Updating Parameters in STARS by Repeated Restarts

The CMG family of simulators don't allow for e.g. dynamic permeability. Such behavior can however be modeled by setting up a sequence of restart runs. In this example we model a rock type which swells on contact with water, i.e. permeability will decrease with increasing water saturation. The model may or may not be realistic, but the point here is to demonstrate the methodology:

1. Run the simulator to a milestone time
2. Retrieve water saturation S_w from restart file at this time
3. Calculate permeability $K = K(S_w)$
4. Continue simulation, as a restart and with the revised permeability field as input permeability
5. Repeat 1 – 4

Comments:

This example is shown for STARS, but the procedure works equally well for IMEX or GEM.

The procedure has become easier to carry through after the 2016-version of CMG software, as result files are now stored in hdf-format, which can readily be read. In this example we use available python modules for general reading of hdf-files.

Detail Procedure Description

In the example we simulate 44 years of oil production in a reservoir produced by four oil producers and two water injectors. The permeability will be updated yearly the first ten years, bi-annually the next ten years and every third year the rest of the time. A total of 22 data files will be needed, and permeability files will be generated at each restart interval. Although most can be done manually, this easily becomes tedious and error-prone, so we prefer to do as much as possible by small programs that generate the data.

Hence we have defined

- A C++ program that pre-generates all the needed data files
- A python program that reads the result file (hdf-format), extracts the SW-data, and constructs the revised permeability field. The relationship between permeability and water saturation is hard-coded into the python program. In this example we use a simple, linear relationship, but this can be extended to include e.g. rock type dependent behavior and / or more complex relationships.
- A Windows (DOS) batch file to control the simulator / interrupt flow.
- Initial flow control / user input is given in a user defined control file.

Example Control Flow File (Windows .bat file)

This is a standard .bat file that defines the sequence of programs to run:

```
CALL st201610.exe -f WFex0.dat
REM starsPermMod.py <fileroot> <perm-ofile-nbr> rst-year rst-month rst-day
CALL python starsPermMod.py WFex0 1 2001 1 1
CALL st201610.exe -f WFexR1.dat
CALL python starsPermMod.py WFexR1 2 2002 1 1
CALL st201610.exe -f WFexR2.dat
…
…
CALL python starsPermMod.py WFexR20 21 2038 1 1
CALL st201610.exe -f WFexR21.dat
CALL python starsPermMod.py WFexR21 22 2041 1 1
CALL st201610.exe -f WFexR22.dat
PAUSE
```

After running the first STARS data file (WFex0.dat), the python program "starsPermMod.py" is run, outputting permeability file number 1 (i.e. "perm1.txt"), which was constructed from saturation data gathered from the hdf file WFex0.sr3 at time (2001 1 1). The "CALL" statements are used to ensure that the calls are done in sequence and ordered, no procedure is called before the previous one is finished. The flow can be improved by testing for successful outcome of the STARS simulations, but we didn't bother.

Example of User Defined Control File

This file is used as input to the program that generates the sequence of data (.dat) files. It includes a *file root name* from which the file name sequence will be constructed, and a list of restart dates that define the sequence. Example contents (# is used to define comments):

```
# Syntax: Froot <fileroot>
# List of restart dates
Froot WFex
RSTdates
2001 1 1
2002 1 1
2003 1 1
…
2035 1 1
2038 1 1
2041 1 1
```

Use: First define a “skeleton” file (in this example WFex.dat) which includes all restart dates, and numerical controls (e.g. by experience, DTWELL will need to be reduced at each restart date). The user’s responsibility is also to insert special comment lines in the skeleton data file, starting with **>. These lines are flags for insertion points for the restart-dependent data.

