Using Python with Eclipse and OPM

© Statoil ASA

Content

1. Python and Eclipse output files
    1. Installing the libecl Python bindings
    2. The Eclipse binary format
    3. The different classes
    4. Some examples

2. Sunbeam: Python wrappers for OPM-parser
    1. Build and install
    2. Examples

 © Statoil ASA

Statoil

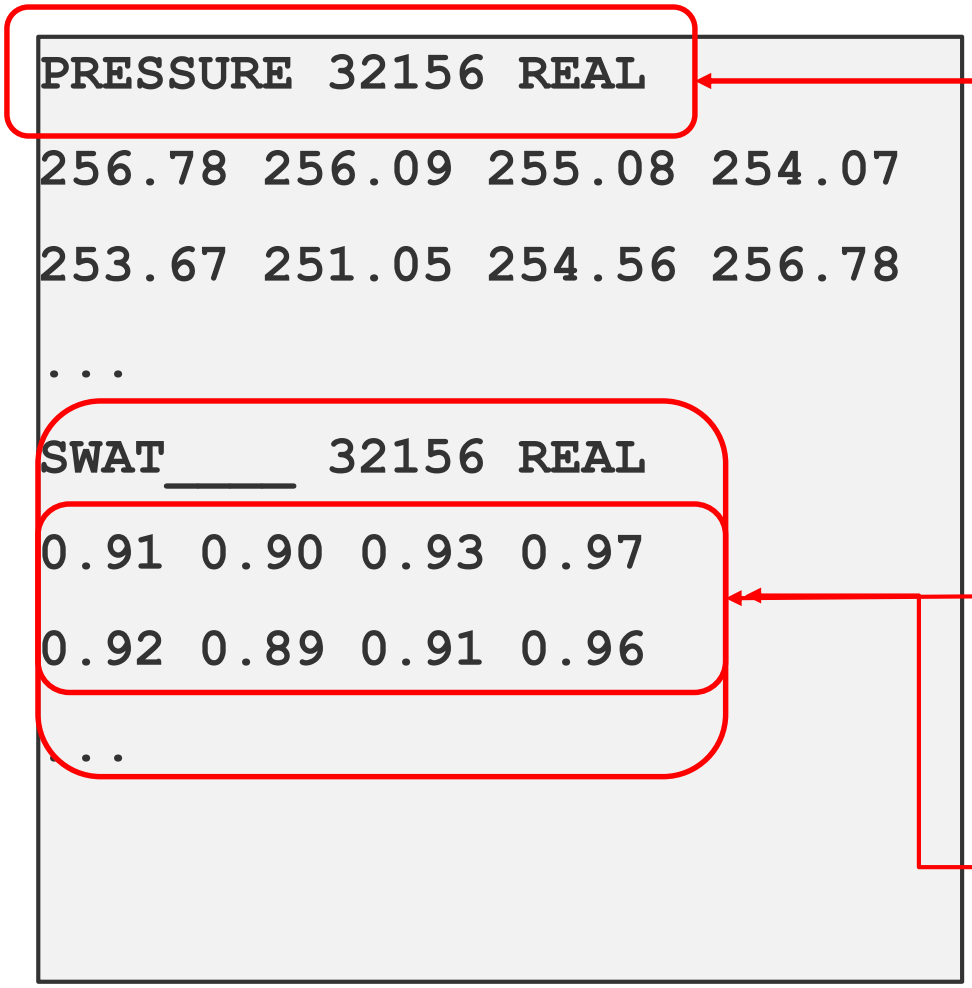
Installation

```
bash% git clone git@github.com:Statoi/libecl
bash% cd libecl
bash% mkdir build ; cd build
bash% cmake .. -DBUILD_PYTHON=ON -DCMAKE_INSTALL_PREFIX=$prefix
bash% make ; make install

bash% export PYTHONPATH=$prefix/lib/python2.7/{site|dist}-
packages
bash% pydoc ecl.ecl.EclGrid
```

 © Statoil ASA

Statoil

The ECLIPSE binary format



1. Name: 8 character string
2. Data size: integer
3. Data type: 4 character string

1. Fortran IO with embedded block sizes.
2. Endian swapping

Data in *blocks.*

EclKW

 © Statoil ASA

Statoil

Small libecl binary utilities

convert.x:

Will convert an ECLIPSE output file back and forth between binary and ASCII represenetation. Will consider file name extension to determine the desired conversion

```
bash% convert.x CASE.UNRST => CASE.FUNRST
```

ecl_pack.x / ecl_unpack.x

Will convert ECLIPSE summary and restart files between the unifed and multiple file formats

```
bash% ecl_pack.x CASE.X* => CASE.UNRST
```

kw_list.x

Will list all the keyword headers from an ECLIPSE output file:

```
bash% ecl_pack.x CASE.X* => CASE.UNRST
```

 © Statoil ASA

EclKW – «interactive» demo:

```
>>> from ecl.ecl import EclKW, EclDataType, openFortIO, FortIO, EclFile
>>> kw = EclKW("MY-KW", 1000, EclDataType.ECL_FLOAT)
>>> print len(kw)
1000
>>> kw[0:500] = 1
>>> kw[500:1000] = 2
>>> print kw.sum(), kw.get_min_max()
1500.0, 1, 2
>>> with openFortIO( "MY_FILE", FortIO.WRITE_MODE) as f:
...     kw.fwrite( f )
...
>>> f = EclFile("MY_FILE")
>>> kw2 = f["MY-KW"][0]
>>> print kw == kw2
True
>>> kw3 = 0.5*kw + kw2
```

Import the required symbols

Create a new EclKW

Access some properties

File IO

Basic arithmetic

 © Statoil ASA

Statoil

Reading restart file

SEQNUM	1:INTE
INTEHEAD	296:INTE
STARTSOL	0:MESS
PRESSURE	34770:REAL
SWAT	34770:REAL
SGAS	34770:REAL
RS	34770:REAL
RV	34770:REAL
ENDSOL	0:MESS

Report ste

```
>>> from ecl.ecl import EclFile
>>> rst_file = EclFile("ECLIPSE.UNRST")
>>> p_list = rst_file["PRESSURE"]
>>> p_list = [ p1, p2, p3 ]
```

SEQNUM	1:INTE
INTEHEAD	296:INTE
STARTSOL	0:MESS
PRESSURE	34770:REAL
SWAT	34770:REAL
SGAS	34770:REAL
RS	34770:REAL
RV	34770:REAL
ENDSOL	0:MESS

R

```
>>> from ecl.ecl import EclRestartFile
>>> rst_file = EclRestartFile("ECLIPSE.UNRST")
>>> view = rst_file.restart_view( report_step = 2 )
```

#st of keyword instances.

SEQNUM	1:INTE
INTEHEAD	296:INTE
STARTSOL	0:MESS
PRESSURE	34770:REAL
SWAT	34770:REAL
SGAS	34770:REAL
RS	34770:REAL
RV	34770:REAL
ENDSOL	0:MESS

Report step 3

© Statoil ASA

EclSummary

```
#!/usr/bin/env python
import sys
from ecl.ecl import EclSum


case = EclSum( sys.argv[1] )
 for key in case.keys('WWCT:*'):
      bt = case.solve_days( key , 0.25)
      print 'Water breakthrough in %s after %g days' % (key,
days)
```

 © Statoil ASA

EclRegion : select grid cells

```
#!/usr/bin/env python
from ecl.ecl import EclGrid, EclRegion, EclFile


grid = EclGrid('CASE.EGRID')
region = EclRegion(grid, False)
init_file = EclFile('CASE.INIT')


satnum = ini
poro = init_
permx = init_file['PERMX'][0]
...
permx.assign(0, mask = region)
with open('permx.grdecl','w') as f:
    permx.write_grdecl( f )
```

Intersect with thin cells:

```
region.select_thin( 0.1, intersect = True)
```

 © Statoil ASA

Statoil

Example 1

Find smallest and largest cell in grid and all thin cells.

```
bash% ./grid_info.py CASE
Smallest cell       : 37945.4
Largest cell        : 103195
Thin active cells : 2
(10,7,4)
(10,8,4)
```

