The new version of DuMu^x including the modules "CRootBox" and "dumux-rosi"

Documentation

Manual written by A. Schnepf

December 16, 2019

Installation

This installation guidelines are for the new version of DuMu^x, version 3, coupled with CRootBox, in Linux systems (e.g. Ubuntu).

Required compilers and tools

If on a recent Ubuntu system, the c++ compiler and python that come with the distribution are recent enough. Otherwise, please make sure you have a recent c++ compiler (e.g. `sudo apt-get install clang`), fortran compiler (sudo apt-get install gfortran) and python3 (e.g. `sudo apt-get install python3.7`).
- Install git:
```
sudo apt-get install git
```
- Install cmake:
```
sudo apt-get install cmake
```
- Install libboost:
```
sudo apt-get install libboost-all-dev
```
- Install pip:
```
sudo apt-get install python3-pip
```
- Install the python package numpy:
```
pip3 install numpy
```
- Install the python package scipy:
```
pip3 install scipy
```
- Install the python package matplotlib:
```
pip3 install matplotlib
```
[1]
- Install the python package VTK:

alug `pip3 install vtk`
- Install the java runtime environment:
```
sudo apt-get install default-jre
```
- Install Paraview
```
sudo apt-get install paraview
```

[1] Known bug in ubuntu 18.04: needs sudo apt-get install libfreetype6-dev libxft-dev installed before.

DuMu^x installation

In all dune modules we stay in version 2.6, the latest stable release version.
- Create a DuMu^x working folder

mkdir DUMUX
cd DUMUX

- Download DUNE core modules:

```
git clone https://gitlab.dune-project.org/core/dune-common.git
cd dune-common
git checkout releases/2.6
cd ..
git clone https://gitlab.dune-project.org/core/dune-geometry.git
cd dune-geometry
git checkout releases/2.6
cd ..
git clone https://gitlab.dune-project.org/core/dune-grid.git
cd dune-grid
git checkout releases/2.6
cd ..
git clone https://gitlab.dune-project.org/core/dune-istl.git
cd dune-istl
git checkout releases/2.6
cd ..
git clone https://gitlab.dune-project.org/core/dune-localfunctions.git
cd dune-localfunctions
git checkout releases/2.6
cd ..
```

- Download DUNE external modules:

```
git clone https://gitlab.dune-project.org/extensions/dune-foamgrid.git
cd dune-foamgrid
git checkout releases/2.6
cd ..
git clone https://gitlab.dune-project.org/extensions/dune-grid-glue.git
cd dune-grid-glue
git checkout master[2]
cd ..
```

-Download dumux and dumux-rosi and the grid manageres alugrid, uggrid and spgrid:

```
git clone https://git.iws.uni-stuttgart.de/dumux-repositories/dumux.git
cd dumux
git checkout releases/3.0
```

[2]There is no release 2.6

```
cd ..
git clone https://github.com/Plant-Root-Soil-Interactions-Modelling/dumux-rosi.git
cd dumux-rosi
git checkout master
cd ..
git clone https://gitlab.dune-project.org/extensions/dune-alugrid.git
cd dune-alugrid
git checkout releases/2.6
cd ..
git clone https://gitlab.dune-project.org/staging/dune-uggrid
cd dune-uggrid
git checkout releases/2.6
cd ..
git clone https://gitlab.dune-project.org/extensions/dune-spgrid
cd dune-spgrid
git checkout releases/2.6
cd ..
```

-Download CRootBox (only needed if root growth is used):

```
git clone https://github.com/Plant-Root-Soil-Interactions-Modelling/CRootBox.git
cd CRootBox
git checkout master
cd ..
```

To build CRootBox and its python shared library, move again into the CRootBox folder and type into the console:

```
cd CRootBox
cmake .
make
```

[3]

(If building CRootBox on the cluster, two lines in the file CRootBox/CMakeLists.txt need to be outcommented before:

```
set(CMAKE_C_COMPILER "/usr/bin/gcc")
set(CMAKE_CXX_COMPILER "/usr/bin/g++")
```

)

Now build DuMu^x with the CRootBox module:

-The configuration file cmake.opts is stored in the dumux folder. Move a copy of this file to your DuMu^x working folder (one level up, DUMUX)

- To build all downloaded modules and check whether all dependencies and prerequisites are met, run dunecontrol:

```
cd ..
```

[3]It may be necessary on your installation to check the CRootBox/src/CMakeLists.txt file regarding required python version and out-commenting line 34.

```
./dune−common/bin/dunecontrol −−opts=cmake.opts all
```

Installation done! Good luck!