Example:

The first part of our WFex file (with flags highlighted):

```
** STARS Sw-dependent perm-red. Oct 2016

*TITLE1
'K(Sw) test 2016'

INUNIT SI

** INPUT / OUTPUT CONTROL =========
**> RESDATE XXXX
WPRN GRID 0
OUTPRN GRID NONE
OUTPRN RES NONE

WSRF WELL 1
WSRF GRID TIME
OUTSRF WELL DOWNHOLE
OUTSRF GRID PRES SO SW W CAPN LOGCAPN KRINTER
OUTSRF SPECIAL AVGVAR PRES
** GRID ===========================

GRID CART 52 89 1
KDIR DOWN

DI CON 25
DJ CON 50
DK CON 50
DTOP 4628*2380

POR CON   0.28
**PERMI CON 1100
**>  INCLUDE 'perm0.txt'

PERMJ EQUALSI
PERMK EQUALSI * 0.1
NETGROSS CON 0.9

END-GRID
…
```

The first data file in the sequence (WFex0.dat) will obviously be a standard file, so we concentrate on the first file to be restarted, **`WFexR1.dat`**. The first part of this file, as generated by the software, with context-dependent contents highlighted:

```
FILENAME INDEX-IN 'WFex0.irf'
** STARS Sw-dependent perm-red. Oct 2016

*TITLE1
'K(Sw) test 2016'

INUNIT SI

** INPUT / OUTPUT CONTROL =========
RESDATE 2001 1 1
WPRN GRID 0
OUTPRN GRID NONE
OUTPRN RES NONE

WSRF WELL 1
WSRF GRID TIME
OUTSRF WELL DOWNHOLE
OUTSRF GRID PRES SO SW W CAPN LOGCAPN KRINTER
OUTSRF SPECIAL AVGVAR PRES
OUTSRF SPECIAL DELP 'WI01' 'OP01'
** GRID ===========================

GRID CART 52 89 1
KDIR DOWN

DI CON 25
DJ CON 50
DK CON 50
DTOP 4628*2380

POR CON   0.28
**PERMI CON 1100
INCLUDE 'perm1.txt'

PERMJ EQUALSI
PERMK EQUALSI * 0.1
NETGROSS CON 0.9

END-GRID
…
```

In addition, the data file's run section will be terminated at the final restart date.

Example python program for constructing permeability data

```
## Original author: Roland Kaufmann (c)

import numpy as np
import makeVectors as mV
import sys
import h5py as hdf
import datetime as dat

datafile_root = sys.argv[1]
datafile = datafile_root + ".dat"

## Not flexible: initial perm is always found in "perm0.txt"
perminit = mV.readFloatVector("perm0.txt")
```

```
perm_ofileID = sys.argv[2]
permfile_root = "perm"
perm_ofile = permfile_root + str(perm_ofileID) + ".txt"

## This version of program assumes time steps have been defined as DATES
## Can modify for times if needed

resultFile = datafile_root + ".sr3"
rstyy = int (sys.argv [3])
rstmm = int (sys.argv [4])
rstdd = int (sys.argv [5])
fDate = rstyy*10000 + rstmm*100 + rstdd

with hdf.File (resultFile, 'r') as f:
    timetable = f ['General/MasterTimeTable']

    # get the starting time of the simulation
    Dates = timetable ['Date'][:]
    timeindx = int (np.where(Dates == fDate)[0])

    # map the timesteps to an index
    rstindx = timetable ['Index'][timeindx]

    # retrieve water saturation at this date
    sw = np.array (f ['SpatialProperties/{0:06d}/SW'.format (rstindx)] [:])

## Now we have Sw and initial perm. May also need swi (assume constant)
## More advanced (save for later): Formula depends on rock type --
reasonably easily handled by maskarrays

swinit = 0.14

## Simple formula of the kind permmodfactor = fact*(sw - swinit)/(1 -
swinit)
## Assuming multiplier is unity at swinit, and mval at sw = 1.
## perm-mult = c1*(sw - swinit) + 1
## c1 = (minmod - 1)/(1 - swinit)

minmod = 0.25
c1 = (minmod - 1.0)/(1.0 - swinit)

modperm = perminit * (1 + c1*(sw - swinit))

mV.writeVector(perm_ofile,"PERMI", modperm)
```

Results

As mentioned, the main purpose of this blog is to present the methodology; hence results are not that important. Still, the results of running this example by the presented procedure are shown in this section.

