swEOS: Salt Water Equation of State

Release 1.0

Zhikui Guo, Lars Rüpke
Jörg Hasenclever, Vehling Falko

Nov 29, 2021

CONTENTS

Welcome to the SaltWater EOS Docs! Here you'll find resources for using SaltWater EOS and examples of what it can do.

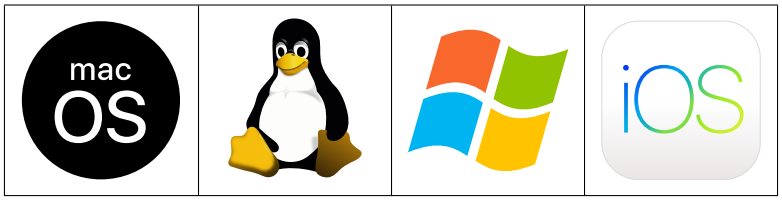
CHAPTER

ONE

INTRODUCTION

Realistic simulations of fluid flow in natural hydrothermal systems require accurate formulation of fluid properties especially for seawater convection at mid-ocean ridges. In order to explore circulation patterns and realistic phase seperation phenomenon, we have to calculate equation of state of binary salt-water fluids over pressure-temperature-salinity ranges encountered in the Earth’ s crust. Fortunately, pure water can be described by the IAPS-84 equation of state and [Driesner & Heinrich, 2007, Driesner, 2007] have developed a set of correction formulations of phase relations and thermodynamic properties for NaCl-H2O system. Further, we have developed a set of multi-language (C++ , Swift , Python , Matlab) and multi-platform (Windows , MacOS , Linux , IOS) tools, including callable C++ library , desktop application with graphical user interface (GUI), command line tools (just like gmt style), and Mobile apps for iphone and ipad . In addition, parallel computing is available for desktop application and command line tool.

1.1 Platforms



1.2 Applications

1.3 Supported Programing Languages

1.4 Authors and Developers

1.5 Cite

```
@software{zhikui_guo_2021_4603878,
   author       = {Zhikui Guo and
                   Lars Rüpke},
   title        = {{swEOS: multi-platform multi-language package of
                      salt-water equation of state}},
   month        = mar,
   year         = 2021,
   publisher    = {Zenodo},
   version      = {1.7.0},
   doi          = {10.5281/zenodo.4603878},
   url          = {https://doi.org/10.5281/zenodo.4603878}
}
```

CHAPTER

TWO

APP INSTALL

The pre-compiled applications include both desktop application with GUI (graphical user interface) and standalone command line tool (cmd) which can be used for batch calculation.

Tip: The desktop app with GUI also contains cmd features, it works well in macOS and linux (ubuntu), but there still are some issues in windows version. Therefore for the windows users, if you want to use cmd feature, please use the standalone cmd app.

2.1 macOS

2.1.1 Download

Please go to the download page of macOS installer of swEOS to download the proper version (e.g. Catalina or newer) according to your system version. If there is not a proper version, please look at the *Build from source* section for help.

2.1.2 Install

The downloaded installer is a `.dmg` file. Then you can get all the files (see Fig. 2.1) of swEOS app by simply double clicking the .dmg file and accepting the licence.

what's in the installer (snapshot)

what's in the installer (file tree)

Listing 2.1: File trees of macOS installer of swEOS app

```
/Volumes/swEOS-MacOSX-Installer
├── API
│   ├── c++
│   │   ├── CMakeLists.txt
│   │   ├── ReadMe.md
```

(continues on next page)

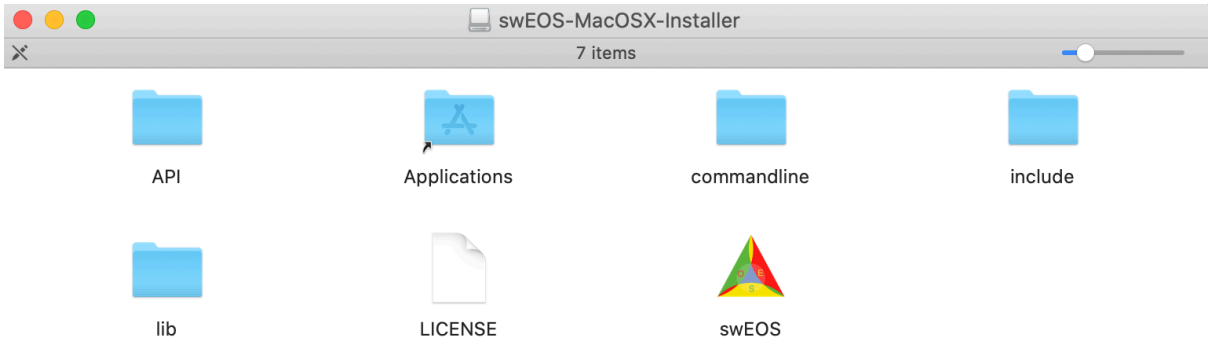


Fig. 2.1: Files in the installer

(continued from previous page)

```
|   |   └── main.cpp
|   ├── python
|   |   ├── H2O.py
|   |   ├── H2ONaCl.py
|   |   ├── NaCl.py
|   |   ├── PhaseDiagramSlice.py
|   |   ├── _H2O.so
|   |   ├── _H2ONaCl.so
|   |   ├── _NaCl.so
|   |   ├── test_H2O.py
|   |   ├── test_H2ONaCl.py
|   |   └── test_NaCl.py
|   └── tcl
|       ├── H2O
|       ├── H2ONaCl
|       ├── NaCl
|       ├── test_H2O.tcl
|       ├── test_H2ONaCl.tcl
|       └── test_NaCl.tcl
├── Applications -> /Applications
├── LICENSE
├── commandline
|   └── swEOS
├── include
|   ├── Gallery.H
|   ├── H2O.H
|   ├── H2ONaCl.H
|   ├── NaCl.H
|   ├── Polynomial.h
|   ├── PolynomialRootFinder.h
|   ├── Validation.H
|   ├── dataStruct_H2ONaCl.H
|   ├── stdfunc.H
|   └── steam4.h
```

(continues on next page)

(continued from previous page)

```
├── lib
│   └── libeosH2ONaCl.a
└── swEOS.app
    └── Contents
        ├── Frameworks
        ├── Info.plist
        ├── MacOS
        ├── PlugIns
        └── Resources
```

Desktop app with GUI

If you want to use the desktop app, just drag the `swEOS` (see Fig. 2.1) to `Applications` folder. This "drag" install process is the same as any other macOS app.

Tip: The app has not been notarized by Apple because there is no funding to support a Apple Developer ID. Therefore the `swEOS` app will be blocked by the Gatekeeper of macOS (see Apple support for more details). In order to allow swEOS to run on your macOS, you have to run the following command with superuser permission(`sudo`) in the terminal:

```
sudo xattr -r -d com.apple.quarantine /Applications/swEOS.app
```

Command line tool

The standalone command line tool(cmd) is also included in the installer, if you want to use this cmd app, just need to copy `commandline/swEOS` (see lines 27-28 in Listing 2.1) file to some directory (e.g. /usr/local/bin) in your file system, or make a symbol link to the environment PATH folder, e.g. `ln -s /Applications/swEOS.app/Contents/MacOS/swEOS /usr/local/bin`。The you can use the cmd app in the for batch calculation purpose (see the following animation).

Tip: If there is a error information of `permission denied: swEOS`, one can run command of `chmod 755 /usr/local/bin/swEOS` to change its file model as an executable.

