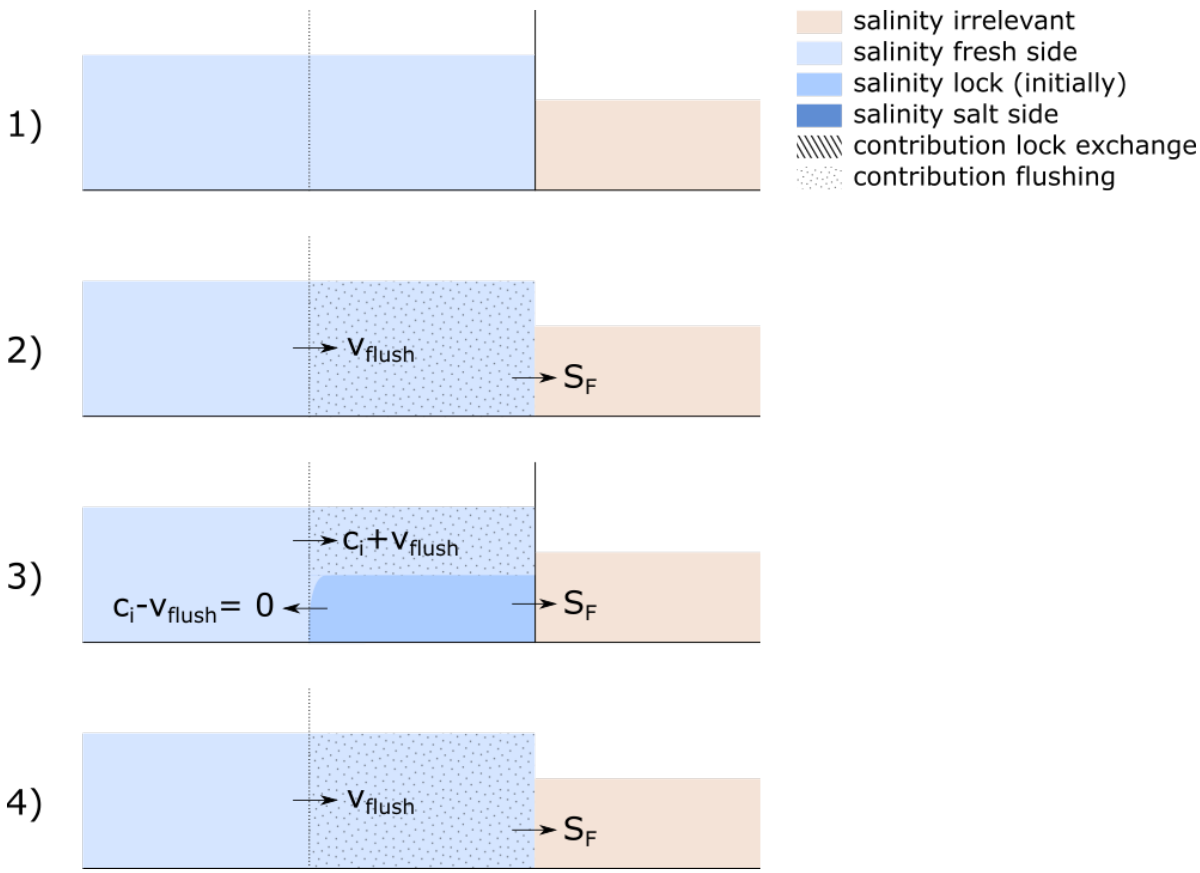


Fig. 5: Schematic overview of the final state of the lock chamber as a function of the flushing discharge

In the hypothetical case that the door is opened infinitely long, the final state in the lock chamber is often a fully fresh one. Only for Scenario 3 there would still be a salt bottom layer present. This is because, based on superposition of velocities alone, this layer never moves. In reality, mixing on the boundary layer between salt and fresh will slowly erode this bottom layer, such that also in this case a fully fresh lock chamber results.

Simplifications and formulation

In previous sections was discussed how the entering and reflecting density current behave if we assume a superposition of velocities. In Section 3.2.5 the lock exchange with flushing discharge is already written down in mathematical relations. Whereas a single formulation could cover all defined scenario's for the entering density current, that is no longer possible when also having to take the reflecting density current or final state into account. There would be an abrupt change in the pace with which the lock chamber's salinity drops, and also an abrupt change in the salinity of the flushing water going to the salt side. Furthermore, in the above discussion mixing/erosion has not been taken into account, even though those would occur in reality.

To come to a simple formulation, we make the following assumptions:

- The final state is always a fully fresh lock chamber (like in Scenario 1, 2 and 4). With this assumption we are closer to reality than would be the case based on superposition of velocities alone.

- The initial pace of salinity change in the lock (for an entering density current) scales with the ratio with the flushing velocity. This pace is kept even when the density current reflects, and slowly goes to zero with a hyperbolic tangent just like in Section 3.2.5.
- The pace of salinity change due to the flushing discharge is constant. In other words, the flushing water that enters the lock has a salinity of that of the fresh side, and the flushing water that exits has a salinity equal to that of the lock chamber when the doors opened. As soon as the lock is entirely fresh, flushing no longer has any effect on the salinity of the lock chamber, and as such we assume that the salinity of the water going to the salt side then switches to being equal to that of the fresh side (and equal to that of the lock chamber in that moment).

Written in equations, we see that the fraction of lock exchange in Scenario 2 scales linearly with the ratio of the flushing velocity to the velocity of the density current, until the situation of Scenario 3 is reached:

$$f_{LE,flush} = max\left(\frac{c_i - v_{flush}}{c_i}, 0\right) \tag{3.19}$$

This leads to the following equation for the exchanged volume due to lock exchange (with V_U as defined in Section 3.2.5):

$$V_{U,F} = f_{LE,flush} V_U \tag{3.20}$$

With these simplifications the lock becomes fresher quicker than in reality, but there is also less salt going to the fresh side. How conservative or optimistic this formulation then is (and with that the salt/fresh load), is unknown. This is also mentioned as a point of attention in Section TODO.

For Scenario 2, 3 and 4 it holds that flushing displaces the following volume:

$$V_{flush} = Q_{flush} T_{open,F} \tag{3.21}$$

The salinity of the flushing discharge is always S_F over the fresh-side head. The salinity of the flushing discharge over the salt-side head is initially S_L, switching to S_F as soon as the lock is fully fresh. In Scenario 3 and 4, where contrary to Scenario 1 and 2 there is no contribution of the lock exchange to the salinity decrease, this switch happens as soon as V_{flush} is equal to V_L.

Flushing salt side

When flushing with the door on the salt side opened, fresh water is entering the lock chamber from the fresh side. Two extremes of that situation are shown in the figure below. In reality, the distribution of the fresh water will be somewhere inbetween the extremes shown. In addition, the boundary between the fresh water and the saltier water in the lock chamber will not be as sharp due to mixing. As such, the water that goes to the the salt side will become fresh before the lock exchange process is done.

In the discussion of the entering and reflecting density current below we neglect the effect the dilution of the lock chamber has on the velocity of said density current.

Entering current

In Scenario 1 in the figure above the situation is shown for a density current with no flushing discharge. The water that is exchanged between the lock chamber and the salt approach harbor can be fully attributed to the density current, traversing with velocity c_i.

In Scenario 2 is visible what happens when one flushes with a velocity lower than that of the density current. The salty approach water that moves across the opened door into the lock chamber can only get there due to the lock exchange process / density current. An equal amount of water from the lock chamber exits to the approach harbor by way of the density current. The remainder of the lock chamber water that exits the lock is due to the flushing discharge. An equal

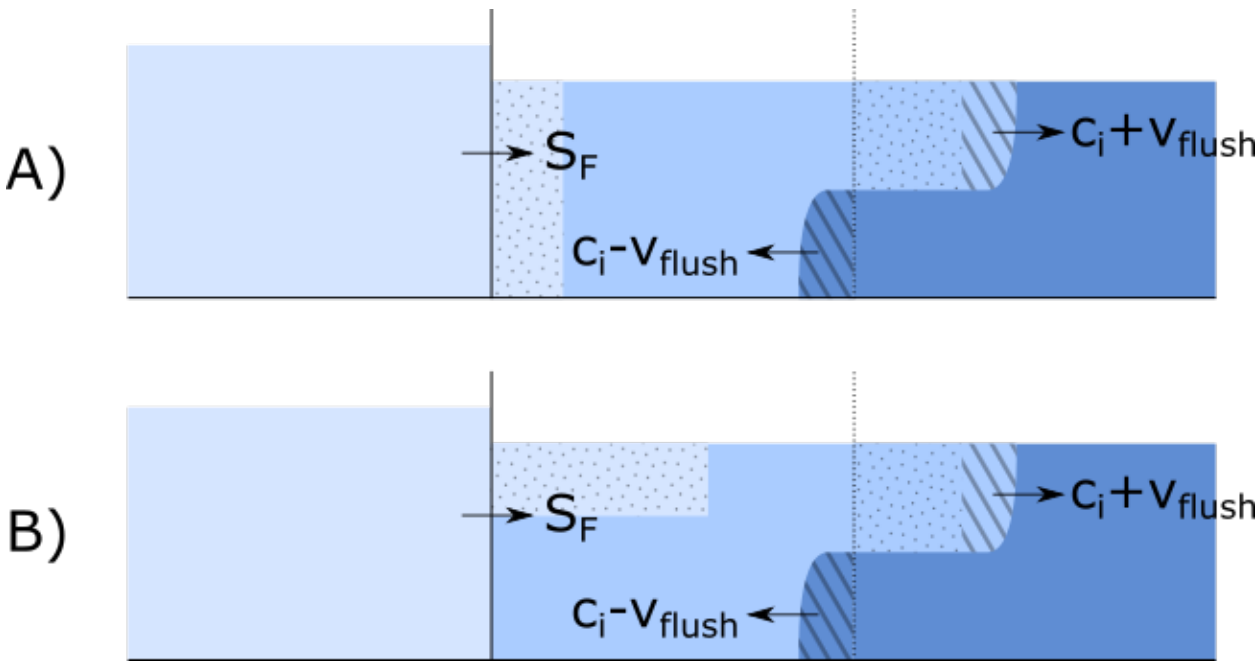


Fig. 6: Schematic overview of fresh flushing water entering the lock when salt-side door is open