We immediately notice that the permeability reduction that was introduced didn’t do that much of a difference. The reduction was such that permeability would never go below 25% of initial permeability, which means the permeability is always moderately high. More drastic changes would probably have been obtained by allowing for very low permeabilities – but as said, that’s really not the issue here…

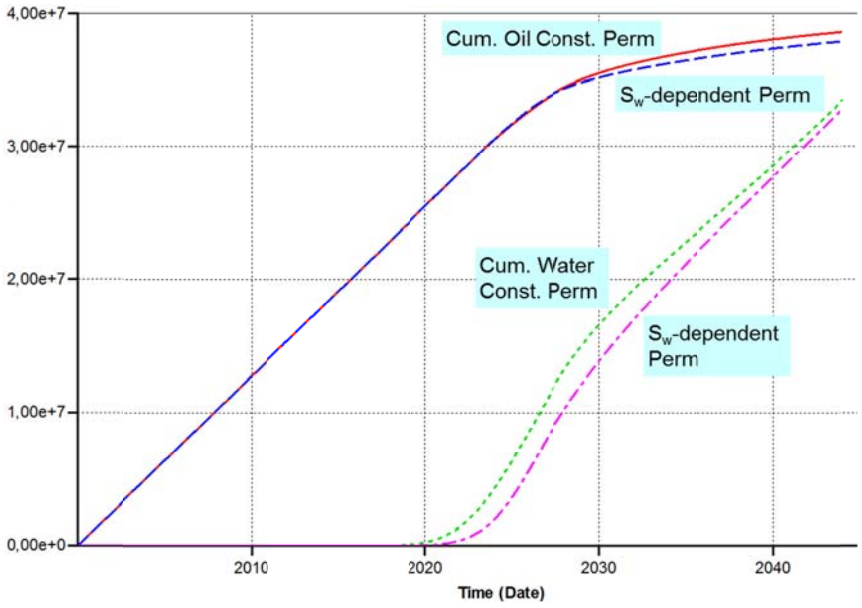

Figure 1. Cumulative production of oil and water. Comparison constant and Sw-dependent permeability