Listing 2.2: Demo of running standalone cmd version of `swEOS` in terminal

```
$ cp /Volumes/swEOS-MacOSX-Installer/commandline/swEOS /usr/local/bin
$ chmod 755 /usr/local/bin/swEOS
$ sudo xattr -r -d com.apple.quarantine /usr/local/bin/swEOS
$ /usr/local/bin/swEOS

```

(continues on next page)

(continued from previous page)

```
************************************************
*                 program swEOS                *
*                 ~~~~~~~ ~~~~~~~              *
*  Version: 1.7.0-git-c5e9907                  *
*                                              *
*  Equation of state of salt-water (H2O-NaCl)  *
*  - Independent variables: PTX, PHX           *
*  - Properties: density, enthalpy, viscosity  *
*  - saturation, salinity, phase diagram       *
*  unit:                                       *
*      temperature-°C,        pressure-bar     *
*      salinity-wt. % NaCl,   density-kg/m3    *
*      enthalpy-kJ/kg,        viscosity-Pa s   *
*                                              *
* (c) Zhikui Guo, GEOMAR, 2021-03-16, Kiel     *
*                                              *
************************************************
```

2.2 Windows

2.2.1 Download

Please go to the download page of Windows installer of swEOS to download the proper version (e.g. windows 10) according to your system version.

2.2.2 Install

The downloaded installer is a `.exe` file. Then double click it to install just like any other general windows software installer, there is nothing special. After successfully installing, one can get something like Fig. 2.2.

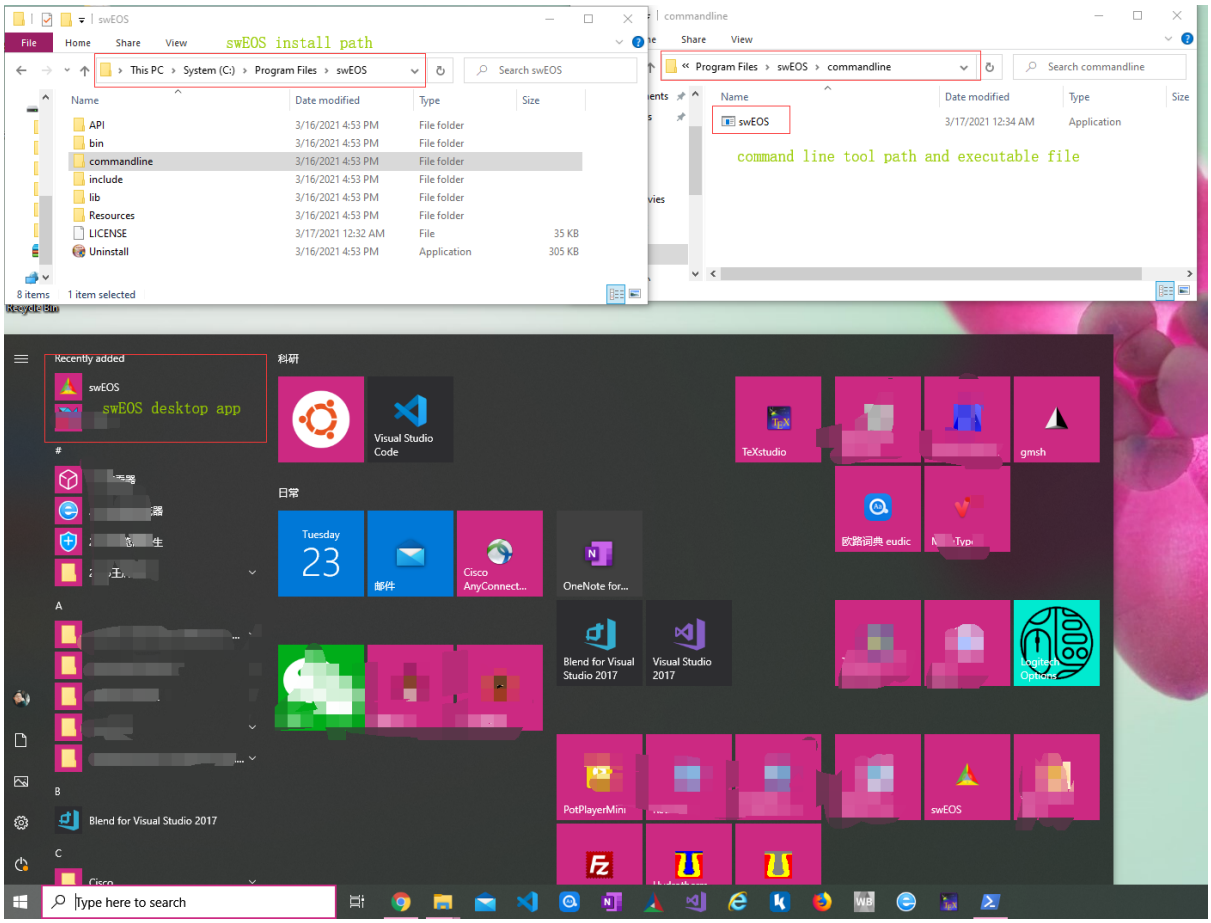


Fig. 2.2: Snapshot of swEOS desktop app installed in windows 10 system.

Desktop app with GUI

If you want to use the GUI version of swEOS, just click the shortcut automatically added in the start menu (see Fig. 2.2).

Command line tool

The standalone command line tool(cmd) is also included in the installer, if you want to use this cmd app, just need to add the command line tool install path (see Fig. 2.2) to the system environment variable of `PATH` (see Fig. 2.3). Then `swEOS` command can be used in PowerShell or traditional cmd terminal.

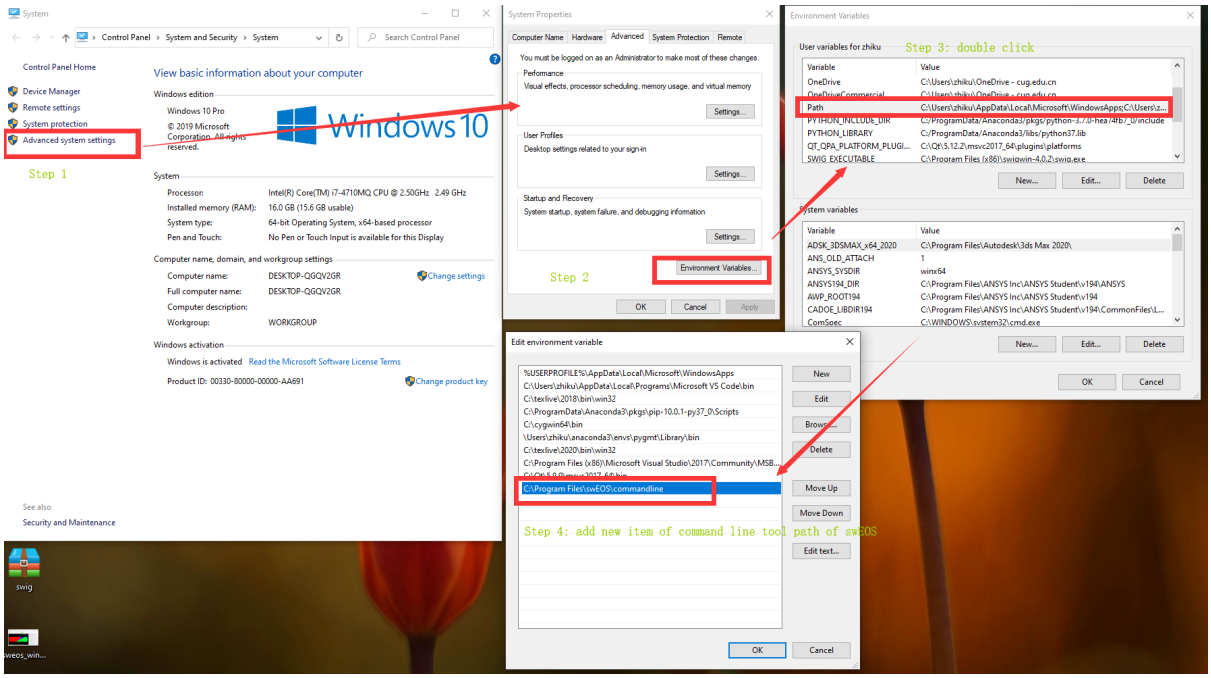


Fig. 2.3: Snapshot of environment variable setting.

2.3 Linux

2.3.1 Download and install

For Linux system, the install process is pretty easy. Similar to *macOS*, download the `.zip` installer from download page of Linux installer of swEOS and unzip it to wherever you like, e.g. `/home/swEOS`.

Desktop app with GUI

Run the following command in terminal to launch the swEOS GUI version.