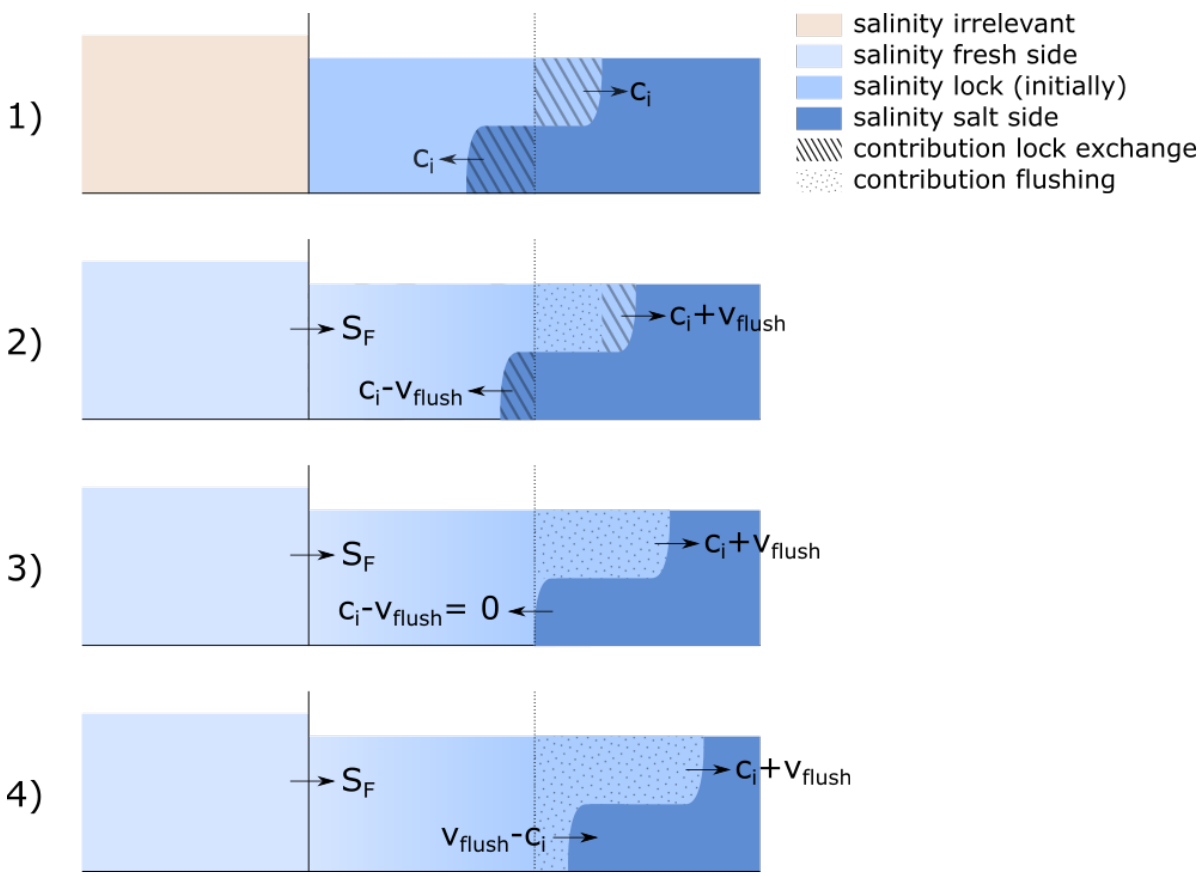


Fig. 7: Schematic overview of the entering density current as a function of the flushing discharge

amount of fresh water enters the lock from the lake side through the closed doors. That means the salinity in the lock is both increasing (near the sea side) and decreasing (near the fresh side).

In Scenario 3 and 4 is visible what happens when the velocity of the flushing discharge equals or surpasses that of the density current. In this case, the density current is not able to enter the lock, and as such we can say that there is no lock exchange taking place. There is however a dilution of the lock chamber, which can be fully attributed to the flushing discharge.

Reflecting current

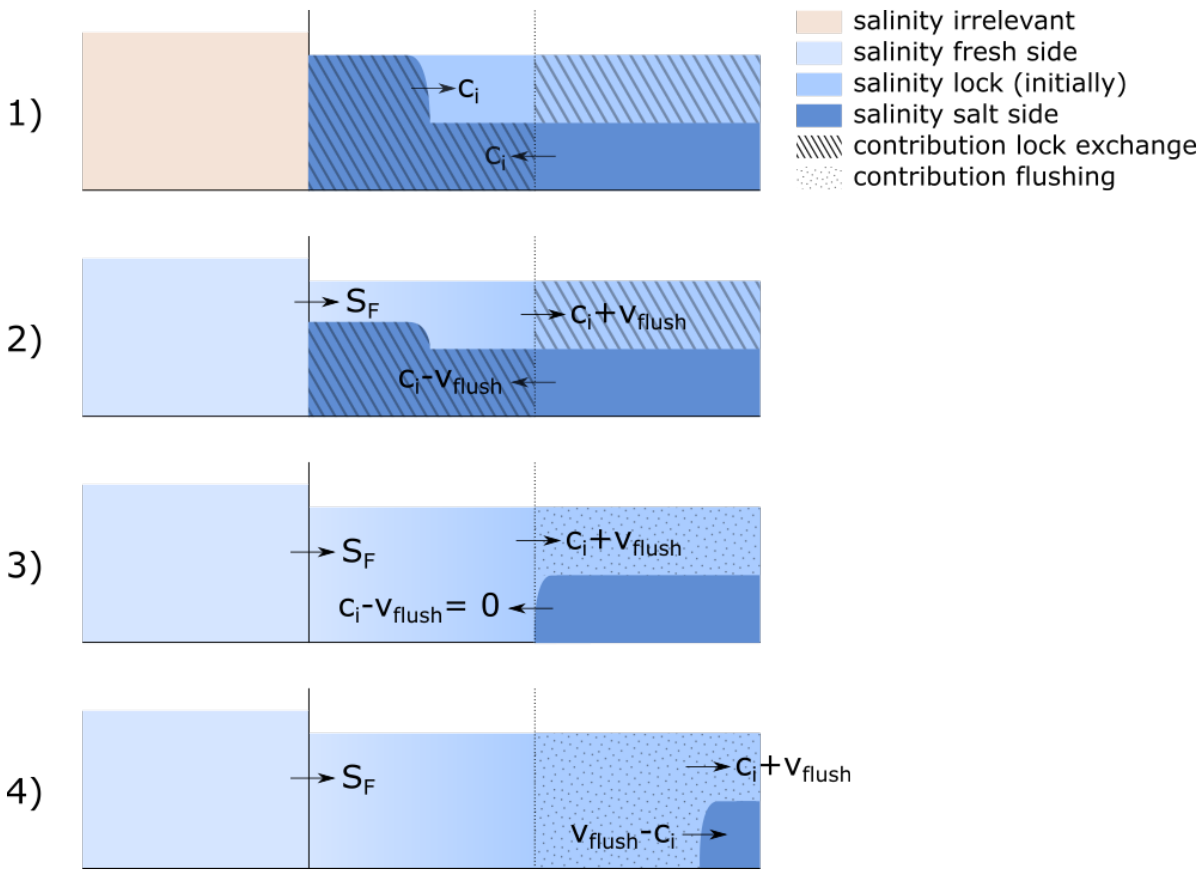


Fig. 8: Schematic overview of the reflecting density current as a function of the flushing discharge

When there is no flushing discharge, the pace of salinization of the lock chamber is equal to that of an entering current, if we assume the velocities as shown in the figure above. In Section 3.2.5 an approach with a hyperbolic tangent is described, which would mean that the pace of salinization would slowly but steadily drop.

In Scenario 2 is visible what happens when a flushing discharge is present. The salt wedge still enters with the same velocity, but exiting wedge (top layer) should exit more quickly because it cannot take up the full (half) height.

In Scenario 3 and 4 the density current does not enter the lock. In reality the density current will slowly start to enter the lock in Scenario 3, because the lock chamber is diluting and with that the driving density difference increases. The salinity of the water that leaves the lock to the salt approach harbor in this phase is likely somewhere between that of the initial salinity of the lock chamber, and that of the fresh side.