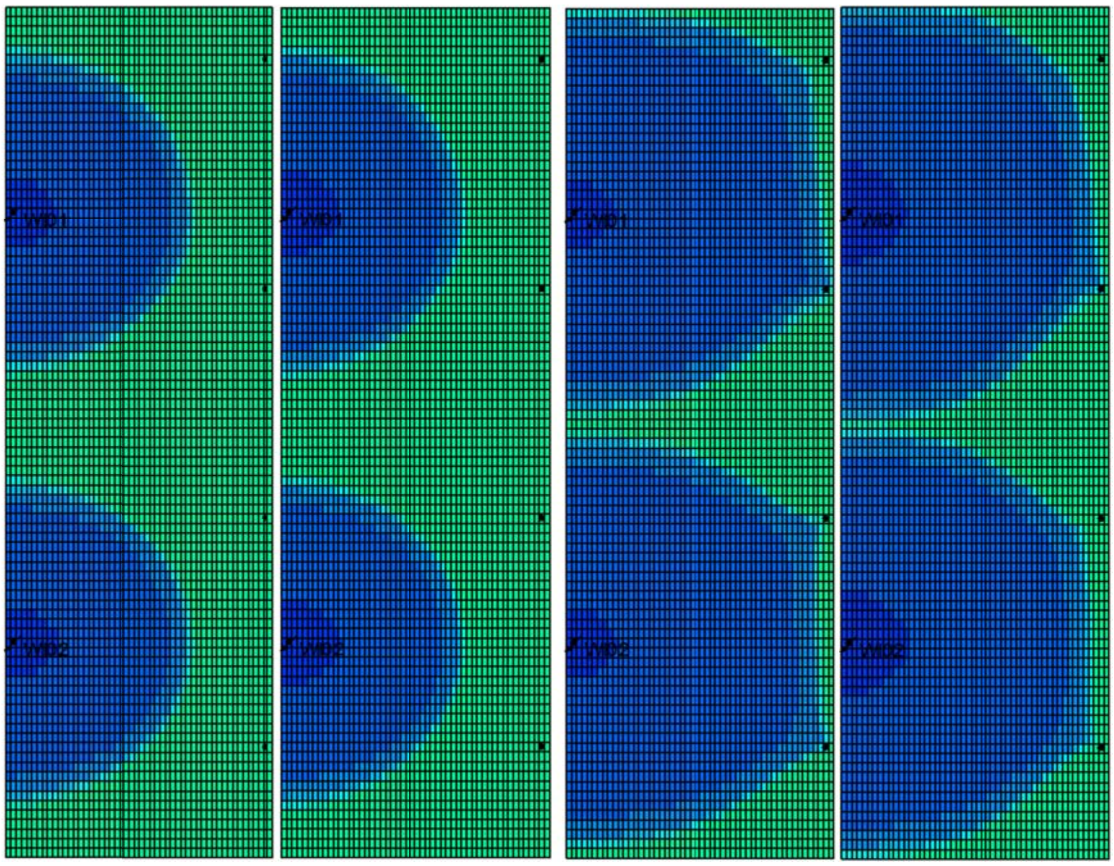

Figure 2. Water saturation fronts. 1) T = 5 years, constant perm – 2) Sw-dependent perm. 3) T = 20 years, constant perm – 4) Sw-dependent perm