Running an example

```
cd  dumux−rosi/build−cmake/rosi_benchmarking/soil
make richards1d       # outcomment if executable is already available
./richards1d benchmarks_1d/b1a.input    # run executable with specific
    input parameter file
```

Installing and running an example on the agrocluster

- Before installing or running DuMu^x on the agrocluster, it is required to type the command `module load dumux` into the console. This sets the compiler versions and other tools to more recent versions than the standard versions of the agrocluster.
- On the cluster, another onfiguration file `optim_cluster.opts` is used. Copy this file to the file to your DuMu^x working folder (one level up).
- To build or run an example on the agrocluster, create a pbs file in your working folder that will put your job in the cluster queue
For example `queue_my_job.pbs`

```
#!/bin/sh
#
#These commands set up the Grid Environment for your job:
#PBS −N DUMUX
#PBS −l nodes=1:ppn=1,walltime=200:00:00,pvmem=200gb
#PBS −q batch\\
#PBS −M a.schnepf@fz−juelich.de
#PBS −m abe

module load dumux
cd  \$HOME/DUMUX/dumux−rosi/build−cmake/rosi_benchmarking/soil
make richards
./richards benchmarks_1d/b1a.input
```

To start the job, run this file in your working folder with the command
`qsub queue_my_job.pbs`
Use Filezilla to move the results to your local machine and use Paraview to visualize them.
If you need to install additional python packages (e.g. scipy) on the cluster (without root access), you may do so by using the `−−user` command:
`pip3 install −−user scipy`

Numerical grids

We distinguish two types of numerical grids: the 3D soil grid and the 1D, branched, root system grid (see Fig. ??).

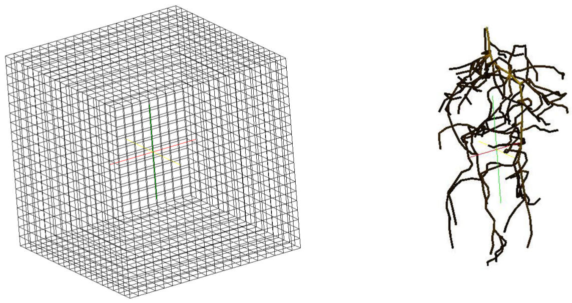

Figure 1: The 3D soil grid and the 1D, branched, grid representing the root architecture

In the example of the coupled problems, both are used simultaneously. In that case, the two grids are merged via source/sink terms in positions where root and soil grids share the same spatial coordinates. This is illustrated in Fig. ??; detailed descriptions can be found in the individual examples.

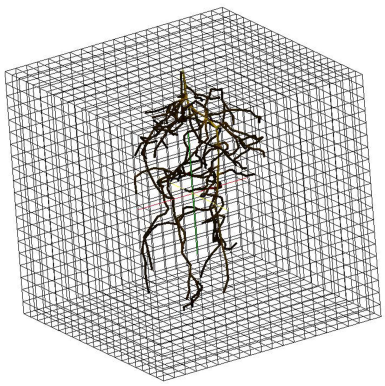

Figure 2: 3D soil grid merged with the 1D, branched, grid representing the root architecture

Grids can be created using different DUNE internal or external grid managers (see documentation of dune-grid). In the input file, the details about the numerical grids are specified in the groups [RootSystem.Grid] or [Soil.Grid]. Each folder contains a folder named “grids” where grids can be provided in dgf format. In the dumux-rosi examples, the soil grid is usually a structured grid created by the default “GridCreator”, where corner points of the domain, spatial resolution and cell type are specified such as in the following example:

```
[ Grid ]
LowerLeft = 0 0 0
UpperRight = 1 1 1
Cells = 10 10 20
CellType = Cube # or Simplex
```

Alternatively, msh-files can be read.