Final (steady) state

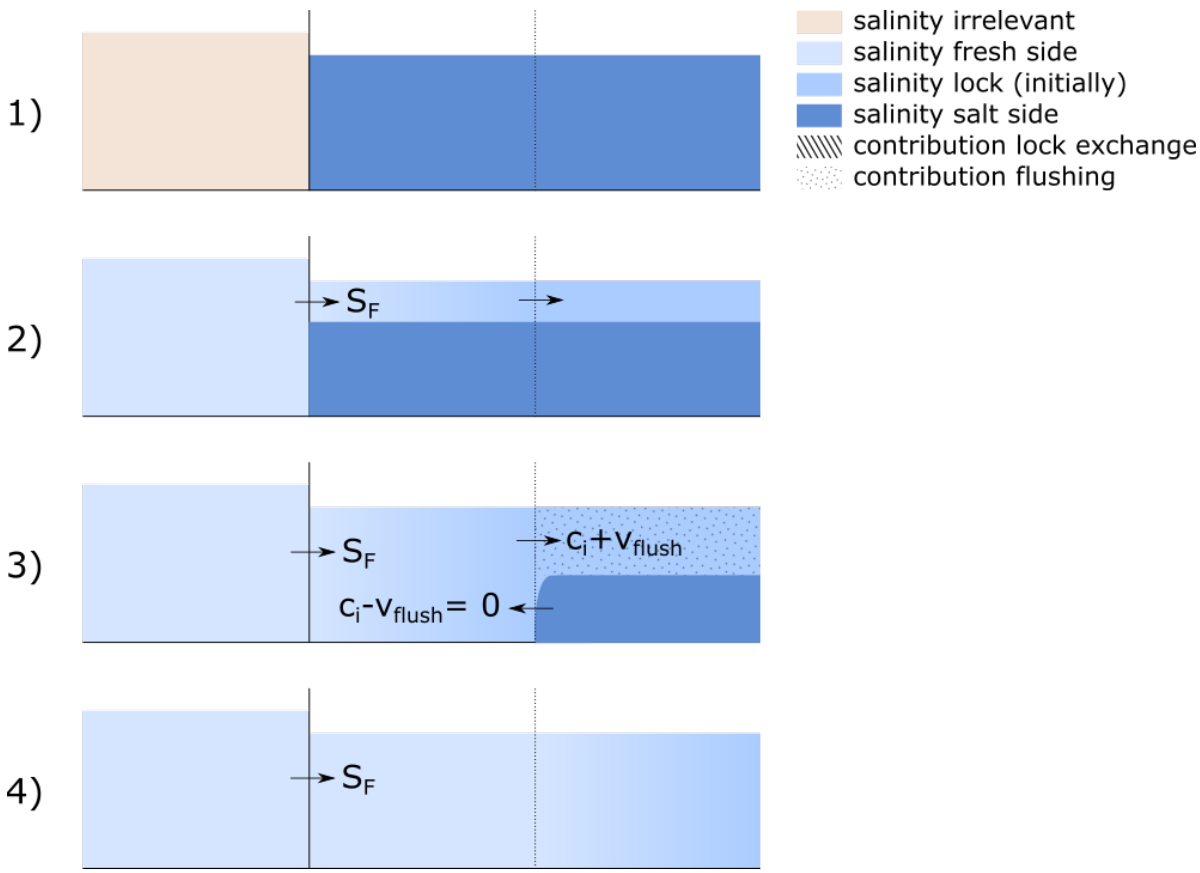


Fig. 9: Schematic overview of the final state of the lock chamber as a function of the flushing discharge

When the door on the sea side is open infinitely long, the equilibrium situation is dependent on the velocity of the flushing discharge.

In Scenario 1 the lock chamber will have acquired a salinity equal to that of the salt side.

In Scenario 2 a certain equilibrium between the driving salinity difference and flushing discharge is obtained. Shown in the figure above we assume a sharp interface between the salt bottom layer and the fresh layer of flushing discharge. In reality, instabilities on this interface will cause mixing, and this in turn will influence the lock exchange.

In Scenario 3 and 4 the lock chamber becomes fully fresh. A nuance is that the driving salinity difference does increase with the lock chamber diluting, and with that a density current might start to enter the lock chamber eventually anyway. That would then lead to an equilibrium state more equal to that of Scenario 2.

Simplifications and formulation

In previous sections was discussed how the entering and reflecting density current behave if we assume a superposition of velocities. In Section 3.2.5 the lock exchange with flushing discharge is already written down in mathematical relations. Contrary to the discussion of the fresh side, there is now a clear distinction between the final steady state in Scenario 1 and 2 (lock chamber fully or just partially salinized), and Scenario 3 and 4 (lock chamber diluted to salinity of fresh side). The formulation that is derived should therefore also make a distinction between these two.

Just like in the discussion for the fresh side, it is difficult to describe the transition from entering to reflecting density current. This is further complicated by the slow dilution of the lock chamber with fresh water, where it is not known how and where this water mixes with that of the lock chamber (i.e. does the fresh water float to the top, or does the entire lock chamber dilute more uniformly).

To come to a simple formulation, we make the following assumptions:

- The height of the top layer that forms in Scenario 2 is a function of the density difference between the salt side and the fresh side, and the flushing discharge. The higher the flushing discharge the thicker this layer of fresh water in the lock chamber. The higher the density difference, the thinner this layer is.
- In Scenario 1 and 2 we use the behavior as discussed in Section 3.2.5, where the pace of the lock exchange steadily decreases. The initial velocity of the density current is $c_i - v_{flush}$, and the maximum volume that can be exchanged is only part of the volume of the lock chamber (as discussed in the previous assumption). That is, the maximum exchangeable volume is $V_L - V_{top-layer}$.
- In Scenario 3 and 4 the lock chamber the salt density current will never enter the lock chamber. The pace of dilution due to the flushing discharge is assumed constant. In other words, the water that enters the lock has a salinity of that of the fresh side, and the water that exits has a salinity equal to the initial (at time of opening of the doors) salinity of the lock chamber. As soon as the lock chamber has been fully diluted, the salinity of the water that exits switches to that of the fresh side (equal to the lock chamber at that point in time).

For Scenario 1 we can then use the formulation of Section 3.2.5 as-is, or the below formulation for Scenario 2 with the height of the top layer equal to zero. For Scenario 2 it holds that we can solve for the thickness of the top layer by rewriting (3.13), and setting c_i equal to the velocity of the water in the top layer $v_{top-layer}$.

$$H_{eq} = 0.5 \frac{\left(2 \cdot v_{top-layer}\right)^2 \cdot \overline{\rho}_{FS}}{g \cdot 0.8 \left(S_S - S_F\right)} \tag{3.22}$$

$$v_{top-layer} = \frac{Q_{flush}}{H_{eq} \cdot W_L} \tag{3.23}$$

To make the process of rewriting easier to understand, we merge all fixed terms into one constant:

$$C = \frac{\overline{\rho}_{FS}}{g \cdot 0.8 \left(S_S - S_F\right)} \tag{3.24}$$

We can then write H_{eq} as a function of itself:

$$H_{eq} = 2 \frac{Q_{flush}^2}{H_{eq}^2 W_L^2} \cdot C \tag{3.25}$$

The last step is to make this height explicit:

$$H_{eq} = \left(2 \cdot \frac{Q_{flush}^2}{W_L^2} \cdot C\right)^{\frac{1}{3}} \tag{3.26}$$

With that, the maximum volume that can be exchanged by the density current then is:

$$f_{LE,flush} = \frac{H_S - H_{eq}}{H_S} \tag{3.27}$$

The time T_{LE} that it takes to exchange this volume is defined as follows:

$$T_{LE,S} = \frac{2 \cdot f_{LE,flush} \cdot L_L}{\eta c_i - v_{flush}} \tag{3.28}$$

This leads to the following equation for the exchanged volume by the density current:

$$V_{U,S} = f_{LE,flush} V_{L,S} \tanh\left(\frac{T_{open}}{T_{LE,S}}\right) \tag{3.29}$$

For Scenario 2 - 4 it holds that the flushing discharge displaces the following volume:

$$V_{flush} = Q_{flush} T_{open,S} \tag{3.30}$$

The salinity of this discharge is always S_F over the fresh-side head, i.e. when it enters the lock chamber. For the sea-side lock head, the salinity is initially S_L, switching to S_F as soon as the lock has been fully diluted. In Scenario 3 and 4, where contrary to Scenario 1 and 2 there is no salinization of the lock chamber, this switch in salinity happens as soon as V_{flush} is equal to V_L.

3.2.7 Overview of the input

Boundary conditions (only in stand-alone version):

h_F, h_S, S_F, S_S, T_F, T_S

Of these, especially h_S will be a time-dependent variable to study the influence of a tide.

Geometry (not taking into account sills and an uneven lock bottom):

Constant in time: L_L, W_L, z_L

Operation and shipping information:

N_cycles, T_{door}, T_{level}, c_{dot}, $C_{F/Avg}$, $V_{Ship,Up}$, $V_{Ship,Down}$

Initially all values are constant in time. However, to describe fluctuations in the operation of the lock in time (depending on how busy it is), and to describe variation in the direction of the overall shipping traffic, N_cycles, $C_{F/Avg}$, $V_{Ship,Up}$, $V_{Ship,Down}$ should be time-dependent parameters.