```
/home/swEOS/swEOS.sh
```

Command line tool

There are two ways to use the command line tool.

1. Set arguments and options directory after `swEOS.sh`, for example `/home/swEOS/swEOS.sh -h`, because the command line arguments and options is also integrated in the GUI version.
2. Use the standalone command line tool, for example `/home/swEOS/CommandLineTool/swEOS`

Tip:

1. If there is a error information of `permission denied` after run `/home/swEOS/CommandLineTool/swEOS` command, please set the swEOS command line tool as executable mode. This can be done by running `chmod 755 /home/swEOS/CommandLineTool/swEOS`
2. If you want to access swEOS much easier, please add the command line tool path to the environment variable, just run command of `echo "PATH="/home/swEOS/CommandLineTool:$PATH"" >>~/.bashrc`.

CHAPTER

THREE

BUILD FROM SOURCE

3.1 Build c++ library

One can download the source code and save it to wherever you like, assuming save the source to `Download` download folder.

2. How to do ?

macOS

Requirements: basic development environment

- CMake : >=3.3
- C++ compiler
- make
- git (optional)

```
# 1. clone source code from github, or just download the source code and skip this␣
↪step.
git clone https://github.com/zguoch/saltwatereos.git

# 2. check directory in to Library folder of the source code
cd Library

# 3. create a build folder
mkdir build
cd build

# 4. cmake without building other APIs
cmake -DBuild_API_MultiLanguage=OFF ..

# 5. build the lib: you will get libeosH2ONaCl.a in the build folder
make
```

(continues on next page)

(continued from previous page)

```
# 6. install the lib to ../lib path, so the ../lib path will be created and␣
↪libeosH2ONaCl.a is copied in this path
make install
```

Linux

Requirements: basic development environment

- CMake : >=3.3
- C++ compiler
- make
- git (optional)

```
# 1. clone source code from github, or just download the source code and skip this␣
↪step.
git clone https://github.com/zguoch/saltwatereos.git

# 2. check directory in to Library folder of the source code
cd Library

# 3. create a build folder
mkdir build
cd build

# 4. cmake without building other APIs
cmake -DBuild_API_MultiLanguage=OFF ..

# 5. build the lib: you will get libeosH2ONaCl.a in the build folder
make

# 6. install the lib to ../lib path, so the ../lib path will be created and␣
↪libeosH2ONaCl.a is copied in this path
make install
```

Windows 10: Visual Studio

Requirements: basic development environment

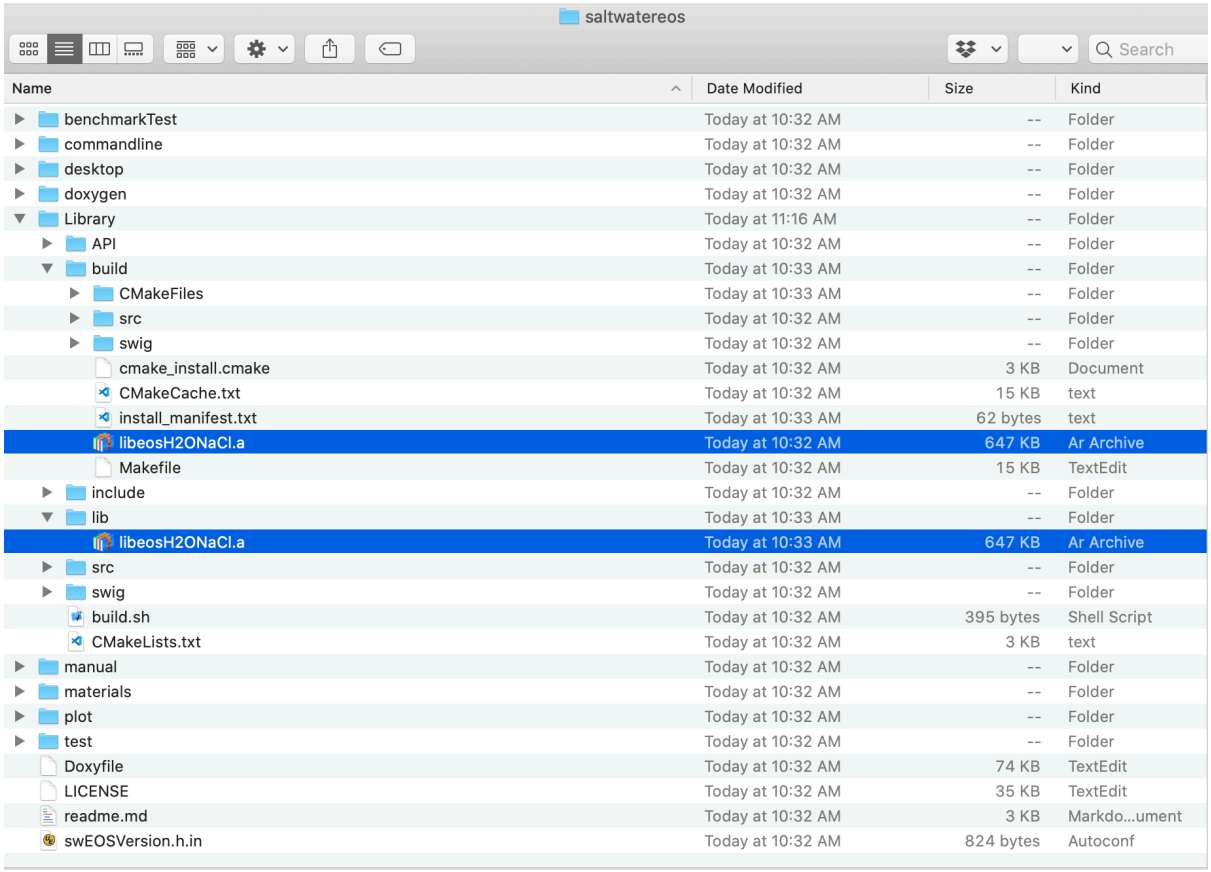
- CMake : >=3.3
- Visual Studio 2017 Community
- git (optional)

The Visual Studio 2017 Community have to be installed and the path of *MSBUILD.exe* is added in the system environment PATH variable. All the following steps are performed in PowerShell.

```
# 1. clone source code from github, or just download the source code and skip this␣
↪step.
git clone https://github.com/zguoch/saltwatereos.git

# 2. check directory in to Library folder of the source code
cd Library

# 3. create a build folder
mkdir build
cd build

# 4. cmake without building other APIs
cmake -DBuild_API_MultiLanguage=OFF -DCMAKE_BUILD_TYPE=Release -DCMAKE_GENERATOR_
↪PLATFORM=x64 ..