 © Statoil ASA

Statoil

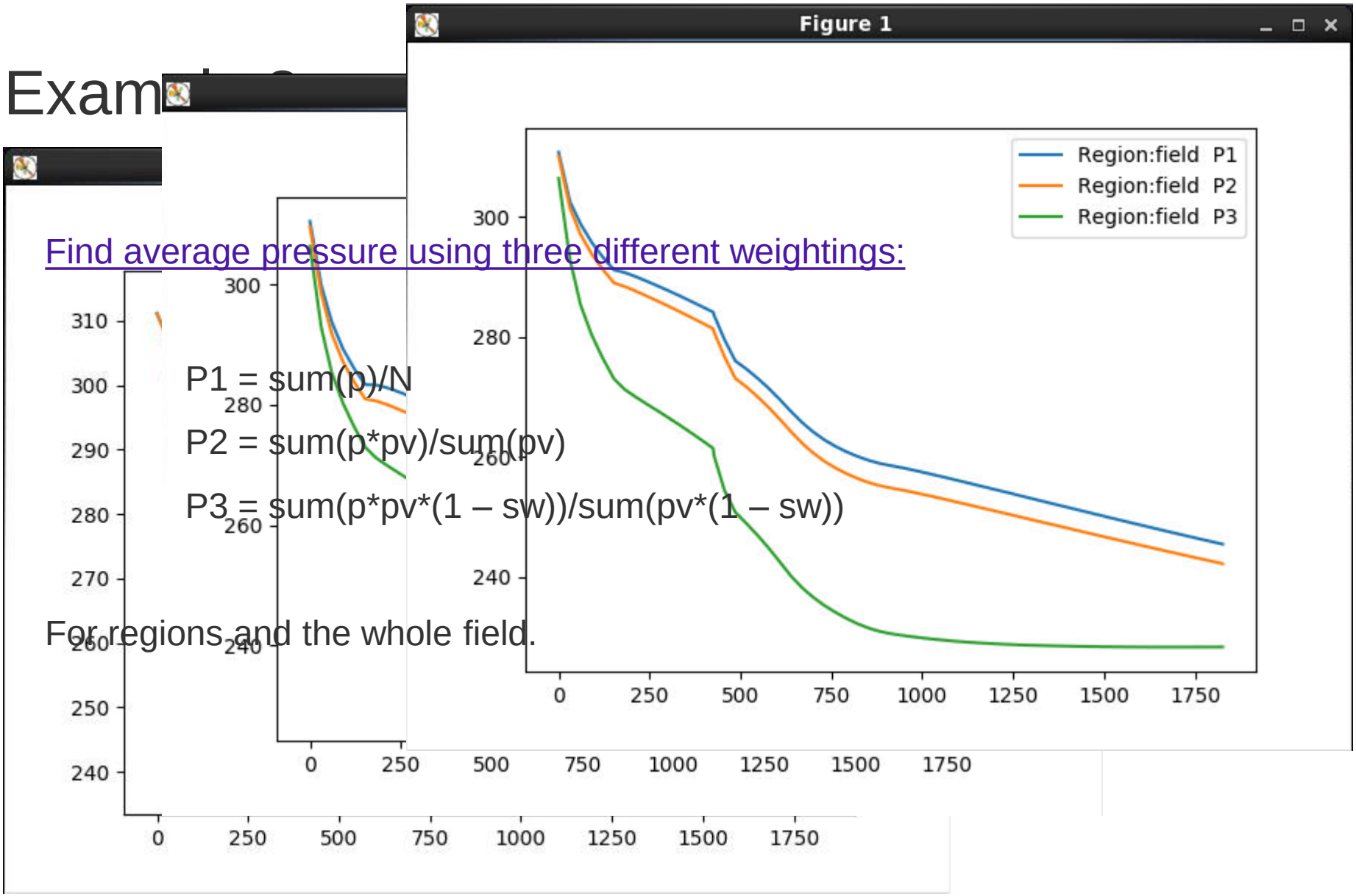
Example 2

Find connected cells and transmissibility for all NNC; sort by index.

```
bash%./cmp_nnc.py CASE
(35,02,01) -> (36,02,01)   T:2.18474
(35,02,02) -> (36,02,02)   T:2.00374
(35,02,03) -> (36,02,03)   T:1.09813
(35,02,04) -> (36,02,04)   T:2.05271
(35,02,05) -> (36,02,05)   T:2.30133
...
```

 © Statoil ASA

Exam



Find average pressure using three different weightings:

P1 = sum(p)/N

P2 = sum(p*pv)/sum(pv)

P3 = sum(p*pv*(1 – sw))/sum(pv*(1 – sw))

For regions and the whole field.