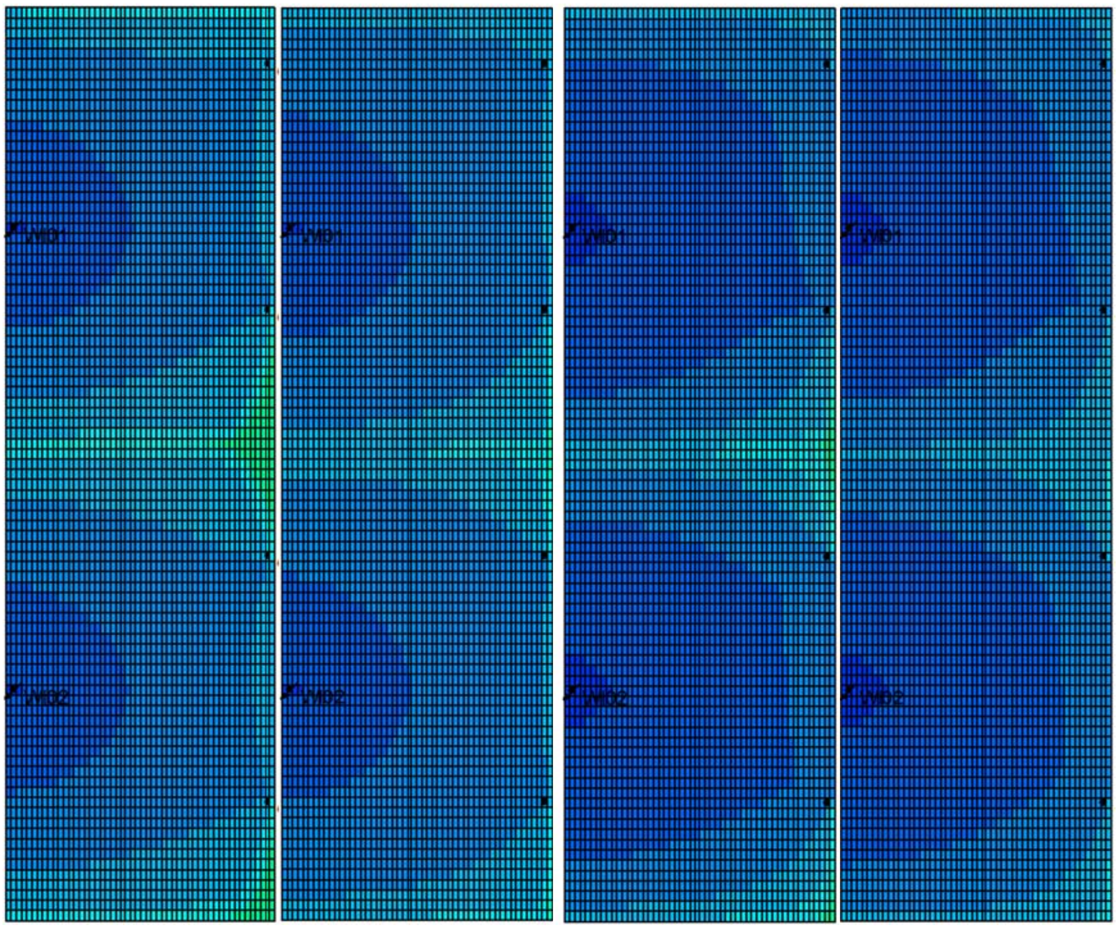

Figure 3. Water saturation fronts. 1) T = 32 years, constant perm – 2) Sw-dependent perm. 3) T = 44 years, constant perm – 4) Sw-dependent perm

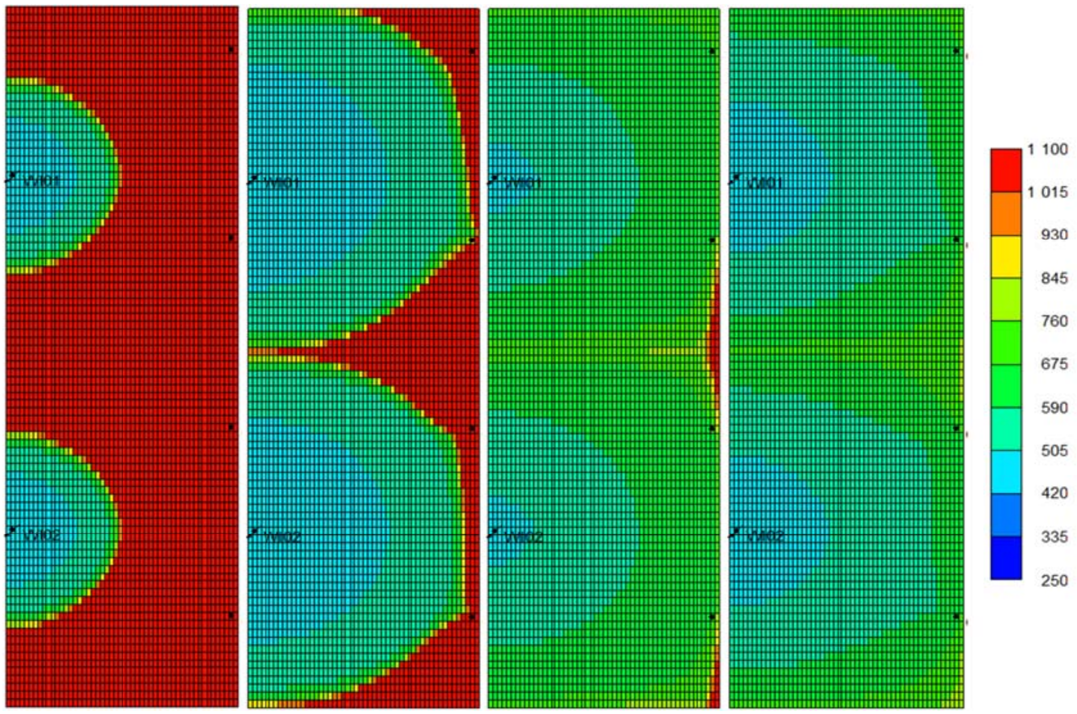

Figure 4. Permeability Field (Initial perm = 1100 mD), 1) 5 years 2) 20 years 3) 30 years 4) 42 year