# 5. build using msbuild.exe: then you will find eosH2ONaCl.lib is generated in␣
↪the Release folder
msbuild /m /p:Configuration=Release eosH2ONaCl.vcxproj

# 6. install (optional): the eosH2ONaCl.lib is copied to ../lib folder
msbuild /m /p:Configuration=Release INSTALL.vcxproj
```

3. What will you get ?

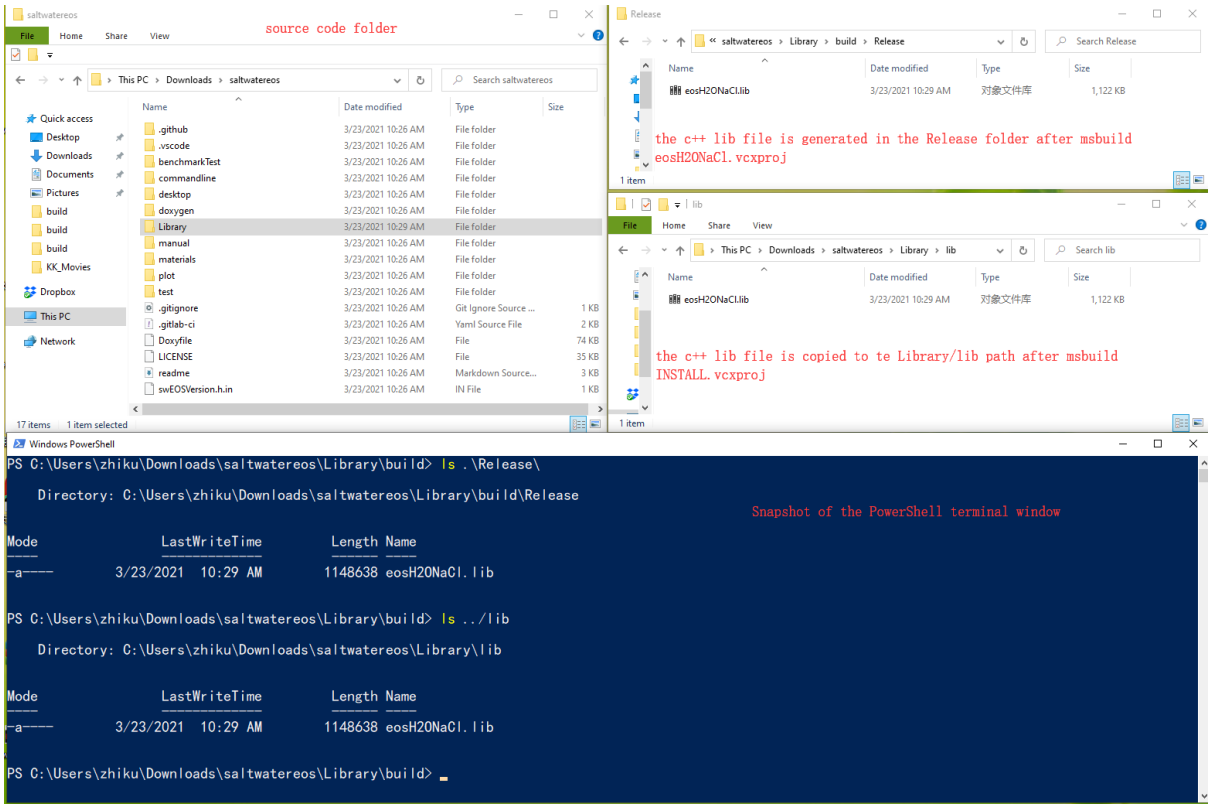
macOS



Linux

need to make snapshot

Windows 10: Visual Studio



3.2 Build standalone command line tool

Important: If you want to compile the standalone command line tool by yourself, you have to finish the previous step of *Build c++ library* firstly!

macOS

Requirements: basic development environment

- All requirements in *Build c++ library* step
- OpenMP: could use *brew install libomp* and check */usr/local/Cellar/libomp/*/include* path. The cmake will automatically detect the OpenMP include files and library files.

```
# 1. go to the commandline folder in the source code path
cd commandline

# 2. create build folder and check directory to the build folder
```

(continues on next page)

(continued from previous page)

```
mkdir build
cd build

# 3. cmake
cmake ..

# 4. build
make

# 5. check if swEOS is generated
ls

# 6. test it
./swEOS -h
```

Linux

Requirements: basic development environment

- All requirements in *Build c++ library* step
- OpenMP: could use *sudo apt-get install libomp*. The cmake will automatically detect the OpenMP include files and library files.

```
# 1. go to the commandline folder in the source code path
cd commandline

# 2. create build folder and check directory to the build folder
mkdir build
cd build

# 3. cmake
cmake ..

# 4. build
make

# 5. check if swEOS is generated
ls

# 6. test it
./swEOS -h
```

Windows 10: Visual Studio

Requirements: basic development environment

- All requirements in *Build c++ library* step

```
# 1. go to the commandline folder in the source code path
cd commandline

# 2. create build folder and check directory to the build folder
mkdir build
cd build

# 4. cmake
cmake -DCMAKE_BUILD_TYPE=Release -DCMAKE_GENERATOR_PLATFORM=x64 ..

# 5. build using msbuild.exe: the swEOS.exe will generated in the Release folder
msbuild /m /p:Configuration=Release swEOS.vcxproj

# 6. Test it
./Release/swEOS.exe -h
```

3.3 Build APIs

Warning: APIs of other programing language, e.g., python, tcl, js, depends on a lot of tools (e.g. swig, npm, python) and need a lot of programing skills. We don't recommend users to try out this unless they master cross-platform and multi-language programing skills. But users can use the APIs for research, for example the python API `pyswEOS` python users.

Of course, users can find details in the source code if they are interested in that.

3.4 Build Desktop App with GUI

Warning: Again, compilation of the desktop app with GUI also not that easy, because the GUI version is depends on Qt and VTK. Therefore, it also needs a lot of programing skills to reach that. If users are interested in building the GUI version by themselves, please read the source code.

CHAPTER

FOUR

TUTORIAL

4.1 Command line tool

4.1.1 Synopsis

swEOS [**-D** [*dimension*]] [**-V** [*variables*]] [**-P** [p_{bar}]] [**-T** [$T_{\circ C}$]] [**-X** [$x_{wt.NaCl}$]] [**-H** [$h_{kJ/kg}$]] [**-R** *min1/delta1/max1/min2/delta2/max2/min3/delta3/max3*] [**-G** [*inputfile*]] [**-O** [*outputfile*]] [**-t** [*threads*]] [**-n**] [**-h**]

Note: No space is allowed between the option flag and the associated arguments.

4.1.2 Description

swEOS can calculate phase relations and thermodynamic properties of salt water in single point, one-dimension, two-dimension and three-dimension, respectively.

4.1.3 Required Arguments

-D [*dimension*] Sets dimension. This is the first key option, the available arguments are **0**, **1**, **2** and **3**.

- **0** means **single point** calculation. If **-D** is set to **0**, the **-V** option only support **PTX** or **PHX**. In addition, the pressure, salinity, temperature or enthalpy must be specified by **-P**, **-X**, **-T** or **-H** option, respectively.
- **1** means only one variable changes and the others are set to fixed value. If **-D** is set to **1**, the **-V** option only support **P**, **T**, **X**, or **H**. The range of variable specified by **-V** option is set by **-R** option. And the other variables are set to fixed value by **-P**, **-X**, **-T** or **-H** options. The result will be saved to file. If the output file name is not specified by **-O** option, swEOS will use the default file name and print the file path in terminal.
- **2** means change two variables and fixed the third variable. Similar to **-D1** case.
- **3** means no fixed variable. Similar to **-D1** case.

-V [*variables*] Sets variables accroding to **-D** option. This is the second key option, the available arguments are **PTX**, **PHX**, **P**, **T**, **X**, **H**, **PT**, **PX**, **TX**, **PH** and **HX**.

- **PTX**, if argument of **-D** option is **0** (single point or multiple points case), then the pressure, temperature and salinity is set by **-P**, **-T** and **-X** option, respectively. While if argument of **-D** is **3** (three dimension

case), the range of pressure, temperature and salinity must be specified by **-R** option **in the same order** of argument of **-V** option. Therefore, for the same calculation, **-V** option can be **PTX**, **PXT**, **TPX**, **TXP**, **XPT** and **XTP** unless you set **-R** option in the save order. For example, `-VXPT -R0/0.1/1/5/1/400/0/1/100` means salinity in range of [0, 1] with interval of 0.1, pressure in range of [5, 400] bar with interval of 1 bar, temperature in range of [0, 100] °C with interval 1 °C . Alternately, you can also do the same thing using command of **-V** *TPX* **-R** *0/1/100/5/1/400/0/0.1/1* .

- **PHX**, similar to **PTX**, it just replaced temperature with enthalpy.
- **T**. This is only valid when argument of **-D** option is **1** (one dimension case). It means temperature is the independent variable, its range is specified by **-R** option, e.g. `-R0/1/100` means temperature in range of [0, 100] °C with interval of 1 °C, pressure and salinity are fixed by **-P** and **-X** option, respectively. In addition, the output file name has to be specified by **-O** option, will write as csv file format. **P**, **X**, **H** similar to **T**.
- **PT**. This is only valid when argument of **-D** option is set to **2** (two dimension case). It means pressure and temperature are the independent variable, their range are specified by **-R** option, e.g. `-R5/1/500/0/1/100` means temperature in range of [0, 100] °C with interval of 1 °C, pressure in range of [5, 500] bar with interval of 1 bar. Salinity is the fixed variable and the fixed value is specified by **-X** option. Note that it doesn' t matter what the order of variable follow **-V** option, e.g. `-VPT` and `-VTP` are all valid, but the order of range follow **-R** option matters, it must be the same order with **-V** option. For example, `-VPT -R5/1/500/0/100` and `-VTP -R0/1/100/5/1/500` are equivalent. Of course, you also have to specify output file by **-O** option. **PX**, **TX**, **PH**, **HX** are similar with **PT**.

-P [*pressure*] Sets fixed pressure value, it should be a float or integer number. The unit is **bar**.

-T [*temperature*] Sets fixed temperature value, it should be a float or integer number. The unit is °C.

-X [*salinity*] Sets fixed salinity value, it should be a float or integer number. The unit is $wt.NaCl$.

-H [*enthalpy*] Sets fixed enthalpy value, it should be a float or integer number.

-R [*min/delta/max*] Sets range and interval of independent variable(s), **-R** option must correspond to **-V** and **-D** options. For example, `-D1 -VT -R0/1/100` , `-D2 -VTX -R0/1/100/0/0.1/0.8` , `-D3 -VTXP -R0/1/100/0/0.1/0.8/5/1/500` .

-G [*inputfile*] Sets input file for multiple points calculation, the input file with three columns with delimiter of space or table(`\t`) correspondint to **-V** option. For example, `-D0 -VPTX -Ginput.txt` means calculate EOS of some points, independent variables of each point are pressure, temperature and salinity which are listed in `input.txt` as belows

PTX

Listing 4.1: Input file for multi-points calculation, three columns are `p(bar) T(deg.C) X(wt% NaCl)`