Measures:

Constant in time: η, $Q_{flush,LT}$, $Q_{flush,HT}$

3.3 Equations per locking phase

Below we discuss, per locking phase, the equations for the mass transport of salt, expressed in terms of volumes with certain salinity. Eventually the transports over the entire locking cycle can then be determined by aggregating the transports per phase.

Because we assume cyclic operation of the lock, the initial condition of the lock (e.g. salinity) are equal to the conditions at the end of the cycle.

3.3.1 Phase 1: Leveling to fresh side

The leveling takes place on the fresh-side head. There are no transports over the sea-side head.

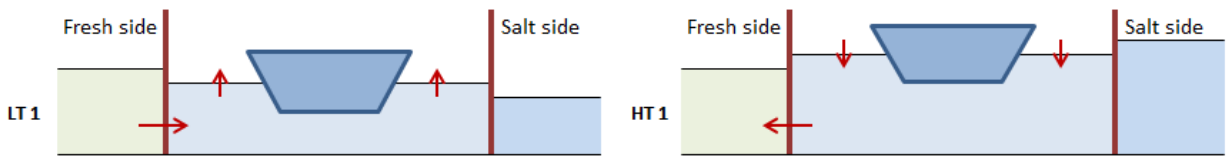


Fig. 10: Schematic overview of leveling to the fresh side during low and high tide

The salt transport due to leveling at low tide (LT) can be described as:

$$LT : M_{F,LT1} = V_{level,LT} S_F \tag{3.31}$$

Similarly for high tide:

$$HT : M_{F,HT1} = -V_{level,HT} S_{L,S} \tag{3.32}$$

By definition (see (3.10) and (3.11)) either $V_{level,LT}$ or $V_{level,HT}$ is zero, we can sum them up into one single equation:

$$M_{F,1} = V_{level,LT} S_F - V_{level,HT} S_{L,S} \tag{3.33}$$

When leveling at low tide, the average salinity of the water in lock chamber drops because fresh water is let in. To calculate this new salinity, we have to take into account the water displacement of ships present in the lock chamber:

$$LT : S_{L,S,Lev} = \frac{S_{L,S}\left(V_{L,S} - V_{Ship,Up}\right) + S_F V_{Lev,LT}}{V_{L,F} - V_{Ship,Up}} \tag{3.34}$$

In the situation around high tide, water is extracted from the lock chamber to lower the level, which does not change the salinity of the lock chamber:

$$HT : S_{L,F,Lev} = S_{L,S} \tag{3.35}$$

With this, we can write (3.33) as follows:

$$M_{F,1} = V_{level,LT} S_F - V_{level,HT} S_{L,S,Lev} \tag{3.36}$$

(The salinity in the lock chamber after Phase 1, $S_{L,S,Lev}$ can also be written as $S_{L,1}$.)

The phase-averaged discharges to and withdrawals from the fresh side are then as follows (note that there are no flows on the salt side):

- withdrawal from the fresh side, with the prevailing salinity S_F:

$$V^-_{F,1} = V_{Lev,LT} \tag{3.37}$$

$$Q^-_{F,1} = \frac{V^-_{F,1}}{T_{Lev}} \tag{3.38}$$

- discharge to the fresh side with salinity $S_{L,S}$:

$$V^+_{F,1} = V_{Lev,HT} \tag{3.39}$$

$$Q^+_{F,1} = \frac{V^+_{F,1}}{T_{Lev}} \tag{3.40}$$

3.3.2 Phase 2: Door open on fresh side

The figure below illustrates that in principle there are no differences between low tide and high tide. In case of flushing through the lock, there is also a transport over the sea-side head.

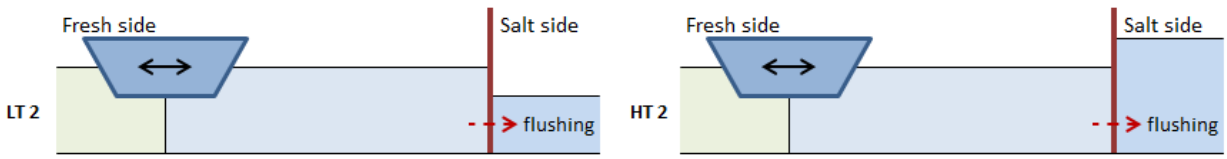


Fig. 11: Schematic overview of flows when door is open on the fresh side during low and high tide

While the doors are open on one side there are various processes that take place that contribute to the transport of salt over the opened lock head. These processes are:

1. Salt transport due to ships exiting the lock chamber
2. Salt transports due to the lock exchange (with or without flushing)
3. Salt transport due to ships entering the lock chamber

If the transports due to these processes are independently calculated before adding them up, there is a possibility of the salt transport being too high. This would result in a salinity in the lock chamber that is lower than the fresh side, or higher than the salt side. To prevent this from happening, Phase 2 has been divided into three subphases, corresponding to the list above. Each of these subphases leads to a new intermediate salinity of the lock chamber:

1: Salt transport due to ships exiting the lock

$$M_{F,2a} = V_{Ship,Up} S_F \tag{3.41}$$

$$S_{L,2a} = \frac{S_{L,S,Lev}\left(V_{L,F} - V_{Ship,Up}\right) + M_{F,2a}}{V_{L,F}} \tag{3.42}$$

2. Salt transport due to lock exchange (with or without flushing)

Contribution of lock exchange:

$$M_{F,2b,LE} = V_{U,F} S_F - V_{U,F} S_{L,2a} \tag{3.43}$$

With $V_{U,F}$ as determined in (3.20).

Contribution of flushing over fresh-side head:

$$V_{flush} = Q_{flush} T_{open,F} \tag{3.44}$$

$$M_{F,2b,flush} = V_{flush} S_F \tag{3.45}$$

Contribution of flushing over the salt-side head. When flushing for such a long time that the lock chamber's salinity reaches that of the fresh side, the salinity of the water going to the sea side changes accordingly:

$$V_{flush,max} = V_{L,F} - V_{U,F} \tag{3.46}$$

$$M_{S,2b,flush} = min\left(V_{flush}, V_{flush,max}\right) S_{L,2a} + max\left(V_{flush} - V_{flush,max}, 0\right) S_F \tag{3.47}$$

The new salinity at the end of this subphase then is:

$$S_{L,2b} = \frac{S_{L,2a} V_{L,F} + M_{F,2b,LE} + M_{F,2b,flush} - M_{Z,2b,flush}}{V_{L,F}} \tag{3.48}$$

3. Salt transport due to ships entering the lock

$$M_{F,2c} = -V_{Ship,Down} S_{L,2b} \tag{3.49}$$

3.3.3 Phase 2: Total transports

The total transport of salt over the fresh-side head in this Phase is the sum of the transports of each subphase:

$$M_{F,2} = M_{F,2a} + M_{F,2b,flush} + M_{F,2b,LE} + M_{F,2c} \tag{3.50}$$

In case of a non-zero flushing discharge, there is also a transport over the salt-side head:

$$M_{S,2} = M_{S,2b,flush} \tag{3.51}$$

The resulting salinity in the lock is then:

$$S_{L,F} = \frac{S_{L,S,Lev}\left(V_{L,F} - V_{Ship,Up}\right) + M_{M,2} - M_{S,2}}{\left(V_{L,F} - V_{Ship,Down}\right)} \tag{3.52}$$

(The salinity in the lock after Phase 2, $S_{L,F}$, can also be written as $S_{L,2}$.)

The phase-averaged discharges to and withdrawals from the fresh and salt side are then as follows:

- withdrawal from the fresh side, with the prevailing salinity S_F:

$$V_{F,2}^{-} = V_{Ship,Up} + V_{U,F} + V_{flush} \tag{3.53}$$

$$Q_{F,2}^{-} = \frac{V_{F,2}^{-}}{T_{open,F}} \tag{3.54}$$

- discharge to the fresh side with salinity $S_{F,2}^{+}$:

$$V_{F,2}^{+} = V_{Ship,Down} + V_{U,F} \tag{3.55}$$

$$Q_{F,2}^{+} = \frac{V_{F,2}^{+}}{T_{open,F}} \tag{3.56}$$

$$S_{F,2}^{+} = -\frac{M_{F,2} - V_{F,2}^{-} \cdot S_F}{V_{F,2}^{+}} \tag{3.57}$$

- there is no withdrawal from the salt side in this Phase
- discharge to the salt side with average salinity S_S^{+}

$$V_{S,2}^{+} = Q_{flush} T_{open,F} \tag{3.58}$$

$$Q_{S,2}^{+} = Q_{flush} \tag{3.59}$$

$$S_{S,2}^{+} = \frac{M_{S,2}}{V_{S,2}^{+}} \tag{3.60}$$

3.3.4 Phase 3: Leveling to salt side

The leveling takes place on the sea-side head. There are no transports over the fresh-side head.