Choice of grid manager for the soil domain

YaspGrid: Structured grids, only non-periodic soil domains
SpGrid: Structured grids, also periodic soil domains
AluGrid: Unstructured grids, only works for `CC`$_2 pfa - scheme$
`UGGrid` $: Unstructured grids, works for both$

To change the grid manager, open the file `dumux-rosi/rosi_benchmarking/coupled_1p_richards/CMakeLists.txt`. If not available, add the following lines to make the gridmanager available to build an executable. For the example of UGGrid:

```
add_executable(coupledUG EXCLUDE_FROM_ALL coupled.cc)
target_compile_definitions(coupledUG PUBLIC DGF GRIDTYPE=Dune:: UGrid<3>)
```

Grids for root systems

There are two options to specify the root system grid. The first option is to specify it as a file in dgf-format that specifies the coordinates and connection of nodes (verteces).

```
DGF
Vertex
0 0 -0.03
-0.003301 -0.000687124 -0.0394144
-0.00339314 -0.00159054 -0.0473627
-0.00590116 -0.00546716 -0.0538958
-0.0115931 -0.00454388 -0.059441
-0.0105464 -0.00572357 -0.067284
-0.0100044 -0.0069212 -0.0751753
-0.00923561 -0.00814568 -0.0830435
-0.0100507 -0.00678732 -0.0908851
-0.010965 -0.00608665 -0.0962792
-0.00103059 0.000281757 -0.0413509
```

```
0.00284233 0.00172545 -0.0449409
0.00619859 0.00411514 -0.0466909
-6.51815e-05 0.00081926 -0.041154
0.000989275 0.00146981 -0.0404847
-0.00013962 0.00162442 -0.0411274
-0.00376417 0.00152097 -0.0477906
-0.00509243 0.00809897 -0.0472907
-0.00667036 0.0130109 -0.0453872
-0.00784569 0.0163412 -0.0446723
-0.00343402 0.00160078 -0.048867
-0.00272936 0.00153492 -0.0511603
-0.00257298 0.00150318 -0.0517729
-0.00605319 0.00128567 -0.051235
-0.00509341 0.00874088 -0.0479026
-0.0051018 0.00890818 -0.0480774
-0.0105648 -0.00490006 -0.0536398
-0.0155357 -0.00427108 -0.0535009
-0.0187161 -0.00393935 -0.0540514
-0.0129246 -0.00794987 -0.0599204
-0.0158011 -0.0137093 -0.0604436
-0.0159531 -0.0140359 -0.0604843
-0.0103269 -0.00178528 -0.069703
-0.0097869 0.000348939 -0.071026
-0.0101116 -0.00387942 -0.0769314
-0.00866863 -0.0079157 -0.0834016
#
SIMPLEX
parameters 10 # id0, id1, order, branchId, surf[cm2], length[cm], radius[
    cm], kz[cm4 hPa-1 d-1], kr[cm hPa-1 d-1], emergence time [d], subType,
     organType
0 1 0 1 0.314159 1 0.05 0 0 0.253641 1 2
1 2 0 1 0.251327 0.8 0.05 0 0 0.461984 1 2
2 3 0 1 0.251327 0.8 0.05 0 0 0.675409 1 2
3 4 0 1 0.251327 0.8 0.05 0 0 0.894171 1 2
4 5 0 1 0.251327 0.8 0.05 0 0 1.11854 1 2
5 6 0 1 0.251327 0.8 0.05 0 0 1.34882 1 2
6 7 0 1 0.251327 0.8 0.05 0 0 1.58532 1 2
7 8 0 1 0.251327 0.8 0.05 0 0 1.82839 1 2
8 9 0 1 0.17328 0.551569 0.05 0 0 2 1 2
1 10 1 2 0.0591391 0.313743 0.03 0 0 0.81929 2 2
10 11 1 2 0.103194 0.547463 0.03 0 0 1.36938 2 2
11 12 1 2 0.084377 0.447634 0.03 0 0 2 2 2
10 13 2 3 0.0141039 0.112236 0.02 0 0 1.88447 3 2
13 14 2 3 0.0176961 0.140821 0.02 0 0 2 3 2
13 15 3 4 0.0101666 0.080903 0.02 0 0 2 4 2
2 16 1 8 0.0596143 0.316264 0.03 0 0 0.976223 2 2
16 17 1 8 0.126845 0.672936 0.03 0 0 1.39819 2 2
17 18 1 8 0.103656 0.549912 0.03 0 0 1.75726 2 2
18 19 1 8 0.0679195 0.360324 0.03 0 0 2 2 2
16 20 2 9 0.0141835 0.112869 0.02 0 0 1.55953 3 2
```

```
20 21 2 9 0.0301593 0.24 0.02 0 0 1.81594 3 2
21 22 2 9 0.00795504 0.0633042 0.02 0 0 2 3 2
20 23 3 10 0.0445473 0.354496 0.02 0 0 2 4 2
17 24 2 12 0.0111449 0.088688 0.02 0 0 1.98626 3 2
24 25 2 12 0.00304236 0.0242104 0.02 0 0 2 3 2
3 26 1 16 0.088687 0.470499 0.03 0 0 1.35817 2 2
26 27 1 16 0.0944815 0.50124 0.03 0 0 1.74414 2 2
27 28 1 16 0.0611608 0.324468 0.03 0 0 2 2 2
4 29 1 19 0.0695225 0.368828 0.03 0 0 1.49515 2 2
29 30 1 19 0.121751 0.645908 0.03 0 0 1.97249 2 2
30 31 1 19 0.00683211 0.0362455 0.03 0 0 2 2 2
5 32 1 22 0.0872183 0.462707 0.03 0 0 1.79818 2 2
32 33 1 22 0.0484141 0.256845 0.03 0 0 2 2 2
6 34 1 24 0.0662369 0.351397 0.03 0 0 2 2 2
7 35 1 25 0.0133627 0.0708913 0.03 0 0 2 2 2
#
BOUNDARYDOMAIN
default 1
#
```