```
316 10      0.032
316 11      0.032
316 12      0.032
316 13      0.032
```

(continues on next page)

(continued from previous page)

```
316 14      0.032
316 15      0.032
316 16      0.032
316 17      0.032
316 18      0.032
316 19      0.032
316 20      0.032
316 21      0.032
316 22      0.032
316 23      0.032
```

PHX

Listing 4.2: Input file for multi-points calculation, three columns are `p(bar) H(kJ/kg) X(wt% NaCl)`

```
316     74.3015 0.032
316     78.2695 0.032
316     82.2389 0.032
316     86.2098 0.032
316     90.1822 0.032
316     94.1561 0.032
316     98.1316 0.032
316     102.109 0.032
316     106.087 0.032
316     110.067 0.032
316     114.049 0.032
316     118.031 0.032
316     122.016 0.032
316     126.001 0.032
316     126.001 0.032
```

-O [***outputfile*** **]** Sets output file name for one, two and three-dimensional calculation. The supported file format for 1D is csv, for 2D and 3D can be one of txt(delimiter is `\t`), csv(delimiter is `,`) and vtk. Because temperature, pressure and salinity have different scaling, if open the vtk file directlly by paraview, you can not see anything in salinity dimension. Therefore there is a python script generated by `swEOS` can deal with the scaling issue automatically. You just need to type a command in terminal to visualize the result. e.g. `paraview --script=test3D.vtk.py`.

-t [***thread*** **]** Sets number of threads for parallel computing.

-n If need to normalize the coordinates of result in vtk file, it is only valid in 3D calculation. Because the coordinates of the results are (P,T,X), (P,H,X), ⋯with different scale, if use paraview to visualize the result, one have to deal with the coordinate scale. One option is set using **-n** to normalize the coordinate. Otherwise, a python script will be generated by swEOS automatically, one can use this python script to visualize results (see also *Three-dimensional calculation*).

```
3D calculation using 8 threads, H in [100, 600] kJ/kg, P in [10, 500] bar, X␣
↪in [0.001, 1]
[##############################]100%
Writing results to file ...
Results have been saved to file: PTX.vtk
Paraview-python script is generated as : PTX.vtk.py
You can use command of  paraview --script=PTX.vtk.py to visualize result in␣
↪paraview
```

-h Print a short message about the syntax of the command.

4.1.4 Examples

Single point calculation

```
swEOS -D 0 -V PXT -P 316 -T 100 -X 0.032
swEOS -D 0 -V PXH -H 438 -P 316 -X 0.032
```

Multi-points calculation

```
swEOS -D 0 -V PHX -G phx.txt -O PHX_0D.csv
swEOS -D 0 -V PTX -G ptx.txt -O PTX_0D.csv
```

One-dimensional calculation

```
swEOS -D 1 -V H -X 0.032 -P 399 -R 43/1/100 -O H_1D.csv
swEOS -D 1 -V T -X 0.032 -P 399 -R 0/1/100 -O T_1D.csv
swEOS -D 1 -V P -X 0.032 -T 100 -R 5/1/100 -O P_1D.csv
swEOS -D 1 -V X -T 100 -P 399 -R 0/0.001/1 -O X_1D.csv
```

Two-dimensional calculation