Just like in Phase LT 1 and HT 1 it holds that by definition either $V_{level,LT}$ or $V_{level,HT}$ is zero. Therefore we can sum the equations for both tidal phases up into a single one:

$$M_{F,3} = V_{level,LT} S_{L,F} - V_{level,HT} S_S \tag{3.61}$$

In the situation around low tide, water is extracted from the lock chamber to lower the level, which does not change the salinity of the lock chamber:

$$LT : S_{L,F,Lev} = S_{L,F} \tag{3.62}$$

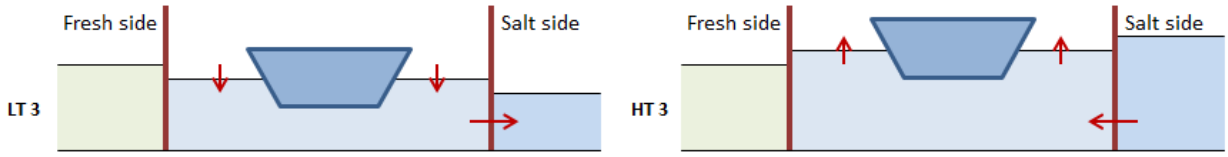


Fig. 12: Schematic overview of leveling to the salt side during low and high tide

When leveling at high tide, the average salinity of the water in lock chamber rises because salt water is let in. To calculate this new salinity, we have to take into account the water displacement of ships present in the lock chamber:

$$HT : S_{L,F,Lev} = \frac{S_{L,F}\left(V_{L,F} - V_{Ship,Down}\right) + S_S V_{Lev,HT}}{V_{L,S} - V_{Ship,Down}} \tag{3.63}$$

With this, we can write (3.61) as follows:

$$M_{S,3} = V_{level,LT} S_{L,F,Lev} - V_{level,HT} S_S \tag{3.64}$$

(The salinity in the lock chamber after Phase 3, $S_{L,F,Lev}$ can also be written as $S_{L,3}$.)

The phase-averaged discharges to and withdrawals from the salt side are then as follows (note that there are no flows on the fresh side):

- withdrawal from the salt side, with the prevailing salinity S_S:

$$V^-_{S,3} = V_{Lev,HT} \tag{3.65}$$

$$Q^-_{S,3} = \frac{V^-_{S,3}}{T_{Lev}} \tag{3.66}$$

- discharge to the salt side with salinity $S_{L,F}$:

$$V^+_{S,3} = V_{Lev,LT} \tag{3.67}$$

$$Q^+_{S,3} = \frac{V^+_{S,3}}{T_{Lev}} \tag{3.68}$$

3.3.5 Phase 4: Door open on salt side

The figure below illustrates that in principle there are no differences between low tide and high tide. In case of flushing through the lock, there is also a transport over the fresh-side head.

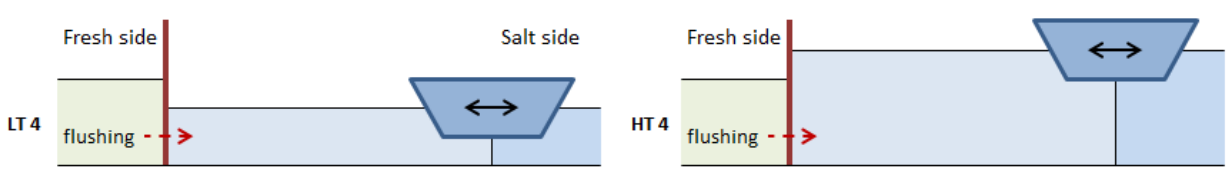


Fig. 13: Schematic overview of flows when door is open on the salt side during low and high tide

While the doors are open on one side there are various processes that take place that contribute to the transport of salt over the opened lock head. These processes are:

1. Salt transport due to ships exiting the lock chamber

2. Salt transports due to the lock exchange (with or without flushing)
3. Salt transport due to ships entering the lock chamber

If the transports due to these processes are independently calculated before adding them up, there is a possibility of the salt transport being too high. This would result in a salinity in the lock chamber that is lower than the fresh side, or higher than the salt side. To prevent this from happening, Phase 4 (just like Phase 2) has been divided into three subphases, corresponding to the list above. Each of these subphases leads to a new intermediate salinity of the lock chamber:

1: Salt transport due to ships exiting the lock

$$M_{S,4a} = -V_{Ship,Down} S_S \tag{3.69}$$

$$S_{L,4a} = \frac{S_{L,F,Lev} \left(V_{L,S} - V_{Ship,Down}\right) + M_{S,4a}}{V_{L,S}} \tag{3.70}$$

2. Salt transport due to lock exchange (with or without flushing)

Contribution of lock exchange:

$$M_{S,4b,LE} = V_{U,S} S_{L,4a} - V_{U,S} S_S \tag{3.71}$$

With $V_{U,S}$ as determined in (3.29).

Contribution of flushing over fresh-side head:

$$V_{flush} = Q_{flush} T_{open,S} \tag{3.72}$$

$$M_{F,4b,flush} = V_{flush} S_F \tag{3.73}$$

Contribution of flushing over the salt-side head. When flushing for a long, an equilibrium situation arises as described in Section 3.2.6. Furthermore, the salinity of the flushing discharge going to the salt side changes from that of the (initial salinity of the) lock chamber to that of the fresh side.

$$V_{flush,max} = V_{L,S} - V_{U,S} \tag{3.74}$$

$$M_{S,4b,flush} = min\left(V_{flush}, V_{flush,max}\right) S_{L,4a} + max\left(V_{flush} - V_{flush,max}, 0\right) S_F \tag{3.75}$$

The new salinity at the end of this subphase then is:

$$S_{L,4b} = \frac{S_{L,4a} V_{L,S} + M_{S,4b,LE} + M_{F,4b,flush} - M_{Z,4b,flush}}{V_{L,S}} \tag{3.76}$$

3. Salt transport due to ships entering the lock

$$M_{S,4c} = V_{Ship,Up} S_{L,4b} \tag{3.77}$$

3.3.6 Phase 4: Total transports

The total transport of salt over the salt-side head in this Phase is the sum of the transports of each subphase:

$$M_{S,4} = M_{S,4a} + M_{S,4b,flush} + M_{S,4b,LE} + M_{S,4c} \tag{3.78}$$

In case of a non-zero flushing discharge, there is also a transport over the fresh-side head:

$$M_{F,4} = M_{F,4b,flush} \tag{3.79}$$

The resulting salinity in the lock is then:

$$S_{L,S} = \frac{S_{L,F,Lev}\left(V_{L,S} - V_{Ship,Down}\right) + M_{M,4} - M_{S,4}}{\left(V_{L,S} - V_{Ship,Up}\right)} \tag{3.80}$$

(The salinity in the lock after Phase 4, $S_{L,S}$, can also be written as $S_{L,4}$.)

The phase-averaged discharges to and withdrawals from the fresh and salt side are then as follows:

- withdrawal from the salt side, with the prevailing salinity S_S:

$$V_{S,4}^{-} = V_{Ship,Down} + V_{U,S} \tag{3.81}$$

$$Q_{S,4}^{-} = \frac{V_{S,4}^{-}}{T_{open,S}} \tag{3.82}$$

- discharge to the salt side with salinity $S_{S,4}^{+}$:

$$V_{S,4}^{+} = V_{Ship,Up} + V_{U,S} + V_{flush} \tag{3.83}$$

$$Q_{S,4}^{+} = \frac{V_{S,4}^{+}}{T_{open,S}} \tag{3.84}$$

$$S_{S,4}^{+} = \frac{M_{S,4} - V_{S,4}^{-} \cdot S_S}{V_{S,4}^{+}} \tag{3.85}$$

- withdrawal from the fresh side, with the prevailing salinity S_F:

$$V_{F,4}^{-} = Q_{flush} T_{open,S} \tag{3.86}$$

$$Q_{F,4}^{-} = Q_{flush} \tag{3.87}$$

- there is no discharge to the lake side in this Phase

3.4 Cycle-averaged flows and salinities

Based on the volumes per locking cycle we now can, for each of the locking heads, determine the total transported volumes with their corresponding salinities. From these volumes the cycle-averaged flows can be determined.