 © Statoil ASA

Sunbeam

Python wrappers for Opm-parser

 © Statoil ASA

Statoil

Installation

```
bash% git clone --recursive git@github.com:Statoi/sunbeam
bash% cd sunbeam
bash% mkdir build ; cd build
bash% c
bash% m
bash% c                                                    -
package
bash% pydoc sunbeam
```

NB: Currently sunbeam does not work with opm-parser master; need to use the tag: sunbeam-2017.10 from opm-parser.

© Statoil ASA

Sunbeam

- Based on Boost Python
- Very much work in progress
- Trying to strike a good balance between Pythonic look and feel and the existing C++ codebase.

 © Statoil ASA

Deck splitter 1

```
if __name__ == "__main__":
    input_deck = sys.argv[1]
    output_path = sys.argv[2]

    deck = sunbeam.parse_deck( input_deck )
    if not os.path.isdir( output_path ):
        os.makedirs( output_path )

    output_deck( deck , output_path, os.path.basename( input_deck ))
```

 © Statoil ASA

Deck splitter 2

```
#!/usr/bin/env python
import sunbeam
import sys
import os

def output_deck(deck, output_path, input_name):
    sections = set(["RUNSPEC","GRID", "PROPS", "REGIONS", "SCHEDULE", "SUMMARY"])
    baseH = open( os.path.join( output_path , input_name), "w")
    fileH = baseH
    for kw in deck:
        if kw.name in sections:
            baseH.write("INCLUDE\n   \'%s\' /\n\n" % kw.name)
            fileH = open( os.path.join( output_path, kw.name ), "w")
        fileH.write( str(kw) )
```

 © Statoil ASA

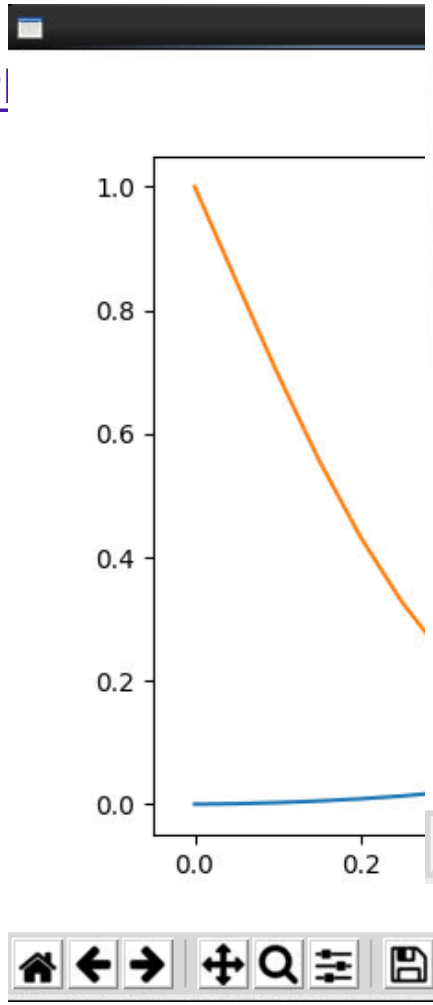
Well target rate:

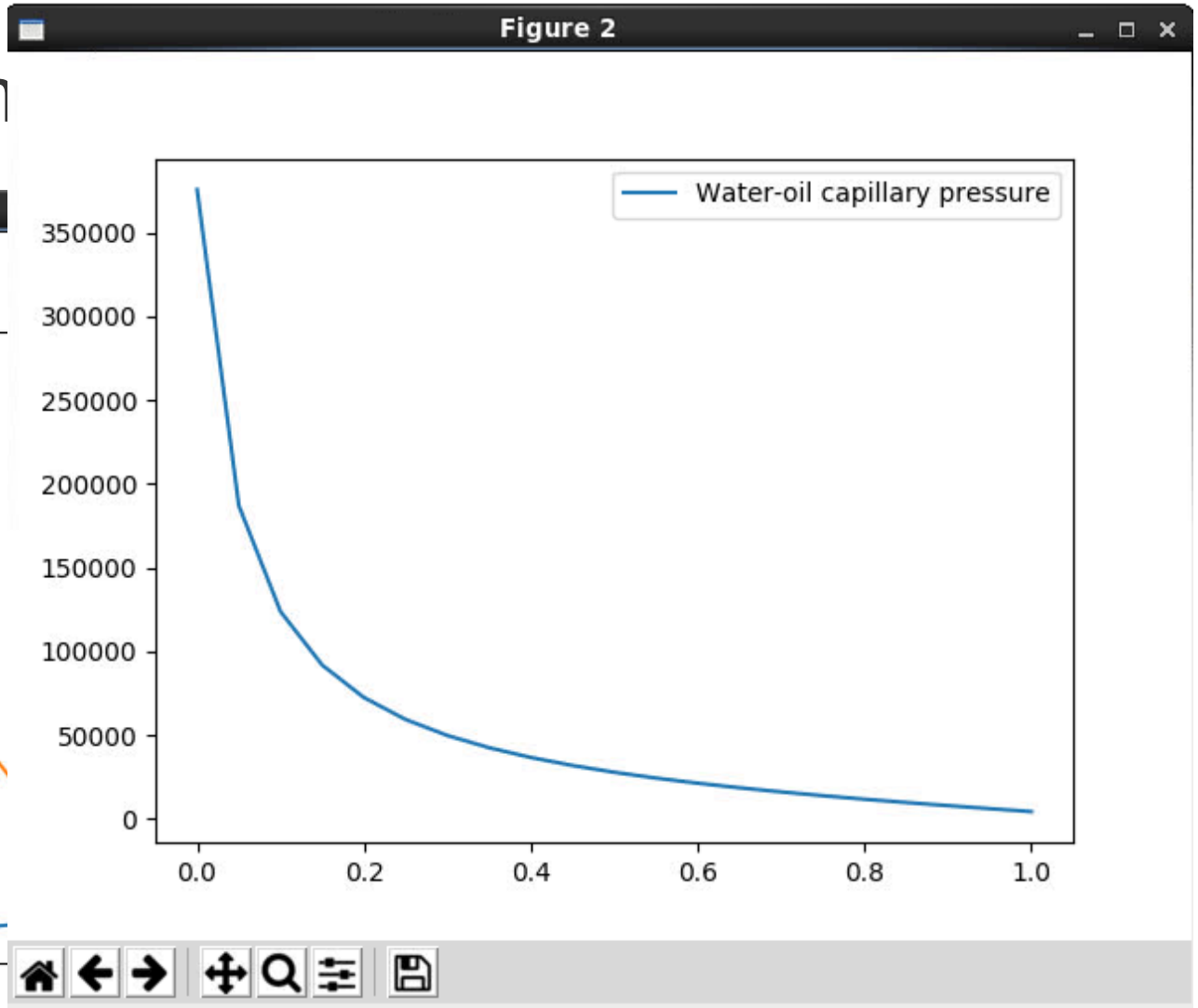
```
#!/usr/bin/env python
import sunbeam
import sys

ecl_state = sunbeam.parse( sys.argv[1] )
sched = ecl_state.sched
for i in len(sched.timesteps):
    print sched.timesteps[i],
    for well in ecl_state.sched.wells:
        print well.guide_rate,
    print
```

 © Statoil ASA

Table plottin

Pl





 © Statoil ASA

Statoil. The Power of Possible

Presentation title

Presenters name/title, etc

www.statoil.com

© Statoil ASA

This presentation, including the contents and arrangement of the contents of each individual page or the collection of the pages, are owned by Statoil. Copyright to all material including, but not limited to, written material, photographs, drawings, images, tables and data remains the property of Statoil. All rights reserved. Any other kind of use, reproduction, translation, adaption, arrangement, any other alteration, distribution or storage of this presentation, in whole or in part, without the prior written permission of Statoil is prohibited. The information contained in this presentation may not be accurate, up to date or applicable to the circumstances of any particular case, despite our efforts. Statoil cannot accept any liability for any inaccuracies or omissions.

Statoil