```
swEOS -D 2 -V PT -R 10/0.1/100/1/1/500 -X 0.032 -O PT_2D.vtk
swEOS -D 2 -V PX -R 100/0.1/800/0/0.01/1 -T 100  -O PX_2D.vtk
swEOS -D 2 -V TX -R 1/1/800/0/0.01/1 -P 100  -O TX_2D.vtk
swEOS -D 2 -V PH -R 100/1/800/100/1/700 -X 0.032  -O PH_2D.vtk
swEOS -D 2 -V XH -R 0.001/0.001/1/100/1/700 -P 200  -O XH_2D.vtk
```

Three-dimensional calculation

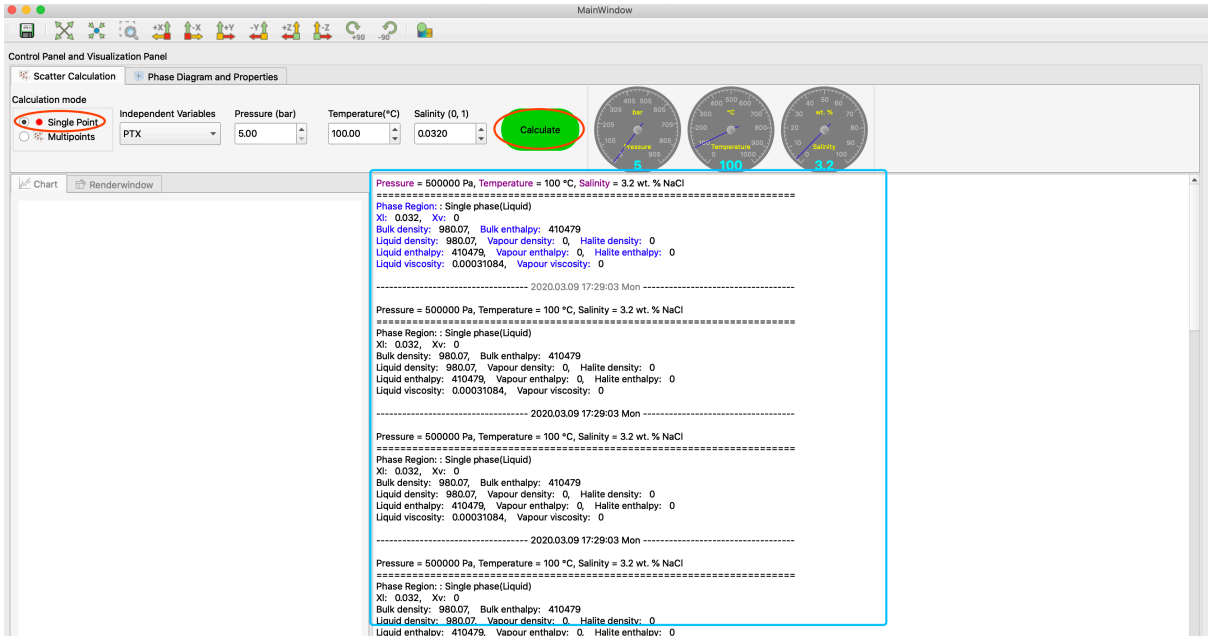
```
swEOS -D 3 -V PTX -R 10/10/500/1/10/600/0/0.01/1 -O PTX.vtk
swEOS -D 3 -V PHX -R 10/10/500/100/10/600/0.001/0.01/1 -O PTX.vtk
```

Phase region index table

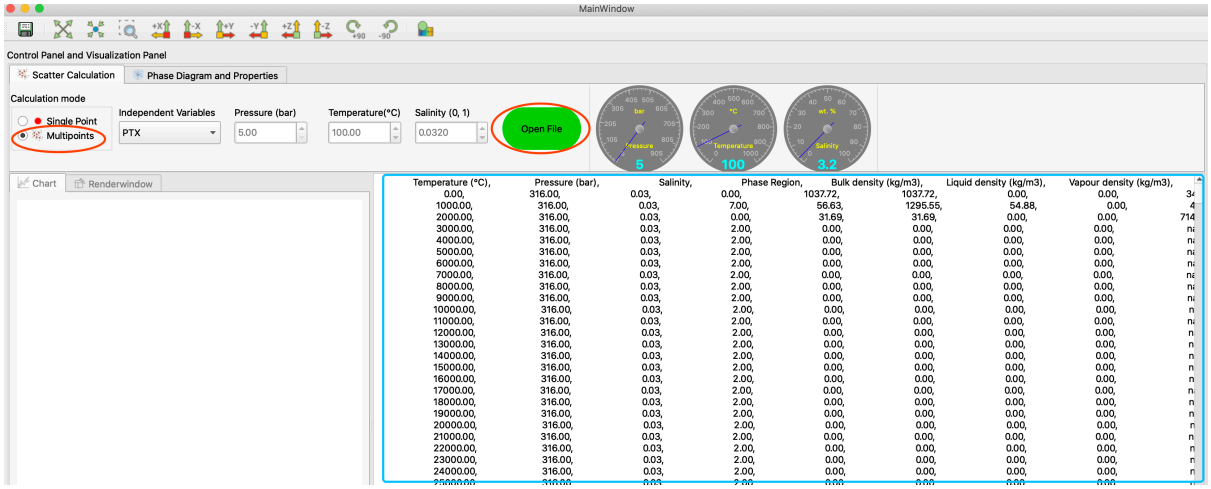
Phase region index	Phase region
0	Single phase(Liquid)
1	Liquid + Vapor at X=0
2	Pure vapour phase
3	Liquid + Halite
4	Vapour + Halite
5	Vapour + Liquid + Halite
6	Vapour + Liquid on the liquid side
7	Vapour + Liquid on the vapour side

4.2 Desktop app with GUI

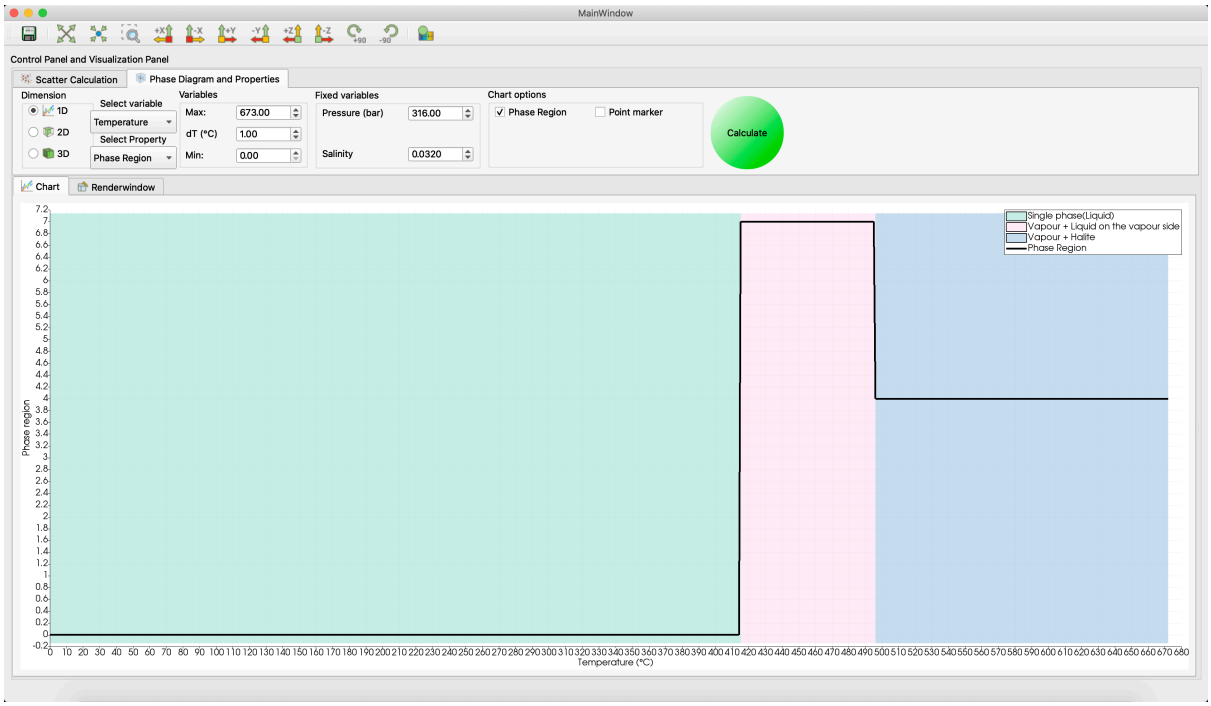
4.2.1 Single point calculation



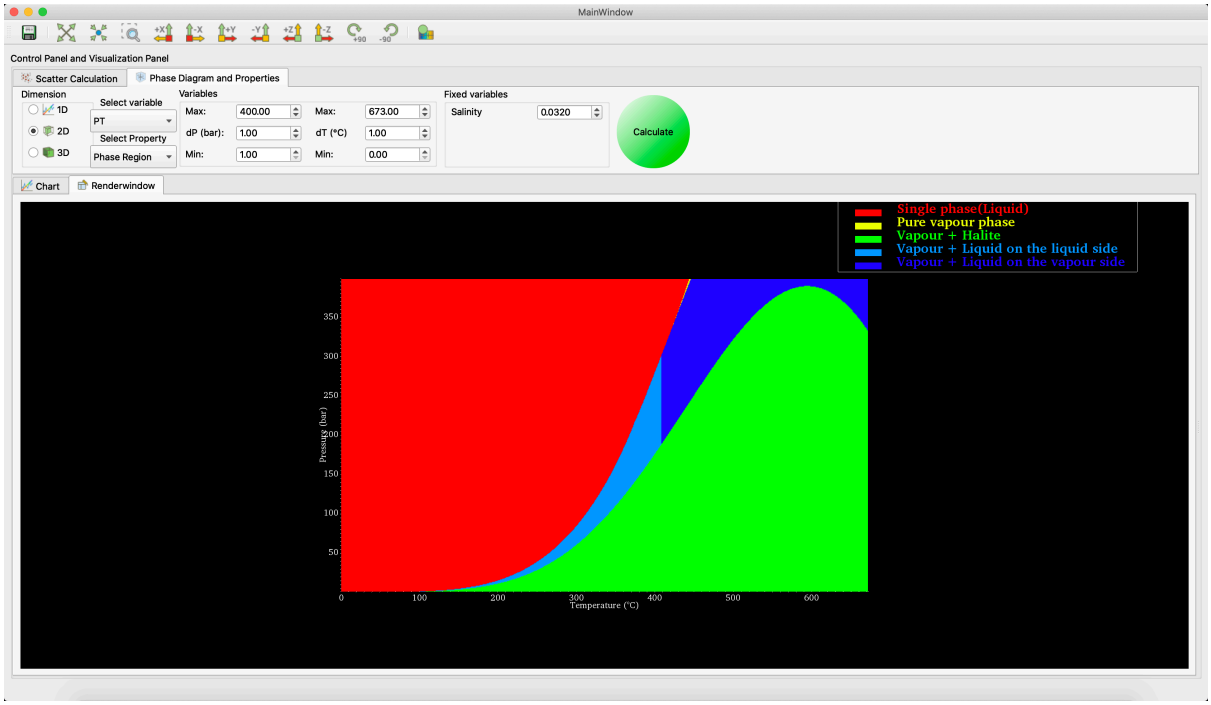
4.2.2 Multi-points calculation



4.2.3 1D chart



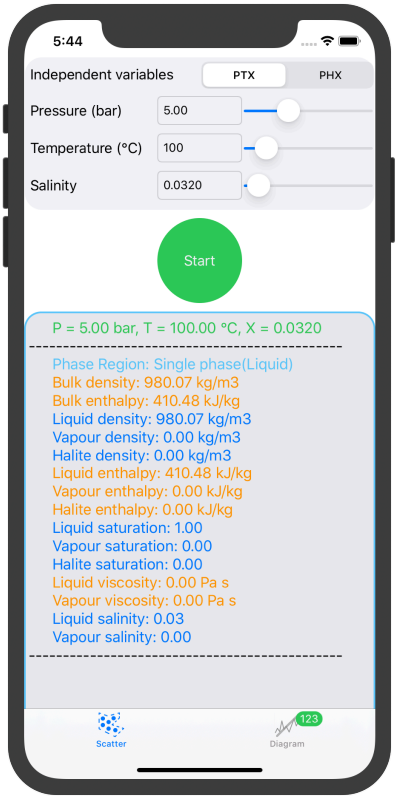
4.2.4 2D chart



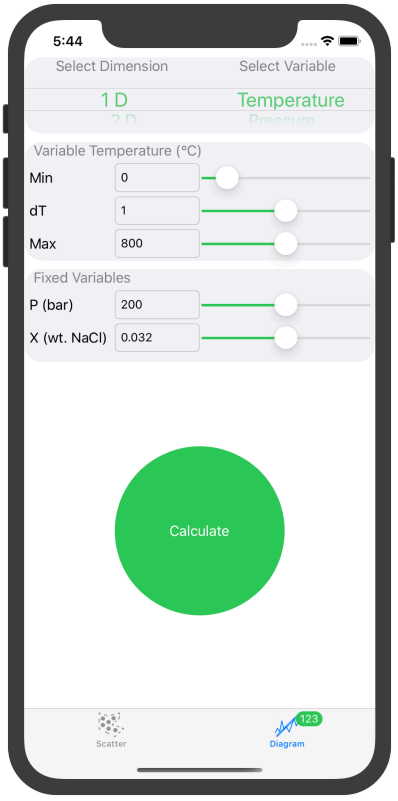
4.3 Apple ios app

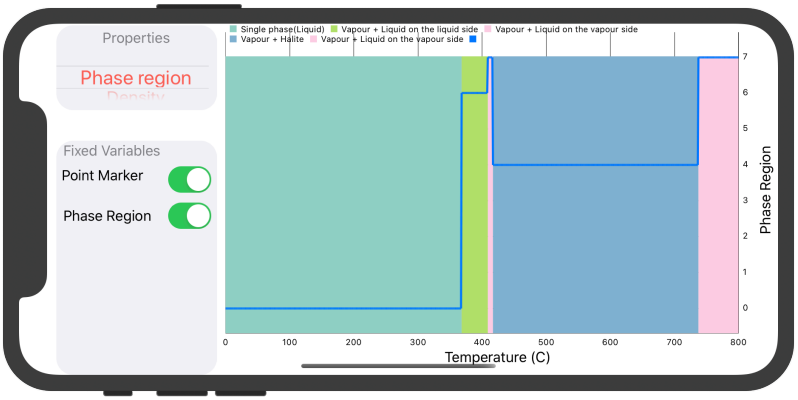
4.3.1 iPhone

Single point calculation

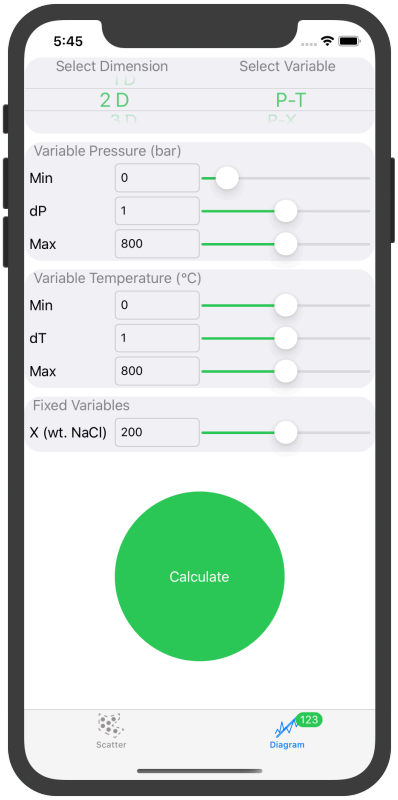


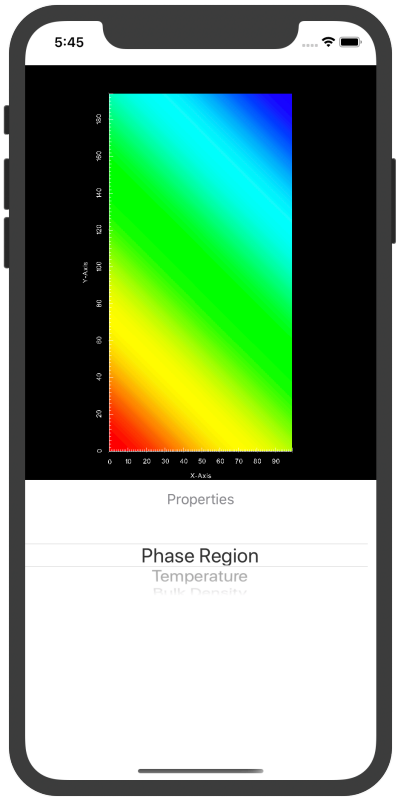
1D chart




Properties
Phase region
Fixed Variables
Point Marker
Phase Region
Single phase(Liquid)
Vapour + Liquid on the liquid side
Vapour + Liquid on the vapour side
Vapour + Halite