3.4.1 Fresh side

The combined equation for the fresh side gives the total transport of the entire locking cycle. This equation is as follows:

$$M_F = M_{F,1} + M_{F,2} + M_{F,4} \tag{3.88}$$

Aside from that we have information about the amount of water that is withdrawn from the fresh side

$$V_F^{-} = V_{Level,LT} + V_{Ship,Up} + V_{U,F} + Q_{flush} \cdot 2 \cdot T_{open} \tag{3.89}$$

and the volume that is discharged to the fresh side

$$V_F^{+} = V_{Level,HT} + V_{U,F} + V_{Ship,Down} \tag{3.90}$$

By dividing both volumes by the time spent on a total locking cycle, we can determine the cycle-averaged flows. Each of these flows has a corresponding discharge, and can be connected to cells in a far-field model as a discharge or withdrawal:

- Withdrawal from the fresh side, with the prevailing salinity S_F:

$$Q_F^- = \frac{V_F^-}{T_{cycle}} \tag{3.91}$$

- Discharge to the fresh side with a to-be-determined average salinity:

$$Q_F^+ = \frac{V_F^+}{T_{cycle}}; S = S_F^+ \tag{3.92}$$

The average salinity for the water discharged to the fresh side is determined from the mass and volume transports:

$$S_F^+ = -\frac{\left(M_F - V_F^- S_F\right)}{V_F^+} \tag{3.93}$$

In case of stand-alone application, but also to compare with other locks or salt intrusion measure configurations, it can be useful to express the salt transport as a mass flux:

$$\dot{M}_F = \frac{M_F}{T_{cycle}} \tag{3.94}$$

3.4.2 Salt side

The combined equation for the salt side is:

$$M_S = M_{S,2} + M_{S,3} + M_{S,4} \tag{3.95}$$

Again, we can write down the volumes going to and from the salt side. For the withdrawal that is:

$$V_S^- = V_{Level,HT} + V_{Ship,Down} + V_{U,S,Flush} \tag{3.96}$$

and the volume that is discharged to the salt side

$$V_S^+ = V_{Level,LT} + V_{U,S,Flush} + V_{Ship,Up} + Q_{flush} \cdot 2 \cdot T_{open} \tag{3.97}$$

By dividing both volumes by the time spent on a total locking cycle, we can determine the cycle-averaged flows. Each of these flows has a corresponding discharge, and can be connected to cells in a far-field model as a discharge or withdrawal:

- Withdrawal from the salt side, with the prevailing salinity S_S:

$$Q_S^- = \frac{V_S^-}{T_{cycle}} \tag{3.98}$$

- Discharge to the salt side with a to-be-determined average salinity:

$$Q_S^+ = \frac{V_S^+}{T_{cycle}}; S = S_S^+ \tag{3.99}$$

The average salinity for the water discharged to the salt side is determined from the mass and volume transports:

$$S_S^+ = \frac{\left(M_S - V_S^- S_S\right)}{V_S^+} \tag{3.100}$$

In case of stand-alone application, but also to compare with other locks or salt intrusion measure configurations, it can be useful to express the salt transport as a mass flux:

$$\dot{M}_S = \frac{M_S}{T_{cycle}} \tag{3.101}$$

For an equilibrium state, with the lock operating with constant operation for long periods of time, it obviously holds that

$$\dot{M}_S = \dot{M}_F \tag{3.102}$$

3.5 Numerical approach cycle-averaged values

The description in Chapter *Equations per locking phase* of how the transports and how the salinity in the lock chamber changes from phase to phase, also gives a recipe for calculating the cycle-averaged values of volumes and salinities. Namely, we can iteratively calculate our way through the locking cycle until the calculated value no longer change significantly. From these steady state values the cycle-averaged flows (discharges and withdrawals) can be determined, together with the salinities (see Section 3.5.1).

An alternative approach is setting up a system of equations (see Section 3.5.2). From Chapter *Equations per locking phase* and *Cycle-averaged flows and salinities* the essential equations can be selected, that together form a system of equations with unknowns. Solving this system of equations than resolves the unknowns that are necessary to calculate the cycle-averaged flows and salinities.

3.5.1 Iteratively calculating the locking cycle

In Chapter *Cycle-averaged flows and salinities* the expression for the flows (discharges and withdrawals) have been set up, but a few unknowns remain. These values arise in the locking cycle, as described in Chapter *Equations per locking phase*.

This cycle can then be considered iterative process: for fixed boundary conditions and after enough cycles, eventually these unknowns will converge to their respective values. To enter this iterative process, e.g. starting at LT 1 / HT 1, we only need an initial guess for the salinity of the lock chamber. Obviously this guess has to be higher than the salinity of the fresh side, and lower than that of the salt side. A good starting point would then be the average of the salinities of the boundary conditions.

In case of calculating through time for varying boundary conditions (e.g. a tide on the sea side, or a time-varying operation of the lock), the converged lock chamber salinity of the previous time step can be chosen as the initial guess. For slowly changing boundary conditions, the previous chamber salinity is a reasonable estimate. The closer the guess to the eventual solution, the fewer iterations are needed to converge.

The numerical approach for determining cycle-averaged values then consists of two steps:

1. Iteratively determining the unknown values
2. Calculate the discharges Q_F^+ with S_F^+ and Q_S^+ with S_S^+, and the withdrawals Q_F^- and Q_S^-.

3.5.2 System of equations

An alternative to the iterative approach as described above is setting up a system of equations in which the number of equations is equal to the number of unknowns. From the previous chapters the relevant equations can be selected, each with a number of unknowns. (The unknowns in this case are all values that cannot be directly calculated from the boundary conditions or input). Not all equations are linear, so it is necessary to repeatedly solve a linearized system of equations until convergence. In practice, this way of solving has very little advantage over calculating iteratively as described in *Iteratively calculating the locking cycle*. We therefore do not elaborate further on this approach.

3.5.3 Overview of output

By the formulation the steady state values of the following quantities are calculated, all as function of time:

M_F, $\dot{M}_F$, Q_F^-, Q_F^+, S_F^+, M_S, $\dot{M}_S$, Q_S^-, Q_S^+, S_S^+

Getting to these parameters is what the formulation was designed for: the mass transports (per cycle) and the fluxes and flows (cycle-averaged) that enter and exit the lock on both sides. Aside from that there are a few other parameters that can help interpret the output. These can be geometric parameters, like the volume of the lock chamber, or operational parameters like door-open times. It is also possible to export the cycle-averaged transports per locking phase.

Aside from that, a few other useful parameters can be determined. These parameters can help compare the ZSF to more naive methods of determined the salt load, or with other theoretical or experimental relations.

Dimensionless salt transport

For the dimensionless salt transport over the lock per cycle, we use the parameter $Z_{fraction}$. This parameter is defined as a factor on the lock chamber volume times the difference in salinity between the boundary conditions:

$$M = Z_{fraction} V_L \cdot \Delta S \tag{3.103}$$

As such, $Z_{fraction}$ indicates what fraction of the lock chamber, in regular locking operation, exchanges and contributes to the salt transport.

In the process of calculating transports we get transports for both lock heads. Aside from that, the volume of the lock chamber, due to a difference in water level on both sides, is not always equal. Therefore, we have to use average values for these quantities. From this, $Z_{fraction}$ can be written as:

$$Z_{fraction} = \frac{\overline{M}}{\overline{V_L} \cdot (S_S - S_F)} = \frac{0.5 \cdot (M_F + M_S)}{0.5 \cdot (V_{L,F} + V_{L,S}) \cdot (S_S - S_F)} \tag{3.104}$$

Dimensionless door-open time

The dimensionless door-open time is T_{LE}/T_{open}. In the calculations the value of T_{LE} is determined per lock head, with the prevailing salinity difference. With that, the value of T_{LE} is not a direct function of the input, but is dependent on the calculations. To get to a T_{LE} that is only determined by input (boundary conditions and geometry), we define a variant: $T_{LE,FS}$. This quantity is based solely on the salinity difference over the lock. This parameter is not used in the calculations, but is sometimes used in determining auxiliary outputs, e.g. of the dimensionless door-open time T_{LE}/T_{open}:

$$c_{i,FS} = \frac{1}{2}\sqrt{g'\overline{H}} = \frac{1}{2}\sqrt{g\frac{\Delta\rho}{\overline{\rho}}\overline{H}} \approx \frac{1}{2}\sqrt{\frac{0.8\,(S_S - S_F)}{\overline{\rho}_{FS}}\left(\frac{H_F + H_S}{2}\right)} \tag{3.105}$$

$$T_{LE,FS} = \frac{2L}{c_{i,FS}} \tag{3.106}$$

CHAPTER

FOUR

API

4.1 C API

4.1.1 Structures

Input

struct **zsf_param_t**

The parameter structure both for phase-wise calculation, and for steady state calculation.

double **lock_length**

The length of the lock in meters.

double **lock_width**

The width of the lock in meters.

double **lock_bottom**

The bottom of the lock in meters with respect to datum (e.g. mNAP).

double **num_cycles**

(Steady) The number of cycles (leveling from lake to sea and back again) the lock makes in a day.

double **door_time_to_open**

(Steady) The time it takes for the door to go from fully closed to fully (open in seconds.

double **leveling_time**

(Steady) The average time it takes to level the lock to the sea or lake level in seconds.

double **calibration_coefficient**

(Steady) The calibration coefficient on the time that the door is open.

double **symmetry_coefficient**

(Steady) The imbalance between the lake door and right door being open, should be in the range (0, 2). A value of 1.0 means that the lake and sea side door are open equally long.

double **ship_volume_sea_to_lake**

The water displacement of ships going from the sea to the lake in m^3.

double **ship_volume_lake_to_sea**

The water displacement of ships going from the sea to the lake in m^3.

double **salinity_lock**

The (initial) salinity of the lock in kg/m^3.

double **head_sea**

The head of the sea in meters with respect to datum (e.g. mNAP).

double **salinity_sea**

The salinity of the sea in kg/m^3.

double **temperature_sea**

The temperature of the sea in degrees Celcius.

double **head_lake**

The head of the lake in meters with respect to datum (e.g. mNAP).

double **salinity_lake**

The salinity of the lake in kg/m^3.

double **temperature_lake**

The temperature of the lake in degrees Celcius.

double **flushing_discharge_high_tide**

The flushing discharge in $m3/s$ when the sea level is higher than (or equal to) lake level.

double **flushing_discharge_low_tide**

The flushing discharge in $m3/s$ when the lake level is strictly below the sea level.

double **density_current_factor_sea**

The factor by which to multiply the velocity of the density current on the sea side.

double **density_current_factor_lake**

The factor by which to multiply the velocity of the density current on the lake side.

double **distance_door_bubble_screen_sea**

Distance of the bubble screen on the lake side to the door in meters. Positive values mean that the screen is outside the lock. Negative values mean that the screen is inside the lock.

double **distance_door_bubble_screen_lake**

Distance of the bubble screen on the lake side to the door in meters. Positive values mean that the screen is outside the lock. Negative values mean that the screen is inside the lock.

double **sill_height_sea**

Sill height on the sea side in meters above the bottom of the lock.

double **sill_height_lake**

Sill height on the lake side in meters above the bottom of the lock.

double **rtol**

(Steady) The relative tolerance of the salinity in the lock after phase 4 to determine whether convergence has been reached.

double **atol**

(Steady) The absolute tolerance of the salinity in the lock after phase 4 to determine whether convergence has been reached.