The paragraph ``SIMPLEX" specifies 10 parameters for each root segment: node1ID, node2ID, order, branchID, surfaceIdx in cm^2, length in cm , radiusIdx in cm, axialCondIdx cm^4 hPa^{-1} d^{-1}, radialCondIdx cm hPa^{-1} d^{-1}, emergenceTimeId[4] Order numbering starts with 0, i.e., primary roots have order 0. Potential artificial shoot segements have order -1.
Root systems in dgf format can be computed from measured root systems as well as with the root architecture module of CPlantBox.
The second option is to provide the root architectural parameters in the input file such that the root architecture and related grid is computed by CRootBox while used as a DuMu^x module.

```
[RootSystem.Grid]
File = Triticum_aestivum_a_Bingham_2011
InitialT = 10 # days
```

Important to know: It is currently necessary to build the code either for option 1 or for option 2 (i.e., two executables can be built that need to be provided with the correct input at runtime).

[4]Note: in the code we often use the term ´´creation time”, however, we always mean ´´emergence time”. Branch nodes exist twice, once in the mother branch, once as the starting node of the daughter branch. They have different emergence times but the same nodeID.

Numerical schemes

Numercical schemes available in DuMu^x

Box Method

CC$_2 pfaMethod$

How to switch numerical scheme in a DuMu^x simulation

Open the main file of your application. For coupled root-soil problems, it is for example the file `coupled.cc`.
Change the line that specifies the numerical scheme for the soil subproblem:
`using SoilTypeTag = Properties::TTag::RichardsBox;`
or `using SoilTypeTag = Properties::TTag::RichardsCC;`

Change the line that specifies the numerical scheme for the root subproblem:
`using RootTypeTag = Properties::TTag::RootsCCTpfa;`
or `using RootTypeTag = Properties::TTag::RootsBox;`

Make sure that the relevant properties file is given (line 62 in coupled.cc).