Vapour + Liquid on the vapour side
Phase Region
Temperature (C)

2D chart


5:45
Select Dimension
Select Variable
2 D
P-T
Variable Pressure (bar)
Min
dP
Max
Variable Temperature (°C)
Min
dT
Max
Fixed Variables
X (wt. NaCl)
Calculate
Scatter
Diagram



4.4 API

swEOS package not only contains cross-platform apps, but also contains API for several other programing languages. A simple demo of start point is shown in this section, see *Cookbooks* for more details of API usage.

4.4.1 C++

The C++ API is provided by head files and a library file named `libeosH2ONaCl.a` for macOS and linux, `libeosH2ONaCl.lib` for windows system, respectively. The head files and library file are contained in the app installer (in the `include` and `lib` folder), see also *App install* section.

Before starting to use c++ api of swEOS, the CMake and c++ compiler have to be installed.

How to start ?

Assuming the library file and head files has been downloaded and saved in `~/Download/swEOS` (please use your own path) path, it means that the `lib` folder and `include` folder are in `~/Download/swEOS` path.

Step 1. Create a source code folder and check directory to this folder.

Step 2. Create c++ source code, e.g. `main.cpp`, see Listing 4.3. Include the head file(line 1) and instantiate a object (line 5) of `cH2ONaCl` class, then all the properties and member functions can be accessed through object `eos`, e.g. density (line 9).

Listing 4.3: Source code of C++ api for calculating density of H2ONaCl.