Steady state output

These structures are output by `zsf_calc_steady()`. Note that `zsf_aux_results_t` is an optional output of this function, and is not output/calculated by default.

struct **zsf_results_t**

For mass and salt transport the definition is such that positive values are in the direction lake → lock → sea. Negative values mean that there is a net withdrawal of salt from the sea or net salt load on the lake.

double **mass_transport_lake**

The mass transport of salt over the lake head in kg.

double **salt_load_lake**

The average salt transport to the lake kg/s.

double **discharge_from_lake**

The average discharge from the lake to the lock in $m3/s$.

double **discharge_to_lake**

The average discharge from the lock to the lake in $m3/s$.

double **salinity_to_lake**

The average salinity of the water going from the lock to the lake in kg/m^3.

double **mass_transport_sea**

The mass transport of salt over the sea head in kg/m^3.

double **salt_load_sea**

The average salt transport to the sea kg/s.

double **discharge_from_sea**

The average discharge from the sea to the lock in $m3/s$.

double **discharge_to_sea**

The average discharge from the lock to the sea in $m3/s$.

double **salinity_to_sea**

The average salinity of the water going from the lock to the sea in kg/m^3.

struct **zsf_aux_results_t**

Additional results that can be calculated for steady state operation.

double **z_fraction**

The dimensionless salt transport per cycle. It is defined as a factor on the lock volume times the difference in salinity between the lake and sea side.

$$M = Z_{fraction} V_{lock} \cdot \Delta S$$

This way it represents what part of the lock, in the regular locking process, is exchanged by the density current and contributes to the salt transport.

Because the salt transports are per head, and the volume of the lock is not always equal on sea and lake side, average values are used for these.

$$Z_{fraction} = \frac{\overline{M}}{\overline{V}_{lock} \cdot (S_{sea} - S_{lake})} = \frac{0.5 \cdot (M_{lake} + M_{sea})}{0.5 \cdot (V_{lock,lake} + V_{lock,sea}) \cdot (S_{sea} - S_{lake})}$$

double **dimensionless_door_open_time**

The dimensionless door open time is T_{LE}/T_{open}. In this calculation the value for T_{LE} is calculated per lock head, with the corresponding salinity difference. That means that T_{LE} is not just a function of the input, but is dependent on the calculation routines, and there are two values (one for each lock head). To get to a single T_{LE} that is determined by the geometry and boundary conditions, we define a variant of $T_{LE,lake-sea}$ based on the density difference over the lock.

$$c_{i,lake-sea} = \frac{1}{2}\sqrt{g'\overline{H}} = \frac{1}{2}\sqrt{g\frac{\Delta\rho}{\bar{\rho}}\overline{H}} \approx \frac{1}{2}\sqrt{g\frac{0.8\left(S_{sea}-S_{lake}\right)}{\bar{\rho}_{lake-sea}}\left(\frac{H_{lake}+H_{sea}}{2}\right)}$$

$$T_{LE,lake-sea} = \frac{2L}{c_{i,lake-sea}}$$

double **volume_to_lake**

The volume that is discharged to the lake per locking cycle in m^3.

double **volume_from_lake**

The volume that is withdrawn from the lake per locking cycle in m^3.

double **volume_to_sea**

The volume that is discharged to the sea per locking cycle in m^3.

double **volume_from_sea**

The volume that is withdrawn from the sea per locking cycle in m^3.

double **volume_lock_at_lake**

The volume of the lock when it is at sea level in m^3.

double **volume_lock_at_sea**

The volume of the lock when it is at sea level in m^3.

double **t_cycle**

The time it takes to complete one locking cycle in seconds.

double **t_open**

The average value of the door open time on the lake and sea side, i.e. the average of *zsf_aux_results_t.t_open_lake* and *zsf_aux_results_t.t_open_sea*.

double **t_open_lake**

The time the door is open on the lake side per locking cycle in seconds.

double **t_open_sea**

The time the door is open on the sea side per locking cycle in seconds.

double **salinity_lock_1**

The salinity of the lock after phase 1 in kg/m^3.

double **salinity_lock_2**

The salinity of the lock after phase 2 in kg/m^3.

double **salinity_lock_3**

The salinity of the lock after phase 3 in kg/m^3.

double **salinity_lock_4**

The salinity of the lock after phase 4 in kg/m^3.

zsf_phase_transports_t **transports_phase_1**

The phase transports in phase 1. See `zsf_phase_transports_t`

zsf_phase_transports_t **transports_phase_2**

The phase transports in phase 2. See `zsf_phase_transports_t`

zsf_phase_transports_t **transports_phase_3**

The phase transports in phase 3. See `zsf_phase_transports_t`

zsf_phase_transports_t **transports_phase_4**

The phase transports in phase 4. See `zsf_phase_transports_t`

Phase-wise output

struct **zsf_phase_state_t**

The state of the lock. Note that some of these values are redundant, but it is faster to store them than recalculate them every time.

double **salinity_lock**

The salinity of the water in the lock in kg/m^3.

double **saltmass_lock**

The amount of salt in the lock in kg.

double **head_lock**

The head of the lock in meters with respect to datum (e.g. mNAP).

double **volume_ship_in_lock**

The water displacement of a ship inside the lock in m^3.

struct **zsf_phase_transports_t**

For mass and salt transport the definition is such that positive values are in the direction lake $\rightarrow$ lock $\rightarrow$ sea. Negative values mean that there is a net withdrawal of salt from the sea or net salt load on the lake.

double **mass_transport_lake**

The mass transport of salt over the lake head in kg.

double **volume_from_lake**

The volume of water that goes from the lake to the lock in m^3.

double **volume_to_lake**

The volume of water that goes from the the lock to the lake in m^3.

double **discharge_from_lake**

The average discharge of water going from the lake to the lock in $m3/s$.

double **discharge_to_lake**

The average discharge of water going from the lock to the lake in $m3/s$.

double **salinity_to_lake**

The average salinity of the water going from the lock to the lake in kg/m^3.

double **mass_transport_sea**

The mass transport of salt over the sea head in kg.

double **volume_from_sea**

The volume of water that goes from the sea to the lock in m^3.

double **volume_to_sea**

The volume of water that goes from the the lock to the sea in m^3.

double **discharge_from_sea**

The average discharge of water going from the sea to the lock in $m3/s$.

double **discharge_to_sea**

The average discharge of water going from the lock to the sea in $m3/s$.

double **salinity_to_sea**

The average salinity of the water going from the lock to the sea in kg/m^3.

4.1.2 Functions

int **zsf_initialize_state**(const *zsf_param_t* *p, *zsf_phase_state_t* *state, double salinity_lock, double head_lock)

Fill the state with an initial condition for an empty (no ships) lock.

int **zsf_step_phase_1**(const *zsf_param_t* *p, double t_level, *zsf_phase_state_t* *state, *zsf_phase_transports_t* *results)

Perform step 1: leveling to lake side

int **zsf_step_phase_2**(const *zsf_param_t* *p, double t_open_lake, *zsf_phase_state_t* *state, *zsf_phase_transports_t* *results)

Perform step 2: door open to lake side:

- Ships leaving the lock chamber
- Lock exchange with or without flushing
- Ships entering the lock chamber

int **zsf_step_phase_3**(const *zsf_param_t* *p, double t_level, *zsf_phase_state_t* *state, *zsf_phase_transports_t* *results)

Perform step 3: leveling to sea side

int **zsf_step_phase_4**(const *zsf_param_t* *p, double t_open_sea, *zsf_phase_state_t* *state, *zsf_phase_transports_t* *results)

Perform step 4: door open to sea side:

- Ships leaving the lock chamber
- Lock exchange with or without flushing
- Ships entering the lock chamber

int **zsf_step_flush_doors_closed**(const *zsf_param_t* *p, double t_flushing, *zsf_phase_state_t* *state, *zsf_phase_transports_t* *results)

Flush the lock with the doors closed.

void **zsf_param_default**(*zsf_param_t* *p)

Fill a `zsf_param_t` with default values.

int **zsf_calc_steady**(const *zsf_param_t* *p, *zsf_results_t* *results, *zsf_aux_results_t* *aux_results)

Calculate the salt intrusion for a set of parameters, assuming steady operation.

const char *`zsf_error_msg`(int code)

Get error message corresponding to error code.

const char *`zsf_version`()

Get version string.