```cpp
#include "H2ONaCl.H"
#include <iostream>
int main()
{
    H2ONaCl::cH2ONaCl eos;
    double p=200E5;    //Pa
    double T=400+273.15;    //K
    double X=0.2;    //wt.% NaCl
    double rho=eos.rho_pTX(p,T,X); //kg/m3
    std::cout<<" Pressure(bar): "<<p/1E5<<"\n"
             <<" Temperature(deg.C): "<<T-273.15<<"\n"
             <<" Salinity (wt.% NaCl): "<<X<<std::endl;
    std::cout<<" Density(kg/m3): "<<rho<<std::endl;
}
```

Step 3. Create `CMakeLists.txt`, see Listing 4.4.

Listing 4.4: CMake file

```cmake
# 0. CMake Minimum version
cmake_minimum_required(VERSION 3.3...3.12 FATAL_ERROR)

project(test_swEOS LANGUAGES CXX)
set(CMAKE_CXX_STANDARD 11)
set(CMAKE_CXX_STANDARD_REQUIRED ON)

# 1. set path of SWEOS library, which can be downloaded from https://github.com/zguoch/saltwatereos/releases according to your OS
set(SWEOS_DIR "../../" CACHE FILEPATH "Main path of SWEOS library")
if(EXISTS ${SWEOS_DIR}/include AND  EXISTS ${SWEOS_DIR}/lib/libeosH2ONaCl.a)
  message(STATUS "SWEOS head files found: " ${SWEOS_DIR}/include)
  message(STATUS "SWEOS libraries found: " ${SWEOS_DIR}/lib/libeosH2ONaCl.a )
else()
  message(FATAL_ERROR "Please specify path of SWEOS library which contains lib and include paths\n cmake -DSWEOS_DIR=path_of_SWEOS .. ")
endif()

aux_source_directory(. SRC_MAIN)
include_directories(${SWEOS_DIR}/include)
link_directories(${SWEOS_DIR}/lib)

add_executable(${PROJECT_NAME} main.cpp)
target_link_libraries(${PROJECT_NAME} eosH2ONaCl)
```

Step 4. Configure and generate project using CMake command line tool or GUI app.

Please remember set cmake cache variable of `SWEOS_DIR` to specify the swEOS library path which contains `lib`

and `include` folders.

```
mkdir build
cd build
cmake cmake -DSWEOS_DIR=~/Download/swEOS ..
```

Note: If `SWEOS_DIR` is not set or set incorrectly, you will get the following error information. Then you just make sure set a correct path for `SWEOS_DIR` to fix the problem.

```
CMake Error at CMakeLists.txt:14 (message):
Please specify path of SWEOS library which contains lib and include paths

    cmake -DSWEOS_DIR=path_of_SWEOS ..

-- Configuring incomplete, errors occurred!
```

Step 5. Compile and build the program.

In macOS or Linux system, just run `make` to compile and build the program. While, in Windows system and using Visual Studio 2017 Community, you will get a `.sln` file. Now, every VS user should know how to do!

Step 6. Run the program.

The executable program name is `test_swEOS`, which is configured in the CMakeLists file (see line 4 of Listing 4.4). Run this program in the terminal, you will get the output like this,

```
./test_swEOS

Pressure(bar): 200
Temperature(deg.C): 400
Salinity (wt.% NaCl): 0.2
Density(kg/m3): 188.056
```

More features

All right, now you must know how to use c++ api of swEOS package in your own program, see *Cookbooks* for usage of more functions.

pip install scipyFoam

CHAPTER

FIVE

COOKBOOKS

5.1 Phase region

findPhaseRegion

C++

Listing 5.1: Source code

```
#include "H2ONaCl.H"
#include <iostream>
H2ONaCl::cH2ONaCl eos;

int main()
{
    H2ONaCl::cH2ONaCl eos;
    double p=200; //bar
    double T=400; //deg.C
    double X=0.032; //wt.% NaCl
    H2ONaCl::PhaseRegion region=eos.findPhaseRegion(T, p, X);
    std::string region_name=eos.getPhaseRegionName(region);
    std::cout<<" Pressure(bar): "<<p<<"\n"
             <<" Temperature(deg.C): "<<T<<"\n"
             <<" Salinity (wt.% NaCl): "<<X<<std::endl;
    std::cout<<" Phase region index: "<<region<<std::endl;
    std::cout<<" Phase region name: "<<region_name<<std::endl;
}
```

python

Listing 5.2: Source code

```python
import pyswEOS
from pyswEOS import H2ONaCl
sw=H2ONaCl.cH2ONaCl()

p=200 #bar
T=400 #deg.C
X=0.032 #wt.% NaCl
region = sw.findPhaseRegion(T,p,X)
region_name=sw.getPhaseRegionName(region);
print(" Pressure(bar): ",p)
print(" Temperature(deg.C): ",T)
print(" Salinity (wt.% NaCl): ",X)
print(" Phase region index: ",region)
print(" Phase region name: ",region_name)
```

tcl

Listing 5.3: Source code

```tcl
load H2ONaCl H2ONaCl
puts "Phase Region"
cH2ONaCl sw
set region [sw findPhaseRegion 400 200 0.032]
puts $region
set region_name [sw getPhaseRegionName $region]
puts $region_name
```

CHAPTER

SIX

BENCHMARKS

6.1 Phase boundaries

6.1.1 Critical curve of H2O-NaCl system

P_X_Critical P_X_Critical calculate the critical pressure (P) and salinity (X) given temperature.

Chart

python

```
import pyswEOS
from pyswEOS import H2ONaCl
from pyswEOS import H2O
sw=H2ONaCl.cH2ONaCl()
# calculate critical curve by giving temperature
T=np.linspace(H2O.T_Critic,H2ONaCl.TMAX_C,100)
p,x = sw.P_X_Critical(T)
x,p = np.array(x)*100,np.array(p)
```

c++

```
#include "H2ONaCl.H"
H2ONaCl::cH2ONaCl eos;
double dT = (H2ONaCl::TMAX_C - H2O::T_Critic)/100;
for (double T = H2O::T_Critic; T <= H2ONaCl::TMAX_C; T=T+dT)
{
    double P,X;
    eos.P_X_Critical(T,P,X);
    cout<<T<<" "<<P<<" "<<X<<endl;
}
```

Below water critical point

The critical salinity at critical point of water is zero, so what's the relationship between critical pressure given by Equation 5a of Driesner(2007a) and boiling pressure given by IAPWS formula (e.g. IAPWS-IF97 TSat_P) ? The calculation results are quite different (up to 9 bar).

BIBLIOGRAPHY

[Driesner & Heinrich, 2007] Thomas Driesner and Christoph A. Heinrich. The system H2O-NaCl. Part I: Correlation formulae for phase relations in temperature-pressure-composition space from 0 to 1000 °C, 0 to 5000 bar, and 0 to 1 XNaCl. *Geochimica et Cosmochimica Acta*, 71(20):4880–4901, 2007. doi:10.1016/j.gca.2006.01.033.

[Driesner, 2007] Thomas Driesner. The system H2O-NaCl. Part II: Correlations for molar volume, enthalpy, and isobaric heat capacity from 0 to 1000 °C, 1 to 5000 bar, and 0 to 1 XNaCl. *Geochimica et Cosmochimica Acta*, 71(20):4902–4919, 2007. doi:10.1016/j.gca.2007.05.026.