4.2 Python API

class pyzsf.**ZSFUnsteady**(*sal_lock*, *head_lock*, ***parameters: float*)

Bases: `object`

A class to calculate a lock in phase-wise fashion.

property state: Dict[str, float]

Get the state of the lock, see also *zsf_phase_state_t*.

Note that this is a read-only property, and any changes made to the dictionary returned by this property do not persist.

step_flush_doors_closed(*t_flushing: float*, ***parameters: float*) → Dict[str, float]

Open the door on sea side. See also *zsf_step_flush_doors_closed()* .

Parameters

- **t_flushing** – Duration of flushing is open in seconds.
- **parameters** – Any parameters that should be changed before performing this step. Note that these changes persist.

Returns

The salt and water transports in this phase. See also *zsf_phase_transports_t*.

step_phase_1(*t_level*, ***parameters: float*) → Dict[str, float]

Level the lock to lake side. See also *zsf_step_phase_1()* .

Parameters

- **t_level** – Duration of the leveling in seconds.
- **parameters** – Any parameters that should be changed before performing this step. Note that these changes persist.

Returns

The salt and water transports in this phase. See also *zsf_phase_transports_t*.

step_phase_2(*t_open_lake: float*, ***parameters: float*) → Dict[str, float]

Open the door on lake side. See also *zsf_step_phase_2()* .

Parameters

- **t_open_lake** – Duration the door is open in seconds.
- **parameters** – Any parameters that should be changed before performing this step. Note that these changes persist.

Returns

The salt and water transports in this phase. See also *zsf_phase_transports_t*.

step_phase_3(*t_level*, ***parameters: float*) → Dict[str, float]

Level the lock to sea side. See also *zsf_step_phase_3()* .

Parameters

- **t_level** – Duration of the leveling in seconds.
- **parameters** – Any parameters that should be changed before performing this step. Note that these changes persist.

Returns

The salt and water transports in this phase. See also *zsf_phase_transports_t*.

step_phase_4(*t_open_sea: float*, ***parameters: float*) → Dict[str, float]

Open the door on sea side. See also *zsf_step_phase_4()* .

Parameters

- **t_open_sea** – Duration the door is open in seconds.
- **parameters** – Any parameters that should be changed before performing this step. Note that these changes persist.

Returns

The salt and water transports in this phase. See also *zsf_phase_transports_t*.

pyzsf.**zsf_calc_steady**(*auxiliary_results: bool = False*, ***parameters: float*) → Dict[str, float]

Calculate the salt intrusion for a set of parameters, assuming steady operation.

Parameters

- **auxiliary_results** – Whether or not to calculate and output auxiliary results. See *zsf_aux_results_t*.
- **kwargs** – Any parameters that should be changed versus the default. See also *zsf_param_t* for an overview of the parameters.

Returns

A dictionary containing the cycle averaged salt fluxes and discharges (see *zsf_results_t*). Also outputs values in *zsf_aux_results_t* if `auxiliary_results` is *True*.

CHAPTER FIVE

SUPPORT

Raise any issue on GitLab such that we can address your problem.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

S

state (*pyzsf.ZSFUnsteady property*), 67
step_flush_doors_closed() (*pyzsf.ZSFUnsteady method*), 67
step_phase_1() (*pyzsf.ZSFUnsteady method*), 67
step_phase_2() (*pyzsf.ZSFUnsteady method*), 67
step_phase_3() (*pyzsf.ZSFUnsteady method*), 67
step_phase_4() (*pyzsf.ZSFUnsteady method*), 68

Z

zsf_aux_results_t (*C struct*), 63
zsf_aux_results_t.dimensionless_door_open_time (*C var*), 63
zsf_aux_results_t.salinity_lock_1 (*C var*), 64
zsf_aux_results_t.salinity_lock_2 (*C var*), 64
zsf_aux_results_t.salinity_lock_3 (*C var*), 64
zsf_aux_results_t.salinity_lock_4 (*C var*), 64
zsf_aux_results_t.t_cycle (*C var*), 64
zsf_aux_results_t.t_open (*C var*), 64
zsf_aux_results_t.t_open_lake (*C var*), 64
zsf_aux_results_t.t_open_sea (*C var*), 64
zsf_aux_results_t.transports_phase_1 (*C var*), 64
zsf_aux_results_t.transports_phase_2 (*C var*), 65
zsf_aux_results_t.transports_phase_3 (*C var*), 65
zsf_aux_results_t.transports_phase_4 (*C var*), 65
zsf_aux_results_t.volume_from_lake (*C var*), 64
zsf_aux_results_t.volume_from_sea (*C var*), 64
zsf_aux_results_t.volume_lock_at_lake (*C var*), 64
zsf_aux_results_t.volume_lock_at_sea (*C var*), 64
zsf_aux_results_t.volume_to_lake (*C var*), 64
zsf_aux_results_t.volume_to_sea (*C var*), 64
zsf_aux_results_t.z_fraction (*C var*), 63
zsf_calc_steady (*C function*), 66
zsf_calc_steady() (*in module pyzsf*), 68
zsf_error_msg (*C function*), 66
zsf_initialize_state (*C function*), 66
zsf_param_default (*C function*), 66
zsf_param_t (*C struct*), 61
zsf_param_t.atol (*C var*), 62
zsf_param_t.calibration_coefficient (*C var*), 61
zsf_param_t.density_current_factor_lake (*C var*), 62
zsf_param_t.density_current_factor_sea (*C var*), 62
zsf_param_t.distance_door_bubble_screen_lake (*C var*), 62
zsf_param_t.distance_door_bubble_screen_sea (*C var*), 62
zsf_param_t.door_time_to_open (*C var*), 61
zsf_param_t.flushing_discharge_high_tide (*C var*), 62
zsf_param_t.flushing_discharge_low_tide (*C var*), 62
zsf_param_t.head_lake (*C var*), 62
zsf_param_t.head_sea (*C var*), 61
zsf_param_t.leveling_time (*C var*), 61
zsf_param_t.lock_bottom (*C var*), 61
zsf_param_t.lock_length (*C var*), 61
zsf_param_t.lock_width (*C var*), 61
zsf_param_t.num_cycles (*C var*), 61
zsf_param_t.rtol (*C var*), 62
zsf_param_t.salinity_lake (*C var*), 62
zsf_param_t.salinity_lock (*C var*), 61
zsf_param_t.salinity_sea (*C var*), 62
zsf_param_t.ship_volume_lake_to_sea (*C var*), 61
zsf_param_t.ship_volume_sea_to_lake (*C var*), 61
zsf_param_t.sill_height_lake (*C var*), 62
zsf_param_t.sill_height_sea (*C var*), 62
zsf_param_t.symmetry_coefficient (*C var*), 61
zsf_param_t.temperature_lake (*C var*), 62
zsf_param_t.temperature_sea (*C var*), 62
zsf_phase_state_t (*C struct*), 65
zsf_phase_state_t.head_lock (*C var*), 65
zsf_phase_state_t.salinity_lock (*C var*), 65
zsf_phase_state_t.saltmass_lock (*C var*), 65
zsf_phase_state_t.volume_ship_in_lock (*C var*), 65
zsf_phase_transports_t (*C struct*), 65

zsf_phase_transports_t.discharge_from_lake (*C var*), 65
zsf_phase_transports_t.discharge_from_sea (*C var*), 66
zsf_phase_transports_t.discharge_to_lake (*C var*), 65
zsf_phase_transports_t.discharge_to_sea (*C var*), 66
zsf_phase_transports_t.mass_transport_lake (*C var*), 65
zsf_phase_transports_t.mass_transport_sea (*C var*), 65
zsf_phase_transports_t.salinity_to_lake (*C var*), 65
zsf_phase_transports_t.salinity_to_sea (*C var*), 66
zsf_phase_transports_t.volume_from_lake (*C var*), 65
zsf_phase_transports_t.volume_from_sea (*C var*), 65
zsf_phase_transports_t.volume_to_lake (*C var*), 65
zsf_phase_transports_t.volume_to_sea (*C var*), 65
zsf_results_t (*C struct*), 63
zsf_results_t.discharge_from_lake (*C var*), 63
zsf_results_t.discharge_from_sea (*C var*), 63
zsf_results_t.discharge_to_lake (*C var*), 63
zsf_results_t.discharge_to_sea (*C var*), 63
zsf_results_t.mass_transport_lake (*C var*), 63
zsf_results_t.mass_transport_sea (*C var*), 63
zsf_results_t.salinity_to_lake (*C var*), 63
zsf_results_t.salinity_to_sea (*C var*), 63
zsf_results_t.salt_load_lake (*C var*), 63
zsf_results_t.salt_load_sea (*C var*), 63
zsf_step_flush_doors_closed (*C function*), 66
zsf_step_phase_1 (*C function*), 66
zsf_step_phase_2 (*C function*), 66
zsf_step_phase_3 (*C function*), 66
zsf_step_phase_4 (*C function*), 66
zsf_version (*C function*), 67
ZSFUnsteady (*class in pyzsf*), 67